TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа Инженерная школа информационных технологий и робототехники
Направление подготовки 09.04.01 Информатика и вычислительная техника
Отделение школы (НОЦ) Отделение информационных технологий

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Тема работы
Система согласованного планирования отпусков сотрудников подразделений университета

УДК 004.774:004.65:004.451:378.662

Студент

Группа	ФИО	Подпись	Дата
8ВМ02	Шкулов Никита Петрович		

Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР	Фадеев А.С.	к.т.н		

КОНСУЛЬТАНТЫ ПО РАЗДЕЛАМ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Профессор ОСГН ШБИП	Жиронкин С.А.	д.э.н.		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Профессор ООД ШБИП	Федоренко О.Ю.	д.м.н.		

ДОПУСТИТЬ К ЗАЩИТЕ:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР	Кочегурова Е.А.	к.т.н.		

Томск – 2022 г.

ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ

по направлению 09.04.01 Информатика и вычислительная техника

Код компетенции	Наименование компетенции
Универсальные компетенции	
УК(У)-1	Способен осуществлять критический анализ проблемных ситуаций на основе системного подхода, вырабатывать стратегию действий
УК(У)-2	Способен управлять проектом на всех этапах его жизненного цикла
УК(У)-3	Способен организовывать и руководить работой команды, вырабатывая командную стратегию для достижения поставленной цели
УК(У)-4	Способен применять современные коммуникативные технологии, в том числе на иностранном (-ых) языке (-ах), для академического и профессионального взаимодействия
УК(У)-5	Способен анализировать и учитывать разнообразие культур в процессе межкультурного взаимодействия
УК(У)-6	Способен определять и реализовывать приоритеты собственной деятельности и способы ее совершенствования на основе самооценки
Общепрофессиональные компетенции	
ОПК(У)-1	Способен самостоятельно приобретать, развивать и применять математические, естественно-научные, социально-экономические и профессиональные знания для решения нестандартных задач, в том числе в новой или незнакомой среде и в междисциплинарном контексте
ОПК(У)-2	Способен разрабатывать оригинальные алгоритмы и программные средства, в том числе с использованием современных интеллектуальных технологий, для решения профессиональных задач
ОПК(У)-3	Способен анализировать профессиональную информацию, выделять в ней главное, структурировать, оформлять и представлять в виде аналитических обзоров с обоснованными выводами и рекомендациями
ОПК(У)-4	Способен применять на практике новые научные принципы и методы исследований
ОПК(У)-5	Способен разрабатывать и модернизировать программное и аппаратное обеспечение информационных и автоматизированных систем
ОПК(У)-6	Способен разрабатывать компоненты программно-аппаратных комплексов обработки информации и автоматизированного проектирования

ОПК(У)-7	Способен адаптировать зарубежные комплексы обработки информации и автоматизированного проектирования к нуждам отечественных предприятий
ОПК(У)-8	Способен осуществлять эффективное управление разработкой программных средств и проектов
Профессиональные компетенции	
ПК(У)-1	Способен разрабатывать и администрировать системы управления базами данных
ПК(У)-2	Способен проектировать сложные пользовательские интерфейсы
ПК(У)-3	Способен управлять процессами и проектами по созданию (модификации) информационных ресурсов
ПК(У)-4	Способен осуществлять руководство разработкой комплексных проектов на всех стадиях и этапах выполнения работ
ПК(У)-5	Способен проектировать и организовывать учебный процесс по образовательным программам с использованием современных образовательных технологий

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа Инженерная школа информационных технологий и робототехники
Направление подготовки 09.04.01 Информатика и вычислительная техника
Отделение школы (НОЦ) Отделение информационных технологий

УТВЕРЖДАЮ:
Руководитель ООП
_____ _____ Кочегурова Е.А.
(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

Магистерской диссертации

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

Студенту:

Группа	ФИО
8ВМ02	Шкулову Никите Петровичу

Тема работы:

Система согласованного планирования отпусков сотрудников подразделений университета	
Утверждена приказом директора (дата, номер)	№ 34-64/с от 03.02.2022

Срок сдачи студентом выполненной работы:	01.06.2022

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе *(наименование объекта исследования или проектирования; производительность или нагрузка; режим работы (непрерывный, периодический, циклический и т. д.); вид сырья или материал изделия; требования к продукту, изделию или процессу; особые требования к особенностям функционирования (эксплуатации) объекта или изделия в плане безопасности эксплуатации, влияния на окружающую среду, энергозатратам; экономический анализ и т. д.).*	Объектом проектирования и разработки является веб-приложение, обеспечивающее возможности по управлению и согласованию отпусков сотрудников руководителями подразделений университета.
Перечень подлежащих исследованию, проектированию и разработке вопросов *(аналитический обзор по литературным источникам с целью выяснения достижений мировой науки техники в рассматриваемой области; постановка задачи исследования, проектирования, конструирования; содержание процедуры исследования, проектирования, конструирования; обсуждение результатов выполненной работы; наименование дополнительных разделов, подлежащих разработке; заключение по работе).*	1. Исследование предметной области планирования и назначения отпусков; 2. Анализ аналогов разрабатываемого веб-приложения; 3. Проектирование приложения; 4. Проектирование методов и форматов сообщений API, подготовка данных для работы приложения; 5. Разработка и тестирование веб-приложения;

	6. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение; 7. Социальная ответственность.
Перечень графического материала *(с точным указанием обязательных чертежей)*	1. Диаграмма вариантов использования; 2. Диаграмма базы данных; 3. Карта веб-приложения; 4. Скриншоты интерфейса веб-приложения; 5. Диаграмма Ганта.

Консультанты по разделам выпускной квалификационной работы
(с указанием разделов)

Раздел	Консультант
Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	Профессор ОСГН ШБИП, д.э.н., Жиронкин С.А.
Социальная ответственность	Профессор ООД ШБИП, д.м.н., Федоренко О.Ю.
Английский язык	Доцент ОИЯ ШБИП, к.ф.н., Диденко А.В.

Названия разделов, которые должны быть написаны на иностранном языке:

Раздел 2 Analysis of technologies. Software requirements and design

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	01.03.2022

Задание выдал руководитель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР	Фадеев А.С.	к.т.н		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ВМ02	Шкулов Никита Петрович		

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа Инженерная школа информационных технологий и робототехники
Направление подготовки 09.04.01 Информатика и вычислительная техника
Уровень образования Магистратура
Отделение школы (НОЦ) Отделение информационных технологий
Период выполнения (осенний / весенний семестр 2021 /2022 учебного года)

Форма представления работы:

Магистерская диссертация

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

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Срок сдачи студентом выполненной работы:	01.06.2022

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
01.06.2022	Анализ предметной области и сбор требований	20
01.06.2022	Проектирование разрабатываемой программы	20
01.06.2022	Реализация программного обеспечения	30
01.06.2022	Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	10
01.06.2022	Социальная ответственность	10
01.06.2022	Приложение на английском языке	10

СОСТАВИЛ:
Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР	Фадеев А.С.	к.т.н.		

СОГЛАСОВАНО:
Руководитель ООП

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ ИШИТР	Кочегурова Е.А.	к.т.н.		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студенту:

Группа	ФИО
8ВМ02	Шкулов Никита Петрович

Школа	ИШИТР	Отделение школы (НОЦ)	Отделение информационных технологий
Уровень образования	Магистратура	**Направление/специальность**	09.04.01 Информатика и вычислительная техника

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
1. *Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих* 2. *Нормы и нормативы расходования ресурсов* 3. *Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования*	*Материальные затраты: 4 200 рублей;* *Оклад руководителя: 26 300 рублей;* *Оклад разработчика: 3100 рублей;* *Районный коэффициент: 1,3;* *Размер отчислений во внебюджетные фонды: 30%.*
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. *Оценка коммерческого и инновационного потенциала НТИ*	*Выполнение предпроектного анализа, поиска потенциальных потребителей, анализа конкурентных технических решений, оценка степени готовности проекта к коммерциализации.*
2. *Разработка устава научно-технического проекта*	*Определение целей и результатов проекта, организационной структуры, ограничений и допущений.*
3. *Планирование процесса управления НТИ: структура и график проведения, бюджет, риски и организация закупок*	*Детализация иерархической структуры работ, создание календарного плана проекта, расчет бюджета научного исследования, выбор организационной структуры проекта, создание плана управления коммуникациями, анализ рисков.*
4. *Определение ресурсной, финансовой, экономической эффективности*	*Расчет интегрального показателя ресурсоэффективности, интегрального финансового показателя, интегрального показателя эффективности, расчет сравнительной эффективности вариантов исполнения.*
Перечень графического материала *(с точным указанием обязательных чертежей):*	
1. *Сегментирование рынка* 2. *Оценка конкурентоспособности технических решений* 3. *Матрица SWOT* 4. *График проведения и бюджет НТИ* 5. *Оценка ресурсной, финансовой и экономической эффективности НТИ*	

Дата выдачи задания для раздела по линейному графику	01.03.2022

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Профессор	Жиронкин С.А.	д-р экон. наук		28.02.2022

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ВМ02	Шкулов Никита Петрович		28.02.2022

ЗАДАНИЕ ДЛЯ РАЗДЕЛА «СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студенту:

Группа		ФИО	
8ВМ02		Шкулов Никита Петрович	
Школа	**ИШИТР**	**Отделение (НОЦ)**	**ОИТ**
Уровень образования	магистратура	**Направление/ специальность**	09.04.01 Информатика и вычислительная техника

Тема ВКР:

Система согласованного планирования отпусков сотрудников подразделений университета	
Исходные данные к разделу «Социальная ответственность»:	
Введение – Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика) и области его применения. – Описание рабочей зоны (рабочего места) при разработке проектного решения/при эксплуатации	*Объект исследования:* информационная система для планирования отпусков в подразделениях университета *Область применения:* управление отпусками *Рабочая зона:* офис. Используется центральное водяное отопление с радиаторами температурой около 70 °С; вентиляция естественная, обеспечивается посредством вентиляционного канала здания и проветриванием рабочего помещения; освещение совмещенное: естественное освещение боковое, искусственное освещение является общим равномерным, источники света – лампы накаливания. *Размеры помещения:* 6*4 м *Количество и наименование оборудования рабочей зоны:* компьютер, 1 ед. *Рабочие процессы, связанные с объектом исследования, осуществляющиеся в рабочей зоне:* выбор отпусков сотрудниками университета, утверждение отпусков руководителями подразделений.
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Правовые и организационные вопросы обеспечения безопасности <u>при разработке проектного решения:</u> – специальные (характерные при эксплуатации объекта исследования, проектируемой рабочей зоны) правовые нормы трудового законодательства; – организационные мероприятия при компоновке рабочей зоны.	Трудовой кодекс Российской Федерации от 30.12.2001 N 197-ФЗ (ред. от 09.03.2021); ГОСТ 12.2.032-78 ССБТ. Рабочее место при выполнении работ сидя. Общие эргономические требования; ГОСТ 22269-76 Система «человек-машина». Рабочее место оператора. Взаимное расположение элементов рабочего стола. Общие эргономические требования; ГОСТ Р ИСО 9241-2-2009 Эргономические требования к проведению офисных работ с использованием видеодисплейных терминалов (VDT). Часть 2. Требования к производственному заданию; ГОСТ Р ИСО 9241-5-2009 Эргономические требования к проведению офисных работ с использованием видеодисплейных терминалов (VDT). Часть 5. Требования к расположению рабочей станции и осанке оператора; ГОСТ 12.0.003-2015 Опасные и вредные производственные факторы; СП 52.133330.2016 Естественное и искусственное освещение. Актуализированная редакция СНиП 23-05-95; СанПиН 1.2.3685-21 Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания; МР 2.2.9.2311 – 07 «Профилактика стрессового состояния работников при различных видах профессиональной деятельности»;

	ГОСТ 12.1.003-2014 ССБТ. Шум. Общие требования безопасности; ГОСТ 12.1.045-84 ССБТ. Электростатические поля. Допустимые уровни на рабочих местах и требования к проведению контроля; ГОСТ 12.1.019-2017 ССБТ. Электробезопасность. Общие требования и номенклатура видов защиты; ГОСТ 12.1.004-91 ССБТ. Пожарная безопасность. Общие требования; ГОСТ 12.1.030-81 ССБТ. Электробезопасность. Защитное заземление. Зануление; ГОСТ 12.1.038-82 ССБТ. Электробезопасность. Предельно допустимые значения напряжений прикосновения и токов.
2. Производственная безопасность при разработке проектного решения: – Анализ выявленных вредных и опасных производственных факторов – Расчет уровня опасного или вредного производственного фактора	**Вредные факторы:** 1. Повышенный уровень шума; 2. Отсутствие или недостаток необходимого естественного освещения; 3. Отсутствие или недостатки необходимого искусственного освещения; 4. Повышенная яркость света; 5. Умственное напряжение, в том числе вызванное информационной нагрузкой; 6. Эмоциональные перегрузки. **Опасные факторы:** Факторы, связанные с электрическим током, вызываемым разницей электрических потенциалов, под действие которого попадает работающий. **Требуемые средства коллективной и индивидуальной защиты от выявленных факторов**: беруши, наушники. **Фактор для расчета**: расчет системы искусственного освещения.
3. Экологическая безопасность при разработке проектного решения:	Воздействие на селитебную зону, гидросферу и атмосферу отсутствует. Воздействие на литосферу: утилизация компьютера и периферийных устройств (принтеры, МФУ, веб-камеры, наушники, колонки, телефоны), люминесцентных ламп, макулатуры.
4. Безопасность в чрезвычайных ситуациях при разработке проектного решения:	**Возможные ЧС**: Природные катастрофы (ураган, наводнения); Геологические воздействия (землетрясения); Техногенные аварии (ЧС с выбросом радиоактивных веществ на СХК в городе Северск, пожары). **Наиболее типичная ЧС**: пожар.
Дата выдачи задания для раздела по линейному графику	01.03.2022

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Профессор ООД ШБИП	Федоренко Ольга Юрьевна	д.м.н.		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
8ВМ02	Шкулов Никита Петрович		

Реферат

Выпускная квалификационная работа, 165 с., 62 рис., 31 табл., 41 источник, 4 прил.

Ключевые слова: отпуска, Трудовой кодекс, API, веб-приложение.

Объектом исследования являются отпуска и процесс их согласованного планирования.

Цель работы – проектирование и реализация веб-приложения, предоставляющего возможности по просмотру, согласованному планированию и управлению отпусками сотрудников подразделений ТПУ.

В процессе исследования были произведены анализ предметной области, сбор требований и анализ аналогов разрабатываемого веб-приложения.

В результате исследования были выявлены слабые и сильные стороны аналогов, а также ожидания пользователей от разрабатываемой системы; были выполнены проектирование и реализация веб-приложения.

Область применения: планирование отпусков в подразделениях университета.

Степень внедрения: работа может быть потенциально введена в использование университетом.

Экономическая эффективность/значимость работы – применение подобной системы позволяет облегчить работу руководителя подразделения или его заместителей при планировании отпусков за счет выполнения всех необходимых проверок отпусков автоматически. Соответственно, это способно сократить время, затрачиваемое на процесс составления графика отпусков.

В будущем планируется работа по внедрению разработки в университет с подключением её к API и системе авторизации ТПУ, а также развертыванию веб-приложения на сервере университета.

Определения, обозначения, сокращения

1. API (Application Programming Interface) – программный интерфейс приложения, набор методов, доступный для обращения со стороны внешних программ.
2. JSON (JavaScript Object Notation) – текстовый формат обмена данными.
3. MS – Microsoft.
4. БД – база данных.
5. ИС – информационная система.
6. ОС – операционная система.
7. ПО – программное обеспечение.
8. РФ – Российская Федерация.
9. СУБД – система управления базами данных.
10. ТК РФ – Трудовой кодекс Российской Федерации.

Оглавление

Введение

Планирование отпусков является важной и сложной задачей, так как, с одной стороны, отпуска необходимы для отдыха сотрудников, но, с другой стороны, они должны быть синхронизированы таким образом, чтобы уход сотрудников в отпуск не навредил работе их подразделения.

Возможны различные способы выполнения такого планирования и распределения периодов отпусков сотрудников в течение года. Например, руководитель может формировать график на бумаге, с помощью календаря или посредством электронных таблиц. Однако все эти способы связаны с определенными сложностями: например, отпуска регламентируются Трудовым кодексом РФ, который необходимо соблюдать, при этом могут быть важны также особенности подразделения руководителя и всей организации в целом. Следовательно, ответственный за распределение отпусков человек обязан помнить одновременно о многих деталях и следить за их соблюдением при создании графика. Создание специализированный информационной системы для планирования отпусков способно решить эту проблему.

Соответственно, актуальность данной разработки подтверждается полезностью подобной программы для задачи согласования отпусков, в том числе и внутри Томского политехнического университета.

Объектом данного исследования являются отпуска и процесс их согласованного планирования.

Предмет исследования – это веб-приложение, реализующее возможности управления отпусками внутри своих подразделений для их руководителей и выбора желаемых периодов отпусков для сотрудников.

Цель работы – проектирование и реализация веб-приложения, предоставляющего возможности по просмотру, согласованному планированию и управлению отпусками сотрудников подразделений ТПУ.

Для достижения этой цели необходимо выполнить следующие задачи:

1. Изучить предметную область.

2. Произвести опрос потенциальных пользователей об ожидаемом функционале и требованиях к системе.
3. Выполнить проектирование системы.
4. Реализовать веб-приложение на языке программирования.

Помимо этого, необходимо также провести экономический анализ и анализ по социальной ответственности разрабатываемого проекта, что позволит оценить перспективы его применения на реальном предприятии.

1 Проектирование веб-приложения планирования отпусков

1.1 Требования к разрабатываемой системе

Перед началом непосредственной работы над проектом было необходимо изучить предметную область и расспросить заинтересованные стороны о том функционале, который они ожидают получить от готовой системы.

Предметная область для данного приложения представлена по большей части одним документом – Трудовым кодексом РФ, а именно его девятнадцатой главой, посвященной отпускам. Статьи данной главы содержат исчерпывающую информацию о видах отпусков, количестве дней, порядке их назначения, переноса, продления, отмены, о различных допущениях и запретах. Работа разрабатываемой системы должна обеспечивать выполнение норм, установленных в этом кодексе, поэтому многие из данных глав могут быть рассмотрены в качестве важных требований к веб-приложению.

Также важным было узнать мнение конкретных сотрудников и руководителей университета, которые будут пользоваться готовым веб-приложением: в то время, как соблюдение Трудового кодекса является минимальным требованием к работе системы, существуют различные возможности в плане функционала системы, которые облегчат работу с ней или сделают её более удобной. Таким образом был получен ряд дополнительных требований, касающихся функций веб-приложения, особенно относящихся к возможностям руководителей подразделений.

В результате изучения предметной области и проведения опроса потенциальных пользователей были разработаны следующие требования к системе согласованного планирования отпусков:

1. Предоставление графического интерфейса с календарем, отображающим периоды отпусков сотрудников и позволяющим задавать эти отпуска сотрудникам и руководителям.

2. Возможность подключения системы в будущем к API ТПУ с загрузкой данных о подразделениях, должностях, сотрудниках университета, их иерархии, а также производственного календаря.
3. Учет в системе различных видов сотрудников и их взаимной иерархии: сотрудники, являющиеся только подчиненными; сотрудники, руководящие одним или несколькими подразделениями; сотрудники, являющиеся подчиненными в одних подразделениях и руководителями в других.
4. Настройка руководителями правил выбора отпусков внутри своего подразделения: какие сотрудники могут или не могут уходить в отпуск одновременно; сколько человек (как минимум) какой должности обязано находиться одновременно на рабочем месте.
5. Получение уведомлений. Сотрудники получают уведомления при изменении отпусков либо их статусов руководителями. Уведомления руководителям отправляются при выборе подчиненными сотрудниками периодов отпусков, а также при их скором уходе или выходе из отпуска.
6. Возможность оставления заявки сотрудником на изменение его уже выбранного и утвержденного отпуска.
7. Создание групп из сотрудников, которым можно задавать имя, описание и правила выбора периодов отпуска.

