Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт Кибернетики
Направление подготовки Прикладная информатика
Кафедра Оптимизации систем управления

БАКАЛАВРСКАЯ РАБОТА

Тема работы
Разработка информационной системы спортклуба

УДК _004.415:796.062_

Студенты

Группа	ФИО	Подпись	Дата
8К21	Былина Татьяна Андреевна		
8К21	Куликова Марина Олеговна		
8К21	Шин Марина Витальевна		

Руководитель

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Старший преподаватель каф. ОСУ	Мокина Елена Евгеньевна	-		

КОНСУЛЬТАНТЫ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент каф. МЕН	Баннова Кристина Алексеевна	к.э.н.		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент каф. ЭБЖ	Мезенцева Ирина Леонидовна	-		

ДОПУСТИТЬ К ЗАЩИТЕ:

Зав. кафедрой	ФИО	Ученая степень, звание	Подпись	Дата
ОСУ	Иванов Максим Анатольевич	к.т.н.		

Томск – 2016 г.

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт Кибернетики
Направление Прикладная информатика
Кафедра Оптимизации систем управления (ОСУ)

УТВЕРЖДАЮ:
Зав. кафедрой ОСУ
_____ _____ М.А. Иванов

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

Бакалаврской работы (в составе команды)

(бакалаврской работы, дипломного проекта/работы, магистерской диссертации)

Студенту:

Группа	ФИО
8К21	Былиной Татьяне Андреевне
	Куликовой Марине Олеговне
	Шин Марине Витальевне

Тема работы:

Разработка информационной системы спортклуба	
Утверждена приказом директора (дата, номер)	

Срок сдачи студентом выполненной работы:	

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе	Информационная система разработана для автоматизации спортклуба. В программе два автоматизированных рабочих места: для администратора и тренера, позволяющий выполнять ряд стандартных и специальных задач, связанных с деятельностью спортклуба. Исходными данными к работе является техническое задание на разработку программного продукта.
Перечень подлежащих исследованию, проектированию и разработке вопросов	1. Описание предметной области. 2. Обзор существующих программных продуктов для автоматизации работы спортклуба. 3. Определение функционала и требований к системе. 4. Проектирование автоматизированных

	рабочих мест администратор и тренера. 5. Разработка автоматизированных рабочих мест администратора и тренера. 6. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение. 7. Социальная ответственность.
Перечень графического материала	Диаграммы методологий idef0, dfd, uml, erd. Изображения разработанного интерфейса.

Консультанты по разделам выпускной квалификационной работы

Раздел	**Консультант**
Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	Баннова Кристина Алексеевна
Социальная ответственность	Мезенцева Ирина Леонидовна

Названия разделов, которые должны быть написаны на русском и иностранном языках:

1 Обзор предметной области. Определение требований к системе

2 Проектирование и разработка информационной системы

3 Руководство пользователя информационной системы

4 Финансовый менеджмент, ресурсоэффективность и ресурсосбережение

5 Социальная ответственность

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	

Задание выдал руководитель:

Должность	**ФИО**	**Ученая степень, звание**	**Подпись**	**Дата**
Старший преподаватель	Мокина Елена Евгеньевна			

Задание принял к исполнению студент:

Группа	**ФИО**	**Подпись**	**Дата**
8К21	Былина Татьяна Андреевна,		
	Куликова Марина Олеговна,		
	Шин Марина Витальевна		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студенту:

Группа	ФИО
8К21	Былиной Татьяне Андреевне
	Куликовой Марине Олеговне
	Шин Марине Витальевне

Институт	Кибернетики	**Кафедра**	Оптимизации систем управления
Уровень образования	Бакалавриат	**Направление/специальность**	Прикладная информатика

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
1. Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих	Информационная система разработана для автоматизации спортклуба. В программе есть два автоматизированных рабочих места: для администратора и тренера, позволяющий выполнять ряд стандартных и специальных задач, связанных с деятельностью спортклуба.
2. Нормы и нормативы расходования ресурсов	
3. Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования	
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Оценка коммерческого и инновационного потенциала НТИ	Оценка конкурентоспособности разработки, анализ перспективности проекта
2. Планирование и формирование бюджета научных исследований	Планирование этапов разработки программы, определение трудоемкости, построение диаграммы Ганта, формирование бюджета НТИ.
3. Определение ресурсной, финансовой, экономической эффективности	Сравнительный анализ интегральных показателей эффективности
Перечень графического материала	
1. Сегментирование рынка *2.* Оценка конкурентоспособности технических решений *3.* Оценочная карта QuaD, Матрица SWOT *4.* График проведения и бюджет НТИ *5.* Оценка ресурсной, финансовой и экономической эффективности НТИ	

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент каф. МЕН	Баннова Кристина Алексеевна	к.э.н.		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8К21	Былиной Татьяне Андреевне		
	Куликовой Марине Олеговне		
	Шин Марине Витальевне		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студенту:

Группа	ФИО
8К21	Былиной Татьяне Андреевне
	Куликовой Марине Олеговне
	Шин Марине Витальевне

Институт	Кибернетики	**Кафедра**	Оптимизации систем управления
Уровень образования	Бакалавриат	**Направление/специальность**	Прикладная информатика

Исходные данные к разделу «Социальная ответственность»:	
1. Характеристика объекта исследования и области его применения	Объект исследования – информационная система по автоматизации работы спортивного клуба.
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Производственная безопасность 1.1. Анализ выявленных вредных факторов при разработке и эксплуатации проектируемого решения в следующей последовательности: 1.2. Анализ выявленных опасных факторов при разработке и эксплуатации проектируемого решения в следующей последовательности:	**1. Производственная безопасность** 1.1. Анализ выявленных вредных факторов при разработке и эксплуатации проектируемого решения в следующей последовательности: – повышенный уровень электромагнитных излучений; – повышенная или пониженная влажность воздуха; – повышенная или пониженная температура воздуха; – повышенный уровень шума; – недостаточная освещенность рабочего места; – эмоциональные перегрузки; – умственное перенапряжение; – монотонность труда. 1.2. Анализ выявленных опасных факторов проектируемого решения в следующей последовательности: – опасность поражения электрическим током.
2. Экологическая безопасность	**2. Экологическая безопасность** – анализ воздействия объекта на литосферу (отходы, связанные с утилизацией вышедшего из строя ПК, люминесцентных ламп и др.); – разработка решению по обеспечению экологической безопасности.

3. Безопасность в чрезвычайных ситуациях	**3. Безопасность в чрезвычайных ситуациях** – Перечень возможных ЧС при разработке и эксплуатации проектируемого решения; – выбор наиболее типичной ЧС (пожар); – разработка превентивных мер по предупреждению пожара; – разработка действий в результате пожара и мер по ликвидации последствий.
4. Правовые и организационные вопросы обеспечения безопасности	**4. Правовые и организационные вопросы обеспечения безопасности:** – специальные правовые нормы трудового законодательства при работе с компьютером и орг. техникой; – требования к организации рабочих мест пользователей.

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент кафедры ЭБЖ	Мезенцева Ирина Леонидовна			

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8К21	Былина Татьяна Андреевна		
	Куликова Марина Олеговна		
	Шин Марина Витальевна		

Министерство образования и науки Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт _____Кибернетики_____
Направление подготовки (специальность)_____Прикладная информатика_____
Уровень образования _____Бакалавриат_____
Кафедра _____Оптимизации систем управления_____
Период выполнения _____(осенний / весенний семестр 2015/2016 учебного года)_____

Форма представления работы:

бакалаврская работа

(бакалаврская работа, дипломный проект/работа, магистерская диссертация)

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Срок сдачи студентом выполненной работы:	15 июня 2016 г.

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
23.10.2015	*1 Обзор предметной области. Определение требований к системе*	
16.02.2016	*2 Проектирование и разработка информационной системы*	
10.03.2016	*3 Руководство пользователя информационной системы*	
24.04.2016	*4 Финансовый менеджмент, ресурсоэффективность и ресурсосбережение*	
24.05.2016	*5 Социальная ответственность*	

Составил преподаватель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Старший преподаватель	Мокина Елена Евгеньевна			

СОГЛАСОВАНО:

Зав. кафедрой	ФИО	Ученая степень, звание	Подпись	Дата
ОСУ	Иванов Максим Анатольевич	к.т.н.		

Реферат

Выпускная квалификационная работа включает в себя 203 страницы, 57 рисунков, 20 таблиц, 27 источников, 1 приложение.

Ключевые слова: спортклуб, программное обеспечение, программа, информационная система, приложение, управление, автоматизация, проектирование, анализ, бизнес-процессы, разработка, 1С, программирование.

Объектом исследования является деятельность спортивного клуба по подготовке высококвалифицированных спортсменов. Объектами разработки являются автоматизированные рабочие места персонала спортклуба: администратора и тренера.

Цель работы спроектировать и разработать автоматизированные рабочие места в информационной системе, которая автоматизирует процессы в спортклубе.

В процессе разработки проводились анализ предметной области, анализ конкурентных решений, определение требований к системе, а также анализ бизнес-процессов.

В результате разработки была спроектирована и реализована информационная система для спортивного клуба.

Степень внедрения: внедрена.

Область применения: спортивные клубы и спортивные ассоциации.

Содержание

Введение

В настоящее время спортивные клубы, спортивные ассоциации и другие организации, занимающиеся подготовкой и переподготовкой спортсменов, становятся все более популярными. В связи с этим возникают определенные проблемы по решению специфических задач.

Целью выпускной квалификационной работы является создание информационной системы для автоматизации работы спортклубов, позволяющей решать различные задачи данной предметной области.

Для достижения поставленной цели были определены задачи:

- сделать обзор предметной области;
- рассмотреть существующие аналоги;
- изучить процессы, протекающие в системе;
- выявить основную проблему и определить факторы, оказывающие на неё влияние;
- выявить основные роли пользователей ИС и описать их функции;
- построить диаграммы потоков данных ИС;
- построить модель ИС.

Объектом исследования является деятельность спортивных клубов и других подобных организаций, включающая в себя различные процессы, начиная от первого посещения спортсменом клуба, продажи ему абонемента и так далее до каждого занятия тренировочного процесса.

Предметом исследования является разрабатываемая информационная система для работы спортклуба.

В целом работа направлена на создание информационной системы, позволяющей решать выявленные проблемы. Данная система не имеет аналогов и уже успешно внедрена в организации-заказчике.

Разработка информационной системы производится на платформе 1С с использованием типовой конфигурации 1С: Управление небольшой фирмой.

1 Описание предметной области. Определение требований к системе

1.1 Описание процессов предметной области

Спортивный клуб – это общественная, либо частная профессиональная организация, которая объединяет спортсменов и иногда любителей спорта. Различают специализированные клубы (по видам спорта) и спортклубы общего типа. Все профессиональные спортклубы имеют свою инфраструктуру, систему управления, базу спортсменов с их личными достижениями, а также сформированные команды для соревнований.

Задачами спортклуба являются:

1. Вовлечение спортсменов в систематические занятия спортом, привлечение их к спортивной жизни и создание мотивации и интереса к спортивным победам.
2. Создание благоприятных условий для тренировочных процессов, а также эффективной организации работы со спортсменами.
3. Обеспечение участия спортсменов в различных соревнованиях.
4. Непрерывное отслеживание прогресса каждого из спортсменов.

Спортклуб осуществляет следующие виды деятельности:

1. Организация мероприятий, которые направлены на популяризацию спорта, включая лекции и соревнования.
2. Организация и контроль за проведением тренировочного процесса в области детского и юношеского спорта.
3. Регистрация и учет спортивных достижений спортсменов.
4. Формирование и поддержка деятельности спортивных команд.
5. Ведение учета проведенных занятий у тренеров.

Услуги предоставляются спортсменам на платной основе в форме спортивного абонемента согласно утвержденному расписанию занятий группы.

Спортивный *абонемент* может включать в себя различные услуги, либо спортсмен может приобрести несколько спортивных абонементов для получения нескольких видов услуг. Спортсмены в группах, в зависимости от

практики клуба, могут приостанавливать действие своих абонементов. На усмотрение спортивного клуба, количество приостановок клуба может быть конечным.

Помимо групповых занятий в спортклубе могут проводиться и индивидуальные, а также существует тренировочный режим, когда спортсмен тренируется в свободное время.

Расписание занятий для групп в спортклубе создается и утверждается администратором.

Спортивный клуб может арендовать или иметь в собственности несколько тренировочных помещений в населенном пункте или ряде населенных пунктов. Спортивный клуб может принадлежать спортивной ассоциации и принимать участие в спортивных мероприятиях различного уровня. В спортивных клубах практикуется прием оплат после посещения клуба – в частности после посещения клуба детьми, оплату их тренировок осуществляет родитель.

Организационная структура спортклуба выглядит следующим образом (рис.1). Во главе спортклуба находится Директор. Так как спортклуб имеет n филиалов, в каждом филиале свой Менеджер, который занимается рекламой и персоналом, у каждого Менеджера свои подчиненные – Администратор филиала спортклуба, Бухгалтер, Обслуживающий персонал и Медицинский работник. В подчинении у Директора клуба находится Главный тренер, который, в свою очередь, подчиняет m тренеров всего спортклуба. Тренеры не привязаны к Менеджерам, так как могут тренировать в любом из филиалов.

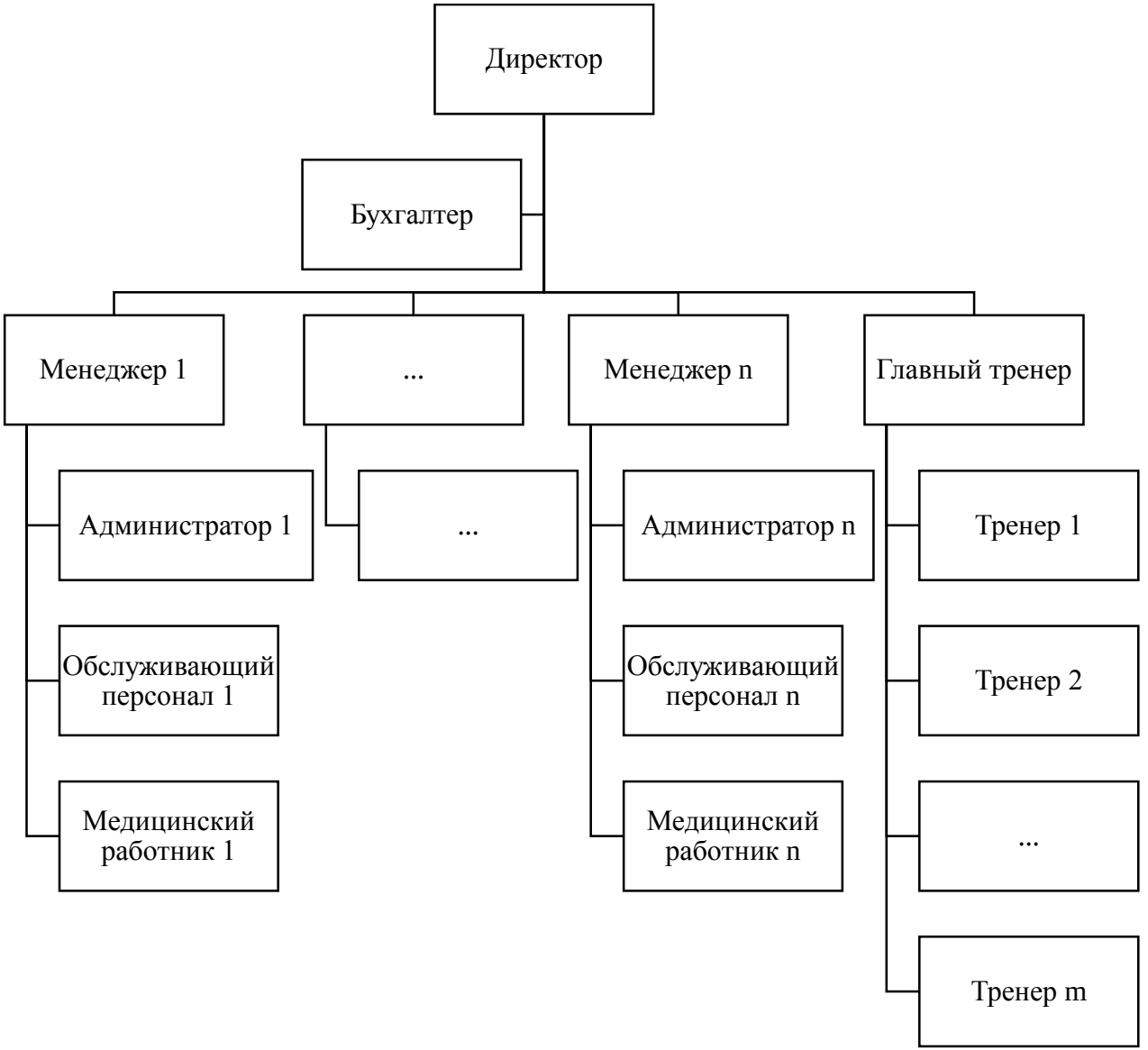


Рис. 1 Организационная структура спортклуба

Основные *роли* в организации деятельности спортклуба:

- администратор спортклуба;
- тренер.

Администратор спортклуба – персонал, который работает со спортсменами вне тренировочного процесса, обеспечивает работу по обслуживанию спортсменов и созданию для них комфортных условий.

Основные функции администратора спортклуба:

- продажа, замораживание, продление абонементов;
- ведение клиентской базы;
- регистрация посещений клуба спортсменами;
- выдача ключей от кабинок;
- запись на занятие, включая запись по телефонному звонку;
- формирование расписаний занятий;
- утверждение графика рассрочки оплат.

Тренер – персонал, который работает со спортсменами во время тренировочного процесса, обеспечивает спортсмена необходимыми спортивными нагрузками и мотивирует к новым достижениям.

Основные функции тренера:

- сообщение спортсмену своих знаний и умений, стимулирование его познавательной активности;
- разносторонняя подготовка спортсмена, а также формирование волевой готовности спортсмена к выступлению на различных соревнованиях;
- оптимизация тренировочного процесса и его совершенствование в условиях регулярного контроля над спортсменами;
- выработка у спортсменов морально-волевых качеств, определенных черт характера и самодисциплины;
- оценка результатов спортсменов, показанные на тренировках и соревнованиях.

1.1.1 Функциональное моделирование предметной области IDEF0

Спортклуб работает в соответствии с законами Российской Федерации и положениями, разработанными руководством спортклуба. Эти ограничения действуют на протяжении всей его деятельности. На входе процесса *Организация деятельности спортклуба – Запрос спортсмена*. Это означает, что вся его деятельность связана с работой со спортсменами. На выходе процесса соответственно *Абонемент* – то, что получает спортсмен при запросе. Механизм, которым управляется процесс: *Персонал* и О*борудование*.

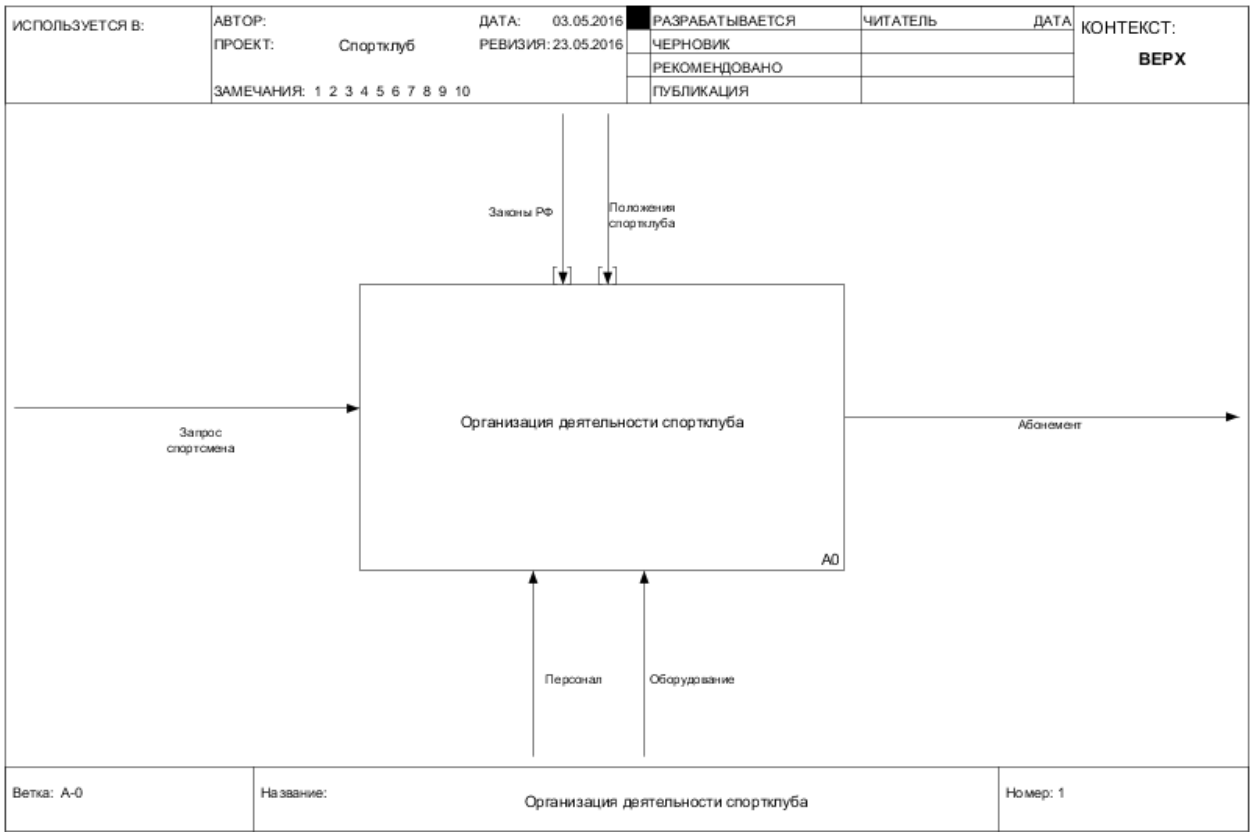


Рис. 2 Контекстная диаграмма IDEF0

Декомпозиция диаграммы разбивает процесс *Организация деятельности спортклуба* на подпроцессы, которые отражают его подробное описание (рис.3).

Подпроцесс *Составление расписания* (A1) имеет на входе два объекта: *График работ тренеров* и *Виды занятий*, на выходе – *Расписание*, в соответствии с которым выполняется третий и четвертый подпроцессы: *Запись на тренировки* и *Проведение занятий*. Ограничением является *Режим работы* спортклуба. Составлением расписания занимается *Администратор*.

Подпроцесс *Продажа абонемента* (A2) имеет на входе объекты *Деньги* и *Запрос спортсмена* на продажу абонемента. На выходе – *Абонемент* и *Чек оплаты*. Продажа осуществляется *Администратором* в соответствии с *Прайс-листом* спортклуба.

Подпроцесс *Запись на тренировки* (A3) на входе имеет *Запрос спортсмена*, на выходе – *Список групп*, которые посетят занятие по расписанию. Запись на тренировки осуществляется *Администратором*.

Подпроцесс *Проведение занятий* (A4) на входе имеет объект *Спортсмен*, на выходе – *Спортсмен'*, который уже потренировался в спортклубе. Занятия проводятся *Тренером*, присутствуют те спортсмены, которые заявлены в *Списке групп*.

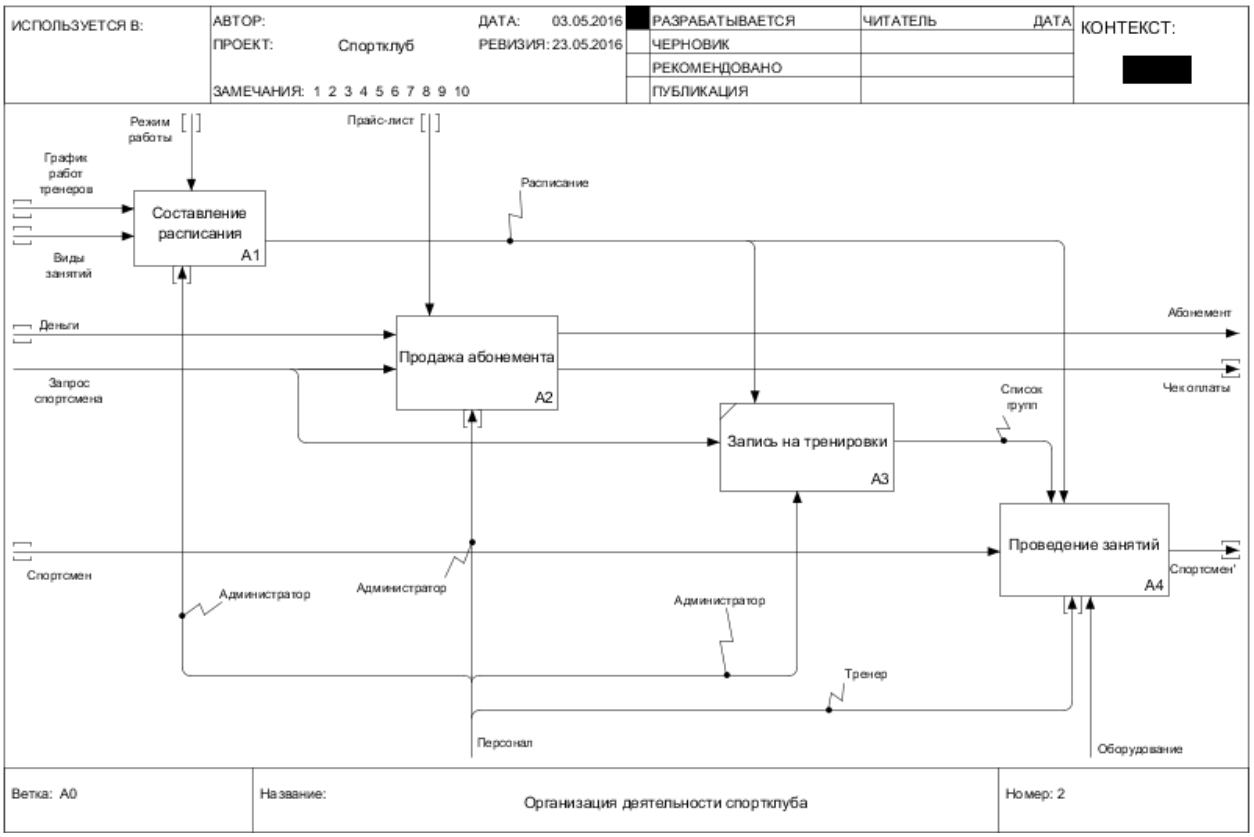


Рис. 3 Декомпозиция контекстной диаграммы IDEF0

1. Декомпозиция подпроцесса *Составление расписания* (рис.4):

Подпроцесс *Обсуждение графика работ с каждым тренером* (A11) имеет на входе *График работ тренеров* и *Виды занятий*. На выходе – *Предварительный график занятий*.

Подпроцесс *Распределение помещений для тренировок* (A12) следует за процессом *Обсуждение графика работ с каждым тренером* и имеет на входе *Предварительный график занятий.* Соответственно на выходе – *Черновик расписания*.

Последним подпроцессом является *Ликвидация наложений в графике* (A13), который на входе имеет *Черновик расписания*. Ограничением для данного подпроцесса является *Режим работы* спортклуба.

Все подпроцессы выполняются *Администратором* клуба с соответствии с его *Положениями*.

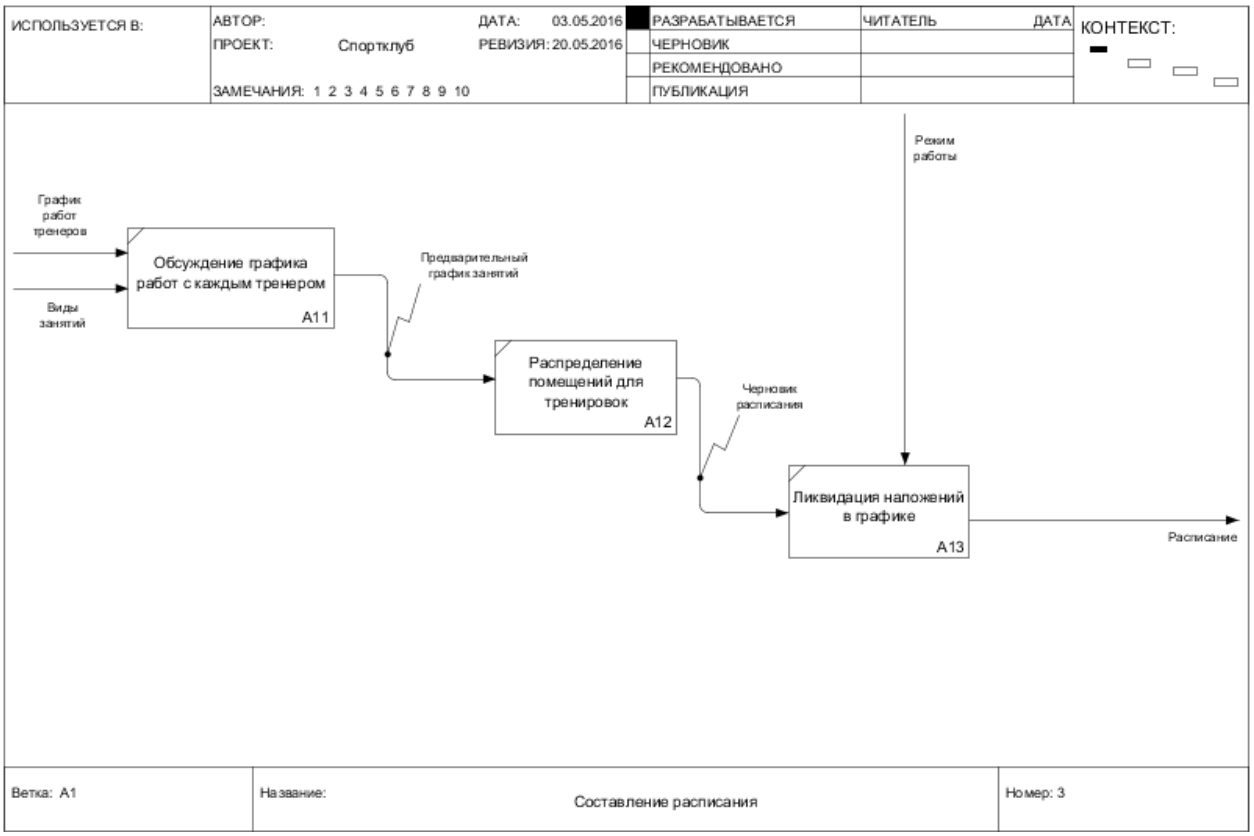


Рис. 4 Декомпозиция подпроцесса *Составление расписания*

2. Декомпозиция подпроцесса *Продажа абонемента* (рис.5):

Первый подпроцесс *Обсуждение вопросов* (A21) на входе имеет *Запрос спортсмена*, на выходе соответственно – *Запрошенный вид абонемента*.

Второй подпроцесс *Заполнение абонемента* (A22) следует за процессом *Обсуждение вопросов* и имеет на входе *Запрошенный вид абонемента*, на выходе – *Абонемент*, который является выходом всего

процесса, и *Данные абонемента*, которые являются входом для третьего процесса.

Третий подпроцесс *Запись о проданном абонемента* (A23) имеет на входе *Данные абонемента*, на выходе – *Квитанция*.

Последний подпроцесс *Прием оплаты* (A24) на входе имеет *Квитанцию*, на выходе – *Чек оплаты*. Ограничением выполнения подпроцесса является *Прайс-лист*.

Все подпроцессы выполняются *Администратором* клуба с соответствии с его *Положениями*.

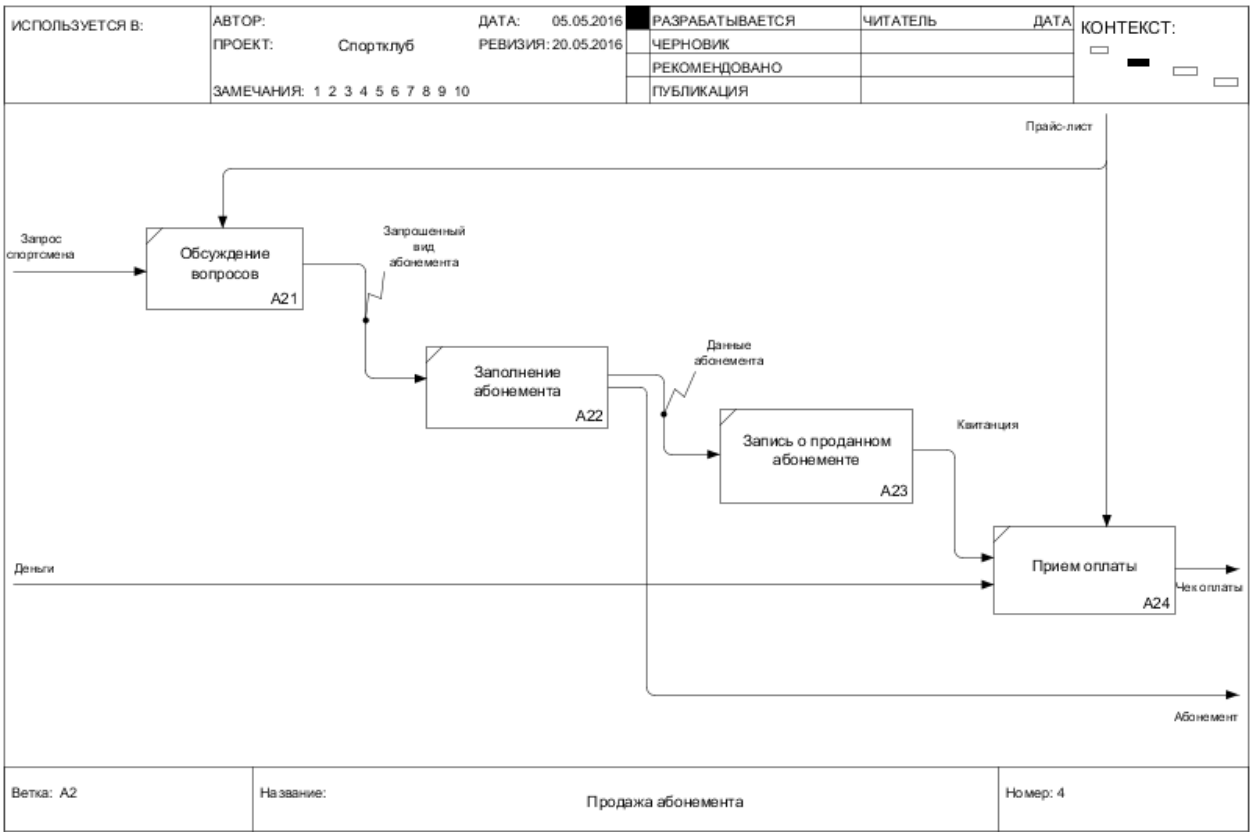


Рис. 5 Декомпозиция подпроцесса *Продажа абонемента*

3. Декомпозиция подпроцесса *Проведение занятий* (рис.6):

Первый подпроцесс *Запись о присутствующем спортсмене* (A41) на входе имеет *Данные о спортсмене*, на выходе соответственно – *Количество присутствующих спортсменов*.

Второй подпроцесс *Выдача инвентаря* (A42) следует за процессом *Запись о присутствующем спортсмене* и имеет на входе *Запрос на выдачу инвентаря*, на выходе – требуемый *Инвентарь*, который является входом

третьего подпроцесса. Ограничением подпроцесса служит *Количество присутствующих спортсменов*.

Третий подпроцесс *Тренировочный процесс* (A43) имеет на входе два объекта: *Инвентарь* и *Спортсмен*, на выходе – *Инвентарь'* и *Спортсмен'*. Ограничениями подпроцесса являются *Список групп* и *Расписание*. А дополнительным механизмом управления подпроцессом является *Оборудование* (т.е., тренажеры).

Последний подпроцесс *Сбор инвентаря* (A44) на входе имеет *Инвентарь'*, на выходе – *Отчет о сдаче инвентаря*.

Все подпроцессы выполняются *Тренером* с соответствии с *Положениями спортклуба*.

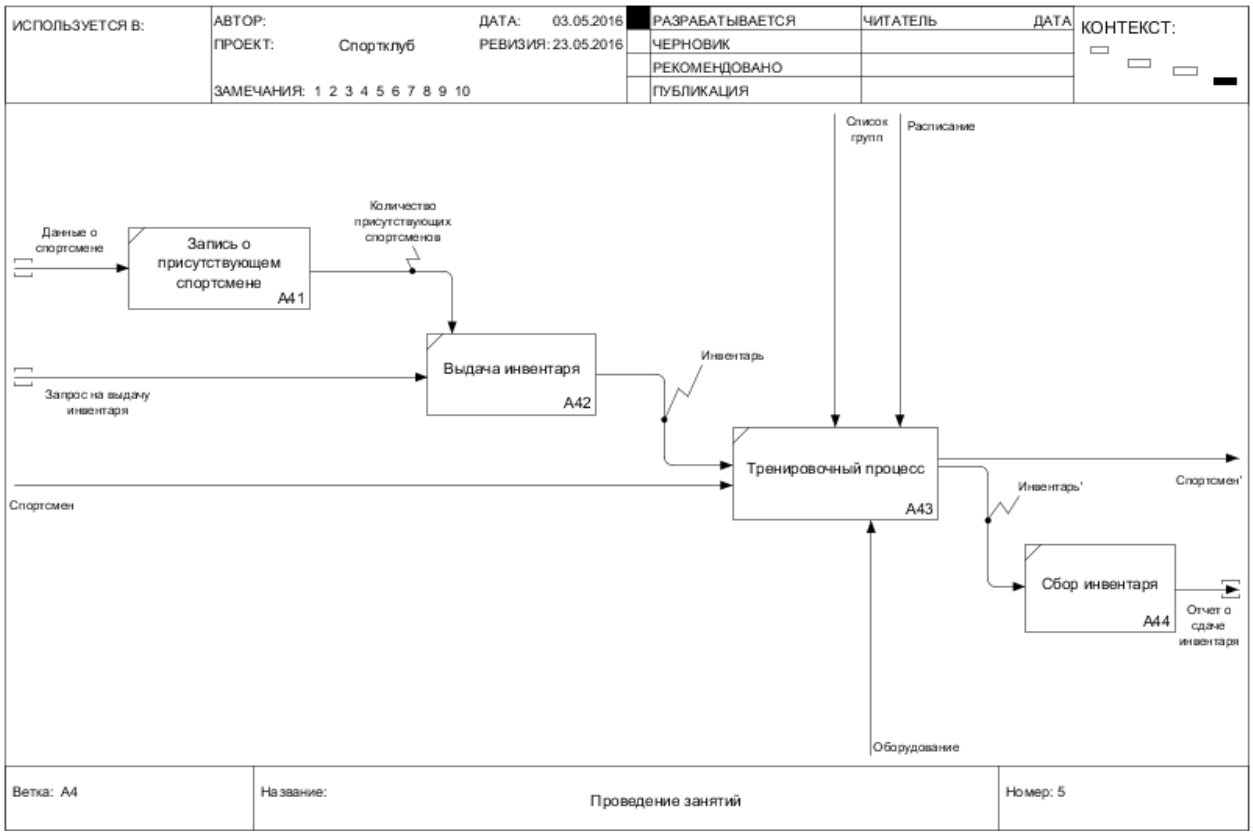


Рис. 6 Декомпозиция подпроцесса *Проведение занятий*

1.1.2 Причинно-следственная диаграмма Исикавы

Для наглядного отображения фактических проблем, повлекших за собой необходимость создания системы, была использована причинно-следственная диаграмма (рис.7).

Рис. 7 Причинно-следственная диаграмма Исикавы

Основной проблемой является некачественное обслуживание клиентов. Во-первых, причиной этому служат сложности в составлении расписания: часто происходят наложения (одновременное использование одного и того же помещения/инвентаря, либо назначение одного и того же тренера); трудно редактировать уже составленное расписание, учитывая все вышеперечисленное, а также равномерно распределять нагрузку на персонал.

Во-вторых, этому способствует медленное обслуживание клиентов: часто возникают очереди из-за того, что персонал не справляется с растущим количеством клиентов, это подкрепляется необходимостью дублировать информацию о посещениях и покупках абонемента как на самом абонементе, так и в специальных учетных документах. Поиск необходимой информации для обслуживания клиента также является трудоемким и требует значительных временных ресурсов.

Также возникают сложности с обработкой больших объемов информации. Документы теряются, путаются и достаточно сложно отследить

состояние каждого из них, как, например, необходимость продления медицинского допуска спортсмена к занятиям. А также представляется трудоемким процесс обработки и обобщения накопленной информации по нужному направлению (например, узнать статистику помещений, проведенных занятий и т.д.).

Еще одной причиной является потеря инвентаря, трудно отследить виновника и ответственного за это администратора, вследствие чего возникает нехватка инвентаря.

Таким образом, существующая система призвана решить вышеперечисленные проблемы и способствовать повышению качества обслуживания в спортивном клубе.

1.2 Обзор существующих решений

На сегодняшний день электронно-вычислительные машины являются неотъемлемой частью жизни общества, и уже сложно представить функционирование какой-либо организации без использования информационных систем и автоматизации части либо всех функциональных процессов. Среди особо трудоемких можно выделить задачи ведения бухгалтерского и налогового учета, консолидации, работы с кадрами, учета платежей, документооборота и т.д. Однако на данном этапе для удовлетворения потребностей современных организаций уже недостаточно автоматизации только ряда функциональных блоков. Необходим комплексный продукт, учитывающий специфику конкретного вида деятельности.

Спортивные клубы и другие организации, осуществляющие физкультурно-оздоровительную деятельность, в последнее время становятся все более популярными. В связи с этим вырос спрос на информационные системы, позволяющие решать не только типовые, но и специфические задачи.

На рынке представлен ряд программных продуктов, призванных выполнять учет задач, характерных для данного вида деятельности.

Для обзора выбрано несколько информационных систем, которые проанализированы по ряду критериев: среда разработки, функциональность системы, стоимость использования, а также выявлены достоинства и недостатки каждой из систем.

Для анализа были выбраны следующие информационные системы: «UNIVERSE – Фитнес», «1С: Фитнес клуб», «Абонемент», «КРАФТ ERP Фитнес-клуб», «Фитнес – клуб».

Все пять программных продуктов позиционируются как программное обеспечение для автоматизации деятельности спортивных клубов, фитнес – центров, тренажерных залов и т.п.

«UNIVERSE – Фитнес». Информационная система, разработанная «UNVERSE – Soft» и призванная обеспечить автоматизированное управление всех процессов, происходящих в фитнес-центре.

Функциональность. Программный продукт состоит из нескольких функциональных блоков.

CRM модуль для отдела продаж. Обеспечивает ведение клиентской базы, сохраняет историю общения с клиентом, включая звонки, электронные письма и т.д., выводит напоминания, регистрирует выполнение задач.

Автоматизация работы рецепции. Обеспечивает регистрацию посещений, управление картами клиентов, электронную запись на занятия, позволяет получать оперативный доступ к основной информации по клиенту.

Управление работой персонала. Включает в себя систему учета контроля рабочего времени, планирование групповых и персональных занятий, автоматический расчет заработной платы.

Учет товаров и ведение склада.

Касса и финансы. Обеспечивает ведение кассы, предоставляет общие финансовые расходы.

Стоимость. Стоимость программного продукта зависит от выбранной конфигурации: локальная конфигурация на 1 компьютер стоимостью 80000 рублей, а также сетевые конфигурации на 2/3/4/5 компьютеров стоимостью с 105000/130000/155000/180000 рублей соответственно и возможностью покупки дополнительных рабочих мест стоимостью 25000 рублей каждое. Также в стоимость контракта входит бесплатная установка, обучение в г. Москва и техническая поддержка в течение одного года, включающая обновления и консультации по электронной почте и телефону.

Достоинства и недостатки. Система отличается довольно широким функционалом. Однако основной упор сделан на работу с клиентами, в то время как в складском и кассовом учете реализован лишь базовый функционал.

Также ценовая политика слишком высокая для среднестатистического потребителя.

«1С: ФИТНЕС КЛУБ». Отраслевое решение, разработанное «Helix-Group» на платформе «1С:Предприятие», предназначенное для автоматизации спортивных клубов и других организаций подобной направленности, сферой деятельности которых является предоставление клиентам как возможности занятия спортом, так и услуг различного рода: солярий, массаж, парикмахерская и т.д.

Функциональность. Данное решение имеет широкий функционал, разработчики выделяют пять основных групп функций.

Работа с клиентами. Позволяет формировать клиентскую базу, и при этом хранить разнообразную информацию о клиенте: начиная с персональных данных и лицевых счетов клиента и заканчивая целями посещения, предпочтениями в напитках и принадлежности к определенному сегменту.

Учет финансов. Данный раздел учитывает все манипуляции с денежными средствами, которые проводятся при работе с клиентами.

Управление персоналом. Основные возможности: хранение данных о сотрудниках, формирование рабочего графика, учет фактически отработанного ими времени, расчет заработной платы, проведение взаиморасчетов с сотрудниками.

Учет запасов на складе. Эта группа функций, представляет автоматизацию взаимодействия с поставщиками, контроль закупок и инвентаризацию.

Проведение маркетинговых мероприятий.

Стоимость. Конечная стоимость складывается из клиентских лицензий на программный продукт и платформу 1С:Предприятие. Таким образом, стоимость 1/5/10/20 рабочих мест составляет 25300/84600/162400/300800 рублей, лицензия на сервер – 50400 рублей. В стоимость программы входит бесплатная доставка и установка, курс обучения по работе с программой, методические материалы и бесплатное гарантийное обслуживание в течение трех месяцев.

Достоинства и недостатки. Следует отметить, что данный программный продукт не предусматривает ведения бухгалтерского и налогового учета. Для реализации этих возможностей необходимо провести интеграцию приложения с «1С: БУХГАЛТЕРИЯ 8», что требует дополнительных ресурсов. Также к недостаткам можно отнести короткий срок бесплатного гарантийного обслуживания по сравнению с другими программными продуктами.

«Абонемент». Программное обеспечение, разработанное компанией «UCS», для автоматизации фитнес-клуба.

Функциональность. Программный продукт «Абонемент» имеет следующие функциональные возможности:

- возможность бронирования тренировок;
- регистрация клиентов и хранение информации о них;
- планирование заданий (расписание индивидуальных и групповых заданий);

- возможность формировать расписание посещений для гостя с учетом его пожеланий;
- ведение статистики посещений;
- возможность e-mail и sms рассылок;
- разработанная система отчетов для руководства.

Стоимость. Стоимость данного программного обеспечения составляет 120000 рублей. При этом в стоимость не включена поддержка клиента, для этого необходимо дополнительное заключение договора на абонентское обслуживание.

Достоинства и недостатки. Информационная система «Абонемент» позволяет автоматизировать в основном учет клиентов и манипуляций с абонементами. Это локальная программа, не подходящая для сети филиалов. К тому же, еще одним недостатком является отсутствие складского учета, а также возможности ведения финансовой, бухгалтерской отчетности без интеграции с другим программным обеспечением.

«КРАФТ Фитнес-клуб». Программный комплекс для фитнес-клубов, разработанный «Бином Софт», позволяющий автоматизировать деятельность не только фитнес-клубов, но и SPA-салонов и других предприятий, выполняющих услуги для клиентов на клубной основе.

Функциональность. Основными функциями программы являются следующие:

- ведение персональной клиентской базы;
- ведение прейскуранта по видам карт и дополнительным услугам;
- работа с договорами клиентов (включая корпоративных клиентов);
- ведение персональных контрактов с индивидуальными условиями;
- ведение баланса клиентов раздельно по договорам, по бару, по дополнительным услугам;
- кассовые операции (оплата договоров и услуг);
- регистрация клиентов на рецепции по карте;

- автоматизация работы бара, включая регистрацию продаж и управление складом;
- аренда и резервирование ресурсов;
- система мониторинга и отчетности.

Достоинства и недостатки. Данное программное обеспечение имеет довольно широкие функциональные возможности по работе с клиентами. Однако в системе не реализована возможность ведения бухгалтерского и налогового учетов. Но интеграция с другими приложениями разработанными организацией «Бином Софт» позволит после покупки дополнительного модуля осуществлять комплексный учет в организации.

«Фитнес – клуб». Система, разработанная организацией ООО «Простой софт». Главная ее задача – создать полноценный учет посещаемости клуба, сформировав при этом базу данных клиентов.

Функциональность. Основные функции, представленные в информационной системе:

- учет клиентов и сбор их контактной информации;
- учет работы всех филиалов сети;
- учет частоты посещений;
- учет числа абонементов, клубных карт и разовых посещений;
- идентификация посетителя по клубной карте;
- учет оказанных услуг;
- создание расписаний групповых занятий в разных залах;
- контроль оплаты;
- создание отчетов.

Стоимость. Стоимость на программный продукт зависит от выбранной конфигурации. Стоимость локальной конфигурации на одно рабочее место составляет 3000, за сетевую конфигурацию на 3/5/10/20 рабочих мест придется заплатить 6000/10000/18000/25000 рублей

соответственно. В стоимость входит подписка на обновления и стандартная поддержка сроком на 1 год.

Достоинства и недостатки. Позволяет осуществлять учет клиентов, абонементов и манипуляций с ними, но набор функций значительно меньше набора функций предыдущих систем. Данный программный продукт следуют использовать в организациях с малой пропускной способностью или в организациях, для которых не критично приостановление на неопределенный срок функционирования системы.

Рассмотренные информационные системы являются отличным примером предложений по автоматизации деятельности фитнес – центров, поскольку можно продемонстрировать набор функций, который удовлетворит потребности организаций разных размеров. «UNIVERSE – Фитнес» следует внедрять в крупную организацию или активно развивающуюся организацию средних размеров. «1С: ФИТНЕС КЛУБ» больше подходит для организаций средних размеров, однако и малые и крупные организации внедряют это программное обеспечение. «КРАФТ Фитнес-клуб» и «Абонемент» представляют собой средний класс программ, который не целесообразно внедрять в крупные предприятия, поскольку сложно организовать эффективный комплексный учет. Программный продукт «Фитнес – клуб» в наибольшей степени подходит малым организациям, поскольку программа имеет небольшой набор функций, легко внедряется и ее стоимость не велика.

В целом на основе проведенного анализа можно сделать вывод, что автоматизация специализированных процессов является перспективным направлением развития информационных технологий, что в свою очередь будет способствовать повышению качества предоставляемых услуг.

В результате обзора ряда программных продуктов, существующих в данный момент на рынке, были выявлены как достоинства, так и недостатки каждого. И, несмотря на то, что некоторые из них имеют широкий функционал, возможности складского и бухгалтерского учетов, а также

доступную ценовую политику, характеристики данных программных продуктов не совпадают с заявленными требованиями.

Во-первых, необходимо учитывать, что спортивный клуб, для которого разрабатывается продукт, принадлежит спортивной ассоциации и принимает участие в мероприятиях различного уровня, для участия в которых спортсмены должны получать аккредитации. Таким образом, необходимо отметить, что в клубе могут заниматься как спортсмены профессионалы, так и любители. Поэтому система должна учитывать аккредитации, выдаваемые на определенный срок, спортивные разряды, квалификации, медицинскую страховку и другие параметры, необходимые при работе с профессиональными спортсменами.

Во-вторых, рассматриваемый спортивный клуб может арендовать или иметь в собственности несколько помещений в одном или разных населенных пунктах, что также необходимо учитывать при разработке системы.

В рассматриваемом клубе могут заниматься как взрослые, так и дети, поэтому система должна это учитывать при регистрации клиента и в случае посещения клуба ребенком, хранить информацию о его родителях/опекунах и т.д.

Таким образом, ряд вышеперечисленных и других тонкостей в работе рассматриваемого спортивного клуба не позволяет воспользоваться каким-либо существующим решением и требует создания нового комплексного продукта, учитывающего все особенности его деятельности.

1.3 Определение требований к системе и выбор средства разработки

В качестве средства разработки был выбрана система «1С:Предприятие», предназначенная для автоматизации деятельности на предприятии.

В системе "1С:Предприятие" существует четкое разделение на платформу и прикладное решение. Платформа представляет собой framework, в котором функционирует прикладное решение:

- платформа служит фундаментом для построения прикладных решений;
- платформа является средой их исполнения;
- платформа содержит инструментарий, необходимый для разработки, администрирования и поддержки прикладных решений.

При этом прикладное решение является самостоятельной сущностью и может выступать в качестве отдельного программного продукта. Но полностью опирается на технологии платформы.

Такой подход позволяет автоматизировать различные виды деятельности, используя единую технологическую платформу.

Функции программного комплекса «1С: Предприятие» классифицируются по направлениям автоматизации и группам пользователей. Эти функции системы имеют своей целью обеспечение руководителей информацией, необходимой для оценки ситуации и принятия актуальных решений. Это, например, такие механизмы, как бюджетирование, анализ рентабельности деятельности предприятия, сбыта продукции и многое другое. Эта функциональность решает задачи работников, которые занимаются торговой, производственной, а также деятельностью в области оказания услуг. С помощью системы можно эффективно организовать ежедневную работу организации: подготовка документов, управление выпуском продукции и запасами, оформление заказов, контроль исполнения задач. Еще одной важной возможностью программного комплекса является учет и отчетность. Данная функция решает задачи бухгалтерии: обеспечение ведения учета в соответствии с актуальными требованиями законодательства. Это такие задачи, как, например, расчет заработной платы.

Задачи управления и учета могут значительно отличаться в зависимости от сферы деятельности фирмы, отрасли, специфики производимой продукции или оказываемых услуг, структуры и размеров предприятия, уровня его автоматизации. Данная программа предназначена для массового использования и удовлетворяет потребности основного

количества предприятий. Таким образом, руководитель будет иметь решение с преимуществами применения массового продукта, которое соответствует специфике организации.

Таким образом, к преимуществам использования решений на данной платформе можно отнести:

- наличие большого количества потенциального персонала для работы в компании с внедренными решениями на базе 1С;
- максимально быстрая и качественная поддержка со стороны компании 1С;
- возможность создания и доработки индивидуальных проектов, учитывающих особенности бизнес-процессов каждой организации;
- существование встроенного объектно – ориентированного языка, специально разработанного компанией 1С, включая различные вспомогательные инструменты;
- полная открытость программных продуктов 1С, дающая возможность изменения и доработки кода любым 1С-программистом;
- единая технологическая платформа, с помощью которой достигается высокая стандартизация разработки, полная масштабируемость проектов и обеспечение быстрого внедрения современных технологий.

В качестве прикладного решения было выбрано «Управление небольшой фирмой». Данное решение рекомендуется использовать для ведения учета в торговых, сервисных и производственных компаниях малого бизнеса.

К преимуществам «1С:Управление небольшой фирмой» можно отнести»:

- хранение данных в одной информационной базе;
- автоматизация рутинных операций, быстрая подготовка информации;

- понятный интерфейс, обеспечивающий легкость освоения для начинающих пользователей и высокую скорость работы для опытных;
- возможность контроля над выполнением поручений;
- широкий спектр встроенных отчетов;
- простота обращения, не требующая глубоких знаний бухгалтерского и налогового учета;
- избегание кассовых разрывов;
- учет и планирование доходов и расходов;
- контроль из любой точки мира посредством настройки web-доступа.

Требования к системе были сформированы на основе предоставленного технического задания.

Система для управления спортклубом должна:

- предоставлять рабочее место администратору клуба с набором функций для:
  - регистрации спортсмена (клиента);
  - печати договора, заявления о приеме;
  - оформления и выдачи пластиковой клубной карты;
  - продажи абонемента;
  - планирования расписаний групп;
  - регистрации оплат;
  - регистрации оказанных услуг;
  - выдачи ключа от шкафчика;
  - блока работы с клиентом;
- давать возможность планировать расписание занятий для группы, спортсмена или тренировочного режима:
  - предоставлять информацию о частичной занятости зала, для занятий спортсменов в тренировочном режиме;
  - позволять редактировать множество занятий;

- представлять информацию о расписаниях в наглядном виде календаря в представлениях по дню, неделе, месяцу;

- хранить произвольные файлы, в том числе сканы различных документов, с привязкой к различным объектам системы;
- напоминать о необходимости продления аккредитации, медицинском осмотре спортсмена;
- позволять назначать спортсмену текстовую и цветовую метки;
- продлять завершившиеся абонементы в исключительных случаях, при наличии высших полномочий пользователя;
- настраивать и нумеровать ключи и шкафчики и контролировать выдачи ключей при регистрации посещения:
  - указание занятия при регистрации;
  - режим регистрации посещений:
    - при входе;
    - при выходе.

В процессе разработки должны быть реализованы следующие интерфейсы:

- рабочий стол администратора;
- рабочий стол тренера.

2 Проектирование и разработка информационной системы

2.1 Функциональное моделирование IDEF0 после внедрения информационной системы

Спортклуб работает в соответствии с законами Российской Федерации и положениями, которые разработало руководство спортклуба. Эти ограничения действуют на протяжении всей его деятельности. На входе процесса *Организация деятельности спортклуба – Запрос спортсмена*. Это означает, что вся его деятельность связана с работой со спортсменами. На выходе процесса *Клубная карта* и *Договор об оказании услуг*. Механизмы, которыми управляется процесс: *Персонал*, *Оборудование* и *Информационная система (ИС)*.

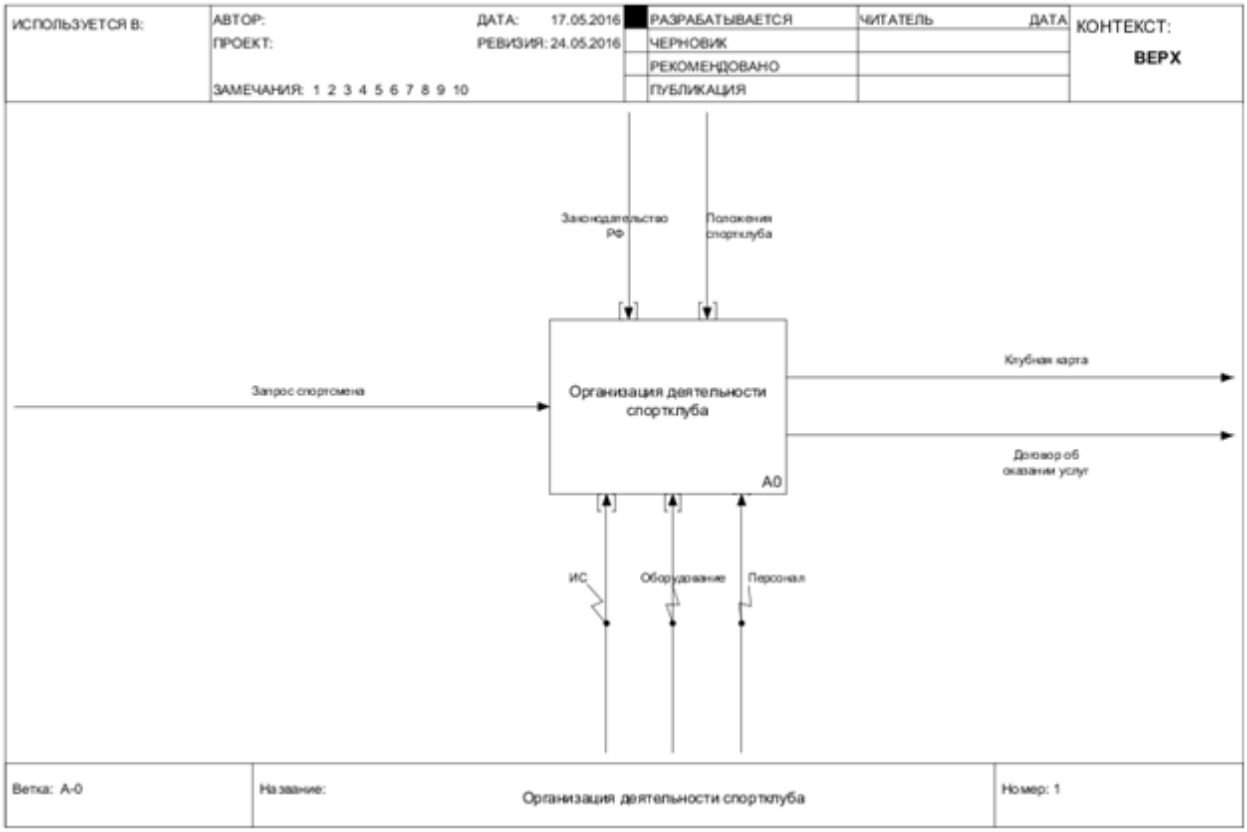


Рис. 8 Контекстная диаграмма IDEF0

Декомпозиция диаграммы разбивает процесс *Организация деятельности спортклуба* на подпроцессы, которые отражают его подробное описание (рис.9).

Первый подпроцесс *Составление расписания* (A1) имеет на входе два объекта: *Информацию о свободных помещениях клуба* и *Виды занятий*, на

выходе – *Расписание*, в соответствии с которым выполняется подпроцесс: *Проведение занятий*. Ограничением является *Режим работы* спортклуба. Составлением расписания занимается *Администратор*.

Второй подпроцесс *Продажа абонемента* (A2) имеет на входе объекты *Оплата абонемента* и *Запрос спортсмена* на продажу абонемента. На выходе – *Договор об оказании услуг, Клубная карта, Абонемент* и *Информация о группе* (в которую определили спортсмена). Продажа осуществляется *Администратором* в соответствии с *Прайс-листом* спортклуба.

Третий подпроцесс *Регистрация входа* (A3) на входе имеет *Клубную карту*, на выходе – *Клубную карту* и *Информацию о нахождении спортсмена в спортклубе*. Регистрация входа осуществляется *Администратором*.

Четвертый подпроцесс *Выдача инвентаря* (A4) на входе имеет *Клубную карту*, на выходе – *Клубную карту* и *Инвентарь*. Ограничением для выдачи инвентаря выступает *Абонемент* спортсмена. Подпроцесс осуществляется *Администратором*.