Кроме того, ряд требований связан непосредственно с выполнением статей Трудового кодекса РФ [1]:

1. Каждый сотрудник имеет право на как минимум 28 дней основного оплачиваемого отпуска каждый год (статья 115).
2. Руководитель должен иметь возможность добавлять сотруднику дополнительные дни оплачиваемого отпуска (статья 116).
3. Сотрудникам с ненормированным рабочим днем предоставляется как минимум 3 дополнительных дня (статья 119).
4. Нерабочие праздничные дни не учитываются как дни отпуска (статья 120).

5. Возможность продления либо переноса уже утвержденного отпуска по различным причинам (статья 124).
6. При отмене либо прерывании отпуска в системе должно сохраниться, что у сотрудника есть неиспользованные дни отпуска, которые должны быть им потрачены в течение следующего года после назначения отмененного либо прерванного отпуска (статьи 124, 125).
7. Нельзя отменять оплачиваемый отпуск два года подряд (статья 124).
8. Возможность разделения отпуска на части, при этом как минимум одна часть должна длиться непрерывно не менее 14 дней (статья 125).
9. Возможность предоставления отпуска без сохранения заработной платы (статья 128).

1.2 Диаграмма вариантов использования

Диаграмма вариантов использования – это диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей [2]. Такая диаграмма позволяет выделить в общей форме на самых ранних этапах проектирования, какие именно основные функции должны будут реализованы на этапе разработки, отвечая таким образом на два вопроса: что будет делать приложение, и какие пользователи будут его использовать.

Для разрабатываемой системы было выделено четыре основные роли. Первая роль – это неавторизованный пользователь. Такой пользователь может только авторизоваться в системе по корпоративному логину и паролю ТПУ, как показано на рисунке 1.1. Соответственно, регистрация системой также не предусмотрена, так как она должна проводиться самим университетом для его сотрудников; система же работает только с уже существующими пользователями.

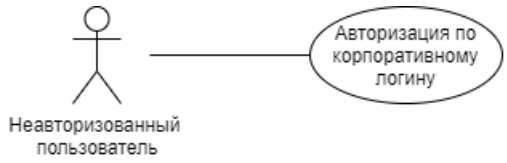


Рисунок 1.1 – Варианты использования для роли «Неавторизованный пользователь»

Вторая роль – это пользователь-администратор (рисунок 1.2). Это особый вид пользователей, предназначенный для управления самой системой, поэтому ему доступны возможности по просмотру списка сотрудников и подразделений, получаемых в конкретный момент времени из API ТПУ, а также информации о выбранном сотруднике или подразделении. Данная роль позволяет отслеживать данные, с которыми работает система, на самом высоком уровне; благодаря этому становится возможным также находить ошибки или нарушения в обработке системой данных из API.

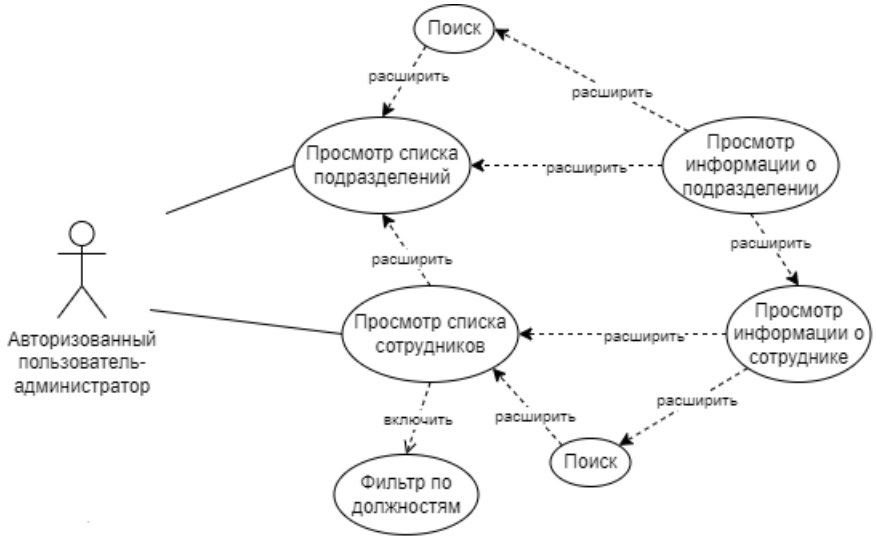


Рисунок 1.2 – Варианты использования для роли «Авторизованный пользователь-администратор»

Третья роль – это авторизованный пользователь-сотрудник некоторого подразделения университета (рисунок 1.3). Основной функционал такого пользователя сводится к возможности управления и просмотра своих отпусков: так, у пользователя есть специальная страница со списком назначенных и утвержденных отпусков, которые могут быть отфильтрованы по году и по типу. При наличии доступных отпускных дней пользователь может выбрать желаемый

период отпуска, а после его добавления – редактировать либо удалить. Если отпуск уже был утвержден, то пользователь может оставить запрос руководителю на изменение этого отпуска.

Помимо этого, пользователь имеет возможность получать уведомления, связанные с изменением статусов его отпусков, а также просматривать количество и источники своих отпускных дней.

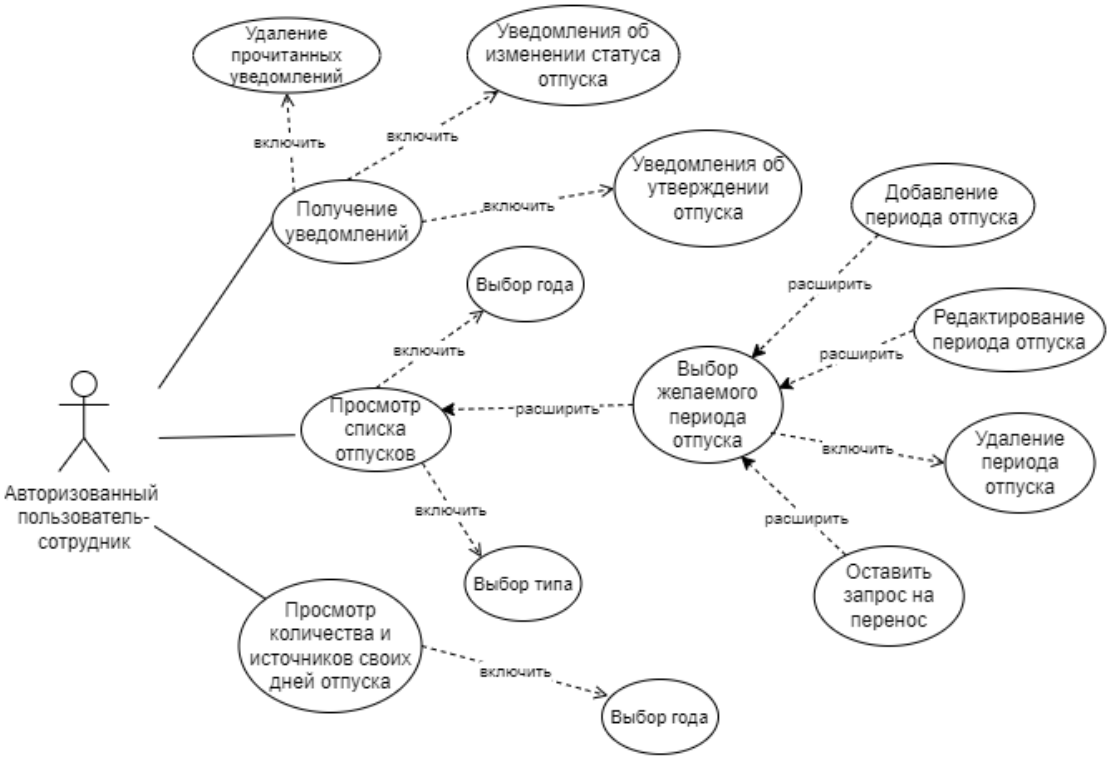


Рисунок 1.3 – Варианты использования для роли «Авторизованный пользователь-сотрудник»

Четвертая роль – это авторизованный пользователь, являющийся руководителем подразделения (рисунок 1.4).

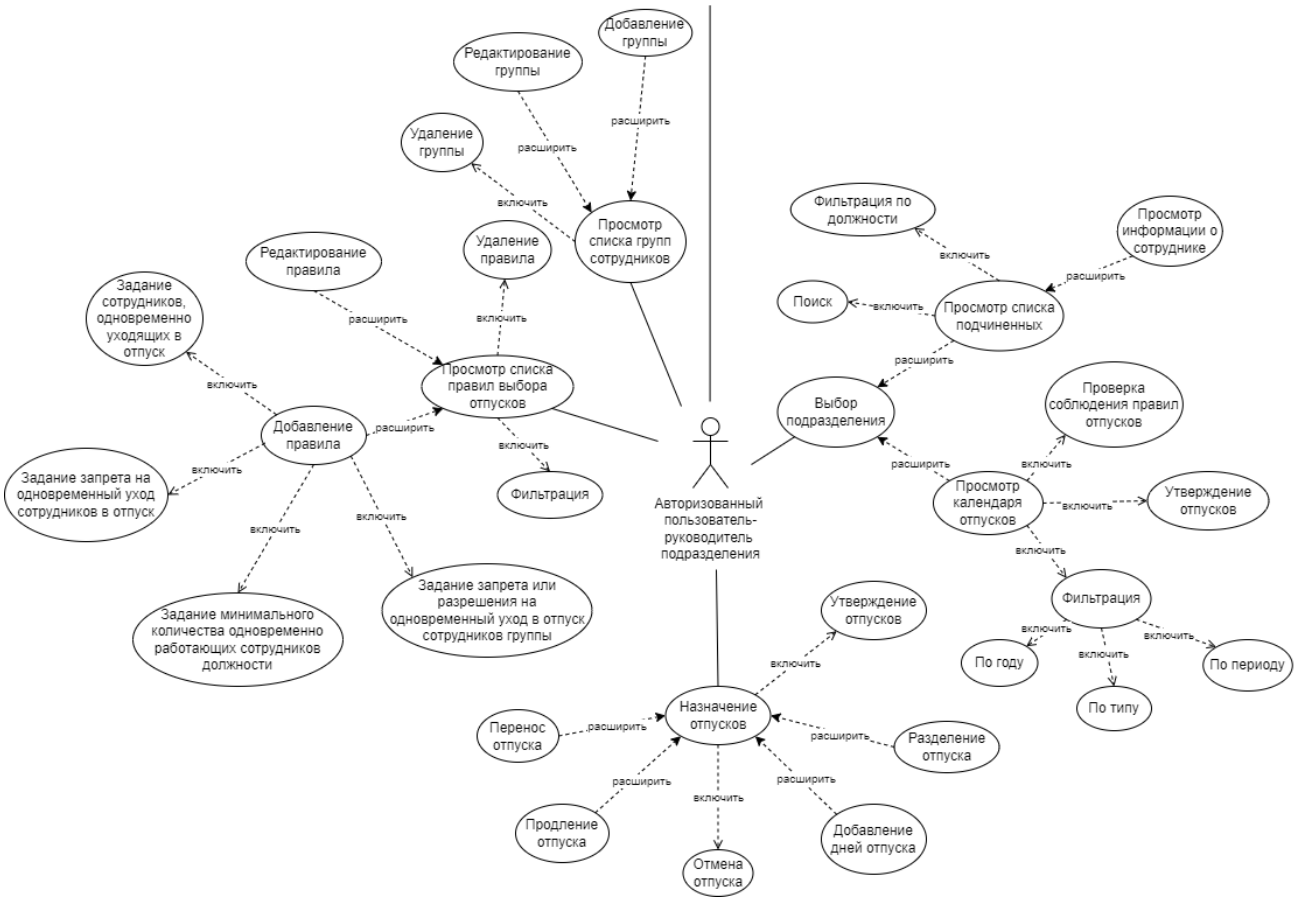


Рисунок 1.4 – Авторизованный пользователь-руководитель подразделения

Этот тип пользователя наследуется от роли «авторизованный пользователь-сотрудник» и поэтому имеет те же возможности в дополнение к своим собственным:

1. Просмотр списка созданных правил выбора отпусков различных типов, их поиск и фильтрация, создание новых правил, редактирование и удаление.
2. Просмотр списка групп сотрудников, создание, редактирование и удаление таких групп.
3. Просмотр списка подчиненных подразделений и сотрудников внутри них, как и информации об отдельных сотрудниках и подразделениях.
4. Календарь отпусков, где можно как просмотреть отпуска за выбранный год в форме графика, так и проверить их на соответствие правилам с последующим утверждением отпусков.

5. Управление отпусками сотрудников: назначение, утверждение, перенос, продление, отмена отпусков, выдача сотрудникам отпускных дней.

1.3 Диаграмма и описание базы данных отпусков

Диаграмма базы данных, реализованной в СУБД MySQL, представлена на рисунке 1.5.

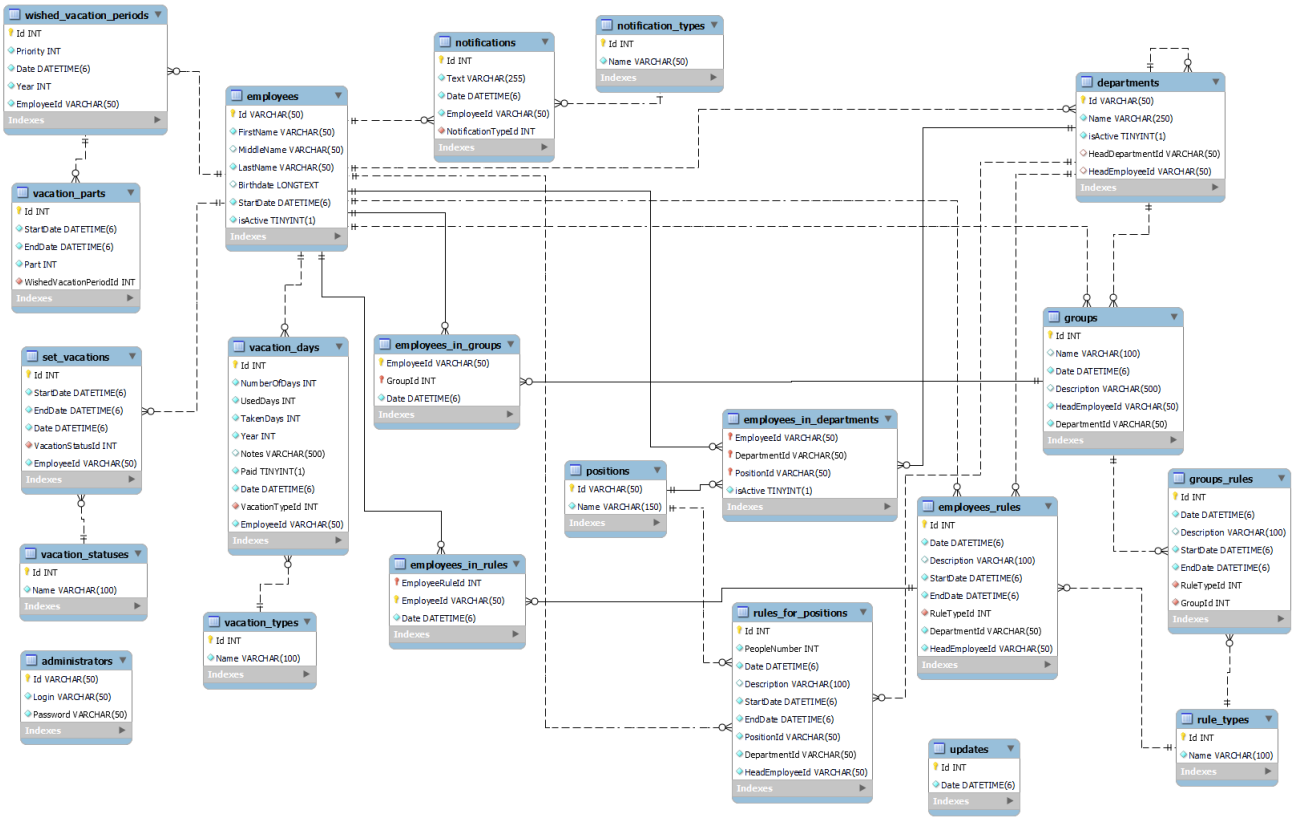


Рисунок 1.5 – Диаграмма базы данных

Условно таблицы в этой БД можно разделить на несколько групп в зависимости от их предназначения. Первый тип таблиц – это таблицы с данными из API: сотрудники (employees), подразделения (departments), должности (positions), а также служебная таблица для соединения сотрудников и подразделений employees_in_departments. В самой БД проекта эти таблицы физически не существуют, так как на самом деле они хранятся на сервере ТПУ.

Вторая группа таблиц отвечает за правила выбора периодов отпусков, которые накладывают определенные ограничения на устанавливаемые пользователями периоды отпусков (таблица 1.1).

Таблица 1.1 – Таблицы правил выбора периодов отпусков

Название таблицы	Перевод	Описание таблицы
rule_types	Типы правил	Справочник, содержащий виды правил для разрешения либо запрещения сотрудникам уходить в отпуск в определенных комбинациях. На момент проектирования рассматриваются два типа правил: «заданные сотрудники уходят в отпуск одновременно» и «заданные сотрудники не могут уходить в отпуск одновременно».
employees_rules	Правила для сотрудников	Таблица, определяющая правила ухода в отпуск для комбинаций конкретных сотрудников (не должностей). Задаются для конкретного руководителя в определенном подразделении.
rules_for_positions	Правила для должностей	Таблица, определяющая правила выбора периодов отпусков для сотрудников определенных должностей, а именно: сколько человек какой должности (как минимум) должно одновременно работать.
employees_in_rules	Сотрудники в правилах	Промежуточная таблица для связи «многие-ко-многим» между сотрудниками и правилами отпуска для них. Таким образом становится возможным задать произвольное количество разных сотрудников, к которым применяются эти правила.
groups	Группы сотрудников	Таблица, предназначенная для хранения создаваемых руководителями групп сотрудников, позволяющих применять и изменять правила отпусков одновременно для всех сотрудников из группы.
employees_in_groups	Сотрудники в группах	Промежуточная таблица для связи «многие-ко-многим», предназначенная для указания принадлежности сотрудников группам.

groups_rules	Правила для групп	Применение к группам правил возможности или невозможности одновременного выхода сотрудников в отпуск

Третья группа таблиц связана непосредственно с процессом выбора отпусков сотрудниками и последующего их утверждения и управления руководителями подразделений. Описание этих таблиц представлено в таблице 1.2.

Таблица 1.2 – Таблицы БД с данными об отпусках

Название таблицы	Перевод	Описание таблицы
wished_vacation_periods	Желаемые периоды отпусков	Выбираемые сотрудниками периоды для отпуска, которые затем переходят на согласование к руководителю.
vacation_parts	Части отпуска	Таблица для хранения отдельных периодов, составляющих вместе один выбранный сотрудником желаемый отпуск на год. Если сотрудник решил потратить все дни отпуска на один отпуск сразу, то в данной таблице будет только один период, иначе несколько.
set_vacations	Назначенные отпуска	Отпуска, уже утвержденные руководителем и передаваемые на формирование графика отпусков. Если отпуск был прерван, продлен или отменен, то указывается соответствующий статус, и при необходимости создается новая запись с новыми датами.
vacation_statuses	Статусы отпусков	Таблица-справочник, хранящая возможные статусы отпусков. На момент проектирования предполагается, что это статусы «Утвержден», «В процессе», «Завершен», «Согласован», «Отменен», «Продлен», «Разделен», «Перенесен», «Прерван».
vacation_types	Виды отпусков	Таблица-справочник для различных источников отпусков, такие как

		основной оплачиваемый отпуск, ненормированный рабочий день, работа с вредными или опасными условиями труда, отпуск по беременности и родам, отпуск без сохранения заработной платы и т.д.
vacation_days	Дни отпусков	Таблица для хранения дней отпусков, получаемых сотрудником по разным причинам (из справочника «Виды отпусков»). Здесь же ведется учет, сколько дней по каждой причине ему было начислено, какую часть из них он уже использовал, и в каком году они были ему назначены.