Пятый подпроцесс *Проведение занятий* (A5) на входе имеет *Спортсмена* и *Инвентарь,* а на выходе – *Инвентарь* и *Спортсмена,* который уже потренировался. Ограничениями являются *Расписание, Информация о группе* и *Абонемент. Проведение занятий* осуществляется *Тренером.*

Шестой подпроцесс *Сбор инвентаря* (A6) на входе имеет *Инвентарь* и *Клубную карту,* на выходе – *Клубную карту.* Подпроцесс осуществляется *Администратором.*

Заключительный седьмой подпроцесс *Регистрация выхода* (A7) на входе имеет *Клубную карту* и *Информацию о нахождении спортсмена в клубе,* на выходе – *Клубную карту*. Подпроцесс осуществляется *Администратором.*

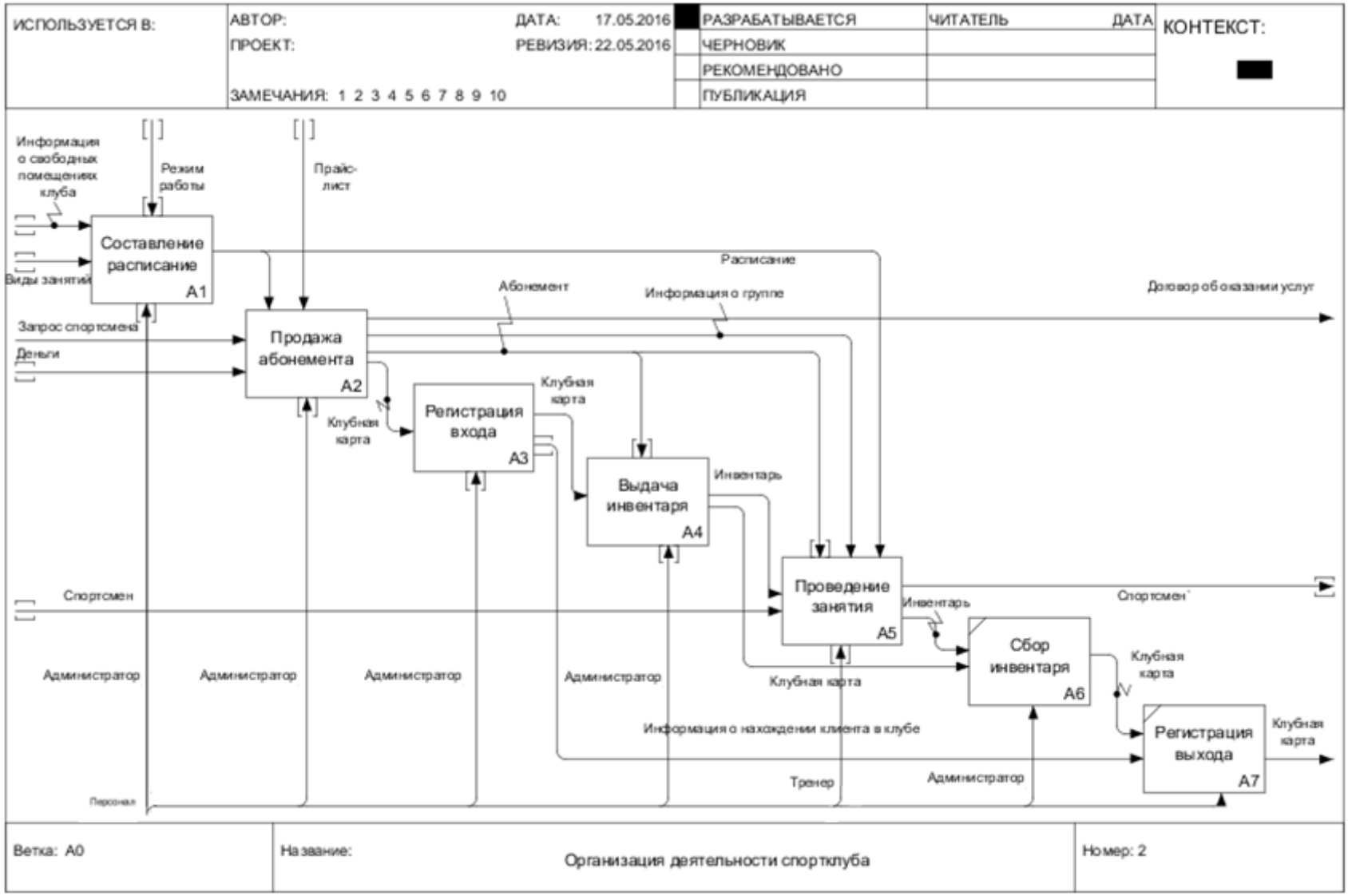


Рис. 9 Декомпозиция контекстной диаграммы IDEF0

1. Декомпозиция подпроцесса *Составление расписания* (рис.10):

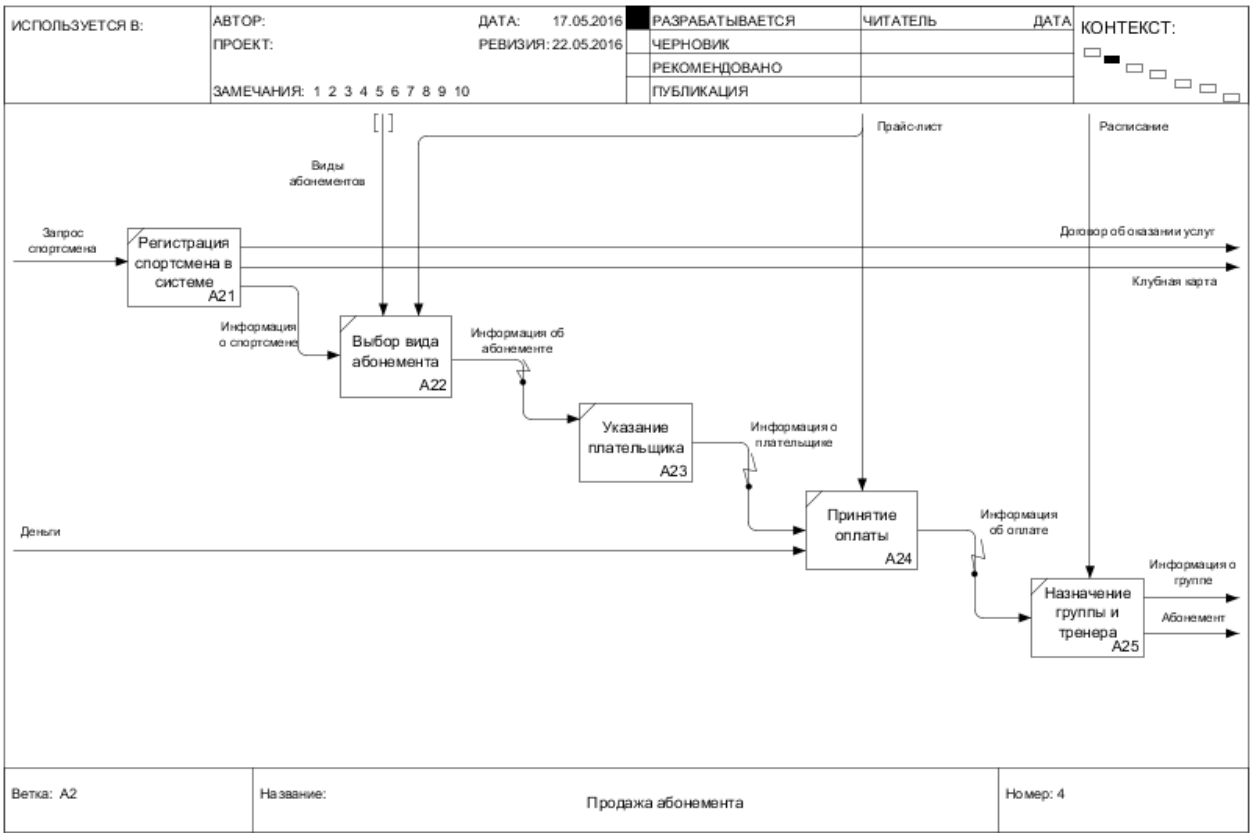


Рис. 10 Декомпозиция подпроцесса Составление расписания

Подпроцесс *Выбор вида занятия* (A11) имеет на входе объект *Виды занятий*. На выходе – *Информацию о выбранном виде занятий*.

Подпроцесс *Выбор тренера* (A12) имеет на входе объект *Виды занятий*. На выходе – *Информацию о выбранном тренере*.

Подпроцесс *Выбор группы* (A13) имеет на входе объект *Информация о тренере*. На выходе – *Информацию о группе*, для которой составляется расписание.

Подпроцесс *Выбор даты* занятия (A14) имеет на входе объект *Информация о группе*. На выходе – *Информацию о выбранной дате* занятия.

Подпроцесс *Выбор помещения* (A15) имеет на входе объекты: *Информация о выбранной дате* и *Информацию о свободных помещениях спортклуба*. На выходе – *Информацию о выбранном виде занятий*.

Подпроцесс *Ввод времени окончания* (A16) имеет на входе объект *Информация о выбранном помещении*. На выходе – *Информацию о времени начала* занятия.

Подпроцесс *Ввод времени окончания* (A17) имеет на входе объект *Информация о времени начала* занятий. На выходе – *Расписание*.

2. Декомпозиция подпроцесса *Продажа абонемента* (рис.11).

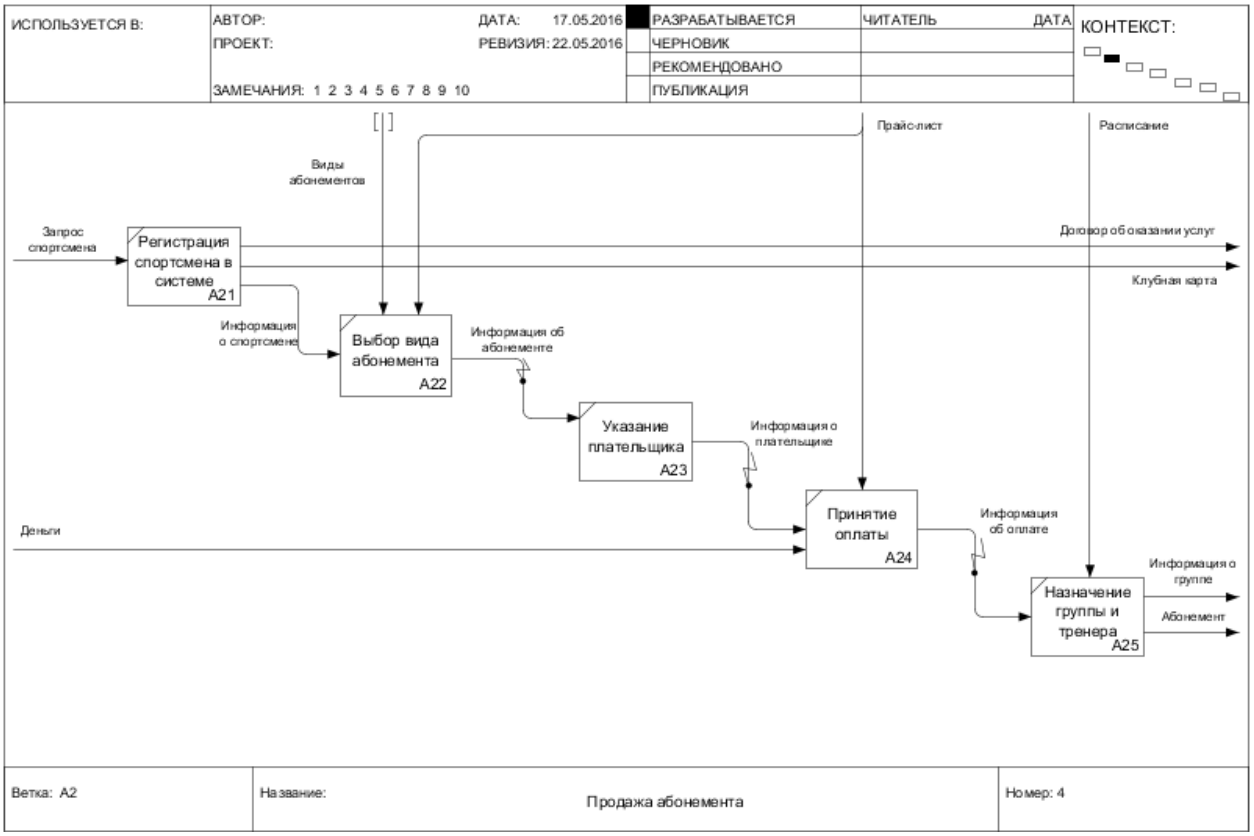


Рис. 11 Декомпозиция подпроцесса Продажа абонемента

Подпроцесс *Регистрация спортсмена в системе* (A21) имеет на входе *Запрос спортсмена,* а на выходе – *Договор об оказании услуг, Клубную карту* и *Информацию о спортсмене.*

Подпроцесс *Выбор вида абонемента* (A22) на входе имеет *Информацию о спортсмене,* а на выходе – *Информацию об абонементе.* При этом ограничениями выступают *Виды абонементов и Прайс-лист.*

Подпроцесс *Указание плательщика* (A23) на входе имеет *Информацию об абонементе*, а на выходе – *Информацию о плательщике.*

Подпроцесс *Принятие оплаты* (A24) на входе имеет *Информацию о плательщике,* а также *Деньги*, а на выходе – *Информацию об оплате*.

Заключительный подпроцесс *Назначение группы и тренера* (A25) имеет на входе *Информацию об оплате,* на выходе – *Информацию о группе,* в

которую определен спортсмен и *Абонемент.* Ограничением в данном случае выступает *Расписание.*

3. Декомпозиция подпроцесса *Регистрация входа* (рис.12).

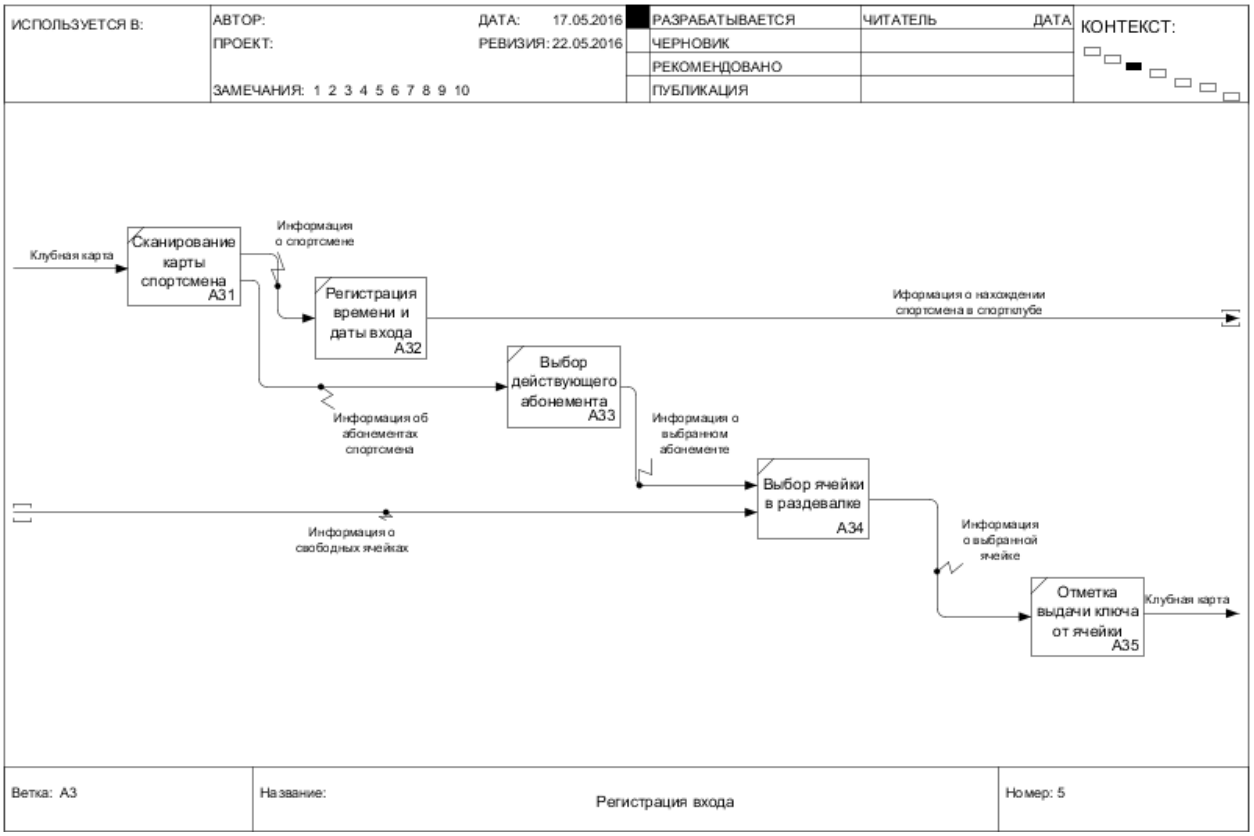


Рис. 12 Декомпозиция подпроцесса *Регистрация входа*

Подпроцесс *Сканирование карты спортсмена* (A31) имеет на входе *Клубную карту,* а на выходе *Информацию о спортсмене* и *Информацию об абонементе спортсмена.*

Подпроцесс *Регистрация времени и даты входа* (A32) имеет на входе *Информацию о спортсмене,* а на выходе *Информацию о нахождении спортсмена в спортклубе.*

*П*одпроцесс *Выбор действующего абонемента* (A33) имеет на входе *Информацию об абонементах спортсмена,* а на выходе *Информацию о выбранном абонементе.*

Подпроцесс *Выбор ячейки в раздевалке* (A34) имеет на входе *Информацию о выбранном абонементе* и *Информацию о свободных ячейках,* а на выходе *Информацию о выбранной ячейке.*

Подпроцесс *Отметка выдачи ключа от ячейки* (А35) имеет на входе *Информацию о выбранной ячейке,* а на выходе *Клубную карту.*

4. Декомпозиция подпроцесса *Выдача инвентаря* (рис.13).

Подпроцесс *Сканирование карты спортсмена* имеет на входе *Клубную карту,* а на выходе *Информацию о спортсмене* и *Информацию об абонементе* спортсмена.

Подпроцесс *Регистрация даты и времени выдачи инвентаря* имеет на входе *Информацию о спортсмене,* а на выходе *Информацию о дате и времени обращения.*

Подпроцесс *Выбор наименований инвентаря* имеет на входе *о дате и времени обращения,* а на выходе *Информацию о выбранном инвентаре.* Ограничения – *Информация об абонементе* и *Ассортимент инвентаря.*

Подпроцесс *Выбор количества инвентаря* имеет на входе *Информацию о выбранном инвентаря,* а на выходе *Инвентарь* и *Клубную карту.* Ограничением является *Ассортимент инвентаря.*

5. Декомпозиция подпроцесса *Проведение занятий* (рис.14).

Подпроцесс *Выбор тренера* имеет на входе *Запрос тренера,* а на выходе *Информацию о тренере.*

Подпроцесс *Выбор даты занятия* имеет на входе *Информацию тренере,* а на выходе *Информацию о дате проведения* занятия.

Подпроцесс *Выбор текущего занятия из списка* имеет на входе *Информацию о дате проведения и тренере,* а на выходе *Информацию о занятии, помещении и времени.* Ограничением является *Расписание.*

Подпроцесс *Отмечание присутствующих* имеет на входе *Информацию о занятии, помещении и времени,* а на выходе *Информацию о посещаемости.* Ограничением является *Информация о группе.*

Подпроцесс *Тренировочный процесс* имеет на входе *Спортсмена и Инвентарь,* а на выходе также *Инвентарь* и *Спортсмен,* но уже тренированный. Во всех подпроцессах, кроме последнего, механизмом выступает информационная система.

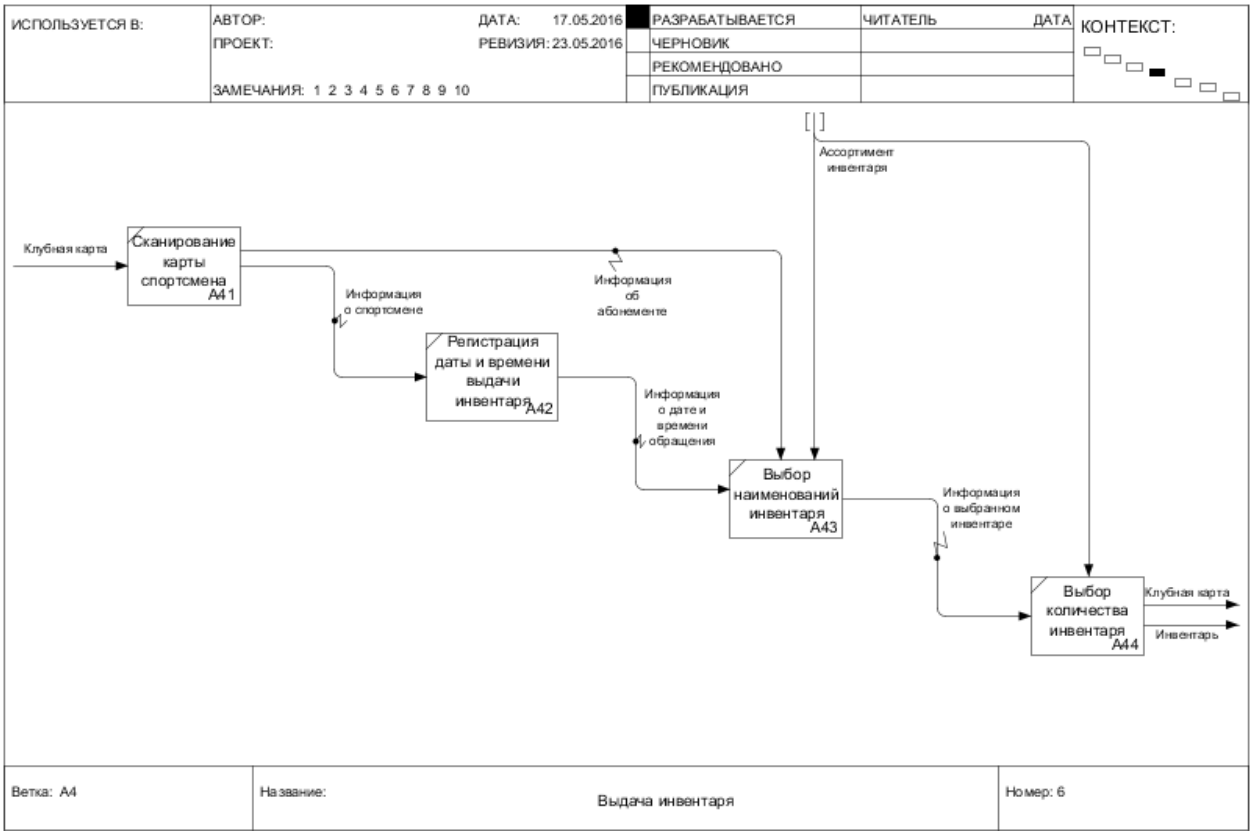


Рис. 13 Декомпозиция подпроцесса *Выдача инвентаря*

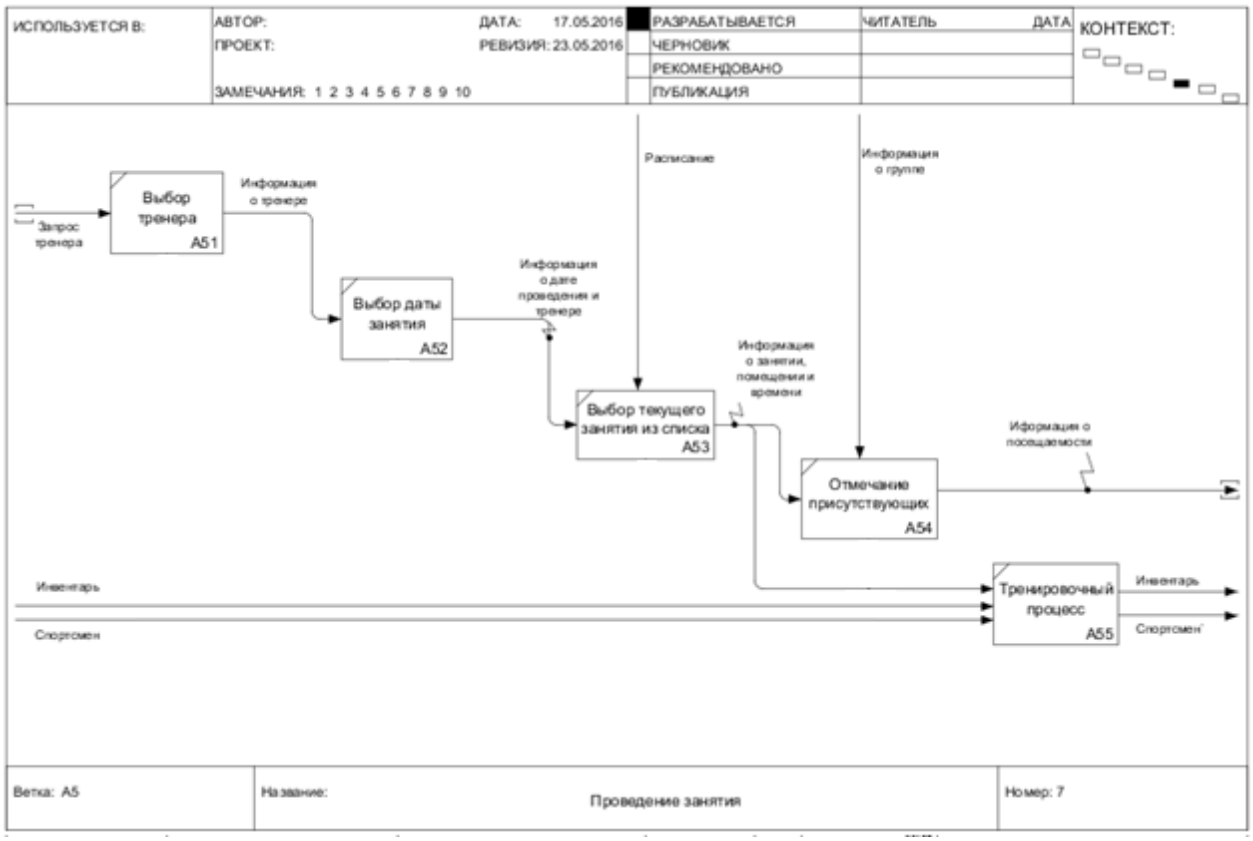


Рис. 14 Декомпозиция подпроцесса *Проведение занятия*

2.2 Роли пользователей в системе. UML моделирование

Информационная система предусматривает две роли пользователя:

- администратор;
- тренер.

1. Роль *Администратора* в информационной системе наиболее всеобъемлющая. Если пользователь находится в системе с этой ролью, то ему доступны все ее функции (просмотр данных о спортсменах, а также задолженности и оплаты услуг, управление абонементами спортсменов, продажи абонементов, составление расписания и пр.). Эту роль занимает администратор спортклуба. Все доступные *Администратору* действия в системе отражены на диаграммах вариантов использования (Use case diagram), разработанные с помощью программы Rational Rose. Общая диаграмма отражает основные подсистемы *Администратора*, далее эта диаграмма разбивается на более подробные, отражающие возможности каждой подсистемы. Такое разбиение диаграмм связано с тем, что *Администратор* выполняет достаточно большое количество действий в системе.

1.1. В первую очередь, *Администратор* занимается регистрацией посещений спортсменов и продажей абонементов. На диаграмме вариантов использования отражены соответствующие подсистемы (рис.15): *Управление абонементами* и *Регистрация посещения*. Также основной его работой является *Составление расписания* и ведение базы спортсменов. Для *Администратора* спортклуба есть возможность *Просмотра информации о спортсменах*, а именно: *Просмотр истории посещений*, *Просмотр информации об абонементах*, *Просмотр личных и спортивных сведений*.

Если для занятия необходимо выдать инвентарь, то перед началом занятия *Администратор* выдает его спортсменам и отмечает выдачу инвентаря в системе. В конце занятия он собирает инвентарь и так же создает отметку о том, что спортсмен его вернул.

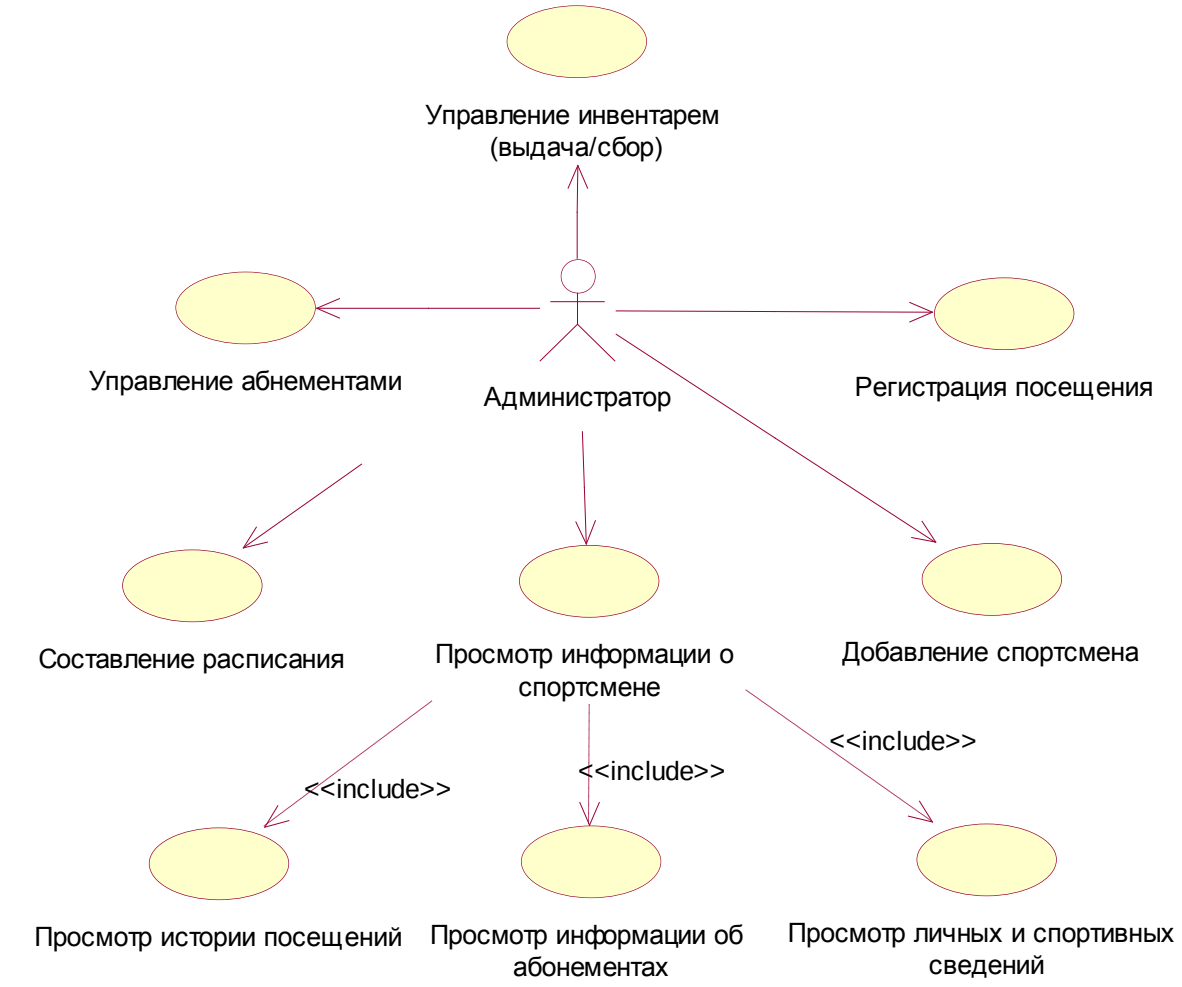


Рис. 15 Диаграмма основных подсистем администратора

1.1.1. Расширение варианта использования *Управление абонементами* (рис.16). Ключевым вариантом использования в этой диаграмме является *Продажа абонемента*, которая включает в себя *Заполнение данных спортсмена* и *Прием оплаты*. При просмотре информации о спортсмене *Администратор* может также *Отобразить все абонементы спортсмена*. При необходимости возможна *Приостановка абонемента* на определенное количество дней (необходимо указать), *Продление абонемента*, *Блокирование абонемента* также на определенное количество дней. Соответственно, *Разблокирование абонемента* и *Закрытие абонемента*. Еще одним вариантом использования является *Продажа абонемента повторно*.

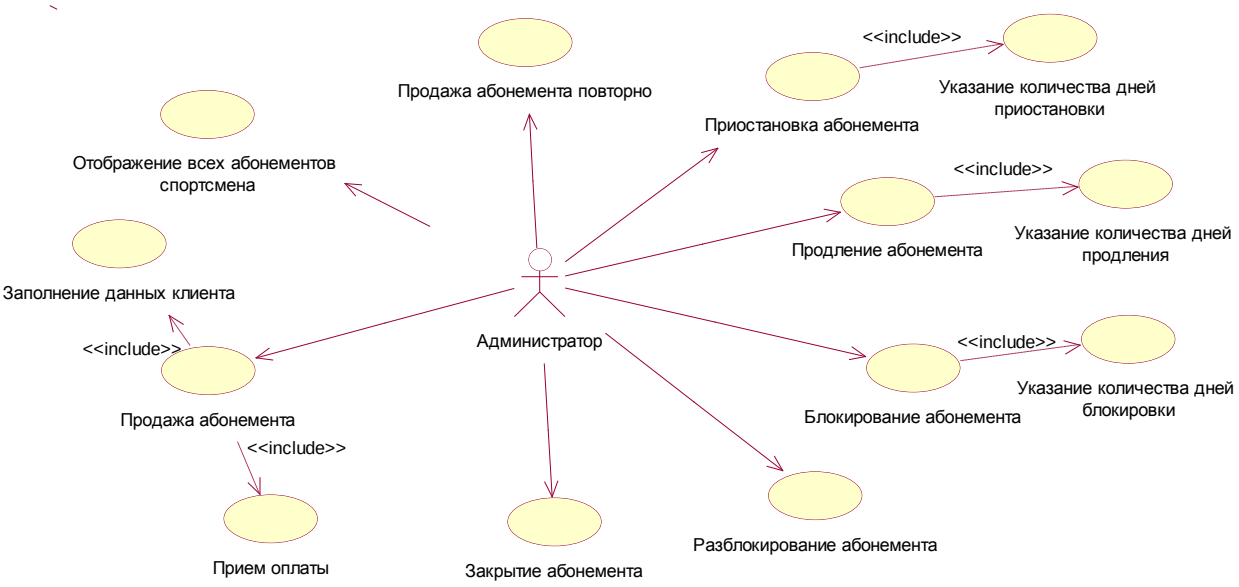


Рис. 16 Диаграмма вариантов использования подсистемы Управление абонементами

1.1.2. Расширение варианта использования *Регистрация посещения* (рис.17). При регистрации посещения спортклуба спортсменом *Администратор*, в первую очередь, идентифицирует спортсмена. Этот вариант использования может быть осуществлен как *Сканированием штрих-кода* (ввод штрих-кода спортсмена в поле), так и *Поиском спортсмена по ФИО*. Следующим обязательным вариантом использования является *Ввод времени входа* (или *выхода*) спортсмена в спортклуб. Также обязательным действием *Администратора* является *Отметка о выдаче ключа от ячейки (о получении ключа от ячейки* в случае выхода спортсмена из спортклуба*)*, которое включает в себя *Выбор пустой ячейки*. *Администратор* вправе *Выбрать абонемент спортсмена* в системе (особенно, если их несколько).

1.1.3. Расширение варианта использования *Управление инвентарем* (рис.18). При выдаче инвентаря *Администратору* предоставляется форма, в которой он заполняет данные об инвентаре, который выдал какому-либо спортсмену. В первую очередь, он *Выбирает спортсмена*, которому был выдан инвентарь. Следующим вариантом использования является *Выбор инвентаря*, который включает в себя *Выбор наименования* и *Выбор количества* инвентаря. Также обязательны *Ввод даты* и *Выбор*

ответственного за инвентарь (ФИО *Тренера*). Еще одним обязательным вариантом использования является *Выбор структурной единицы* – название филиала спортклуба. Опциональным вариантом использования является оставление комментария *Администратором*.

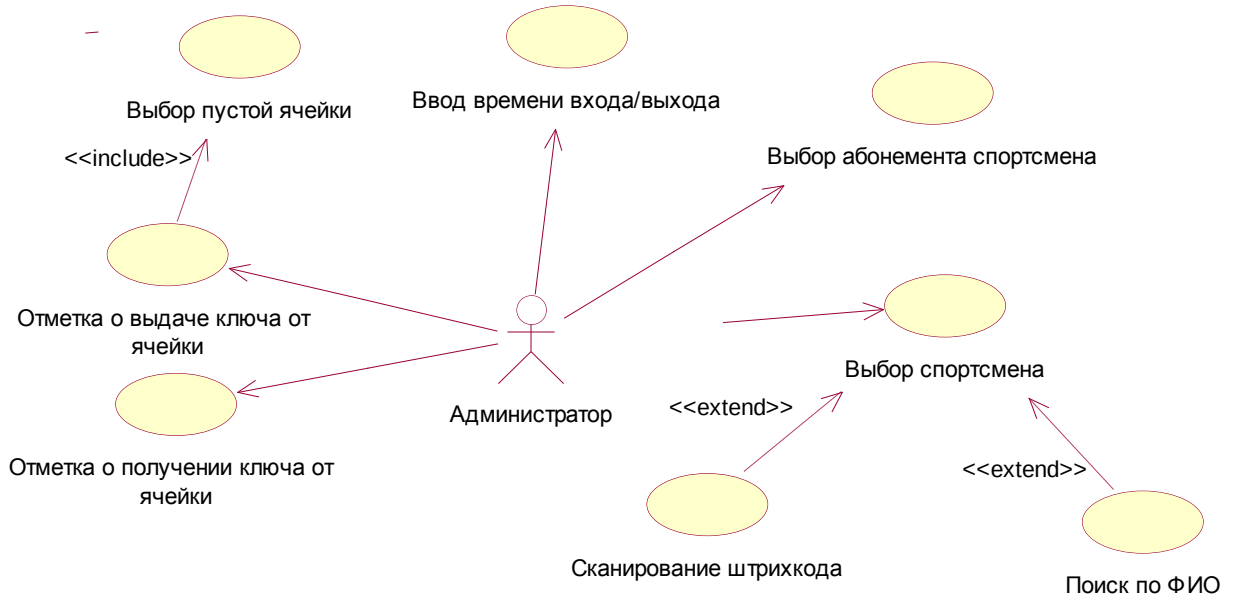


Рис. 17 Диаграмма вариантов использования подсистемы Регистрация посещений

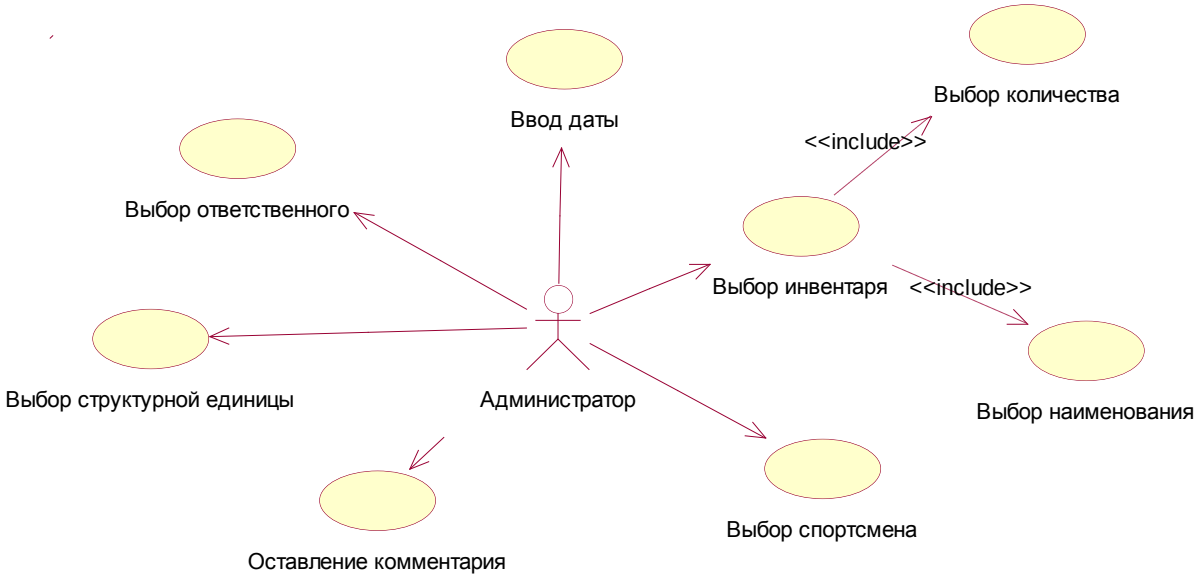


Рис. 18 Диаграмма вариантов использования подсистемы Управление инвентарем

1.1.4. Расширение варианта использования *Добавление спортсмена* (рис.19). Вариант использования *Добавление спортсмена* включает в себя *Ввод основных сведений о спортсмене*, а именно: *Ввод ФИО* спортсмена, *Ввод основной информации* (даты рождения, пола), опциональным вариантом

использования является *Ввод информации об опекунах* (если спортсмену нет 18 лет), который включает в себя *Ввод ФИО опекуна* и *Выбор родственной связи* со спортсменом. Также обязательным действием при добавлении нового спортсмена в базу является *Выдача карты*. Карта содержит штрих-код, по которому можно идентифицировать спортсмена.

При добавлении спортсмена в систему необязательным вариантом использования является *Ввод спортивных сведений о спортсмене*. Этот вариант использования включает в себя *Выбор первого тренера*, *Ввод информации о медицинском допуске*, *Ввод срока истечения допуска* (если таковой имеется), *Выбор спортивной организации*, если спортсмен занимался в каком-либо другом спортклубе до этого. Если у спортсмена имеется какая-либо квалификация, то обязательным вариантом использования является *Ввод* его *спортивной квалификации*, а также *Ввод* его *технической квалификации*.

1.1.5. Расширение варианта использования *Составление расписания* (рис.20). При составлении расписания *Администратор* сначала выбирает вид занятия (групповые или индивидуальные). Затем планирует эти занятия. *Планирование групповых занятий* включает в себя *Ввод периода*, в течение которого будет действовать расписание, *Ввод номера помещения*, *Ввод группы*, для которой составляется расписание, затем добавляется ФИО тренера и время тренировки. Расширением варианта использования *Планирование групповых занятий* является *Планирование индивидуальных занятий*, которое предусматривает расписание для 1-2 спортсменов. Также *Планирование расписания* предусматривает *Планирование мероприятий*. Оно включает в себя *Ввод наименования* мероприятия, *Ввод* ее *даты*, *места проведения*, обязательно должен быть указан организатор мероприятия. В таблице *Администратор* указывает ФИО участников мероприятия, по желанию может добавить его описание. Так как расписание согласовывается с тренерами, при составлении групповых и индивидуальных тренировок указывается ответственный за расписание.

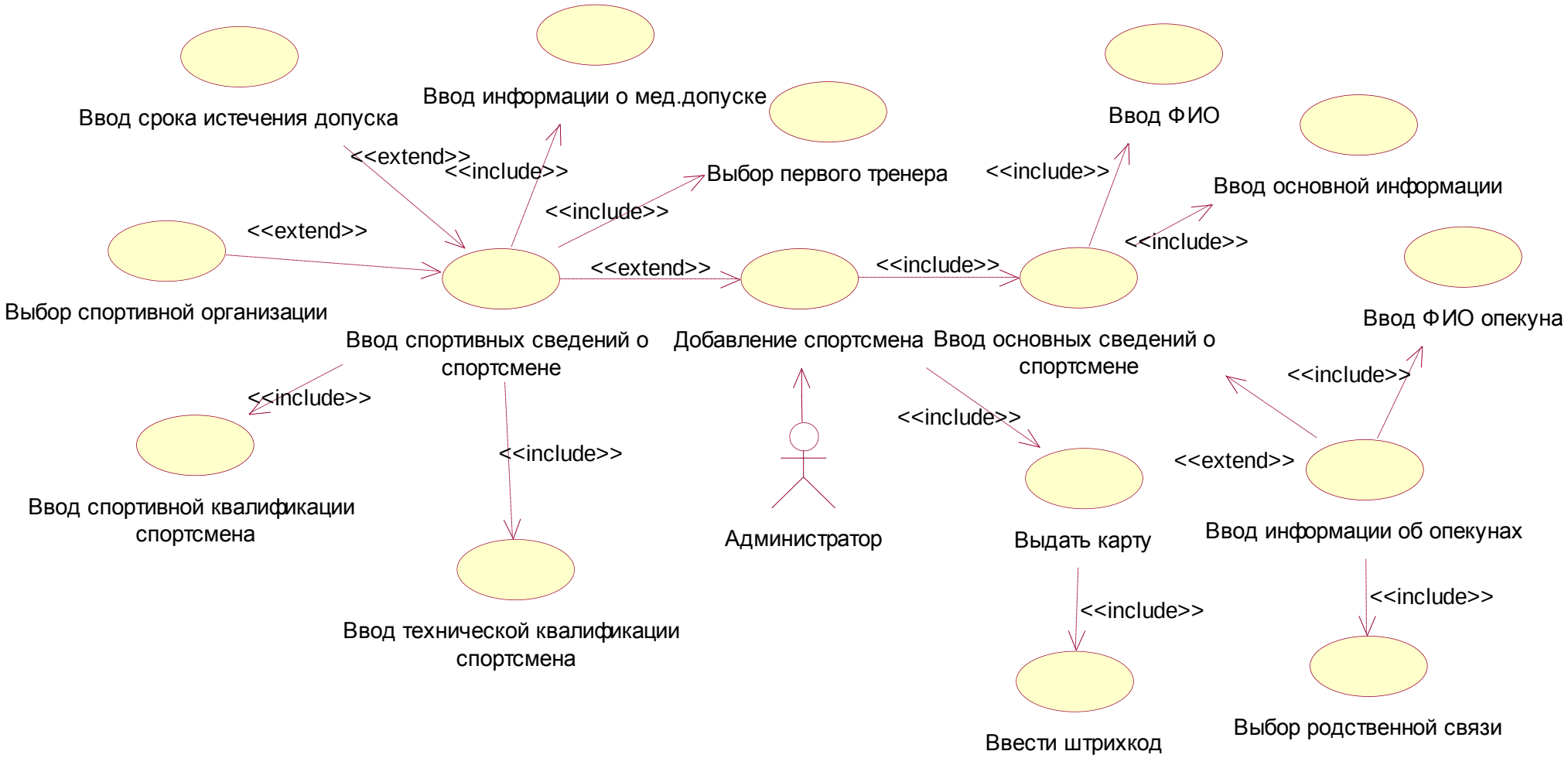


Рис. 19 Диаграмма вариантов использования подсистемы Добавление спортсмена

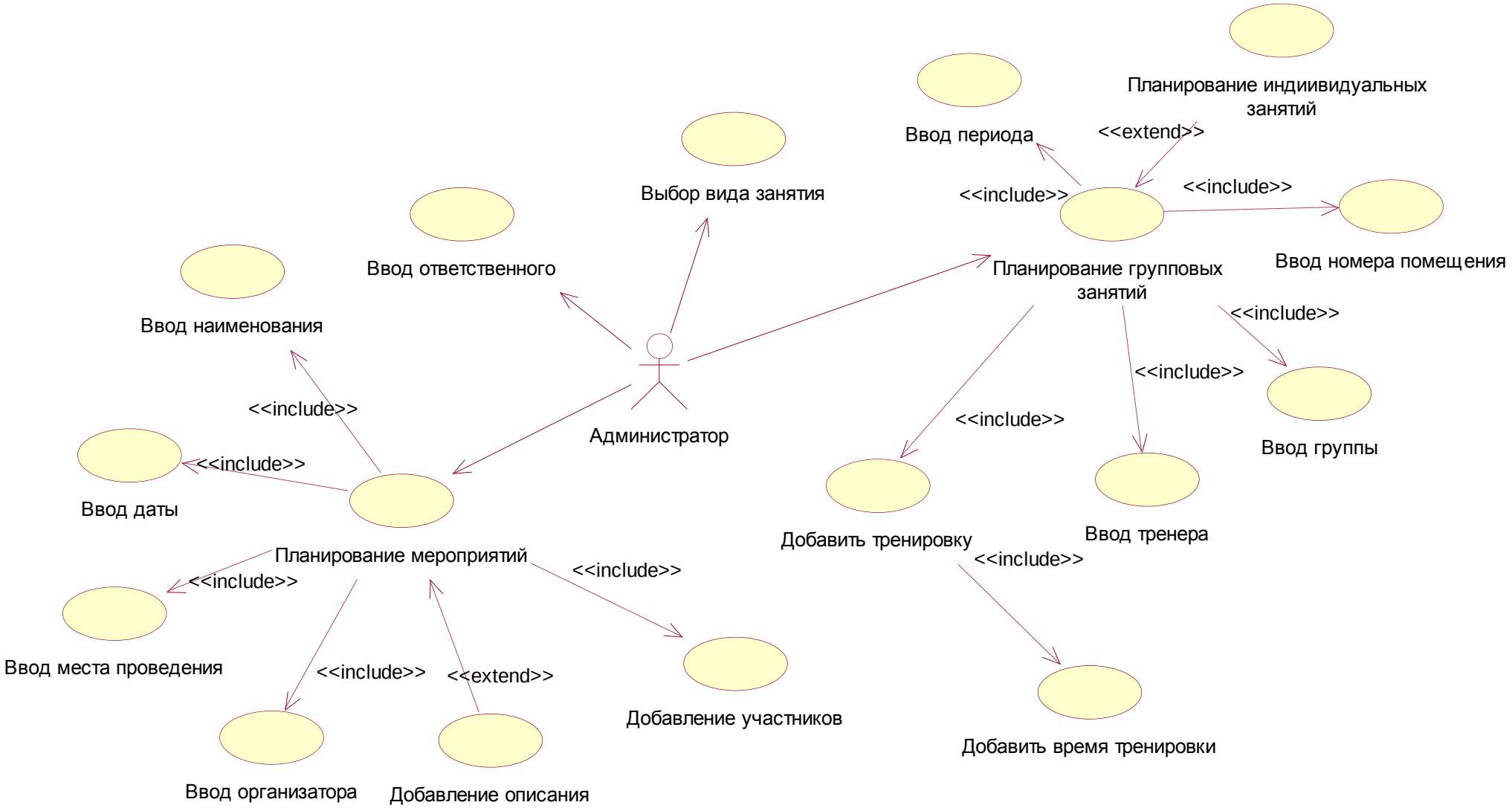


Рис. 20 Диаграмма вариантов использования подсистемы Составление расписания

1.2. Помимо диаграмм вариантов использования были спроектированы диаграммы последовательности (Sequence diagram) с помощью тех же средств разработки, что и диаграммы вариантов использования. Они отражают необходимую последовательность действий в системе, которая должна осуществляться при ее использовании.

1.2.1. Диаграмма последовательности «Добавление нового спортсмена» (рис.21) показывает, какие действия должен совершить *Администратор* для добавления спортсмена в справочник. На *Форме «Создание спортсмена»* он вводит ФИО спортсмена, основные данные (пол, дата рождения), информацию об опекунах (если спортсмену нет 18 лет). Если спортсмен состоял уже в других спортивных клубах, то *Администратор* вводит спортивные сведения о спортсмене, если нет – оставляет таблицу пустой. После этих действий он отправляет *Запрос на добавление спортсмена* в справочник. Новый спортсмен добавляется в *Справочник «Спортсмены»*.

Рис. 21 Диаграмма последовательности «Добавление нового спортсмена»

1.2.2. Диаграмма последовательности «Продажа абонемента» (рис.22) спроектирована для пользователя с ролью *Администратора*. Для того чтобы продать абонемент, он сначала указывает данные спортсмена (покупателя) на

Форме «Параметры абонемента»: выбирает из справочника спортсменов ФИО покупателя, указывает вид клубной карты (если спортсмен покупает ее. В противном случае не выбирается), ФИО плательщика из справочника контрагентов (следует указать опекуна, если спортсмену нет 18 лет).

Далее на *Форме «Добавление абонемента»* в таблице *Администратор* указывает *Вид абонемента*, который выбрал покупатель. В этой же таблице он также может отметить возможность заморозки абонемента с указанием количества дней, если таковое отмечено.

После ввода данных *Администратор* запрашивает открытие Акта продажи. Обрабатывая его запрос, система отправляет данные с *Форм «Параметры абонемента»* и *«Добавление абонемента»* на *Форму «Акт продажи»*. Открывшаяся перед *Администратором* форма предлагает ввести ему способ оплаты. Имеется возможность начисления скидки (если ее нет, ничего не указывается), после ввода этих данных на форму принимается оплата от спортсмена и сумма оплаты указывается на *Форме*. После этого *Администратор* отправляет *Запрос на создание Акта продажи*. Документ создается и добавляется к *Актам выполненных работ*, последним действием *Администратор* запрашивает создание *Документа «Продажа абонемента»*, документ создается, вместе с ним создаются необходимые бухгалтерские проводки.

1.2.3. Диаграмма последовательности «Регистрация посещения» (рис.23) предполагает, что *Администратор* на *Главной форме* вводит ФИО пришедшего спортсмена из справочника спортсменов, затем выбирает его абонемент, по которому он пришел заниматься в списке имеющихся абонементов у спортсмена. После этого открывает *Форму «Посещения»,* и данные о спортсмене с *Главной формы* передаются на текущую форму. На этой форме *Администратор* указывает время входа спортсмена в спортклуб, отмечает выдачу ключа спортсмену и вводит номер этого ключа. После этого он отправляет *Запрос на создание документа*, если все данные внесены, то *Документ «Посещения»* успешно создается.

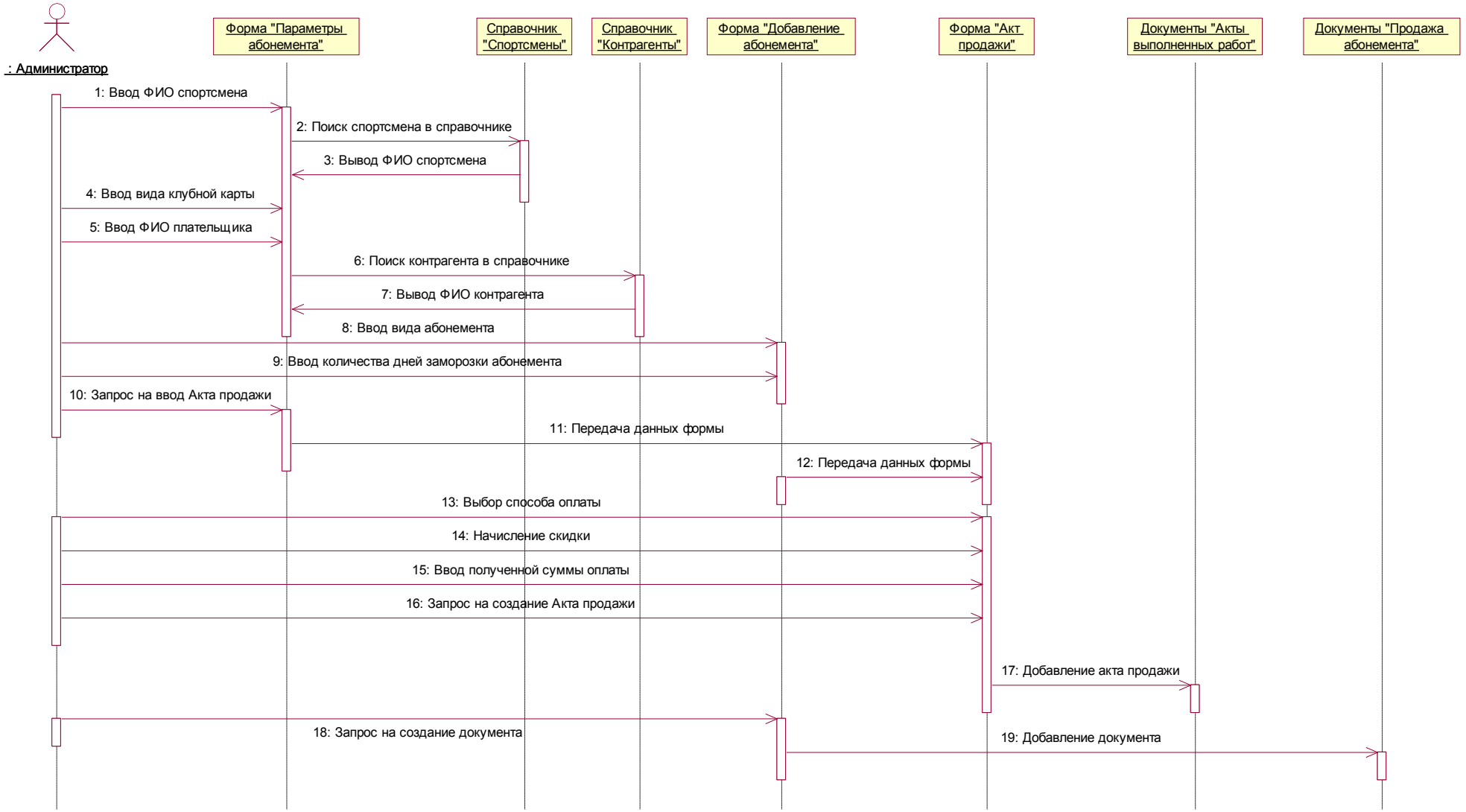


Рис. 22 Диаграмма последовательности «Продажа абонемента»

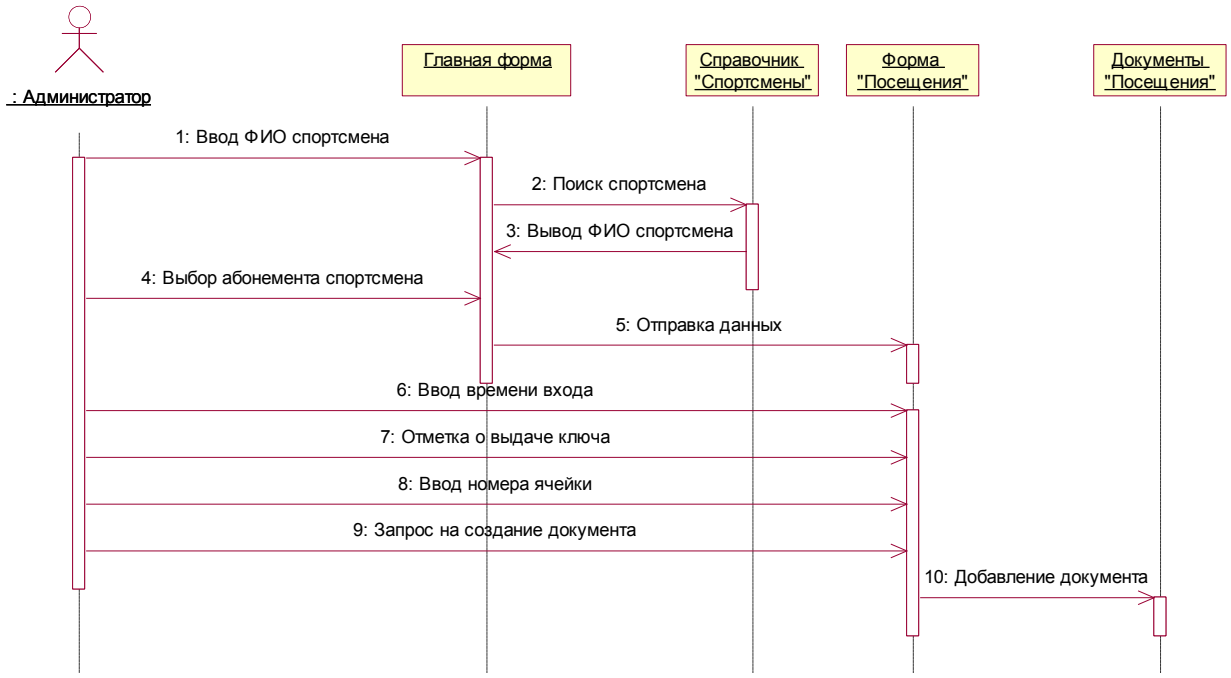


Рис. 23 Диаграмма последовательности «Регистрация посещения»

1.2.4. Диаграмма последовательности «Выдача и получение инвентаря» (рис.24) создана для определения действий *Администратора* в случае необходимости использования инвентаря на тренировках.

При запросе спортсмена на выдачу инвентаря *Администратор* вводит ФИО спортсмена из справочника спортсменов на *Форме «Выдача инвентаря»,* текущую дату, структурную единицу из справочника организационно-структурных единиц компании, в таблице вводит наименования выданного инвентаря из справочника номенклатуры и его количество. Внизу формы – ФИО ответственного. В качестве ответственного выступает тренер спортклуба (также выбирается из справочника тренеров). После этого *Администратор* отправляет *Запрос на создание документа*. Все данные заносятся в *Документ «Выдача инвентаря».* После тренировки *Администратор* на *Форме «Получение инвентаря»* вводит ФИО спортсмена (из справочника спортсменов), который сдает инвентарь, а также документ, на основании которого он выдал инвентарь этому спортсмену. Форма получает данные с указанного документа, затем *Администратор* указывает в таблице, какой инвентарь сдает спортсмен, отправляет *Запрос на добавление Документа «Возврат инвентаря»*. После этого документ создается.

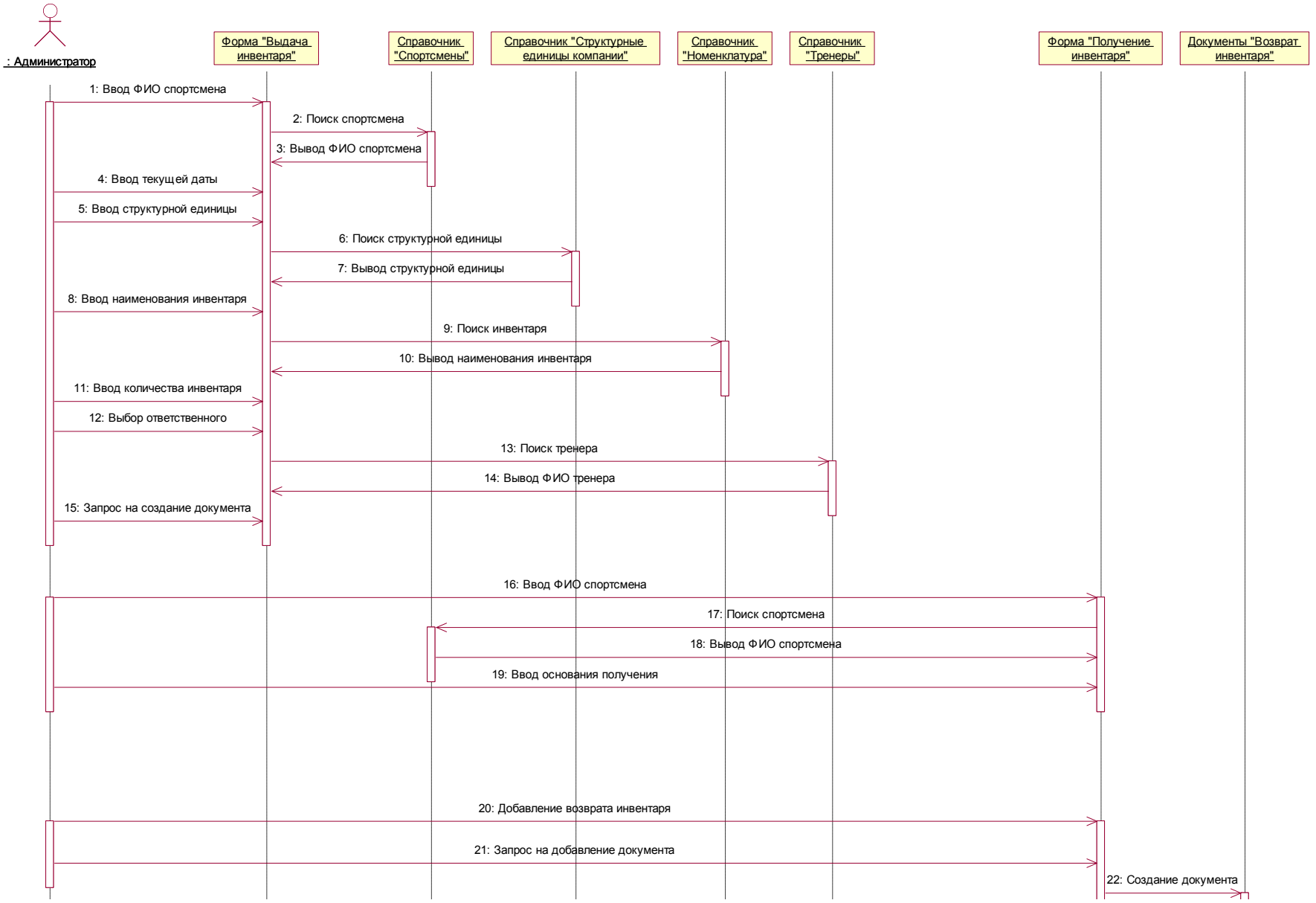


Рис. 24 Диаграмма последовательности «Выдача и получение инвентаря»

1.2.5. Диаграмма последовательности «Составление расписания» (рис.25) показывает, какие действия совершает *Администратор* для создания расписания для групповых занятий.

На *Форме «Планирование занятий» Администратор* вводит период, в течение которого будет действовать это расписание, также выбирает номер помещения из справочника помещений, группу из справочника групп, тип занятия (групповое или индивидуальное) и ФИО тренера из справочника тренеров. В таблице на этой же форме добавляется тренировка – конкретно время, на которое она запланирована. Далее *Администратор* отправляет запрос на создание расписания. После этого создается *Документ «Планирование занятий»*.

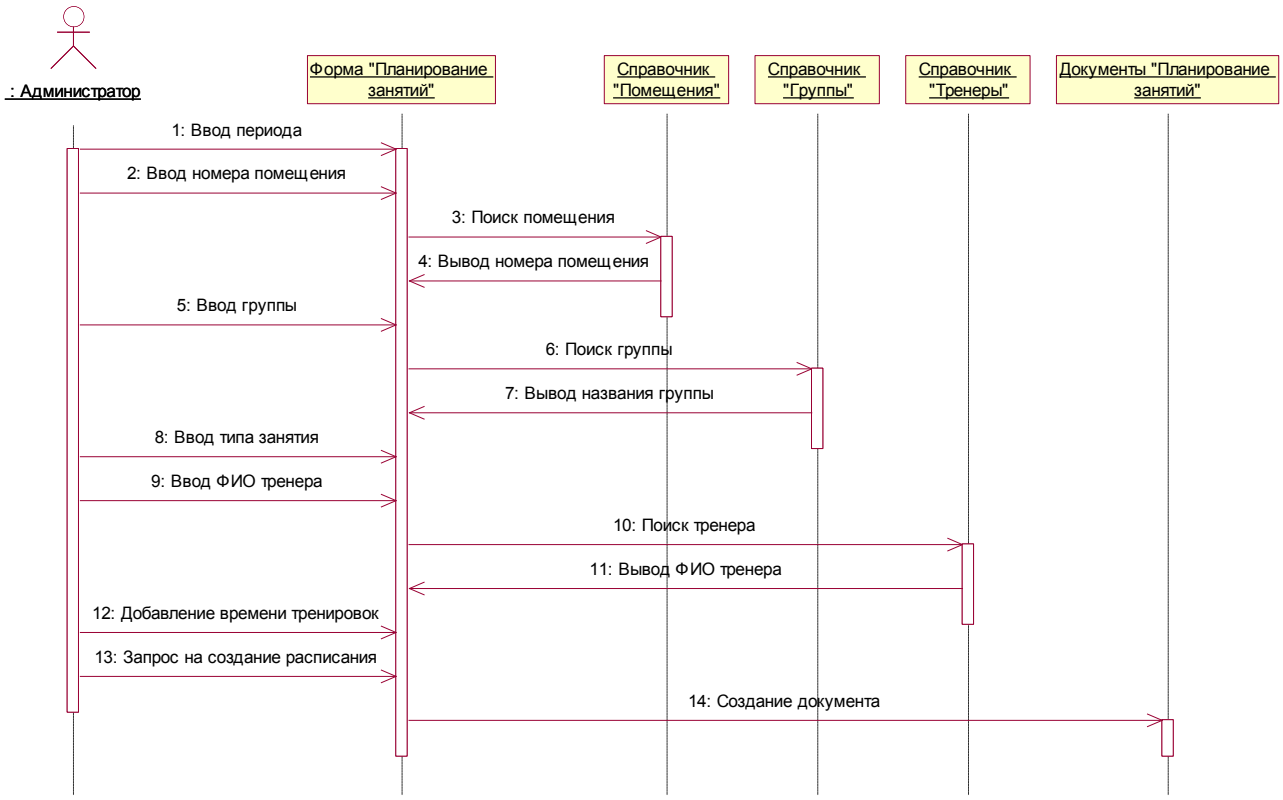


Рис. 25 Диаграмма последовательности «Составление расписания»

2. Роль *Тренера* в информационной системе занимает тренер спортклуба, который проводит занятия. В основном его пребывание в системе связано с регистрацией посещения занятий спортсменами. Были спроектированы диаграмма вариантов использования для тренера и диаграммы последовательности. Все доступные *Тренеру* действия в системе отражены на диаграмме вариантов использования (рис.26).

2.1. Во-первых, *Тренер* должен заполнить данные о проводимом занятии. Эти данные включают в себя: дату и время проведения занятия, номер помещения, услугу, которую оказывает *Тренер* (индивидуальные тренировки, разовое занятие, различные виды спорта), и свое имя, которое уже есть в системе, если тренер работает в спортклубе.

Во-вторых, *Тренер* должен отметить присутствующих на занятии спортсменов. Это действие включает в себя добавление присутствующего спортсмена в список посетивших спортсменов, добавление вида абонемента спортсмена (т.к. у одного спортсмена может быть несколько видов абонемента), и непосредственно отметка о присутствии спортсмена на занятии. Также необходимо добавить статус занятия (завершено, запланировано, проводится и т.д.).

Тренер также может посмотреть расписание групп на выбранную дату. Отображается группа, которая посещает тренировки в этот день; время, на которое назначена тренировка; и помещение, в котором будет проводиться занятие.

Система также предусматривает создание бланка проведенного занятия, который включает в себя все данные о нем. Соответствующим вариантом использования для *Тренера* будет *Просмотр бланка занятия*. Этот бланк при необходимости можно распечатать, т.к. он является печатным документом, и предоставить администратору для отчетности.

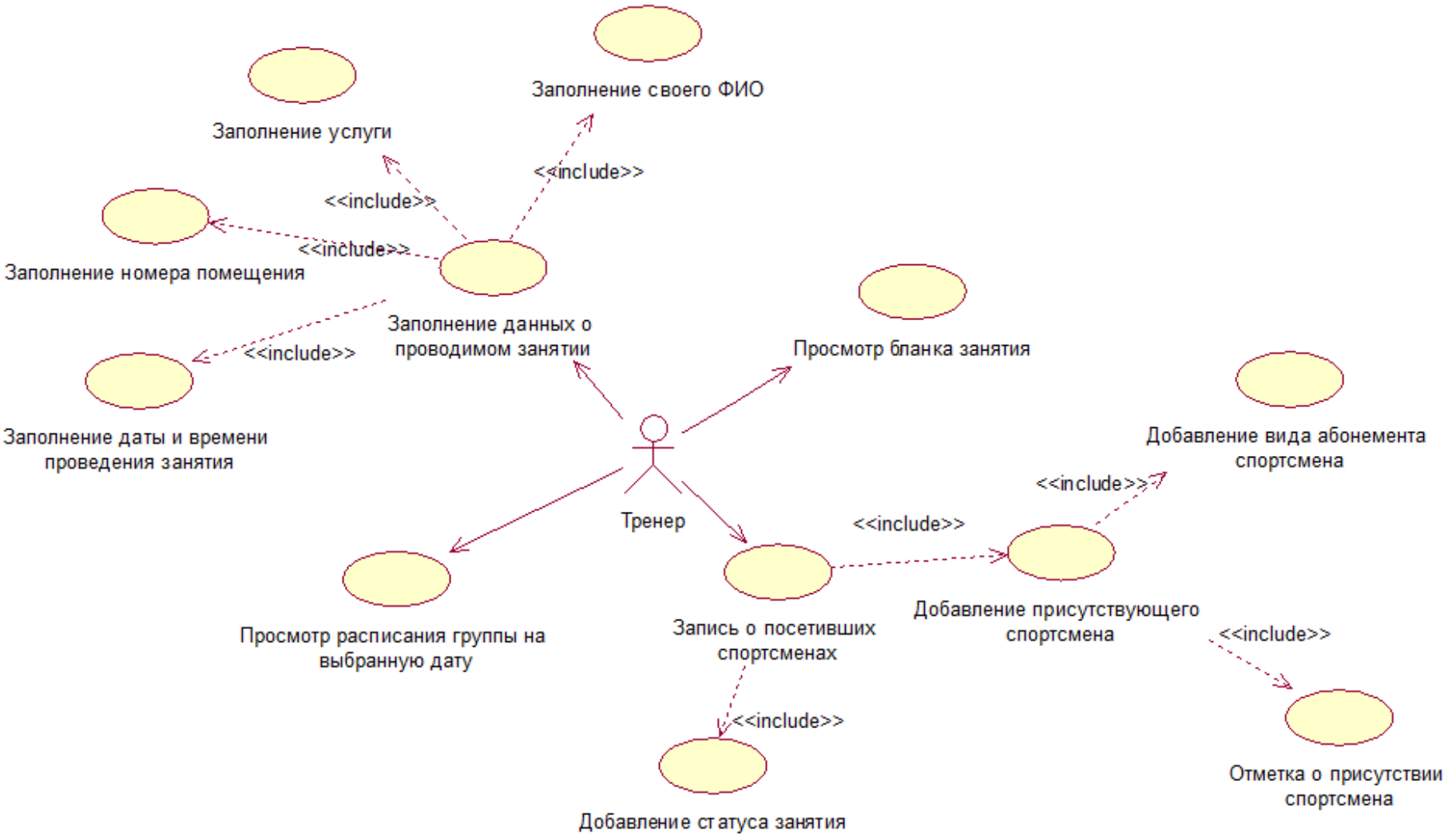


Рис. 26 Диаграмма вариантов использования тренера

2.2. Также для роли *Тренера* были спроектированы диаграммы последовательности, отражающие действия тренера в системе.

2.2.1. Диаграмма последовательности «Заполнение данные о проводимом занятии» (рис.27) демонстрирует то, как *Тренер* осуществляет заполнение информации о тренировке. На *Форме «Занятие» Тренер* указывает дату и время тренировки, вводит наименование своей услуги из справочника услуг, номер помещения из справочника помещений, а также свое имя из справочника тренеров. На *Форме «Посетившие» Тренер* вводит данные о присутствующих спортсменах. В таблицу он добавляет ФИО присутствующего спортсмена из справочника спортсменов, статус занятия, вид абонемента спортсмена также из справочника спортсменов, затем отмечает его присутствие на тренировке.

Чтобы получить бланк занятия, на *Форме «Занятие» Тренеру* необходимо запросить его. С этой формы данные передаются на *Печатный документ*, а тот затем отображает все данные о занятии в удобной форме.

2.2.2. Диаграмма последовательности «Вывод расписания» (рис.28) показывает действия *Тренера* при просмотре занятий групп на выбранную дату. Для начала Тренеру необходимо ввести свое имя из справочника тренеров, затем интересующую его дату в поле на *Форме расписания*. После отправить *Запрос на вывод расписания* в таблице, нажав соответствующую кнопку. Если есть какие-либо занятия в этот день у тренера, то они найдутся среди *Документов «Планирование занятий»* и отобразятся в таблице с расписанием. В итоге на форме расписания тренер увидит группы на выбранную дату, помещение, в котором будут проводиться тренировки, и время тренировок.

Таким образом, для информационной системы были спроектированы диаграммы вариантов использования и диаграммы последовательности для двух ролей: *Администратора* и *Тренера* спортклуба.

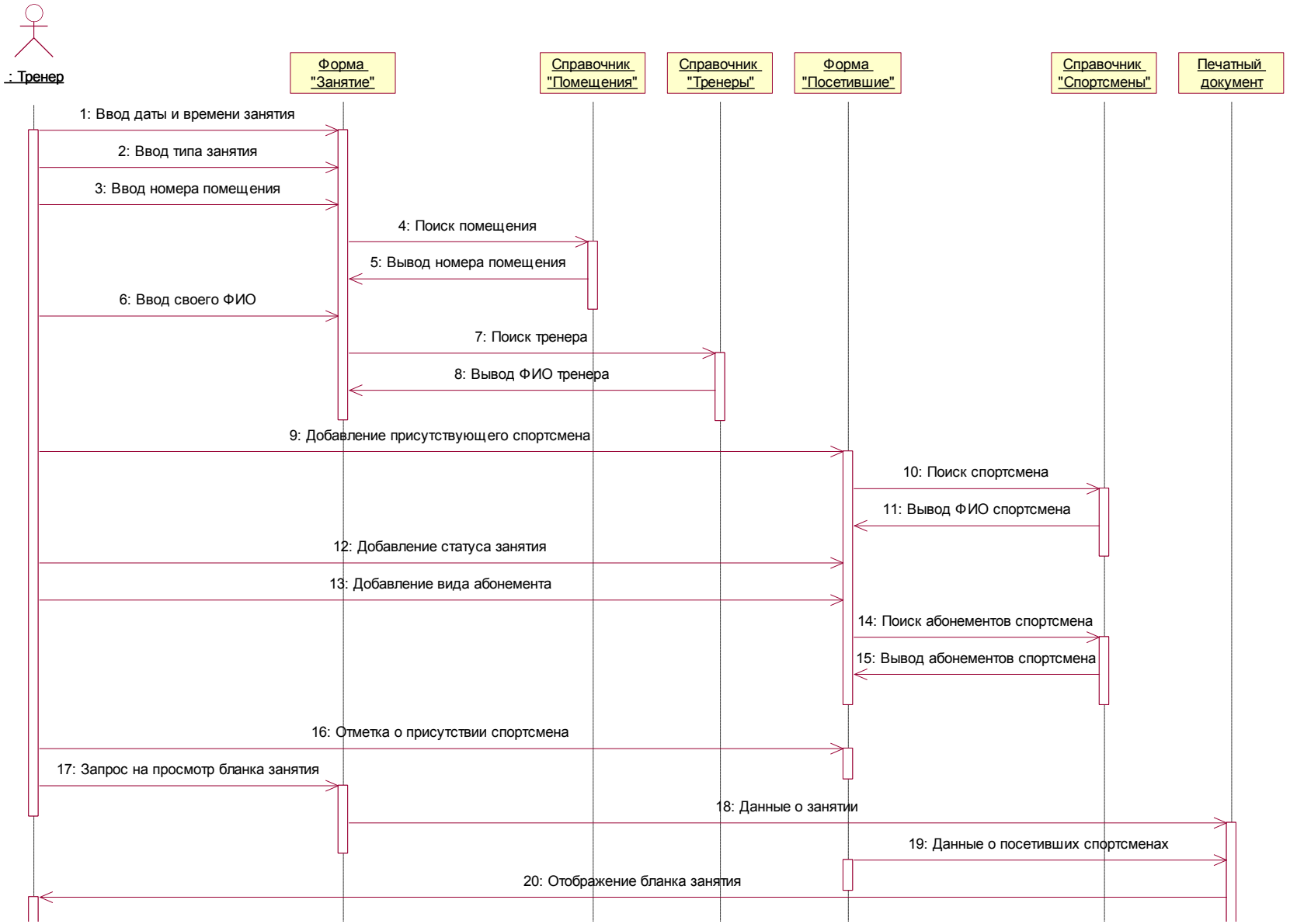


Рис. 27 Диаграмма последовательности «Заполнение данных о проводимом занятии»

Рис. 28 Диаграмма последовательности «Вывод расписания»

2.3 Диаграммы потоков данных

Для описания реально существующих в организации потоков данных были разработаны DFD-диаграммы. Администратор имеет возможность вносить новые данные в информационную систему или получать уже имеющуюся информацию. Для тренера в ИС существует страница, на которой он может посмотреть расписание и информацию о конкретном занятии, во время проведения которого может вносить данные о присутствии спортсменов. Описанные взаимодействия представлены на рис.29.

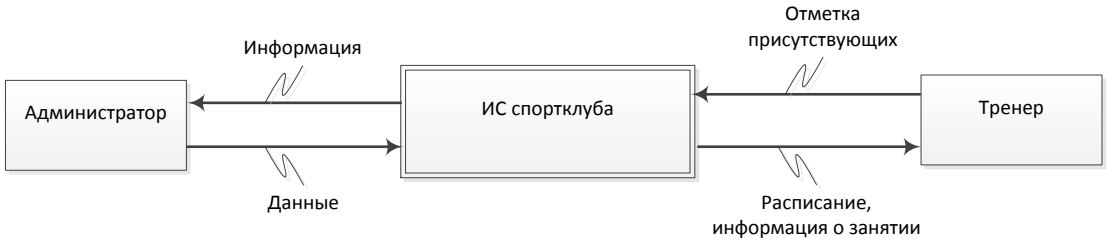


Рис. 29 Взаимодействия с информационной системой

ИС разделена на 3 основных функциональных блока: *АРМ администратора*, *АРМ тренера* и *Расписание*. Первый блок содержит в себе следующие функции:

- просмотр информации о спортсмене;

- продажа абонемента;
- изменение абонемента;
- управление инвентарем;
- регистрация посещения;
- создание спортсмена.

Второй функциональный блок *АРМ тренера* включает в себя 3 основные функции: заполнение данных о проводимом занятии, отметка присутствующих и просмотр бланка занятия.

Третий функциональный блок включает в себя работу с расписанием, содержащую следующие функции:

- создание занятия;
- редактирование занятия;
- выбор периода отображения;
- отображение для выбранного тренера/группы.

В целом функционирование информационной системы схематично изображено на рис.30.

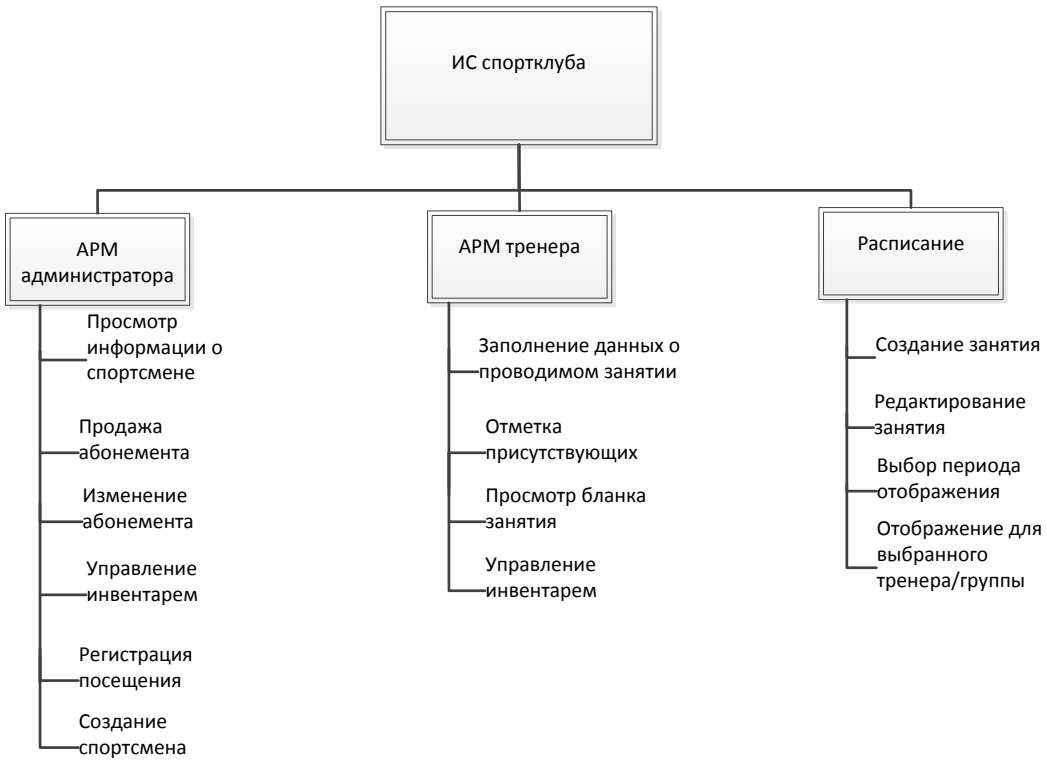


Рис. 30 Функционирование ИС

Далее рассмотрены подробно некоторые функции. Для создания спортсмена администратор должен выполнить следующие действия (рис.31.):

- ввести информацию о спортсмене;
- ввести информацию о карте путем выбора из справочника существующей карты или создания новой;
- ввести информацию о первом тренере, работая со справочником *Тренеры*;
- выбрать весовую категорию из перечисления;
- выбрать квалификацию из справочника. При отсутствии создать новую.

После выполнения вышеприведенных действий будет создан новый элемент справочника *Спортсмены* и выведено соответствующее сообщение.

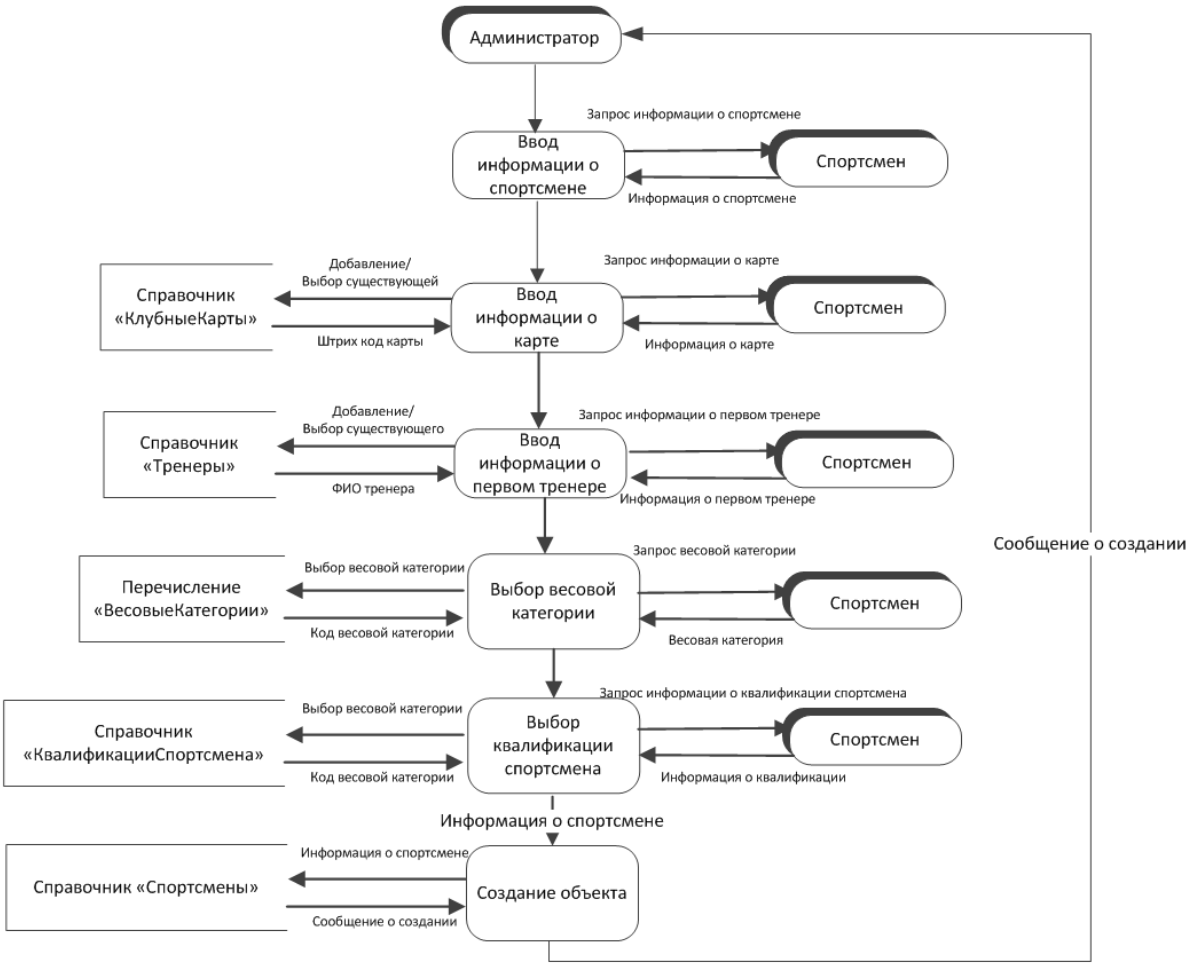


Рис. 31 Декомпозиция функции «Создание спортсмена»

Продавать абонементы на посещение спортклуба может только администратор. Для этого необходимо выбрать из справочника спортсмена, которому необходимо продать абонемент, и ввести информацию о карте, контрагенте и абонементе, используя соответствующие справочники (рис.32.). В результате создается документ *Продажа абонемента*.

Рис. 32 Декомпозиция функции «Продажа абонемента»

Когда администратор отмечает посещение спортсмена, он вводит дату и время входа или выхода спортсмена, выбирает спортсмена, абонемент и номер используемой ячейки, используя справочники, указанные на рис.33. Эти действия фиксируются в документе *Посещение*.

Выдавать и собирать инвентарь может администратор в соответствии с абонементом спортсмена. При создании документа *Выдача инвентаря* ИС обращается к справочникам *Спортсмены*, *Инвентарь*, *Сотрудники* и *Структурные единицы* для получения соответствующей информации. Полная декомпозиция этой функции представлена на рис.34.

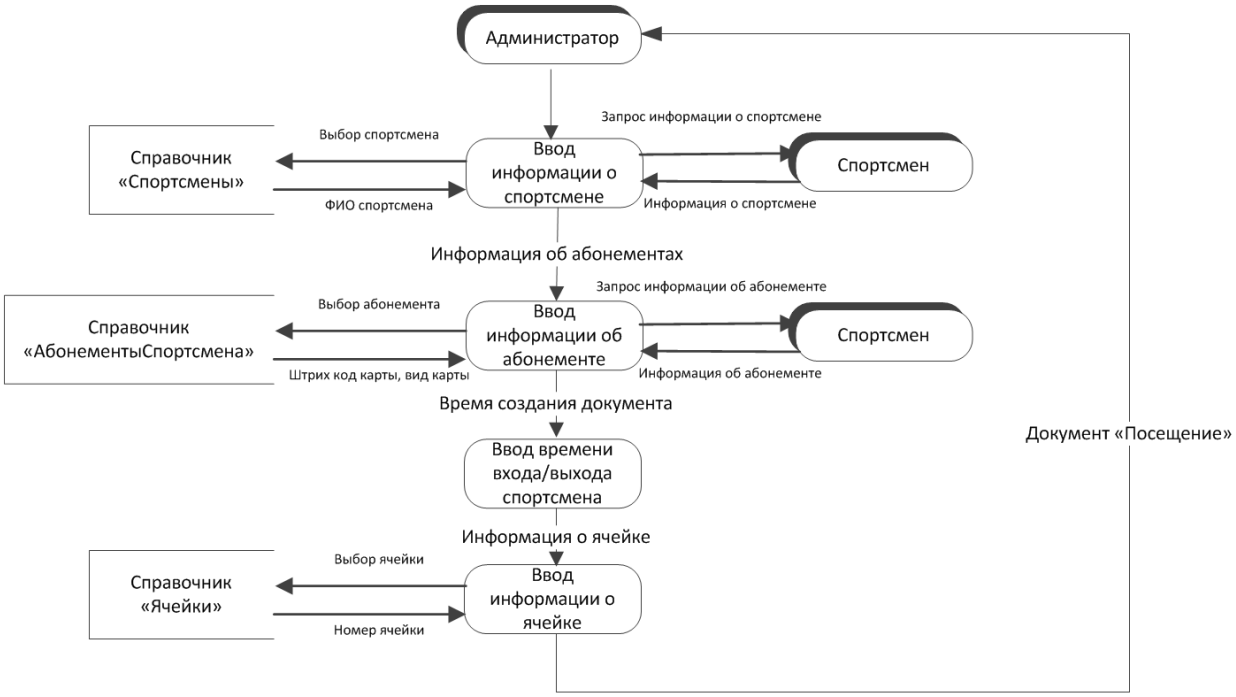


Рис. 33 Декомпозиция функции «Отметка посещения»

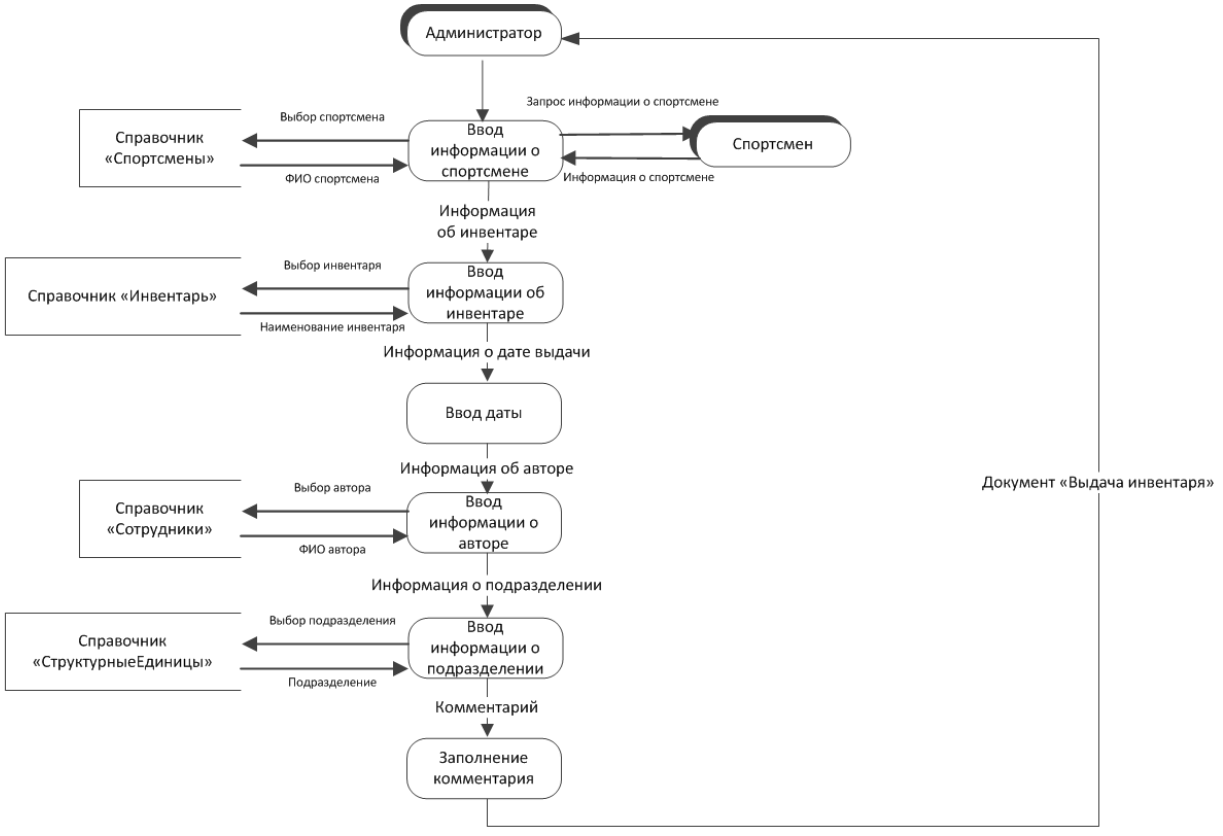


Рис. 34 Декомпозиция функции «Выдача инвентаря»

Создание занятия производится администратором в функциональном блоке Расписание и подтверждается документом *Планирование занятий* (рис.35). При этом необходимо:

- ввести дату и время проведения занятия;
- выбрать из справочника *Помещения* нужное помещение;
- выбрать группу, в которой планируется проведение занятия, из справочника *Группы*;
- из справочника Тренеры выбрать ответственного за проведение занятия.

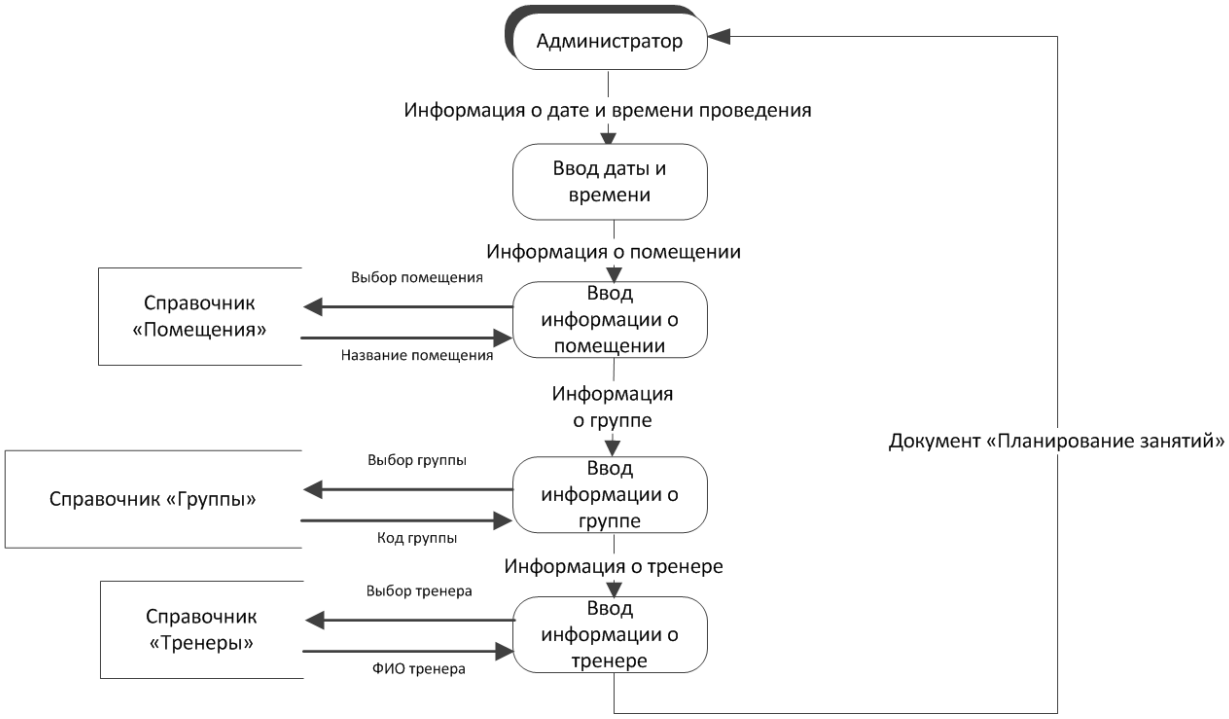


Рис. 35 Декомпозиция функции "Создание занятия"

Таким образом, разработанные диаграммы позволяют продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами.

2.4 Разработка объектов информационной системы

2.4.1 Создание справочников

Справочники – это прикладные объекты конфигурации. Они позволяют хранить в информационной базе данные, имеющие одинаковую структуру и списочный характер.

2.4.1.1 Справочник «Спортсмены»

Данный справочник содержит список спортсменов, посещающих спортивный клуб. Справочник содержит следующие обязательные атрибуты: *Контрагент* (тип СправочникСсылка.Контрагенты), *Комментарий* (тип Строка) – для ввода дополнительной информации о спортсменах, *Фото* (тип ХранилищеЗначений). И следующие необязательные: *ПервыйТренер* (тип СправочникСсылка.Тренеры), *Квалификация спортсмена* (тип СправочникСсылка.КвалификацияСпортсмена), *МедицинскийДопуск* (тип Строка), *СрокИстеченияМедицинскогоДопуска* (тип Дата), *Город* (тип Строка), *СпортивнаяОрганизация* (тип СправочникСсылка.Контрагент), *ВесоваяКатегория* (тип СправочникСсылка.ВесовыеКатегории).

2.4.1.2 Справочник «Тренеры»

Данный справочник содержит список тренеров, ведущих занятия в спортивном клубе. Справочник содержит следующие атрибуты: *Сотрудник* (тип СправочникСсылка.Сотрудники), *Фото* (тип ХранилищеЗначения).

Также есть две табличных части: *Услуги* с реквизитом *Услуга* (тип СправочникСсылка.Номенклатура) – информация о видах услуг, которые может оказывать тренер, и *Аттестации* с реквизитами *Аттестация* (тип Строка) и *ДатаПолучения* (тип Дата).

2.4.1.3 Справочник «Клубные карты»

Данный справочник является подчиненным, его родителем является справочник «Спортсмены». Справочник содержит два стандартных реквизита: *Код* (Штрихкод Клубной карты) и *Наименование*, которое в данном случае является ссылкой на спортсмена – владельца карты. Для этого необходимо произвести следующую процедуру:

```
&НаКлиенте
Процедура ПередЗаписью(Отказ, ПараметрыЗаписи)
Объект.Наименование = Объект.Владелец;
КонецПроцедуры
```

2.4.1.4 Справочник «Виды абонементов»

Справочник содержит следующие атрибуты: *ДлительностьОграничена* (тип Булево), *ДлительностьДней* (тип Число), *ДлительностьМесяцев* (тип Число), *ДлительностьЛет* (тип Число), *ДопустимыЗаморозки* (тип Булево), *КоличествоЗаморозок* (тип Число), *ДнейЗаморозки* (тип Число), *Групповой* (тип Булево). Также есть две табличных части: *Услуги* с реквизитами *Услуга* (тип СправочникСсылка.Номенклатура), *Количество* (тип Число), *Цена* (тип Число), *Сумма* (тип Число) и *ФилиалыВКоторыхДействует* с реквизитом *СтруктурнаяЕдиница* (тип СправочникСсылка.СтруктурныеЕдиницы).

2.4.1.5 Справочник «Абонементы спортсмена»

Данный справочник является подчиненным, его родителем является справочник «Спортсмены». Содержит следующие реквизиты: *Карта* (тип СправочникСсылка.КлубныеКарты), *ВидАбонемента* (СправочникСсылка.ВидыАбонементов).

2.4.1.6 Справочник «Группы»

Справочник содержит следующие реквизиты: *Услуга* (тип СправочникСсылка.Номенклатура), *ОсновнойТренер* (тип СправочникСсылка.Тренеры), *МаксимальныйРазмерГруппы* (тип Число), а также табличную часть *Расписание* с реквизитами *ДеньНедели* (тип Число), *ВремяНачала* (тип Дата), *Время окончания* (тип Дата).

2.4.1.7 Справочник «Весовые категории»

Справочник, помимо стандартных реквизитов (*Код* и *Наименование*), содержит реквизиты: *ВесОт* (тип Число), *ВесДо* (тип Число).

2.4.1.8 Справочник «Ячейки»

Справочник содержит только стандартные реквизиты (*Код* и *Наименование*).

2.4.1.9 Справочник «Помещения»

Справочник, помимо стандартных реквизитов (*Код* и *Наименование*), содержит: *Вместимость* (тип Число), *Подразделение* (СправочникСсылка.СтруктурныеЕдиницы) и табличную часть

ВидыАбонементов с реквизитом *ВидАбонемента* (тип СправочникСсылка.Абонементы).

2.4.2 Создание перечислений

Перечисления – это прикладные объекты конфигурации. Они позволяют хранить в информационной базе наборы значений, которые не изменяются в процессе работы прикладного решения.

2.4.2.1 Перечисление «Виды занятий»

Данное перечисление имеет два значения, которые определяют вид занятия: *Групповое* – для занятий в группах и *Индивидуальное* – для индивидуальных занятий.

2.4.2.2 Перечисление «Виды операций изменения абонемента»

Данное перечисление имеет несколько значений, которые определяют вид действия с абонементами: *Закрытие* – в случае окончания срока действия абонемента или использования количества занятий, определенных в абонементе. *Продление* – увеличение количества дней, в течение которого абонемент будет действителен (в случае непредвиденных обстоятельств). *Приостановление* – замораживание абонемента на несколько дней в случае, если спортсмен находится не в городе и не имеет возможности посещать тренировки. *Передача* – передача абонемента другому спортсмену в клубе. *Блокировка* – приостановление абонемента без возможности посещения занятий. *Возобновление* – происходит по возвращении спортсмена в спортклуб, если абонемент был приостановлен. *Разблокировка* - происходит по возвращении спортсмена в спортклуб, если абонемент был заблокирован.

2.4.3 Создание документов

Документы – это прикладные объекты конфигурации. Они позволяют хранить в прикладном решении информацию о совершенных хозяйственных операциях или о событиях, произошедших в "жизни" предприятия вообще.

2.4.3.1. Документ «Выдача инвентаря»

Документ необходим для регистрации информации о выдаче инвентаря. Содержит следующие реквизиты: *Автор* (тип

СправочникСсылка.Сотрудники) – ответственный за документ, *Спортсмен* (тип СправочникСсылка.Спортсмены) – тот, кому выдают инвентарь на занятие, *Структурная единица* (тип СправочникСсылка.СтруктурныеЕдиницы) – филиал, в котором происходит выдача инвентаря, и *Комментарий* (тип Строка) – для ввода дополнительной информации. Также документ имеет табличную часть *Инвентарь* с реквизитами *Инвентарь* (тип СправочникСсылка.Номенклатура) и его *Количество* (тип Число).

2.4.3.2 Документ «Посещение»

Документ необходим для фиксации посещений спортсмена: времени входа и выхода в клуб, а также информации о выдаче и получении ключа от ячейки. Содержит следующие реквизиты: *Автор* (тип СправочникСсылка.Сотрудники) – ответственный за документ, *Спортсмен* (тип СправочникСсылка.Спортсмены) – тот, кому выдают инвентарь на занятие, *Структурная единица* (тип СправочникСсылка.Спортсмены) – филиал, в котором происходит выдача инвентаря, *Комментарий* (тип Строка) – для ввода дополнительной информации, *Ячейка* (тип СправочникСсылка.Ячейки), *КлючВыдан* (тип Булево), *КлючПолучен* (тип Булево), *ДатаВремяВыхода* (тип Дата), *ДатаВремяВхода* (тип Дата).

Также в документе есть табличная часть *ВыданныйИнвентарь* с соответствующими реквизитами *Инвентарь* (тип СправочникСсылка.Номенклатура) и *Количество* (тип Число).

2.4.3.3 Документ «Планирование занятий»

Документ необходим для составления расписания. Используются следующие реквизиты: *Автор* (тип СправочникСсылка.Сотрудники) – ответственный за документ, *ВидЗанятия* (ПеречислениеСсылка.ВидыЗанятий), *ДатаНачала* (тип Дата), *ВремяНачала* (тип Дата), *ДатаОкончания* (тип Дата), *ВремяОкончания* (тип Дата), *Группа* (тип СправочникСсылка.Группы), *Комментарий* (тип Строка) – для ввода дополнительной информации, *Помещение* СправочникСсылка.Помещения),

Тренер (СправочникСсылка.Тренеры), *Услуга* (СправочникСсылка.Номенклатура).

Также есть следующие табличные части: *Спортсмены* с реквизитами *Спортсмен* (тип СправочникСсылка.Спортсмены), *Присутствовал* (тип Булево), *Абонемент* (тип СправочникСсылка.АбонементыСпортсменов) и *Расписание* с реквизитами *ДатаНачала* (тип Дата), *ВремяНачала* (тип Дата), *ВремяОкончания* (тип Дата), *Помещение* (тип СправочникСсылка.Помещения).

2.4.3.4 Документ «Изменение абонемента»

Документ необходим для проведения различных операций над абонементом. Содержит следующие реквизиты: *Автор*, *ВидОперации*, *Комментарий*. Также есть табличная часть *Абонементы* с реквизитами *Абонемент* (тип СправочникСсылка.АбонементыСпортсменов), *НачалоИзменения* (тип Дата), *ОкончаниеИзменения* (тип Дата), *АктивироватьПоОкончании* (тип Булево), *НовыйВладелец* (тип СправочникСсылка.Спортсмены), *Спортсмен* (тип СправочникСсылка.Спортсмены), *Группа* (тип СправочникСсылка.Группы).

2.4.4 Создание регистров

Регистры – это таблицы для накопления оперативных данных и получения сводной информации.

2.4.4.1 Регистры сведений

Регистры сведений – это прикладные объекты конфигурации. Они позволяют хранить в прикладном решении произвольные данные в разрезе нескольких измерений.

2.4.4.1.1 Регистр сведений «Доступность ячеек»

Регистр необходим для проверки ячеек от кабинок на доступность. Он имеет измерения: *Спортсмен* (тип СправочникСсылка.Спортсмены), *Ячейка* (тип СправочникСсылка.Ячейки), *Занятие* (тип ДокументСсылка.ПланированиеЗанятий), *Помещение* (тип

СправочникСсылка.Помещения), *Группа* (тип СправочникСсылка.Группы), *СтруктурнаяЕдиница* (тип СправочникСсылка.СтруктурныеЕдиницы).

Проверка на доступность ячеек в регистре осуществляется с помощью ресурса *Состояние* (тип Булево).

2.4.4.1.2 Регистр сведений «Посещения Спортсменов»

Регистр служит для регистрации посещения спортсмена и записи занятой им ячейки. Имеет следующие измерения: *ДатаВремяВхода* (тип Дата), *ДатаВремяВыхода* (тип Дата), *Организация* (тип СправочникСсылка.Организации), *Спортсмен* (тип СправочникСсылка.Спортсмены), *СтруктурнаяЕдиница* (тип СправочникСсылка.СтруктурныеЕдиницы).

Ресурсами регистра являются объекты: *Ячейка* (тип СправочникСсылка.Ячейки), *ДокументПосещения* (тип ДокументСсылка.Посещения) и *СостояниеСпортсмена* (тип ПеречислениеСсылка.СостоянияСпортсменов) – имеет значения ВКлубе, НеВКлубе, НаБольничном, НаМероприятии.

2.4.4.1.3 Регистр сведений «Расписание занятий»

Регистр необходим для планирования тренировок и имеет следующие измерения: *ВидЗанятия* (тип ПеречислениеСсылка.ВидыЗанятий), *ВремяНачала* (тип Дата), *ВремяОкончания* (тип Дата), *Группа* (тип СправочникСсылка.Группы), *ДатаНачала* (тип Дата), *ДатаОкончания* (тип Дата), *Организация* (тип СправочникСсылка.Организации), *Помещение* (тип СправочникСсылка.Помещения), *СостояниеОбъекта* (тип Строка) – определяется, планируемое это занятие или проведенное, *Спортсмен* (тип СправочникСсылка.Спортсмены), *СтруктурнаяЕдиница* (тип СправочникСсылка.СтруктурныеЕдиницы), *ТипОбъекта* (тип ПеречислениеСсылка.ТипыОбъектовРасписания) – имеет значения Занятие и Мероприятие, *Тренер* (тип СправочникСсылка.Тренеры), *Услуга* (тип СправочникСсылка.Номенклатура).

2.4.4.1.4 Регистр сведений «Составы групп»

Регистр служит для определения состава групп и их состояния. Имеет следующие измерения: *Группа* (тип СправочникСсылка.Группы), *Спортсмен* (тип СправочникСсылка.Спортсмены), *Абонемент* (тип СправочникСсылка.АбонементыСпортсменов). Состояние групп определяется ресурсом *Активен* (тип Булево).

2.4.4.1.5 Регистр сведений «Состояния абонементов»

Регистр служит для определения состояния абонемента и определения срока его окончания. Имеет измерение *Абонемент* (тип СправочникСсылка.АбонементыСпортсменов) и ресурсы: *СостояниеАбонемента* (тип ПеречислениеСсылка.СостоянияАбонементов), *ДатаОграничения* (тип Дата), *СрокОкончания* (тип Дата).

2.4.4.1.6 Регистр сведений «Цены абонементов»

Регистр служит для определения цен на различные виды абонементов и услуг. Имеет следующие измерения: *ВидАбонемента* (тип СправочникСсылка.ВидыАбонементов), *Организация* (тип СправочникСсылка.Организации), *Абонемент* (тип СправочникСсылка.АбонементыСпортсменов), *Услуга* (тип СправочникСсылка.Номенклатура). Для определения цен на номенклатуру существует два ресурса: *Цена* (тип Число) и *ЦенаЗанятия* (тип Число).

2.4.4.1.7 Регистр сведений «Тренеры групп»

Регистр необходим для распределения каждой группы своему тренеру. Имеет измерение *Группа* (тип СправочникСсылка.Группы) и ресурс *Тренер* (тип СправочникСсылка.Тренеры).

2.4.4.2 Регистры накоплений

Регистры накопления – это прикладные объекты конфигурации. Они составляют основу механизма учета движения средств (финансов, товаров, материалов и т.д.), который позволяет автоматизировать такие направления, как складской учет, взаиморасчеты, планирование.

2.4.4.2.1 Регистр накоплений «Занятия по абонементам»

Регистр служит для фиксации количества занятий в абонементе, оставшихся для посещения. Имеет следующие измерения: *Абонемент* (тип СправочникСсылка.АбонементыСпортсменов), *Спортсмен* (тип СправочникСсылка.Спортсмены), *Услуга* (СправочникСсылка.Номенклатура), *Тренер* (СправочникСсылка.Тренеры), *Организация* (тип СправочникСсылка.Организации), ДокументПродажи (имеет составной тип: ДокументСсылка.ПродажаАбонемента, ДокументСсылка.ВводНачальныхОстатковСпортклуба), *ДатаОкончания* (тип Дата). Ресурс для определения количества оставшихся занятий: *Количество* (тип Число). Также регистр накопления имеет реквизит *ВидНачисленияСписанияЗанятий* – имеет значения Продажа, ПроведенноеЗанятие, СгоревшееЗанятие, Корректировка (тип ПеречислениеСсылка.ВидыНачисленийСписанийЗанятий).

2.4.4.2.2 Регистр накоплений «Учет приостановок абонемента»

Регистр необходим для подсчета количества приостановок абонемента. Имеет измерение *Абонемент* (тип СправочникСсылка.АбонементыСпортсменов) и ресурс *КоличествоПриостановок* (тип Число).

2.4.5 Создание обработок

Обработки – это прикладные объекты конфигурации. Они предназначены для выполнения различных действий над информацией.

2.4.5.1 Обработка «АРМ Администратора»

Данная обработка представляет собой автоматизированное рабочее место администратора. Содержит следующие реквизиты: *Вид* (тип Число), *ДатаНачала* (тип Дата), ДатаОкончания (тип Дата), *Помещение* (тип СправочникСсылка.Помещения), *Тренер* (тип СправочникСсылка.Тренеры), *Спортсмен* (тип СправочникСсылка.Спортсмены), *Абонемент* (тип СправочникСсылка.АбонементыСпортсменов). Табличная часть *Абонементы* содержит информацию об абонементах выбранного спортсмена

и имеет следующие реквизиты: *АбонементСпортсмена* (СправочникСсылка.АбонементыСпортсмена), *Состояние* (тип ПеречислениеСсылка.СостоянияАбонементов), ДатаОграничения (тип Дата), ЗанятийОсталось (тип Число), СрокДействияАбонемента (тип Строка).

2.4.5.2 Обработка «АРМ Тренера»

Данная обработка представляет собой автоматизированное рабочее место тренера. Содержит следующие реквизиты: *Группа* (тип СправочникСсылка.Группы), *ДатаВремяПроведения* (тип Дата), *Помещение* (тип СправочникСсылка.Помещения), *Тренер* (тип СправочникСсылка.Тренеры), *Услуга* (тип СправочникСсылка.Номенклатура), *ВидЗанятий* (тип ПеречислениеСсылка.ВидыЗанятий), *ВремяНачала* (тип Дата), *ВремяОкончания* (тип Дата), *Занятие* (тип ДокументСсылка.ПланированиеЗанятий).

АРМ Тренера содержит две табличные части. Первая табличная часть *ПосетителиЗанятия* отображает спортсменов, пришедших на тренировку. Содержит следующие реквизиты: *Отметка* (тип Булево), *Спортсмен* (тип СправочникСсылка.Спортсмены), *ДоступностьДляРедактирования* (тип Булево), *Абонемент* (тип СправочникСсылка.АбонементыСпортсменов), *ДатаРождения* (тип Дата), *ЗанятийОсталось* (тип Число), *СрокДействияАбонемента* (тип Дата), *СтатусЗанятия* (тип ПеречислениеСсылка.СтатусыЗанятий), *СостояниеАбонемента* (тип ПеречислениеСсылка.СостоянияАбонементов.

Вторая табличная часть *РасписаниеЗанятий* служит для просмотра тренером расписания и содержит следующие реквизиты: *ВидЗанятий* (тип ПеречислениеСсылка.ВидыЗанятий), *Услуга* (тип СправочникСсылка.Номенклатура), *ВремяНачала* (тип Дата), ВремяОкончания (тип Дата), *Группа* (тип СправочникСсылка.Группы), *Помещение* (тип СправочникСсылка.Помещения), *Занятие* (тип ДокументСсылка.ПланированиеЗанятий), *ДатаЗанятия* (тип Дата).

2.5 Диаграмма сущность-связь

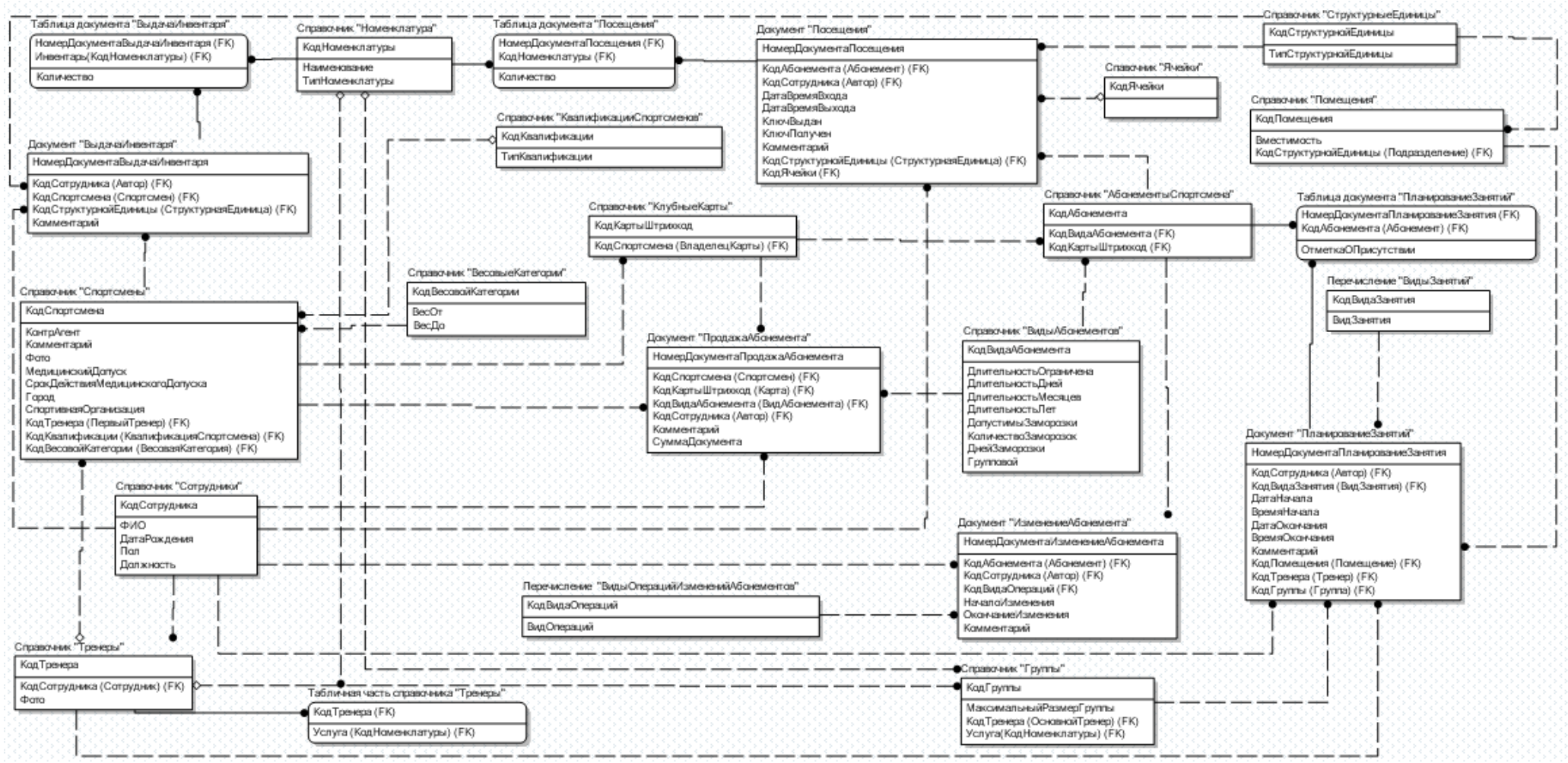


Рис. 36 Диаграмма сущность-связь

3 Руководство пользователя информационной системы

3.1 АРМ Администратора

АРМ администратора имеет следующий интерфейс (рис. 37):

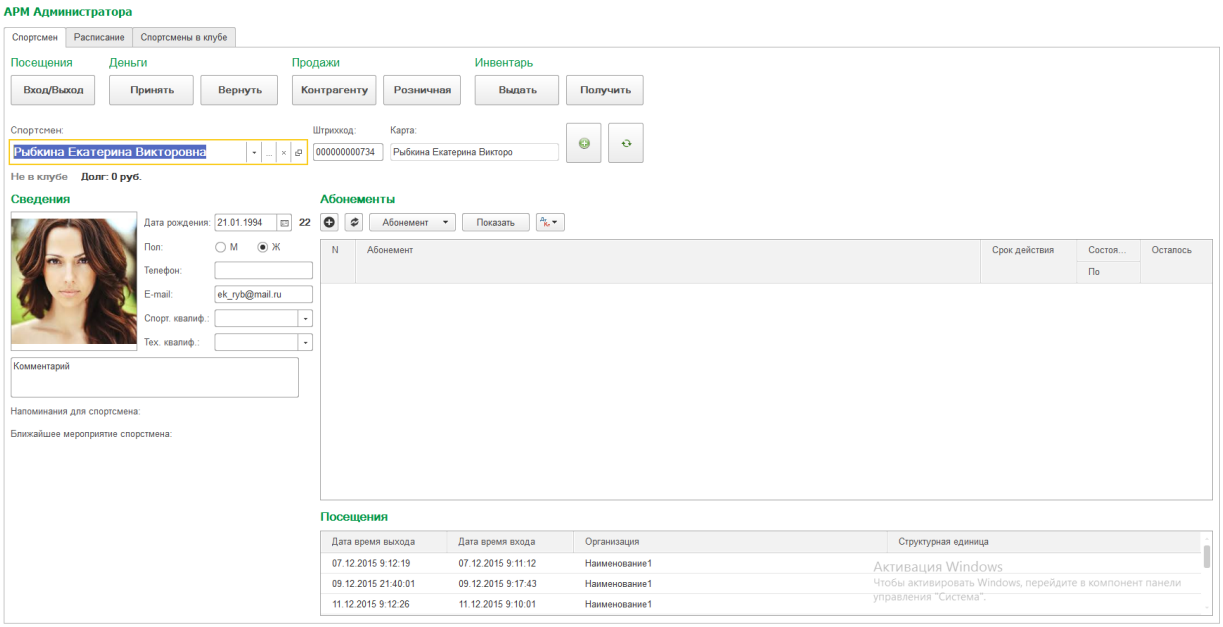


Рис. 37 Рабочее место администратора

Сверху расположены вкладки «Спортсмен», «Расписание» и «Спортсмены в клубе».

Интерфейс, представленный на первой вкладке «Спортсмен» позволяет администратору фиксировать посещения спортсменов, просматривать базовую информацию о спортсмене и изменять ее, управлять абонементами, выдавать необходимый инвентарь.

При создании спортсмена открывается форма *Спортсмены (создание)*. Во вкладке *Основные сведения* в соответствующих полях администратором заполняется информация о новом спортсмене (рис. 38): ФИО, пол, дата рождения и фотография (необязательно). Код спортсмена заполняется системой автоматически. А долг у нового спортсмена автоматически ставится 0 руб.

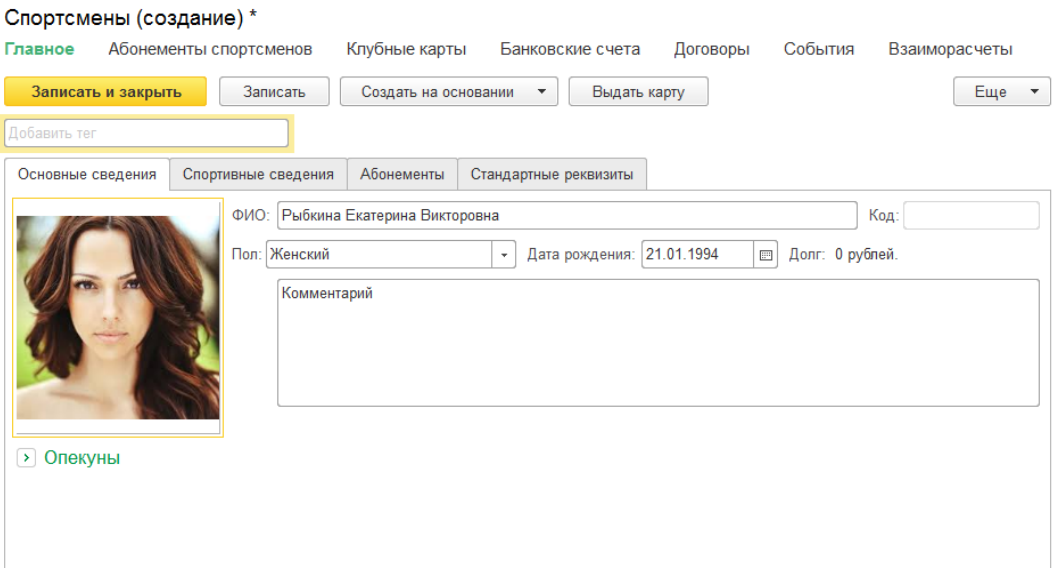


Рис. 38 Создание спортсмена

Во вкладке *Спортивные сведения* (рис. 39) добавляется информация о первом тренере (если имеется), квалификация спортсмена, медицинский допуск и срок его истечения, город и спортивная организация. Также добавляется основной тренер.

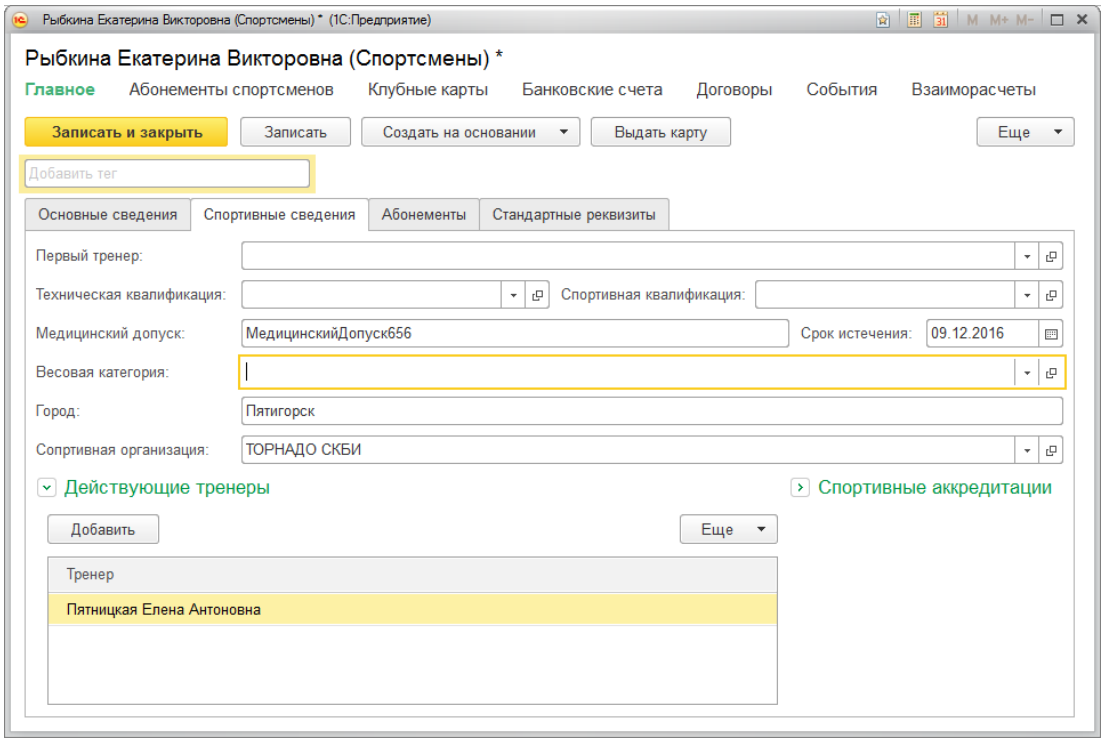


Рис. 39 Спортивные сведения спортсмена

Во вкладке *Абонементы* (рис. 40) отображаются действующие абонементы спортсмена и их состояние (активен либо заморожен). Также на

этой форме есть возможность продать еще один абонемент, нажав кнопку *Продать*.

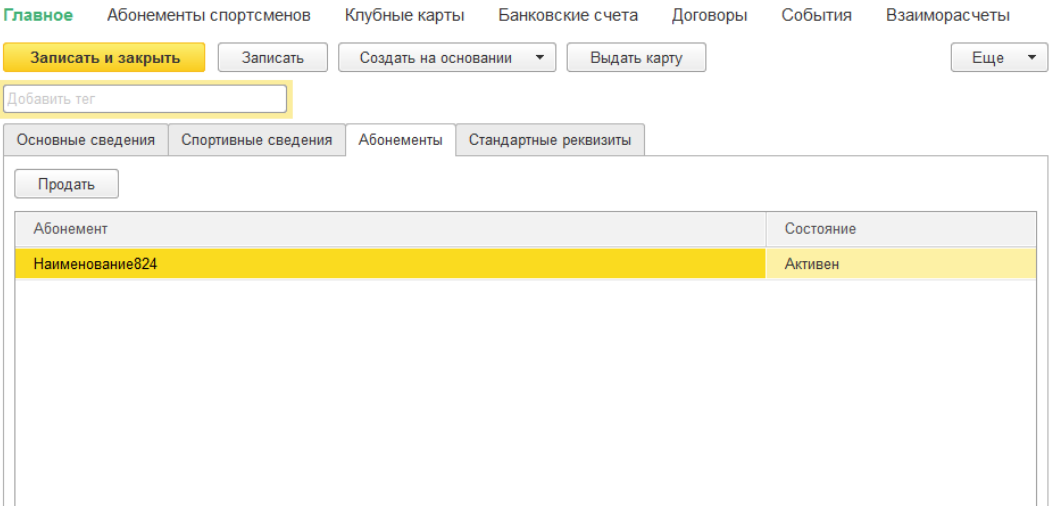


Рис. 40 Действующие абонементы спортсмена

У существующего спортсмена можно просмотреть все купленные им абонементы во вкладке *Абонементы спортсмена* (рис. 41).