Таблицы четвертой группы не имеют какого-либо общего предназначения и выполняют различные функции. Их описание представлено в таблице 1.3.

Таблица 1.3 – Описание оставшихся таблиц БД

Название таблицы	Перевод	Описание таблицы
notifications	Уведомления	Таблица для хранения уведомлений для пользователей, включая текст уведомления, дату отправки и идентификатор сотрудника-получателя уведомления.
notification_types	Типы уведомлений	Таблица-справочник для различных типов уведомлений: приближающийся уход подчиненного сотрудника в отпуск, выход сотрудника из отпуска, выбор сотрудником периода отпуска, утверждение отпуска, прерывание, перемещение или отмена; заявка на изменение утвержденного отпуска.
administrators	Администраторы	Таблица для локальных пользователей-администраторов системы, имеющих собственные логин и пароль.
updates	Обновления	Таблица для отслеживания ежедневных обновлений записей об отпусках сотрудников. Каждый день в таблицу вносится дата очередного обновления.

1.4 Формат обмена данными с API ТПУ

Как было указано ранее, для работы системы необходимо получать определенные данные из API ТПУ, на основе которых будет выполняться функционирование всей системы. Получаемые данные должны быть представлены в формате JSON-сообщений, поэтому было выполнено проектирование структуры таких сообщений, а также конкретных методов API, требуемых для работы системы. Результаты проектирования кратко представлены в таблице 1.4.

Таблица 1.4 – Методы API

Метод API	Параметры	Результат	Данные
Календарь выходных и праздничных дней	Год	Список дат выходных и праздничных дней.	Даты, в которые сотрудники университета отдыхают
Список должностей	Нет	Список всех должностей, которые есть в ТПУ.	Идентификаторы, наименования должностей.
Список подразделений	Нет	Список всех подразделений ТПУ.	Идентификаторы, наименования всех подразделений, а также идентификаторы их руководителей и старших подразделений.
Подразделение	Идентификатор подразделения	Подразделение с переданным в качестве параметра идентификатором.	Идентификатор и наименование подразделения, идентификатор руководителя, идентификатор старшего подразделения.
Должности сотрудника	Идентификатор сотрудника	Список должностей указанного сотрудника.	Список из пар идентификаторов подразделений и соответствующих

			должностей сотрудника в них.
Сотрудник	Идентификатор сотрудника	Сотрудник с переданным в качестве параметра идентификатором.	Идентификатор сотрудника, его ФИО, ставка, дата начала работы в ТПУ, дата рождения, список должностей в ТПУ (пары идентификаторов должностей и подразделений).
Сотрудники подразделения	Идентификатор подразделения	Список сотрудников указанного подразделения.	Список данных о сотрудниках в формате, аналогичном ответу метода API получения данных об одном сотруднике.
Старшее подразделение	Идентификатор подразделения	Информация о подразделении, старшем в иерархии относительно указанного.	Аналогичны ответу метода API получения данных о подразделении.
Руководитель подразделения	Идентификатор подразделения	Информация о сотруднике, являющемся руководителем указанного подразделения.	Аналогичны формату данных об одном сотруднике.
Должности в подразделениях	Идентификатор подразделения, идентификатор сотрудника	Информация о должностях указанного сотрудника в одном подразделении.	Список пар идентификаторов подразделений и должностей в них.
Должность	Идентификатор должности	Информация о должности с указанным идентификатором.	Идентификатор, наименование должности в ТПУ.

Младшие подразделения	Идентификатор подразделения	Список подразделений, являющихся младшими в иерархии относительно указанного.	Список данных в формате, аналогичном данным об одном подразделении.
Подчиненные подразделения	Идентификатор руководителя	Список подразделений, которыми управляет указанный сотрудник.	Список подразделений в формате, аналогичном данным об одном подразделении.
Подчиненные сотрудники	Идентификатор руководителя	Список сотрудников, подчиненными указанному руководителю.	Список сотрудников в формате, аналогичном данным об одном сотруднике.

1.5 Карта веб-приложения

На рисунке 1.6 представлена карта веб-приложения. Как можно видеть, она разделена на 5 зон:

1. Зона неавторизованного пользователя, включающая в себя только страницу авторизации.
2. Зона личного кабинета, доступ к которой имеют авторизованные пользователи всех типов.
3. Зона администратора, включающая в себя страницы со списками сотрудников и подразделений и информацией об отдельных сотрудниках и подразделениях.
4. Зона сотрудника, в которой он может управлять графиком своих отпусков, просматривать список уведомлений и информацию о своих отпускных днях.
5. Зона руководителя, включающая в себя рассмотренные ранее функции по управлению отпусками внутри подразделения.

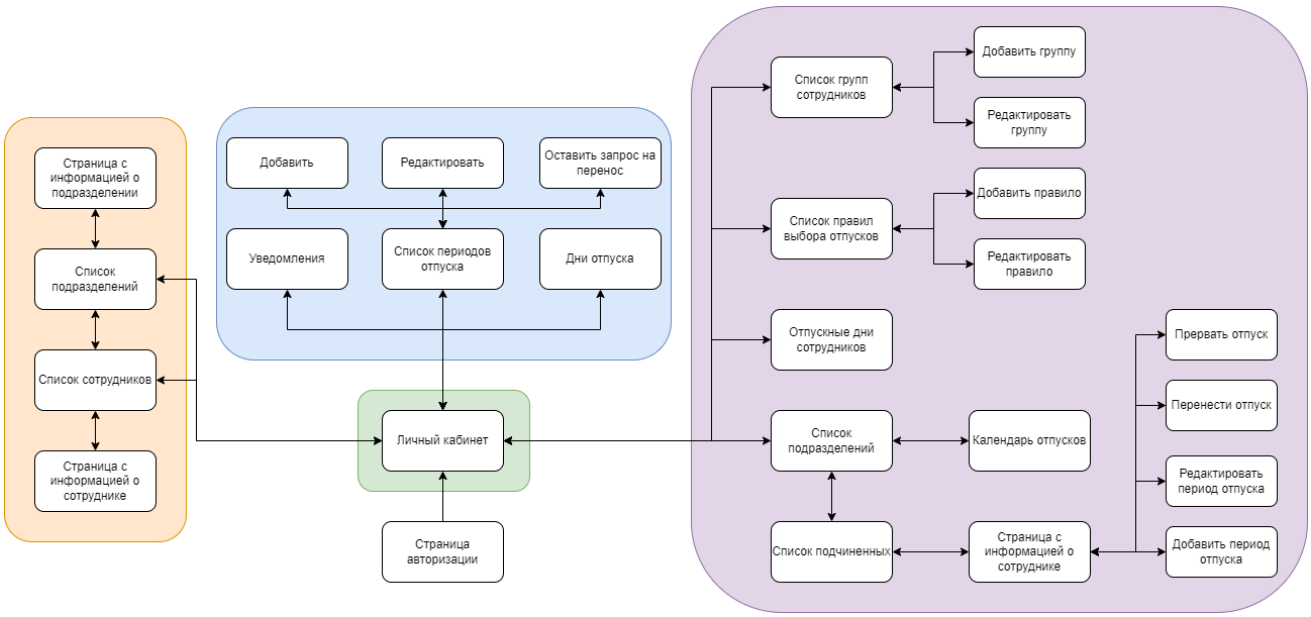


Рисунок 1.6 – Карта веб-приложения

2 Выбор технологий для разработки веб-приложения

Согласно требованиям потенциальных пользователей системы, разрабатываемая программа должна представлять собой веб-приложение, обращающееся к API ТПУ для получения части данных и имеющее также собственную базу данных для хранения сведений об отпусках сотрудников. Кроме того, отдельным важным требованием была выдвинута кроссплатформенность: как СУБД, так и веб-приложение должны иметь возможность работы на операционных системах семейства Linux.

Веб-приложение – это программа, хранимая и работающая на удаленном сервере, доступ к которой осуществляется через интернет посредством веб-браузера [3]. Одним из главных преимуществ подобных приложений в плане удобства пользователей заключается в том, что для работы с ним не требуется загружать и устанавливать программу на свой компьютер. Вместо этого, все функции приложения доступны через обычный веб-браузер, который пользователь использует для выхода в интернет. Следовательно, сразу множество пользователей всегда получает доступ к одной и той же, наиболее актуальной версии приложения, причем они могут пользоваться им с любой платформы и из любого браузера.

Как было указано выше, для обмена данными между приложением и сервером ТПУ необходимо использовать API. API – это совокупность методов и инструментов в виде интерфейса, через который возможно обеспечение взаимодействия между программами [4].

Такой способ организации обмена данными имеет ряд достоинств [5]:

1. Безопасность. Приложение-клиент, обращающееся к серверу за данными, не имеет никакого доступа к внутренней логике обработки данных на сервере или его хранилищу данных, тогда как сервер также ничего не знает о программе-клиенте. Вместо этого обе стороны обмениваются только запрашиваемыми небольшими объемами

данных в заданных заранее форматах и требуемых для решения каких-то определенных задач.

2. API разрабатываются не только как простой программный код, но и как полноценный продукт, предназначенный для определенной аудитории, имеющий документацию и собственные обновления.
3. Как правило, API достаточно стандартизированы, благодаря чему обеспечивается более высокий уровень безопасности, простоты поддержки и работы с ними.

В качестве формата сообщений для получения данных из API используется JSON – формат обмена данными, удобный для чтения и написания как человеком, так и компьютером [6]. Этот формат состоит из двух основных структур данных, которые имеют свою реализацию в абсолютном большинстве распространенных языков программирования: коллекции пар, состоящих из ключа и значения, и упорядоченные списки значений.

К преимуществам JSON относятся простота его чтения, благодаря чему человек может работать с такими сообщениями без необходимости предварительных преобразований; поддержка практически всеми современными языками программирования, так что возможно найти подходящие библиотеки для этого формата при работе с любым языком.

XML (eXtensible Markup Language) – расширяемый язык разметки, используемый для хранения и передачи данных, благодаря чему он может использоваться не только в API, но и в коде (например, он имеет широкое применение в Java-приложениях, таких как веб-приложения на основе фреймворка Spring и мобильные приложения на ОС Android) [8].

В то время, как JSON представляет собой совокупность пар «ключ-значение» и списков значений, сообщения и файлы в формате XML состоят из элементов, заключенных в теги. Теги определяют имена параметров сообщения, а строки либо числа внутри них – значения этих параметров. Кроме того, элементы могут иметь атрибуты, несущие дополнительную информацию, что

значительно повышает возможности её передачи и использования посредством XML-сообщений.

Как XML, так и JSON имеют множество библиотек, облегчающие работу с ними в практически любом современном языке программирования. Однако при их сравнении предпочтение было отдано именно JSON по причине простоты и легковесности сообщений этого формата [7]. XML потенциально имеет большую гибкость и мощность в его сообщениях (например, за счет атрибутов тегов), также он более безопасный и позволяет проще находить ошибки в их структуре [9]. Однако в данном веб-приложении эти возможности оказываются не так важны, так как они используются для передачи относительно простой и однотипной информации: сотрудники, подразделения, должности, а также списки из этих же объектов. Следовательно, система тегов в XML в данном случае оказывается недостатком из-за неоправданного увеличения размера сообщений по сравнению с аналогичными сообщениями в формате JSON.

При выборе языка программирования было необходимо ориентироваться на существующие у языков популярные инструменты и фреймворки для разработки веб-приложений. В итоге для рассмотрения были взяты такие технологии, как Spring (10-е место по популярности среди веб-фреймворков на 2021 год [10]), Django (9-е место) и ASP.NET Core (6-е место).

Django – это высокоуровневый фреймворк с открытым кодом на языке программирования Python, предназначенный для быстрой разработки с возможностью фокусироваться на написании требуемого функционала приложения, не отвлекаясь на реализацию тривиальных и многократно созданных ранее частей таких приложений [11]. Этот фреймворк обладает высокой популярностью, и на нем были написаны такие сайты, как Pinterest, Washington Times и BitBucket. К преимуществам Django относятся [12]:

1. Использование языка программирования Python, который, в свою очередь, имеет такие достоинства, как простота написания и чтения кода, большое количество учебных материалов и разнообразных мощных библиотек, в том числе предназначенных для сферы искусственного интеллекта.
2. Большое количество уже готового функционала: этот фреймворк написан веб-разработчиками для веб-разработчиков, поэтому они знают, какие проблемы программисты чаще всего встречают при создании веб-приложений.
3. Возможность быстрой разработки и масштабируемость за счет использования в архитектуре веб-приложений Django слабосвязанных компонентов, так что становится возможным работать независимо над различными модулями одновременно, и при этом сами модули не оказывают друг на друга большого влияния.

Тем не менее, данный фреймворк имеет и недостатки. Например, Django требует соблюдения определенного стиля, он имеет свои предопределенные переменные, структуру файлов и правил, которые обязательно нужно изучить перед началом работы с ним и соблюдать, потому что в противном случае не удастся создать работающее веб-приложение.

Spring – это один из самых популярных фреймворков для создания веб-приложений на Java, обеспечивающий базовую поддержку управления зависимостями, управление транзакциями веб-приложений, доступ к данным, обмен сообщениями и т.д. [13]. Spring включает в себя также отдельные модули, которые могут быть подключены или отключены от проекта в зависимости от их надобности: Data Access (транзакции данных, ORM – технология доступа к данным), Testing (тестирование), Core (внедрение зависимостей, события, ресурсы, валидация, привязка данных) и другие. Веб-приложения, написанные с использованием Spring, находят применение в самых разных областях: государственные, страховые, оборонительные, образовательные сайты и т.д.

К достоинствам этого фреймворка можно отнести [14]:

1. Гибкость – Spring имеет разнообразные и многочисленные библиотеки для работы, позволяет выбирать между форматами для настроек конфигурации, в целом предоставляет большое количество функций и возможностей, что делает разработку быстрее и проще.
2. Поддержка кэширования, валидации, транзакций, форматирования.
3. Высокая производительность.
4. Безопасность – фреймворк следит за зависимостями от сторонних разработчиков, предоставляет регулярные обновления для сохранения данных приложений в безопасности, также включает в себя отдельный фреймворк Spring Security с различными решениями для того, чтобы сделать разрабатываемую систему безопасной и защищенной.
5. Возможность интеграции с другими программами, основанными на Spring, причем не только веб-приложениями.

Одним из главных недостатков является сложность данного фреймворка: он имеет высокий порог вхождения, и для его освоения желательно иметь большое количество опыта в разработке. Более того, некоторые особенности также усложняют работу с ним: например, существует разнообразные варианты для распараллеливания кода, из которых сложно сделать правильный выбор для разработчиков, только начинающих работу со Spring, что может иметь значительные негативные последствия.

ASP.NET Core – кроссплатформенная, высокопроизводительная среда с открытым исходным кодом для создания современных облачных приложений, подключенных к Интернету [15]. В целом, эта технология обладает рядом различных преимуществ, таких как:

1. Поддержка паттерна MVC. Данный шаблон, с одной стороны, облегчает процесс создания веб-приложения, так как он задает логическое разделение трех основных типов компонентов приложения: модель, вид и контроллер. С другой стороны, такое разделение обеспечивает и гибкость созданного приложения: при необходимости внесения изменений и добавления нового

функционала нужно редактировать или дополнять код только в соответствующих модулях, тогда как другие модули либо остаются нетронутыми, либо изменения в них будут минимальными.

2. Кроссплатформенность. Как было упомянуто ранее, разработанное веб-приложение может быть развернуто не только в ОС Windows, но также на Mac OS и Linux [20], что позволяет разработчику не ограничивать себя использованием только одной поддерживаемой операционной системой. Для развертывания веб-приложения может использоваться либо IIS, либо кроссплатформенный веб-сервер Kestrel.
3. Модульность. Необходимые модули могут быть легко подключены посредством менеджера пакетов Nuget, что позволяет значительно упростить разработку за счет использования большого разнообразия уже существующих библиотек и фреймворков. Например, для данного проекта предполагается использовать такие пакеты, как Entity Framework Core (для работы с базой данных) и Newtonsoft.Json (для парсинга JSON-сообщений).
4. Использование языка программирования C#. C# – язык, изначально предназначавшийся для веб-разработки, позволяющий строить крупные, но гибкие, масштабируемые и расширяемые приложения, причем он относительно прост для освоения и имеет большое количество специальных конструкций, разработанных для облегчения процесса написания и понимания кода [15].

При непосредственном сравнении ASP.NET Core и Spring можно отметить, что в этих веб-фреймворках одинаковые функции реализованы схожим образом, поэтому в плане функциональности выбор может быть сделан в пользу любого из них без вреда для разрабатываемого приложения [16]. Кроме того, как Java, так и C# имеют похожую производительность, и проекты, написанные на этих фреймворках, являются кроссплатформенными, что в данном проекте является необходимым условием. Существуют некоторые

различия в подходах, используемых в их языках программирования: например, Java – консервативный язык, многие элементы которого выполнены в других языках (в том числе в C#) намного проще и эффективнее, что облегчает разработку на них. С другой стороны, благодаря этому обеспечивается высокий уровень надежности Java [17].

Тем не менее, решающим фактором для выбора в данном случае оказалось наличие предшествующего опыта разработки на языке программирования C#: как было отмечено ранее, у Spring относительно высокий порог вхождения, требующий наличия большого предшествующего опыта веб-программирования, предварительного внимательного изучения и чтения документации, получения опыта работы с этим фреймворком. В связи с ограниченными сроками выполнения проекта было необходимо использовать технологию, с помощью которой можно было бы начать разработку без долгой предварительной подготовки, поэтому в сравнении ASP.NET Core и Spring был сделан выбор в пользу первого варианта.

Одним из отличий Django и ASP.NET Core является используемый паттерн фреймворка. Так, ASP.NET Core использует MVC (Model-View-Controller) – шаблон проектирования, согласно которому приложение должно быть разбито на три компонента [18]:

1. Контроллер – класс, обеспечивающий связь между пользователем и данными. Контроллер получает запрос от пользователя, обрабатывает его, обращаясь к слою данных приложения (моделям), и возвращает результат, который может быть представлен веб-страницей (представлением), файлом, строкой и т.д.
2. Представление – это HTML-страницы, отображаемые пользователю как интерфейс веб-приложения, посредством которого он может взаимодействовать с системой.
3. Модель – это данные веб-приложения и связанная с ними логика.

Достоинством этого паттерна является факт взаимной независимости компонентов, благодаря чему возможно работать одновременно над разными

частями приложения, практически не затрагивая другие компоненты MVC, а также вносить изменения без необходимости редактировать также код других компонентов.

В Django используется собственная интерпретация MVC – MTV (Model-Template-View). Ключевое различие между паттернами в Django и ASP.NET Core заключается в роли контроллера MVC и представления MTV. В отличие от MVC, в Django роль контроллера, передающего запрос требуемого представления, выполняет сам фреймворк, тогда как представление отвечает за передачу в шаблон (компонент Template) необходимых данных, которые нужно представить пользователю.

При сравнении паттернов, используемых в рассматриваемых фреймворках, предпочтение было отдано MVC: несмотря на то, что MTV позволяет очень просто редактировать компоненты приложения, MVC лучше приспособлен для более крупных веб-приложений, к которым относится и данная разработка, так как она будет включать в себя большое количество разнообразных модулей, многие из которых имеют ряд собственных методов контроллеров и представлений [19]. Тем не менее, здесь стоит отметить, что Django был бы однозначным выбором для данного проекта, если в нем нужно было бы использовать библиотеки для машинного обучения или Big Data, которые являются сильной стороной Python, как и простота этого языка программирования для изучения.