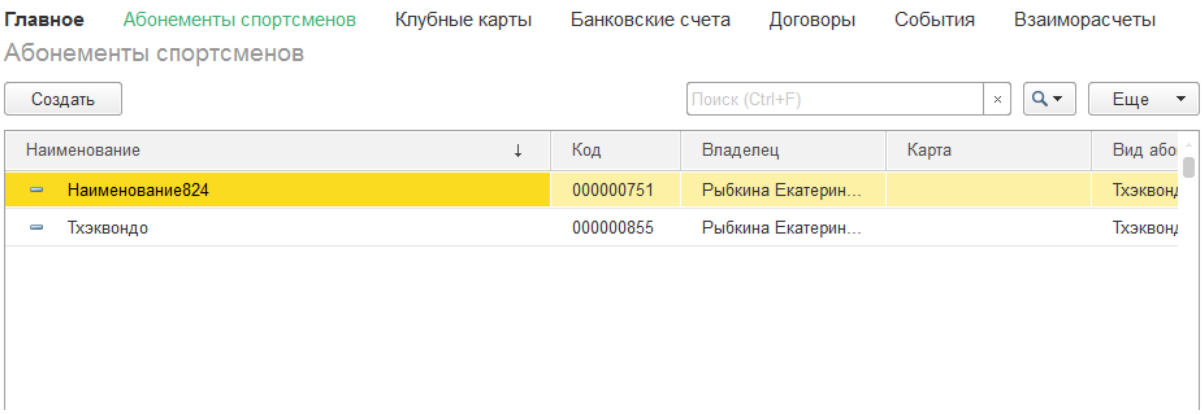


Рис. 41 Абонементы спортсмена

Также у существующего спортсмена во вкладке *Клубные карты* отображаются карты, которые были приобретены (рис. 42).

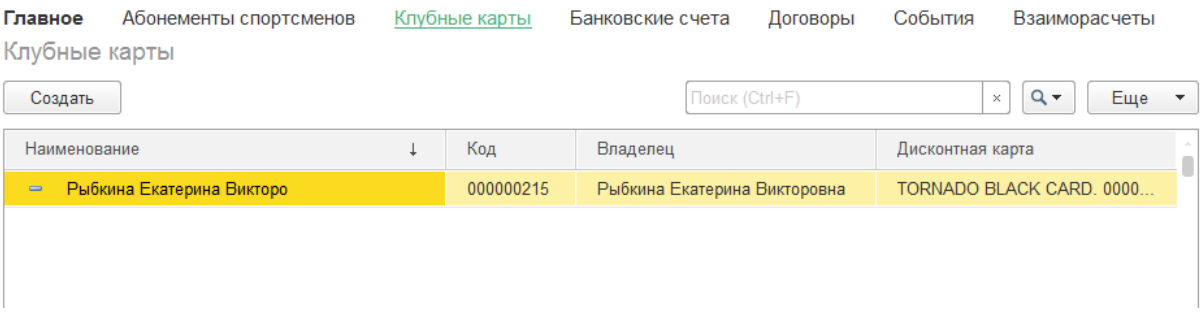


Рис. 42 Клубные карты спортсмена

Интерфейс, представленный на второй вкладке «Расписание» позволяет просматривать расписание тренера в разрезе дня, недели или месяца.

Интерфейс, представленный на третьей вкладке «Спортсмены в клубе» предоставляет информацию обо всех спортсменах, находящихся в клубе на данный момент.

Посещение спортсменом спортклуба

Для того чтобы зафиксировать вход/выход спортсмена, необходимо считать штрих-код с карты либо ввести его самостоятельно в поле *Штрих-код.* После этого в поле *Карта спортсмена* отобразится ФИО держателя карты, в поле *Спортсмен* – ФИО спортсмена. Раздел *Инфо* используется для отображения дополнительной информации и фото. В разделе *Абонементы* содержится информация об активных абонементах: номер, состояние, срок действия и количество оставшихся занятий. При выборе абонемента из списка в разделе *Посещения* отображается информация о посещенных занятиях по выбранному абонементу (дата и время входа и выхода, организация и структурная единица). Если в данных момент этого спортсмена в клубе нет, система предложит зафиксировать вход (рис. 43).

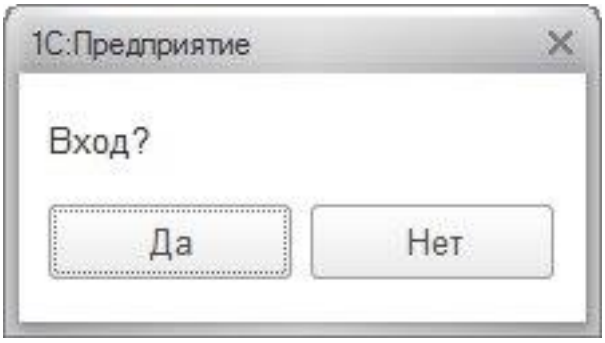


Рис. 43 Предложение зарегистрировать вход спортсмена

При нажатии на кнопку *Да* открывается форма создания посещения (рис. 44). Поля *Спортсмен* и *Абонемент* заполняются автоматически из базы. В поле *Вход* заносится текущее значение даты и времени.

Администратор имеет возможность поставить отметку о выданном ключе и указать номер ячейки в соответствующем поле. Для того, чтобы

зафиксировать вход, необходимо нажать кнопку *Провести* либо *Провести и закрыть*, если работа с формой закончена. При повторном считывании/вводе штрих-кода, т.е. если вход спортсмена был зарегистрирован, система предлагает зарегистрировать выход, при этом в поде *Выход* проставляются дата и время выхода. Если был выдан ключ, необходимо поставить отметку *Ключ получен.* В противном случае программа не позволит провести документ.

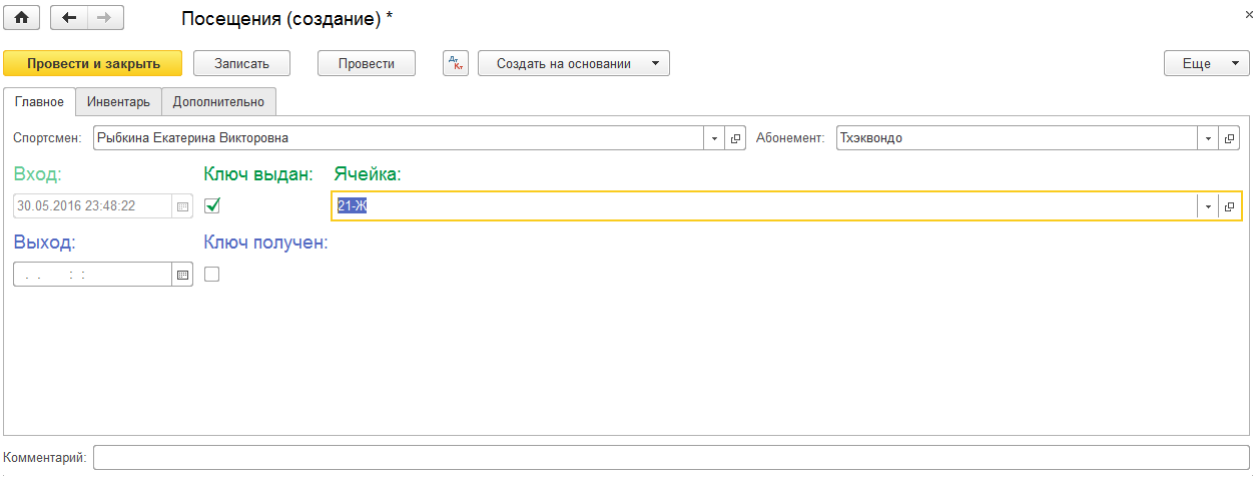


Рис. 44 Форма создания посещения

Продажа абонемента

Администратор может продать абонемент. Для этого в разделе *Абонементы* необходимо нажать на кнопку . На экране отобразится форма продажи абонемента (рис. 45). Необходимо заполнить поля *Спортсмен*, *Номер*, *Дата* (заполняется автоматически), *Карта* (заполнится автоматически, если у спортсмена она уже есть), *Организация* и *Плательщик*. Система располагает возможностью продажи сразу нескольких абонементов. Для этого необходимо заполнить раздел *Абонементы*. Для каждого необходимо указать вид, спортсмена, стоимость, информацию о заморозках, длительности и сроках абонемента и лицевой счет для зачисления бонусов; сумма зачисления рассчитывается автоматически. Далее нужно указать услуги, которыми планирует пользоваться спортсмен. Для этого нужно внести номер группы, тренера и количество. Поля *Ответственный* и *Автор* по умолчанию заполнены значением *Администратор*. При необходимости

их можно заменить. В конце формы в поле *Комментарий* может быть введена любая дополнительная информация. Чтобы осуществить продажу, нужно нажать на кнопку *Провести* или *Провести и закрыть*, если работа с формой закончена.

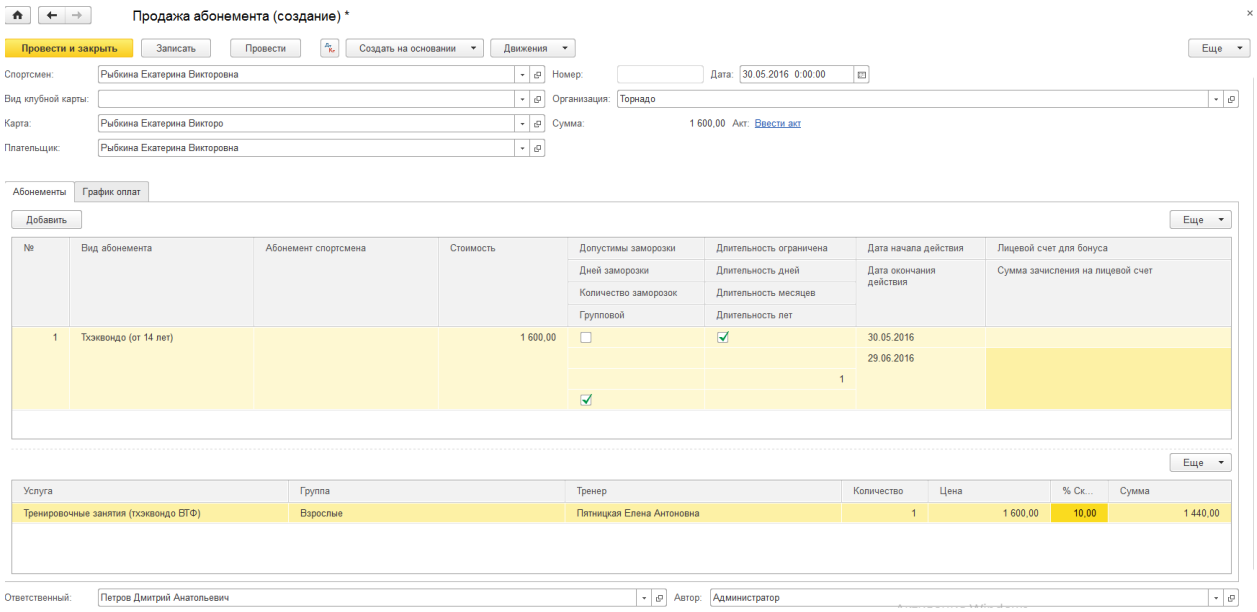


Рис. 45 Форма продажи абонемента

Планирование занятий

Администратор имеет возможность планирования занятий для тренера. Для создания нового занятия нужно выделить любую область в расписании и, вызвав контекстное меню правой кнопкой мыши, нажать *Создать -> Групповое занятие*, как показано на рис. 46.

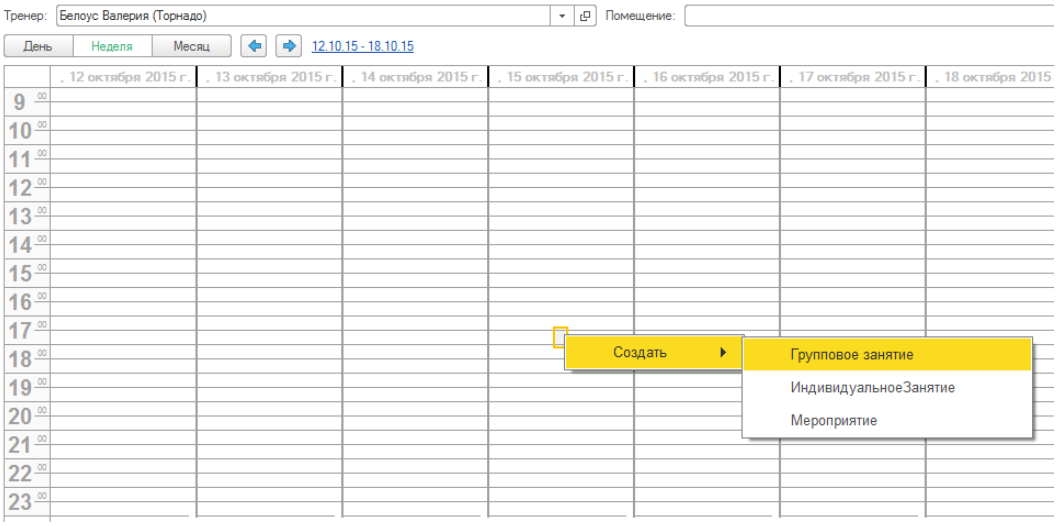


Рис. 46 Создание группового занятия у тренера

Далее открывается форма создания группового занятия (рис. 4).

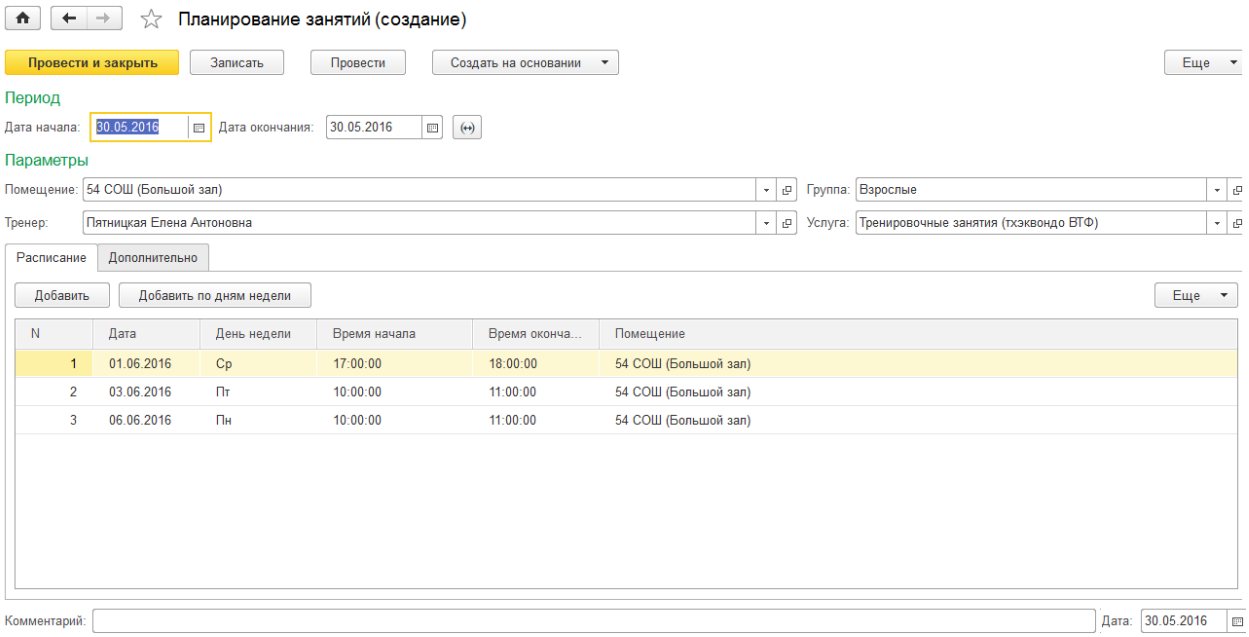


Рис. 47 Форма создания группового занятия

В первую очередь, вводится дата начала и окончания занятия (по умолчанию ставится текущая дата). Далее заполняются параметры занятий. В поле «Помещение» выбирается уже существующее название помещения из списка, либо добавляется новое. Это помещение, в котором будет проходить занятие. В поле «Группа» вводится название группы, для которой составляется занятие. Группа также выбирается из списка, либо создается новая. В поле «Тренер» вводится ФИО тренера, который будет проводить занятие. ФИО тренера выбирается из списка, либо создается новое.

В области «Расписание» есть две кнопки «Добавить» и «Добавить по дням недели». Нажав кнопку «Добавить», появляется новая строка для создания расписания занятия (рис. 48).

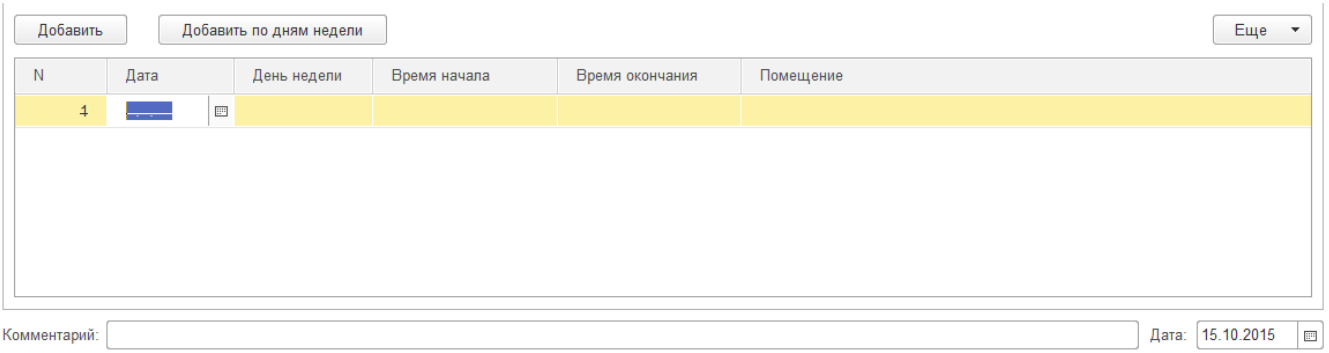


Рис. 48 Создание расписания

В ячейку «Дата» вводится дата занятия, автоматически высвечивается ячейка «День недели». В ячейки «Время начала» и «Время окончания» вводится соответствующее время занятия. В последней ячейке выбирается помещение, в которой будет проводиться занятие. При заполнении ячеек система предлагает скорректировать расписание на последующие дни (рис. 49). При нажатии «Да» таблица обновляется в соответствии с введенными данными. При желании можно записать комментарий внизу формы.

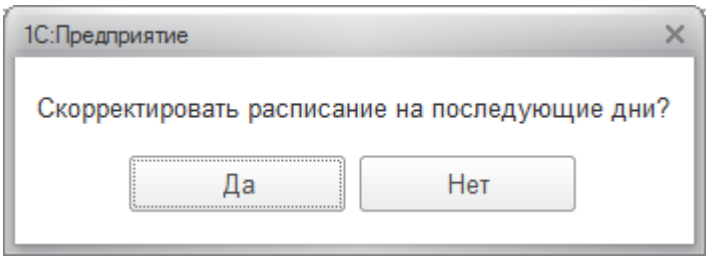


Рис. 49 Предложение системы скорректировать расписание

При нажатии кнопки «Добавить по дням недели» появляется другая форма (рис. 50).

Тогда все данные, введенные в форме создания группового занятия (Помещение, Тренер, Группа, Услуга) будут перенесены на эту форму. Администратор может указать период занятий, дни недели и время соответственно, когда будут проводиться занятия.

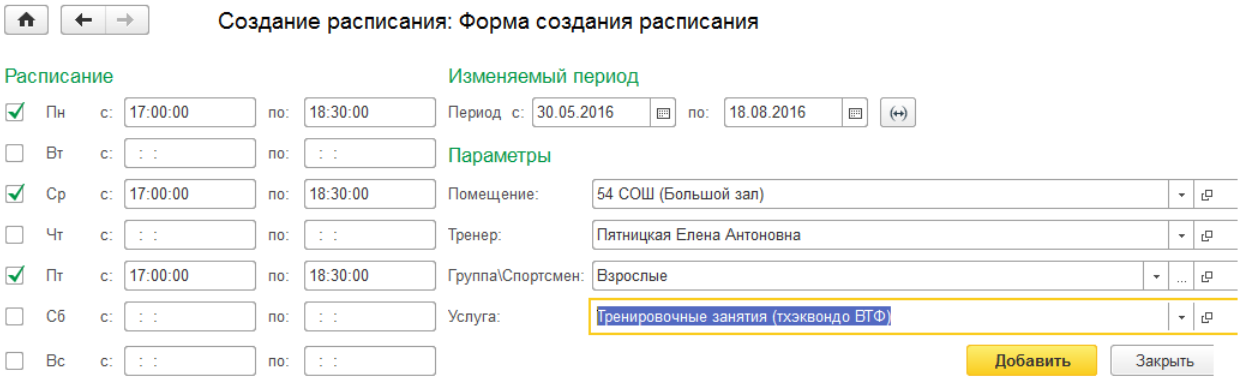


Рис. 50 Форма создания расписания

3.2 АРМ тренера

Для открытия автоматизированного рабочего места тренера необходимо нажать вкладку «Спортклуб» и выбрать «АРМ тренера» (рис. 51).

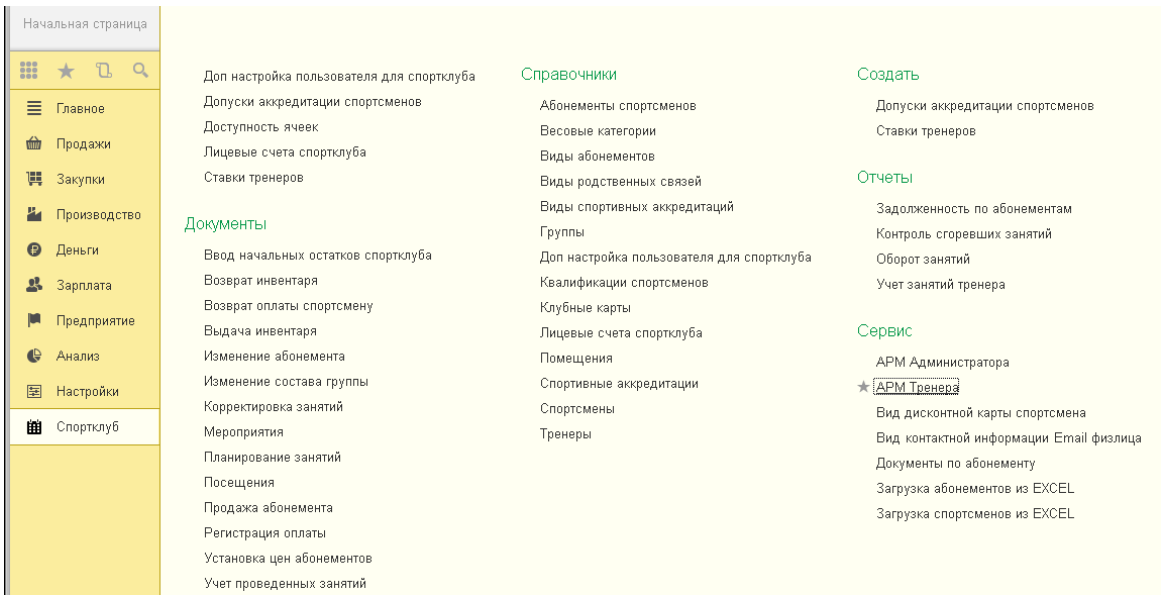


Рис. 51 Выбор рабочего места тренера

Для проведения переклички необходимо:

1. Выбрать тренера, проводящего занятие, из списка либо самостоятельно ввести его имя (рис. 52).

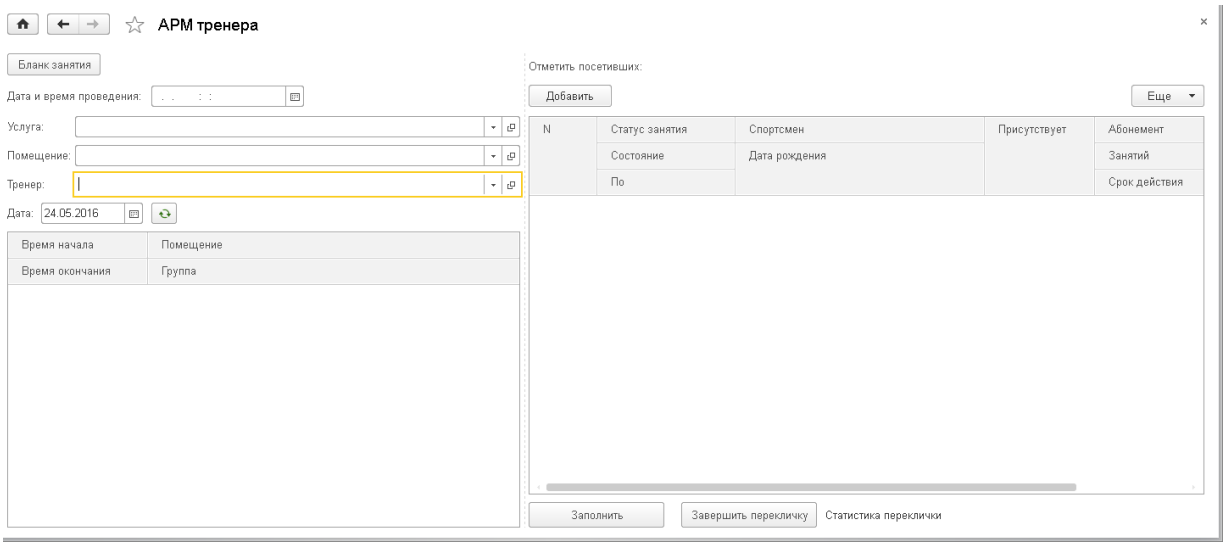


Рис. 52 Выбор тренера

При этом автоматически будут заполнены поля с названием услуги и помещения, в котором она проводится (рис. 53).

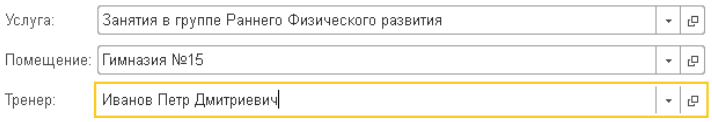


Рис. 53 Информация об услуге

В таблице слева появится информация о времени начала и окончания занятия, помещении и номере группы (рис. 54).

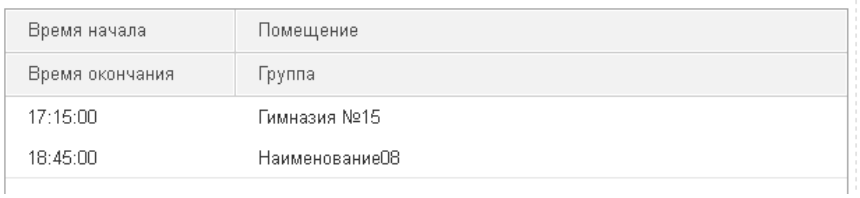


Рис. 54 Информация о занятии

В центре появится список спортсменов, входящих в группу, с информацией о каждом: ФИО спортсмена, дата рождения спортсмена, вид абонемента, срок действия абонемента, количество оставшихся занятий и статус занятия (рис. 55).

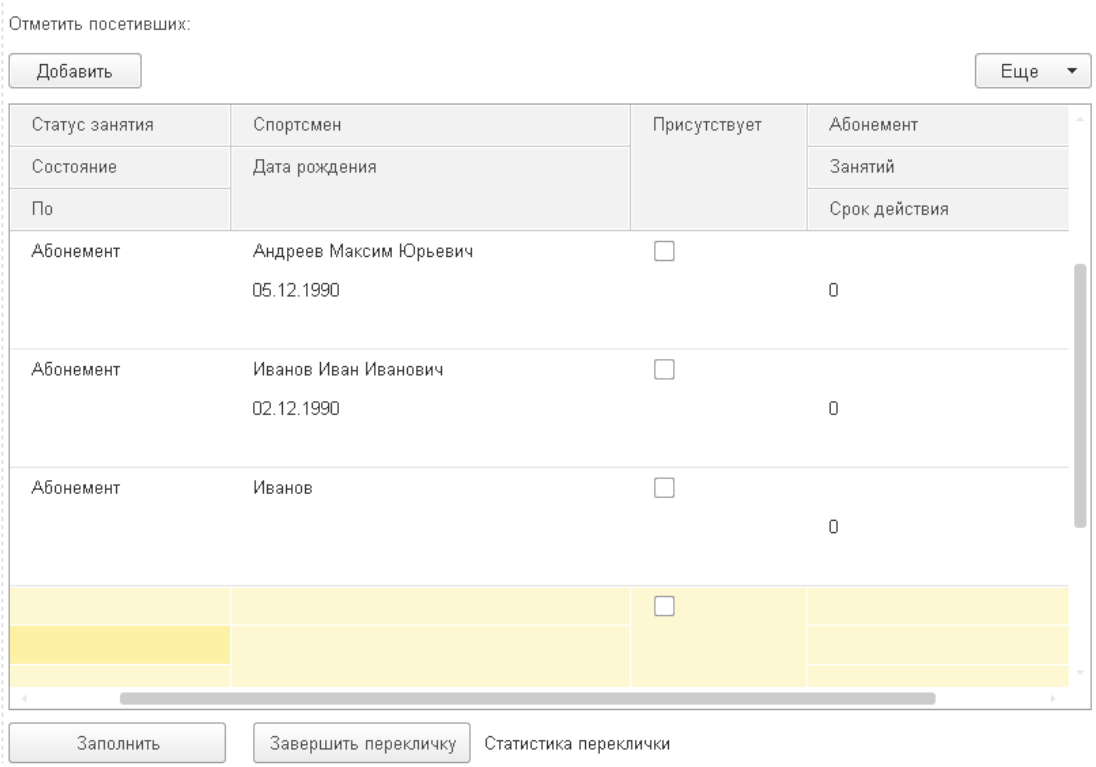


Рис. 55 Информация о спортсменах

2. При присутствии спортсмена на занятии тренеру необходимо поставить галочку напротив фамилии спортсмена в графе «Присутствует».

При этом если спортсмен не был отмечен на входе у администратора, данная функция будет недоступна (рис. 56).

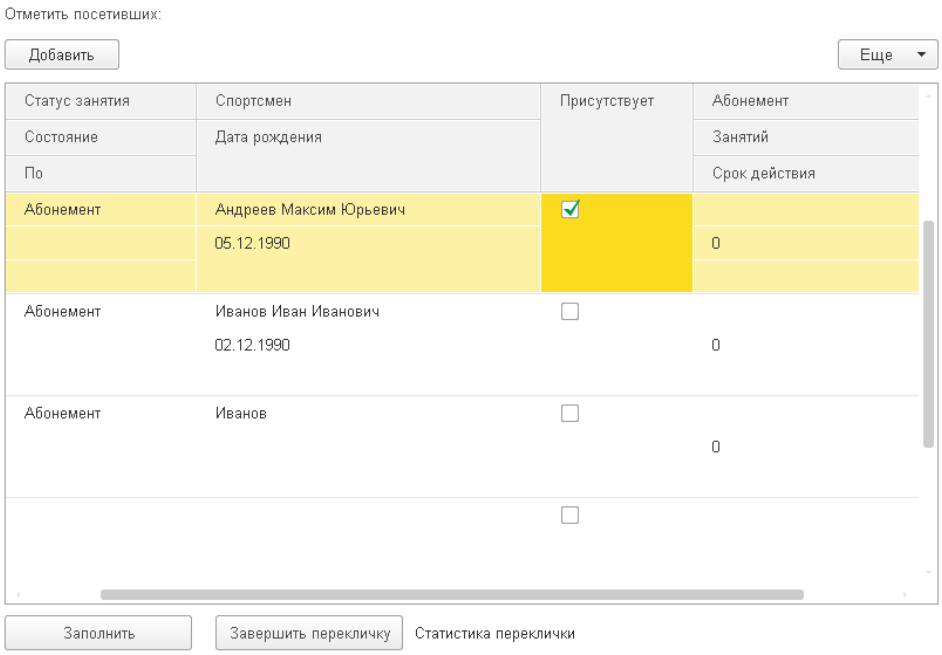


Рис. 56 Присутствие спортсмена на занятии

3. После того, как все присутствующие спортсмены отмечены, необходимо нажать на кнопку «Завершить перекличку». После этого появится сообщение, что перекличка завершена (рис. 57).

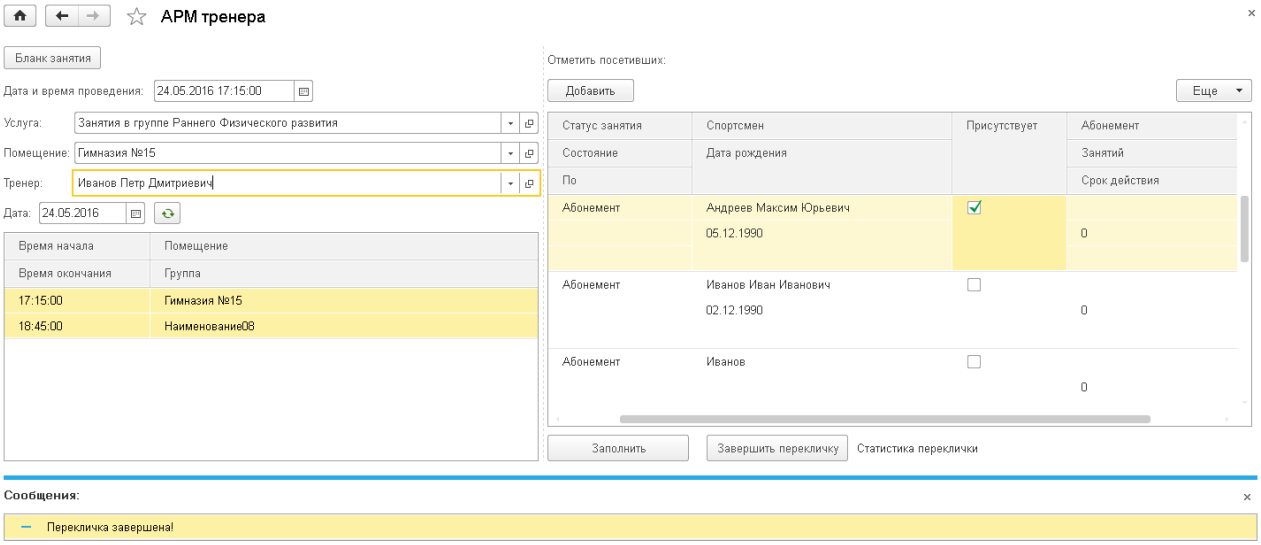


Рис. 57 Завершение переклички

4 Финансовый менеджмент, ресурсоэффективность и ресурсосбережение

4.1 Оценка коммерческого потенциала и перспективности проведения научных исследований с позиции ресурсоэффективности и ресурсосбережения

4.1.1 Потенциальные потребители результатов исследования

Для анализа потребителей результатов исследования необходимо рассмотреть целевой рынок и провести его сегментирование.

Целевой рынок – сегменты рынка, на котором будет продаваться в будущем разработка.

Сегментирование – это разделение покупателей на однородные группы, для каждой из которых может потребоваться определенный товар.

В зависимости от категории потребителей (коммерческие организации, физические лица) необходимо использовать соответствующие критерии сегментирования. В данном случае были выбраны критерии местоположения компании и отрасль, в которой она осуществляет деятельность (таблица 1). На текущий момент предприятие реализует себя в основном на местном рынке разработки программных продуктов для сферы услуг и торговли. В дальнейшем оно намерено расширять как границы своих проектов, так и осваивать новые отрасли. В будущем наиболее привлекательным и перспективным представляется сотрудничество с международными компаниями в сферах производства и логистики.

Таблица 1. *Сегментирование рынка*

<table>
<tr><td colspan="2"></td><td colspan="4">Отрасль компании</td></tr>
<tr><td colspan="2"></td><td>Сфера услуг</td><td>Торговля</td><td>Производство</td><td>Логистика</td></tr>
<tr><td rowspan="4">Местоположение компании</td><td>Международные</td><td></td><td></td><td colspan="2">Будущие перспективы</td></tr>
<tr><td>Отечественные</td><td colspan="4">Направление развития</td></tr>
<tr><td>Региональные</td><td colspan="2">Направление развития</td><td></td><td></td></tr>
<tr><td>Местные</td><td colspan="2">Текущее положение</td><td></td><td></td></tr>
</table>

4.1.2 Анализ конкурентных технических решений

Детальный анализ конкурирующих разработок, существующих на рынке, необходимо проводить систематически, поскольку рынки пребывают в постоянном движении. Такой анализ помогает вносить коррективы в научное исследование, чтобы успешнее противостоять своим соперникам.

Анализ конкурентных технических решений с позиции ресурсоэффективности и ресурсосбережения позволяет провести оценку сравнительной эффективности научной разработки и определить направления для ее будущего повышения.

Для этого была использована оценочная карта (таблица 2).

Таблица 2. *Оценочная карта для сравнения конкурентных технических решений*

Критерии оценки	**Вес крите-рия**	**Баллы**			**Конкуренто-способность**		
		$Б_ф$	$Б_{к1}$	$Б_{к2}$	$К_ф$	$К_{к1}$	$К_{к2}$
1	2	3	4	5	6	7	8
Технические критерии оценки ресурсоэффективности							
1. Удобство в эксплуатации	0,1	5	4	4	0,5	0,4	0,4
2. Надежность	0,1	4	5	4	0,4	0,5	0,4
3. Потребность в ресурсах памяти	0,05	4	4	3	0,2	0,2	0,15
4. Возможности программного продукта	0,15	5	4	5	0,75	0,6	0,6
5. Простота эксплуатации	0,1	5	4	4	0,5	0,4	0,4
6. Качество интерфейса	0,15	4	3	5	0,6	0,45	0,75
Экономические критерии оценки эффективности							
1. Конкурентоспособность программного продукта	0,1	4	4	3	0,4	0,4	0,3
2. Цена программного продукта	0,1	5	4	5	0,5	0,4	0,5
3. Сервисное обслуживание	0,1	4	2	3	0,4	0,2	0,3
4. Наличие сертификации используемой разработки	0,05	5	5	5	0,25	0,25	0,25
Итого	**1**				**4,7**	**3,8**	**4,05**

Для оценки ресурсоэффективности были выбраны следующие критерии: удобство в эксплуатации, надежность, потребность в ресурсах памяти, функциональная мощность, простота эксплуатации, качество интеллектуального интерфейса. Наиболее значимые среди них

функциональная мощность и качество интеллектуального интерфейса. Ведь клиент покупает программный продукт именно из-за предоставляемых возможностей, а пользователями являются люди без специальной подготовки, поэтому интеллектуальный интерфейс, связывающий ЭВМ и человека, имеет решающее значение и обязан быть качественным.

Для оценки эффективности были выбраны следующие экономические критерии: конкурентоспособность продукта, цена, послепродажное обслуживание, наличие сертификации разработки.

Результаты анализа выявили, что созданный программный продукт выгодно отличается от конкурентов. И одним из конкурентных преимуществ является внимание и забота о потребителе: продукт более удобен и прост в эксплуатации, предоставляет полный спектр функциональных возможностей. Также целью является не только продажа программного продукта, но и дальнейшее послепродажное обслуживание. Кроме того, предприятие отличается доступной ценовой политикой. Именно эти свойства помогут заинтересовать и завоевать доверие покупателей.

4.1.3 Технология QuaD

Технология QuaD (QUality ADvisor) представляет собой гибкий инструмент измерения характеристик, описывающих качество новой разработки и ее перспективность на рынке и позволяющие принимать решение целесообразности вложения денежных средств в научно-исследовательский проект.

В основе технологии QuaD лежит нахождение средневзвешенной величины следующих групп показателей: показателей оценки коммерческого потенциала разработки и показателей оценки качества разработки.

Анализ проводится в виде оценочной карты (таблица 3).

Таблица 3. *Оценочная карта QuaD*

Критерии оценки	**Вес критерия**	**Баллы**	**Макси-маль-ный балл**	**Относитель-ное значение** (3/4)	**Средневзвешен-ное значение** (5x2)
1	2	3	4	5	

Показатели оценки качества разработки					
1. Удобство в эксплуатации	0,1	90	100	0,9	0,09
2. Надежность	0,1	70	100	0,7	0,07
3. Потребность в ресурсах памяти	0,05	70	100	0,7	0,035
4. Возможности программного продукта	0,15	90	100	0,9	0,135
5. Простота эксплуатации	0,1	95	100	0,95	0,095
6. Качество интерфейса	0,15	85	100	0,85	0,1275
Показатели оценки коммерческого потенциала разработки					
1. Конкурентоспо-собность программного продукта	0,1	80	100	0,8	0,08
2. Цена программного продукта	0,1	95	100	0,95	0,095
3. Сервисное обслуживание	0,1	85	100	0,85	0,085
4. Наличие сертификации используемой разработки	0,05	100	100	1	0,05
Итого	**1**				**0,8625**

По итогам анализа можно отметить, что разработка является перспективной и в нее стоит инвестировать.

4.1.4 SWOT-анализ

SWOT – Strengths (сильные стороны), Weaknesses (слабые стороны), Opportunities (возможности) и Threats (угрозы) – представляет собой комплексный анализ научно-исследовательского проекта. SWOT-анализ применяют для исследования внешней и внутренней среды проекта.

Первый этап заключается в описании сильных и слабых сторон проекта, в выявлении возможностей и угроз для реализации проекта, которые проявились или могут появиться в его внешней среде.

Сильные стороны – это ресурсы или возможности, которыми располагает руководство проекта и которые могут быть эффективно использованы для достижения поставленных целей.

Слабые стороны – это то, что плохо получается в рамках проекта или где он располагает недостаточными возможностями или ресурсами по сравнению с конкурентами.

Возможности включают в себя любую предпочтительную ситуацию в настоящем или будущем, возникающую в условиях окружающей среды проекта.

Угроза представляет собой любую нежелательную ситуацию, тенденцию или изменение в условиях окружающей среды проекта, которые имеют разрушительный или угрожающий характер для его конкурентоспособности в настоящем или будущем.

Результаты SWOT-анализа представлены на таблице 4.

Таблица 4. *SWOT-анализ*

	Сильные стороны: С1. Широкий спектр функциональных возможностей программных продуктов. С2. Удобная и простая эксплуатация. С3. Низкие цены на программный продукт. С4. Поддержка в обслуживании.	**Слабые стороны:** Сл1. Дорогостоящее ПО. Сл2. Неквалифицированные пользователи. Сл3. Быстрое устаревание ПО и программного продукта. Сл4. Текучесть кадров.
Возможности: В1. Расширение географии рынка сбыта. В2. Осваивание новых отраслей. В3. Увеличение масштаба проектов. В4. Повышение квалификации сотрудников. В5. Расширение ассортимента услуг и поиск новых ниш	1.Привлечение зарубежных и международных клиентов высоким качеством программных продуктов и послепродажным обслуживаием. 2.Привлечение клиентов из новых отраслей низкими ценами и послепродажным обслуживанием.	1.Учитывать стаж работы в компании при повышении квалификации сотрудника и распределении др. бонусов. 2.Ввести дополнительные услуги по обучению пользователей. 3. Выгодно предлагать своевременное обновление ПО и программного продукта.
Угрозы: У1. Ужесточение	1.Поддерживать существующую ценовую	1. Разработать систему лояльности клиентов для

конкуренции У2. Выход новых конкурентов на рынок У3. Подорожание ресурсов. У4. Экономическая нестабильность. У5. Снижение клиентской платежеспособности.	политику для повышения конкурентоспособности. 2. Предлагать типовые решения, ранее разработанные фирмой, по сниженным ценам.	стимулирования повторных обращений. 2.Снижать издержки и оптимизировать ресурсы.

Таким образом, в результате SWOT-анализа были выявлены слабые и сильные стороны, а также возможные варианты повышения эффективности и минимизации угроз.

4.2 Планирование проектных работ

4.2.1 Структура работ в рамках проекта

Планирование комплекса предполагаемых работ осуществлено в следующем порядке:

- определение структуры работ проекта;
- определение участников каждой работы;
- установление продолжительности работ;
- построение графика проведения проектной работы.

Для выполнения технического задания была сформирована рабочая группа. По каждому виду запланированных работ установлена соответствующая должность исполнителей.

В данном разделе составлен перечень этапов и работ проекта, а также произведено распределение исполнителей по видам работ (табл. 5).

Таблица 5. *Перечень этапов, работ и распределение исполнителей*

Основные этапы	**№ раб**	**Содержание работ**	**Должность исполнителя**
Разработка технического задания	1	Составление и утверждение технического задания с заказчиком	Руководитель
Распределение работ	2	Определение вида информационной системы (торговля, оказание услуг, автоматизированные рабочие места и т.д.)	Руководитель
	3	Разделение информационной системы на функциональные подсистемы	Руководитель
	4	Календарное планирование работ	Руководитель

Проектирование информационной системы	5	Создание UML-диаграмм (диаграмм вариантов использования) для автоматизированных рабочих мест тренера и администратора	Дипломник
	6	Создание IDEF1X-диаграмм (проектирование базы данных информационной системы)	Дипломник
	7	Создание IDEF0-диаграмм (описание бизнес-процессов информационной системы)	Дипломник
	8	Проверка соответствия диаграмм техническому заданию	Руководитель
Разработка информационной системы	9	Создание АРМ тренера и администратора	Дипломник Программист
	10	Тестирование информационной системы, ввод корректировок	Дипломник Программист
Разработка технической документации	11	Описание информационной системы с точки зрения пользователя	Дипломник
Изложение выполненной работы в пояснительной записке	12	Создание отчета по проектной работе	Дипломник

4.2.2 Определение трудоемкости выполнения работ

Трудовые затраты в большинстве случаев образуют основную часть стоимости разработки, поэтому важным моментом является определение трудоемкости работ каждого из участников проекта.

Трудоемкость выполнения проекта оценивается экспертным путем в человеко-днях и носит вероятностный характер, т.к. зависит от множества трудно учитываемых факторов. Для определения ожидаемого (среднего) значения трудоемкости $t_{ожi}$ используется следующая формула:

$$t_{\text{ожі}} = \frac{3t_{\min i} + 2t_{\max i}}{5}, \qquad (3)$$

где $t_{ожi}$ – ожидаемая трудоемкость выполнения *i*-ой работы чел.-дн.;

$t_{\min i}$ – минимально возможная трудоемкость выполнения заданной *i*-ой работы (оптимистическая оценка: в предположении наиболее благоприятного стечения обстоятельств), чел.-дн.;

$t_{\max i}$ – максимально возможная трудоемкость выполнения заданной *i*-ой работы (пессимистическая оценка: в предположении наиболее неблагоприятного стечения обстоятельств), чел.-дн.

Расчеты $t_{ожi}$ занесены в таблицу 7.

Исходя из ожидаемой трудоемкости работ, определяется продолжительность каждой работы в рабочих днях $T_р$, учитывающая параллельность выполнения работ несколькими исполнителями. Такое вычисление необходимо для обоснованного расчета заработной платы, так как удельный вес зарплаты в общей сметной стоимости проекта составляет около 65 %.

$$T_{р_i} = \frac{t_{ожi}}{Ч_i}, \quad (4)$$

где $T_{рi}$ – продолжительность одной работы, раб. дн.;

$t_{ожi}$ – ожидаемая трудоемкость выполнения одной работы, чел.-дн.

$Ч_i$ – численность исполнителей, выполняющих одновременно одну и ту же работу на данном этапе, чел.

На протяжении выполнения проекта число дипломников составляло 3. Расчеты продолжительности работ представлены также в таблице 7.

4.2.3 Разработка графика проведения проекта

Наиболее удобным и наглядным способом отслеживания выполнения проектной работы является диаграмма Ганта.

Диаграмма Ганта – горизонтальный ленточный график, на котором работы по теме представляются протяженными во времени отрезками, характеризующимися датами начала и окончания выполнения данных работ.

Для удобства построения графика, длительность каждого из этапов работ из рабочих дней следует перевести в календарные дни. Для этого необходимо воспользоваться следующей формулой:

$$T_{кi} = T_{рi} \cdot k_{кал}, \quad (5)$$

где $T_{кi}$– продолжительность выполнения *i*-й работы в календарных днях;

$T_{рi}$ – продолжительность выполнения *i*-й работы в рабочих днях;

$k_{кал}$ – коэффициент календарности.

Коэффициент календарности определяется по следующей формуле:

$$k_{кал} = \frac{T_{кал}}{T_{кал} - T_{вых} - T_{пр}} = \frac{366}{247} = 1{,}4818, \quad (6)$$

где $T_{кал}$ – количество календарных дней в году;

$T_{вых}$ – количество выходных дней в году;

$T_{пр}$ – количество праздничных дней в году.

Тогда длительность каждого из этапов работ в календарных днях будет равна $T_{кi} = T_{рi} \cdot k_{кал} = T_{рi} * 1{,}4818$.

Все рассчитанные значения сведены в таблицу 6.

Таблица 6. *Временные показатели проведения научного исследования*

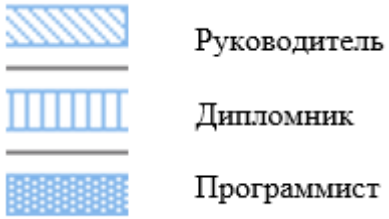
Название работы	**Трудоёмкость работ**			**Исполнители**	**Длительность работ в рабочих днях** $T_{рi}$	**Длительность работ в календарных днях** $T_{кi}$
	t_{min}, чел-дни	t_{max}, чел-дни	$t_{ожi}$ чел-дни			
Составление и утверждение ТЗ с заказчиком	2	5	3	Руководитель	3	4
Определение вида ИС	1	2	1	Руководитель	1	1
Разделение ИС на функциональные подсистемы	2	3	2	Руководитель	2	3
Календарное планирование работ	1	2	1	Руководитель	1	1
Создание UML-диаграмм для АРМ тренера и администратора	3	5	4	Дипломник (2)	2	3
Создание IDEF1X-диаграмм	4	6	5	Дипломник (2)	3	4
Создание IDEF0-диаграмм	4	8	6	Дипломник (2)	3	4

Проверка соответствия диаграмм ТЗ	1	3	2	Руководитель	1	1
Создание АРМ тренера и администратора	180	240	204	Дипломник (3) Программист (2)	85	126
Тестирование ИС, ввод корректировок	30	60	42	Дипломник Программист	42	62
Описание ИС с точки зрения пользователя	5	8	6	Дипломник (2)	3	4
Изложение выполненной работы в пояснительной записке	3	5	4	Дипломник (2)	2	3

На основе табл. 6 построен календарный план-график для максимального по длительности исполнения работ в рамках выполняемого проекта. В табл. 7 разбивка по месяцам и декадам (10 дней) за период времени дипломирования. При этом работы на графике выделены различной штриховкой (в зависимости от исполнителей), ответственные за ту или иную работу.

Таблица 7. *Календарный план-график проведения проектной работы*

№ раб	Вид работ	Исполнители	$T_{кi}$, кал. дн.	Продолжительность выполнения работ
				февраль: 1, 2, 3; март: 1, 2, 3; апрель: 1, 2, 3; май: 1, 2, 3
1	Составление и утверждение ТЗ с заказчиком	Руководитель	4	
2	Определение вида ИС	Руководитель	1	
3	Разделение ИС на функциональные подсистемы	Руководитель	3	
4	Календарное планирование работ	Руководитель	1	
5	Создание UML-диаграмм для АРМ тренера и администратора	Дипломник	3	
6	Создание IDEF1X-диаграмм	Дипломник	4	
7	Создание IDEF0-диаграмм	Дипломник	4	
8	Проверка соответствия диаграмм ТЗ	Руководитель	1	
9	Создание АРМ тренера и администратора	Дипломник Программист	126	
10	Тестирование ИС, ввод корректировок	Дипломник Программист	62	
11	Описание ИС с точки зрения пользователя	Дипломник	4	
12	Изложение выполненной работы в пояснительной записке	Дипломник	3	



4.2.4 Бюджет научно-технического исследования (НТИ)

При планировании бюджета НТИ должно быть обеспечено полное и достоверное отражение всех видов расходов, связанных с его выполнением. В процессе формирования бюджета НТИ используется следующая группировка затрат по статьям:

- материальные затраты НТИ;
- затраты на специальное оборудование для научных (экспериментальных) работ;
- основная заработная плата исполнителей темы;
- дополнительная заработная плата исполнителей темы;
- отчисления во внебюджетные фонды (страховые отчисления);
- затраты научные и производственные командировки;
- контрагентные расходы;
- накладные расходы.

4.2.4.1 Расчет материальных затрат НТИ

Данная статья включает стоимость всех материалов, используемых при разработке проекта:

- приобретаемые со стороны сырье и материалы, необходимые для создания научно-технической продукции;

- покупные материалы, используемые в процессе создания научно-технической продукции для обеспечения нормального технологического процесса и для упаковки продукции или расходуемых на другие производственные и хозяйственные нужды (проведение испытаний, контроль, содержание, ремонт и эксплуатация оборудования, зданий, сооружений, других основных средств и прочее), а также запасные части для ремонта оборудования, износа инструментов, приспособлений, инвентаря, приборов, лабораторного оборудования и других средств труда, не относимых к основным средствам, износ спецодежды и других малоценных и быстроизнашивающихся предметов;

- покупные комплектующие изделия и полуфабрикаты, подвергающиеся в дальнейшем монтажу или дополнительной обработке;

- сырье и материалы, покупные комплектующие изделия и полуфабрикаты, используемые в качестве объектов исследований (испытаний) и для эксплуатации, технического обслуживания и ремонта изделий – объектов испытаний (исследований);

В материальные затраты, помимо вышеуказанных, включаются дополнительно затраты на канцелярские принадлежности, диски, картриджи и т.п. Однако их учет ведется в данной статье только в том случае, если в научной организации их не включают в расходы на использование оборудования или накладные расходы. В первом случае на них определяются соответствующие нормы расхода от установленной базы. Во втором случае их величина учитывается как некая доля в коэффициенте накладных расходов.

Расчет материальных затрат осуществляется по следующей формуле:

$$З_{м} = (1 + k_T) \cdot \sum_{i=1}^{m} Ц_i \cdot N_{расхi} \ , \qquad (7)$$

где m – количество видов материальных ресурсов, потребляемых при выполнении научного исследования;

$N_{расхi}$ – количество материальных ресурсов i-го вида, планируемых к использованию при выполнении научного исследования (шт., кг, м, м^2 и т.д.);

$Ц_i$ – цена приобретения единицы i-го вида потребляемых материальных ресурсов (руб./шт., руб./кг, руб./м, руб./м^2 и т.д.);

k_T – коэффициент, учитывающий транспортно-заготовительные расходы.

Значения цен на материальные ресурсы могут быть установлены по данным, размещенным на соответствующих сайтах в Интернете предприятиями-изготовителями (либо организациями-поставщиками).

Величина коэффициента (k_T), отражающего соотношение затрат по доставке материальных ресурсов и цен на их приобретение, зависит от условий договоров поставки, видов материальных ресурсов,

территориальной удаленности поставщиков и т.д. Транспортные расходы принимаются в пределах 15-25% от стоимости материалов. Материальные затраты, необходимые для данной разработки, заносятся в таблицу 8.

Таблица 8. *Материальные затраты*

Наименование	Единица измерения	Количество	Цена за ед., руб.	Затраты на материалы, (**З**м), руб.
Ноутбук	шт.	5	25 000	125 000
Стол компьютерный	шт.	5	3 000	15 000
Стул офисный	шт.	5	1 200	6 000
Мышь компьютерная	шт.	5	700	3 500
Типовая конфигурация 1С: Управление небольшой фирмой	шт.	1	4 500	4 500
Итого				**154 000**

4.2.4.2 Основная заработная плата исполнителей темы

В настоящую статью включается основная заработная плата научных и инженерно-технических работников, рабочих макетных мастерских и опытных производств, непосредственно участвующих в выполнении работ по данной теме. Величина расходов по заработной плате определяется исходя из трудоемкости выполняемых работ и действующей системы окладов и тарифных ставок. В состав основной заработной платы включается премия, выплачиваемая ежемесячно из фонда заработной платы в размере 20 –30 % от тарифа или оклада. Расчет основной заработной платы сводится в табл. 9.

Статья включает основную заработную плату работников, непосредственно занятых выполнением НТИ, (включая премии, доплаты) и дополнительную заработную плату:

$$З_{зп} = З_{осн} + З_{доп}, \quad (8)$$

где $З_{осн}$ – основная заработная плата;

$З_{доп}$ – дополнительная заработная плата (12-20 % от $З_{осн}$).

Основная заработная плата ($З_{осн}$) руководителя (программиста) от предприятия (при наличии руководителя от предприятия) рассчитывается по следующей формуле:

$$З_{осн} = З_{дн} \cdot T_р, \quad (9)$$

где $З_{осн}$ – основная заработная плата одного работника;

$Т_р$ – продолжительность работ, выполняемых научно-техническим работником, раб. дн. (табл. 8);

$З_{дн}$ – среднедневная заработная плата работника, руб.

Таблица 9. *Расчет основной заработной платы*

№ п/п	Наименование этапов	Исполнители по категориям	Трудоемкость, чел.-дн.	Заработная плата, приходящаяся на один чел.-дн., тыс. руб.	Всего заработная плата по тарифу (окладам), тыс. руб.
1	Составление и утверждение ТЗ с заказчиком	Руководитель	3	1,4	4,2
2	Определение вида ИС	Руководитель	1	1,4	1,4
3	Разделение ИС на функциональные подсистемы	Руководитель	2	1,4	2,8
4	Календарное планирование работ	Руководитель	1	1,4	1,4
5	Создание UML-диаграмм для АРМ тренера и администратора	Дипломник (2)	4	0	0
6	Создание IDEF1X-диаграмм	Дипломник (2)	5	0	0
7	Создание IDEF0-диаграмм	Дипломник (2)	6	0	0
8	Проверка соответствия диаграмм ТЗ	Руководитель	2	1,4	2,8
9	Создание АРМ тренера и администратора	Дипломник (3) Программист (2)	204	0 1,1	0 448,8
10	Тестирование ИС, ввод	Дипломник Программист	42	0 1,1	0 46,2

	корректировок				
11	Описание ИС с точки зрения пользователя	Дипломник (2)	6	0	0
12	Изложение выполненной работы в пояснительной записке	Дипломник (2)	4	0	0
Итого:					507,6

Среднедневная заработная плата рассчитывается по формуле:

$$З_{дн} = \frac{З_м \cdot М}{F_д}, \qquad (10)$$

где $З_м$ – месячный должностной оклад работника, руб.;

М – количество месяцев работы без отпуска в течение года:

при отпуске в 24 раб. дня М =11,2 месяца, 5-дневная неделя;

при отпуске в 48 раб. дней М=10,4 месяца, 6-дневная неделя;

$F_д$ – действительный годовой фонд рабочего времени научно-технического персонала, раб. дн. (табл. 10).

Таблица 10. *Баланс рабочего времени*

Показатели рабочего времени	**Руководитель**	**Программист**	**Дипломник**
Календарное число дней	10	188	206
Количество нерабочих дней - выходные дни - праздничные дни	2	61	66
Потери рабочего времени - отпуск - невыходы по болезни	0	0	0
Действительный годовой фонд рабочего времени	260	260	260

Месячный должностной оклад работника:

$$З_м = З_{тс} \cdot (1 + k_{пр} + k_д) \cdot k_р, \qquad (11)$$

где $З_{тс}$ – заработная плата по тарифной ставке, руб.;

$k_{пр}$ – премиальный коэффициент, равный 0,3 (т.е. 30% от $З_{тс}$);

$k_д$ – коэффициент доплат и надбавок составляет примерно 0,2 – 0,5 (в НИИ и на промышленных предприятиях – за расширение сфер

обслуживания, за профессиональное мастерство, за вредные условия: 15-20 % от $З_{тс}$);

$k_р$ – районный коэффициент, равный 1,3 (для Томска).

Тарифная заработная плата $З_{тс}$ находится из произведения тарифной ставки работника 1-го разряда $Т_{сi}$ = 600 руб. на тарифный коэффициент $k_т$ и учитывается по единой для бюджетных организации тарифной сетке. Для предприятий, не относящихся к бюджетной сфере, тарифная заработная плата (оклад) рассчитывается по тарифной сетке, принятой на данном предприятии. Расчёт основной заработной платы приведён в табл. 11.

Таблица 11. *Расчёт основной заработной платы*

Исполнители	**$З_{тс}$, руб.**	**$k_{пр}$**	**$k_д$**	**$k_р$**	**$З_м$, руб**	**$З_{дн}$, руб.**	**$Т_р$, раб. дн.**	**$З_{осн}$, руб.**
Руководитель	30 800	0,3	0,2	1,3	60 060	2 772	8	22 176
Программист	24 200	0,3	0	1,3	40 898	1 887,6	127	239 725,2
							Итого $З_{осн}$	261 901,2

4.2.4.3 Дополнительная заработная плата исполнителей темы

Затраты по дополнительной заработной плате исполнителей темы учитывают величину предусмотренных Трудовым кодексом РФ доплат за отклонение от нормальных условий труда, а также выплат, связанных с обеспечением гарантий и компенсаций (при исполнении государственных и общественных обязанностей, при совмещении работы с обучением, при предоставлении ежегодного оплачиваемого отпуска и т.д.).

Расчет дополнительной заработной платы ведется по следующей формуле:

$$З_{доп} = k_{доп} \cdot З_{осн} \qquad (12)$$

где $k_{доп}$ – коэффициент дополнительной заработной платы (на стадии проектирования принимается равным 0,12 – 0,15).

Таблица 12. *Расчёт дополнительной заработной платы*

Исполнители	**$З_{осн}$, руб.**	**$k_{доп}$**	**$З_{доп}$, руб.**
Руководитель	22 176	0,12	2661,12
Программист	239 725,2	0,12	28 767,02
		Итого $З_{доп}$	31 428,14

4.2.4.4 Отчисления во внебюджетные фонды (страховые отчисления)

В данной статье расходов отражаются обязательные отчисления по установленным законодательством Российской Федерации нормам органам государственного социального страхования (ФСС), пенсионного фонда (ПФ) и медицинского страхования (ФФОМС) от затрат на оплату труда работников.

Величина отчислений во внебюджетные фонды определяется исходя из следующей формулы:

$$З_{внеб} = k_{внеб} \cdot (З_{осн} + З_{доп}), \quad (13)$$

где $k_{внеб}$ – коэффициент отчислений на уплату во внебюджетные фонды (пенсионный фонд, фонд обязательного медицинского страхования и пр.).

На 2016 г. в соответствии с Федерального закона от 24.07.2009 №212-ФЗ установлен размер страховых взносов равный 30,2%.

Отчисления во внебюджетные фонды представлены в таблице 13.

Таблица 13. *Отчисления во внебюджетные фонды*

Исполнитель	**Основная заработная плата, руб.**	**Дополнительная заработная плата, руб.**
Руководитель проекта	22 167	2661,12
Программист	239 725,2	28 767,02
Коэффициент отчислений во внебюджетные фонды	0,302	
Итого:	**88 582,74**	

4.2.4.5 Накладные расходы

Накладные расходы учитывают прочие затраты организации, не попавшие в предыдущие статьи расходов: печать и ксерокопирование материалов исследования, оплата услуг связи, электроэнергии, почтовые и телеграфные расходы, размножение материалов и т.д. Их величина определяется по следующей формуле:

$$З_{накл} = (\text{сумма статей } 1 \div 4) \cdot k_{нр}, \quad (14)$$

где $k_{нр}$ – коэффициент, учитывающий накладные расходы.

Величина коэффициента накладных расходов 16%.

Таким образом, $З_{накл} = 535325{,}38 \cdot 0{,}16 = 85652{,}06$

4.2.4.6 Формирование бюджета затрат научно-исследовательского проекта

Рассчитанная величина затрат научно-исследовательской работы (темы) является основой для формирования бюджета затрат проекта, который при формировании договора с заказчиком защищается научной организацией в качестве нижнего предела затрат на разработку научно-технической продукции.

Определение бюджета затрат на научно-исследовательский проект приведено в табл. 14.

Таблица 14. *Расчет бюджета затрат НТИ*

Наименование статьи	Сумма, руб.	Примечание
1. Материальные затраты НТИ	154 000	Пункт 3.4.1
2. Затраты по основной заработной плате исполнителей темы	261 901,2	Пункт 3.4.2
3. Затраты по дополнительной заработной плате исполнителей темы	31 428,14	Пункт 3.4.3
4. Отчисления во внебюджетные фонды	88 582,74	Пункт 3.4.4
5. Накладные расходы	85 745,93	16 % от суммы ст. 1-4
6. Бюджет затрат НТИ	621 658,01	Сумма ст. 1- 5

4.3 Определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования

Определение эффективности происходит на основе расчета интегрального показателя эффективности научного исследования. Его нахождение связано с определением двух средневзвешенных величин: финансовой эффективности и ресурсоэффективности.

Интегральный финансовый показатель разработки определяется как:

$$I_{финр} = \frac{Ф_р}{Ф_{max}}, \qquad (15)$$

где $I_{\text{финр}}$ – интегральный финансовый показатель разработки;

$\Phi_{\text{р}}$ – стоимость исполнения;

$\Phi_{\max}$ – максимальная стоимость исполнения научно-исследовательского проекта (в т.ч. аналоги).

$$I_{\text{финр}} = \frac{621\,658{,}01}{700\,000} = 0{,}89$$

Полученная величина интегрального финансового показателя разработки отражает соответствующее численное удешевление стоимости разработки в разах.

Интегральный показатель ресурсоэффективности исполнения объекта исследования можно определить следующим образом:

$$\text{I}_{\text{р}} = \sum a \cdot b \ , \qquad (16)$$

где I_p – интегральный показатель ресурсоэффективности;

a – весовой коэффициент;

b – бальная оценка, устанавливается экспертным путем по выбранной шкале оценивания;

n – число параметров сравнения.

Расчет интегрального показателя ресурсоэффективности приведен в таблице 15.

Таблица 15. *Сравнительная оценка характеристик вариантов исполнения проекта*

Критерии / Объект исследования	Весовой коэффициент параметра	Оценка выполнения
1. Способствует росту производительности труда пользователя	0,15	4
2. Удобство в эксплуатации (соответствует требованиям потребителей)	0,25	5
3. Надежность	0,1	4
4. Экономия времени	0,1	4
5. Единство исполнения	0,4	5
ИТОГО	1	

I_p =4*0,15+5*0,25+4*0,1+4*0,1+5*0,4=4,65;

Интегральный показатель эффективности исполнения разработки ($I_{исп.}$) определяется на основании интегрального показателя ресурсоэффективности и интегрального финансового показателя по формуле:

$$I_{исп.} = \frac{I_p}{I_{финр}} = \frac{4,65}{0,89} = 5,22 \quad (17)$$

Полученное значение интегрального показателя эффективности исполнения разработки превысил максимальный балл в системе оценивания. Это говорит о том, что результат работы можно считать положительным, так как оценка интегрального показателя ресурсоэффективности близка к максимальной, при этом стоимость разработки ниже, чем у ряда аналогов, рассмотренных при анализе конкурентных решений.

В целом данные, полученные при анализе конкурентных решений и оценочной карты Quad, позволяют сделать вывод, что разработка программного продукта является перспективной и привлекательной для инвесторов. Продукт имеет множество преимуществ перед рассмотренными конкурентными решениями, в особенности по таким критериям, как удобство в эксплуатации, функциональные возможности и цена.

SWOT-анализ позволил выявить слабые и сильные стороны, позволяющие повысить эффективность и сократить угрозы, что, в свою очередь, будет способствовать реализации планов по расширению направлений развития.

Также была распланирована структура работ проекта и определены ответственные должности для их выполнения. В соответствии с назначенными работами была рассчитана их трудоемкость и составлен график работ (диаграмма Ганта). Общая длительность проектирования и разработки программного продукта составила 216 дней.

Общий бюджет НТИ составил 621 658,01 рублей. Он включает в себя затраты на основную и дополнительную заработную плату работников,

материальные затраты, отчисления на внебюджетные фонды и накладные расходы.

5 Социальная ответственность

В процессе трудовой деятельности на сотрудника офиса могут оказывать воздействие различного рода производственные факторы. Для их предупреждения и сохранения здоровья работника предусматривается ряд мер по обеспечению безопасности трудовой деятельности.

В данном разделе рассматривается анализ вредных и опасных факторов труда, определяются необходимые меры защиты от них, оцениваются условия труда и предоставляются рекомендации по их оптимизации.

Как правило, офисные работники сталкиваются с повышенным уровнем шума, нарушением температурного режима, недостаточной освещенностью и т.д. Важную роль играют и психофизические факторы: зрительное, слуховое, умственное перенапряжение, монотонность труда и т.д.

Создаваемый программный продукт является комплексным и охватывает различные сферы деятельности спортивного клуба. Основным ядром системы являются автоматизированные рабочие места администратора и тренера. Основные функциональные блоки, доступные администратору: управление абонементами, инвентарем, регистрация посещения, добавление спортсмена, просмотр информации о спортсмене; тренеру: просмотр расписания группы, просмотр бланка занятия; заполнение данных о проводимом занятии, отметка посетивших занятие спортсменов.

Все работы по разработке данного проекта были выполнены в офисе компании ООО «Центр внедрений». Характеристика помещения:

- ширина рабочего помещения 8 м, длина – 6 м, высота – 3,5м;
- площадь – 48 м^2;
- объём помещения - 168 м^3;

- имеется кондиционер, а также естественная вентиляция: вытяжное вентиляционное отверстие, щели, двери, окна;
- искусственное освещение;
- естественное освещение.

В данном помещении работает семь человек. Следовательно, в среднем на одного сотрудника приходится около 24 $м^3$ объема помещения и 6,9 $м^2$ площади, что удовлетворяет требованиям санитарных норм, согласно которым для одного работника должны быть предусмотрены площадь величиной не менее 6 $м^2$ и объем не менее 24 $м^3$, с учетом максимального числа одновременно работающих в смену.

5.1 Производственная безопасность.

Опасные и вредные производственные факторы подразделяются на 4 группы по оказываемому влиянию на человека: физические, химические, биологические и психофизиологические. Так как на состояние офисных работников (программистов) химические и биологические факторы не оказывают существенного влияния, то основное внимание будет уделено физическим и психофизиологическим факторам.

Для представления всех вредных и опасных факторов необходимо классифицировать их в соответствии с нормативными документами.

Таблица 16. *Классификация вредных и опасных факторов*

Наименование видов работ и параметров производственного процесса	Факторы (ГОСТ 12.0.003-74 ССБТ)		Нормативные документы
	Вредные	Опасные	
1	2	3	4

Работа с компьютером и орг. техникой	1. Повышенная или пониженная влажность воздуха 2. Повышенная (пониженная) температура воздуха 3. Повышенный уровень шума 4. Повышенный уровень электромагнитных излучений 5. Недостаточная освещенность рабочего места 6. Эмоциональные перегрузки 7. Умственное перенапряжение 8. Монотонность труда 9.	1. Опасность поражения электрическим током	1. ГОСТ 12.0.003-74 2. СанПиН 2.2.4.548-96 3. ГОСТ 12.1.006–84 4. СанПиН 2.2.1/2.1.1.1278-03 5. СанПиН 2.2.2/2.4.1340-03 6. СНиП 2.04.05-91

5.1.1 Анализ выявленных вредных факторов при разработке и эксплуатации проектируемого решения

5.1.1.1 Микроклимат рабочего помещения

Микроклимат производственных (рабочих) помещений – климат внутренней среды этих помещений, который определяется действующими на организм человека сочетаниями температуры, влажности и скорости движения воздуха, а также интенсивности теплового излучения от нагретых поверхностей.

Мероприятия по доведению микроклиматических показателей до нормативных значений включаются в комплексные планы предприятий по охране труда. Для создания благоприятных условий работы, соответствующих физиологическим потребностям человеческого организма, санитарные нормы устанавливают оптимальные и допустимые метеорологические условия в рабочей зоне помещения.

Выполняемая работа относится к категории легкая (1б).

Таблица 17. *Оптимальные величины показателей микроклимата на рабочих местах производственных помещений (СанПиН 2.2.4.548-96)*

Период года	Температура воздуха, ^{0}C	Температура поверхностей, ^{0}C	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	21 - 23	20 – 24	60-40	0,1
Теплый	23-25	22-26	60-40	0,1

Период года	Температура воздуха, °С		Температура поверхностей, °С	Относительная влажность воздуха, %	Скорость движения воздуха, м/с	
	диапазон ниже оптимальных величин	диапазон выше оптимальных величин			для диапазона температур воздуха ниже оптимальных величин, не более	для диапазона температур воздуха выше оптимальных величин, не более
Холодный	19,0 - 20,9	23,1 - 24,0	18,0 - 25,0	15 - 75	0,1	0,2
Теплый	20,0 - 21,9	24,1 - 28,0	19,0 - 29,0	15 - 75	0,1	0,3

В данном случае температура воздуха и температура поверхностей составляют 22^0С и 21^0С при относительной влажности 45% в холодный период года; 24^0С и 23^0С при относительной влажности воздуха 50% в теплый период года, что соответствует нормам.

5.1.1.2 Производственное освещение

Освещение – получение, распределение и использование световой энергии для обеспечения благоприятных условий видения предметов и объектов. Оно влияет на настроение и общее самочувствие, определяет эффективность труда. Нерационально организованное освещение может явиться причиной травматизма: плохо освещенные опасные зоны, слепящие источники света и блики от них, резкие тени и пульсации освещенности ухудшают видимость и могут вызвать неадекватное восприятие наблюдаемого объекта. В компьютерных залах должно быть естественное и искусственное освещение. Естественное освещение обеспечивается за счет оконных проемов, коэффициент искусственного освещения (КОЕ) которых должен быть не менее 1,2% в местах, где имеется снежный покров и не менее 1,5% на остальной территории. Свет из окна должен быть с левой стороны от пользователя. Естественное освещение в офисе осуществляется через два оконных проема размером 2 на 1.5 метра в наружной стене. Нормируемые показатели естественного, искусственного и совмещенного освещения в соответствии с СанПиН 2.2.1/2.1.1.1278-03 указаны в таблице.

Таблица 18. *Нормируемые показатели естественного, искусственного и совмещенного освещения в соответствии с СанПиН 2.2.1/2.1.1.1278-03*

Помещения	Рабочая поверхно	Естественное освещение	Совмещенное освещение	Искусственное освещение

	сть и плоскость нормирования КЕО и освещенности и высота плоскости над полом, м	КЕО e_н, %		КЕО e_н, %		Освещенность, лк				
		При верхнем или комбинированном освещении	При боковом освещении	При верхнем или комбинированном освещении	При боковом освещении	При комбинированном освещении		При общем освещении	Показатель дискомфорта, М, не более	Коэффициент пульсации освещенности, К_п, %, не более
						всего	от общего			
1	2	3	4	5	6	7	8	9	10	11
Кабинеты, рабочие комнаты	Г – 0,8	3,0	1,0	1,8	0,6	400	200	300	40	15
Помещения для работы с дисплеями и видеотерминалами, залы ЭВМ	Г – 0,8 Экран монитора : В – 1,2	3,5 -	1,2 -	2,1 -	0,7 -	500 -	300 -	400 200	15 -	10

Для искусственного освещения помещений с персональными компьютерами следует применять светильники типа ЛПО36. Допускается применять светильники прямого света, преимущественно отраженного света типа ЛПО13, ЛПО5, ЛСО4, ЛПО34, ЛПО31 с люминесцентными лампами типа ЛБ. Допускается применение светильников местного освещения с лампами накаливания. Светильники должны располагаться линиями (прямыми или прерывающимися) так, чтобы при разном положении машин они были параллельно линии зрения пользователя. Защитный угол светильников должен быть не менее 40 градусов.

Чтобы поддерживать освещение в помещении по всем соответствующим нормам, необходимо хотя бы два раза в год стекла и светильники, а так же по мере необходимости заменять перегоревшие лампы.

В утреннее и вечернее время вводится общее искусственное освещение. Основными источниками искусственного освещения являются лампы белого и дневного света ЛБ-20 и ЛД-20.

5.1.1.3 Производственные шумы

Шум – это совокупность различных звуков, возникающих в процессе производства и неблагоприятно воздействующих на организм.

Шум может привести к нарушениям слуха (в случае постоянного нахождения при шуме более 85 децибел), может являться фактором стресса и повысить систолическое кровяное давление.

Дополнительно, он может способствовать несчастным случаям, маскируя предупреждающие сигналы и мешая сконцентрироваться.

Для рассматриваемого помещения основными источниками шума являются персональные компьютеры, кондиционер и вытяжные вентиляторы на окнах. Нормативным документом, регламентирующим уровни шума для различных категорий рабочих мест служебных помещений, является ГОСТ 12.1.003-83 «ССБТ. Шум. Общие требования безопасности».

Помещения, в которых для работы используются ПК не должны граничить с помещениями, в которых уровни шума превышают нормируемые значения.

В помещениях, оборудованных ПК, которые являются основным источником шума при выполнении данных видов работ, уровень шума на рабочем месте не должен превышать 50 дБА.

5.1.1.4 Электромагнитные поля

При работе с персональным компьютером (ПК) человек подвергает воздействию ряда вредных факторов: электромагнитного и электростатического полей.

Электромагнитное излучение, создаваемое персональным компьютером, имеет сложный спектральный состав в диапазоне частот от 0 Гц до 1000 МГц, а также электрическую (Е) и магнитную (Н) составляющие.

Основным источником электромагнитных излучений от мониторов ПЭВМ (ПК) является трансформатор высокой частоты строчной развертки. На сегодняшний день ЭЛТ-мониторы практически повсюду заменены на ЖК-мониторы, электромагнитное излучение от которых в разы меньше, чем от ЭЛТ-мониторов.

В соответствии с СанПиН 2.2.4.1191-03 нормы допустимых уровней напряженности электрических полей зависят от времени пребывания

человека в контролируемой зоне. Время допустимого пребывания в рабочей зоне в часах составляет T=50/E–2. Работа в условиях облучения электрическим полем с напряженностью 20–25 кВ/м продолжается не более 10 минут. При напряженности не выше 5 кВ/м присутствие людей в рабочей зоне разрешается в течение 8 часов.

Безопасные уровни излучений также регламентируются нормами Госкомсанэпидемнадзора «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы» (СанПиН 2.2.4.1340-03).

В таблицах ниже представлены предельно-допустимые уровни напряженности на рабочих местах и допустимые уровни электромагнитных полей.

Таблица 19. *Предельно-допустимые уровни напряженности на рабочих местах*

Время воздействия за рабочий день, мин	Условия воздействия			
	Общее		локальное	
	ПДУ напряженности кА/м	ПДУ магнитной индукции мТл	ПДУ напряженности кА/м	ПДУ магнитной индукции мТл
0 - 10	24	30	40	50
11 - 60	16	20	24	30
61 - 480	8	10	12	15

Таблица 20. *Допустимые уровни электромагнитных полей согласно СанПиН 2.2.4.1340-03*

Наименование параметра	
Напряженность электромагнитного поля на расстоянии 50 см вокруг дисплея до электрической составляющей, В/м, не более: в диапазоне частот 5 Гц – 2 кГц в диапазоне частот 2 – 400 кГц	 25 2,5
Плотность магнитного потока на расстоянии 50 см вокруг дисплея, нТл, не более: в диапазоне частот 5 Гц – 2 кГц в диапазоне частот 2 – 400 кГц	 250 25
Поверхностный электростатический потенциал, В, не более	500

Мероприятия по снижению излучений включают:

- сертификацию ПЭВМ и аттестацию рабочих мест;
- применение экранов и фильтров;

- организационно-технические мероприятия;
- применение средств индивидуальной защиты путем экранирования пользователя ПЭВМ целиком или отдельных зон его тела;
- использование и применение профилактических напитков;
- использование иных технических средств защиты от патогенных излучений.

5.1.2 Анализ выявленных опасных факторов при разработке и эксплуатации проектируемого решения

5.1.2.1 Электробезопасность

Электробезопасность – система организационных и технических мероприятий и средств, обеспечивающих защиту людей от вредного и опасного для жизни воздействия электрического тока, электрической дуги, электромагнитного поля и статического электричества.

Опасное и вредное воздействия на людей электрического тока и электрической дуги проявляются в виде электротравм и профессиональных заболеваний.

Помещение, где расположены персональные вычислительные машины, относится к помещениям без повышенной опасности, так как отсутствуют следующие факторы:

- сырость;
- токопроводящая пыль;
- токопроводящие полы;
- высокая температура;
- возможность одновременного прикосновения человека к имеющим соединение с землёй металлоконструкциям зданий, технологическим аппаратам и механизмам и металлическим корпусам электрооборудования.

К мероприятиям по предотвращению возможности поражения электрическим током следует отнести:

- при производстве монтажных работ необходимо использовать только исправный инструмент, аттестованный службой КИПиА;

- с целью защиты от поражения электрическим током, возникающим между корпусом приборов и инструментом при пробое сетевого напряжения на корпус, корпуса приборов и инструментов должны быть заземлены;

- при включенном сетевом напряжении работы на задней панели должны быть запрещены;

- все работы по устранению неисправностей должен производить квалифицированный персонал;

- необходимо постоянно следить за исправностью электропроводки.

Перед началом работы следует убедиться в отсутствии свешивающихся со стола или висящих под столом проводов электропитания, в целостности вилки и провода электропитания, в отсутствии видимых повреждений аппаратуры и рабочей мебели, в отсутствии повреждений и наличии заземления приэкранного фильтра.

Токи статического электричества, наведенные в процессе работы компьютера на корпусах монитора, системного блока и клавиатуры, могут приводить к разрядам при прикосновении к этим элементам. Такие разряды опасности для человека не представляют, но могут привести к выходу из строя компьютера. Для снижения величин токов статического электричества используются нейтрализаторы, местное и общее увлажнение воздуха, использование покрытия полов с антистатической пропиткой.

5.2 Экологическая безопасность.

Охрана окружающей среды сводится к устранению отходов бытового мусора и отходам жизнедеятельности человека. В случае выхода из строя ПК, они списываются и отправляются на специальный склад, который при необходимости принимает меры по утилизации списанной техники и комплектующих.

На сегодняшний день одним из самых распространенных источников ртутного загрязнения являются вышедшие из эксплуатации люминесцентные

лампы. Каждая такая лампа, кроме стекла и алюминия, содержит около 60 мг ртути. Поэтому отслужившие свой срок люминесцентные лампы, а также другие приборы, содержащие ртуть, представляют собой опасный источник токсичных веществ.

В целом, утилизация ламп предполагает передачу использованных ламп предприятиям – переработчикам, которые с помощью специального оборудования перерабатывают вредные лампы в безвредное сырье – сорбент, которое в последующем используют в качестве материала для производства, например тротуарной плитки.

Под хранением отходов понимается временное размещение их в специально отведенных для этого местах или объектах до их утилизации. Отработанные люминесцентные лампы, согласно Классификатору отходов ДК 005-96, утвержденному приказом Госстандарта № 89 от 29.02.96 г., относятся к отходам, которые сортируются и собираются отдельно, поэтому утилизация люминесцентных ламп и их хранение должны отвечать определенные требованиям.

5.3 Безопасность в чрезвычайных ситуациях.

В данном случае на объекте (офис) могут возникать чрезвычайные ситуации (ЧС) следующего характера:

- техногенные;
- экологические;
- природные.

Наиболее типичной ЧС для помещения, котором производится выполнение ВКР, является пожар. Данная ЧС может произойти в случае замыкания электропроводки оборудования, обрыву проводов, не соблюдению мер пожаробезопасности и т.д.

Пожарная безопасность – комплекс организационных и технических мероприятий, направленных на обеспечение безопасности людей, на предотвращение пожара, ограничение его распространения, а также на создание условий для успешного тушения пожара.

Рабочее помещение, в котором производится работа по выполнению ВКР по пожарной и взрывной опасности относят к категории В.

К противопожарным мероприятиям в помещении относят следующие мероприятия:

1) помещение должно быть оборудовано: средствами тушения пожара (огнетушителями, ящиком с песком, стендом с противопожарным инвентарем); средствами связи; должна быть исправна электрическая проводка осветительных приборов и электрооборудования.

2) каждый сотрудник должен знать место нахождения средств пожаротушения и средств связи; помнить номера телефонов для сообщения о пожаре; уметь пользоваться средствами пожаротушения.

Помещение обеспечено средствами пожаротушения в соответствии с нормами:

1) пенный огнетушитель ОП-10 – 1 шт.

2) углекислотный огнетушитель ОУ-5 – 1 шт.

Помещение и этаж оборудованы следующими средствами оповещения:

- световая индикация в коридорах этажа;
- звуковая индикация в виде громкоговорителя;
- пассивными датчиками задымленности.

Для того чтобы избежать возникновения пожара необходимо проводить следующие профилактические работы, направленные на устранение возможных источников возникновения пожара:

- периодическая проверка проводки;
- отключение оборудования при покидании рабочего места;
- проведение инструктажа работников о пожаробезопасности.

Чтобы увеличить устойчивость офисного помещения к ЧС необходимо устанавливать системы противопожарной сигнализации, реагирующие на дым и другие продукты горения, установка огнетушителей,

обеспечить офис и проинструктировать рабочих о плане эвакуации из офиса, а также назначить ответственных за эти мероприятия. Два раза в год (в летний и зимний период) проводить учебные тревоги для отработки действий при пожаре. В ходе осмотра офисного помещения были выявлены системы, сигнализирующие о наличие пожара или задымленности помещения и наличие огнетушителей.

В случае возникновения ЧС как пожар, необходимо предпринять меры по эвакуации персонала из офисного помещения в соответствии с планом эвакуации. При отсутствии прямых угроз здоровью и жизни произвести попытку тушения возникшего возгорания огнетушителем. В случае потери контроля над пожаром, необходимо эвакуироваться вслед за сотрудниками по плану эвакуации и ждать приезда специалистов, пожарников. При возникновении пожара должна сработать система пожаротушения, издав предупредительные сигналы, и передав на пункт пожарной станции сигнал о ЧС, в случае если система не сработала, по каким-либо причинам, необходимо самостоятельно произвести вызов пожарной службы по телефону 101, сообщить место возникновения ЧС и ожидать приезда специалистов.

5.4 Правовые и организационные вопросы обеспечения безопасности.

Требования к организации рабочих мест пользователей:

- рабочее место должно быть организовано с учетом эргономических требований согласно ГОСТ 12.2.032-78 «ССБТ. Рабочее место при выполнении работ сидя. Общие эргономические требования» и ГОСТ 12.2.061-81 «ССБТ. Оборудование производственное. Общие требования безопасности к рабочим местам»;
- конструкция рабочей мебели (рабочий стол, кресло, подставка для ног) должна обеспечивать возможность индивидуальной регулировки соответственно росту пользователя и создавать удобную позу для работы. Вокруг ПК должно быть обеспечено свободное пространство не менее 60-120см;

- на уровне экрана должен быть установлен оригинал-держатель.

В соответствии с государственными стандартами и правовыми нормами обеспечения безопасности предусмотрена рациональная организация труда в течение смены, которая предусматривает:

- длительность рабочей смены не более 8 часов;
- установление двух регламентируемых перерывов (не менее 20 минут после 1-2 часов работы, не менее 30 минут после 2 часов работы);
- обеденный перерыв не менее 40 минут.

Обязательно предусмотрен предварительный медосмотр при приеме на работу и периодические медосмотры.

Каждый сотрудник должен пройти инструктаж по технике безопасности перед приемом на работу и в дальнейшем, должен быть пройден инструктаж по электробезопасности и охране труда.

Заключение

В результате проектирования бизнес-процессов предметной области была разработана и внедрена система информационной поддержки деятельности спортивного клуба. Были рассмотрены все проблемы, касающиеся этой области, которые привели к необходимости создания программного продукта. Для реализации информационной системы было выбрано наиболее подходящее средство разработки – платформа «1С», решение «1С:Управление небольшой фирмой 8», которое полностью себя оправдало.

При обзоре существующих решений был выявлен один недостаток имеющихся информационных систем: они не были ориентированы на профессиональных спортсменов. А главной проблемой проектируемой системы была необходимость отслеживания медицинских допусков спортсменов к занятиям. При внедрении информационной системы проблемы, описанные в причинно-следственной диаграмме Исикавы, были устранены.

В ходе разработки основные задачи проекта были выполнены, и результатом является комплексный программный продукт, позволяющий эффективно управлять деятельностью клуба: составлять расписание занятий, отслеживать посещения, управлять абонементами и инвентарем, хранить и оперативно изменять информацию о клиентах, вести бухгалтерский и управленческий учет.

При расчете экономической составляющей проекта значение интегрального показателя эффективности исполнения разработки превысил максимальный балл в системе оценивания. Это значит, что результат работы можно считать положительным, так как оценка интегрального показателя ресурсоэффективности близка к максимальной, при этом стоимость разработки ниже, чем у ряда аналогов, рассмотренных при анализе конкурентных решений.

Список публикаций

1. Shin M.V. Designing and Development of Information System for Sport Club Management // Информационные технологии в науке, управлении, социальной сфере и медицине: Сборник трудов III Международной научной конференции / Национальный исследовательский Томский политехнический университет. – Томск, 2016.
2. Былина Т.А., Куликова М.О., Мокина Е.Е. Разработка системы информационной поддержки деятельности спортивного клуба // Информационные технологии в науке, управлении, социальной сфере и медицине: Сборник трудов III Международной научной конференции / Национальный исследовательский Томский политехнический университет. – Томск, 2016.
3. Куликова М.О. Автоматизация и интеграция бизнес-процессов предприятия на платформе 1С // Молодежь и современные информационные технологии: Сборник трудов XIII Международной научно-практической конференции студентов, аспирантов и молодых учёных: в 2 томах / Национальный исследовательский Томский политехнический университет, Институт кибернетики (ИК); под ред. Т. Е. Мамоновой [и др.]. – Томск, 2016. – С. 135-136.

Список использованных источников

1. 1С:Предприятие. Конфигурирование и администрирование. – М.: ООО «1С-Паблишинг», 2001.
2. Белов, Сергей Викторович. Безопасность жизнедеятельности и защита окружающей среды (техносферная безопасность) [Электронный ресурс] : учебник для бакалавров / С. В. Белов. — 4-е изд. — Мультимедиа ресурсы (10 директорий; 100 файлов; 740MB). — Москва: Юрайт, 2013. — 1 Мультимедиа CD-ROM. — Бакалавр. Базовый курс. — Бакалавр. Углубленный курс. — Электронные учебники издательства Юрайт. — Электронная копия печатного издания. — Доступ из корпоративной сети ТПУ. — Системные требования: Pentium 100 MHz, 16 Mb RAM, Windows 95/98/NT/2000, CDROM, SVGA, звуковая карта, Internet Explorer 5.0 и выше.Схема доступа: http://www.lib.tpu.ru/fulltext2/m/2013/FN/fn-2440.pdf
3. Буч Г., Якобсон А., Рамбо Дж UML 2.0 СПб.: Питер, 2006, 735 с.
4. Волкова В.Н. Информационные системы: Учеб. пособие / Под ред. В.Н. Волковой, Б.И. Кузина. – СПб.: СПбГТУ, – 2001. – 216 с.
5. Гвоздева Т.В. Проектирование информационных систем: учеб. пособие / Т.В. Гвоздева, Б.А. Баллод. – Ростов н/Д: Феникс, 2009. –508 с.
6. ГОСТ 12.1.003-83 ССБТ. Шум. Общие требования безопасности.
7. ГОСТ 17.4.3.04-85. Охрана природы. Почвы. Общие требования к контролю и охране от загрязнения.
8. ГОСТ Р 12.1.009-2009 Система стандартов безопасности труда. Электробезопасность. Термины и определения.
9. ГОСТ Р 12.1.019-2009 Система стандартов безопасности труда. Электробезопасность. Общие требования и номенклатура видов защиты.
10. Душин В.К. Теоретические основы информационных процессов и систем: Учебник / В.К. Душин. – М.: Издательско-торговая корпорация «Дашков и К°», 2006. – 348 с.
11. Жуков, Виктор Ильич. Защита и безопасность в чрезвычайных ситуациях : учебное пособие / В. И. Жуков, Л. Н. Горбунова; Сибирский

федеральный университет (СФУ). — Москва; Красноярск: Инфра-М Изд-во СФУ, 2014. — 392 с.: ил. — Высшее образование. Бакалавриат. — Библиогр.: с. 384-387.

12. Избачков Ю.С. Информационные системы: Учебник для вузов / Ю.С. Избачков, В.Н. Петров. – 2-е изд. – СПБ.: Питер, 2005. – 656 с.
13. Колесов Ю.Б. Моделирование систем. Объектно-ориентированный подход. Учебное пособие / Ю.Б. Колесов, Ю.Б. Сениченков. – СПБ.: БХВ-Петербург, 2006. – 192 с.
14. Кузьмина Е.А, Кузьмин А.М. Методы поиска новых идей и решений "Методы менеджмента качества" №1, - М., 2003 г.
15. Кузьмина Е.А, Кузьмин А.М. Функционально-стоимостный анализ. Экскурс в историю. "Методы менеджмента качества" №7, - М., 2002 г.
16. Назаренко, Ольга Брониславовна. Безопасность жизнедеятельности : учебное пособие / О. Б. Назаренко, Ю. А. Амелькович; Национальный исследовательский Томский политехнический университет (ТПУ). — 3-е изд., перераб. и доп. — Томск: Изд-во ТПУ, 2013. — 177 с
17. Основы функционально-стоимостного анализа: Учебное пособие / Под ред. М.Г. Карпунина и Б.И. Майданчика. - М.: Энергия, 1980. - 175 с.
18. Профессиональная разработка в системе «1С:Предприятие 8» в 2-х томах. – М.: ООО «1С-Паблишинг»; СПб.: Питер, 2012. – 808 с.: ил.
19. Радченко М.Г. 1С:Предприятие 8.2 Практическое пособие разработчика. Примеры и типовые приему / М. Г. Радченко, Е. Ю. Хрусталева. – М.: ООО «1С-Паблишинг», 2009. – 874 с.: ил.
20. СанПиН 2.2.1/2.1.1.1278-03 Гигиенические требования к естественному, искусственному и совмещенному освещению жилых и общественных зданий
21. СанПиН 2.2.2/2.4.1340-03 Гигиенические требования к персональным электронно-вычислительным машинам и организации работы
22. СанПиН 2.2.4.1191-03

23. Скворцов Ю.В. Организационно-экономические вопросы в дипломном проектировании: Учебное пособие. – М.: Высшая школа, 2006. – 399 с.
24. СНиП 41-01-2003 Отопление, вентиляция и кондиционирование.
25. Тимофеев Г.С., Шумейко Д.А. Конфигурирование и администрирование 1С:Предприятия. Серия «Учебный курс». Ростов н/Д: Феникс, 2001. – 320 с.
26. Фаулер М., Скотт К UML. Основы. – М.: СПб: Символ, 2006, 184 с.
27. Черемных, С.В. Моделирование и анализ систем. IDEF-технологии/ С.В. Черемных, И.О. Семенов, В.С. Ручкин. – М.: Финансы и статистика, 2001. – 208с.