В итоге было принято решение использовать для разработки ASP.NET Core из-за поддержки MVC, большого сообщества разработчиков и количества разнообразных обучающих материалов, кроссплатформенности, а также наличия предшествующего опыта работы с данной технологией, благодаря чему возможно обеспечить как более высокий уровень качества исполнения готового приложения, так и скорости работы над ним.

В качестве СУБД была выбрана MySQL – одна из самых популярных СУБД, распространяемая свободно и имеющая возможности эффективной работы с интернет-сайтами и веб-приложениями. Она имеет ряд достоинств,

повлиявших на выбор именно этой СУБД, такие как простота в использовании, обширный функционал, безопасность, масштабируемость и высокая скорость работы [22].

В качестве основного аналога для использования была рассмотрена СУБД MS SQL Server. У этой системы управления базами данных есть большое количество сильных сторон: полная поддержка проектами .NET, благодаря чему подключение базы данных к проекту выполняется очень просто; богатый функционал и удобство в использовании связанного с данной СУБД инструмента SQL Server Management Studio, который лучше при сравнении с аналогичным инструментом для MySQL – MySQL Workbench; высокая производительность. Однако у этой СУБД есть важный недостаток, не позволяющий её использовать для данного проекта – возможность развертывания только на операционных системах семейства Windows, тогда как необходима кроссплатформенность. Это и стало главной причиной выбора MySQL в качестве СУБД для разрабатываемого веб-приложения.

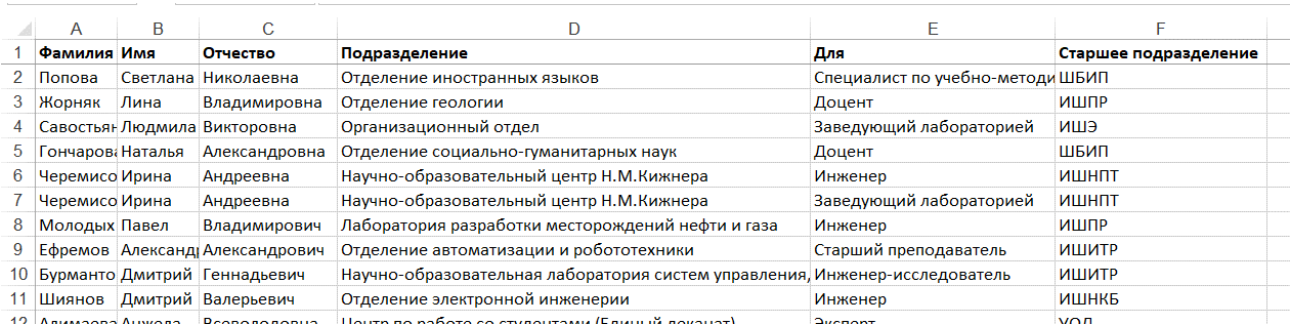
3 Разработка веб-приложения для согласованного планирования отпусков

3.1 Подготовка данных о сотрудниках, подразделениях и должностях

При реальной работе веб-приложение должно получать данные из API ТПУ посредством выделенных на этапе проектирования методов. Однако на этапе разработки было решено сымитировать получение данных из API локальными файлами в формате JSON. Это решение было принято по следующим причинам:

1. Необходимость проверки работы приложения с разработанными ранее JSON-сообщениями. Возможно, что на этапе проектирования были допущены ошибки, и какие-то данные должны передаваться в другом формате, какие-то необходимые данные были пропущены, а какие-то, включенные в структуру сообщений, наоборот, избыточны. Для того, чтобы не обращаться в отдел разработки API ТПУ для правок, сначала следует проверить работу программы на тестовых JSON-сообщениях.
2. Расширенные возможности тестирования. Если API ТПУ возвращает реальные данные, то в локальных файлах имеется возможность править их множеством различных способов, чтобы проверить корректность работы программы при разных входных данных. Помимо этого, таким образом возможно осуществлять «авторизацию» любых пользователей с любыми ролями, тогда как при подключении к API ТПУ будет возможна работа только от своего профиля.

Для того, чтобы не создавать JSON-файлы, имитирующие ответы API, вручную, было решено использовать в качестве первичного источника данных Excel-таблицу со списком сотрудников и их должностей в отделениях, представленную на рисунке 3.1.



	A	B	C	D	E	F
1	**Фамилия**	**Имя**	**Отчество**	**Подразделение**	**Для**	**Старшее подразделение**
2	Попова	Светлана	Николаевна	Отделение иностранных языков	Специалист по учебно-методи	ШБИП
3	Жорняк	Лина	Владимировна	Отделение геологии	Доцент	ИШПР
4	Савостьян	Людмила	Викторовна	Организационный отдел	Заведующий лабораторией	ИШЭ
5	Гончарова	Наталья	Александровна	Отделение социально-гуманитарных наук	Доцент	ШБИП
6	Черемисо	Ирина	Андреевна	Научно-образовательный центр Н.М.Кижнера	Инженер	ИШНПТ
7	Черемисо	Ирина	Андреевна	Научно-образовательный центр Н.М.Кижнера	Заведующий лабораторией	ИШНПТ
8	Молодых	Павел	Владимирович	Лаборатория разработки месторождений нефти и газа	Инженер	ИШПР
9	Ефремов	Александ	Александрович	Отделение автоматизации и робототехники	Старший преподаватель	ИШИТР
10	Бурманто	Дмитрий	Геннадьевич	Научно-образовательная лаборатория систем управления,	Инженер-исследователь	ИШИТР
11	Шиянов	Дмитрий	Валерьевич	Отделение электронной инженерии	Инженер	ИШНКБ
12	Алимаева	Анжела	Всеволодовна	Центр по работе со студентами (Единый деканат)	Эксперт	УОД

Рисунок 3.1 – Таблица с данными о сотрудниках ТПУ

Для того, чтобы выделить данные из этой таблицы и преобразовать в JSON-сообщения, было создано вспомогательное консольное приложение, работающее с таблицей сотрудников в формате CSV. Данное приложение включает в себя два статических класса, описание которых представлено в таблице 3.1.

Таблица 3.1 – Описание статических классов приложения парсера

Название класса	Описание класса	Название метода	Описание метода
ListReader	Предназначен для считывания строк из исходного файла CSV и генерации списка из данных, хранящихся в каждой такой строке.	ReadList	Открытие и считывание исходного файла, создание посредством него списка с данными.
		ReplaceName	Приватный метод для замены кратких наименований подразделений на полные.
		ClearName	Приватный метод для удаления лишних специальных символов из наименований.
MessagesCreator	Преобразует список строк из исходного файла в объекты классов сотрудников, должностей и подразделений; формирует JSON-файлы ответов от API.	WriteToFile	Создает и заполняет файл в формате JSON с указанными расположением, именем и данными.
		FillTime	На основе имеющегося количества должностей сотрудников генерирует для них размер ставки.
		FillDates	Случайным образом заполняет данные о дате рождения и начале работы

			сотрудника в ТПУ (в связи с отсутствием этих данных в исходной таблице).
		Методы Create	На основе строк из исходного файла формируются объекты классов сотрудников, подразделений и должностей.
		Методы Make	Создают JSON-файлы, имитирующие ответы от API, рассмотренные в разделе 1.4.

Возможно рассмотреть принцип работы программы-парсера на примере формирования файлов с данными о подразделениях. Так, сначала вызывается метод ReadList класса ListReader, который построчно считывает исходный файл CSV и формирует из него список объектов, соответствующих строкам этих файлов. Соответствующий класс представлен в листинге 3.1.

Листинг 3.1 – Класс Item для хранения данных из строк CSV-файла

```
internal class Item
{
    // имя
    public string FirstName { get; set; }
    // отчество
    public string MiddleName { get; set; }
    // фамилия
    public string LastName { get; set; }
    // подразделение
    public string Department { get; set; }
    // должность
    public string Position { get; set; }
    // старшее подразделение
    public string HeadDepartment { get; set; }
}
```

Далее вызывается метод CreateDepartmentsList класса MessagesCreator. На основе полученного списка строк этот метод формирует список объектов класса подразделения Department (листинг 3.2).

Листинг 3.2 – Класс для сохранения данных о подразделении

```
internal class Department
{
```

```
    public string Id { get; set; }
    public string Name { get; set; }
    public string Head { get; set; }
    public string HeadDepartment { get; set; }
}
```

Работа данного метода состоит из двух частей. Сначала из строк, полученных из CSV файла, заполняется список подразделений, у которых указываются идентификатор, имя и наименование старшего подразделения, как показано в листинге 3.3.

Листинг 3.3 – Заполнение списка подразделений

```
foreach(Item item in items)
    if(!departments.Any(d => d.Name == item.Department))
        departments.Add(new Department
        {
            Id = departments.Count().ToString(),
            Name = item.Department,
            HeadDepartment = item.HeadDepartment,
            Head = null
        });
```

Во второй части метода (листинг 3.4) наименования старших подразделений конвертируются в их идентификаторы (на основе уже готового списка подразделений).

Листинг 3.4 – Преобразование имен подразделений в идентификаторы

```
foreach(Department department in departments)
{
    Department headDep = FindDepartmentByName
    (department.HeadDepartment, departments);
    if (headDep != null)
        department.HeadDepartment = headDep.Id;
    else
        department.HeadDepartment = null;
}
```

Для окончания обработки подразделений вызывается также метод FillHeads, используемый после заполнения списка всех сотрудников из CSV-файла. Так как теперь известны как все подразделения, так и все их сотрудники со своими идентификаторами, то возможно получать списки сотрудников внутри подразделений (для чего была реализована приватная функция

GetEmployeesOfDepartment). В свою очередь, метод FillHeads получает для всех подразделений списки их сотрудников и записывает в поле Head идентификаторы одного случайного сотрудника (так как в исходном CSV-файле нет информации о руководителях, и при необходимости JSON-файлы будут редактироваться), как можно видеть в листинге 3.5.

Листинг 3.5 – Заполнение руководителей подразделений

```
Random rnd = new Random();
foreach(Department dep in DepList)
{
    List<Employee> emps = GetEmployeesOfDepartment(dep.Id);
    int index = rnd.Next(0, emps.Count);
    dep.Head = emps[index].Id;
}
```

При запуске программы отображаются ход и успешность создания JSON-файлов, как можно видеть на рисунке 3.2.

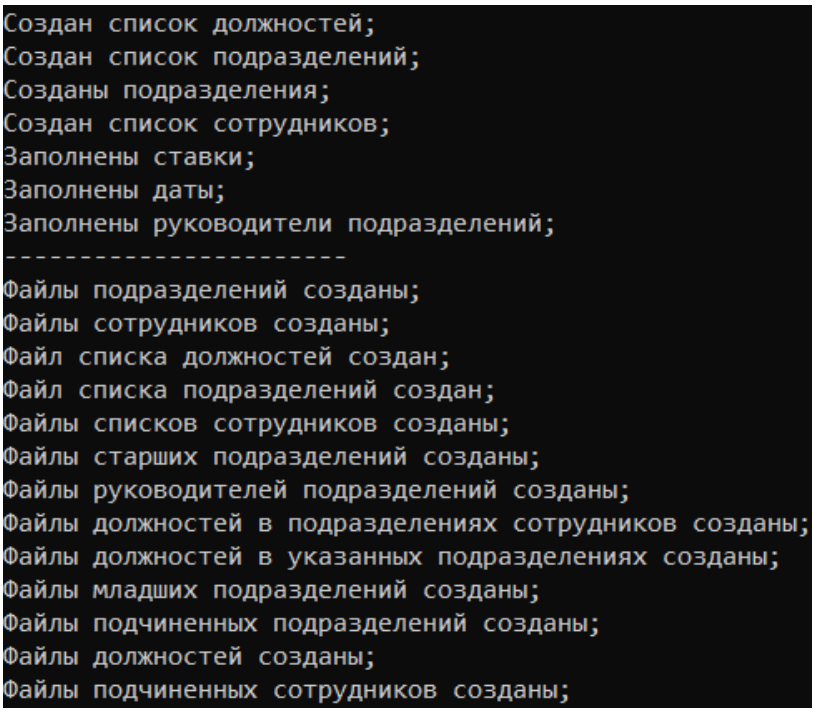


Рисунок 3.2 – Ход выполнения программы-парсера

При успешном выполнении JSON-файлами заполняются все папки (симулирующие методы API), представленные на рисунке 3.3.

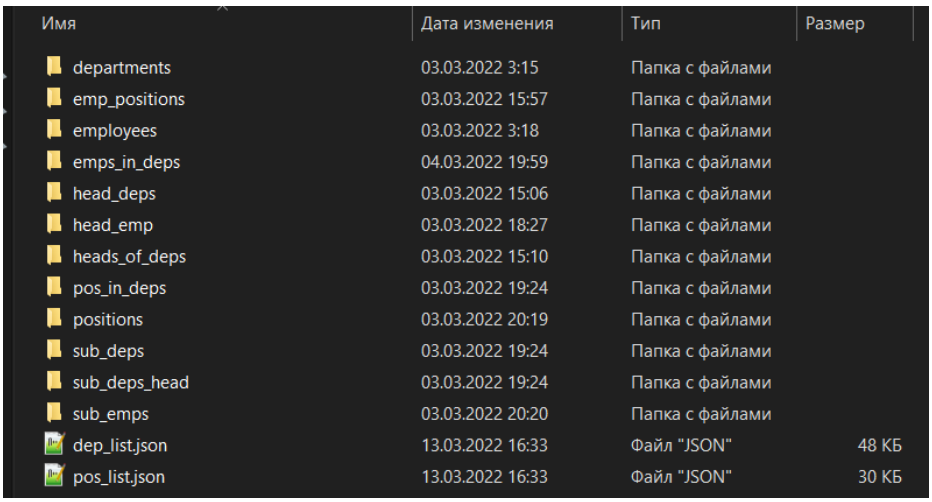


Рисунок 3.3 – Папки с JSON-файлами, имитирующими ответы от API

В качестве примера содержания файлов можно привести данные о подразделении с идентификатором «9», приведенные в листинге 3.6.

Листинг 3.6 – Структура JSON-сообщения с данными о подразделении

```
{
  "Id": "9",
  "Name": "Центр по работе со студентами (Единый деканат)",
  "Head": "9",
  "HeadDepartment": "166"
}
```

Другой пример – JSON-файл с данными о сотруднике, показанный в листинге 3.7.

Листинг 3.7 – Структура JSON-сообщения с данными о сотруднике

```
{
  "Id": "100",
  "FirstName": "Михаил",
  "MiddleName": "Иванович",
  "LastName": "Окунцов",
  "Time": 0.5,
  "StartDate": "2019-06-17T00:00:00",
  "BirthDate": "1991-09-07T00:00:00",
  "Positions": [
    {
      "Position": "18",
      "Department": "62"
    }
  ]
}
```

3.2 Создание базы данных для отпусков

После выполненного проектирования, выбора технологий и подготовки данных для работы системы возможно начать непосредственную разработку веб-приложения.

В первую очередь, в проекте были созданы модели – классы, используемые технологией Entity Framework Core для отображения и работы с таблицами БД. Каждая модель представляет собой отдельную таблицу и при этом может быть использована в коде программы для создания объектов этих классов – строк таблицы.

На рисунке 3.4 представлены модели, созданные в проекте.

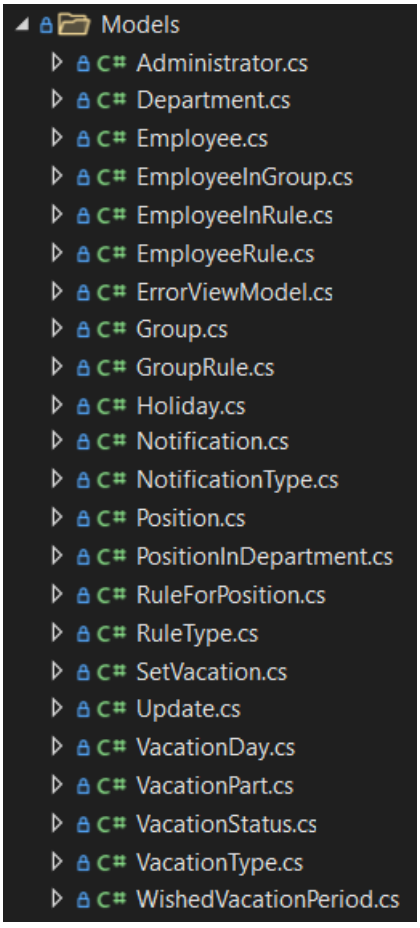


Рисунок 3.4 – Модели проекта

В качестве примеры структуры класса-модели в листинге 3.8 приведен код класса SetVacation (утвержденный отпуск). Для модели в аннотации указано, что она должна быть сохранена в базе данных под именем «set_vacations», тогда как поля этого класса станут атрибутами таблицы, причем для некоторых из них

тоже указаны аннотации: Required для обязательных значений и MaxLength для ограничения размера строки. Кроме того, здесь можно видеть внешние ключи на модели VacationStatus (таблицу-справочник со статусами отпусков) и Employee (сотрудник, которому назначается отпуск).

Листинг 3.8 – Модель SetVacation

```
[Table("set_vacations")]
public class SetVacation
{
    public int Id { get; set; }

    [Required]
    public DateTime StartDate { get; set; }

    [Required]
    public DateTime EndDate { get; set; }

    public DateTime Date { get; set; }

    public int VacationStatusId { get; set; }
    public VacationStatus VacationStatus { get; set; }

    [Required, MaxLength(50)]
    public string EmployeeId { get; set; }
    public Employee Employee { get; set; }
}
```

Для тех таблиц, которых физически в базе данных быть не должно (например, таблицы сотрудников и подразделений), указывается специальная аннотация NotMapped (листинг 3.9), означающая игнорирование этой модели при создании БД.

Листинг 3.9 – Использование аннотации NotMapped для модели Employee

```
[NotMapped, Table("employees")]
public class Employee
{ … }
```

Созданные модели необходимо включить в контекст базы данных и соединить с ней. Поэтому далее был отредактирован файл appsettings.json, в который была добавлена строка подключения к БД, как показано в листинге 3.11.

Листинг 3.10 – Подключение к БД в файле appsettings.json

```
  "ConnectionStrings": {
    "DefaultConnection":
     "server=localhost;
```

```
        user=user_name;
        password=password;
        database=vacationsdb3;"
    }
```

Далее был создан класс ApplicationContext, основанный на классе DbContext (листинг 3.11).

Листинг 3.11 – Создание класса ApplicationContext для работы с БД

```
public class ApplicationContext : DbContext
{ … }
```

В этом классе было определено все необходимое для подключения и работы с БД. Например, указывается список всех тех моделей, которые принадлежат данной БД в качестве наборов данных, как показано на рисунке 3.5.

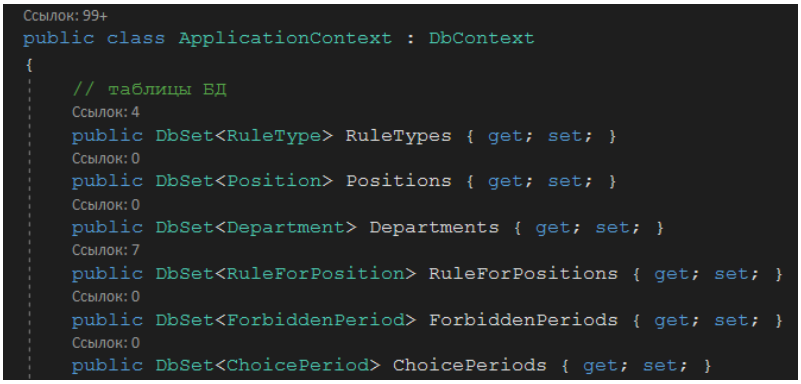


Рисунок 3.5 – Наборы данных БД

В конструкторе по умолчанию, как можно видеть в листинге 3.12, выполняется метод EnsureCreated, который проверяет, создана ли уже база данных. Если она уже существует, то не предпринимается никаких действий. Если её нет, то она создается со структурой, определенной в контексте данных и подключенных моделях.