Приложение А

```
ОБЩИЙ МОДУЛЬ УправлениеРасписанием
Функция ДатаПлюсВремя(ПараметрДата, ПараметрВремя) Экспорт
	Возврат ПараметрДата + Час(ПараметрВремя) * 60 * 60 + Минута(ПараметрВремя)*60 +
Секунда(ПараметрВремя);
КонецФункции

Функция ПодготовитьМассивПараметров(ВходящиеПараметры) Экспорт
	Результат = Новый Массив;

	МассивДнейНедели =  ВходящиеПараметры.МассивДнейНедели;
	МассивВремениНачала =  ВходящиеПараметры.МассивВремениНачала;
	МассивВремениОкончания =  ВходящиеПараметры.МассивВремениОкончания;

	ТекДата = НачалоДня(ВходящиеПараметры.ДатаНачалаПериода);
	Пока ТекДата <= КонецДня(ВходящиеПараметры.ДатаОкончанияПериода) Цикл
		н = МассивДнейНедели.Найти(ДеньНедели(ТекДата));
		Если НЕ н = Неопределено Тогда
			СтруктураПараметров = Новый Структура("ДатаНачала, ДатаОкончания,
ВремяНачала, ВремяОкончания, Помещение, Тренер, ГруппаСпортсмен, Услуга");
			ЗаполнитьЗначенияСвойств(СтруктураПараметров, ВходящиеПараметры,
"Помещение, Тренер, ГруппаСпортсмен, Услуга");
			ДатаНачала = УправлениеРасписанием.ДатаПлюсВремя(ТекДата,
МассивВремениНачала[н]);
			ДатаОкончания = УправлениеРасписанием.ДатаПлюсВремя(ТекДата,
МассивВремениОкончания[н]);
			СтруктураПараметров.ДатаНачала = ДатаНачала;
			СтруктураПараметров.ДатаОкончания = ДатаОкончания;
			СтруктураПараметров.ВремяНачала = МассивВремениНачала[н];
			СтруктураПараметров.ВремяОкончания = МассивВремениОкончания[н];
			Результат.Добавить(СтруктураПараметров);
		КонецЕсли;

		ТекДата = ТекДата + 60*60*24;
	КонецЦикла;

	Возврат Результат;
КонецФункции

//ПараметрыЗаписей - массив содержащий параметры записей расписания
Процедура ЗаписатьРасписанияЗанятий(ПараметрыЗаписей, Отказ) Экспорт
	Если Отказ Тогда
		Возврат;
	КонецЕсли;
	Для каждого ПараметрыЗаписи из ПараметрыЗаписей Цикл
		Если Не Отказ Тогда
			ЗаписатьРассписаниеЗанятия(ПараметрыЗаписи, Отказ);
		КонецЕсли;
	КонецЦикла;
КонецПроцедуры

//ПараметрыЗаписи - структура прараметров записи расписания
Процедура ЗаписатьРассписаниеЗанятия(ПараметрыЗаписи, Отказ) Экспорт

	Если ЗначениеЗаполнено(ПараметрыЗаписи.Ссылка) Тогда
		ДокументЗанятие = ПараметрыЗаписи.Ссылка.ПолучитьОбъект();
	Иначе
		ДокументЗанятие = Документы.ПланированиеЗанятий.СоздатьДокумент();
	КонецЕсли;

	ДокументЗанятие.Дата = НачалоДня(ПараметрыЗаписи.ДатаНачала);
```

```
		//DAY 30.06.2015
		//ДокументЗанятие.Статус = Перечисления.СтатусыЗанятий.Запланировано;
		//DAY END
		ЗаполнитьЗначенияСвойств(ДокументЗанятие, ПараметрыЗаписи);
		Попытка
			ДокументЗанятие.Записать(РежимЗаписиДокумента.Проведение);
		Исключение
			Отказ = Истина;
		КонецПопытки;
КонецПроцедуры

//ПараметрыЗаписи - структура прараметров записи расписания
Процедура ПроверитьВозможностьЗаписиРасписания(ПараметрыЗаписи, Отказ, ПараметрОбъект,
НовоеРасписание = Истина) Экспорт
		Для н = 0 По (ПараметрыЗаписи.МассивДнейНедели.Количество() - 1) Цикл
			Запрос = Новый Запрос;
			Запрос.Текст = "ВЫБРАТЬ
				|	РасписаниеЗанятий.Помещение,
				|	РасписаниеЗанятий.Тренер,
				|	РасписаниеЗанятий.ГруппаСпортсмен,
				|	РасписаниеЗанятий.ДатаНачала,
				|	РасписаниеЗанятий.ДатаОкончания,
				|	РасписаниеЗанятий.ВремяНачала,
				|	РасписаниеЗанятий.ВремяОкончания
				|ИЗ
				|	РегистрСведений.РасписаниеЗанятий КАК РасписаниеЗанятий
				|ГДЕ
				|	РасписаниеЗанятий.Помещение = &Помещение
				|	И РасписаниеЗанятий.ДатаНачала >= &ДатаНачала
				|	И РасписаниеЗанятий.ДатаОкончания <= &ДатаОкончания
				|	И (РасписаниеЗанятий.ВремяНачала МЕЖДУ &ВремяНачала И
&ВремяОкончания
				|		ИЛИ РасписаниеЗанятий.ВремяОкончания МЕЖДУ
&ВремяНачала И &ВремяОкончания)
				|	И ДЕНЬНЕДЕЛИ(РасписаниеЗанятий.ДатаНачала) = &ДеньНедели" +
					?(НЕ НовоеРасписание, "
				|	И НЕ РасписаниеЗанятий.ГруппаСпортсмен = &ГруппаСпортсмен", "");
			Запрос.УстановитьПараметр("Помещение", ПараметрыЗаписи.Помещение);
			Запрос.УстановитьПараметр("ДатаНачала", ПараметрыЗаписи.ДатаНачалаПериода);
			Запрос.УстановитьПараметр("ДатаОкончания", ПараметрыЗаписи.ДатаОкончанияПериода);
			Запрос.УстановитьПараметр("ВремяНачала", ПараметрыЗаписи.МассивВремениНачала[н]);
			Запрос.УстановитьПараметр("ВремяОкончания",
ПараметрыЗаписи.МассивВремениОкончания[н]);
			Запрос.УстановитьПараметр("ДеньНедели", ПараметрыЗаписи.МассивДнейНедели[н]);
			Если НЕ НовоеРасписание Тогда
				Запрос.УстановитьПараметр("ГруппаСпортсмен",
ПараметрыЗаписи.ГруппаСпортсмен);
			КонецЕсли;


			Выборка = Запрос.Выполнить().Выбрать();
			Пока Выборка.Следующий() Цикл
				ТекстСообщения = НСтр("ru = 'Помещение " + ПараметрыЗаписи.Помещение + "
занято " + Формат(Выборка.ДатаНачала, "ДФ=dd.MM.yy") + " с " + Формат(Выборка.ВремяНачала,
"ДФ=ЧЧ:мм") + " по "  + Формат(Выборка.ВремяОкончания, "ДФ=ЧЧ:мм") + " !'");
				УправлениенебольшойФирмойСервер.СообщитьОбОшибке(ПараметрОбъект,
ТекстСообщения, , , , Отказ);
				Отказ = Истина;
			КонецЦикла;
		КонецЦикла;
КонецПроцедуры
```

```
Процедура ГрупповоеИзменениеЗанятий(Отказ, ПараметрОбъект) Экспорт
	МассивДнейНедели = Новый Массив;
	МассивВремениНачала = Новый Массив;
	МассивВремениОкончания = Новый Массив;


	МассивДнейНедели.Добавить(ДеньНедели(ПараметрОбъект.Дата));
	МассивВремениНачала.Добавить(ПараметрОбъект.ВремяНачала);
	МассивВремениОкончания.Добавить(ПараметрОбъект.ВремяОкончания);

	ЗначенияПараметров = Новый Структура;
	ЗначенияПараметров.Вставить("ДатаНачалаПериода", НачалоДня(ПараметрОбъект.Дата));
	ЗначенияПараметров.Вставить("ДатаОкончанияПериода", КонецДня(ПараметрОбъект.Дата +
60*60*24*365*10));
	ЗначенияПараметров.Вставить("МассивДнейНедели", МассивДнейНедели);
	ЗначенияПараметров.Вставить("МассивВремениНачала", МассивВремениНачала);
	ЗначенияПараметров.Вставить("МассивВремениОкончания", МассивВремениОкончания);

	ЗначенияПараметров.Вставить("Помещение", ПараметрОбъект.Помещение);
	ЗначенияПараметров.Вставить("Тренер", ПараметрОбъект.Тренер);
	ЗначенияПараметров.Вставить("ГруппаСпортсмен", ПараметрОбъект.ГруппаСпортсмен);
	ЗначенияПараметров.Вставить("Услуга", ПараметрОбъект.Услуга);
	ЗначенияПараметров.Вставить("ПараметрОбъект", ПараметрОбъект);


	МассивПараметров =
УправлениеРасписанием.ПодготовитьМассивПараметров(ЗначенияПараметров);
	УправлениеРасписанием.ПроверитьВозможностьЗаписиРасписания(ЗначенияПараметров, Отказ,
ПараметрОбъект);
	Если Отказ Тогда
		Возврат;
	КонецЕсли;
	УправлениеРасписанием.ЗаписатьРасписанияЗанятий(МассивПараметров, Отказ);

КонецПроцедуры


ФормаСозданияРасписания
&НаКлиенте
Процедура УстановитьИнтервал(Команда)
	Диалог = Новый ДиалогРедактированияСтандартногоПериода();

	Диалог.Период.ДатаНачала    = ДатаНачалаПериода;
	Диалог.Период.ДатаОкончания = ДатаОкончанияПериода;

	Диалог.Показать(Новый ОписаниеОповещения("УстановитьИнтервалЗавершение", ЭтотОбъект,
Новый Структура("Диалог", Диалог)));
КонецПроцедуры

&НаКлиенте
Процедура УстановитьИнтервалЗавершение(Результат, ДополнительныеПараметры) Экспорт

	Диалог = ДополнительныеПараметры.Диалог;

	Если ЗначениеЗаполнено(Результат) Тогда
		ДатаНачалаПериода = Диалог.Период.ДатаНачала;
		ДатаОкончанияПериода = Диалог.Период.ДатаОкончания;
	КонецЕсли;

КонецПроцедуры

&НаСервере
```

```
Процедура СформироватьНаСервере(ВходящиеПараметры, Отказ)
	МассивПараметров =
УправлениеРасписанием.ПодготовитьМассивПараметров(ВходящиеПараметры);
	УправлениеРасписанием.ПроверитьВозможностьЗаписиРасписания(ВходящиеПараметры, Отказ,
ЭтотОбъект);
	Если Отказ Тогда
		Возврат;
	КонецЕсли;
	УправлениеРасписанием.ЗаписатьРасписанияЗанятий(МассивПараметров, Отказ);
КонецПроцедуры

&НаКлиенте
Процедура Сформировать(Команда)

	Отказ = Ложь;
	ПроверитьЗаполнениеРеквизитовФормы(Отказ);
	Если Отказ Тогда
		Возврат;
	КонецЕсли;

	МассивДнейНедели = Новый Массив;
	МассивВремениНачала = Новый Массив;
	МассивВремениОкончания = Новый Массив;

	Для н = 1 по 7 Цикл
		Если ЭтаФорма["День" + н] Тогда
			МассивДнейНедели.Добавить(н);
			МассивВремениНачала.Добавить(ЭтаФорма["ВремяНачала" + н]);
			МассивВремениОкончания.Добавить(ЭтаФорма["ВремяОкончания" + н]);
		КонецЕсли;
	КонецЦикла;

	ЗначенияПараметров = Новый Структура;
	ЗначенияПараметров.Вставить("ДатаНачалаПериода", ДатаНачалаПериода);
	ЗначенияПараметров.Вставить("ДатаОкончанияПериода", ДатаОкончанияПериода);
	ЗначенияПараметров.Вставить("МассивДнейНедели", МассивДнейНедели);
	ЗначенияПараметров.Вставить("МассивВремениНачала", МассивВремениНачала);
	ЗначенияПараметров.Вставить("МассивВремениОкончания", МассивВремениОкончания);

	ЗначенияПараметров.Вставить("Помещение", Помещение);
	ЗначенияПараметров.Вставить("Тренер", Тренер);
  ЗначенияПараметров.Вставить("ГруппаСпортсмен", ГруппаСпортсмен);
	ЗначенияПараметров.Вставить("Услуга", Услуга);

	МассивОшибок = Новый Массив;
	СформироватьНаСервере(ЗначенияПараметров, Отказ);
	Если Отказ Тогда
		Возврат;
	КонецЕсли;

	Закрыть();
КонецПроцедуры

&НаКлиенте
Процедура ПроверитьЗаполнениеРеквизитовФормы(Отказ)

	Если НЕ ЗначениеЗаполнено(ДатаНачалаПериода) Тогда
		Сообщение = Новый СообщениеПользователю();
		Сообщение.Текст = НСтр("ru = 'Не заполнена дата начала периода!'");
		Сообщение.Поле = "ДатаНачалаПериода";
		Сообщение.Сообщить();
		Отказ = Истина;
```

```
	КонецЕсли;

	Если НЕ ЗначениеЗаполнено(ДатаОкончанияПериода) Тогда
		ТекстСообщения = НСтр("ru = 'Не заполнена дата окончания периода!'");
		УправлениеНебольшойФирмойКлиент.СообщитьОбОшибке(ЭтотОбъект, ТекстСообщения,
, , "ДатаОкончанияПериода", Отказ);
	КонецЕсли;

	Если НЕ ЗначениеЗаполнено(Помещение) Тогда
		ТекстСообщения = НСтр("ru = 'Не заполнено помещение!'");
		УправлениеНебольшойФирмойКлиент.СообщитьОбОшибке(ЭтотОбъект, ТекстСообщения,
, , "Помещение", Отказ);
	КонецЕсли;

	Если НЕ ЗначениеЗаполнено(Тренер) Тогда
		ТекстСообщения = НСтр("ru = 'Не заполнен тренер!'");
		УправлениеНебольшойФирмойКлиент.СообщитьОбОшибке(ЭтотОбъект, ТекстСообщения,
, , "Тренер", Отказ);
	КонецЕсли;

	Если НЕ ЗначениеЗаполнено(ГруппаСпортсмен) Тогда
		ТекстСообщения = НСтр("ru = 'Не заполнена группа или спортсмен!'");
		УправлениеНебольшойФирмойКлиент.СообщитьОбОшибке(ЭтотОбъект, ТекстСообщения,
, , "ГруппаСпортсмен", Отказ);
	КонецЕсли;

	Если НЕ ЗначениеЗаполнено(Услуга) Тогда
		ТекстСообщения = НСтр("ru = 'Не заполнена услуга!'");
		УправлениеНебольшойФирмойКлиент.СообщитьОбОшибке(ЭтотОбъект, ТекстСообщения,
, , "Услуга", Отказ);
	КонецЕсли;

	ЕстьОтмеченныйДень = Ложь;
	Для н = 1 по 7 Цикл
		Если ЭтаФорма["День" + н] Тогда
			ЕстьОтмеченныйДень = Истина;
			Если НЕ ЗначениеЗаполнено(ЭтаФорма["ВремяНачала" + н]) Тогда
				ТекстСообщения = НСтр("ru = 'Не заполнено время начала!'");
				УправлениеНебольшойФирмойКлиент.СообщитьОбОшибке(ЭтотОбъект,
ТекстСообщения, , , "ВремяНачала" + н, Отказ);
			КонецЕсли;

			Если НЕ ЗначениеЗаполнено(ЭтаФорма["ВремяОкончания" + н]) Тогда
				ТекстСообщения = НСтр("ru = 'Не заполнено время окончания!'");
				УправлениеНебольшойФирмойКлиент.СообщитьОбОшибке(ЭтотОбъект,
ТекстСообщения, , , "ВремяОкончания" + н, Отказ);
			КонецЕсли;
		КонецЕсли;
	КонецЦикла;

	Если НЕ ЕстьОтмеченныйДень Тогда
		ТекстСообщения = НСтр("ru = 'Не выбран ни один день недели!'");
		УправлениеНебольшойФирмойКлиент.СообщитьОбОшибке(ЭтотОбъект, ТекстСообщения,
, , , Отказ);
	КонецЕсли;
КонецПроцедуры

ДОКУМЕНТ ИзменениеАбонемента Модуль объекта
Процедура ОбработкаПроведения(Отказ, Режим)
	Если ВидОперации = Перечисления.ВидыОперацийИзмененийАбонементов.Блокировка Тогда
		Состояние = Перечисления.СостоянияАбонементов.Заблокирован;
	ИначеЕсли ВидОперации = Перечисления.ВидыОперацийИзмененийАбонементов.Закрытие Тогда
```

```
                Состояние = Перечисления.СостоянияАбонементов.Истек;
                Движения.СоставыГрупп.Записывать = Истина;

                Для Каждого ТекСтрокаАбонементы из Абонементы Цикл
                        Движение = Движения.СоставыГрупп.Добавить();
                        Движение.Период =ТекСтрокаАбонементы.НачалоИзменения;
                        Движение.Группа = ТекСтрокаАбонементы.Группа;
                        Движение.Спортсмен = ТекСтрокаАбонементы.Спортсмен;
                        Движение.Абонемент = ТекСтрокаАбонементы.Абонемент;
                        Движение.Активен = Ложь;
                КонецЦикла;

        ИначеЕсли ВидОперации =
Перечисления.ВидыОперацийИзмененийАбонементов.Приостановление Тогда
                Состояние = Перечисления.СостоянияАбонементов.Приостановлен;

                Запрос = Новый Запрос;
                Запрос.МенеджерВременныхТаблиц = Новый МенеджерВременныхТаблиц;
                Запрос.Текст =
                        "ВЫБРАТЬ РАЗЛИЧНЫЕ
                        |       ИзменениеАбонементаАбонементы.Абонемент КАК Абонемент
                        |ПОМЕСТИТЬ ДокТЧАбонементы
                        |ИЗ
                        |       Документ.ИзменениеАбонемента.Абонементы КАК
ИзменениеАбонементаАбонементы
                        |ГДЕ
                        |       ИзменениеАбонементаАбонементы.Ссылка = &Ссылка
                        |
                        |ИНДЕКСИРОВАТЬ ПО
                        |       Абонемент
                        |;
                        |
                        |////////////////////////////////////////////////////////////////////////////////
                        |ВЫБРАТЬ
                        |       ДокТЧАбонементы.Абонемент
                        |ИЗ
                        |       ДокТЧАбонементы КАК ДокТЧАбонементы";
                Запрос.УстановитьПараметр("Ссылка", Ссылка);
                Результат = Запрос.Выполнить();

                Блокировка = Новый БлокировкаДанных();
                ЭлБлок = Блокировка.Добавить("РегистрНакопления.УчетПриостановокАбонемента");
                ЭлБлок.Режим = РежимБлокировкиДанных.Исключительный;
                ЭлБлок.ИсточникДанных = Результат;
                ЭлБлок.ИспользоватьИзИсточникаДанных("Абонемент", "Абонемент");
                Блокировка.Заблокировать();

                Если Режим = РежимПроведенияДокумента.Оперативный Тогда
                        Движения.УчетПриостановокАбонемента.БлокироватьДляИзменения = Истина;
                        Движения.УчетПриостановокАбонемента.Записать();

                        МоментВремени = Неопределено;
                Иначе
                        МоментВремени = МоментВремени();
                КонецЕсли;

                Запрос.Текст =
                        "ВЫБРАТЬ
                        |       ДокТЧАбонементы.Абонемент,
                        |
        ЕСТЬNULL(УчетПриостановокАбонементаОстатки.КоличествоПриостановокОстаток, 0) КАК
КоличествоПриостановок,
```

```
                        |        ПРЕДСТАВЛЕНИЕ(ДокТЧАбонементы.Абонемент)
                        |ИЗ
                        |        ДокТЧАбонементы КАК ДокТЧАбонементы
                        |            ЛЕВОЕ СОЕДИНЕНИЕ
РегистрНакопления.УчетПриостановокАбонемента.Остатки(
                        |                                    &МоментВремени,
                        |                                    Абонемент В
                        |                                        (ВЫБРАТЬ
                        |                                            ДокТЧАбонементы.Абонемент
                        |                                        ИЗ
                        |                                            ДокТЧАбонементы КАК
ДокТЧАбонементы)) КАК УчетПриостановокАбонементаОстатки
                        |            ПО ДокТЧАбонементы.Абонемент =
УчетПриостановокАбонементаОстатки.Абонемент";
        Запрос.УстановитьПараметр("МоментВремени", МоментВремени());
        Запрос.УстановитьПараметр("Ссылка", Ссылка);
        Результат = Запрос.Выполнить();

        Выборка = Результат.Выбрать();

        Пока Выборка.Следующий() Цикл
            КоличествоПриостановокОстаток = Выборка.КоличествоПриостановок;
            Если КоличествоПриостановокОстаток = 0 Тогда
                Отказ = Истина;
                Сообщить("Для абонемента - " + Выборка.АбонементПредставление + "
больше не предусмотренно приостановок!");
            КонецЕсли;
        КонецЦикла;

        Если Отказ Тогда
            Возврат;
        КонецЕсли;

        Выборка.Сбросить();

        Пока Выборка.Следующий() Цикл
            Движение = Движения.УчетПриостановокАбонемента.ДобавитьРасход();
            Движение.Период = Дата;
            Движение.Абонемент = Выборка.Абонемент;
            Движение.КоличествоПриостановок = 1;
        КонецЦикла;
    ИначеЕсли ВидОперации = Перечисления.ВидыОперацийИзмененийАбонементов.Возобновление
Тогда
        Состояние = Перечисления.СостоянияАбонементов.Активен;
    ИначеЕсли ВидОперации = Перечисления.ВидыОперацийИзмененийАбонементов.Разблокировка
Тогда
        Состояние = Перечисления.СостоянияАбонементов.Активен;
    ИначеЕсли ВидОперации = Перечисления.ВидыОперацийИзмененийАбонементов.Передача Тогда
        Движения.СоставыГрупп.Записывать=Истина;
        Движения.СоставыГрупп.Очистить();

        Для Каждого Стр Из Абонементы Цикл
            НовыйВладелец = Стр.НовыйВладелец;
            СтарыйВладелец=Стр.Спортсмен;
            Группа = Стр.Группа;
            Если ЗначениеЗаполнено(НовыйВладелец) Тогда
                Абонемент = Стр.Абонемент;


                Запрос = Новый Запрос;
                Запрос.Текст =
                    "ВЫБРАТЬ ПЕРВЫЕ 1
```

```
                                        |       КлубныеКарты.Ссылка
                                        |ИЗ
                                        |       Справочник.КлубныеКарты КАК КлубныеКарты
                                        |ГДЕ
                                        |       НЕ КлубныеКарты.ПометкаУдаления
                                        |       И КлубныеКарты.Владелец = &Владелец";

                                Запрос.УстановитьПараметр("Владелец", НовыйВладелец);

                                Результат = Запрос.Выполнить();

                                Если НЕ Результат.Пустой() Тогда
                                        Выборка = Результат.Выбрать();
                                        Выборка.Следующий();

                                        КартаНовогоВладельца = Выборка.Ссылка;
                                Иначе
                                        КартаНовогоВладельца =
Справочники.КлубныеКарты.ПустаяСсылка();
                                КонецЕсли;

                                ЭлАбонемент = Абонемент.ПолучитьОбъект();
                                ЭлАбонемент.Владелец = НовыйВладелец;
                                ЭлАбонемент.Карта = КартаНовогоВладельца;
                                ЭлАбонемент.Записать();

                        // Регистр СоставыГрупп удаление старого спортсмена из группы

                        Движение = Движения.СоставыГрупп.Добавить();
                        Движение.Период = Дата;
                        Движение.Абонемент = Абонемент;
                        Движение.Активен = Ложь;
                        Движение.Группа = Группа;
                        Движение.Спортсмен = СтарыйВладелец;

                        // Регистр СоставыГрупп добавление нового спортсмена в группы

                        Движение = Движения.СоставыГрупп.Добавить();
                        Движение.Период = Дата;
                        Движение.Абонемент = Абонемент;
                        Движение.Активен = Истина;
                        Движение.Группа = Группа;
                        Движение.Спортсмен = НовыйВладелец;

                        Иначе
                                Отказ = Истина;
                                Сообщить("Заполнены не все новые владельцы абонементов!");
                                Возврат;
                        КонецЕсли;
                КонецЦикла;

                Состояние = Перечисления.СостоянияАбонементов.Активен;


        ИначеЕсли ВидОперации = Перечисления.ВидыОперацийИзмененийАбонементов.Продление Тогда
                Состояние = Перечисления.СостоянияАбонементов.Активен;
        Иначе
                Отказ = Истина;
                Сообщить("Не указан вид операции!");
                Возврат;
        КонецЕсли;
        // регистр СостоянияАбонементов
```

```
	Движения.СостоянияАбонементов.Записывать = Истина;
	Движения.СостоянияАбонементов.Очистить();

	Для Каждого ТекСтрокаАбонементы Из Абонементы Цикл

		Если ВидОперации = Перечисления.ВидыОперацийИзмененийАбонементов.Продление
Тогда
			Движение = Движения.СостоянияАбонементов.Добавить();
			Движение.Период = Дата;
			Движение.Абонемент = ТекСтрокаАбонементы.Абонемент;
			Движение.СостояниеАбонемента = Состояние;
			Движение.СрокОкончания = ТекСтрокаАбонементы.ОкончаниеИзменения;
			Движение.ДатаОграничения = ТекСтрокаАбонементы.ОкончаниеИзменения;
		Иначе
			Движение = Движения.СостоянияАбонементов.Добавить();
			Движение.Период = ТекСтрокаАбонементы.НачалоИзменения;
			Движение.Абонемент = ТекСтрокаАбонементы.Абонемент;
			Движение.СостояниеАбонемента = Состояние;
			Если ТекСтрокаАбонементы.АктивироватьПоОкончании и
ЗначениеЗаполнено(ТекСтрокаАбонементы.ОкончаниеИзменения) Тогда
				Движение = Движения.СостоянияАбонементов.Добавить();
				Движение.Период = ТекСтрокаАбонементы.ОкончаниеИзменения;
				Движение.Абонемент = ТекСтрокаАбонементы.Абонемент;
				Движение.СостояниеАбонемента =
Перечисления.СостоянияАбонементов.Активен;
			КонецЕсли;
		КонецЕсли;

	КонецЦикла;
	КонецПроцедуры
ДОКУМЕНТ ИзменениеАбонемента ФормаДокумента

&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)
	УправлениеНебольшойФирмойСервер.ЗаполнитьШапкуДокумента(
		Объект,
		,
		Параметры.ЗначениеКопирования,
		Параметры.Основание,
		РазрешеноПроведение,
		Параметры.ЗначенияЗаполнения
	);

	Если Параметры.Свойство("Спортсмен") Тогда
		НоваяСтрока = Объект.Абонементы.Добавить();
		НоваяСтрока.Спортсмен = Параметры.Спортсмен;
	КонецЕсли;
	Если Параметры.Свойство("ВидОперации") Тогда
		Объект.ВидОперации = Параметры.ВидОперации;
	КонецЕсли;
	Если Параметры.Свойство("Абонемент") Тогда
		НоваяСтрока = Объект.Абонементы.Добавить();
		Абонемент = Параметры.Абонемент;
		НоваяСтрока.Абонемент = Абонемент;
		НоваяСтрока.Спортсмен = Абонемент.Владелец;
		НоваяСтрока.НачалоИзменения = ТекущаяДата();
	КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура ПередЗакрытием(Отказ, СтандартнаяОбработка)
```

```
        ЭтаФорма.Закрыть(Объект.Абонементы[0].Спортсмен);
КонецПроцедуры

&НаКлиенте
Процедура ПриОткрытии(Отказ)
        УправлениеВидимостью();
КонецПроцедуры

&НаКлиенте
Процедура ВидОперацииПриИзменении(Элемент)
        УправлениеВидимостью();
КонецПроцедуры

&НаКлиенте
Процедура УправлениеВидимостью()
        Элементы.АбонементыНачалоИзменения.Видимость  = Истина;
        Элементы.АбонементыОкончаниеИзменения.Видимость= Истина;
        Элементы.АбонементыАктивироватьПоОкончании.Видимость= Истина;
        Элементы.АбонементыНовыйВладелец.Видимость=Истина;
        Если  Объект.ВидОперации =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Закрытие") Тогда
                Элементы.АбонементыОкончаниеИзменения.Видимость= Ложь;
                Элементы.АбонементыАктивироватьПоОкончании.Видимость= Ложь;
                Элементы.АбонементыНовыйВладелец.Видимость= Ложь;
                Элементы.АбонементыНачалоИзменения.Заголовок = "Дата Закрытия";
        ИначеЕсли Объект.ВидОперации =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Продление") Тогда
                Элементы.АбонементыНачалоИзменения.Видимость  = Ложь;
                Элементы.АбонементыОкончаниеИзменения.Заголовок = "Срок действия";
                Элементы.АбонементыАктивироватьПоОкончании.Видимость = Ложь;
                Элементы.АбонементыНовыйВладелец.Видимость=Ложь;
        ИначеЕсли Объект.ВидОперации  =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Приостановление")
Тогда
                Элементы.АбонементыНовыйВладелец.Видимость= Ложь;
                Элементы.АбонементыНачалоИзменения.Заголовок = "Дата Приостановления";
        ИначеЕсли Объект.ВидОперации =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Передача") Тогда
                Элементы.АбонементыОкончаниеИзменения.Видимость=Ложь;
                Элементы.АбонементыАктивироватьПоОкончании.Видимость=Ложь;
                Элементы.АбонементыНачалоИзменения.Заголовок = "Дата Передачи";
        ИначеЕсли Объект.ВидОперации =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Блокировка") Тогда
                Элементы.АбонементыНовыйВладелец.Видимость= Ложь;
                Элементы.АбонементыНачалоИзменения.Заголовок = "Дата Блокировки";
        ИначеЕсли Объект.ВидОперации =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Возобновление") Тогда
                Элементы.АбонементыОкончаниеИзменения.Видимость=Ложь;
                Элементы.АбонементыАктивироватьПоОкончании.Видимость=Ложь;
                Элементы.АбонементыНовыйВладелец.Видимость=Ложь;
                Элементы.АбонементыНачалоИзменения.Заголовок = "Дата Возобновления";
        Иначе
                Элементы.АбонементыОкончаниеИзменения.Видимость=Ложь;
                Элементы.АбонементыАктивироватьПоОкончании.Видимость=Ложь;
                Элементы.АбонементыНовыйВладелец.Видимость=Ложь;
                Элементы.АбонементыНачалоИзменения.Заголовок = "Дата Разблокировки";
        КонецЕсли;

//Объект.ВидОперации =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Разблокировка") Тогда

КонецПроцедуры
```

```
&НаСервере
Процедура АбонементыСпортсменАбонементПриИзменении(Элемент)
	Стр=ЭтаФорма.Абонементы.ТекущиеДанные;
	Если Не УправлениеСпортклубом.ПолучитьГруппуПоАбонементуСпортсмену(Стр.Абонемент,
Стр.Спортсмен, Объект.Дата)= Неопределено Тогда
		Стр.Группа =
УправлениеСпортклубом.ПолучитьГруппуПоАбонементуСпортсмену(Стр.Абонемент, Стр.Спортсмен,
Объект.Дата) ;
	Иначе ПредопределенноеЗначение("Справочник.Группы.ПустаяСсылка");
	КонецЕсли;
КонецПроцедуры


ДОКУМЕНТ ПланированиеЗанятий Модуль объекта
Процедура ОбработкаПроведения(Отказ, РежимПроведения)

	Движения.РасписаниеЗанятий.Записывать = Истина;
	Для Каждого ТекСтрокаРасписание Из Расписание Цикл
		Движение = Движения.РасписаниеЗанятий.Добавить();
		Движение.ДатаНачала = ТекСтрокаРасписание.ДатаНачала;
		Движение.ДатаОкончания = ТекСтрокаРасписание.ДатаНачала;
		Движение.Группа = Группа;
		Движение.Помещение = ТекСтрокаРасписание.Помещение;
		Движение.Услуга = Услуга;
		Движение.Тренер = Тренер;
		Движение.ВремяНачала = ТекСтрокаРасписание.ВремяНачала;
		Движение.ВремяОкончания = ТекСтрокаРасписание.ВремяОкончания;
		Движение.ВидЗанятия = Перечисления.ВидыЗанятий.Групповое;
		Движение.ТипОбъекта = Перечисления.ТипыОбъектовРасписания.Занятие;
		Движение.Организация = Организация;
	КонецЦикла;

КонецПроцедуры

	ДОКУМЕНТ ПланированиеЗанятий ФормаДокумента
&НаКлиенте
Перем СтараяСтрокаРасписания;

&НаКлиенте
Процедура ДобавитьПоДнямНедели(Команда)

	Оповещение = Новый ОписаниеОповещения("ДобавитьРассписание", ЭтаФорма);

	СтруктураПараметров = Новый Структура("Объект, Помещение, Тренер, ГруппаСпортсмен, Услуга,
ДатаНачалаПериода, ДатаОкончанияПериода",
		Объект,
		Объект.Помещение,
		Объект.Тренер,
		Объект.Группа,
		Объект.Услуга,
		Объект.ДатаНачала,
		Объект.ДатаОкончания);

	ОткрытьФорму("Документ.ПланированиеЗанятий.Форма.ФормаСозданияРасписания",
СтруктураПараметров,,,,,Оповещение);
КонецПроцедуры

&НаКлиенте
Процедура ДобавитьРассписание(Результат, Параметры) Экспорт
```

```
	Если ТипЗнч(Результат) = Тип("Структура") Тогда
		ДобавитьРассписаниеНаСервере(Результат);
	КонецЕсли;

КонецПроцедуры

&НаСервере
Процедура ДобавитьРассписаниеНаСервере(Результат)

	МассивПараметров = УправлениеРасписанием.ПодготовитьМассивПараметров(Результат);
	//УправлениеРасписанием.ПроверитьВозможностьЗаписиРасписания(ВходящиеПараметры, Отказ,
ЭтотОбъект);
	//Если Отказ Тогда
	//	Возврат;
	//КонецЕсли;
	//УправлениеРасписанием.ЗаписатьРасписанияЗанятий(МассивПараметров, Отказ);
	ДобавитьСтрокиРасписания(МассивПараметров);

	Если Объект.ДатаОкончания < Результат.ДатаОкончанияПериода Тогда
		Объект.ДатаОкончания = Результат.ДатаОкончанияПериода
	КонецЕсли;

	Если Объект.ДатаНачала > Результат.ДатаНачалаПериода Тогда
		Объект.ДатаНачала = Результат.ДатаНачалаПериода
	КонецЕсли;

КонецПроцедуры

Процедура ДобавитьСтрокиРасписания(МассивПараметров)
	Для каждого ТекПараметр Из МассивПараметров Цикл
		НовоеРасписание = Объект.Расписание.Добавить();
		ЗаполнитьЗначенияСвойств(НовоеРасписание, ТекПараметр);
	КонецЦикла;
	Объект.Расписание.Сортировать("ДатаНачала");
КонецПроцедуры

&НаКлиенте
Процедура УстановитьИнтервал(Команда)
	Диалог = Новый ДиалогРедактированияСтандартногоПериода();

	Диалог.Период.ДатаНачала    = Объект.ДатаНачала;
	Диалог.Период.ДатаОкончания = Объект.ДатаОкончания;

	Диалог.Показать(Новый ОписаниеОповещения("УстановитьИнтервалЗавершение", ЭтотОбъект,
Новый Структура("Диалог", Диалог)));
КонецПроцедуры

&НаКлиенте
Процедура УстановитьИнтервалЗавершение(Результат, ДополнительныеПараметры) Экспорт

	Диалог = ДополнительныеПараметры.Диалог;

	Если ЗначениеЗаполнено(Результат) Тогда
		Объект.ДатаНачала = Диалог.Период.ДатаНачала;
		Объект.ДатаОкончания = Диалог.Период.ДатаОкончания;
	КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура РасписаниеПриОкончанииРедактирования(Элемент, НоваяСтрока, ОтменаРедактирования)
	СтрокаТабличнойЧасти = Элементы.Расписание.ТекущиеДанные;
```

```
        ИзменилосьРасписание(СтрокаТабличнойЧасти);
КонецПроцедуры

&НаКлиенте
Процедура РасписаниеПриНачалеРедактирования(Элемент, НоваяСтрока, Копирование)
        ЗаполнитьЗначенияСвойств(СтараяСтрокаРасписания, Элемент.ТекущиеДанные);
КонецПроцедуры

&НаКлиенте
Процедура ИзменилосьРасписание(СтрокаТабличнойЧасти, УдаленаСтрока = Ложь)
        СтруктураСтрокиТабличнойЧасти = Новый Структура("ДатаНачала, ВремяНачала,
ВремяОкончания, Помещение");
        ЗаполнитьЗначенияСвойств(СтруктураСтрокиТабличнойЧасти, СтрокаТабличнойЧасти);
        СтруктураРазличий =
УправлениеСпортклубомОбщегоНазначения.СравнитьСтруктуры(СтруктураСтрокиТабличнойЧасти,
СтараяСтрокаРасписания);

        Если СтруктураРазличий.Количество() > 0 Тогда
                ПараметрыВопроса = Новый Структура("СтрокаТабличнойЧасти, СтруктураРазличий,
УдаленаСтрока", СтрокаТабличнойЧасти, СтруктураРазличий, УдаленаСтрока);
                ОписаниеОповещения = Новый ОписаниеОповещения("СкорректироватьРасписание",
ЭтотОбъект, ПараметрыВопроса);
                ПоказатьВопрос(ОписаниеОповещения, "Скорректировать расписание на последующие
дни?", РежимДиалогаВопрос.ДаНет);
        КонецЕсли;
КонецПроцедуры

&НаКлиенте
Процедура СкорректироватьРасписание(Ответ, ДополнительныеПараметры) Экспорт

        Если Ответ = КодВозвратаДиалога.Нет Тогда
                Возврат;
        КонецЕсли;

        СтрокаТабличнойЧасти = ДополнительныеПараметры.СтрокаТабличнойЧасти;
        СтруктураРазличий = ДополнительныеПараметры.СтруктураРазличий;
        УдаленаСтрока = ДополнительныеПараметры.УдаленаСтрока;


        НачатьОбработку = Ложь;

        ИзмененныйДеньНедели = ДеньНедели(?(СтруктураРазличий.Свойство("ДатаНачала") и НЕ
УдаленаСтрока, СтруктураРазличий.ДатаНачала, СтрокаТабличнойЧасти.ДатаНачала));
        ВремяНачала = ?(СтруктураРазличий.Свойство("ВремяНачала") и НЕ УдаленаСтрока,
СтруктураРазличий.ВремяНачала, СтрокаТабличнойЧасти.ВремяНачала);
        ВремяОкончания = ?(СтруктураРазличий.Свойство("ВремяОкончания") и НЕ УдаленаСтрока,
СтруктураРазличий.ВремяОкончания, СтрокаТабличнойЧасти.ВремяОкончания);
        Помещение = ?(СтруктураРазличий.Свойство("Помещение") и НЕ УдаленаСтрока,
СтруктураРазличий.Помещение, СтрокаТабличнойЧасти.Помещение);

        Если СтруктураРазличий.Свойство("ДатаНачала") и НЕ УдаленаСтрока Тогда
                РазницаДней = СтрокаТабличнойЧасти.ДатаНачала - СтруктураРазличий.ДатаНачала;
        КонецЕсли;

        МассивУдаленныхСтрок = Новый Массив;

        Для каждого ТекущаяСтрокаРасписания ИЗ Объект.Расписание Цикл

                Если УдаленаСтрока Тогда
                        Если ТекущаяСтрокаРасписания.ДатаНачала >=
СтрокаТабличнойЧасти.ДатаНачала Тогда
                                НачатьОбработку = Истина;
```

```
                        КонецЕсли;
                Иначе
                        Если ТекущаяСтрокаРасписания = СтрокаТабличнойЧасти Тогда
                                НачатьОбработку = Истина;
                                Продолжить;
                        КонецЕсли;
                КонецЕсли;

                Если НЕ НачатьОбработку Тогда
                        Продолжить;
                КонецЕсли;

                Если ДеньНедели(ТекущаяСтрокаРасписания.ДатаНачала) = ИзмененныйДеньНедели
Тогда
                        Если ТекущаяСтрокаРасписания.ВремяНачала = ВремяНачала И
ТекущаяСтрокаРасписания.ВремяОкончания = ВремяОкончания Тогда
                                Если ДополнительныеПараметры.УдаленаСтрока Тогда
                                        МассивУдаленныхСтрок.Добавить(ТекущаяСтрокаРасписания);
                                Иначе
                                        Для каждого Изменение Из СтруктураРазличий  Цикл
                                                Если Изменение.Ключ = "ДатаНачала" Тогда
                                                        ТекущаяСтрокаРасписания[Изменение.Ключ] =
ТекущаяСтрокаРасписания[Изменение.Ключ] + РазницаДней;
                                                Иначе
                                                        ТекущаяСтрокаРасписания[Изменение.Ключ] =
СтрокаТабличнойЧасти[Изменение.Ключ];
                                                КонецЕсли;
                                        КонецЦикла;
                                КонецЕсли;
                        КонецЕсли;
                КонецЕсли;
        КонецЦикла;

        Для каждого УдаленнаяСтрока Из МассивУдаленныхСтрок Цикл
                Объект.Расписание.Удалить(УдаленнаяСтрока);
        КонецЦикла;

        Объект.Расписание.Сортировать("ДатаНачала");

КонецПроцедуры

&НаКлиенте
Процедура РасписаниеПередУдалением(Элемент, Отказ)
        СтрокаТабличнойЧасти = Элементы.Расписание.ТекущиеДанные;
        ИзменилосьРасписание(СтрокаТабличнойЧасти, Истина);
КонецПроцедуры

&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)
        УправлениеНебольшойФирмойСервер.ЗаполнитьШапкуДокумента(
                Объект,
                ,
                Параметры.ЗначениеКопирования,
                Параметры.Основание,
                РазрешеноПроведение,
                Параметры.ЗначенияЗаполнения
        );
КонецПроцедуры

&НаКлиенте
Процедура ГруппаПриИзменении(Элемент)
        ГруппаПриИзмененииНаСервере();
```

```
КонецПроцедуры

&НаСервере
Процедура ГруппаПриИзмененииНаСервере()
	Объект.Услуга = ОбщегоНазначения.ЗначениеРеквизитаОбъекта(Объект.Группа, "Услуга");
КонецПроцедуры
СтараяСтрокаРасписания = Новый Структура("ДатаНачала, ВремяНачала, ВремяОкончания, Помещение");

ДОКУМЕНТ ПланированиеЗанятий ФормаСозданияРасписания
&НаКлиенте
Процедура УстановитьИнтервал(Команда)
	Диалог = Новый ДиалогРедактированияСтандартногоПериода();

	Диалог.Период.ДатаНачала    = ДатаНачалаПериода;
	Диалог.Период.ДатаОкончания = ДатаОкончанияПериода;

	Диалог.Показать(Новый ОписаниеОповещения("УстановитьИнтервалЗавершение", ЭтотОбъект,
Новый Структура("Диалог", Диалог)));
КонецПроцедуры

&НаКлиенте
Процедура УстановитьИнтервалЗавершение(Результат, ДополнительныеПараметры) Экспорт

	Диалог = ДополнительныеПараметры.Диалог;

	Если ЗначениеЗаполнено(Результат) Тогда
		ДатаНачалаПериода = Диалог.Период.ДатаНачала;
		ДатаОкончанияПериода = Диалог.Период.ДатаОкончания;
	КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура Сформировать(Команда)

	Отказ = Ложь;
	ПроверитьЗаполнениеРеквизитовФормы(Отказ);
	Если Отказ Тогда
		Возврат;
	КонецЕсли;

	МассивДнейНедели = Новый Массив;
	МассивВремениНачала = Новый Массив;
	МассивВремениОкончания = Новый Массив;

	Для н = 1 по 7 Цикл
		Если ЭтаФорма["День" + н] Тогда
			МассивДнейНедели.Добавить(н);
			МассивВремениНачала.Добавить(ЭтаФорма["ВремяНачала" + н]);
			МассивВремениОкончания.Добавить(ЭтаФорма["ВремяОкончания" + н]);
		КонецЕсли;
	КонецЦикла;

	ЗначенияПараметров = Новый Структура;
	ЗначенияПараметров.Вставить("ДатаНачалаПериода", ДатаНачалаПериода);
	ЗначенияПараметров.Вставить("ДатаОкончанияПериода", ДатаОкончанияПериода);
	ЗначенияПараметров.Вставить("МассивДнейНедели", МассивДнейНедели);
	ЗначенияПараметров.Вставить("МассивВремениНачала", МассивВремениНачала);
	ЗначенияПараметров.Вставить("МассивВремениОкончания", МассивВремениОкончания);

	ЗначенияПараметров.Вставить("Помещение", Помещение);
	ЗначенияПараметров.Вставить("Тренер", Тренер);
```

```
    ЗначенияПараметров.Вставить("ГруппаСпортсмен", ГруппаСпортсмен);
        ЗначенияПараметров.Вставить("Услуга", Услуга);

        Закрыть(ЗначенияПараметров);

КонецПроцедуры

&НаКлиенте
Процедура ПроверитьЗаполнениеРеквизитовФормы(Отказ)

        Если НЕ ЗначениеЗаполнено(ДатаНачалаПериода) Тогда
                Сообщение = Новый СообщениеПользователю();
                Сообщение.Текст = НСтр("ru = 'Не заполнена дата начала периода!'");
                Сообщение.Поле = "ДатаНачалаПериода";
                Сообщение.Сообщить();
                Отказ = Истина;
        КонецЕсли;

        Если НЕ ЗначениеЗаполнено(ДатаОкончанияПериода) Тогда
                ТекстСообщения = НСтр("ru = 'Не заполнена дата окончания периода!'");
                УправлениеНебольшойФирмойКлиент.СообщитьОбОшибке(ЭтотОбъект, ТекстСообщения,
, , "ДатаОкончанияПериода", Отказ);
        КонецЕсли;

        Если НЕ ЗначениеЗаполнено(Помещение) Тогда
                ТекстСообщения = НСтр("ru = 'Не заполнено помещение!'");
                УправлениеНебольшойФирмойКлиент.СообщитьОбОшибке(ЭтотОбъект, ТекстСообщения,
, , "Помещение", Отказ);
        КонецЕсли;

        Если НЕ ЗначениеЗаполнено(Тренер) Тогда
                ТекстСообщения = НСтр("ru = 'Не заполнен тренер!'");
                УправлениеНебольшойФирмойКлиент.СообщитьОбОшибке(ЭтотОбъект, ТекстСообщения,
, , "Тренер", Отказ);
        КонецЕсли;

        Если НЕ ЗначениеЗаполнено(ГруппаСпортсмен) Тогда
                ТекстСообщения = НСтр("ru = 'Не заполнена группа или спортсмен!'");
                УправлениеНебольшойФирмойКлиент.СообщитьОбОшибке(ЭтотОбъект, ТекстСообщения,
, , "ГруппаСпортсмен", Отказ);
        КонецЕсли;

        Если НЕ ЗначениеЗаполнено(Услуга) Тогда
                ТекстСообщения = НСтр("ru = 'Не заполнена услуга!'");
                УправлениеНебольшойФирмойКлиент.СообщитьОбОшибке(ЭтотОбъект, ТекстСообщения,
, , "Услуга", Отказ);
        КонецЕсли;

        ЕстьОтмеченныйДень = Ложь;
        Для н = 1 по 7 Цикл
                Если ЭтаФорма["День" + н] Тогда
                        ЕстьОтмеченныйДень = Истина;
                        Если НЕ ЗначениеЗаполнено(ЭтаФорма["ВремяНачала" + н]) Тогда
                                ТекстСообщения = НСтр("ru = 'Не заполнено время начала!'");
                                УправлениеНебольшойФирмойКлиент.СообщитьОбОшибке(ЭтотОбъект,
ТекстСообщения, , , "ВремяНачала" + н, Отказ);
                        КонецЕсли;

                        Если НЕ ЗначениеЗаполнено(ЭтаФорма["ВремяОкончания" + н]) Тогда
                                ТекстСообщения = НСтр("ru = 'Не заполнено время окончания!'");
                                УправлениеНебольшойФирмойКлиент.СообщитьОбОшибке(ЭтотОбъект,
ТекстСообщения, , , "ВремяОкончания" + н, Отказ);
```

```
                        КонецЕсли;

                        Если ЭтаФорма["ВремяНачала" + н] >= ЭтаФорма["ВремяОкончания" + н] Тогда
                                ТекстСообщения = НСтр("ru = 'Время окончания должно быть больше
времени начала!'");
                                УправлениеНебольшойФирмойКлиент.СообщитьОбОшибке(ЭтотОбъект,
ТекстСообщения, , , "ВремяОкончания" + н, Отказ);
                        КонецЕсли;

                КонецЕсли;
        КонецЦикла;

        Если НЕ ЕстьОтмеченныйДень Тогда
                ТекстСообщения = НСтр("ru = 'Не выбран ни один день недели!'");
                УправлениеНебольшойФирмойКлиент.СообщитьОбОшибке(ЭтотОбъект, ТекстСообщения,
, , , Отказ);
        КонецЕсли;
КонецПроцедуры

&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

        ТекЗначение = ДанныеФормыВЗначение(Параметры.Объект,
Тип("ДокументОбъект.ПланированиеЗанятий"));
        ЗначениеВРеквизитФормы(ТекЗначение, "Объект");
        Помещение = Параметры.Помещение;
        Тренер = Параметры.Тренер;
        ГруппаСпортсмен = Параметры.ГруппаСпортсмен;
        Услуга = Параметры.Услуга;
        ДатаНачалаПериода = Макс(ТекущаяДата(), Параметры.ДатаНачалаПериода);
        ДатаОкончанияПериода = Параметры.ДатаОкончанияПериода;

КонецПроцедуры

ДОКУМЕНТ Посещения Модуль объекта
Процедура ОбработкаПроведения(Отказ, Режим)
        // регистр ПосещенияСпортсменов
        Движения.ПосещенияСпортсменов.Записывать = Истина;

        Движение = Движения.ПосещенияСпортсменов.Добавить();
        Движение.Период = Дата;
        Движение.ДатаВремяВхода = ДатаВремяВхода;
        Движение.ДатаВремяВыхода = ДатаВремяВыхода;
        Движение.Организация = Организация;
        Движение.Спортсмен = Спортсмен;
        Движение.СостояниеСпортсмена = ?(ЗначениеЗаполнено(ДатаВремяВыхода),
Перечисления.СостоянияСпортсменов.НеВКлубе, Перечисления.СостоянияСпортсменов.ВКлубе);
        Движение.Ячейка = Ячейка;
        Движение.СтруктурнаяЕдиница = СтруктурнаяЕдиница;
        Движение.ДокументПосещения = Ссылка;

        // регистр АрендаИнвентаря Приход
        Движения.АрендаИнвентаря.Записывать = Истина;
        Для Каждого ТекСтрокаВыданныйИнвентарь Из ВыданныйИнвентарь Цикл
                Движение = Движения.АрендаИнвентаря.Добавить();
                Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
                Движение.Период = Дата;
                Движение.Спорстмен = Спортсмен;
                Движение.Организация = Организация;
                Движение.Инвентарь = ТекСтрокаВыданныйИнвентарь.Инвентарь;
                Движение.Количество = ТекСтрокаВыданныйИнвентарь.Количество;
        КонецЦикла;
```

```
КонецПроцедуры

ДОКУМЕНТ Посещения ФормаДокумента
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

	Если НЕ ЗначениеЗаполнено(Объект.Ссылка) Тогда
		УправлениеНебольшойФирмойСервер.ЗаполнитьШапкуДокумента(
			Объект,
			,
			Параметры.ЗначениеКопирования,
			Параметры.Основание,
			РазрешеноПроведение,
			Параметры.ЗначенияЗаполнения
		);

		Объект.ДатаВремяВхода = ТекущаяДата();

		ЗаполнитьЗначенияСвойств(Объект, Параметры);
		Объект.Абонемент =
УправлениеСпортклубом.ПолучитьАбонементыСпортсмена(Объект.Дата, Объект.Спортсмен, Истина,
Перечисления.СостоянияАбонементов.Активен);
	КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура СпортсменПриИзменении(Элемент)
	СпортсменПриИзмененииСервер();
КонецПроцедуры

&НаСервере
Процедура СпортсменПриИзмененииСервер()
	Объект.Абонемент = УправлениеСпортклубом.ПолучитьАбонементыСпортсмена(Объект.Дата,
Объект.Спортсмен, Истина, Перечисления.СостоянияАбонементов.Активен);
КонецПроцедуры

&НаКлиенте
Процедура ПриОткрытии(Отказ)
	Если ЗначениеЗаполнено(Объект.ДатаВремяВхода) Тогда
		ЭтаФорма.Элементы.ДатаВремяПосещения.Доступность = Ложь;
	КонецЕсли;
КонецПроцедуры

ДОКУМЕНТ ПродажаАбонемента Модуль объекта
Процедура ОбработкаПроведения(Отказ, Режим)
	СоздатьАбонементы();
	ЗаполнитьГрафикОплат();

	Движения.СостоянияАбонементов.Записывать = Истина;
	Движения.СостоянияАбонементов.Очистить();

	Движения.ЦеныАбонементов.Записывать = Истина;
	Движения.ЦеныАбонементов.Очистить();


	Движения.ВзаиморасчетыПоАбонементам.Записывать = Истина;
	Движения.ВзаиморасчетыПоАбонементам.Очистить();

	Движения.СоставыГрупп.Записывать = Истина;
	Движения.СоставыГрупп.Очистить();
```

```
	Движения.ЗанятияПоАбонементам.Записывать = Истина;
	Движения.ЗанятияПоАбонементам.Очистить();

	Движения.УчетПриостановокАбонемента.Записывать = Истина;
	Движения.УчетПриостановокАбонемента.Очистить();

	Запрос = Новый Запрос;
	Запрос.Текст =
		"ВЫБРАТЬ
		|	ПродажаАбонементаАбонементы.АбонементСпортсмена КАК
АбонементСпортсмена,
		|	СУММА(ПродажаАбонементаВсеУслуги.Сумма) КАК Сумма,
		|	ПродажаАбонементаВсеУслуги.Услуга,
		|	СУММА(ПродажаАбонементаВсеУслуги.Количество) КАК Количество,
		|	ПродажаАбонементаВсеУслуги.Тренер,
		|	ПродажаАбонементаАбонементы.ДлительностьОграничена КАК
ДлительностьОграничена,
		|	ПродажаАбонементаАбонементы.ДлительностьДней КАК ДлительностьДней,
		|	ПродажаАбонементаАбонементы.ДлительностьМесяцев КАК
ДлительностьМесяцев,
		|	ПродажаАбонементаАбонементы.ДлительностьЛет КАК ДлительностьЛет,
		|	ПродажаАбонементаАбонементы.ДатаНачалаДействия КАК ДатаНачалаДействия,
		|	ПродажаАбонементаАбонементы.ДатаОкончанияДействия КАК
ДатаОкончанияДействия,
		|	ПродажаАбонементаАбонементы.ВидАбонемента,
		|	ПродажаАбонементаАбонементы.ВидАбонемента.Организация
		|ИЗ
		|	Документ.ПродажаАбонемента.Абонементы КАК
ПродажаАбонементаАбонементы
		|		ЛЕВОЕ СОЕДИНЕНИЕ Документ.ПродажаАбонемента.ВсеУслуги КАК
ПродажаАбонементаВсеУслуги
		|		ПО ПродажаАбонементаАбонементы.ВидАбонемента =
ПродажаАбонементаВсеУслуги.ВидАбонемента
		|			И ПродажаАбонементаАбонементы.Ссылка =
ПродажаАбонементаВсеУслуги.Ссылка
		|ГДЕ
		|	ПродажаАбонементаАбонементы.Ссылка = &Ссылка
		|	И ПродажаАбонементаВсеУслуги.Ссылка = &Ссылка
		|
		|СГРУППИРОВАТЬ ПО
		|	ПродажаАбонементаАбонементы.АбонементСпортсмена,
		|	ПродажаАбонементаВсеУслуги.Услуга,
		|	ПродажаАбонементаВсеУслуги.Тренер,
		|	ПродажаАбонементаАбонементы.ДлительностьОграничена,
		|	ПродажаАбонементаАбонементы.ДлительностьДней,
		|	ПродажаАбонементаАбонементы.ДлительностьМесяцев,
		|	ПродажаАбонементаАбонементы.ДлительностьЛет,
		|	ПродажаАбонементаАбонементы.ДатаНачалаДействия,
		|	ПродажаАбонементаАбонементы.ДатаОкончанияДействия,
		|	ПродажаАбонементаАбонементы.ВидАбонемента
		|ИТОГИ
		|	СУММА(Сумма),
		|	МАКСИМУМ(ДлительностьОграничена),
		|	МАКСИМУМ(ДлительностьДней),
		|	МАКСИМУМ(ДлительностьМесяцев),
		|	МАКСИМУМ(ДлительностьЛет),
		|	МИНИМУМ(ДатаНачалаДействия),
		|	МАКСИМУМ(ДатаОкончанияДействия)
		|ПО
		|	АбонементСпортсмена
		|;
```

```
                |
                |////////////////////////////////////////////////////////////////////////////
                |ВЫБРАТЬ РАЗЛИЧНЫЕ
                |       ПродажаАбонементаВсеУслуги.Группа,
                |       ПродажаАбонементаАбонементы.АбонементСпортсмена,
                |       ПродажаАбонементаАбонементы.КоличествоЗаморозок
                |ИЗ
                |       Документ.ПродажаАбонемента.ВсеУслуги КАК ПродажаАбонементаВсеУслуги
                |           ЛЕВОЕ СОЕДИНЕНИЕ Документ.ПродажаАбонемента.Абонементы КАК
ПродажаАбонементаАбонементы
                |           ПО ПродажаАбонементаВсеУслуги.Ссылка =
ПродажаАбонементаАбонементы.Ссылка
                |               И ПродажаАбонементаВсеУслуги.ВидАбонемента =
ПродажаАбонементаАбонементы.ВидАбонемента
                |ГДЕ
                |       ПродажаАбонементаВсеУслуги.Ссылка = &Ссылка
                |       И ПродажаАбонементаАбонементы.Ссылка = &Ссылка
                |;
                |
                |////////////////////////////////////////////////////////////////////////////
                |ВЫБРАТЬ
                |       ПродажаАбонементаГрафикОплат.ВидАбонемента,
                |       ПродажаАбонементаГрафикОплат.Абонемент,
                |       ПродажаАбонементаГрафикОплат.ДатаОплаты КАК ДатаОплаты,
                |       ПродажаАбонементаГрафикОплат.Сумма
                |ИЗ
                |       Документ.ПродажаАбонемента.ГрафикОплат КАК
ПродажаАбонементаГрафикОплат
                |ГДЕ
                |       ПродажаАбонементаГрафикОплат.Ссылка = &Ссылка
                |
                |УПОРЯДОЧИТЬ ПО
                |       ДатаОплаты";
        Запрос.УстановитьПараметр("Ссылка", Ссылка);
        Результат = Запрос.ВыполнитьПакет();

        Если НЕ Результат[0].Пустой() Тогда
                ВыборкаИтоги = Результат[0].Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам);

                Пока ВыборкаИтоги.Следующий() Цикл
                        Абонемент = ВыборкаИтоги.АбонементСпортсмена;
                        // регистр СостоянияАбонементов
                        Движение = Движения.СостоянияАбонементов.Добавить();
                        Движение.Период = Дата;
                        Движение.Абонемент = Абонемент;
                        Движение.СрокОкончания = ВыборкаИтоги.ДатаОкончанияДействия;
                        Движение.СостояниеАбонемента = Перечисления.СостоянияАбонементов.Активен;

                        ДлительностьОграничена = ВыборкаИтоги.ДлительностьОграничена;
                        ДлительностьДней = ВыборкаИтоги.ДлительностьДней;
                        ДлительностьМесяцев = ВыборкаИтоги.ДлительностьМесяцев;
                        ДлительностьЛет = ВыборкаИтоги.ДлительностьЛет;

                        Если ВыборкаИтоги.ДлительностьОграничена И
                                (ЗначениеЗаполнено(ДлительностьДней) ИЛИ
                                ЗначениеЗаполнено(ДлительностьМесяцев) ИЛИ
                                ЗначениеЗаполнено(ДлительностьЛет))Тогда
                                ДатаОграничения = НачалоДня(ВыборкаИтоги.ДатаНачалаДействия);
                                ДатаОграничения = ДобавитьМесяц(ДатаОграничения,
ДлительностьМесяцев + 12 * ДлительностьЛет) + 24 * 3600 * ДлительностьДней;

                                Движение.ДатаОграничения = ДатаОграничения;
```

```
                        КонецЕсли;

                        Выборка = ВыборкаИтоги.Выбрать();
                        Пока Выборка.Следующий() Цикл
                                // регистр ЗанятияПоАбонементам Приход
                                Движение = Движения.ЗанятияПоАбонементам.ДобавитьПриход();
                                Движение.Период = Дата;
                                Движение.Абонемент = Абонемент;
                                Движение.Спортсмен = Спортсмен;
                                Движение.Услуга = Выборка.Услуга;
                                Движение.Организация = Организация;
                                Движение.Количество = Выборка.Количество;
                                Движение.ВидНачисленияСписанияЗанятий =
Перечисления.ВидыНачисленийСписанийЗанятий.Продажа;
                                Движение.ДокументПродажи = Ссылка;
                                Движение.ДатаОкончания = Выборка.ДатаОкончанияДействия;

                                //регистр ЦеныАбонементов
                                //НЗ = РегистрыСведений.ЦеныАбонементов.СоздатьНаборЗаписей();
                                //НЗ.Отбор.ВидАбонемента.Установить(Выборка.ВидАбонемента);
                                //НЗ.Отбор.Организация.Значение = Организация;
                                //НЗ.Отбор.Абонемент.Значение = Абонемент;
                                //НЗ.Отбор.Услуга.Значение = Выборка.Услуга;
                                //НЗ.Прочитать();
                                //НЗ[0].ЦенаЗанятия = ?(Выборка.Количество = 0, 0, Выборка.Сумма /
Выборка.Количество);
                                //НЗ.Записать();
                                Движение = Движения.ЦеныАбонементов.Добавить();

                                Движение.Период = Дата;
                                Движение.ВидАбонемента = Выборка.ВидАбонемента;
                                Движение.Организация = Организация;
                                Движение.Абонемент = Абонемент;
                                Движение.Услуга = Выборка.Услуга;
                                Движение.ЦенаЗанятия = ?(Выборка.Количество = 0, 0, Выборка.Сумма /
Выборка.Количество);

                        КонецЦикла;
                КонецЦикла;
        КонецЕсли;

        Если НЕ Результат[1].Пустой() Тогда
                Выборка = Результат[1].Выбрать();

                Пока Выборка.Следующий() Цикл
                        // регистр СоставыГрупп Приход
                        Абонемент = Выборка.АбонементСпортсмена;
                        КоличествоЗаморозок = Выборка.КоличествоЗаморозок;

                        Движение = Движения.СоставыГрупп.Добавить();
                        Движение.Период = Дата;
                        Движение.Группа = Выборка.Группа;
                        Движение.Спортсмен = Спортсмен;
                        Движение.Активен = Истина;
                        Движение.Абонемент = Абонемент;

                        Если ЗначениеЗаполнено(КоличествоЗаморозок) Тогда
                                Движение = Движения.УчетПриостановокАбонемента.ДобавитьПриход();
                                Движение.Период = Дата;
                                Движение.Абонемент = Абонемент;
                                Движение.КоличествоПриостановок = КоличествоЗаморозок;
                        КонецЕсли;
```

```
                КонецЦикла;
        КонецЕсли;

        // регистр ВзаиморасчетыПоАбонементам Приход
        Если НЕ Результат[2].Пустой() Тогда
                Выборка = Результат[2].Выбрать();

                Пока Выборка.Следующий() Цикл
                        Движение = Движения.ВзаиморасчетыПоАбонементам.ДобавитьПриход();
                        Движение.Период = Выборка.ДатаОплаты;
                        Движение.Абонемент = Выборка.Абонемент;
                        Движение.Спортсмен = Спортсмен;
                        Движение.Сумма = Выборка.Сумма;
                КонецЦикла;
        КонецЕсли;

        Если НЕ Отказ Тогда
                СоздатьАктВыполненныхРабот();
        КонецЕсли;

КонецПроцедуры

Процедура СоздатьАбонементы()
        Для Каждого Эл Из Абонементы Цикл
                Если НЕ ЗначениеЗаполнено(Эл.АбонементСпортсмена) Тогда
                        НовыйЭлемент = Справочники.АбонементыСпортсменов.СоздатьЭлемент();
                        НовыйЭлемент.УстановитьНовыйКод();
                        НовыйЭлемент.Наименование = "" + Эл.ВидАбонемента.Наименование + " " +
Спортсмен.Наименование;
                        НовыйЭлемент.ВидАбонемента = Эл.ВидАбонемента;
                        НовыйЭлемент.Владелец = Спортсмен;
                        НовыйЭлемент.Записать();

                        Эл.АбонементСпортсмена = НовыйЭлемент.Ссылка;

                        Отбор = Новый Структура("ВидАбонемента", Эл.ВидАбонемента);
                        Строки = ГрафикОплат.НайтиСтроки(Отбор);
                        Для Каждого Стр Из Строки Цикл
                                Стр.Абонемент = Эл.АбонементСпортсмена;
                        КонецЦикла;
                КонецЕсли;
        КонецЦикла;

        ЭтотОбъект.Записать(РежимЗаписиДокумента.Запись);
КонецПроцедуры

Процедура ПередЗаписью(Отказ, РежимЗаписи, РежимПроведения)
        СуммаДокумента = Абонементы.Итог("Стоимость");
КонецПроцедуры

Процедура СоздатьАктВыполненныхРабот()
        АктВыполненныхРаботСсылка = ПолучитьПрявязанныйАктВыполненныхРабот();
        Если АктВыполненныхРаботСсылка <> Неопределено Тогда
                ДокументОбъект = АктВыполненныхРаботСсылка.ПолучитьОбъект();
        Иначе
                ДокументОбъект = Документы.АктВыполненныхРабот.СоздатьДокумент();
        КонецЕсли;

        Запрос = Новый Запрос;
        Запрос.Текст =
                "ВЫБРАТЬ
                |       Контрагенты.ДоговорПоУмолчанию КАК Договор,
```

```
                |       Контрагенты.ДоговорПоУмолчанию.ВидЦен КАК ВидЦен,
                |       Контрагенты.Ссылка КАК Контрагент
                |ИЗ
                |       Справочник.Контрагенты КАК Контрагенты
                |ГДЕ
                |       Контрагенты.Ссылка = &Ссылка
                |;
                |
                |////////////////////////////////////////////////////////////////////////////////
                |ВЫБРАТЬ
                |       КурсыВалютСрезПоследних.Валюта,
                |       КурсыВалютСрезПоследних.Курс,
                |       КурсыВалютСрезПоследних.Кратность
                |ИЗ
                |       РегистрСведений.КурсыВалют.СрезПоследних(
                |                       ,
                |                       Валюта В
                |                               (ВЫБРАТЬ
                |                                       ВалютаУчета.Значение
                |                               ИЗ
                |                                       Константа.ВалютаУчета КАК ВалютаУчета)) КАК
КурсыВалютСрезПоследних
                |;
                |
                |////////////////////////////////////////////////////////////////////////////////
                |ВЫБРАТЬ
                |       НастройкиПользователей.Значение КАК Подразделение,
                |       НастройкиПользователей.Пользователь
                |ИЗ
                |       РегистрСведений.НастройкиПользователей КАК НастройкиПользователей
                |ГДЕ
                |       НастройкиПользователей.Пользователь = &Пользователь
                |       И НастройкиПользователей.Настройка = &Настройка";
        Запрос.УстановитьПараметр("Ссылка", Спортсмен.Контрагент);
        Запрос.УстановитьПараметр("Пользователь", ПараметрыСеанса.ТекущийПользователь);
        Запрос.УстановитьПараметр("Настройка",
ПланыВидовХарактеристик.НастройкиПользователей.ОсновноеПодразделение);
        Результат = Запрос.ВыполнитьПакет();

        ВыборкаКонтрагент = Результат[0].Выбрать();
        ВыборкаКонтрагент.Следующий();

        ВыборкаВалюта = Результат[1].Выбрать();
        ВыборкаВалюта.Следующий();

        ВыборкаНастройкиПользователя = Результат[2].Выбрать();
        ВыборкаНастройкиПользователя.Следующий();

        ДокументОбъект.Дата = Дата;
        ДокументОбъект.Организация = Организация;
        ДокументОбъект.Контрагент = ВыборкаКонтрагент.Контрагент;
        ДокументОбъект.Договор = ВыборкаКонтрагент.Договор;
        ДокументОбъект.ВидЦен = ВыборкаКонтрагент.ВидЦен;
        ДокументОбъект.ВалютаДокумента = ВыборкаВалюта.Валюта;
        ДокументОбъект.Курс = ВыборкаВалюта.Курс;
        ДокументОбъект.Кратность = ВыборкаВалюта.Кратность;
        ДокументОбъект.НалогообложениеНДС =
Перечисления.ТипыНалогообложенияНДС.НеОблагаетсяНДС;
        ДокументОбъект.ДокументОснование = Ссылка;
        ДокументОбъект.Ответственный = Ответственный;
        ДокументОбъект.Автор = ВыборкаНастройкиПользователя.Пользователь;
        ДокументОбъект.Подразделение = ВыборкаНастройкиПользователя.Подразделение;
```

```
		ТабРаботыИУслуги = ДокументОбъект.РаботыИУслуги;
		ТабРаботыИУслуги.Очистить();
		Для Каждого Стр Из ВсеУслуги Цикл
			Строка = ТабРаботыИУслуги.Добавить();
			Услуга = Стр.Услуга;
			Строка.Номенклатура = Услуга;
			Строка.ЕдиницаИзмерения = Услуга.ЕдиницаИзмерения;
			Строка.Количество = 1;
			Сумма = Стр.Сумма;
			Строка.Цена = Сумма;
			Строка.Сумма = Сумма;
			Строка.Всего = Сумма;
			СтавкаНДС = Справочники.СтавкиНДС.НайтиПоНаименованию("Без НДС", Истина);
			Строка.СтавкаНДС = СтавкаНДС;
		КонецЦикла;

		ДокументОбъект.Записать(РежимЗаписиДокумента.Проведение);
КонецПроцедуры

Процедура ОбработкаУдаленияПроведения(Отказ)
		АктВыполненыхРаботСсылка = ПолучитьПрявязанныйАктВыполненныхРабот();
		Если АктВыполненыхРаботСсылка <> Неопределено Тогда
			ДокОбъект = АктВыполненыхРаботСсылка.ПолучитьОбъект();
			ДокОбъект.Записать(РежимЗаписиДокумента.ОтменаПроведения);
		КонецЕсли;
КонецПроцедуры

Функция ПолучитьПрявязанныйАктВыполненныхРабот()
		Запрос = Новый Запрос;
		Запрос.Текст =
			"ВЫБРАТЬ ПЕРВЫЕ 1
			|	АктВыполненныхРабот.Ссылка
			|ИЗ
			|	Документ.АктВыполненныхРабот КАК АктВыполненныхРабот
			|ГДЕ
			|	АктВыполненныхРабот.ДокументОснование = &ДокументОснование";
		Запрос.УстановитьПараметр("ДокументОснование", Ссылка);
		Результат = Запрос.Выполнить();

		Если НЕ Результат.Пустой() Тогда
			Выборка = Результат.Выбрать();
			Выборка.Следующий();

			Возврат Выборка.Ссылка;
		КонецЕсли;

		Возврат Неопределено;
КонецФункции

Процедура ПриЗаписи(Отказ)
		АктВыполненныхРабот = ПолучитьПрявязанныйАктВыполненныхРабот();

		Если АктВыполненныхРабот <> Неопределено Тогда
			ДокументОбъект = АктВыполненныхРабот.ПолучитьОбъект();
			ДокументОбъект.УстановитьПометкуУдаления(ПометкаУдаления);
		КонецЕсли;
КонецПроцедуры

Процедура ЗаполнитьГрафикОплат()
		Если ГрафикОплат.Количество() = 0 Тогда
			Для Каждого Эл Из Абонементы Цикл
```

```
                        Абонемент = Эл.АбонементСпортсмена;
                        ВидАбонемента = Эл.ВидАбонемента;
                        Если ЗначениеЗаполнено(Абонемент) И ЗначениеЗаполнено(ВидАбонемента)
Тогда
                            Стр = ГрафикОплат.Добавить();
                            Стр.Абонемент = Абонемент;
                            Стр.ДатаОплаты = Дата;
                            Стр.ВидАбонемента = ВидАбонемента;
                            Стр.Сумма = Эл.Стоимость;
                        КонецЕсли;
                КонецЦикла;
                ЭтотОбъект.Записать(РежимЗаписиДокумента.Запись);
        КонецЕсли;
КонецПроцедуры

ДОКУМЕНТ ПродажаАбонемента ФормаДокумента
&НаКлиенте
Процедура УслугиПроцентСкидкиНаценкиПриИзменении(Элемент)

        РассчитатьСуммуВСтрокеТабличнойЧасти();

КонецПроцедуры
&НаКлиенте
Процедура РассчитатьСуммуВСтрокеТабличнойЧасти(СтрокаТабличнойЧасти = Неопределено,
СброситьФлагСкидкиРассчитаны = Истина)

        Если СтрокаТабличнойЧасти = Неопределено Тогда
                СтрокаТабличнойЧасти = Элементы.Услуги.ТекущиеДанные;
        КонецЕсли;

        СтрокаТабличнойЧасти.Сумма = СтрокаТабличнойЧасти.Количество *
СтрокаТабличнойЧасти.Цена;

        Если СтрокаТабличнойЧасти.ПроцентСкидкиНаценки = 100 Тогда

                СтрокаТабличнойЧасти.Сумма = 0;

        ИначеЕсли НЕ СтрокаТабличнойЧасти.ПроцентСкидкиНаценки = 0
                И НЕ СтрокаТабличнойЧасти.Количество = 0 Тогда

                СтрокаТабличнойЧасти.Сумма = СтрокаТабличнойЧасти.Сумма * (1 -
СтрокаТабличнойЧасти.ПроцентСкидкиНаценки / 100);

        КонецЕсли;

        //РассчитатьСуммуНДС(СтрокаТабличнойЧасти);
        //СтрокаТабличнойЧасти.Всего = СтрокаТабличнойЧасти.Сумма + ?(Объект.СуммаВключаетНДС,
0, СтрокаТабличнойЧасти.СуммаНДС);

        // АвтоматическиеСкидки.
        //Если СброситьФлагСкидкиРассчитаны Тогда
        //      ТребуетсяПересчетАвтоматическихСкидок =
СброситьФлагСкидкиРассчитаныКлиент("РассчитатьСуммуВСтрокеТабличнойЧасти");
        //КонецЕсли;

        //СтрокаТабличнойЧасти.ПроцентАвтоматическойСкидки = 0;
        //СтрокаТабличнойЧасти.СуммаАвтоматическойСкидки = 0;
        //СтрокаТабличнойЧасти.ОбщаяСуммаСкидкиБольшеСуммы = Ложь;
        // Конец АвтоматическиеСкидки

КонецПроцедуры // РассчитатьСуммуВСтрокеТабличнойЧасти()
```

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)
	УправлениеНебольшойФирмойСервер.ЗаполнитьШапкуДокумента(
		Объект,
		,
		Параметры.ЗначениеКопирования,
		Параметры.Основание,
		РазрешеноПроведение,
		Параметры.ЗначенияЗаполнения
	);

	Если Параметры.Свойство("Спортсмен") Тогда
		Объект.Спортсмен = Параметры.Спортсмен;
	КонецЕсли;

	Если Параметры.Свойство("Карта") и ЗначениеЗаполнено(Параметры.Карта) Тогда
		Объект.Карта = Параметры.Карта;
	КонецЕсли;

	Если Параметры.Свойство("ВидДисконтнойКарты") и
ЗначениеЗаполнено(Параметры.ВидДисконтнойКарты) Тогда
		Объект.ВидДисконтнойКарты = Параметры.ВидДисконтнойКарты;
	КонецЕсли;

	Если Параметры.Свойство("ДанныеДляЗаполненияТабЧастиАбонементы") Тогда
		СтруктураДанных = Параметры.ДанныеДляЗаполненияТабЧастиАбонементы;
		ВидАбонемента = СтруктураДанных.ВидАбонемента;
		Абонемент = СтруктураДанных.Абонемент;

		Если ЗначениеЗаполнено(ВидАбонемента) И ЗначениеЗаполнено(Абонемент) Тогда
			Стр = Объект.Абонементы.Добавить();
			Стр.ВидАбонемента = ВидАбонемента;
			Стр.АбонементСпортсмена = Абонемент;

			СтруктураДанныхОбщ =
ПолучитьДанныеДляЗаполненияТабЧастей(ВидАбонемента, Объект.Организация, Абонемент);

			ЗаполнитьЗначенияСвойств(Стр, СтруктураДанныхОбщ.ТабАбонементы);

			МассивДанныхУслуги = СтруктураДанныхОбщ.ТабУслуги;

			ТабУслуги = Услуги;
			ТабУслуги.Очистить();

			ВсеУслуги = Объект.ВсеУслуги;

			Отбор = Новый Структура("ВидАбонемента");
			Отбор.Вставить("ВидАбонемента", ВидАбонемента);
			Строки = ВсеУслуги.НайтиСтроки(Отбор);
			Для Каждого Эл Из Строки Цикл
				Индекс = ВсеУслуги.Индекс(Эл);
				ВсеУслуги.Удалить(Индекс);
			КонецЦикла;

			Для Каждого Эл Из МассивДанныхУслуги Цикл
				СтрокаТабУслуги = ТабУслуги.Добавить();
				СтрокаВсеУслуги = ВсеУслуги.Добавить();

				ЗаполнитьЗначенияСвойств(СтрокаВсеУслуги, Эл);
				ЗаполнитьЗначенияСвойств(СтрокаТабУслуги, СтрокаВсеУслуги);
			КонецЦикла;
```

```
                    МассивДанныхГрафикОплат = СтруктураДанныхОбщ.ТабГрафикОплат;

                ТабГрафикОплат = Объект.ГрафикОплат;

                    Отбор = Новый Структура("ВидАбонемента", ВидАбонемента);
                    Строки = ТабГрафикОплат.НайтиСтроки(Отбор);
                    Для Каждого Стр Из Строки Цикл
                        Индекс = ТабГрафикОплат.Индекс(Стр);
                        ТабГрафикОплат.Удалить(Индекс);
                    КонецЦикла;

                    Для Каждого Эл Из МассивДанныхГрафикОплат Цикл
                        Стр = ТабГрафикОплат.Добавить();
                        Стр.Абонемент = Абонемент;
                        Стр.Сумма = Эл.Сумма;
                        ДатаДействия = Объект.Дата;
                        ДатаОплаты = ДобавитьМесяц(?(ЗначениеЗаполнено(ДатаДействия),
ДатаДействия, ТекущаяДата()), Эл.Месяц);
                        Стр.ДатаОплаты = ДатаОплаты;
                        Стр.ВидАбонемента = ВидАбонемента;
                    КонецЦикла;
                КонецЕсли;
            КонецЕсли;

            УстановитьТекстПроАкт();
КонецПроцедуры

&НаКлиенте
Процедура АбонементыВидАбонементаПриИзменении(Элемент)
            Стр = ТекущийЭлемент.ТекущиеДанные;
            ВидАбонемента = Стр.ВидАбонемента;

            Если ЗначениеЗаполнено(ВидАбонемента) Тогда
                ТабАбонементы = Объект.Абонементы;
                Стр.АбонементСпортсмена = ПолучитьПустоеЗначениеАбонементаСпортсмена();

                Счетчик = 0;
                Для Каждого Эл Из ТабАбонементы Цикл
                    Если Эл.ВидАбонемента = ВидАбонемента Тогда
                        Счетчик = Счетчик + 1;
                        Если Счетчик > 1 Тогда
                            Сообщить("Такой вид абонемента уже есть в табличной части,
выберите другой!");
                            Стр.ВидАбонемента =
ПолучитьПустоеЗначениеВидаАбонемента();
                            Возврат;
                        КонецЕсли;
                    КонецЕсли;
                КонецЦикла;

                СтруктураДанныхОбщ = ПолучитьДанныеДляЗаполненияТабЧастей(ВидАбонемента,
Объект.Организация, Объект.Абонемент);

                ЗаполнитьЗначенияСвойств(Стр, СтруктураДанныхОбщ.ТабАбонементы);

                МассивДанныхУслуги = СтруктураДанныхОбщ.ТабУслуги;

                ТабУслуги = Услуги;
                ТабУслуги.Очистить();

                ВсеУслуги = Объект.ВсеУслуги;
```

```
                Отбор = Новый Структура("ВидАбонемента");
                Отбор.Вставить("ВидАбонемента", ВидАбонемента);
                Строки = ВсеУслуги.НайтиСтроки(Отбор);
                Для Каждого Эл Из Строки Цикл
                        Индекс = ВсеУслуги.Индекс(Эл);
                        ВсеУслуги.Удалить(Индекс);
                КонецЦикла;

                Для Каждого Эл Из МассивДанныхУслуги Цикл
                        СтрокаТабУслуги = ТабУслуги.Добавить();
                        СтрокаВсеУслуги = ВсеУслуги.Добавить();

                        ЗаполнитьЗначенияСвойств(СтрокаВсеУслуги, Эл);
                        ЗаполнитьЗначенияСвойств(СтрокаТабУслуги, СтрокаВсеУслуги);
                КонецЦикла;

                МассивДанныхГрафикОплат = СтруктураДанныхОбщ.ТабГрафикОплат;

            ТабГрафикОплат = Объект.ГрафикОплат;

                Отбор = Новый Структура("ВидАбонемента", ВидАбонемента);
                Строки = ТабГрафикОплат.НайтиСтроки(Отбор);
                Для Каждого Стр Из Строки Цикл
                        Индекс = ТабГрафикОплат.Индекс(Стр);
                        ТабГрафикОплат.Удалить(Индекс);
                КонецЦикла;

                Для Каждого Эл Из МассивДанныхГрафикОплат Цикл
                        Стр = ТабГрафикОплат.Добавить();
                        Стр.Сумма = Эл.Сумма;
                        ДатаДействия = Объект.Дата;
                        ДатаОплаты = ДобавитьМесяц(?(ЗначениеЗаполнено(ДатаДействия),
ДатаДействия, ТекущаяДата()), Эл.Месяц);
                        Стр.ДатаОплаты = ДатаОплаты;
                        Стр.ВидАбонемента = ВидАбонемента;
                КонецЦикла;
        КонецЕсли;
КонецПроцедуры

&НаСервереБезКонтекста
Функция ПолучитьДанныеДляЗаполненияТабЧастей(ВидАбонемента,
Организация,Абонемент=неопределено)

        Запрос = Новый Запрос;
        Запрос.Текст =
                "ВЫБРАТЬ
                |       СоставыГруппСрезПоследних.Группа,
                |       СоставыГруппСрезПоследних.Абонемент.ВидАбонемента,
                |       ТренерыГруппСрезПоследних.Тренер
                |ПОМЕСТИТЬ ТекущиеГруппы
                |ИЗ
                |       РегистрСведений.СоставыГрупп.СрезПоследних(, Абонемент = &Абонемент) КАК
СоставыГруппСрезПоследних
                |               ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.ТренерыГрупп.СрезПоследних
КАК ТренерыГруппСрезПоследних
                |               ПО СоставыГруппСрезПоследних.Группа =
ТренерыГруппСрезПоследних.Группа
                |ГДЕ
                |       СоставыГруппСрезПоследних.Активен = ИСТИНА
                |;
                |
                |////////////////////////////////////////////////////////////////////////////
```

```
|ВЫБРАТЬ
|	ВидыАбонементовУслуги.Ссылка.ДопустимыЗаморозки КАК
ДопустимыЗаморозки,
|	ВидыАбонементовУслуги.Ссылка.ДнейЗаморозки КАК ДнейЗаморозки,
|	ВидыАбонементовУслуги.Ссылка.КоличествоЗаморозок КАК
КоличествоЗаморозок,
|	ВидыАбонементовУслуги.Ссылка.ДлительностьОграничена КАК
ДлительностьОграничена,
|	ВидыАбонементовУслуги.Ссылка.Групповой КАК Групповой,
|	ВидыАбонементовУслуги.Ссылка.СуммаЗачисленияНаЛицевойСчет КАК
СуммаЗачисленияНаЛицевойСчет,
|	ВидыАбонементовУслуги.Ссылка.ЛицевойСчетДляБонуса КАК
ЛицевойСчетДляБонуса,
|	ВидыАбонементовУслуги.Услуга,
|	ВидыАбонементовУслуги.Количество,
|	ВидыАбонементовУслуги.Ссылка.ДлительностьДней КАК ДлительностьДней,
|	ВидыАбонементовУслуги.Ссылка.ДлительностьМесяцев КАК
ДлительностьМесяцев,
|	ВидыАбонементовУслуги.Ссылка.ДлительностьЛет КАК ДлительностьЛет,
|	ЦеныАбонементовСрезПоследних.Цена КАК Стоимость,
|	&ДатаНачалаДействия КАК ДатаНачалаДействия,
|	ВЫБОР
|		КОГДА ВидыАбонементовУслуги.Ссылка.ДлительностьДней = 0
|				И ВидыАбонементовУслуги.Ссылка.ДлительностьМесяцев
= 0
|				И ВидыАбонементовУслуги.Ссылка.ДлительностьЛет = 0
|			ТОГДА &ДатаНачалаДействия
|		ИНАЧЕ
ДОБАВИТЬКДАТЕ(ДОБАВИТЬКДАТЕ(ДОБАВИТЬКДАТЕ(ДОБАВИТЬКДАТЕ(&ДатаНачалаДействия,
ДЕНЬ, ВидыАбонементовУслуги.Ссылка.ДлительностьДней), МЕСЯЦ,
ВидыАбонементовУслуги.Ссылка.ДлительностьМесяцев), ГОД,
ВидыАбонементовУслуги.Ссылка.ДлительностьЛет), ДЕНЬ, -1)
|	КОНЕЦ КАК ДатаОкончанияДействия,
|	ВидыАбонементовУслуги.Цена,
|	ВидыАбонементовУслуги.Сумма,
|	ЕСТЬNULL(ТекущиеГруппы.Группа,
ЗНАЧЕНИЕ(Справочник.Группы.ПустаяСсылка)) КАК Группа,
|	ТекущиеГруппы.Тренер
|ИЗ
|	Справочник.ВидыАбонементов.Услуги КАК ВидыАбонементовУслуги
|		ЛЕВОЕ СОЕДИНЕНИЕ
РегистрСведений.ЦеныАбонементов.СрезПоследних(
|				,
|				ВидАбонемента = &Ссылка
|					И Организация = &Организация
|					И НЕ Цена = 0) КАК
ЦеныАбонементовСрезПоследних
|		ПО (ЦеныАбонементовСрезПоследних.ВидАбонемента =
ВидыАбонементовУслуги.Ссылка)
|		ЛЕВОЕ СОЕДИНЕНИЕ ТекущиеГруппы КАК ТекущиеГруппы
|		ПО ВидыАбонементовУслуги.Ссылка =
ТекущиеГруппы.АбонементВидАбонемента
|ГДЕ
|	ВидыАбонементовУслуги.Ссылка = &Ссылка
|ИТОГИ
|	МАКСИМУМ(ДнейЗаморозки),
|	МАКСИМУМ(КоличествоЗаморозок),
|	МАКСИМУМ(ДлительностьОграничена),
|	МАКСИМУМ(Групповой),
|	МАКСИМУМ(СуммаЗачисленияНаЛицевойСчет),
|	МАКСИМУМ(ЛицевойСчетДляБонуса),
|	МАКСИМУМ(ДлительностьДней),
```

```
                    |           МАКСИМУМ(ДлительностьМесяцев),
                    |           МАКСИМУМ(ДлительностьЛет),
                    |           МАКСИМУМ(Стоимость),
                    |           МАКСИМУМ(ДатаНачалаДействия),
                    |           МАКСИМУМ(ДатаОкончанияДействия)
                    |ПО
                    |           ДопустимыЗаморозки
                    |;
                    |
                    |////////////////////////////////////////////////////////////////////////////
                    |ВЫБРАТЬ
                    |           ВидыАбонементовШаблонРассрочки.Месяц КАК Месяц,
                    |           СУММА(ВидыАбонементовШаблонРассрочки.Сумма) КАК Сумма
                    |ИЗ
                    |           Справочник.ВидыАбонементов.ШаблонРассрочки КАК
ВидыАбонементовШаблонРассрочки
                    |ГДЕ
                    |           ВидыАбонементовШаблонРассрочки.Ссылка = &Ссылка
                    |
                    |СГРУППИРОВАТЬ ПО
                    |           ВидыАбонементовШаблонРассрочки.Месяц
                    |
                    |УПОРЯДОЧИТЬ ПО
                    |           Месяц";
          Запрос.УстановитьПараметр("Ссылка", ВидАбонемента);
          Запрос.УстановитьПараметр("Организация", Организация);
          Запрос.УстановитьПараметр("ДатаНачалаДействия", НачалоДня(ТекущаяДата()));
          Запрос.УстановитьПараметр("Абонемент",?(Абонемент=неопределено, Абонемент=
Справочники.АбонементыСпортсменов.ПустаяСсылка(),Абонемент)
);
          Результат = Запрос.ВыполнитьПакет();

          СтруктураДанныхОбщ = Новый Структура("ТабАбонементы, ТабУслуги, ТабГрафикОплат");
          СтруктураДанныхАбонементы = Новый Структура("ДопустимыЗаморозки, ДнейЗаморозки,
КоличествоЗаморозок, ДлительностьОграничена, Групповой, СуммаЗачисленияНаЛицевойСчет,
ЛицевойСчетДляБонуса, ДлительностьДней, ДлительностьЛет, ДлительностьМесяцев, Стоимость,
ДатаНачалаДействия, ДатаОкончанияДействия");

          МассивДанныхУслуги = Новый Массив;
          МассивДанныхГрафикОплат = Новый Массив;

          Если НЕ Результат[1].Пустой() Тогда
                    ВыборкаИтоги = Результат[1].Выбрать(ОбходРезультатаЗапроса.ПоГруппировкам);
                    Пока ВыборкаИтоги.Следующий() Цикл
                              ЗаполнитьЗначенияСвойств(СтруктураДанныхАбонементы, ВыборкаИтоги);

                              Выборка = ВыборкаИтоги.Выбрать();

                              Пока Выборка.Следующий() Цикл
                                        СтруктураДанныхУслуги = Новый Структура("Услуга, Количество,
ВидАбонемента, Цена, Сумма, Группа, Тренер");
                                        ЗаполнитьЗначенияСвойств(СтруктураДанныхУслуги, Выборка);
                                        СтруктураДанныхУслуги.Вставить("ВидАбонемента", ВидАбонемента);

                                        МассивДанныхУслуги.Добавить(СтруктураДанныхУслуги);
                              КонецЦикла;
                    КонецЦикла;
          КонецЕсли;

          Если НЕ Результат[2].Пустой() Тогда
                    Выборка = Результат[2].Выбрать();
```

```
                Пока Выборка.Следующий() Цикл
                    Если НЕ ЗначениеЗаполнено(Выборка.Месяц) Тогда
                        Продолжить;
                    КонецЕсли;
                    СтруктураДанныхГрафикОплат = Новый Структура("Месяц, Сумма");
                    ЗаполнитьЗначенияСвойств(СтруктураДанныхГрафикОплат, Выборка);
                    МассивДанныхГрафикОплат.Добавить(СтруктураДанныхГрафикОплат);
                КонецЦикла;
            КонецЕсли;

            СтруктураДанныхОбщ.Вставить("ТабАбонементы", СтруктураДанныхАбонементы);
            СтруктураДанныхОбщ.Вставить("ТабУслуги", МассивДанныхУслуги);
            СтруктураДанныхОбщ.Вставить("ТабГрафикОплат", МассивДанныхГрафикОплат);

            Возврат СтруктураДанныхОбщ;
КонецФункции

&НаКлиенте
Процедура УслугиКоличествоПриИзменении(Элемент)
            РассчитатьСуммуВСтрокеТабличнойЧасти();
            ПерезаписатьВсеУслуги();
КонецПроцедуры

&НаКлиенте
Процедура УслугиЦенаПриИзменении(Элемент)
            РассчитатьСуммуВСтрокеТабличнойЧасти();
            ПерезаписатьВсеУслуги();
КонецПроцедуры

&НаКлиенте
Процедура УслугиТренерПриИзменении(Элемент)
            ПерезаписатьВсеУслуги()
КонецПроцедуры

&НаКлиенте
Процедура УслугиГруппаПриИзменении(Элемент)
            ПерезаписатьВсеУслуги()
КонецПроцедуры

&НаКлиенте
Процедура ПерезаписатьВсеУслуги()
            Стр = Элементы.Услуги.ТекущиеДанные;
            Стр.Тренер = ПолучитьТренераПоГруппе(Стр.Группа);
            Отбор = Новый Структура("НомерСтроки, ВидАбонемента");
            Отбор.Вставить("НомерСтроки", Стр.НомерСтроки);
            Отбор.Вставить("ВидАбонемента", Стр.ВидАбонемента);

            ТабВсеУслуги = Объект.ВсеУслуги;
            Строки = ТабВсеУслуги.НайтиСтроки(Отбор);
            Если Строки.Количество() > 1 Тогда
                Сообщить("Произошла ошибка, обратитесь к разработчику!");
                Возврат;
            КонецЕсли;

            Строка = Строки[0];
            ЗаполнитьЗначенияСвойств(Строка, Стр);
КонецПроцедуры

&НаСервереБезКонтекста
Функция ПолучитьТренераПоГруппе(Группа)
            Возврат Группа.ОсновнойТренер;
КонецФункции
```

```
&НаКлиенте
Процедура АбонементыПриАктивизацииСтроки(Элемент)
	Если Объект.ВсеУслуги.Количество() <> 0 Тогда
		Стр = Элемент.ТекущиеДанные;
		ОбновлениеУслуг(Стр);
	КонецЕсли;
КонецПроцедуры

&НаКлиенте
Процедура ОбновлениеУслуг(Стр)
	Если Стр <> Неопределено Тогда
		ВидАбонемента = Стр.ВидАбонемента;
		Отбор = Новый Структура("ВидАбонемента", ВидАбонемента);
		АбонементыПриАктивизацииСтрокиНаСервере(Отбор);
	КонецЕсли;
КонецПроцедуры

&НаСервере
Процедура АбонементыПриАктивизацииСтрокиНаСервере(Отбор)
	Услуги.Загрузить(Объект.ВсеУслуги.Выгрузить(Отбор));
КонецПроцедуры

&НаКлиенте
Процедура АбонементыПередУдалением(Элемент, Отказ)
	Стр = Элементы.Абонементы.ТекущиеДанные;
	Если Стр <> Неопределено Тогда
		ВсеУслуги = Объект.ВсеУслуги;
		ТабГрафикОплат = Объект.ГрафикОплат;

		ВидАбонемента = Стр.ВидАбонемента;
		Отбор = Новый Структура("ВидАбонемента", ВидАбонемента);

		Если ВсеУслуги.Количество() <> 0 Тогда
			Строки = ВсеУслуги.НайтиСтроки(Отбор);
			Для Каждого Эл Из Строки Цикл
				Индекс = ВсеУслуги.Индекс(Эл);
				ВсеУслуги.Удалить(Индекс);
			КонецЦикла;
			Услуги.Очистить();
		КонецЕсли;

		Если ТабГрафикОплат.Количество() <> 0 Тогда
			Строки = ТабГрафикОплат.НайтиСтроки(Отбор);
			Для Каждого Эл Из Строки Цикл
				Индекс = ТабГрафикОплат.Индекс(Эл);
				ТабГрафикОплат.Удалить(Индекс);
			КонецЦикла;
		КонецЕсли;
	КонецЕсли;
КонецПроцедуры

&НаСервереБезКонтекста
Функция ПолучитьПустоеЗначениеВидаАбонемента()
	Возврат Справочники.ВидыАбонементов.ПустаяСсылка();
КонецФункции

&НаСервереБезКонтекста
Функция ПолучитьПустоеЗначениеАбонементаСпортсмена()
	Возврат Справочники.АбонементыСпортсменов.ПустаяСсылка();
КонецФункции
```

```
&НаКлиенте
Процедура АбонементыДопустимыЗаморозкиПриИзменении(Элемент)
	Стр = Элементы.Абонементы.ТекущиеДанные;

	Значение = Стр.ДопустимыЗаморозки;

	Элементы.АбонементыДнейЗаморозки.Доступность = Значение;
	Элементы.АбонементыКоличествоЗаморозок.Доступность = Значение;
КонецПроцедуры

&НаКлиенте
Процедура АбонементыДлительностьОграниченаПриИзменении(Элемент)
	Стр = Элементы.Абонементы.ТекущиеДанные;

	Значение = Стр.ДлительностьОграничена;

	Элементы.АбонементыДлительностьДней.Доступность = Значение;
	Элементы.АбонементыДлительностьМесяцев.Доступность = Значение;
	Элементы.АбонементыДлительностьЛет.Доступность = Значение;
КонецПроцедуры

&НаКлиенте
Процедура СпортсменПриИзменении(Элемент)
	Объект.Карта = УправлениеСпортклубом.ПолучитьКартуСпортсмена(Объект.Спортсмен);
	Объект.Плательщик = ПолучитьПлательщика();

КонецПроцедуры

&НаСервере
Функция ПолучитьПлательщика()
	ЭтаФорма.Элементы.НадписьСтатус.Заголовок = "";
	Если ЗначениеЗаполнено(Объект.Спортсмен.ФизЛицо.ДатаРождения) Тогда
		Если (ТекущаяДата() - Объект.Спортсмен.ФизЛицо.ДатаРождения < 60*60*24*365*18)
Тогда //спортсмен несовершеннолетен на текущий момент
			ЭтаФорма.Элементы.НадписьСтатус.Заголовок = "Спортсмен несовершеннолетен,
укажите плательщика, отличного от спортсмена.";
			Возврат Справочники.Контрагенты.ПустаяСсылка();
		ИначеЕсли НЕ ЗначениеЗаполнено(Объект.Спортсмен.Контрагент) Тогда
			ЭтаФорма.Элементы.НадписьСтатус.Заголовок = "У спортсмена не указан
контрагент. Контрагент необходим для оформления документов продажи.";
		Иначе
			Возврат Объект.Спортсмен.Контрагент;
		КонецЕсли;
	Иначе
			ЭтаФорма.Элементы.НадписьСтатус.Заголовок = "Не указана дата рождения
спортсмена.";
	КонецЕсли;
КонецФункции

&НаКлиенте
Процедура АктТекстНажатие(Элемент, СтандартнаяОбработка)
	СтандартнаяОбработка = Ложь;
	УправлениеСпортклубомКлиент.ОткрытьАкт(ЭтаФорма);
КонецПроцедуры

&НаКлиенте
Процедура ПослеЗаписи(ПараметрыЗаписи)
	УстановитьТекстПроАкт();

	Оповестить("ОбновлениеСпискаАбонементов");
КонецПроцедуры
```

```
&НаСервере
Процедура УстановитьТекстПроАкт()

	УправлениеСпортклубом.УстановитьТекстПроАкт(ЭтаФорма);

КонецПроцедуры

&НаКлиенте
Процедура ОбработкаОповещения(ИмяСобытия, Параметр, Источник)
	Если ИмяСобытия = "ОбновлениеТекстаПроАкт"
		И ТипЗнч(Параметр) = Тип("Структура")
		И Параметр.ДокументОснование = Объект.Ссылка Тогда

		АктТекст = Параметр.Представление;
	КонецЕсли;
КонецПроцедуры

&НаКлиенте
Процедура АбонементыДлительностьДнейПриИзменении(Элемент)
	ПересчитатьДатуОкончанияДействия();
КонецПроцедуры

&НаКлиенте
Процедура АбонементыДлительностьМесяцевПриИзменении(Элемент)
	ПересчитатьДатуОкончанияДействия();
КонецПроцедуры

&НаКлиенте
Процедура АбонементыДлительностьЛетПриИзменении(Элемент)
	ПересчитатьДатуОкончанияДействия();
КонецПроцедуры

&НаКлиенте
Процедура АбонементыДатаНачалаДействияПриИзменении(Элемент)
	ПересчитатьДатуОкончанияДействия();
КонецПроцедуры

&НаКлиенте
Процедура ПересчитатьДатуОкончанияДействия()
	Стр = Элементы.Абонементы.ТекущиеДанные;

	ДатаНачалаДействия = Стр.ДатаНачалаДействия;

	Если ЗначениеЗаполнено(ДатаНачалаДействия) Тогда
		СекундВСутках = 24 * 3600;
		Дней = Стр.ДлительностьДней;
		Месяцев = Стр.ДлительностьМесяцев;
		Лет = Стр.ДлительностьЛет;
		Если Дней = 0 И Месяцев = 0 И Лет = 0 Тогда
			ДатаОкончанияДействия = ДатаНачалаДействия;
		Иначе
		   ДатаОкончанияДействия = ДобавитьМесяц(ДатаНачалаДействия + СекундВСутках *
(Дней - 1), Месяцев + 12 * Лет);
		КонецЕсли;

		Стр.ДатаОкончанияДействия = ДатаОкончанияДействия;
	КонецЕсли;
КонецПроцедуры

ДОКУМЕНТ ВыдачаИнвентаря Модуль объекта

Процедура ОбработкаПроведения(Отказ, Режим)
```

```
	//{{__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
	// Данный фрагмент построен конструктором.
	// При повторном использовании конструктора, внесенные вручную изменения будут утеряны!!!