Листинг 3.12 – Проверка существования БД при обращении к ней

```
public ApplicationContext()
{
    Database.EnsureCreated();
}
```

Был переопределен метод OnModelCreating, выполняющийся при создании моделей, как показано на рисунке 3.6. Некоторые ограничения не могли быть указаны непосредственно в моделях, поэтому они были созданы

здесь, например, составные первичные ключи, являющиеся также внешними ключами.

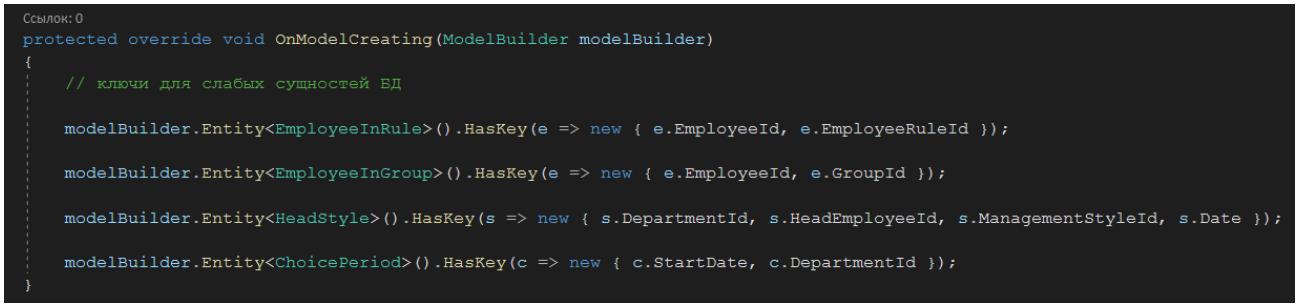


Рисунок 3.6 – Переопределение метода OnModelCreating

Также был переопределен метод OnConfiguring, отвечающий за настройки контекста данных, как показано в листинге 3.13. Такое переопределение было необходимо, чтобы указать параметры подключения к БД из файла appsettings.json, поэтому в коде выполняется обращение к этому файлу и создание новой конфигурации, основанной на получаемой строке подключения.

Листинг 3.13 – Переопределение метода OnConfiguring

```
var builder = new ConfigurationBuilder();
// установка пути к текущему каталогу
builder.SetBasePath(Directory.GetCurrentDirectory());
// получение конфигурации из файла appsettings.json
builder.AddJsonFile("appsettings.json");
// создание конфигурации
var config = builder.Build();
// получение строки подключения
string connectionString =
config.GetConnectionString("DefaultConnection");
```

Затем указывается, что следует использовать СУБД MySql, и в качестве параметров подключения передаются полученная ранее строка ConnectionString и используемая версия СУБД (листинг 3.14).

Листинг 3.14 – Указание параметров подключения к БД

```
optionsBuilder.UseMySql(
    connectionString,
    new MySqlServerVersion(new Version(5, 7, 35))
    );
```

После выполненной настройки контекста данных возможны создание (при первом обращении) и работа с базой данных.

3.3 Получение данных о ТПУ из JSON-сообщений

Как было указано ранее, часть данных программа получает из файлов в формате JSON, имитирующих реальные ответы от API ТПУ. Реализация получения этих данных была обеспечена посредством создания нескольких специальных классов.

Прежде всего, стоит отметить, что формат JSON-сообщений, содержащих данные об отдельных сотрудниках, подразделениях и должностях, был приравнен формату моделей соответствующих сущностей, поэтому для их парсинга не было необходимости создавать отдельные классы. Однако часть ответов API представляет собой списки из таких объектов, и для них были созданы вспомогательные классы, содержащие единственное поле – массив объектов. На рисунке 3.7 наименования таких классов оканчиваются на List. Среди JSON-сообщений есть также такие, которые не могут быть отнесены к какой-то модели – для них был создан класс PositionsInDepartments, представляющий собой пару значений из идентификатора должности и подразделения сотрудника.

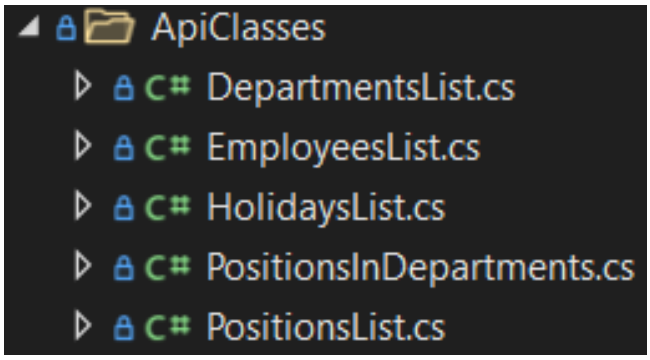


Рисунок 3.7 – Классы для парсинга JSON-сообщений

За подключение к API отвечает статический класс Connector. На данном этапе он выполняет поиск необходимых файлов по названию «метода» и идентификатору искомого объекта, тогда как в будущем при реальном использовании он должен будет подключаться к самому API ТПУ, для чего нужно будет заменить только один метод.

Список методов данного класса:

1. ReadReply – обращается к API и в результате возвращает JSON-ответ в формате строки string.
2. Parse – общий метод, возвращающий объект, получаемый в результате применения к полученной строке JSON метода, передаваемого в делегат Func.
3. Методы Get – методы, соответствующие каждому из методов API; обращаются к методу ReadReply для получения JSON-ответа и затем получают необходимый объект, передавая в метод Parse подходящую функцию-парсер.

Пример одного из методов Get, получающего список всех подразделений, приведен в листинге 3.15.

Листинг 3.15 – Метод получения списка подразделений из БД

```
public static List<Department> GetDepartments()
{
    string deps = ReadReply("dep_list");
    List<Department> departments = (List<Department>)Parse
                                   (Parser.ParseDepartments,
                               deps);
    return departments;
}
```

Как можно видеть, здесь используется метод-парсер из статического класса Parse. В этом классе было реализовано 8 методов для обработки JSON-ответов от API, соответствуя всем возможным форматам сообщений: данные о подразделениях, сотрудниках, должностях (и списки этих данных), производственном календаре, а также списки из пар идентификаторов должностей и подразделений.

Пример одного из методов приведен в листинге 3.16. Получая на вход JSON-сообщение со списком должностей, данный метод десериализует его, используя библиотеку System.Text.Json, в соответствующий промежуточный класс PositionsList. Далее, если этот класс содержит список должностей, он возвращается в формате List<Position> как результат работы метода.

Листинг 3.16 – Метод-парсер списка должностей

```
static public List<Position> ParsePositions(string json)
```

```
{
    try
    {
        PositionsList list = JsonSerializer
                             .Deserialize<PositionsList>(json);
        if (list == null)
            return null;

        if (list.Positions == null)
            return new List<Position>();

        return new List<Position>(list.Positions);
    }
    catch (Exception ex)
    { … }
}
```

3.4 Работа с базой данных отпусков

Для взаимодействия с базой данных был создан целый ряд статических классов DataHandler, описание которых представлено в таблице 3.2.

Таблица 3.2 – Описание классов для работы с БД

Наименование класса	Назначение
DatabaseHandler	Операции для работы с базой данных в целом. Создание и пересоздание БД (методы, необходимые на этапе разработки), добавление начальных данных при создании БД: создание пользователя-администратора, заполнение справочников с типами правил отпусков, видами отпусков, статусов отпусков, уведомлений.
EmployeeRuleDataHandler	Получение, добавление, редактирование, удаление правил выбора отпусков для сотрудников.
GroupDataHandler	Получение, добавление, редактирование, удаление групп сотрудников.
GroupRuleDataHandler	Получение, добавление, редактирование, удаление правил выбора отпусков для групп сотрудников.
NotificatonDataHandler	Получение, добавление, удаление уведомлений; получение типа уведомлений по идентификатору и имени.
PositionRuleDataHandler	Получение, добавление, редактирование и удаление правил выбора отпусков для должностей.
UpdateDataHandler	Проверка последнего обновления данных об отпусках сотрудников, добавление записей о последнем обновлении.

VacationDataHandler	Получение, добавление, редактирование, удаление записей о желаемых и утвержденных отпусках; утверждение отпусков, их перенос, прерывание и отмена, получение статуса отпуска.
VacationDayDataHandler	Назначение сотруднику отпускных дней и редактирование их количества; получение отпускных дней в распоряжении сотрудника (всех или не занятых никаким отпуском); отметка, что отпускные дни были задействованы в некотором отпуске либо, наоборот, что они были высвобождены.
DataHandler	Операции, которые не были включены в другие классы: получение данных о пользователе-администраторе, получение типов правил выбора отпусков из справочника.

Как можно видеть, большинство из данных классов предоставляют методы для простейших операций доступа и работы с данными: получения, добавления, редактирования и удаления. Это было сделано с той целью, чтобы вынести сложную логику работы с данными за пределы данных операций в другие классы. Соответственно, методы классов для работы с БД, с одной стороны, являются универсальными и могут быть использованы повторно множество раз; с другой стороны, это способно облегчить процесс оптимизации алгоритмов как на стороне обращения к БД, так и на стороне логики приложения.

Среди всех классов особо выделяется DatabaseHandler, предназначенный для работы со всей БД в целом. Этот класс имеет следующие методы:

1. RecreateDB – метод, использующийся на этапе разработки, который удаляет и создает заново базу данных. Необходим в тех случаях, когда в структуру БД вносятся изменения вроде редактирования, добавления или удаления моделей.
2. FillInitialData – метод, отвечающий за заполнение базы данных начальными данными, такие как данные справочников и профиль администратора.
3. AddAdmin – приватный метод, добавляющий в БД профиль администратора по умолчанию.

4. AddRuleTypes, AddVacationTypes, AddVacationStatuses, AddNotificationTypes – приватные методы, заполняющие различные таблицы-справочники.

В свою очередь, методы доступа к данным из различных таблиц могут иметь разную структуру. В качестве примера в листинге 3.17 приведен код обращения к БД метода GetVacationType. Данный метод получает идентификатор типа отпуска в качестве параметра, после чего использует его в LINQ-запросе для нахождения соответствующей строки в таблице vacation_types.

Листинг 3.17 – Получение типа отпуска по его идентификатору

```
using (ApplicationContext db = new ApplicationContext())
{
    return db.VacationTypes.FirstOrDefault(vt => vt.Id == id);
}
```

3.5 Контроллеры веб-приложения

Согласно паттерну MVC, контроллеры представляют собой часть веб-приложения, отвечающую за обработку запросов пользователей. Фактически, каждый контроллер представляет собой отдельный модуль приложения, и все созданные контроллеры описаны в нижеследующих разделах.

3.5.1 HomeController

Контроллер Home создается в проектах ASP.NET Core по умолчанию, и при запуске веб-приложения вызывается метод Index именно этого контроллера, если оставить маршруты приложения в исходном состоянии без изменений.

В данном веб-приложении было решено оставить данный контроллер и передать ему логику проверки авторизации пользователя, а также отображение профиля пользователя.

В методе Index (листинг 3.18) можно видеть, как приложение пытается получить идентификатор текущего пользователя из сессии. Если значения для

ключа «id», не найдено, то пользователь будет перенаправлен на страницу авторизации, иначе в свой профиль.

Листинг 3.18 – Метод Index контроллера Home

```
public IActionResult Index()
{
    Updater.Update();

    if (HttpContext.Session.GetString("id") != null)
         return RedirectToAction("Profile", "Home");
    else
         return RedirectToAction("Index", "Login");
}
```

Также здесь можно видеть вызов метода Update класса Updater. Данный класс имеет два метода:

1. SendNotifications – приватный метод, выполняющий проверку утвержденных отпусков сотрудников системы. Если отпуск сотрудника начнется или закончится через несколько дней, то его руководителям отправляется соответствующее уведомление.
2. Update – метод, проверяющий дату последнего обновления. Если в текущий день обновлений ещё не было, то в таблицу Updates добавляется текущая дата, и начинается выполнение ежедневных задач вроде упомянутой проверки приближения начала или конца отпусков сотрудников.

Метод Profile контроллера Home отвечает за подбор сотруднику подходящей страницы профиля. Если пользователь не авторизован в системе, то он перенаправляется на страницу авторизации; если пользователь – администратор системы, то он будет направлен в панель администратора. Если авторизованный пользователь – руководитель некоторого подразделения, то в сессии сохраняется соответствующая пометка, в зависимости от значения которой пользователю отображаются страницы приложения.

Кроме того, если пользователь является сотрудником университета, то в данном методе выполняется получение всех уведомлений данного пользователя, которые будут выведены в виде списка на главной странице его профиля.

3.5.2 AdminController

Контроллер Admin отвечает за функционал панели администратора. Методы данного контроллера кратко описаны в таблице 3.3.

Таблица 3.3 – Методы контроллера AdminController

Название метода	Параметры	Назначение
Index	Нет	Отображение главной страницы панели администратора
Departments	Поисковой запрос (наименование подразделений)	Список всех подразделений университета
Department	Идентификатор подразделения	Получение информации об отдельном подразделении
Employees	Поисковой запрос (ФИО сотрудников), идентификатор подразделения	Список всех сотрудников университета либо подразделения
Employee	Идентификатор сотрудника	Получение информации об отдельном сотруднике
CheckAdminPermission	Нет	Выполняет проверку прав доступа к панели администратора

Для администратора системы не предусмотрен никакой функционал по изменению данных: вместо этого ему отводится наблюдательная роль для отслеживания корректного размещения сотрудников и подразделений университета во взаимной иерархии. По этой причине методы данного контроллера достаточно простые и выполняют только запросы на получение данных, как можно видеть на примере метода Employee в листинге 3.19.

Листинг 3.19 – Метод Employee контроллера Admin

```
public IActionResult Employee(string id)
{
    // проверка авторизации
    if (!CheckAdminPermission())
        return RedirectToAction("Index", "Login");

    // попробовать найти сотрудника с указанным идентификатором
    Employee emp = Connector.GetEmployee(id);

    if (emp == null)
```

```
            {
                ViewBag.Error = "Не удалось получить данные о
сотруднике";
                return View();
            }

            // объект модели представления с данными о сотруднике
            EmployeeViewModel employee =
EmployeeHelper.ConvertEmployeeToViewModel(emp);

            return View(employee);
        }
```

3.5.3 LoginController

Данный контроллер отвечает за выполнение авторизации пользователей в системе и состоит из трех методов: Index (с методами действий HttpGet и HttpPost) и Logout. Здесь стоит отметить, что в рабочем состоянии система предполагает использование систему авторизации ТПУ, поэтому на этапе разработки приложения авторизация выполнена в виде прототипа.

Тем не менее, в этом контроллере была реализована логика проверки типа авторизуемого пользователя. Администраторы, сотрудники и руководители пользуются одной формой для входа, поэтому необходимо определить, какую роль имеет пользователь, выполняющий вход в систему.

В первую очередь, метод контроллера пытается получить пользователя по его логину. Здесь используется отдельный класс LoginHandler с единственным методом GetUser. Сначала этот метод обращается к внутренней БД приложения и пытается найти там администратора с введенным логином. Если такого администратора нет, то выполняется попытка получить из API сотрудника с таким логином. Если и эта попытка оканчивается неудачно, то пользователю в представлении Index контроллера Login будет передано соответствующее сообщение, как показано в листинге 3.20.

Листинг 3.20 – Передача сообщения об ошибке при попытке авторизации

```
Object user = LoginHandler.GetUser(login);

// нет пользователя с таким логином
```

```
if (user == null)
{
    TempData["Error"] = "Пользователь с таким логином не найден";
    return View();
}
```

Если полученный пользователь не равен NULL, то он может быть сотрудником или администратором, для чего выполняется проверка типа объекта. В зависимости от результата проверки пользователь будет перенаправлен на панель администратора либо страницу профиля, как можно видеть в листинге 3.22.

Листинг 3.21 – Перенаправление пользователя в зависимости от его роли

```
if (user.GetType() == typeof(Administrator))
{
    …
    // тип пользователя
    HttpContext.Session.SetString("user_type", "administrator");
    // идентификатор пользователя
    HttpContext.Session.SetString("id", admin.Id);

    return RedirectToAction("Index", "Admin");
}
else if (user.GetType() == typeof(Employee))
{
    …

    // тип пользователя
    HttpContext.Session.SetString("user_type", "employee");
    // идентификатор пользователя
    HttpContext.Session.SetString("id", employee.Id);

    return RedirectToAction("Profile", "Home");
}
```

3.5.4 HeadController

Данный контроллер отвечает за ряд простых функций для руководителя, в целом похожих на функционал панели администратора, но распространяющийся только на те подразделения, которыми управляет данный руководитель.

Список методов контроллера с кратким описанием приведен в таблице 3.4.

Таблица 3.4 – Методы контроллера Head

Название метода	Параметры	Назначение
Departments	Нет	Получение списка подразделений, которыми управляет авторизованный руководитель
Department	Идентификатор подразделения	Получение информации об отдельном подразделении
Employees	Идентификатор подразделения, идентификатор должности, поисковой запрос (ФИО)	Получение списка сотрудников из подразделения, подчиненного авторизованному руководителю
Employee	Идентификатор сотрудника	Получение информации об отдельном сотруднике
FilterByPosition	Список сотрудников, идентификаторы подразделения и должности	Фильтрация списка сотрудников по их должности в подразделении

Как и в случае панели администратора, руководители подразделений могут только просматривать списки сотрудников и подразделений без возможности вносить в эту информацию какие-либо изменения, так как источником этих данных должен являться сервер университета. С другой стороны, в отличие от администратора, у руководителей на страницах с информацией об отдельных сотрудниках и подразделениях имеются кнопки, позволяющие управлять отпусками для этих сотрудников и подразделений, как можно будет видеть далее в разделе, посвященном представлениям веб-приложения.

3.5.5 GroupsController

Контроллер Groups отвечает за возможность создания и управления группами сотрудников. Согласно требованиям к системе, в ней должно быть возможно назначать правила выбора отпусков для произвольной совокупности сотрудников, для чего был реализован специальный тип правил модели EmployeeRule. Однако в некоторых случаях может быть удобнее сгруппировать сотрудников по некоторому признаку и сохранить в виде одной группы: таким образом, при наличии информативного имени и описания группы просмотр правил выбора отпусков может быть более наглядным, чем в случае простого списка из имен выбранных сотрудников.

Описание методов контроллера Group приведено в таблице 3.5.

Таблица 3.5 – Методы контроллера Groups

Название метода	HTTP-метод	Параметры	Назначение
Index	GET	Идентификатор подразделения, поисковой запрос	Получение списка групп, созданных руководителем
ViewGroup	GET	Идентификатор группы	Получение данных об отдельной группе сотрудников
AddGroup	GET	Нет	Отображение страницы для создания группы
AddGroup	POST	Наименование, описание группы, идентификатор подразделения, идентификаторы пользователей в группе	Сохранение в базе данных созданной руководителем новой группы
DeleteGroup	GET	Идентификатор группы	Удаление группы сотрудников
EditGroup	GET	Идентификатор группы	Отображение страницы для редактирования группы
EditGroup	POST	Идентификатор группы, новые наименование, описание и список	Сохранение в БД изменений, внесенных в уже существующую группу сотрудников

		идентификаторов сотрудников	

3.5.6 RulesController

Данный контроллер является крайне важным для функционала веб-приложения, так как он отвечает за создание правил выбора отпусков, задаваемых руководителем.