	// регистр АрендаИнвентаря Приход
	Движения.АрендаИнвентаря.Записывать = Истина;
	Для Каждого ТекСтрокаИнвентарь Из Инвентарь Цикл
		Движение = Движения.АрендаИнвентаря.Добавить();
		Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
		Движение.Период = Дата;
		Движение.Спорстмен = Спортсмен;
		Движение.Организация = Организация;
		Движение.Инвентарь = ТекСтрокаИнвентарь.Инвентарь;
		Движение.Количество = ТекСтрокаИнвентарь.Количество;
	КонецЦикла;

	//}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
КонецПроцедуры

ДОКУМЕНТ ВозвратИнвентаря Модуль объекта

Процедура ОбработкаЗаполнения(ДанныеЗаполнения, СтандартнаяОбработка)
	//{{__КОНСТРУКТОР_ВВОД_НА_ОСНОВАНИИ
	// Данный фрагмент построен конструктором.
	// При повторном использовании конструктора, внесенные вручную изменения будут утеряны!!!
	Если ТипЗнч(ДанныеЗаполнения) = Тип("ДокументСсылка.ВыдачаИнвентаря") Тогда
		// Заполнение шапки
		Организация = ДанныеЗаполнения.Организация;
		Спортсмен = ДанныеЗаполнения.Спортсмен;
		Основание = ДанныеЗаполнения.Ссылка;
		Для Каждого ТекСтрокаИнвентарь Из ДанныеЗаполнения.Инвентарь Цикл
			НоваяСтрока = Инвентарь.Добавить();
			НоваяСтрока.Инвентарь = ТекСтрокаИнвентарь.Инвентарь;
			НоваяСтрока.Количество = ТекСтрокаИнвентарь.Количество;
		КонецЦикла;
	ИначеЕсли ТипЗнч(ДанныеЗаполнения) = Тип("ДокументСсылка.Посещения") Тогда
		// Заполнение шапки
		Организация = ДанныеЗаполнения.Организация;
		Спортсмен = ДанныеЗаполнения.Спортсмен;
		Основание = ДанныеЗаполнения.Ссылка;
		Для Каждого ТекСтрокаВыданныйИнвентарь Из ДанныеЗаполнения.ВыданныйИнвентарь
Цикл
			НоваяСтрока = Инвентарь.Добавить();
			НоваяСтрока.Инвентарь = ТекСтрокаВыданныйИнвентарь.Инвентарь;
			НоваяСтрока.Количество = ТекСтрокаВыданныйИнвентарь.Количество;
		КонецЦикла;
	КонецЕсли;
	//}}__КОНСТРУКТОР_ВВОД_НА_ОСНОВАНИИ
КонецПроцедуры

Процедура ОбработкаПроведения(Отказ, Режим)
	//{{__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
	// Данный фрагмент построен конструктором.
	// При повторном использовании конструктора, внесенные вручную изменения будут утеряны!!!

	// регистр АрендаИнвентаря Приход
	Движения.АрендаИнвентаря.Записывать = Истина;
	Для Каждого ТекСтрокаИнвентарь Из Инвентарь Цикл
		Движение = Движения.АрендаИнвентаря.Добавить();
		Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
		Движение.Период = Дата;
		Движение.Спорстмен = Спортсмен;
```

```
                Движение.Организация = Организация;
                Движение.Инвентарь = ТекСтрокаИнвентарь.Инвентарь;
                Движение.Количество = ТекСтрокаИнвентарь.Количество;
        КонецЦикла;

        //}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
КонецПроцедуры

ОБРАБОТКА АРМАдминистратора Модуль объекта
Функция СформироватьКалендарь(ПараметрыПостроения, ВидОтображения = "ГрупповоеЗанятие")
        Линия = Новый Линия(ТипЛинииЯчейкиТабличногоДокумента.Сплошная,2);
        ТЗРасписание = Новый ТаблицаЗначений();
        //Таблица расписания должна проверять по какому помещению формируется и заполняться данными о рабочем времени помещения с интервалами работы помещения

        ТабличныйДокумент= Новый ТабличныйДокумент();
        КрупныйШрифт = Новый Шрифт(ШрифтыСтиля.ШрифтТекста, "Calibri Light", 25, Ложь,Ложь,Ложь,Ложь);


        МассивТипов = Новый Массив;
        МассивТипов.Добавить(Тип("Число"));
        КЧ = Новый КвалификаторыЧисла(10,3);
        ОписаниеЧисло = Новый ОписаниеТипов(МассивТипов,КЧ);
        ОписаниеДата = Новый ОписаниеТипов();
        ТЗРасписание = ПолучитьДанныеПоЗанятиям('20150407000000','20150427235959');

                //МакетДня = Обработки.УправлениеРасписаниемЗанятий.ПолучитьМакет("Макет30");
        МакетДня = ПолучитьОбщийМакет("День30");
        ОбластьЧасов = МакетДня.Область(1,1, 48, 166);
        МакетСобытия = Обработки.АРМАдминистратора.ПолучитьМакет("МакетСобытие");
        ОбластьСобытия = МакетСобытия.ПолучитьОбласть();
        ТабличныйДокумент.ВставитьОбласть(МакетДня.Область(1,1, 48, 166), ТабличныйДокумент.Область(2,1, 49, 166));
        ТабличныйДокумент.Область(1,3, 49, 166).ШиринаКолонки = 1;
        ТабличныйДокумент.Область(1,1, 49, 1).ШиринаКолонки = 4;
        ТабличныйДокумент.Область(1,2, 49, 2).ШиринаКолонки = 2;



        //определение первой колонки под час дня, верхняя строка - под заголовок
        Для Каждого ТЗСтр из ТЗРасписание Цикл
                ТабличныйДокумент.Область(Час(ТЗСтр.Нач)*2+1,3,Час(ТЗСтр.Кон)*2+1,100).ЦветФона = ЦветаСтиля.ФонАктивногоЗначенияОтбора;

        ТабличныйДокумент.Область(Час(ТЗСтр.Нач)*2+1,3,Час(ТЗСтр.Кон)*2+1,100).Объединить();

        ТабличныйДокумент.Область(Час(ТЗСтр.Нач)*2+1,3,Час(ТЗСтр.Кон)*2+1,100).Обвести(Линия,Линия,Линия,Линия);
                ТабличныйДокумент.Область(Час(ТЗСтр.Нач)*2+1,3,Час(ТЗСтр.Кон)*2+1,100).Текст = "" + ТЗСтр.событие + ", " + ТЗСтр.Тренер;

        ТабличныйДокумент.Область(Час(ТЗСтр.Нач)*2+1,3,Час(ТЗСтр.Кон)*2+1,100).ВертикальноеПоложение = ВертикальноеПоложение.Центр;

        ТабличныйДокумент.Область(Час(ТЗСтр.Нач)*2+1,3,Час(ТЗСтр.Кон)*2+1,100).ГоризонтальноеПоложение = ГоризонтальноеПоложение.Центр;
        КонецЦикла;
        Возврат ТабличныйДокумент
КонецФункции
```

```
Функция ПолучитьДанныеПоЗанятиям(ДатаНачала, ДатаОкончания, Помещение = Неопределено, Тренер =
Неопределено, Представление = "День")

	Если Представление = "День" Тогда
	ИначеЕсли Представление = "Неделя" Тогда
	ИначеЕсли Представление = "Месяц" Тогда
	КонецЕсли;

	Запрос = Новый Запрос;
	Запрос.Текст = "ВЫБРАТЬ
		|	РасписаниеЗанятий.ДатаНачала,
		|	РасписаниеЗанятий.ДатаОкончания,
		|	РасписаниеЗанятий.Группа,
		|	РасписаниеЗанятий.Спортсмен,
		|	РасписаниеЗанятий.Помещение,
		|	РасписаниеЗанятий.Услуга,
		|	РасписаниеЗанятий.Тренер,
		|	РасписаниеЗанятий.ВремяНачала,
		|	РасписаниеЗанятий.ВремяОкончания
		|ИЗ
		|	РегистрСведений.РасписаниеЗанятий КАК РасписаниеЗанятий";

	Запрос.Параметры.Вставить();
КонецФункции

ОБРАБОТКА АРМАдминистратора Форма

&НаКлиенте
Процедура ССылкаНаФотоНажатие(Элемент, СтандартнаяОбработка)
	// Вставить содержимое обработчика.
КонецПроцедуры

&НаКлиенте
Процедура Неделя(Команда)
	НеделяНаСервере();
КонецПроцедуры

&НаСервере
Процедура НеделяНаСервере()
	// Установка свойств
	ТабличныйДокумент.ФиксацияСверху = 1;
	ТабличныйДокумент.ФиксацияСлева  = 2;
	ТабличныйДокумент.ТолькоПросмотр = Истина;
	НастройкиФормирования = Новый Структура("ОтборПомещение, ОтборГруппаКонтрагентов,
СтраницаМенеджера, РазрешениеЭкрана, СтруктурнаяЕдиница, Дата, ДатаНачала, ДатаОкончания,
НаВсюФорму, ВыводитьКартинки");
	//НастройкиФормирования.ОтборПомещение		= РЗОтборПомещение;
	//НастройкиФормирования.ОтборГруппаКонтрагентов	= РЗОтборГруппаКонтрагентов;
	//НастройкиФормирования.СтруктурнаяЕдиница	= Объект.СтруктурнаяЕдиница;
	//НастройкиФормирования.СтраницаМенеджера	= ПолучитьСтраницуМенеджера();
	//НастройкиФормирования.РазрешениеЭкрана	= РазрешениеЭкрана;
	НастройкиФормирования.ДатаНачала		= Объект.ДатаНачала;
	НастройкиФормирования.ДатаОкончания		= Объект.ДатаОкончания;
	//НастройкиФормирования.НаВсюФорму		= Истина;
	//НастройкиФормирования.ВыводитьКартинки	= Не ЗапускВэбКлиент;
	// Вызов процедуры формирования календаря
	ГрафикСобытий = УправлениеСпортклубом.СформироватьКалендарь(НастройкиФормирования,
"Неделя");
	ТабличныйДокумент.Очистить();

	ТабличныйДокумент.ВставитьОбласть(ГрафикСобытий.Область());
	// Устанавливаем необходимые свойства
```

```
	Элементы.ТабличныйДокумент.ГоризонтальнаяПолосаПрокрутки = Истина;
	ТабличныйДокумент.ФиксацияСлева  = 2;
	//ИнтервалГрафикаЗанятий = Пользователи.ПолучитьЗначениеПоУмолчаниюПользователя(,
"ИнтервалГрафикаЗанятий");
	//ИнтервалГрафикаЗанятий = ?(ЗначениеЗаполнено(ИнтервалГрафикаЗанятий),
Число(Сред(ИнтервалГрафикаЗанятий, 1, 2)), 15);
	//РЗЗаполнен		= Истина;

КонецПроцедуры

&НаКлиенте
Процедура ПредставлениеДень(Команда)
	ДеньНаСервере()
КонецПроцедуры

&НаСервере
Процедура ДеньНаСервере()
	// Установка свойств
	ТабличныйДокумент.ФиксацияСверху = 1;
	ТабличныйДокумент.ФиксацияСлева  = 2;
	ТабличныйДокумент.ТолькоПросмотр = Истина;
	НастройкиФормирования = Новый Структура("ОтборПомещение, ОтборГруппаКонтрагентов,
СтраницаМенеджера, РазрешениеЭкрана, СтруктурнаяЕдиница, Дата, ДатаНачала, ДатаОкончания,
НаВсюФорму, ВыводитьКартинки");
	//НастройкиФормирования.ОтборПомещение		= РЗОтборПомещение;
	//НастройкиФормирования.ОтборГруппаКонтрагентов	= РЗОтборГруппаКонтрагентов;
	//НастройкиФормирования.СтруктурнаяЕдиница	= Объект.СтруктурнаяЕдиница;
	//НастройкиФормирования.СтраницаМенеджера	= ПолучитьСтраницуМенеджера();
	//НастройкиФормирования.РазрешениеЭкрана	= РазрешениеЭкрана;
	НастройкиФормирования.ДатаНачала		= Объект.ДатаНачала;
	НастройкиФормирования.ДатаОкончания		= Объект.ДатаОкончания;
	//НастройкиФормирования.НаВсюФорму		= Истина;
	//НастройкиФормирования.ВыводитьКартинки	= Не ЗапускВэбКлиент;
	// Вызов процедуры формирования календаря
	ГрафикСобытий = УправлениеСпортклубом.СформироватьКалендарь(НастройкиФормирования,
"День");
	ТабличныйДокумент.Очистить();

	ТабличныйДокумент.ВставитьОбласть(ГрафикСобытий.Область());
	// Устанавливаем необходимые свойства
	Элементы.ТабличныйДокумент.ГоризонтальнаяПолосаПрокрутки = Истина;
	ТабличныйДокумент.ФиксацияСлева  = 2;
	//ИнтервалГрафикаЗанятий = Пользователи.ПолучитьЗначениеПоУмолчаниюПользователя(,
"ИнтервалГрафикаЗанятий");
	//ИнтервалГрафикаЗанятий = ?(ЗначениеЗаполнено(ИнтервалГрафикаЗанятий),
Число(Сред(ИнтервалГрафикаЗанятий, 1, 2)), 15);
	//РЗЗаполнен		= Истина;

КонецПроцедуры

&НаКлиенте
Процедура ЗакрытьСмену(Команда)
	// Вставить содержимое обработчика.
КонецПроцедуры

&НаКлиенте
Процедура ПродатьТовар(Команда)
	ОткрытьФорму("Документ.РасходнаяНакладная.Форма.ФормаДокумента");
КонецПроцедуры

&НаКлиенте
Процедура ЗарегистрироватьПосещение(Команда)
```

```
        РегистрацияВходаВыхода(Объект.Спортсмен)
КонецПроцедуры

&НаКлиенте
Процедура СоздатьИндивидуальноеЗанятие(Команда)
        ОткрытьФорму("Документ.ПланированиеЗанятий.ФормаОбъекта");
КонецПроцедуры

&НаКлиенте
Процедура СоздатьМероприятие(Команда)
        ОткрытьФорму("Документ.Мероприятия.ФормаОбъекта");
КонецПроцедуры

&НаСервере
Процедура СоздатьСобытиеНаСервере(НовыйОбъект, СтруктураДат)
        НовыйОбъект.ДатаНачала                  = СтруктураДат.НачалоДень;
        НовыйОбъект.ДатаОкончания    = СтруктураДат.ОкончаниеДень;
        НовыйОбъект.ВремяНачала      = СтруктураДат.НачалоВремя;
        НовыйОбъект.ВремяОкончания  = СтруктураДат.ОкончаниеВремя;
КонецПроцедуры

&НаКлиенте
//Процедура СоздатьСобытие(Команда)
//
//        ВыделенныеОбласти     = ТабличныйДокумент.ВыделенныеОбласти[0];
//        Верх                            = ВыделенныеОбласти.Верх;
//        Низ                             = ВыделенныеОбласти.Низ;
//        Лево                            = ВыделенныеОбласти.Лево;
//        Право                           = ВыделенныеОбласти.Право;
//

//        Форма = ПолучитьФорму("Документ.ПланированиеЗанятий.Форма.ФормаДокумента");
//        ДанныеФормы = Форма.Объект;
//        СоздатьСобытиеНаСервере(ДанныеФормы, Верх, Низ, Лево = Неопределено, Право =
Неопределено);
//        КопироватьДанныеФормы(ДанныеФормы, Форма.Объект);
//        Форма.Открыть();

//КонецПроцедуры

Функция ПолучитьПериодИзТабличногоДокумента(ДатаНачалаИнтервала, Верх, Низ, Лево, Право)
        СтруктураДат = Новый Структура;
        СтруктураДат.Вставить("НачалоДень", 0);
        СтруктураДат.Вставить("ОкончаниеДень", 0);
        СтруктураДат.Вставить("НачалоВремя", 0);
        СтруктураДат.Вставить("ОкончаниеВремя", 0);

        Если Объект.Вид = 0 Тогда //День

        ИначеЕсли Объект.Вид = 1 Тогда //Неделя
                ДеньНеделиНач = ДатаНачалаИнтервала + Цел((лево-2)/19)*60*60*24;
                ДеньНеделиКон           = ДатаНачалаИнтервала + Цел((Право-2)/19)*60*60*24;
                ВремяНачала     = (Верх - 2)/2;
                ВремяОкончания          = (Низ - 1)/2;


                СтруктураДат.Вставить("НачалоДень", ДеньНеделиНач);
                СтруктураДат.Вставить("ОкончаниеДень", ДеньНеделиКон);
                СтруктураДат.Вставить("НачалоВремя", ВремяНачала);
                СтруктураДат.Вставить("ОкончаниеВремя", ВремяОкончания);
        ИначеЕсли Объект.Вид = 2 Тогда //Месяц
```

```
        КонецЕсли;
        //Нужно преобразовать полученные дни недели отталкиваясь от известной даты в даты начала и
окончания интервала и то же со временем.
        // ДатаНачалаЗанятий = Дата(СтруктураДат.НачалоДень = 0, ДатаНачалаИнтервала,
Дата(СтруктураДат.НачалоДень)
        //ДатаОкончанияЗанятий = ()
        //ВремяНачалаЗанятий =
        //ВремяОкончанияЗанятий =
        Возврат СтруктураДат;
КонецФункции

&НаКлиенте
Процедура СоздатьГрупповоеЗанятие(Команда)

        ВыделенныеОбласти      = ТабличныйДокумент.ВыделенныеОбласти[0];
        Верх                            = ВыделенныеОбласти.Верх;
        Низ                             = ВыделенныеОбласти.Низ;
        Лево                            = ВыделенныеОбласти.Лево;
        Право                           = ВыделенныеОбласти.Право;

        Форма = ПолучитьФорму("Документ.ПланированиеЗанятий.ФормаОбъекта");
        ДанныеФормы = Форма.Объект;
        СтруктураДат = ПолучитьПериодИзТабличногоДокумента(объект.ДатаНачала, Верх, Низ, Лево,
Право);
        СоздатьСобытиеНаСервере(ДанныеФормы, СтруктураДат);
        КопироватьДанныеФормы(ДанныеФормы, Форма.Объект);
        Форма.Открыть();

        //ОткрытьФорму("Документ.ПланированиеЗанятий.ФормаОбъекта");
        //СоздатьГрупповоеЗанятиеНаСервере();
КонецПроцедуры

&НаКлиенте
Процедура Следующий(Команда)
        Объект.ДатаОкончания = КонецНедели(Объект.ДатаОкончания + 60*60*24);
        Объект.ДатаНачала = НачалоНедели(Объект.ДатаОкончания);
        СформироватьНадписьПериода();
        ВидПриИзмененииНаСервере();
КонецПроцедуры

&НаКлиенте
Процедура Предыдущий(Команда)
        Объект.ДатаНачала = НачалоНедели(Объект.ДатаНачала - 1);
        Объект.ДатаОкончания = КонецНедели(Объект.ДатаНачала);
        СформироватьНадписьПериода();
        ВидПриИзмененииНаСервере();
КонецПроцедуры

&НаСервере
Процедура СформироватьНадписьПериода(НачалоПериод = Неопределено, КонецПериода = Неопределено)
        Элементы.ПериодНадпись.Заголовок = "" + Формат(Объект.ДатаНачала,"ДФ=dd.MM.yy") + " - " +
Формат(Объект.ДатаОкончания,"ДФ=dd.MM.yy");
КонецПроцедуры

&НаКлиенте
Процедура ПриОткрытии(Отказ)

        ИспользоватьПодключаемоеОборудование = Истина;
        // ПодключаемоеОборудование
        МенеджерОборудованияКлиентПереопределяемый.НачатьПодключениеОборудованиеПриОткрыти
иФормы(ЭтаФорма, "СканерШтрихкода,ДисплейПокупателя");
        // Конец ПодключаемоеОборудование
```

```
        ПриОткрытииНаСервере();

КонецПроцедуры

&НаСервере
Процедура ПриОткрытииНаСервере()
        Объект.ДатаНачала =  НачалоНедели(ТекущаяДата());
        Объект.ДатаОкончания =  КонецНедели(ТекущаяДата());
        СформироватьНадписьПериода();

КонецПроцедуры

&НаКлиенте
Процедура ПериодНадписьНажатие(Элемент)
        //вставить выбор интервала по календарю
КонецПроцедуры

&НаКлиенте
Процедура ВидПриИзменении(Элемент)
        ВидПриИзмененииНаСервере();
КонецПроцедуры

&НаСервере
Процедура ВидПриИзмененииНаСервере()
        Если    Объект.Вид = 0 Тогда
                ДеньНаСервере();
        ИначеЕсли Объект.Вид = 1 Тогда
                НеделяНаСервере();
        ИначеЕсли Объект.Вид = 2 Тогда
        КонецЕсли;
КонецПроцедуры

&НаКлиенте
Процедура ВыдатьИнвентарь(Команда)
        ОткрытьФорму("Документ.ВыдачаИнвентаря.ФормаОбъекта");
КонецПроцедуры

&НаКлиенте
Процедура ПолучитьИнвентарь(Команда)
        ОткрытьФорму("Документ.ВозвратИнвентаря.ФормаОбъекта");
КонецПроцедуры

&НаКлиенте
Процедура ЗарегистрироватьВыход(Команда)
        ОткрытьФорму("Документ.Посещения.ФормаВыбора");
КонецПроцедуры

&НаКлиенте
Процедура ПродатьАбонемент(Команда)
        ПараметрыПосещения = Новый Структура;
        ПараметрыПосещения.Вставить("Спортсмен", Объект.Спортсмен);
        Если ЗначениеЗаполнено(КартаСпортсмена) Тогда
                ПараметрыПосещения.Вставить("Карта", КартаСпортсмена);
        Иначе
                ПараметрыПосещения.Вставить("Карта",
УправлениеСпортклубом.ПолучитьКартуСпортсмена(Объект.Спортсмен));
        КонецЕсли;
        ОткрытьФорму("Документ.ПродажаАбонемента.Форма.ФормаДокумента", ПараметрыПосещения,
ЭтаФорма);
КонецПроцедуры
```

```
&НаКлиенте
Процедура ВнестиОплату(Команда)
	Спортсмен = Объект.Спортсмен;
	Стр = Элементы.Абонементы.ТекущиеДанные;

	Если НЕ ЗначениеЗаполнено(Спортсмен) Тогда
		Сообщить("Не выбран спортсмен.");
		Возврат;
	КонецЕсли;

	Если Стр = Неопределено Тогда
		Сообщить("Не выбран абонемент спортсмена.");
		Возврат;
	КонецЕсли;

	Абонемент = Стр.АбонементСпортсмена;
	ОстатокОплаты = ПолучитьОстатокОплатыПоАбонементу(Абонемент, Спортсмен);

	ПараметрыОплаты = Новый Структура;
	ПараметрыОплаты.Вставить("Спортсмен", Спортсмен);
	ПараметрыОплаты.Вставить("Плательщик", ПолучитьКонтрагента(Спортсмен));
	ПараметрыОплаты.Вставить("Абонемент", Абонемент);
	ПараметрыОплаты.Вставить("Сумма", ОстатокОплаты);

	ОткрытьФорму("Документ.РегистрацияОплаты.Форма.ФормаДокумента", ПараметрыОплаты);
КонецПроцедуры

&НаСервереБезКонтекста
Функция ПолучитьОстатокОплатыПоАбонементу(Абонемент, Спортсмен)
	Запрос = Новый Запрос;
	Запрос.Текст =
		"ВЫБРАТЬ
		|	ВзаиморасчетыПоАбонементамОстатки.СуммаОстаток
		|ИЗ
		|	РегистрНакопления.ВзаиморасчетыПоАбонементам.Остатки(
		|			&Дата,
		|			Абонемент = &Абонемент
		|				И Спортсмен = &Спортсмен) КАК
ВзаиморасчетыПоАбонементамОстатки";
	Запрос.УстановитьПараметр("Абонемент", Абонемент);
	Запрос.УстановитьПараметр("Дата", ТекущаяДата());
	Запрос.УстановитьПараметр("Спортсмен", Спортсмен);
	Результат = Запрос.Выполнить();

	СуммаОстаток = 0;
	Если НЕ Результат.Пустой() Тогда
		Выборка = Результат.Выбрать();
		Выборка.Следующий();

		СуммаОстаток = Выборка.СуммаОстаток;
	КонецЕсли;

	Возврат СуммаОстаток;
КонецФункции

&НаСервереБезКонтекста
Функция ПолучитьКонтрагента(Спортсмен)

	Возврат Спортсмен.Контрагент;

КонецФункции
```

```
&НаКлиенте
Процедура СпортсменПриИзменении(Элемент)

	ССылкаНаФото = ПолучитьНавигационнуюССылку(Объект.Спортсмен, "Фото");
	Объект.Абонемент =
ПредопределенноеЗначение("Справочник.АбонементыСпортсменов.ПустаяССылка");
	СпортсменПриИзмененииНаСервере();
	Элементы.НадписьДолг.Заголовок = ТекстДолга();
КонецПроцедуры

&НаСервере
Процедура СпортсменПриИзмененииНаСервере()
	ФизЛицо = Объект.Спортсмен.ФизЛицо;
	ЗаполнитьЗначенияСвойств(ЭтаФорма, ФизЛицо);
	ЗаполнитьЗначенияСвойств(ЭтаФорма, Объект.Спортсмен);

	//Запрос = Новый Запрос;
	//Запрос.Текст = "ВЫБРАТЬ
	//|	АбонементыСпортсменов.Ссылка
	//|ИЗ
	//|	Справочник.АбонементыСпортсменов КАК АбонементыСпортсменов
	//|ГДЕ
	//|	АбонементыСпортсменов.Владелец = &Владелец";
	//
	//Запрос.УстановитьПараметр("Владелец", Объект.Спортсмен);
	//Выборка = Запрос.Выполнить().Выбрать();
	//Если Выборка.Следующий() Тогда
	//	Объект.Абонемент = Выборка.Ссылка;
	//КонецЕсли;



	ОбновитьКонтактнуюИнформацию(ФизЛицо);
	Если ЗначениеЗаполнено(Объект.Спортсмен) Тогда
		Сост = ПолучитьСостояниеСпортсмена(Объект.Спортсмен);
		Если Сост = перечисления.СостоянияСпортсменов.ВКлубе Тогда
			ЭтаФорма.Элементы.СостояниеСпортсмена.ЦветТекста = WebЦвета.Зеленый;
		ИначеЕсли Сост = перечисления.СостоянияСпортсменов.НеВКлубе Тогда
			ЭтаФорма.Элементы.СостояниеСпортсмена.ЦветТекста = WebЦвета.Серый;
		ИначеЕсли Сост = перечисления.СостоянияСпортсменов.НаБольничном Тогда
			ЭтаФорма.Элементы.СостояниеСпортсмена.ЦветТекста = WebЦвета.Красный;
		Иначе
			ЭтаФорма.Элементы.СостояниеСпортсмена.ЦветТекста = WebЦвета.Серый;
			Сост = "Не в клубе";
		КонецЕсли;
		ЗаполнитьАбонементыСпортсмена(Объект.Спортсмен);
		Если ЗначениеЗаполнено(Объект.Спортсмен.ФизЛицо.ДатаРождения) Тогда
			ЭтаФорма.Элементы.НадписьВозраст.Заголовок = окр((ТекущаяДата() -
объект.Спортсмен.ФизЛицо.ДатаРождения)/365/24/60/60);
		Иначе
			ЭтаФорма.Элементы.НадписьВозраст.Заголовок = "";
		КонецЕсли;
	Иначе
		Сост = "Не в клубе";
	КонецЕсли;
	ЭтаФорма.Элементы.СостояниеСпортсмена.Заголовок = "" + Сост;
	КартаСпортсмена = ПолучитьКартуСпортсмена(Объект.Спортсмен);
	Штрихкод = КартаСпортсмена.ДисконтнаяКарта.КодКартыШтрихкод;
	ПосещенияСпортсмена.Загрузить(УправлениеСпортклубом.ПосещенияСпортсмена(Объект.Спортс
мен));
КонецПроцедуры
```

```
&НаСервере
Функция ПолучитьКартуСпортсмена(Спортсмен)
	Запрос = Новый Запрос();
	Запрос.Текст = "ВЫБРАТЬ
		|	КлубныеКарты.Ссылка
		|ИЗ
		|	Справочник.КлубныеКарты КАК КлубныеКарты
		|ГДЕ
		|	КлубныеКарты.Владелец = &Спортсмен";
	Запрос.Параметры.Вставить("Спортсмен", Спортсмен);
	Выборка = Запрос.Выполнить().Выбрать();
	Если Выборка.Количество() > 0 Тогда
		Выборка.Следующий();
		Возврат Выборка.Ссылка;
	Иначе
		Возврат Неопределено;
	КонецЕсли;
КонецФункции

&НаСервере
//Функция возвращает состояние спортсмена на текущее время сервера.
//Необходимо доработать функцию таким образом, чтобы она учитывала организацию и структурную единицу, для которой выясняется состояние спортсмена.
Функция ПолучитьСостояниеСпортсмена(Спортсмен)
	Запрос = Новый Запрос();
	Запрос.Текст = "ВЫБРАТЬ
		|	ПосещенияСпортсменовСрезПоследних.СостояниеСпортсмена
		|ИЗ
		|	РегистрСведений.ПосещенияСпортсменов.СрезПоследних(&ДатаСостояния,
Спортсмен = &Спортсмен И ДатаВремяВыхода =  ДАТАВРЕМЯ(1,1,1)) КАК
ПосещенияСпортсменовСрезПоследних";
	Запрос.Параметры.Вставить("ДатаСостояния", ТекущаяДата());
	Запрос.Параметры.Вставить("Спортсмен", Спортсмен);
	Выборка = Запрос.Выполнить().Выбрать();
	Если Выборка.Количество() > 0 Тогда
		Выборка.Следующий();
		Возврат Выборка.СостояниеСпортсмена;
	Иначе
		Возврат "Не в клубе";
	КонецЕсли;
КонецФункции

&НаСервере
Процедура ЗаполнитьАбонементыСпортсмена(Спортсмен) Экспорт
	Объект.Абонементы.Очистить();
	Запрос = Новый Запрос();
	Если НЕ ПоказыватьВсеАбонементы Тогда
		Запрос.Текст = "ВЫБРАТЬ
			|	СостоянияАбонементовСрезПоследних.Абонемент КАК
АбонементСпортсмена,
			|	СостоянияАбонементовСрезПоследних.СостояниеАбонемента КАК
Состояние,
			|	СостоянияАбонементовСрезПоследних.ДатаОграничения,
			|	СостоянияАбонементовСрезПоследних.СрокОкончания КАК
СрокДействияАбонемента,
			|	ЗанятияПоАбонементамОстатки.КоличествоОстаток КАК ЗанятийОсталось
			|ИЗ
			|	РегистрСведений.СостоянияАбонементов.СрезПоследних(
			|			&ТекДата,
			|			Абонемент.Владелец = &Спортсмен
			|				И СостояниеАбонемента = &Активен) КАК
СостоянияАбонементовСрезПоследних
```

```
                        |                ЛЕВОЕ СОЕДИНЕНИЕ
РегистрНакопления.ЗанятияПоАбонементам.Остатки(, ) КАК ЗанятияПоАбонементамОстатки
                        |                        ЛЕВОЕ СОЕДИНЕНИЕ Справочник.ФизическиеЛица
КАК ФизическиеЛица
                        |                        ПО ЗанятияПоАбонементамОстатки.Спортсмен.ФизЛицо
= ФизическиеЛица.Ссылка
                        |                ПО СостоянияАбонементовСрезПоследних.Абонемент =
ЗанятияПоАбонементамОстатки.Абонемент
                        |ГДЕ
                        |        СостоянияАбонементовСрезПоследних.СрокОкончания >= &ТекДата";
        Иначе
                Запрос.Текст = "ВЫБРАТЬ
                        |        СостоянияАбонементовСрезПоследних.Абонемент КАК
АбонементСпортсмена,
                        |        ВЫБОР
                        |                КОГДА СостоянияАбонементовСрезПоследних.СрокОкончания
>= &ТекДата
                        |                        ТОГДА
СостоянияАбонементовСрезПоследних.СостояниеАбонемента
                        |                ИНАЧЕ ""Истек""
                        |        КОНЕЦ КАК Состояние,
                        |        СостоянияАбонементовСрезПоследних.ДатаОграничения,
                        |        СостоянияАбонементовСрезПоследних.СрокОкончания КАК
СрокДействияАбонемента,
                        |        ЗанятияПоАбонементамОстатки.КоличествоОстаток КАК ЗанятийОсталось
                        |ИЗ
                        |        РегистрСведений.СостоянияАбонементов.СрезПоследних(,
Абонемент.Владелец = &Спортсмен) КАК СостоянияАбонементовСрезПоследних
                        |                ЛЕВОЕ СОЕДИНЕНИЕ
РегистрНакопления.ЗанятияПоАбонементам.Остатки(, ) КАК ЗанятияПоАбонементамОстатки
                        |                        ЛЕВОЕ СОЕДИНЕНИЕ Справочник.ФизическиеЛица
КАК ФизическиеЛица
                        |                        ПО ЗанятияПоАбонементамОстатки.Спортсмен.ФизЛицо
= ФизическиеЛица.Ссылка
                        |                ПО СостоянияАбонементовСрезПоследних.Абонемент =
ЗанятияПоАбонементамОстатки.Абонемент";
        КонецЕсли;
        Запрос.Параметры.Вставить("ТекДата", НачалоДня(ТекущаяДата()));
        Запрос.Параметры.Вставить("Спортсмен", Спортсмен);
        Запрос.Параметры.Вставить("Активен", Перечисления.СостоянияАбонементов.Активен);
        ТЗ = Запрос.Выполнить().Выгрузить();
        Если ТЗ.Количество() > 0 Тогда
                Объект.Абонементы.Загрузить(ТЗ);
                Для Каждого ТабСтр из Объект.Абонементы Цикл
                        Если НЕ ЗначениеЗаполнено(ТабСтр.Состояние) Тогда
                                ТабСтр.Состояние = Перечисления.СостоянияАбонементов.Истек;
                        КонецЕсли;
                КонецЦикла;
        КонецЕсли;
        ЭтаФорма.Элементы.Абонементы.Обновить();
КонецПроцедуры

&НаСервере
Процедура ОбновитьКонтактнуюИнформацию(ФизЛицо)

        НомерСтрокиТелефон = 0;
        НомерСтрокиEmail = 0;
        Телефон = "";
        EMail = "";

        Запрос = Новый Запрос;
        Запрос.Текст = "ВЫБРАТЬ
```

```
                    |        ФизическиеЛицаКонтактнаяИнформация.Ссылка,
                    |        ФизическиеЛицаКонтактнаяИнформация.НомерСтроки,
                    |        ФизическиеЛицаКонтактнаяИнформация.Тип КАК Тип,
                    |        ФизическиеЛицаКонтактнаяИнформация.Вид,
                    |        ФизическиеЛицаКонтактнаяИнформация.Представление,
                    |        ФизическиеЛицаКонтактнаяИнформация.ЗначенияПолей,
                    |        ФизическиеЛицаКонтактнаяИнформация.Страна,
                    |        ФизическиеЛицаКонтактнаяИнформация.Регион,
                    |        ФизическиеЛицаКонтактнаяИнформация.Город,
                    |        ФизическиеЛицаКонтактнаяИнформация.АдресЭП,
                    |        ФизическиеЛицаКонтактнаяИнформация.ДоменноеИмяСервера,
                    |        ФизическиеЛицаКонтактнаяИнформация.НомерТелефона,
                    |        ФизическиеЛицаКонтактнаяИнформация.НомерТелефонаБезКодов
                    |ИЗ
                    |        Справочник.ФизическиеЛица.КонтактнаяИнформация КАК
ФизическиеЛицаКонтактнаяИнформация
                    |ГДЕ
                    |        ФизическиеЛицаКонтактнаяИнформация.Ссылка = &Ссылка
                    |        И ФизическиеЛицаКонтактнаяИнформация.Тип В
(ЗНАЧЕНИЕ(Перечисление.ТипыКонтактнойИнформации.АдресЭлектронноПочты),
ЗНАЧЕНИЕ(Перечисление.ТипыКонтактнойИнформации.Телефон))
                    |
                    |УПОРЯДОЧИТЬ ПО
                    |        Тип";


        Запрос.УстановитьПараметр("Ссылка", ФизЛицо);

        Выборка = Запрос.Выполнить().Выбрать();
        Пока Выборка.Следующий() Цикл
                Если Выборка.Тип = Перечисления.ТипыКонтактнойИнформации.Телефон Тогда
                        Если НЕ НомерСтрокиТелефон = 0 Тогда
                                Продолжить;
                        КонецЕсли;
                        Телефон = Выборка.НомерТелефона;
                        НомерСтрокиТелефон  = Выборка.НомерСтроки;
                ИначеЕсли Выборка.Тип =
Перечисления.ТипыКонтактнойИнформации.АдресЭлектронноПочты Тогда
                        Если НЕ НомерСтрокиEmail = 0 Тогда
                                Продолжить;
                        КонецЕсли;
                        EMail = Выборка.АдресЭП;
                        НомерСтрокиEmail  = Выборка.НомерСтроки;
                КонецЕсли;
        КонецЦикла;

        Комментарий = Объект.Спортсмен.Комментарий;

КонецПроцедуры

&Насервере
Процедура ПриИзмененииДанныхСпортсмена(Реквизит)

        Спортсмен = Объект.Спортсмен.ПолучитьОбъект();
        Спортсмен[Реквизит] = Этаформа[Реквизит];
        Спортсмен.Записать();

КонецПроцедуры

&Насервере
Процедура ПриИзмененииДанныхФизлица(Реквизит)
```

```
	Физлицо = Объект.Спортсмен.ФизЛицо.ПолучитьОбъект();
	Физлицо[Реквизит] = Этаформа[Реквизит];
	Физлицо.Записать();

КонецПроцедуры

&НаКлиенте
Процедура ВернутьОплату(Команда)
	ОткрытьФорму("Документ.ВозвратОплатыСпортсмену.ФормаОбъекта");
КонецПроцедуры

&НаКлиенте
Процедура ЗарегистрироватьСпортсмена(Команда)
	ОткрытьФорму("Справочник.Спортсмены.Форма.ФормаЭлемента");
КонецПроцедуры

&НаКлиенте
Процедура ОбработкаОповещения(ИмяСобытия, Параметр, Источник)

	// ПодключаемоеОборудование
	Если Источник = "ПодключаемоеОборудование"
		И ВводДоступен()
		Тогда
		Если ИмяСобытия = "ScanData" Тогда
			//Преобразуем предварительно к ожидаемому формату
			Данные = Новый Массив();
			Если Параметр[1] = Неопределено Тогда
				Данные.Добавить(Новый Структура("Штрихкод, Количество",
Параметр[0], 1)); // Достаем штрихкод из основных данных
			Иначе
				Данные.Добавить(Новый Структура("Штрихкод, Количество",
Параметр[1][1], 1)); // Достаем штрихкод из дополнительных данных
			КонецЕсли;

			Если Данные.Количество() <= 0 Тогда
				Сообщить("Ошибка чтения штрихкода");
				Возврат;
			КонецЕсли;

			Штрихкод = Данные[0].Штрихкод;
			ОбработатьШтрихкод(Штрихкод);
		КонецЕсли;
	КонецЕсли;
	// Конец ПодключаемоеОборудование

	Если ИмяСобытия = "ОбновлениеСпискаАбонементов" Тогда
		Если ЗначениеЗаполнено(Объект.Спортсмен) Тогда
			ЗаполнитьАбонементыСпортсмена(Объект.Спортсмен);
			Элементы.НадписьДолг.Заголовок = ТекстДолга();
		КонецЕсли;
	ИначеЕсли ИмяСобытия = "ПолученаОплата" Тогда
		Элементы.НадписьДолг.Заголовок = ТекстДолга();
	КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура ОбработатьШтрихкод(Штрихкод)

	ЕстьКарта = Ложь;
	Карта = ПредопределенноеЗначение("Справочник.КлубныеКарты.ПустаяСсылка");
```

```
        ПоискКарты(Штрихкод, ЕстьКарта, Карта);

        КартаСпортсмена = Карта;

        Если НЕ ЕстьКарта Тогда
                Сообщить("Карта не найдена. Идет поиск товара");
                ЕстьТовар = Ложь;
                Товар = ПредопределенноеЗначение("Справочник.Номенклатура.ПустаяСсылка");
                ПоискТовара(Штрихкод, ЕстьТовар, Товар);

                Если ЕстьТовар Тогда
                        ТекстВопроса = "Найден товар. Ввести чек?";

                        Кнопки = Новый СписокЗначений;
                        Кнопки.Добавить(КодВозвратаДиалога.Да);
                        Кнопки.Добавить(КодВозвратаДиалога.Нет);

                        Ответ = Вопрос(ТекстВопроса, Кнопки, , КодВозвратаДиалога.Да);
                        Если Ответ = КодВозвратаДиалога.Да Тогда
                                ОбработкаОтветаПоТовару(Товар);
                        КонецЕсли;
                Иначе
                        Сообщить("В системе отсутствуют данные по считанному штрихкоду");
                КонецЕсли;

        Иначе
                ПСпортсмен = ПолучитьСпортсменаПоКарте(Карта);

                ДанныеАбонемента = НайтиАбонементы(ПСпортсмен);
                Действие = ДанныеАбонемента.Действие;
                Абонемент = ДанныеАбонемента.Абонемент;

                Если Действие = "Вход" Тогда

                        СостояниеСпорт = ПолучитьСостояниеСпортсмена(ПСпортсмен);
                        ТекстВопроса = ПолучитьТекстВопросаПосещения(СостояниеСпорт);
                        Кнопки = Новый СписокЗначений;
                        Кнопки.Добавить(КодВозвратаДиалога.Да);
                        Кнопки.Добавить(КодВозвратаДиалога.Нет);

                        Ответ = Вопрос(ТекстВопроса, Кнопки, , КодВозвратаДиалога.Да);
                        Если Ответ = КодВозвратаДиалога.Да Тогда
                                РегистрацияВходаВыхода(ПСпортсмен);
                        КонецЕсли;
                ИначеЕсли Действие = "Продление" Тогда
                        ЗадатьВопросИзменениеАбонемента("Продлить абонемент?", Действие,
Абонемент);
                ИначеЕсли Действие = "Возобновление" Тогда
                        ЗадатьВопросИзменениеАбонемента("Возобновить действие абонемента?",
Действие, Абонемент);
                ИначеЕсли Действие = "Разблокировка" Тогда
              ЗадатьВопросИзменениеАбонемента("Разблокировать абонемент?", Действие, Абонемент);
                КонецЕсли;

                Объект.Спортсмен = ПСпортсмен;
                СпортсменПриИзменении(Элементы.Спортсмен);

        КонецЕсли;

КонецПроцедуры

&НаКлиенте
```

```
Процедура РегистрацияВходаВыхода(Знач Спортсмен)

	Перем Кнопки, Ответ, СостояниеСпорт, ТекстВопроса;

	ДокументПосещения = УправлениеСпортклубом.ПолучитьДокументВходаВКлуб(Спортсмен);
	Если ДокументПосещения = Неопределено Тогда
		ПараметрыПосещения = Новый Структура;
		ПараметрыПосещения.Вставить("Спортсмен", Спортсмен);
		ОткрытьФорму("Документ.Посещения.ФормаОбъекта", ПараметрыПосещения);
	Иначе
		ТекДата = ТекущаяДата();

		Отбор = Новый Структура("Ключ");
		Отбор.Вставить("Ключ", ДокументПосещения);

		ФормаДокументаПосещения = ПолучитьФорму("Документ.Посещения.ФормаОбъекта",
Отбор);
		ФормаДокументаПосещения.Объект.ДатаВремяВыхода = ТекДата;
		ФормаДокументаПосещения.Открыть();
	КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура ЗадатьВопросИзменениеАбонемента(ТекстВопроса, Действие, Абонемент)

	ДополнительныеПараметры = Новый Структура;
	ДополнительныеПараметры.Вставить("ВидОперации",  Действие);
	Кнопки = Новый СписокЗначений;
	Кнопки.Добавить(КодВозвратаДиалога.Да);
	Кнопки.Добавить(КодВозвратаДиалога.Нет);

	Ответ = Вопрос(ТекстВопроса, Кнопки, , КодВозвратаДиалога.Да);
	Если Ответ = КодВозвратаДиалога.Да Тогда
		СоздатьИзменениеАбонемента(Действие, Абонемент);
	КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура СоздатьИзменениеАбонемента(ВидОперации, Абонемент) Экспорт

	ПараметрыОткрытия = Новый Структура;
	ПараметрыОткрытия.Вставить("ВидОперации",
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов." + ВидОперации));
	ПараметрыОткрытия.Вставить("Абонемент", Абонемент);

	ОткрытьФорму("Документ.ИзменениеАбонемента.ФормаОбъекта", ПараметрыОткрытия);

КонецПроцедуры

&НаСервере
Функция ПолучитьСпортсменаПоКарте(Карта)
	Если ЗначениеЗаполнено(Карта) Тогда
		Возврат Карта.Владелец
	КонецЕсли;
КонецФункции
&НаСервере
Функция ПолучитьТекстВопросаПосещения(Состояние)
		Если Состояние = Перечисления.СостоянияСпортсменов.ВКлубе Тогда
			Возврат "Выход?";
		Иначе
```

```
                        Возврат "Вход?";
                КонецЕсли;
КонецФункции

&НаСервере
Процедура ПоискКарты(Штрихкод, КартаНайдена, Карта)

        ДисконтнаяКарта =
ДисконтныеКартыВызовСервера.НайтиДисконтныеКартыПоШтрихкоду(Штрихкод);
        Если ДисконтнаяКарта.ЗарегистрированныеДисконтныеКарты.Количество() <= 0 Тогда
                КартаНайдена = Ложь;
                Возврат;
        КонецЕсли;

        Карта = Справочники.КлубныеКарты.НайтиПоРеквизиту("ДисконтнаяКарта",
ДисконтнаяКарта.ЗарегистрированныеДисконтныеКарты[0].Ссылка);

        Если ЗначениеЗаполнено(Карта) Тогда
                КартаНайдена = Истина;
        Иначе
                КартаНайдена = Ложь;
        КонецЕсли;


КонецПроцедуры

&НаСервере
Функция НайтиАбонементы(Спортсмен)

        Абонементы = УправлениеСпортКлубом.ПолучитьАбонементыСпортсмена(ТекущаяДата(),
Спортсмен);

        СтруктураВозврата = Новый Структура("Действие, Абонемент");
        СтруктураВозврата.Действие = "";
        СтруктураВозврата.Абонемент = Справочники.АбонементыСпортсменов.ПустаяСсылка();

        Отбор = Новый Структура;
        Отбор.Вставить("СостояниеАбонемента", Перечисления.СостоянияАбонементов.Активен);
        Найденные = Абонементы.НайтиСтроки(Отбор);
        Если Найденные.Количество() > 0 Тогда
                СтруктураВозврата.Действие = "Вход";
                СтруктураВозврата.Абонемент =Найденные[0].Абонемент;
                Возврат СтруктураВозврата;
        Конецесли;

        Отбор = Новый Структура;
        Отбор.Вставить("СостояниеАбонемента", Перечисления.СостоянияАбонементов.Истек);
        Найденные = Абонементы.НайтиСтроки(Отбор);
        Если Найденные.Количество() > 0 Тогда
                СтруктураВозврата.Действие = "Продление";
                СтруктураВозврата.Абонемент =Найденные[0].Абонемент;
                Возврат СтруктураВозврата;
        Конецесли;

        Отбор = Новый Структура;
        Отбор.Вставить("СостояниеАбонемента", Перечисления.СостоянияАбонементов.Приостановлен);
        Найденные = Абонементы.НайтиСтроки(Отбор);
        Если Найденные.Количество() > 0 Тогда
                СтруктураВозврата.Действие = "Возобновление";
                СтруктураВозврата.Абонемент =Найденные[0].Абонемент;
                Возврат СтруктураВозврата;
        Конецесли;
```

```
		Отбор = Новый Структура;
		Отбор.Вставить("СостояниеАбонемента", Перечисления.СостоянияАбонементов.Заблокирован);
		Найденные = Абонементы.НайтиСтроки(Отбор);
		Если Найденные.Количество() > 0 Тогда
			СтруктураВозврата.Действие = "Разблокировка";
			СтруктураВозврата.Абонемент =Найденные[0].Абонемент;
			Возврат СтруктураВозврата;
		Конецесли;

		Возврат СтруктураВозврата;

КонецФункции

&НаКлиенте
Процедура ОбработкаОтветаПоТовару(Товар) Экспорт

		ПараметрыЧека = ПолучитьКассуККМИТерминалПоУмолчанию();
		ПараметрыЧека.Вставить("Товар", Товар);
		ОткрытьФорму("Документ.ЧекККМ.ФормаОбъекта", ПараметрыЧека);

КонецПроцедуры

&Насервере
Процедура ПоискТовара(Штрихкод, ТоварНайден, Товар)

		СписокШтрихкодов = Новый Массив;
		СписокШтрихкодов.Добавить(Новый Структура("ШтрихКод",Штрихкод));
		ДанныеПоТовару =
РегистрыСведений.ШтрихкодыНоменклатуры.ПолучитьДанныеПоШтрихкодам(СписокШтрихкодов);

		Если ДанныеПоТовару[Штрихкод].Количество() > 0 Тогда
			Товар = ДанныеПоТовару[Штрихкод].Номенклатура;
			ТоварНайден = Истина;
		Иначе
			ТоварНайден = Ложь;
		КонецЕсли;

КонецПроцедуры

Функция ПолучитьДанныеЗаполненияПосещения(Карта)

		ПараметрыПосещения = Новый Структура;
		ПараметрыПосещения.Вставить("Спортсмен", Карта.Владелец);
		Возврат ПараметрыПосещения;

КонецФункции

&НаКлиенте
Процедура ПриЗакрытии()

		// ПодключаемоеОборудование
		МенеджерОборудованияКлиентПереопределяемый.НачатьОтключениеОборудованиеПриЗакрытии
Формы(ЭтаФорма);
		// Конец ПодключаемоеОборудование

КонецПроцедуры

&НаКлиенте
Процедура ШтрихкодПриИзменении(Элемент)

		Если ЗначениеЗаполнено(Штрихкод) Тогда
```

```
            ОбработатьШтрихкод(Штрихкод);
        Конецесли;

КонецПроцедуры

Функция ПолучитьКассуККМИТерминалПоУмолчанию() Экспорт

        СтруктураПараметров = Новый Структура("КассаККМ, ЭквайринговыйТерминал,
КоличествоЭквайринговыхТерминалов, Организация, СтруктурнаяЕдиница",
            Справочники.КассыККМ.ПустаяСсылка(),
Справочники.ЭквайринговыеТерминалы.ПустаяСсылка(), 0);

        Запрос = Новый Запрос(
        "ВЫБРАТЬ ПЕРВЫЕ 2
        |    КассыККМ.Ссылка КАК КассаККМ,
        |    КассыККМ.СтруктурнаяЕдиница,
        |    КассыККМ.Владелец КАК Организация
        |ПОМЕСТИТЬ КассыККМ
        |ИЗ
        |    Справочник.КассыККМ КАК КассыККМ
        |ГДЕ
        |    НЕ КассыККМ.ПометкаУдаления
        |    И КассыККМ.ТипКассы = &ТипКассы
        |;
        |
        |////////////////////////////////////////////////////////////////////////////////
        |ВЫБРАТЬ
        |    ЭквайринговыеТерминалы.Ссылка КАК ЭквайринговыйТерминал,
        |    ЭквайринговыеТерминалы.Касса
        |ИЗ
        |    Справочник.ЭквайринговыеТерминалы КАК ЭквайринговыеТерминалы
        |        ВНУТРЕННЕЕ СОЕДИНЕНИЕ КассыККМ КАК КассыККМ
        |        ПО ЭквайринговыеТерминалы.Касса = КассыККМ.КассаККМ
        |ГДЕ
        |    НЕ ЭквайринговыеТерминалы.ПометкаУдаления
        |    И ЭквайринговыеТерминалы.Касса.ТипКассы = &ТипКассы
        |;
        |
        |////////////////////////////////////////////////////////////////////////////////
        |ВЫБРАТЬ
        |    КассыККМ.КассаККМ,
        |    КассыККМ.СтруктурнаяЕдиница,
        |    КассыККМ.Организация
        |ИЗ
        |    КассыККМ КАК КассыККМ");

        Запрос.УстановитьПараметр("ТипКассы", Перечисления.ТипыКассККМ.ФискальныйРегистратор);

        МРезультатов = Запрос.ВыполнитьПакет();
        Выборка = МРезультатов[2].Выбрать();
        КоличествоКассККМ = Выборка.Количество();
        Если КоличествоКассККМ = 1
          И Выборка.Следующий() Тогда

            СтруктураПараметров.КассаККМ = Выборка.КассаККМ;
            СтруктураПараметров.СтруктурнаяЕдиница = Выборка.СтруктурнаяЕдиница;
            СтруктураПараметров.Организация = Выборка.Организация;

        Иначе

            СтруктураПараметров.КассаККМ = Неопределено;
```

```
		КонецЕсли;

		Если КоличествоКассККМ = 1 Тогда
			ТЗ_ЭТ = МРезультатов[1].Выгрузить();

			ЭТКассыПоУмолчанию = ТЗ_ЭТ.НайтиСтроки(Новый Структура("Касса",
СтруктураПараметров.КассаККМ));

			СтруктураПараметров.КоличествоЭквайринговыхТерминалов =
ЭТКассыПоУмолчанию.Количество();
			Если ЭТКассыПоУмолчанию.Количество() = 1 Тогда

				СтруктураПараметров.ЭквайринговыйТерминал =
ЭТКассыПоУмолчанию[0].ЭквайринговыйТерминал;

			Иначе

				СтруктураПараметров.ЭквайринговыйТерминал = Неопределено;

			КонецЕсли;
		Иначе
			СтруктураПараметров.ЭквайринговыйТерминал = Неопределено;
		КонецЕсли;

		Возврат СтруктураПараметров;

КонецФункции // ПолучитьКассуПоУмолчанию()

&НаКлиенте
Процедура СпортивнаяКвалификацияПриИзменении(Элемент)

		ПриИзмененииДанныхСпортсмена(Элемент.Имя);

КонецПроцедуры

&НаКлиенте
Процедура ТехническаяКвалификацияПриИзменении(Элемент)

		ПриИзмененииДанныхСпортсмена(Элемент.Имя);

КонецПроцедуры

&НаКлиенте
Процедура ДатаРожденияПриИзменении(Элемент)

		ПриИзмененииДанныхФизлица(Элемент.Имя);

КонецПроцедуры

&НаКлиенте
Процедура ПолПриИзменении(Элемент)

		ПриИзмененииДанныхФизлица(Элемент.Имя);

КонецПроцедуры

&НаКлиенте
Процедура EMailПриИзменении(Элемент)
		EMailПриИзмененииНаСервере();
КонецПроцедуры

&НаСервере
```

```
Процедура EMailПриИзмененииНаСервере()

	Если Не ЗначениеЗаполнено(EMail) Тогда
		Возврат;
	КонецЕсли;

	Физлицо = Объект.Спортсмен.ФизЛицо;
	ФизлицоОбъект = Физлицо.ПолучитьОбъект();
	Если НЕ НомерСтрокиEmail = 0 Тогда
		НомерСтроки = НомерСтрокиEmail - 1;
		СтрокаEmail = ФизлицоОбъект.КонтактнаяИнформация[НомерСтроки];
		СтрокаEmail.АдресЭП = EMail;
		СтрокаEmail.Представление = EMail;
		СтрокаEmail.ЗначенияПолей =
УправлениеКонтактнойИнформациейСлужебныйВызовСервера.КонтактнаяИнформацияXMLПоПредставле
нию(EMail, СтрокаEmail.Вид);
	Иначе
		НоваяСтрока = ФизлицоОбъект.КонтактнаяИнформация.Добавить();
		НоваяСтрока.АдресЭП = EMail;
		НоваяСтрока.Представление = EMail;
		Новаястрока.Тип = Перечисления.ТипыКонтактнойИнформации.АдресЭлектронноПочты;
		НоваяСтрока.Вид = Константы.ВидКонтактнойИнформацииEmailФизлица.Получить();
		Поз = Найти(EMail, "@");
		Если Поз <> 0 Тогда
			НоваяСтрока.ДоменноеИмяСервера = Сред(EMail, Поз+1);
		КонецЕсли;
		НоваяСтрока.ЗначенияПолей =
УправлениеКонтактнойИнформациейСлужебныйВызовСервера.КонтактнаяИнформацияXMLПоПредставле
нию(EMail, НоваяСтрока.Вид);
		НомерСтрокиEmail = Новаястрока.НомерСтроки;
	КонецЕсли;
	ФизлицоОбъект.Записать();

КонецПроцедуры

&НаКлиенте
Процедура ТелефонНачалоВыбора(Элемент, ДанныеВыбора, СтандартнаяОбработка)

	Если НомерСтрокиТелефон = 0 Тогда
		Вид =
ПредопределенноеЗначение("Справочник.ВидыКонтактнойИнформации.ТелефонФизЛица");
		ЗначенияПолей = "";
		Представление = "";
		Комментарий = "";
	Иначе
		ПараметрыТелефона = ПолучитьПараметрыТелефона();
		Вид = ПараметрыТелефона.Вид;
		ЗначенияПолей = ПараметрыТелефона.ЗначенияПолей;
		Представление = ПараметрыТелефона.Представление;
	КонецЕсли;

	ПараметрыОткрытия = Новый Структура;
	ПараметрыОткрытия.Вставить("ВидКонтактнойИнформации", Вид);
	ПараметрыОткрытия.Вставить("ЗначенияПолей", ЗначенияПолей);
	ПараметрыОткрытия.Вставить("Представление", Представление);
	ПараметрыОткрытия.Вставить("Комментарий", Комментарий);

	Оповещение = Новый ОписаниеОповещения("ТелефонНачалоВыбораЗавершение", ЭтотОбъект,
Новый Структура);
	//Оповещение.ДополнительныеПараметры.Вставить("ДанныеЗаполнения",  ДанныеЗаполнения);
	//Оповещение.ДополнительныеПараметры.Вставить("ЭтоТабличнаяЧасть", ЭтоТабличнаяЧасть);
	//Оповещение.ДополнительныеПараметры.Вставить("ДанныеСтроки",      ДанныеСтроки);
```

```
	//Оповещение.ДополнительныеПараметры.Вставить("Элемент",		Элемент);
	//Оповещение.ДополнительныеПараметры.Вставить("Результат",	Результат);
	//Оповещение.ДополнительныеПараметры.Вставить("Форма",		Форма);

	УправлениеКонтактнойИнформациейКлиент.ОткрытьФормуКонтактнойИнформации(ПараметрыОт
крытия,, Оповещение);

	//ТелефонНачалоВыбораНаСервере();

КонецПроцедуры

&НаСервере
Функция ПолучитьПараметрыТелефона()

	Структура = Новый Структура;
	Строка = Объект.Спортсмен.ФизЛицо.КонтактнаяИнформация[НомерСтрокиТелефон-1];
	Структура.Вставить("Вид", Строка.Вид);
	Структура.Вставить("ЗначенияПолей", Строка.ЗначенияПолей);
	Структура.Вставить("Представление", Строка.Представление);
	Возврат Структура;

КонецФункции

Процедура ТелефонНачалоВыбораЗавершение(РезультатЗакрытия, ДополнительныеПараметры)

	Если НЕ РезультатЗакрытия = Неопределено Тогда

		Физлицо = Объект.Спортсмен.ФизЛицо;
		ФизлицоОбъект = Физлицо.ПолучитьОбъект();
		Если НЕ НомерСтрокиТелефон = 0 Тогда
			НомерСтроки = НомерСтрокиТелефон - 1;
			СтрокаТелефона = ФизлицоОбъект.КонтактнаяИнформация[НомерСтроки];
		Иначе
			СтрокаТелефона = ФизлицоОбъект.КонтактнаяИнформация.Добавить();
			СтрокаТелефона.Вид =
ПредопределенноеЗначение("Справочник.ВидыКонтактнойИнформации.ТелефонФизЛица");
			СтрокаТелефона.Тип =
ПредопределенноеЗначение("Перечисление.ТипыКонтактнойИнформации.Телефон");
			НомерСтрокиТелефон = СтрокаТелефона.НомерСтроки;
		КонецЕсли;
		СтрокаТелефона.Представление = РезультатЗакрытия.Представление;
		СтрокаТелефона.ЗначенияПолей = РезультатЗакрытия.КонтактнаяИнформация;

	УправлениеКонтактнойИнформацией.ЗаполнитьДополнительныеРеквизитыКонтактнойИнформаци
и(СтрокаТелефона, РезультатЗакрытия.Представление, РезультатЗакрытия.КонтактнаяИнформация);
		Телефон = СтрокаТелефона.НомерТелефона;
		ФизлицоОбъект.Записать();

	КонецЕсли;

КонецПроцедуры

Функция ТекстДолга()

	Запрос = Новый Запрос;
	Запрос.Текст = "ВЫБРАТЬ
		|	ВзаиморасчетыПоАбонементамОстатки.СуммаОстаток
		|ИЗ
		|	РегистрНакопления.ВзаиморасчетыПоАбонементам.Остатки(&ДатаДолга
		|			,
		|			Спортсмен = &Спортсмен
		|				И ВЫБОР
```

```
                |                                                КОГДА &ЕстьОтборПоАбонементу
                |                                                        ТОГДА Абонемент = &Абонемент
                |                                                ИНАЧЕ ИСТИНА
                |                                            КОНЕЦ) КАК ВзаиморасчетыПоАбонементамОстатки";
        Запрос.УстановитьПараметр("Спортсмен", Объект.Спортсмен);
        Запрос.УстановитьПараметр("Абонемент", Объект.Абонемент);
        Запрос.УстановитьПараметр("ДатаДолга", ТекущаяДата());
        Запрос.УстановитьПараметр("ЕстьОтборПоАбонементу", ЗначениеЗаполнено(Объект.Абонемент));