Суть правил состоит в следующем: у каждого подразделения есть специфика собственной работы, связанная с сотрудниками, находящихся на рабочем месте. Иногда возможны ситуации, когда необходимо, чтобы все сотрудники подразделения работали (например, сотрудники приемной комиссии в период поступления); иногда, наоборот, у подразделения может быть относительно свободное время, в течение которого сотрудники могут уйти в отпуск без вреда для работы отделения (например, летом для преподавателей). Такие правила при расстановке отпусков обычно приходится отслеживать самостоятельно, тогда как при использовании данной разработки возможно внести их в систему, и в случае нарушения какого-либо правила приложение само сообщит руководителю об этом.

В системе предусмотрено три типа правил (для сотрудников, для должностей, для групп сотрудников), и для всех них был подготовлен схожий по функционалу набор методов, позволяющих просматривать (View), добавлять (Add), редактировать (Edit) и удалять (Delete) правила. Кроме того, был создан общий метод Index, возвращающий пользователю список, состоящий из всех правил, установленных для некоторого подразделения.

Рассмотрим работу метода Index. Без учета различных проверок на нулевые значения, алгоритм подготовки списка правил выглядит, как показано в листинге 3.22.

Листинг 3.22 – Получение списка правил выбора отпусков

```
public IActionResult Index(string depId)
{
    // идентификатор авторизованного руководителя
```

```
        string headId = HttpContext.Session.GetString("id");

        // список правил текущего руководителя
        List<RuleViewModel> rules = RuleHelper
             .GetRulesList(headId, depId);

        // список подразделений текущего руководителя
        List<Department> departments = Connector
             .GetSubordinateDepartments(headId);

        ViewBag.departments = departments;

        return View(rules.OrderByDescending(r => r.Date).ToList());
    }
```

В начале метода выполняется получение идентификаторов подразделения и текущего авторизованного руководителя. Эти данные нужны для получения списка правил в формате RuleViewModel – унифицированной модели представления, позволяющей записать в себя данные о правилах любого из трех типов (листинг 3.23).

Листинг 3.23 – Модель представления RuleViewModel для отображения правил выбора отпусков

```
public class RuleViewModel
{
    public RuleViewModel() { }
    // Идентификатор правила в БД
    public int Id { get; set; }
    // Наименование правила
    public string Name { get; set; }
    // Описание правила
    public string Description { get; set; }
    // Тип правила
    public string RuleType { get; set; }
    // Дата создания правила
    public DateTime Date { get; set; }
    // Период года, в который действует правило
    public string Period { get; set; }
    // Подразделение, в котором действует правило
    public Department Department { get; set; }
    // Описание правила на основе его типа и содержания
    public string SystemDescription { get; set; }
}
```

Список из объектов данной модели представления можно получить с помощью статического класса RuleHelper, содержащего различные методы для

работы с правилами выбора отпусков. Так, метод GetRulesList (листинг 3.24) проходит по всем типам правил, находит созданные авторизованным руководителем правила и конвертирует их в требуемый формат.

Листинг 3.24 – Метод GetRulesList

```
static public List<RuleViewModel> GetRulesList(string headId,
string depId)
{
    List<RuleViewModel> result = new List<RuleViewModel>();

    // добавить все правила в список
    result.AddRange(ConvertEmployeeRulesToVmList(headId, depId));
    result.AddRange(ConvertPositionRulesToVmList(headId, depId));
    result.AddRange(ConvertGroupRulesToVmList(headId, depId));

    return result;
}
```

3.5.7 VacationDaysController

Данный контроллер отвечает за управление отпускными днями сотрудников и включает в себя три метода.

Первый метод – List. Этот метод предназначен для вывода информации об имеющихся у авторизованного сотрудника отпускных днях (соответственно, доступ к этой функции имеют не только руководители подразделений, но и рядовые сотрудники).

При своей работе этот метод получает из базы данных все отпускные дни сотрудника, назначенные на текущий и предыдущий год, и вычисляет общую сумму дней по каждому основанию выдачи отпуска (основной оплачиваемый отпуск, дополнительный оплачиваемый отпуск и т.д.). В приложении Б представлен код метода GetDaysInfo (из статического класса VacationDayHelper), отвечающего за заполнение словаря, состоящего из пар значений «тип отпуска – количество назначенных дней».

Отдельно стоит отметить ещё один метод класса VacationDayHelper – AddMainVacationDays(). Суть этого метода состоит в выполнении статьи 115 ТК РФ, согласно которой у всех сотрудников есть право на основной оплачиваемый

отпуск длительностью 28 дней. Соответственно, перед любыми действиями, связанными с отпусками или отпускными днями, выполняется следующий код:

```
VacationDayHelper.AddMainVacationDays(id);
```

Благодаря данному методу, любому сотруднику всегда присваивается 28 дней положенного отпуска, если у него их ещё нет. Это необходимо, так как:

1. Позволяет сократить работу руководителю, ведь ему не нужно назначать всем своим сотрудникам 28 дней основного оплачиваемого отпуска вручную.
2. Выполняет заполнение изначально пустой таблицы с отпускными днями сотрудников.
3. Добавляет дополнительный уровень защиты на случай, если по невнимательности руководителя или по другой причине законные 28 дней для сотрудника будут удалены из таблицы БД.

Другие два метода контроллера, SetDays с HTTP-методами GET и POST, отвечают за отображение формы редактирования отпускных дней выбранного сотрудника и сохранение внесенных изменений в базу данных.

3.5.8 VacationController

Контроллер Vacation предназначен для управления отпусками как сотрудниками, так и руководителями подразделений. Здесь стоит отметить, что, согласно логике системы, отпуска могут иметь два состояния, которые физически представлены разными таблицами БД:

1. Запланированные или желаемые отпуска – это периоды отпуска, выбранные самими сотрудниками и ещё не утвержденные руководителем. Такие отпуска могут свободно редактироваться как самим сотрудником, так и его руководителями, однако они должны соблюдать требования ТК РФ. В БД представлены таблицей wished_vacation_periods и связанной с ней vacation_parts.

2. Утвержденные отпуска – отпуска, которые уже прошли процедуру проверки на соответствие внутренним правилам системы. Могут быть отредактированы, прерваны либо отменены руководителем сотрудника; сам сотрудник больше не может вносить изменения в такой отпуск (но может оставить заявку на это действие). В БД представлены таблицей set_vacations.

Описание методов контроллера Vacation представлено в таблице 3.6.

Таблица 3.6 – Методы контроллера Vacation

Название метода	HTTP-метод	Параметры	Назначение
Index	GET	Идентификатор сотрудника, тип отпуска, год	Получение списка отпусков, выбранных либо назначенных сотруднику на указанный год
AddVacation	GET	Идентификатор сотрудника, год	Отображение страницы для выбора желаемых периодов отпуска
AddVacation	POST	Идентификатор сотрудника, массивы начальных и конечных дат периодов отпуска	Сохранение в базе данных желаемого периода отпуска с выбранными периодами
EditWished-Vacation	GET	Идентификаторы сотрудника и отпуска	Отображение страницы редактирования желаемого отпуска
EditWished-Vacation	POST	Идентификаторы сотрудника и отпуска, массивы начальных и конечных дат периодов отпуска	Сохранение в БД изменений, внесенных в желаемый отпуск сотрудника
DeleteWished-Vacation	GET	Идентификатор отпуска	Удаление запланированного отпуска
Interrupt	GET	Идентификатор отпуска	Отображение страницы для выбора даты прерывания отпуска
Interrupt	POST	Идентификатор отпуска, дата прерывания	Прерывание утвержденного отпуска по указанной дате
Cancel	GET	Идентификатор отпуска	Отмена указанного утвержденного отпуска

EditSetVacation	GET	Идентификаторы сотрудника и отпуска	Отображение страницы редактирования утвержденного отпуска руководителем
EditSetVacation	POST	Идентификаторы сотрудника и отпуска, массивы начальных и конечных дат периодов отпуска	Сохранение в БД изменений, внесенных в уже утвержденный отпуск
VacationRequest	GET	Идентификаторы сотрудника и отпуска	Отображение страницы для написания заявки на изменение утвержденного отпуска
VacationRequest	POST	Идентификаторы сотрудника, отпуска, руководителя, текст сообщения	Сохранение в БД (отправка руководителю) заявки на изменение отпуска

Важным моментом является разносторонняя проверка выбранных периодов отпусков перед их сохранением при добавлении либо редактировании отпуска. Для этого был создан класс VacationChecker, который проверяет все отпуска сотрудника в течение года (как запланированные, так и уже утвержденные), чтобы их периоды имели правильный порядок дат начала и конца, не включались друг в друга, не пересекались, не были пустыми. Также проверяется соблюдение статьи 125 ТК РФ, согласно которой как минимум один период отпуска должен длиться непрерывно 14 дней.

3.5.9 CalendarController

Данный контроллер содержит методы для отображения важнейшей части веб-приложения – календаря отпусков подразделения. Кроме того, посредством методов именно этого контроллера выполняется как проверка отпусков сотрудников на соответствие внутренним правилам подразделения, так и утверждение отпусков после проведения этой проверки.

Метод Department контроллера выполняет все необходимые действия для отображения календаря. Он формируется на основе сразу нескольких моделей представления, описание которых приведено в таблице 3.7.

Таблица 3.7 – Модели представления для формирования календаря отпусков подразделения

Модель представления	Назначение
CalendarViewModel	Представляет собой весь календарь в целом и потому хранит общие данные: подразделение календаря, выбранные пользователем фильтры, начальная и конечная дата отображения в календаре, список данных об отпусках всех сотрудников в календаре.
EmpVacationViewModel	Модель представления, отвечающая за хранение данных об отпусках одного сотрудника, поэтому в число полей входят ФИО сотрудника и данные о днях отпусков.
EmpVacationPeriodViewModel	Модель представления календаря самого низкого уровня; один объект этого класса хранит данные о только одном дне из календаря для указанного сотрудника: входит ли данный день в отпуск сотрудника; если да, то отпуск уже утвержден или только запланирован; если нет, является этот день будним, выходным или праздничным. Каждому сотруднику сопоставлен список таких объектов, количество которых равно по умолчанию числу дней в выбранном году.

Метод CheckVacation выполняет проверку запланированных отпусков сотрудников подразделения на соответствие правилам этого подразделения, при этом все нарушения сохраняются в специальный список с объектами-предупреждениями, который выводится руководителю на экран при их наличии (листинг 3.25).

Листинг 3.25 – Получение предупреждений о нарушениях правил

```
// проверка правил для сотрудников
List<RuleWarning> warnings = EmployeeRulesChecker
.CheckEmployeeRules(tempEmployees, headId, depId);

// проверка правил для должностей
warnings.AddRange(PositionRulesChecker
```

```
.CheckPositionRules(tempEmployees, headId, depId));

// проверка правил для групп
warnings.AddRange(GroupRulesChecker
.CheckGroupRules(tempEmployees, headId, depId));
```

При проверке правил для сотрудников система попарно перебирает всех сотрудников из правила и ищет пересечения: если сотрудники должны уходить в отпуск одновременно, то все их периоды должны полностью совпадать (либо, если у сотрудников разное количество дней отпуска, отпуска сотрудников с меньшим числом дней должны быть включены в периоды отпуска сотрудника с большим числом); если они не должны уходить в отпуск одновременно, то нарушение произойдет, даже если хотя бы один день отпусков у двух затронутых сотрудников пересекается. Аналогично выполняются проверки для групп сотрудников.

При проверке правил для должностей сотрудники подразделения фильтруются по этой должности. Затем анализируется каждый день периода, в течение которого действует правило, и вычисляется число сотрудников указанной должности, которые находятся в отпуске в этот день. Если это число для какого-либо дня превышает порог (указанный в правиле), то обнаруживается нарушение.

Ещё один метод контроллера – SetVacation. Это простой метод, который преобразует объекты запланированных отпусков сотрудников в объекты SetVacation (утвержденные отпуска) и сохраняет их в базе данных.

3.6 Представления веб-приложения

Для данного проекта был создан ряд представлений, которые можно видеть на рисунке 3.8.

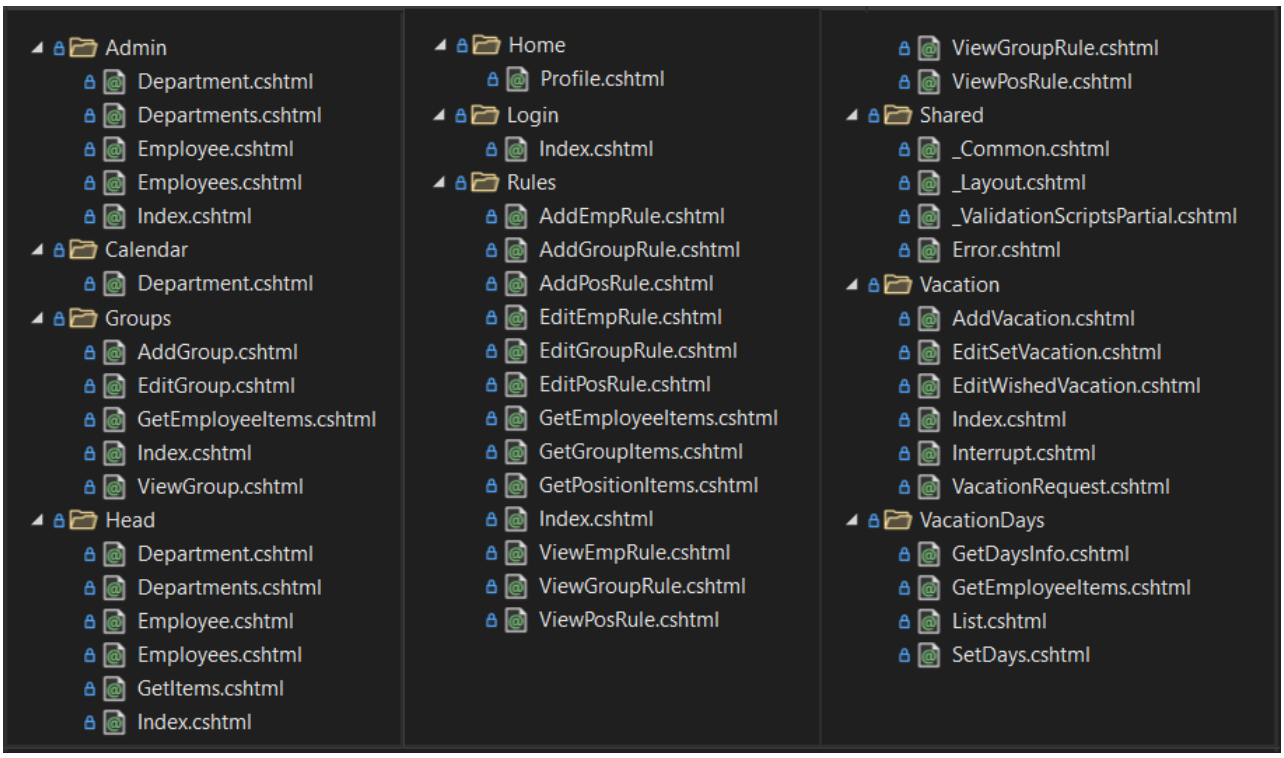


Рисунок 3.8 – Представления проекта

В первую очередь, были внесены изменения в мастер-страницу _Layout: был удален ряд элементов оформления, созданных автоматически, а хэдер в верхней части страницы был отредактирован таким образом, чтобы в нем отображался логотип ТПУ.

Далее была создана дополнительная мастер-страница, использующая шаблон _Layout: _Common. Код данной страницы приведен в приложении В, и он включает в себя:

1. Получение из сессии типа текущего пользователя для определения, какие функции должны быть доступны пользователю.
2. Обработка сообщений (message), ошибок (error) и сообщений об успешности операций (success), сохраняемые контроллерами в словарь TempData.
3. Блок всплывающих сообщений, которые становятся видны только после срабатывания специального скрипта.
4. Выбор вида меню в зависимости от типа текущего пользователя.
5. Вывод сообщений, полученных из TempData.

Таким образом, в эту мастер-страницу были вынесены те части пользовательского интерфейса, которые должны быть общими для всех страниц веб-приложения, что позволило сократить дублирование кода и упростить возможность его редактирования.

В следующих разделах представлен функционал веб-приложения и вид пользовательского интерфейса для различных ролей пользователей.

3.6.1 Администратор системы

При открытии веб-приложения (вне зависимости от роли пользователя) отображается страница авторизации, представленная на рисунке 3.9, так как пользователь ещё не вошел в систему. Так как система ещё не была введена в реальное использование с подключением к внутренней авторизации ТПУ, вход выполняется только по логину, представляющему собой идентификатор пользователя.

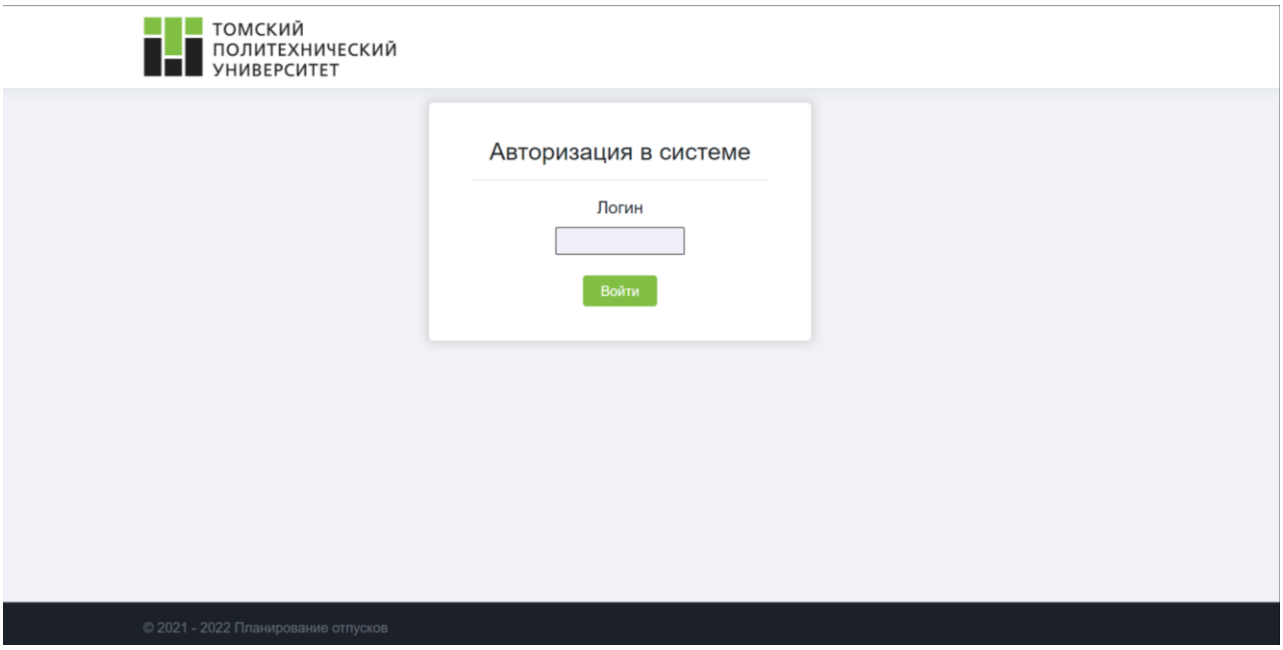


Рисунок 3.9 – Страница авторизации пользователя

Вид главной страницы панели администратора показан на рисунке 3.10. Здесь у администратора системы есть возможность перейти к просмотру списков всех подразделений и всех сотрудников университета. Аналогичные ссылки представлены в боковом меню.

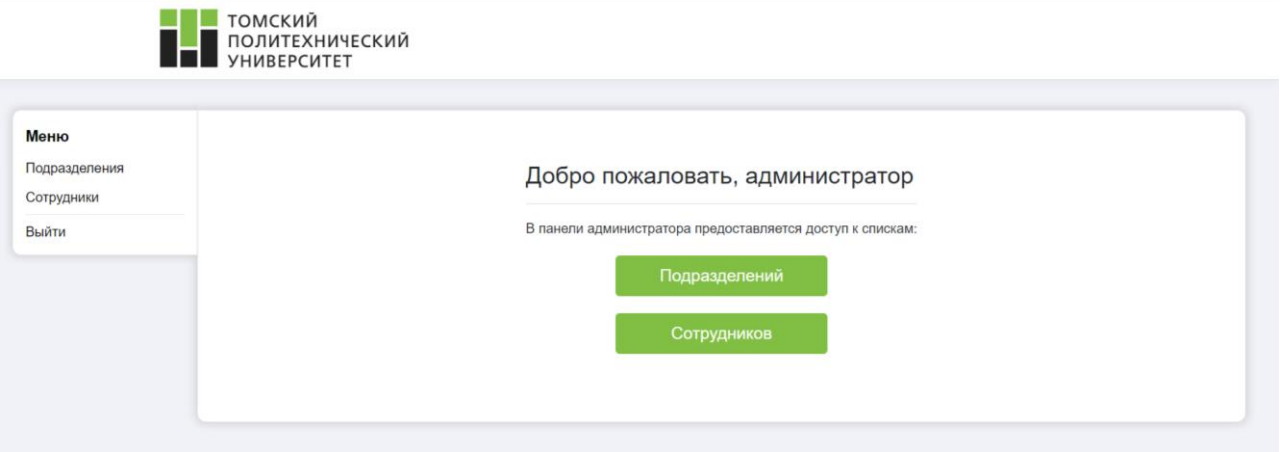


Рисунок 3.10 – Панель администратора

На рисунке 3.11 можно видеть список подразделений университета, отсортированный по наименованиям.

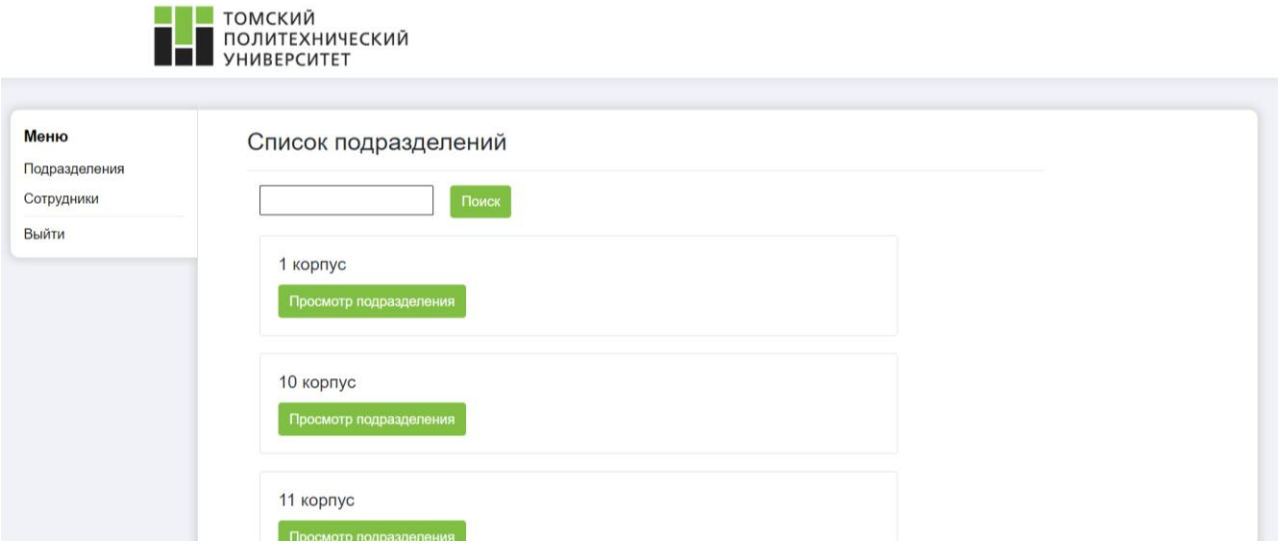


Рисунок 3.11 – Список подразделений университета

В поиске также возможно ввести произвольную часть имени подразделения для облегчения нахождения его в общем списке, как показано на рисунке 3.12.

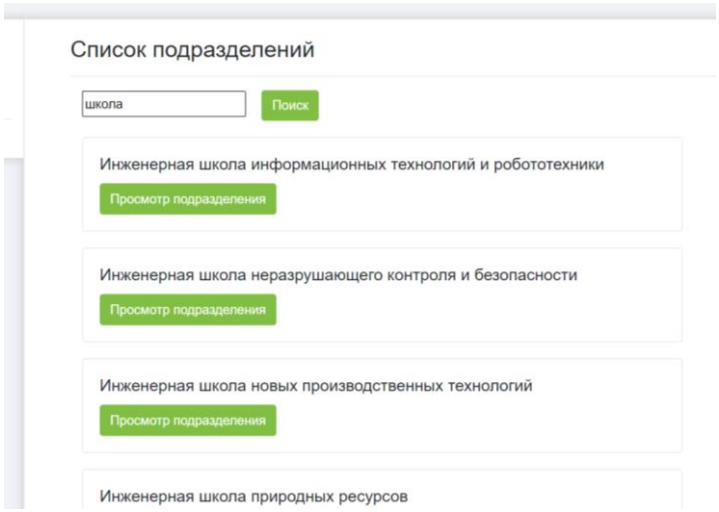


Рисунок 3.12 – Получение списка школ ТПУ по поисковому запросу «школа»

При выборе подразделения можно узнать его старшее подразделение, список младших подразделений (если есть), а также перейти к списку сотрудников данного подразделения (рисунок 3.13).

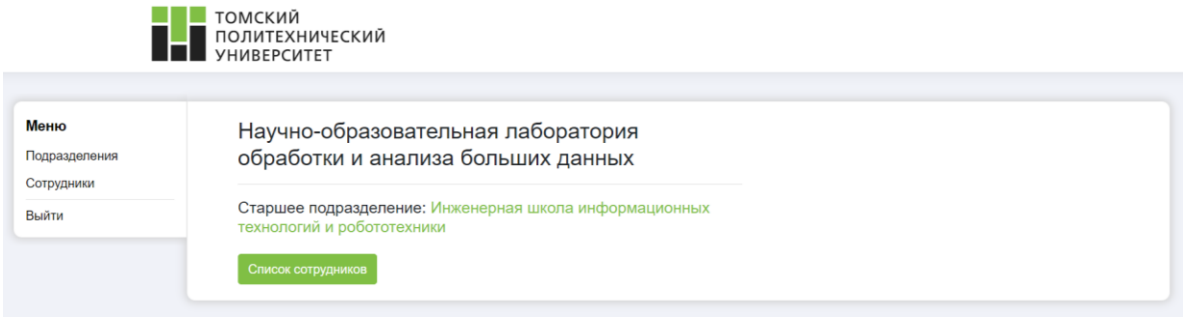


Рисунок 3.13 – Просмотр информации о подразделении

Страница списка сотрудников подразделения показана на рисунке 3.14. Здесь есть поиск по ФИО сотрудников и возможность просмотреть подробную информацию об отдельном сотруднике.

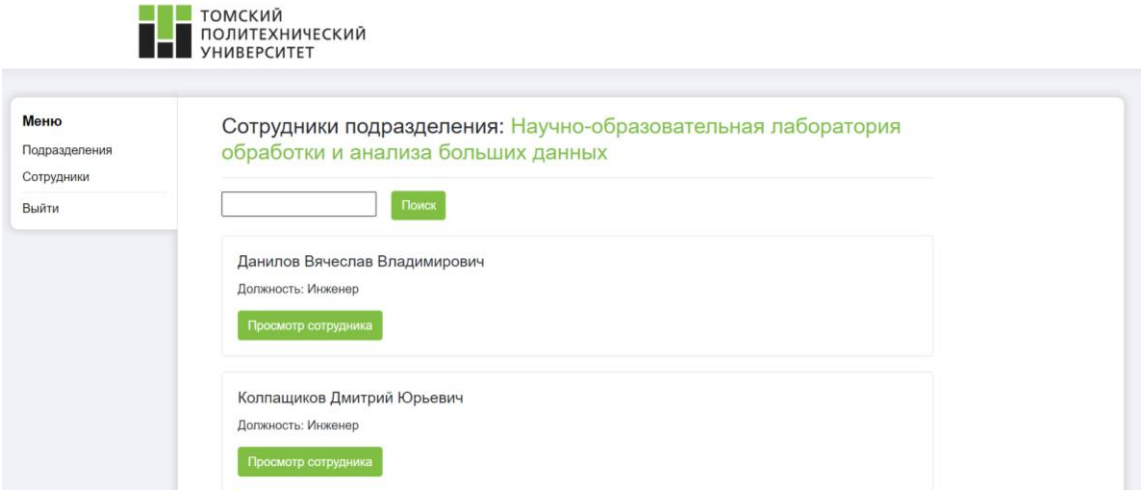


Рисунок 3.14 – Список сотрудников выбранного подразделения

При выборе сотрудника можно узнать, какие должности в каких подразделениях он занимает, а также какими подразделениями данный сотрудник руководит (если является руководителем), как можно видеть на рисунке 3.15.

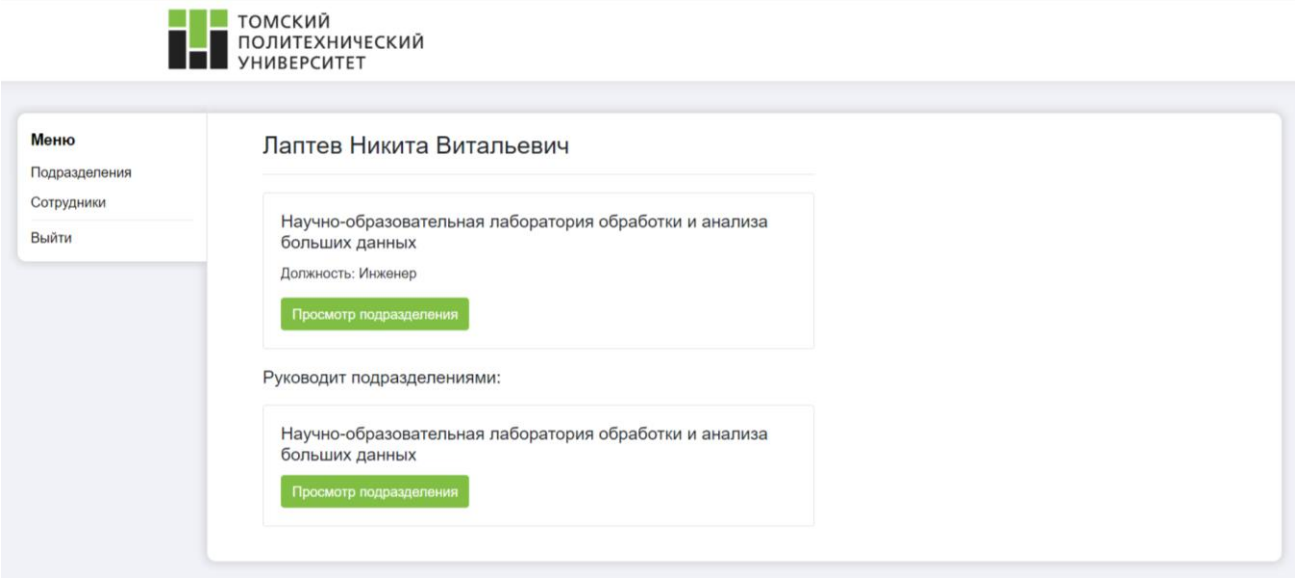


Рисунок 3.15 – Просмотр информации о сотруднике

3.6.2 Рядовой сотрудник подразделения ТПУ

Профиль сотрудника, не являющегося руководителем никакого подразделения, показан на рисунке 3.16. Как можно видеть, в боковом меню у такого пользователя есть другие опции по сравнению с администратором: отпуска, отпускные дни и уведомления.

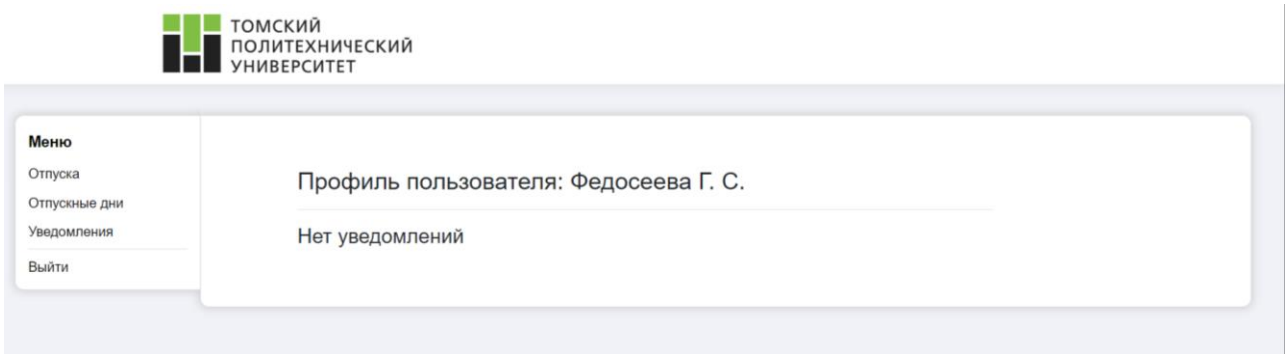


Рисунок 3.16 – Профиль сотрудника подразделения

При переходе по ссылке «Отпускные дни» (рисунок 3.17) сотрудник может узнать, сколько дней для отпуска имеется в его распоряжении, к каким

видам отпуска они относятся и какое число из них доступно (то есть не было передано в какой-либо желаемый или утвержденный отпуск). Кроме того, можно выбирать год, на который дни отпуска были назначены сотруднику, и в таком случае данные на странице автоматически обновятся.

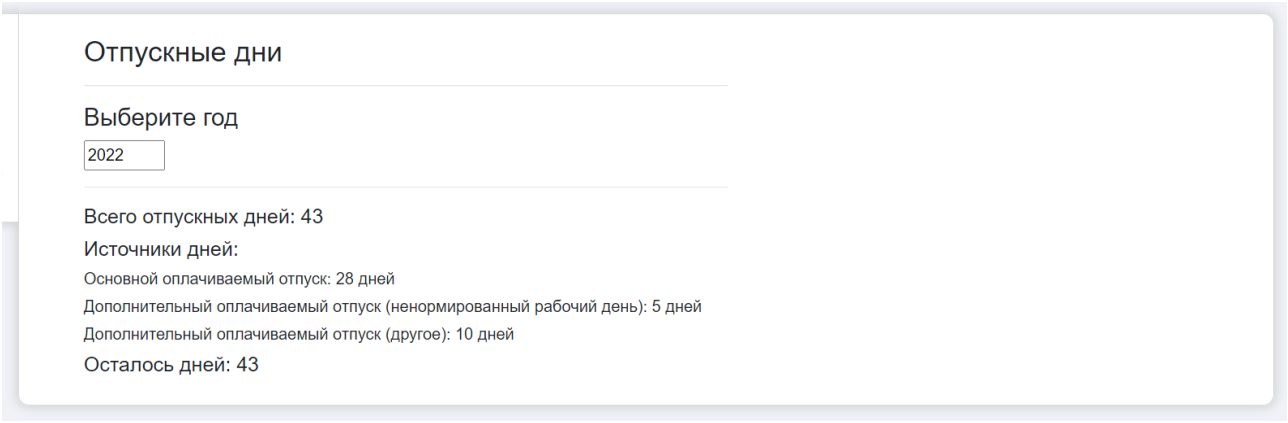


Рисунок 3.17 – Просмотр сотрудником своих отпускных дней

По ссылке «Отпуска» сотруднику открывается страница для выбора желаемого периода отпуска на заданный год, как показано на рисунке 3.18. Здесь сотрудник снова может видеть общее количество доступных дней отпуска на каждый год, а также проводить фильтрацию по видам отпусков (запланированные либо утвержденные).

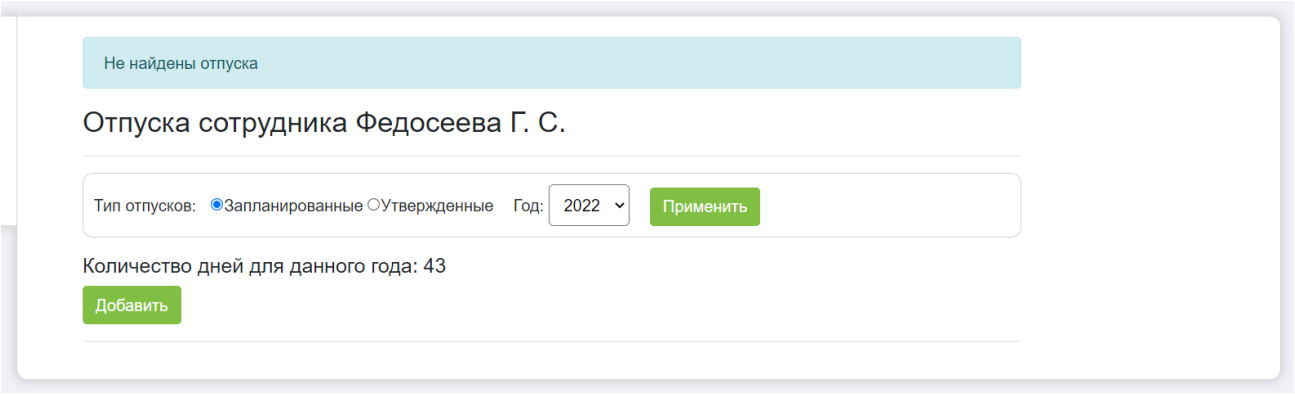


Рисунок 3.18 – Список отпусков сотрудника

При нажатии на кнопку «Добавить» сотрудник получает форму выбора периодов отпуска. Как можно видеть на рисунке 3.19, в данной форме можно выбрать год, на который назначается желаемый отпуск (при этом система автоматически показывает количество отпускных дней, доступных сотруднику для этого года), и периоды отпуска. Если сотрудник желает разбить отпуск на

несколько периодов, то он может добавлять и удалять их посредством соответствующих кнопок.

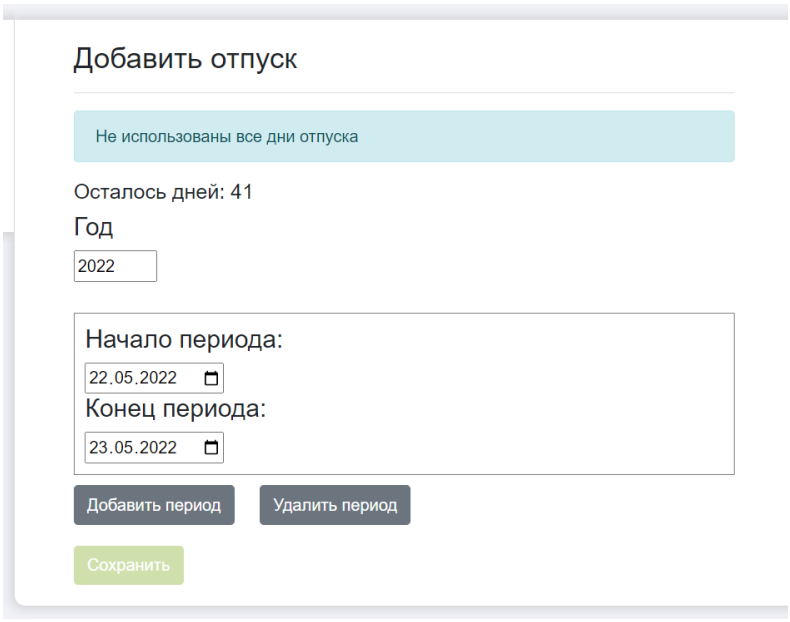


Рисунок 3.19 – Форма выбора желаемого отпуска

При выборе периодов также проверяется корректность их задания: периоды не должны пересекаться, включаться друг в друга, быть пустыми, иметь неправильный порядок дат начала и конца; также выполняется проверка выполнения статьи 125 ТК РФ. При обнаружении каких-либо ошибок или нарушений в верхней части страницы выводятся соответствующие сообщения (рисунок 3.20).