        Результат = Запрос.Выполнить();
        Если Не Результат.Пустой() Тогда
                Выборка = Результат.Выбрать();
                Если Выборка.Следующий() Тогда
                        Долг = Выборка.СуммаОстаток;
                Конецесли;
        Иначе
                Долг = 0;
        КонецЕсли;

        Если Долг >= 0 Тогда
                ДолгАванс = "Долг";
        Иначе
                ДолгАванс = "Аванс";
        КонецЕсли;

        Если ЗначениеЗаполнено(Объект.Спортсмен) Тогда
                Если ЗначениеЗаполнено(Объект.Абонемент) Тогда
                        ТекстДолга = ДолгАванс + ": " + ?(Долг < 0, Долг * (-1), Долг) + " руб.";
                Иначе
                        ТекстДолга = ДолгАванс + ": " + ?(Долг < 0, Долг * (-1), Долг) + " руб.";
                КонецЕсли;
        КонецЕсли;


        Возврат ТекстДолга;

КонецФункции

&НаКлиенте
Процедура АбонементПриИзменении(Элемент)
        Элементы.НадписьДолг.Заголовок = ТекстДолга();
КонецПроцедуры


#Область РаботаСФото

&НаКлиенте

Процедура ВыбратьФото(Элемент)
        Если ЗначениеЗаполнено(Объект.Спортсмен) Тогда
                Оповещение = Новый ОписаниеОповещения("ОбработатьВыборФайла", ЭтотОбъект);
                НачатьПомещениеФайла(Оповещение, , , Истина, УникальныйИдентификатор);
        Иначе
                Сообщить("Выберите спортсмена");
        КонецЕсли;
КонецПроцедуры

&НаКлиенте
Процедура ОбработатьВыборФайла(Результат, Адрес, ВыбранноеИмяФайла, ДополнительныеПараметры)
Экспорт
        Если Не Результат Тогда
                Возврат;
```

```
		КонецЕсли;
		ССылкаНаФото = Адрес;
		ЗаписатьФотоНаСервере(Объект.Спортсмен, ССылкаНаФото);
КонецПроцедуры

&НаСервере
Процедура ЗаписатьФотоНаСервере(ТекущийОбъект, ССылкаНаФото)
		ТекущийОбъект = ТекущийОбъект.ПолучитьОбъект();
		Если ЭтоАдресВременногоХранилища(ССылкаНаФото) Тогда
			ТекущийОбъект.Фото = Новый
ХранилищеЗначения(ПолучитьИзВременногоХранилища(ССылкаНаФото));
		КонецЕсли;
КонецПроцедуры

&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)

		ССылкаНаФото = ПолучитьНавигационнуюССылку(Объект.Спортсмен, "Фото");
		//Если ЗначениеЗаполнено(Объект.ФизЛицо) Тогда
		//		ТелефонСп =
УправлениеКонтактнойИнформацией.КонтактнаяИнформацияОбъекта(Объект.ФизЛицо.Ссылка,
Справочники.ВидыКонтактнойИнформации.ТелефонФизЛица);
		//		ПочтаСп =
УправлениеКонтактнойИнформацией.КонтактнаяИнформацияОбъекта(Объект.ФизЛицо.Ссылка,
Справочники.ВидыКонтактнойИнформации.EmailКонтактногоЛица);
		//КонецЕсли;
		ИспользоватьПодключаемоеОборудование =
УправлениеНебольшойФирмойПовтИсп.ИспользоватьПодключаемоеОборудование();
		ПоказыватьВсеАбонементы = Ложь;
КонецПроцедуры

&НаСервереБезКонтекста
Процедура ЗавершитьДеньНаСервере()
		СпортсменыВКлубе = УправлениеСпортклубом.СпортсменыВКлубе();
		Для каждого СпортсменВКлубе Из СпортсменыВКлубе Цикл
			ДокументПосещения = СпортсменВКлубе.ДокументПосещения.ПолучитьОбъект();
			ДокументПосещения.ДатаВремяВыхода = ТекущаяДата();
			ДокументПосещения.Записать(РежимЗаписиДокумента.Проведение);
		КонецЦикла;
КонецПроцедуры

&НаКлиенте
Процедура ЗавершитьДень(Команда)
		ЗавершитьДеньНаСервере();
		Элементы.СпортсменыВКлубе.Обновить();
КонецПроцедуры

&НаКлиенте
Процедура СпортсменыВКлубеВыбор(Элемент, ВыбраннаяСтрока, Поле, СтандартнаяОбработка)
		СтрокаСпортсмена = Элемент.ТекущиеДанные;
		ДокументПосещения = СтрокаСпортсмена.ДокументПосещения;
		ОткрытьФорму("Документ.Посещения.ФормаОбъекта", Новый Структура("Ключ",
ДокументПосещения));
КонецПроцедуры

&НаКлиенте
Процедура СостоянияАбонементов(Команда)
		ПоказатьДвижениеПоАбонементам("РегистрСведений.СостоянияАбонементов");
КонецПроцедуры

&НаКлиенте
Процедура ВзаиморасчетыПоАбонементам(Команда)
```

```
	ПоказатьДвижениеПоАбонементам("РегистрНакопления.ВзаиморасчетыПоАбонементам");
КонецПроцедуры

&НаКлиенте
Процедура ЗанятияПоАбонементам(Команда)
	ПоказатьДвижениеПоАбонементам("РегистрНакопления.ЗанятияПоАбонементам");
КонецПроцедуры

&НаКлиенте
Процедура ПоказатьДвижениеПоАбонементам(НаименованиеДвижения)
	Стр = Элементы.Абонементы.ТекущиеДанные;
	Если Стр <> Неопределено Тогда
		АбонементСпортсмена = Стр.АбонементСпортсмена;

		Если ЗначениеЗаполнено(АбонементСпортсмена) Тогда
			ФормаСпискаРегистра = ПолучитьФорму(НаименованиеДвижения +
".ФормаСписка", , ЭтаФорма);
			ЭлОтбора =
ФормаСпискаРегистра.Список.Отбор.Элементы.Добавить(Тип("ЭлементОтбораКомпоновкиДанных"));
			ЭлОтбора.ВидСравнения = ВидСравненияКомпоновкиДанных.Равно;
			ЭлОтбора.Использование = Истина;
			ЭлОтбора.ПравоеЗначение = АбонементСпортсмена;
			ЭлОтбора.ЛевоеЗначение = Новый ПолеКомпоновкиДанных("Абонемент");

			ФормаСпискаРегистра.Открыть();

		КонецЕсли;
	КонецЕсли;
КонецПроцедуры

&НаКлиенте
Процедура ПриостановитьАбонемент(Команда)
	ВидОперации =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Приостановление");
	СоздатьДокументИзменениеАбонемента(ВидОперации);
КонецПроцедуры

&НаКлиенте
Процедура ВозобновитьДействиеАбонемента(Команда)
	ВидОперации =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Возобновление");

	СоздатьДокументИзменениеАбонемента(ВидОперации);
КонецПроцедуры

&НаКлиенте
Процедура ПродлитьАбонемент(Команда)
	ВидОперации =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Продление");
	СоздатьДокументИзменениеАбонемента(ВидОперации);
КонецПроцедуры

&НаКлиенте
Процедура ПередатьАбонемент(Команда)
	ВидОперации =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Передача");
	СоздатьДокументИзменениеАбонемента(ВидОперации);
КонецПроцедуры

&НаКлиенте
Процедура ЗаблокироватьАбонемент(Команда)
```

```
	ВидОперации =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Блокировка");
	СоздатьДокументИзменениеАбонемента(ВидОперации);
КонецПроцедуры

&НаКлиенте
Процедура РазблокироватьАбонемент(Команда)
	ВидОперации =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Разблокировка");

	СоздатьДокументИзменениеАбонемента(ВидОперации);
КонецПроцедуры

&НаКлиенте
Процедура ЗакрытьАбонемент(Команда)
	ВидОперации =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Закрытие");
	СоздатьДокументИзменениеАбонемента(ВидОперации);
КонецПроцедуры

&НаКлиенте
Процедура СоздатьДокументИзменениеАбонемента(ВидОперации)
	Стр = Элементы.Абонементы.ТекущиеДанные;
	Если Стр <> Неопределено Тогда
		Абонемент = Стр.АбонементСпортсмена;

		СтруктураДанных = ПолучитьДанныеДляЗаполненияДокументаИзменениеАбонемента();
		СтруктураДанных.Вставить("ВидОперации", ВидОперации);
		СтруктураДанных.Вставить("Абонемент", Абонемент);

		ОпОповещения = Новый ОписаниеОповещения("ОбновитьАбонементыНаФорме",
ЭтаФорма, Неопределено);

		ОткрытьФорму("Документ.ИзменениеАбонемента.Форма.ФормаДокумента",
СтруктураДанных, ЭтаФорма,,,, ОпОповещения);
	Иначе
		Сообщить("Выберите абонемент.");
	КонецЕсли;
КонецПроцедуры

&НаСервереБезКонтекста
Функция ПолучитьДанныеДляЗаполненияДокументаИзменениеАбонемента()
	ТекПользователь = ПараметрыСеанса.ТекущийПользователь;

	Запрос = Новый Запрос;
	Запрос.Текст =
		"ВЫБРАТЬ
		|	НастройкиПользователей.Значение КАК Ответственный,
		|	НастройкиПользователей1.Значение КАК Организация
		|ИЗ
		|	РегистрСведений.НастройкиПользователей КАК НастройкиПользователей
		|		ПОЛНОЕ СОЕДИНЕНИЕ РегистрСведений.НастройкиПользователей КАК
НастройкиПользователей1
		|		ПО НастройкиПользователей.Пользователь =
НастройкиПользователей1.Пользователь
		|ГДЕ
		|	НастройкиПользователей.Пользователь = &Пользователь
		|	И НастройкиПользователей.Настройка = &НастройкаОтветственный
		|	И НастройкиПользователей1.Настройка = &НастройкаОрганизация";
	Запрос.УстановитьПараметр("НастройкаОтветственный",
ПланыВидовХарактеристик.НастройкиПользователей.ОсновнойОтветственный);
```

```
	Запрос.УстановитьПараметр("НастройкаОрганизация",
ПланыВидовХарактеристик.НастройкиПользователей.ОсновнаяОрганизация);
	Запрос.УстановитьПараметр("Пользователь", ТекПользователь);
	Результат = Запрос.Выполнить();

	Если НЕ Результат.Пустой() Тогда
		Выборка = Результат.Выбрать();
		Выборка.Следующий();

		Сотрудник = Выборка.Ответственный;
		Организация = Выборка.Организация;
	Иначе
		Сотрудник = Справочники.Сотрудники.ПустаяСсылка();
		Организация = Справочники.Организации.ПустаяСсылка();
	КонецЕсли;

	СтруктураДанных = Новый Структура;
	СтруктураДанных.Вставить("Автор", ТекПользователь);
	СтруктураДанных.Вставить("Сотрудник", Сотрудник);
	СтруктураДанных.Вставить("Организация", Организация);

	Возврат СтруктураДанных;
КонецФункции

&НаКлиенте
Процедура АбонементыПриАктивизацииСтроки(Элемент)
	Если Элемент.ТекущиеДанные <> Неопределено Тогда
		СостояниеАбонемента = Элемент.ТекущиеДанные.Состояние;

		СтруктураДанных = Новый Структура;

		Если СостояниеАбонемента =
ПредопределенноеЗначение("Перечисление.СостоянияАбонементов.Активен") Тогда
			СтруктураДанных.Вставить("КнопкаПриостановить",	Истина);
			СтруктураДанных.Вставить("КнопкаВозобнавить",		Ложь);
			СтруктураДанных.Вставить("КнопкаЗакрыть",		Истина);
			СтруктураДанных.Вставить("КнопкаПродлить",		Истина);
			СтруктураДанных.Вставить("КнопкаПередать",		Истина);
			СтруктураДанных.Вставить("КнопкаЗаблокировать",	Истина);
			СтруктураДанных.Вставить("КнопкаРазблокировать",	Ложь);
		ИначеЕсли СостояниеАбонемента =
ПредопределенноеЗначение("Перечисление.СостоянияАбонементов.Приостановлен") Тогда
			СтруктураДанных.Вставить("КнопкаПриостановить",	Ложь);
			СтруктураДанных.Вставить("КнопкаВозобнавить",		Истина);
			СтруктураДанных.Вставить("КнопкаЗакрыть",		Истина);
			СтруктураДанных.Вставить("КнопкаПродлить",		Ложь);
			СтруктураДанных.Вставить("КнопкаПередать",		Истина);
			СтруктураДанных.Вставить("КнопкаЗаблокировать",	Истина);
			СтруктураДанных.Вставить("КнопкаРазблокировать",	Ложь);
		ИначеЕсли СостояниеАбонемента =
ПредопределенноеЗначение("Перечисление.СостоянияАбонементов.Истек") Тогда
			СтруктураДанных.Вставить("КнопкаПриостановить",	Ложь);
			СтруктураДанных.Вставить("КнопкаВозобнавить",		Ложь);
			СтруктураДанных.Вставить("КнопкаЗакрыть",		Ложь);
			СтруктураДанных.Вставить("КнопкаПродлить",		Истина);
			СтруктураДанных.Вставить("КнопкаПередать",		Истина);
			СтруктураДанных.Вставить("КнопкаЗаблокировать",	Ложь);
			СтруктураДанных.Вставить("КнопкаРазблокировать",	Ложь);
		ИначеЕсли СостояниеАбонемента =
ПредопределенноеЗначение("Перечисление.СостоянияАбонементов.Заблокирован") Тогда
			СтруктураДанных.Вставить("КнопкаПриостановить",	Ложь);
			СтруктураДанных.Вставить("КнопкаВозобнавить",		Ложь);
```

```
                        СтруктураДанных.Вставить("КнопкаЗакрыть",                   Истина);
                        СтруктураДанных.Вставить("КнопкаПродлить",                  Ложь);
                        СтруктураДанных.Вставить("КнопкаПередать",                  Истина);
                        СтруктураДанных.Вставить("КнопкаЗаблокировать",     Ложь);
                        СтруктураДанных.Вставить("КнопкаРазблокировать",    Истина);
                Иначе
                        Возврат;
                КонецЕсли;

                УправлениеОтображениемКнопокПанели(СтруктураДанных);
        КонецЕсли;
КонецПроцедуры

&НаКлиенте
Процедура УправлениеОтображениемКнопокПанели(СтруктураДанных)
        Элементы.ПриостановитьАбонементТаб.Видимость                    =
СтруктураДанных.КнопкаПриостановить;
        Элементы.ВозобновитьАбонементТаб.Видимость                              =
СтруктураДанных.КнопкаВозобнавить;
        Элементы.АбонементыЗакрытьАбонемент.Видимость                   =
СтруктураДанных.КнопкаЗакрыть;
        Элементы.АбонементыПродлитьАбонемент.Видимость                  =
СтруктураДанных.КнопкаПродлить;
        //Элементы.АбонементыПередатьАбонемент.Видимость                =
СтруктураДанных.КнопкаПередать;
        Элементы.АбонементыЗаблокироватьАбонемент.Видимость     =
СтруктураДанных.КнопкаЗаблокировать;
        Элементы.АбонементыРазблокироватьАбонемент.Видимость    =
СтруктураДанных.КнопкаРазблокировать;
КонецПроцедуры

&НаКлиенте
Процедура ОбновитьАбонементыНаФорме(Результат, Параметры) Экспорт
        Если Результат <> Неопределено Тогда
                Если ТипЗнч(Результат) = Тип("СправочникСсылка.Спортсмены") Тогда
                        ЗаполнитьАбонементыСпортсмена(Результат);
                КонецЕсли;
        КонецЕсли;
КонецПроцедуры

&НаКлиенте
Процедура ПродатьАбонементПовторно(Команда)
        ПараметрыПосещения = Новый Структура;
        ПараметрыПосещения.Вставить("Спортсмен", Объект.Спортсмен);
        Если ЗначениеЗаполнено(КартаСпортсмена) Тогда
                ПараметрыПосещения.Вставить("Карта", КартаСпортсмена);
        Иначе
                ПараметрыПосещения.Вставить("Карта",
УправлениеСпортклубом.ПолучитьКартуСпортсмена(Объект.Спортсмен));
        КонецЕсли;

        Стр = Элементы.Абонементы.ТекущиеДанные;
        СтруктураДанных =
ПолучитьМассивДанныхДляЗаполненяТабЧастиАбонементы(Стр.АбонементСпортсмена);
        ПараметрыПосещения.Вставить("ДанныеДляЗаполненияТабЧастиАбонементы",
СтруктураДанных);
        ОткрытьФорму("Документ.ПродажаАбонемента.Форма.ФормаДокумента", ПараметрыПосещения,
ЭтаФорма);
КонецПроцедуры

&НаСервереБезКонтекста
Функция ПолучитьМассивДанныхДляЗаполненяТабЧастиАбонементы(Абонемент)
```

```
	СтруктураДанных = Новый Структура;
	СтруктураДанных.Вставить("Абонемент", Абонемент);
	СтруктураДанных.Вставить("ВидАбонемента", Абонемент.ВидАбонемента);

	Возврат СтруктураДанных;
КонецФункции

&НаКлиенте
Процедура КомментарийПриИзменении(Элемент)
	КомментарийПриИзмененииНаСервере();
КонецПроцедуры

&НаСервере
Процедура КомментарийПриИзмененииНаСервере()
	СпортсменОбъект = Объект.Спортсмен.ПолучитьОбъект();
	СпортсменОбъект.Комментарий = Комментарий;
	СпортсменОбъект.Записать();
КонецПроцедуры

&НаКлиенте
Процедура ВсеАбонементы(Команда)
	ПоказыватьВсеАбонементы = НЕ ПоказыватьВсеАбонементы;
	ЗаполнитьАбонементыСпортсмена(Объект.Спортсмен);
	ЭтаФорма.Элементы.АбонементыВсеАбонементы.Заголовок = ?(ПоказыватьВсеАбонементы,
"Скрыть", "Показать");
КонецПроцедуры

&НаКлиенте
Процедура Обновить(Команда)
	ТекстДолга();
	СпортсменПриИзмененииНаСервере();
КонецПроцедуры


#КонецОбласти


ОБРАБОТКА АРМТренера Форма

&НаКлиенте
Процедура Заполнить(Команда)
	ЗаполнениеПосетителей();
	ОформлениеСтрок();
КонецПроцедуры

&НаСервереБезКонтекста
Процедура СоздатьДокументУчетаПроведенныхЗанятий(СтруктураДанных)
	ТекущаяДата = ТекущаяДата();
	Тренер = СтруктураДанных.Тренер;
	Пользователь = Тренер.Пользователь;
	ВидЗанятия = СтруктураДанных.ВидЗанятий;
	ВремяНачала = СтруктураДанных.ВремяНачала;
	ВремяОкончания = СтруктураДанных.ВремяОкончания;
	Группа = СтруктураДанных.Группа;
	Комментарий = "#Создан автоматически из АРМ тренера!";
	Организация = СтруктураДанных.Организация;
	Сотрудник = Тренер.Сотрудник;
	Помещение = СтруктураДанных.Помещение;
	Услуга = СтруктураДанных.Услуга;

	Запрос = Новый Запрос;
	Запрос.Текст =
```

```
		"ВЫБРАТЬ
		|	УчетПроведенныхЗанятий.Ссылка
		|ИЗ
		|	Документ.УчетПроведенныхЗанятий КАК УчетПроведенныхЗанятий
		|ГДЕ
		|	УчетПроведенныхЗанятий.Проведен
		|	И УчетПроведенныхЗанятий.Автор = &Автор
		|	И УчетПроведенныхЗанятий.ВидЗанятия = &ВидЗанятия
		|	И УчетПроведенныхЗанятий.ВремяНачала = &ВремяНачала
		|	И УчетПроведенныхЗанятий.ВремяОкончания = &ВремяОкончания
		|	И УчетПроведенныхЗанятий.Дата МЕЖДУ &ДатаНачала И &ДатаОкончания
		|	И УчетПроведенныхЗанятий.Группа = &Группа
		|	И УчетПроведенныхЗанятий.Комментарий ПОДОБНО &Комментарий
		|	И УчетПроведенныхЗанятий.Организация = &Организация
		|	И УчетПроведенныхЗанятий.Помещение = &Помещение
		|	И УчетПроведенныхЗанятий.Тренер = &Тренер
		|	И УчетПроведенныхЗанятий.Услуга = &Услуга";
Запрос.УстановитьПараметр("Автор", Пользователь);
Запрос.УстановитьПараметр("ВидЗанятия", ВидЗанятия);
Запрос.УстановитьПараметр("ВремяНачала", ВремяНачала);
Запрос.УстановитьПараметр("ВремяОкончания", ВремяОкончания);
Запрос.УстановитьПараметр("Группа", Группа);
Запрос.УстановитьПараметр("ДатаНачала", НачалоДня(ТекущаяДата));
Запрос.УстановитьПараметр("ДатаОкончания", КонецДня(ТекущаяДата));
Запрос.УстановитьПараметр("Комментарий", Комментарий);
Запрос.УстановитьПараметр("Организация", Организация);
Запрос.УстановитьПараметр("Помещение", Помещение);
Запрос.УстановитьПараметр("Тренер", Тренер);
Запрос.УстановитьПараметр("Услуга", Услуга);
Результат = Запрос.Выполнить();

Если НЕ Результат.Пустой() Тогда
	Выборка = Результат.Выбрать();
	Выборка.Следующий();

	ДокОбъект = Выборка.Ссылка.ПолучитьОбъект();
	ДокОбъект.Записать(РежимЗаписиДокумента.ОтменаПроведения);
Иначе
	ДокОбъект = Документы.УчетПроведенныхЗанятий.СоздатьДокумент();
КонецЕсли;

ДокОбъект.Дата = ТекущаяДата;
ДокОбъект.Автор = Пользователь;
ДокОбъект.ВидЗанятия =  ВидЗанятия;
ДокОбъект.ВремяНачала = ВремяНачала;
ДокОбъект.ВремяОкончания = ВремяОкончания;
ДокОбъект.Группа = Группа;
ДокОбъект.Комментарий = Комментарий;
ДокОбъект.Организация = Организация;
ДокОбъект.Ответственный = Сотрудник;
ДокОбъект.Помещение = Помещение;
ДокОбъект.Тренер = Тренер;
ДокОбъект.Услуга = Услуга;

ТабСпортсмены = ДокОбъект.Спортсмены;
ТабСпортсмены.Очистить();
МассивДляЗаполнения = СтруктураДанных.МассивДляЗаполненияТабЧасти;

Для Каждого Эл Из МассивДляЗаполнения Цикл
	Стр = ТабСпортсмены.Добавить();
	ЗаполнитьЗначенияСвойств(Стр, Эл);
	Стр.Присутствовал = Эл.Отметка;
```

```
        КонецЦикла;

        ДокОбъект.Записать(РежимЗаписиДокумента.Проведение,
РежимПроведенияДокумента.Оперативный);
КонецПроцедуры

&НаКлиенте
Процедура ЗавершитьПерекличку(Команда)
        Выход = Ложь;
        Тренер = Объект.Тренер;
        Группа = Объект.Группа;
        Помещение = Объект.Помещение;
        Услуга = Объект.Услуга;
        ВремяНачала = Объект.ВремяНачала;
        ВремяОкончания = Объект.ВремяОкончания;
        ВидЗанятий = Объект.ВидЗанятий;
        Организация = ПолучитьРеквизитОбъекта(Объект.Занятие, "Организация");

        ТабПосетителиЗанятия = Объект.ПосетителиЗанятия;

        Если НЕ ЗначениеЗаполнено(Тренер) Тогда
                Сообщить("Не заполнено поле ""Тренер""!");
                Выход = Истина;
        КонецЕсли;

        Если НЕ ЗначениеЗаполнено(Группа) Тогда
                Сообщить("Не заполнено поле ""Группа""!");
                Выход = Истина;
        КонецЕсли;

        Если НЕ ЗначениеЗаполнено(Помещение) Тогда
                Сообщить("Не заполнено поле ""Помещение""!");
                Выход = Истина;
        КонецЕсли;

        Если НЕ ЗначениеЗаполнено(Услуга) Тогда
                Сообщить("Не заполнено поле ""Услуга""!");
                Выход = Истина;
        КонецЕсли;

        Если НЕ ЗначениеЗаполнено(ВремяНачала) Тогда
                Сообщить("Не заполнено поле ""Время начала""!");
                Выход = Истина;
        КонецЕсли;

        Если НЕ ЗначениеЗаполнено(ВремяОкончания) Тогда
                Сообщить("Не заполнено поле ""Время окончания""!");
                Выход = Истина;
        КонецЕсли;

        Если НЕ ЗначениеЗаполнено(ВидЗанятий) Тогда
                Сообщить("Не заполнено поле ""Вид занятий""!");
                Выход = Истина;
        КонецЕсли;

        Если ТабПосетителиЗанятия.Количество() = 0 Тогда
                Сообщить("Не заполнена табличная часть ""Посетители занятия""!");
                Выход = Истина;
        КонецЕсли;

        Если Выход Тогда
                Возврат;
```

```
        КонецЕсли;

        СтруктураДанных = Новый Структура("Тренер, Группа, Помещение, Услуга, ВремяНачала,
ВремяОкончания, ВидЗанятий, Организация, МассивДляЗаполненияТабЧасти");

        МассивДляЗаполненияТабЧасти = Новый Массив;
        ТабПосетителиЗанятия = Объект.ПосетителиЗанятия;

        Для Каждого Эл Из ТабПосетителиЗанятия Цикл
                СтруктураДляЗаполнения = Новый Структура("Спортсмен, Отметка, Абонемент,
СтатусЗанятия");
          ЗаполнитьЗначенияСвойств(СтруктураДляЗаполнения, Эл);

                МассивДляЗаполненияТабЧасти.Добавить(СтруктураДляЗаполнения);
        КонецЦикла;

        СтруктураДанных.Вставить("Тренер", Тренер);
        СтруктураДанных.Вставить("Группа", Группа);
        СтруктураДанных.Вставить("Помещение", Помещение);
        СтруктураДанных.Вставить("Услуга", Услуга);
        СтруктураДанных.Вставить("ВремяНачала", ВремяНачала);
        СтруктураДанных.Вставить("ВремяОкончания", ВремяОкончания);
        СтруктураДанных.Вставить("ВидЗанятий", ВидЗанятий);
        СтруктураДанных.Вставить("Организация", Организация);
        СтруктураДанных.Вставить("МассивДляЗаполненияТабЧасти", МассивДляЗаполненияТабЧасти);

        СоздатьДокументУчетаПроведенныхЗанятий(СтруктураДанных);

        Сообщить("Перекличка завершена!");
КонецПроцедуры

&НаКлиенте
Процедура ЗанятиеИзменилось(СтруктураДанных)
        Если СтруктураДанных <> Неопределено Тогда
                ЗанятиеПриИзмененииНаСервере(СтруктураДанных)
        Иначе
                Объект.ПосетителиЗанятия.Очистить();
                Объект.Группа = ПредопределенноеЗначение("Справочник.Группы.ПустаяСсылка");
                Объект.Помещение =
ПредопределенноеЗначение("Справочник.Помещения.ПустаяСсылка");
                Объект.Услуга = ПредопределенноеЗначение("Справочник.Номенклатура.ПустаяСсылка");
                Объект.ВидЗанятий =
ПредопределенноеЗначение("Перечисление.ВидыЗанятий.ПустаяСсылка");
                Объект.ВремяНачала = '00010101';
                Объект.ВремяОкончания = '00010101';
                Объект.ДатаВремяПроведения = '00010101';
                Объект.Занятие =
ПредопределенноеЗначение("Документ.ПланированиеЗанятий.ПустаяСсылка");
        КонецЕсли;
        ОформлениеСтрок();
КонецПроцедуры

&НаСервере
Процедура ЗанятиеПриИзмененииНаСервере(СтруктураДанных)
        ЗаполнитьЗначенияСвойств(Объект, СтруктураДанных);
        ЗаполнениеПосетителей();
КонецПроцедуры

&НаСервере
Процедура ЗаполнениеПосетителей()

        Запрос = Новый Запрос;
```

```
Запрос.Текст = "ВЫБРАТЬ
	|	СоставыГруппСрезПоследних.Спортсмен,
	|	ВЫБОР
	|		КОГДА СостоянияАбонементовСрезПоследних.СрокОкончания <
НАЧАЛОПЕРИОДА(&Дата, ДЕНЬ)
	|			ТОГДА ЛОЖЬ
	|		ИНАЧЕ ВЫБОР
	|				КОГДА
СостоянияАбонементовСрезПоследних.СостояниеАбонемента ЕСТЬ NULL
	|					ИЛИ НЕ
СостоянияАбонементовСрезПоследних.СостояниеАбонемента =
ЗНАЧЕНИЕ(Перечисление.СостоянияАбонементов.Активен)
	|					ТОГДА ЛОЖЬ
	|				ИНАЧЕ ВЫБОР
	|					КОГДА
ПосещенияСпортсменовСрезПоследних.СостояниеСпортсмена ЕСТЬ NULL
	|						И НЕ &Администратор
	|					ТОГДА ЛОЖЬ
	|					ИНАЧЕ ИСТИНА
	|				КОНЕЦ
	|			КОНЕЦ
	|	КОНЕЦ КАК ДоступностьДляРедактирования,
	|	СоставыГруппСрезПоследних.Абонемент,
	|	ФизическиеЛица.ДатаРождения,
	|	ЗанятияПоАбонементамОстатки.КоличествоОстаток КАК ЗанятийОсталось,
	|	СостоянияАбонементовСрезПоследних.СрокОкончания КАК
СрокДействияАбонемента,
	|	ЗНАЧЕНИЕ(Перечисление.СтатусыЗанятий.Абонемент) КАК СтатусЗанятия,
	|	СостоянияАбонементовСрезПоследних.СостояниеАбонемента,
	|	СостоянияАбонементовСрезПоследних.ДатаОграничения КАК ДатаОграничения
	|ПОМЕСТИТЬ СоставГруппы
	|ИЗ
	|	РегистрСведений.СоставыГрупп.СрезПоследних(&Дата, Группа = &Группа) КАК
СоставыГруппСрезПоследних
	|		ЛЕВОЕ СОЕДИНЕНИЕ
РегистрСведений.ПосещенияСпортсменов.СрезПоследних(
	|			&ДатаКонецДня,
	|			СостояниеСпортсмена =
ЗНАЧЕНИЕ(Перечисление.СостоянияСпортсменов.ВКлубе)
	|				И НАЧАЛОПЕРИОДА(&Дата, ДЕНЬ) =
НАЧАЛОПЕРИОДА(Период, ДЕНЬ)) КАК ПосещенияСпортсменовСрезПоследних
	|		ПО СоставыГруппСрезПоследних.Спортсмен =
ПосещенияСпортсменовСрезПоследних.Спортсмен
	|		ЛЕВОЕ СОЕДИНЕНИЕ Справочник.ФизическиеЛица КАК
ФизическиеЛица
	|		ПО СоставыГруппСрезПоследних.Спортсмен.ФизЛицо =
ФизическиеЛица.Ссылка
	|		ЛЕВОЕ СОЕДИНЕНИЕ
РегистрНакопления.ЗанятияПоАбонементам.Остатки КАК ЗанятияПоАбонементамОстатки
	|		ПО СоставыГруппСрезПоследних.Абонемент =
ЗанятияПоАбонементамОстатки.Абонемент
	|		ЛЕВОЕ СОЕДИНЕНИЕ
РегистрСведений.СостоянияАбонементов.СрезПоследних(
	|			,
	|			ДатаОграничения = ДАТАВРЕМЯ(1, 1, 1, 0, 0, 0)
	|				ИЛИ ДатаОграничения > &Дата) КАК
СостоянияАбонементовСрезПоследних
	|		ПО СоставыГруппСрезПоследних.Абонемент =
СостоянияАбонементовСрезПоследних.Абонемент
	|ГДЕ
	|	СоставыГруппСрезПоследних.Активен
	|;
```

```
                    |
                    |////////////////////////////////////////////////////////////////////////////
                    |ВЫБРАТЬ
                    |    УчетПроведенныхЗанятийСпортсмены.Ссылка КАК Ссылка1,
                    |    УчетПроведенныхЗанятийСпортсмены.НомерСтроки,
                    |    УчетПроведенныхЗанятийСпортсмены.Спортсмен,
                    |    УчетПроведенныхЗанятийСпортсмены.Присутствовал,
                    |    УчетПроведенныхЗанятийСпортсмены.Абонемент,
                    |    УчетПроведенныхЗанятийСпортсмены.СтатусЗанятия
                    |ПОМЕСТИТЬ УчетПосещений
                    |ИЗ
                    |    Документ.УчетПроведенныхЗанятий.Спортсмены КАК
УчетПроведенныхЗанятийСпортсмены
                    |ГДЕ
                    |    УчетПроведенныхЗанятийСпортсмены.Ссылка.Проведен
                    |    И НАЧАЛОПЕРИОДА(УчетПроведенныхЗанятийСпортсмены.Ссылка.Дата, ДЕНЬ)
= &ДатаНачала
                    |    И УчетПроведенныхЗанятийСпортсмены.Ссылка.Группа = &Группа
                    |    И УчетПроведенныхЗанятийСпортсмены.Ссылка.ВремяНачала = &ВремяНачала
                    |    И УчетПроведенныхЗанятийСпортсмены.Ссылка.ВремяОкончания =
&ВремяОкончания
                    |;
                    |
                    |////////////////////////////////////////////////////////////////////////////
                    |ВЫБРАТЬ
                    |    СоставГруппы.Спортсмен,
                    |    СоставГруппы.ДоступностьДляРедактирования,
                    |    СоставГруппы.Абонемент,
                    |    СоставГруппы.ДатаРождения,
                    |    СоставГруппы.ЗанятийОсталось,
                    |    СоставГруппы.СрокДействияАбонемента,
                    |    СоставГруппы.СтатусЗанятия,
                    |    ЕСТЬNULL(УчетПосещений.Присутствовал, ЛОЖЬ) КАК Присутствовал,
                    |    СоставГруппы.ДатаОграничения,
                    |    СоставГруппы.СостояниеАбонемента
                    |ИЗ
                    |    СоставГруппы КАК СоставГруппы
                    |        ЛЕВОЕ СОЕДИНЕНИЕ УчетПосещений КАК УчетПосещений
                    |        ПО СоставГруппы.Спортсмен = УчетПосещений.Спортсмен
                    |            И СоставГруппы.Абонемент = УчетПосещений.Абонемент";
            Выход = Ложь;
            Тренер = Объект.Тренер;
            Группа = Объект.Группа;
            Помещение = Объект.Помещение;
            Услуга = Объект.Услуга;
            ВремяНачала = Объект.ВремяНачала;
            ВремяОкончания = Объект.ВремяОкончания;
            ВидЗанятий = Объект.ВидЗанятий;
            Организация = ПолучитьРеквизитОбъекта(Объект.Занятие, "Организация");
            Комментарий = "#Создан автоматически из АРМ тренера!";
            Сотрудник = Тренер.Сотрудник;
            //СтатусЗавершено = Перечисления.СтатусыЗанятий.Завершено;
            Пользователь = Тренер.Пользователь;

            Запрос.УстановитьПараметр("Дата", ДатаАРМа);
            Запрос.УстановитьПараметр("ДатаКонецДня", КонецДня(ДатаАРМа));
            Запрос.УстановитьПараметр("Группа", Объект.Группа);
            Запрос.УстановитьПараметр("Администратор", Объект.Группа.ТренерАдминистратор);
            Запрос.УстановитьПараметр("Автор", Пользователь);
            Запрос.УстановитьПараметр("ВидЗанятия", ВидЗанятий);
            Запрос.УстановитьПараметр("ВремяНачала", ВремяНачала);
            Запрос.УстановитьПараметр("ВремяОкончания", ВремяОкончания);
```

```
	Запрос.УстановитьПараметр("Группа", Группа);
	Запрос.УстановитьПараметр("ДатаНачала", НачалоДня(ТекущаяДата()));
	Запрос.УстановитьПараметр("ДатаОкончания", КонецДня(ТекущаяДата()));
	Запрос.УстановитьПараметр("Комментарий", Комментарий);
	Запрос.УстановитьПараметр("Организация", Организация);
	Запрос.УстановитьПараметр("Помещение", Помещение);
	//Запрос.УстановитьПараметр("Статус", СтатусЗавершено);
	Запрос.УстановитьПараметр("Тренер", Тренер);
	Запрос.УстановитьПараметр("Услуга", Услуга);
	Результат = Запрос.Выполнить();
	Если Результат.Пустой() Тогда
		Сообщить("Нет данных для заполнения");
		Возврат;
	КонецЕсли;

	Обработка = РеквизитФормыВЗначение("Объект");
	Обработка.ПосетителиЗанятия.Очистить();

	Выборка = Результат.Выбрать();

	Пока Выборка.Следующий() Цикл
		НоваяСтрока = Обработка.ПосетителиЗанятия.Добавить();
		ЗаполнитьЗначенияСвойств(НоваяСтрока, Выборка);
		НоваяСтрока.Отметка = Выборка.Присутствовал;
		НоваяСтрока.По = Выборка.ДатаОграничения;
	КонецЦикла;

	ЗначениеВРеквизитФормы(Обработка, "Объект");

КонецПроцедуры

&НаСервере
Процедура ОформлениеСтрок()

	//Перенес в настройку условного оформления формы

	//Для каждого СтрТЧ Из Объект.ПосетителиЗанятия Цикл
	//	НеактивныйАбонемент = (НЕ СтрТЧ.СостояниеАбонемента =
Перечисления.СостоянияАбонементов.Активен);
	//	Если НЕ СтрТЧ.ДоступностьДляРедактирования ИЛИ НеактивныйАбонемент Тогда
	//	ЭлементУсловногоОформления = УсловноеОформление.Элементы.Добавить();
	//	ЭлементУсловногоОформления.Использование	= Истина;

	//	ОформлениеУО	= ЭлементУсловногоОформления.Оформление;
	//	ОтборУО		= ЭлементУсловногоОформления.Отбор;
	//	ОформляемыеПоляУО	= ЭлементУсловногоОформления.Поля;

	//	ОформлениеУО.УстановитьЗначениеПараметра("Доступность", Ложь);
	//		ОформлениеУО.УстановитьЗначениеПараметра("ТолькоПросмотр", Истина);
	//		ОформлениеУО.УстановитьЗначениеПараметра("ЦветТекста", WebЦвета.Серый);
	//
	//	ЭлементОтбора		=
ОтборУО.Элементы.Добавить(Тип("ЭлементОтбораКомпоновкиДанных"));
	//	ЭлементОтбора.ЛевоеЗначение	= ?(НеактивныйАбонемент, Новый
ПолеКомпоновкиДанных("Объект.ПосетителиЗанятия.СостояниеАбонемента"), Новый
ПолеКомпоновкиДанных("Объект.ПосетителиЗанятия.ДоступностьДляРедактирования"));
	//	ЭлементОтбора.ВидСравнения	= ?(НеактивныйАбонемент,
ВидСравненияКомпоновкиДанных.НеРавно, ВидСравненияКомпоновкиДанных.Равно);
	//	ЭлементОтбора.ПравоеЗначение = ?(НеактивныйАбонемент,
Перечисления.СостоянияАбонементов.Активен, Ложь);
	//	ЭлементОтбора.Использование	= Истина;
```

```
	//	Для каждого РеквизитТЧ Из
Метаданные["Обработки"]["АРМТренера"].ТабличныеЧасти["ПосетителиЗанятия"].Реквизиты Цикл

	//		НазваниеРеквизита	= РеквизитТЧ.Имя;
	//		ОформляемоеПоле		= ОформляемыеПоляУО.Элементы.Добавить();
	//		ОформляемоеПоле.Поле = Новый ПолеКомпоновкиДанных("ПосетителиЗанятия" +
НазваниеРеквизита);

	//	КонецЦикла;

	//	ОформляемоеПоле		= ОформляемыеПоляУО.Элементы.Добавить();
	//	ОформляемоеПоле.Поле = Новый ПолеКомпоновкиДанных("ПосетителиЗанятия" +
"НомерСтроки");
	//		КонецЕсли;
	//КонецЦикла;
	//
КонецПроцедуры

&НаКлиенте
Процедура ПриОткрытии(Отказ)
	ДатаАРМа = ТекущаяДата();
	ПерезаполнитьАРМТренера();
КонецПроцедуры

&НаКлиенте
Процедура ТренерПриИзменении(Элемент)
	ПерезаполнитьАРМТренера();
КонецПроцедуры

&НаКлиенте
Процедура ПерезаполнитьАРМТренера()
	ЗаполнитьРасписание();

	ТабРасписаниеЗанятий = Объект.РасписаниеЗанятий;

	СтруктураДанных = Неопределено;
	Если ТабРасписаниеЗанятий.Количество() <> 0 Тогда
		Стр = Объект.РасписаниеЗанятий[0];
		ЗаполнитьРеквизитыЗанятия(Стр);
	КонецЕсли;

КонецПроцедуры

&НаКлиенте
Процедура ЗаполнитьРеквизитыЗанятия(Знач Стр)

	Перем СтруктураДанных, ТренерАдминистратор;

	СтруктураДанных = Новый Структура("ДатаВремяПроведения, Группа, Услуга, Помещение,
ВидЗанятий, ВремяНачала, ВремяОкончания");
	СтруктураДанных.Вставить("ДатаВремяПроведения", Стр.ДатаЗанятия +
Час(Стр.ВремяНачала)*3600 + Минута(Стр.ВремяНачала)*60 + Секунда(Стр.ВремяНачала));
	СтруктураДанных.Вставить("Группа", Стр.Группа);
	СтруктураДанных.Вставить("Услуга", Стр.Услуга);
	СтруктураДанных.Вставить("Помещение", Стр.Помещение);
	СтруктураДанных.Вставить("ВидЗанятий", Стр.ВидЗанятий);
	СтруктураДанных.Вставить("ВремяНачала", Стр.ВремяНачала);
	СтруктураДанных.Вставить("ВремяОкончания", Стр.ВремяОкончания);
	СтруктураДанных.Вставить("Занятие", Стр.Занятие);


	ЗанятиеИзменилось(СтруктураДанных);
```

```
	Если СтруктураДанных <> Неопределено Тогда
		ТренерАдминистратор = ПолучитьРеквизитОбъекта(Стр.Группа, "ТренерАдминистратор");
		УправлениеОтображениемКнопокПосетителиЗанятия(ТренерАдминистратор);
	Иначе
		УправлениеОтображениемКнопокПосетителиЗанятия(Ложь);
	КонецЕсли;

КонецПроцедуры

&НаСервереБезКонтекста
Функция ПолучитьРеквизитОбъекта(ОбъектДляПолучения, Реквизит)
	Возврат ОбщегоНазначения.ЗначениеРеквизитаОбъекта(ОбъектДляПолучения, Реквизит);
КонецФункции

&НаСервере
Процедура ПриОткрытииНаСервере()

	ЗаполнитьРасписание();

КонецПроцедуры

&НаСервере
Процедура ЗаполнитьРасписание()

	Перем Выборка, Запрос, НачалоДня, Результат, Стр, ТабРасписаниеЗанятий, ТекущаяДата, Тренер;

	Тренер = Объект.Тренер;
	Если НЕ ЗначениеЗаполнено(Тренер) Тогда
		//Поиск тренера
		Запрос = Новый Запрос;
		Запрос.Текст = "ВЫБРАТЬ ПЕРВЫЕ 1
		|	Тренеры.Ссылка
		|ИЗ
		|	Справочник.Тренеры КАК Тренеры
		|ГДЕ
		|	Тренеры.Пользователь = &Пользователь
		|	И НЕ Тренеры.ПометкаУдаления";
		Запрос.УстановитьПараметр("Пользователь",
Пользователи.АвторизованныйПользователь());
		Результат = Запрос.Выполнить();

		Выборка = Результат.Выбрать();
		Выборка.Следующий();

		Объект.Тренер = Выборка.Ссылка;
		Тренер = Объект.Тренер;
	КонецЕсли;

	ТекущаяДата = ТекущаяДата();
	НачалоДня = НачалоДня(ТекущаяДата);

	//ЗаполнениеТаблицы Занятий
	Запрос = Новый Запрос;
	Запрос.Текст =
	"ВЫБРАТЬ РАЗЛИЧНЫЕ
	|	РасписаниеЗанятий.Регистратор КАК Занятие,
	|	РасписаниеЗанятий.ВидЗанятия КАК ВидЗанятий,
	|	РасписаниеЗанятий.ДатаНачала КАК ДатаЗанятия,
	|	РасписаниеЗанятий.ВремяНачала КАК ВремяНачала,
	|	РасписаниеЗанятий.ВремяОкончания,
	|	РасписаниеЗанятий.Помещение,
```

```
	|	РасписаниеЗанятий.Услуга,
	|	ТренерыГруппСрезПоследних.Группа КАК Группа
	|ИЗ
	|	РегистрСведений.ТренерыГрупп.СрезПоследних(&ДатаОкончания, ) КАК
ТренерыГруппСрезПоследних
	|		ЛЕВОЕ СОЕДИНЕНИЕ РегистрСведений.РасписаниеЗанятий КАК
РасписаниеЗанятий
	|		ПО ТренерыГруппСрезПоследних.Группа = РасписаниеЗанятий.Группа
	|ГДЕ
	|	РасписаниеЗанятий.ДатаНачала >= &ДатаНачала
	|	И РасписаниеЗанятий.ДатаОкончания <= &ДатаОкончания
	|	И РасписаниеЗанятий.Регистратор ССЫЛКА Документ.ПланированиеЗанятий
	|	И ТренерыГруппСрезПоследних.Тренер = &Тренер
	|
	|УПОРЯДОЧИТЬ ПО
	|	ВремяНачала";
	Запрос.УстановитьПараметр("ДатаНачала", ДатаАРМа);
	Запрос.УстановитьПараметр("ДатаОкончания", КонецДня(ДатаАРМа));
	Запрос.УстановитьПараметр("Тренер", Тренер);
	Результат = Запрос.Выполнить();

	ТабРасписаниеЗанятий = Объект.РасписаниеЗанятий;
	ТабРасписаниеЗанятий.Очистить();

	Если НЕ Результат.Пустой() Тогда
		Выборка = Результат.Выбрать();

		Пока Выборка.Следующий() Цикл
			Стр = ТабРасписаниеЗанятий.Добавить();
			ЗаполнитьЗначенияСвойств(Стр, Выборка);
		КонецЦикла;
	КонецЕсли;

КонецПроцедуры

&НаСервере
Функция ПолучитьПродление()
	Возврат Перечисления.ВидыОперацийИзмененийАбонементов.Продление
КонецФункции

&НаКлиенте
Процедура РасписаниеЗанятийВыбор(Элемент, ВыбраннаяСтрока, Поле, СтандартнаяОбработка)
	Стр = Элемент.ТекущиеДанные;
	СтруктураДанных = Неопределено;

	Если Стр <> Неопределено Тогда
		ЗаполнитьРеквизитыЗанятия(Стр);
	КонецЕсли;
КонецПроцедуры

&НаКлиенте
Процедура УправлениеОтображениемКнопокПосетителиЗанятия(Значение)
	Элементы.ПосетителиЗанятияПриостановитьАбонемент.Видимость = Значение;
	Элементы.ПосетителиЗанятияВозобновитьДействиеАбонемента.Видимость = Значение;
	Элементы.ПосетителиЗанятияЗакрытьАбонемент.Видимость = Значение;
	Элементы.ПосетителиЗанятияПродлитьАбонемент.Видимость = Значение;
	Элементы.ПосетителиЗанятияПередатьАбонемент.Видимость = Значение;
	Элементы.ПосетителиЗанятияЗаблокироватьАбонемент.Видимость = Значение;
	Элементы.ПосетителиЗанятияРазблокироватьАбонемент.Видимость = Значение;
КонецПроцедуры

&НаКлиенте
```

```
Процедура СоздатьДокументИзменениеАбонемента(ВидОперации)
	Стр = Элементы.ПосетителиЗанятия.ТекущиеДанные;
	Если Стр <> Неопределено Тогда
		Абонемент = Стр.Абонемент;

		СтруктураДанных = ПолучитьДанныеДляЗаполненияДокументаИзменениеАбонемента();
		СтруктураДанных.Вставить("ВидОперации", ВидОперации);
		СтруктураДанных.Вставить("Абонемент", Абонемент);

		ОпОповещения = Новый ОписаниеОповещения("ОбновитьАбонементыНаФорме",
ЭтаФорма, Неопределено);

		ОткрытьФорму("Документ.ИзменениеАбонемента.Форма.ФормаДокумента",
СтруктураДанных, ЭтаФорма,,,, ОпОповещения);
	Иначе
		Сообщить("Выберите абонемент.");
	КонецЕсли;
КонецПроцедуры

&НаСервереБезКонтекста
Функция ПолучитьДанныеДляЗаполненияДокументаИзменениеАбонемента()
	ТекПользователь = ПараметрыСеанса.ТекущийПользователь;

	Запрос = Новый Запрос;
	Запрос.Текст =
		"ВЫБРАТЬ
		|	НастройкиПользователей.Значение КАК Ответственный,
		|	НастройкиПользователей1.Значение КАК Организация
		|ИЗ
		|	РегистрСведений.НастройкиПользователей КАК НастройкиПользователей
		|		ПОЛНОЕ СОЕДИНЕНИЕ РегистрСведений.НастройкиПользователей КАК
НастройкиПользователей1
		|		ПО НастройкиПользователей.Пользователь =
НастройкиПользователей1.Пользователь
		|ГДЕ
		|	НастройкиПользователей.Пользователь = &Пользователь
		|	И НастройкиПользователей.Настройка = &НастройкаОтветственный
		|	И НастройкиПользователей1.Настройка = &НастройкаОрганизация";
	Запрос.УстановитьПараметр("НастройкаОтветственный",
ПланыВидовХарактеристик.НастройкиПользователей.ОсновнойОтветственный);
	Запрос.УстановитьПараметр("НастройкаОрганизация",
ПланыВидовХарактеристик.НастройкиПользователей.ОсновнаяОрганизация);
	Запрос.УстановитьПараметр("Пользователь", ТекПользователь);
	Результат = Запрос.Выполнить();

	Если НЕ Результат.Пустой() Тогда
		Выборка = Результат.Выбрать();
		Выборка.Следующий();

		Сотрудник = Выборка.Ответственный;
		Организация = Выборка.Организация;
	Иначе
		Сотрудник = Справочники.Сотрудники.ПустаяСсылка();
		Организация = Справочники.Организации.ПустаяСсылка();
	КонецЕсли;

	СтруктураДанных = Новый Структура;
	СтруктураДанных.Вставить("Автор", ТекПользователь);
	СтруктураДанных.Вставить("Сотрудник", Сотрудник);
	СтруктураДанных.Вставить("Организация", Организация);

	Возврат СтруктураДанных;
```

```
КонецФункции

&НаКлиенте
Процедура ПриостановитьАбонемент(Команда)
	ВидОперации =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Приостановление");
	СоздатьДокументИзменениеАбонемента(ВидОперации);
КонецПроцедуры

&НаКлиенте
Процедура ВозобновитьДействиеАбонемента(Команда)
	ВидОперации =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Возобновление");

	СоздатьДокументИзменениеАбонемента(ВидОперации);
КонецПроцедуры

&НаКлиенте
Процедура ЗакрытьАбонемент(Команда)
	ВидОперации =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Закрытие");
	СоздатьДокументИзменениеАбонемента(ВидОперации);
КонецПроцедуры

&НаКлиенте
Процедура ПродлитьАбонемент(Команда)
	ВидОперации =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Продление");
	СоздатьДокументИзменениеАбонемента(ВидОперации);
КонецПроцедуры

&НаКлиенте
Процедура ПередатьАбонемент(Команда)
	ВидОперации =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Передача");
	СоздатьДокументИзменениеАбонемента(ВидОперации);
КонецПроцедуры

&НаКлиенте
Процедура ЗаблокироватьАбонемент(Команда)
	ВидОперации =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Блокировка");
	СоздатьДокументИзменениеАбонемента(ВидОперации);
КонецПроцедуры

&НаКлиенте
Процедура РазблокироватьАбонемент(Команда)
	ВидОперации =
ПредопределенноеЗначение("Перечисление.ВидыОперацийИзмененийАбонементов.Разблокировка");

	СоздатьДокументИзменениеАбонемента(ВидОперации);
КонецПроцедуры

&НаКлиенте
Процедура ОбновитьАбонементыНаФорме(Результат, Параметры) Экспорт
	Если Результат <> Неопределено Тогда
		Если ТипЗнч(Результат) = Тип("СправочникСсылка.Спортсмены") Тогда
			ЗаполнениеПосетителей();
		КонецЕсли;
	КонецЕсли;
КонецПроцедуры
```

```
&НаКлиенте
Процедура ПосетителиЗанятияПриАктивизацииСтроки(Элемент)
	Если Элемент.ТекущиеДанные <> Неопределено Тогда
		Абонемент = Элемент.ТекущиеДанные.Абонемент;
		Если ЗначениеЗаполнено(Абонемент) тогда
			СтруктураДанных = Новый Структура;

			ТренерАдминистратор = ПолучитьРеквизитОбъекта(Объект.Группа,
"ТренерАдминистратор");

			Если НЕ ТренерАдминистратор ИЛИ НЕ ЗначениеЗаполнено(Абонемент) Тогда
				СтруктураДанных.Вставить("КнопкаПриостановить",	Ложь);
				СтруктураДанных.Вставить("КнопкаВозобнавить",		Ложь);
				СтруктураДанных.Вставить("КнопкаЗакрыть",		Ложь);
				СтруктураДанных.Вставить("КнопкаПродлить",		Ложь);
				СтруктураДанных.Вставить("КнопкаПередать",		Ложь);
				СтруктураДанных.Вставить("КнопкаЗаблокировать",	Ложь);
				СтруктураДанных.Вставить("КнопкаРазблокировать",	Ложь);

				УправлениеОтображениемКнопокПанели(СтруктураДанных);

				Возврат;
			КонецЕсли;

			СостояниеАбонемента =
ПолучитьСостояниеАбонемента(Элемент.ТекущиеДанные.Абонемент);

			Если СостояниеАбонемента =
ПредопределенноеЗначение("Перечисление.СостоянияАбонементов.Активен") Тогда
				СтруктураДанных.Вставить("КнопкаПриостановить",	Истина);
				СтруктураДанных.Вставить("КнопкаВозобнавить",		Ложь);
				СтруктураДанных.Вставить("КнопкаЗакрыть",		Истина);
				СтруктураДанных.Вставить("КнопкаПродлить",		Истина);
				СтруктураДанных.Вставить("КнопкаПередать",		Истина);
				СтруктураДанных.Вставить("КнопкаЗаблокировать",	Истина);
				СтруктураДанных.Вставить("КнопкаРазблокировать",	Ложь);
			ИначеЕсли СостояниеАбонемента =
ПредопределенноеЗначение("Перечисление.СостоянияАбонементов.Приостановлен") Тогда
				СтруктураДанных.Вставить("КнопкаПриостановить",	Ложь);
				СтруктураДанных.Вставить("КнопкаВозобнавить",		Истина);
				СтруктураДанных.Вставить("КнопкаЗакрыть",		Истина);
				СтруктураДанных.Вставить("КнопкаПродлить",		Ложь);
				СтруктураДанных.Вставить("КнопкаПередать",		Истина);
				СтруктураДанных.Вставить("КнопкаЗаблокировать",	Истина);
				СтруктураДанных.Вставить("КнопкаРазблокировать",	Ложь);
			ИначеЕсли СостояниеАбонемента =
ПредопределенноеЗначение("Перечисление.СостоянияАбонементов.Истек") Тогда
				СтруктураДанных.Вставить("КнопкаПриостановить",	Ложь);
				СтруктураДанных.Вставить("КнопкаВозобнавить",		Ложь);
				СтруктураДанных.Вставить("КнопкаЗакрыть",		Ложь);
				СтруктураДанных.Вставить("КнопкаПродлить",		Истина);
				СтруктураДанных.Вставить("КнопкаПередать",		Истина);
				СтруктураДанных.Вставить("КнопкаЗаблокировать",	Ложь);
				СтруктураДанных.Вставить("КнопкаРазблокировать",	Ложь);
			ИначеЕсли СостояниеАбонемента =
ПредопределенноеЗначение("Перечисление.СостоянияАбонементов.Заблокирован") Тогда
				СтруктураДанных.Вставить("КнопкаПриостановить",	Ложь);
				СтруктураДанных.Вставить("КнопкаВозобнавить",		Ложь);
				СтруктураДанных.Вставить("КнопкаЗакрыть",		Истина);
				СтруктураДанных.Вставить("КнопкаПродлить",		Ложь);
				СтруктураДанных.Вставить("КнопкаПередать",		Истина);
				СтруктураДанных.Вставить("КнопкаЗаблокировать",	Ложь);
```

```
                        СтруктураДанных.Вставить("КнопкаРазблокировать",    Истина);
                    Иначе
                        Возврат;
                    КонецЕсли;

                    УправлениеОтображениемКнопокПанели(СтруктураДанных);
                КонецЕсли;
        КонецЕсли;
КонецПроцедуры

&НаКлиенте
Процедура УправлениеОтображениемКнопокПанели(СтруктураДанных)
        Элементы.ПосетителиЗанятияПриостановитьАбонемент.Видимость                  =
СтруктураДанных.КнопкаПриостановить;
        Элементы.ПосетителиЗанятияВозобновитьДействиеАбонемента.Видимость       =
СтруктураДанных.КнопкаВозобнавить;
        Элементы.ПосетителиЗанятияЗакрытьАбонемент.Видимость                        =
СтруктураДанных.КнопкаЗакрыть;
        Элементы.ПосетителиЗанятияПродлитьАбонемент.Видимость                       =
СтруктураДанных.КнопкаПродлить;
        Элементы.ПосетителиЗанятияПередатьАбонемент.Видимость                       =
СтруктураДанных.КнопкаПередать;
        Элементы.ПосетителиЗанятияЗаблокироватьАбонемент.Видимость                  =
СтруктураДанных.КнопкаЗаблокировать;
        Элементы.ПосетителиЗанятияРазблокироватьАбонемент.Видимость             =
СтруктураДанных.КнопкаРазблокировать;
КонецПроцедуры

&НаСервереБезКонтекста
Функция ПолучитьСостояниеАбонемента(Абонемент)
        Если ЗначениеЗаполнено(Абонемент) Тогда
            Запрос = Новый Запрос;
            Запрос.Текст =
                "ВЫБРАТЬ
                |    СостоянияАбонементовСрезПоследних.СостояниеАбонемента
                |ИЗ
                |    РегистрСведений.СостоянияАбонементов.СрезПоследних(&Дата,
Абонемент = &Абонемент) КАК СостоянияАбонементовСрезПоследних";
            Запрос.УстановитьПараметр("Абонемент", Абонемент);
            Запрос.УстановитьПараметр("Дата", ТекущаяДата());
            Результат = Запрос.Выполнить();

            Если НЕ Результат.Пустой() Тогда
                Выборка = Результат.Выбрать();
                Выборка.Следующий();

                Возврат Выборка.СостояниеАбонемента;
            Иначе
                Возврат Перечисления.СостоянияАбонементов.ПустаяСсылка();
            КонецЕсли;
        КонецЕсли;
КонецФункции

&НаКлиенте
Процедура ОбновитьАРМ(Команда)
        ПерезаполнитьАРМТренера();
КонецПроцедуры

&НаКлиенте
Процедура ПосетителиЗанятияСтатусЗанятияПриИзменении(Элемент)
        ПриИзмененииРеквизитаПосетителиЗанятия(Элемент.Имя,
Элементы.ПосетителиЗанятия.ТекущаяСтрока);
```

```
КонецПроцедуры

&НаСервере
Функция ПриИзмененииРеквизитаПосетителиЗанятия(ИмяЭлемента, НомерСтроки)
	СтрокаПосетителиЗанятия = Объект.ПосетителиЗанятия[НомерСтроки];
	//Очистка реквизитов, которые должны быть очищены

	Если ИмяЭлемента = "ПосетителиЗанятияСтатусЗанятия" ИЛИ ИмяЭлемента =
"ПосетителиЗанятияСпортсмен" ИЛИ ИмяЭлемента = "ПосетителиЗанятияАбонемент" Тогда
		СтрокаПосетителиЗанятия.Отметка = Ложь;
	КонецЕсли;

	Если ИмяЭлемента = "ПосетителиЗанятияСтатусЗанятия" ИЛИ ИмяЭлемента =
"ПосетителиЗанятияСпортсмен" Тогда
		СтрокаПосетителиЗанятия.Абонемент =
ПредопределенноеЗначение("Справочник.АбонементыСпортсменов.ПустаяСсылка");
		СтрокаПосетителиЗанятия.ЗанятийОсталось = 0;
		СтрокаПосетителиЗанятия.СрокДействияАбонемента = Дата(1,1,1);
		СтрокаПосетителиЗанятия.СостояниеАбонемента =
ПредопределенноеЗначение("Перечисление.СостоянияАбонементов.ПустаяСсылка");
		СтрокаПосетителиЗанятия.По = Дата(1,1,1);
	КонецЕсли;

	Если ИмяЭлемента = "ПосетителиЗанятияСтатусЗанятия" Тогда
		СтрокаПосетителиЗанятия.Спортсмен =
ПредопределенноеЗначение("Справочник.Спортсмены.ПустаяСсылка");
		СтрокаПосетителиЗанятия.ДатаРождения = Дата(1,1,1);
	КонецЕсли;

	//Установка новых значений реквизитов
	Если ИмяЭлемента = "ПосетителиЗанятияСпортсмен" Тогда
		СтрокаПосетителиЗанятия.ДатаРождения =
СтрокаПосетителиЗанятия.Спортсмен.ФизЛицо.ДатаРождения;
	КонецЕсли;

	Если ИмяЭлемента = "ПосетителиЗанятияАбонемент" Тогда
		НовыеЗначенияРеквизитов =
УправлениеСпортклубом.ПолучитьДанныеПоАбонементу(СтрокаПосетителиЗанятия.Абонемент);
		ЗаполнитьЗначенияСвойств(СтрокаПосетителиЗанятия, НовыеЗначенияРеквизитов);
	КонецЕсли;

	СтрокаПосетителиЗанятия.ДоступностьДляРедактирования = Истина; //тут нужна процедура
определения доступности

КонецФункции

&НаКлиенте
Процедура ПосетителиЗанятияСпортсменПриИзменении(Элемент)
	ПриИзмененииРеквизитаПосетителиЗанятия(Элемент.Имя,
Элементы.ПосетителиЗанятия.ТекущаяСтрока);
КонецПроцедуры

&НаКлиенте
Процедура ПосетителиЗанятияАбонементПриИзменении(Элемент)
	ПриИзмененииРеквизитаПосетителиЗанятия(Элемент.Имя,
Элементы.ПосетителиЗанятия.ТекущаяСтрока);
КонецПроцедуры

&НаКлиенте
Процедура ПосетителиЗанятияПриИзменении(Элемент)
	// Вставить содержимое обработчика.
КонецПроцедуры
```

```
&НаКлиенте
Процедура БланкЗанятия(Команда)
	ТабДок = БланкЗанятияСервер();
	ТабДок.Показать();
КонецПроцедуры

&НаСервере
Функция БланкЗанятияСервер()
	ТабДок = Новый ТабличныйДокумент();
	Макет =обработки.АРМТренера.ПолучитьМакет("БланкЗанятия");
	ОбластьШапка = Макет.ПолучитьОбласть("Шапка");
	ОбластьСтрока = Макет.ПолучитьОбласть("Строка");
	ПараметрыШапка = Новый Структура;
	ПараметрыШапка.Вставить("Тренер", объект.Тренер.Наименование);
	ПараметрыШапка.Вставить("Дата", объект.ДатаВремяПроведения);
	ПараметрыШапка.Вставить("ВремяЗанятия", объект.ВремяНачала);
	ПараметрыШапка.Вставить("Подразделение", "");
	ПараметрыШапка.Вставить("Помещение", Объект.Помещение);
	ПараметрыШапка.Вставить("Группа", объект.Группа);
	ОбластьШапка.Параметры.Заполнить(ПараметрыШапка);
	ТабДок.Вывести(ОбластьШапка);

	Для Каждого ТабСтр из Объект.ПосетителиЗанятия Цикл
		ОбластьСтрока.Параметры.Заполнить(ТабСтр);
		ТабДок.Присоединить(ОбластьСтрока);
	КонецЦикла;
	Возврат ТабДок;

КонецФункции
```