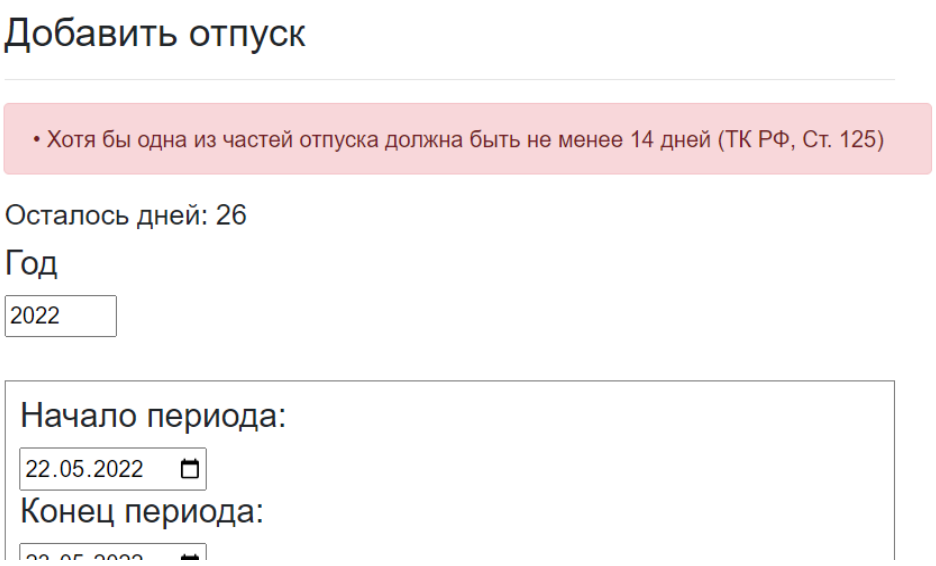


Рисунок 3.20 – Вывод сообщения о нарушении ТК РФ

При успешном прохождении всех проверок желаемый отпуск сохраняется в базе данных, и теперь он отображается в списке отпусков, как показано на рисунке 3.21. Кроме того, здесь также можно видеть, что, если у сотрудника не осталось отпускных дней, то он больше не может добавлять отпуска: для этого была реализована дополнительная проверка.

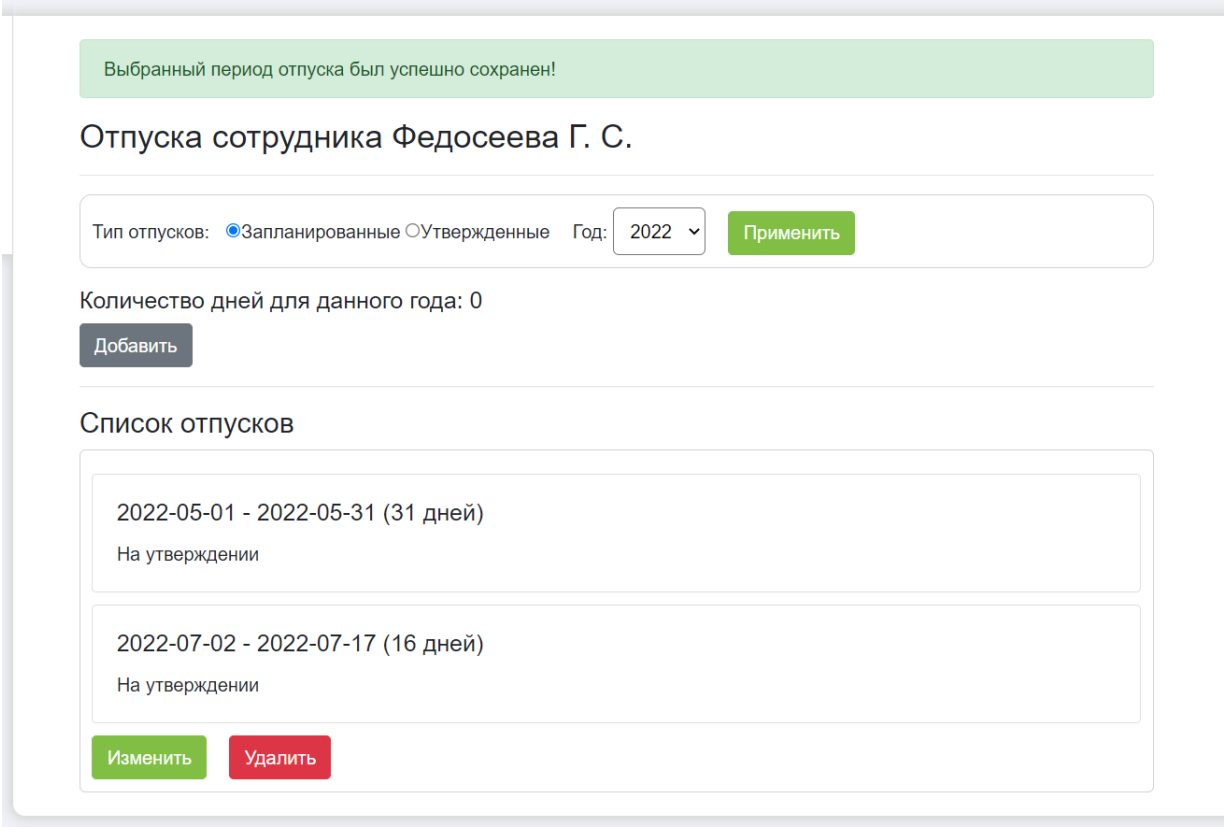


Рисунок 3.21 – Список с добавленным отпуском

При нажатии на кнопку «Изменить» открывается такая же форма, как и для добавления отпусков, но в ней сразу же вводятся выбранные ранее периоды отпусков, которые могут быть отредактированы, как видно на рисунке 3.22. Кнопка «Удалить», соответственно, стирает выбор отпуска пользователя и освобождает затронутые отпускные дни.

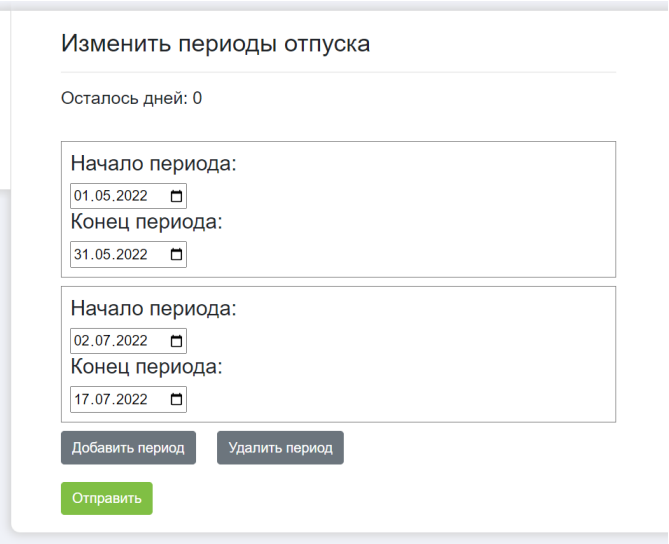


Рисунок 3.22 – Форма редактирования желаемых периодов отпуска

Если желаемые периоды отпусков были утверждены руководителем, то теперь их редактировать нельзя. С другой стороны, сотрудник может оставить заявку на перенос отпуска, для чего добавлена соответствующая кнопка (рисунок 3.23).

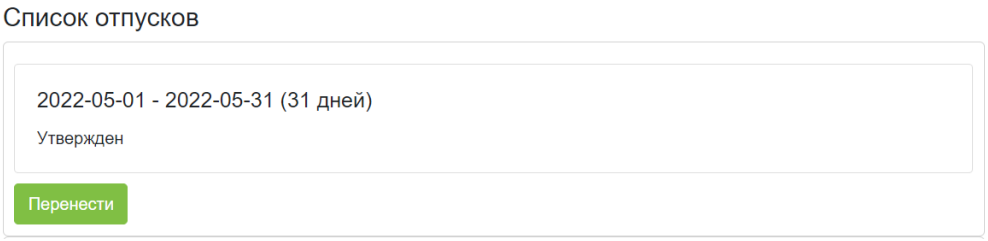


Рисунок 3.23 – Кнопка переноса у утвержденного отпуска

При нажатии на эту кнопку открывается форма, представленная на рисунке 3.24, в которой сотрудник может попросить руководителя как-либо отредактировать уже утвержденный отпуск.

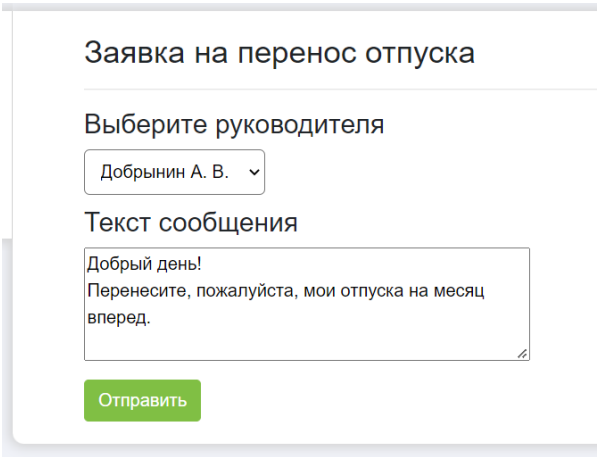


Рисунок 3.24 – Заявка на изменение отпуска

Также в профиле пользователя есть блок уведомлений, в котором сотрудник информируется о любых действиях руководителя над его отпусками, как можно видеть на рисунке 3.25.

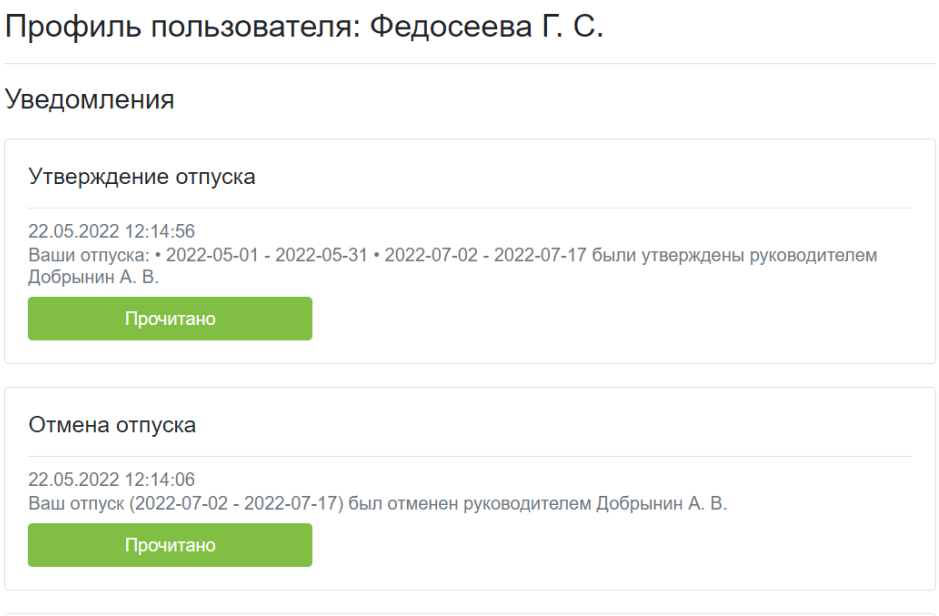


Рисунок 3.25 – Уведомления сотрудника о действиях, связанных с его отпусками

3.6.3 Руководитель подразделения ТПУ

На главной странице профиля руководителя пользователь, как и рядовой сотрудник, может видеть список уведомлений, как показано на рисунке 3.26. Посредством уведомлений руководитель информируется о действиях сотрудников, заявках на изменение утвержденных отпусков, а также о приближающемся начале или окончании отпусков сотрудников (рисунок 3.27). Кроме того, можно видеть, что руководитель имеет в боковом меню как те же опции, что и рядовой сотрудник (так как технически он также является сотрудником своих подразделений), так и новые, предоставляющие функционал по управлению отпусками в подразделениях.

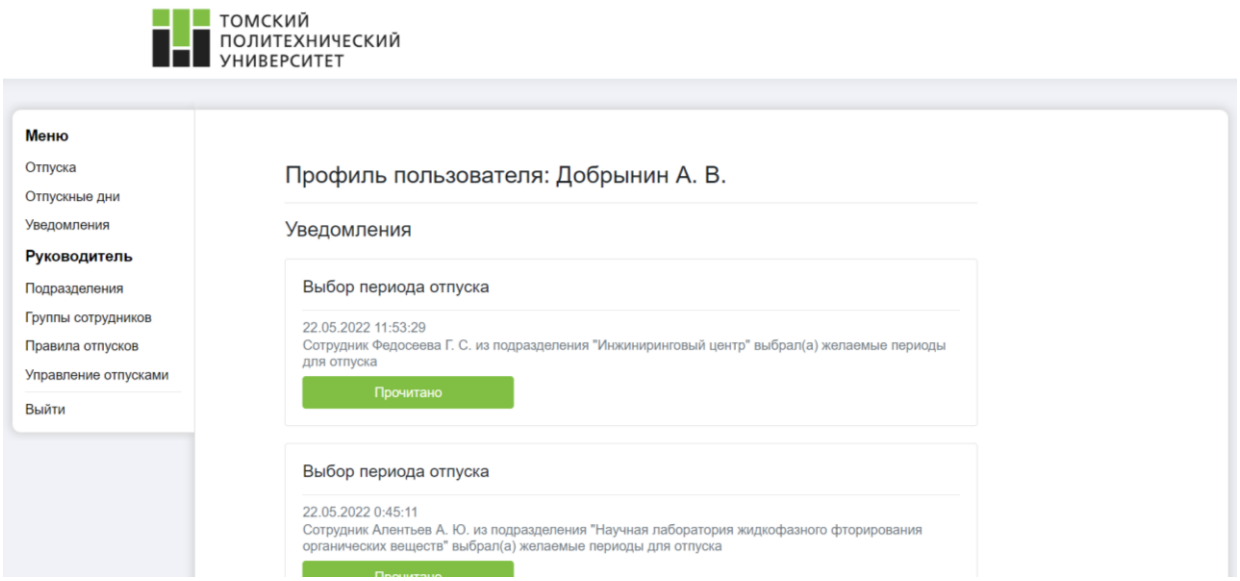


Рисунок 3.26 – Блок уведомлений в профиле руководителя

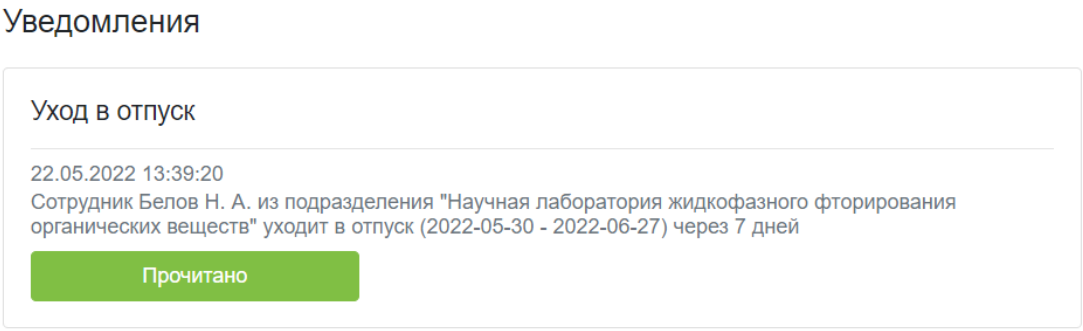


Рисунок 3.27 – Уведомление о приближающемся отпуске сотрудника

Во вкладке «Управление отпусками» расположена страница, позволяющая назначать либо удалять отпускные дни сотрудников подчиненных подразделений, как показано на рисунке 3.28.

В данной форме можно выбрать одно из подчиненных подразделений, при этом автоматически обновится список сотрудников из данного подразделения. При выборе любого сотрудника на странице также отображаются уже имеющиеся у него дни отпуска и их основания (рисунок 3.29). Можно выбрать год, на который назначаются дни отпуска (текущий либо следующий), тип отпуска согласно ТК РФ, количество добавляемых либо отменяемых дней и примечания.

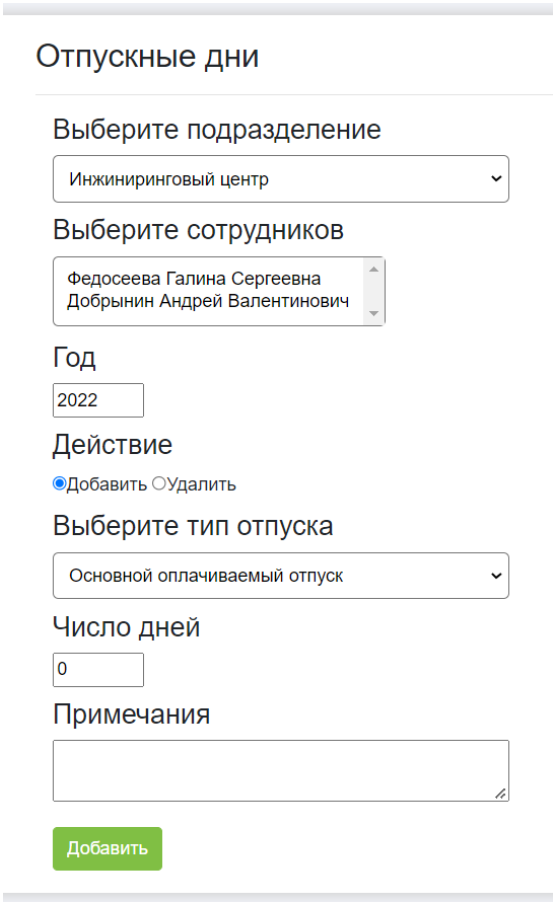


Рисунок 3.28 – Форма для управления отпускными днями сотрудников

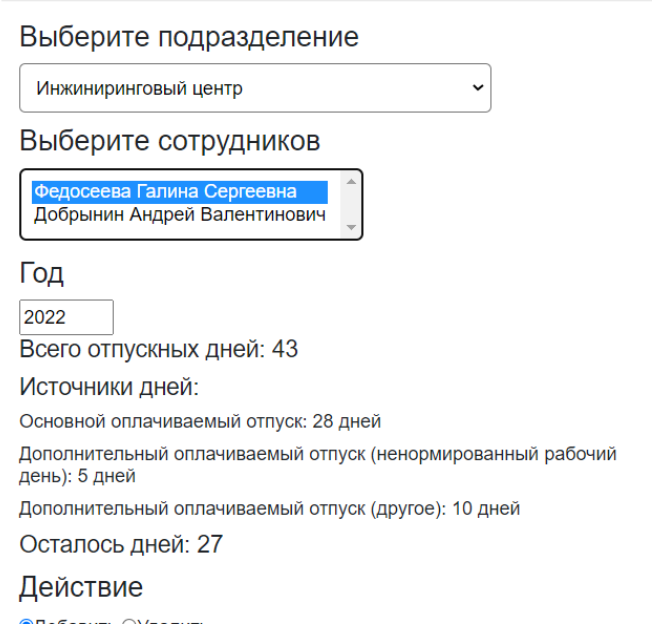


Рисунок 3.29 – Отображение уже имеющихся у сотрудника отпускных дней и их типов

Во вкладке «Группы сотрудников» руководителю отображается страница со списком созданных им групп сотрудников, как показано на рисунке 3.30.

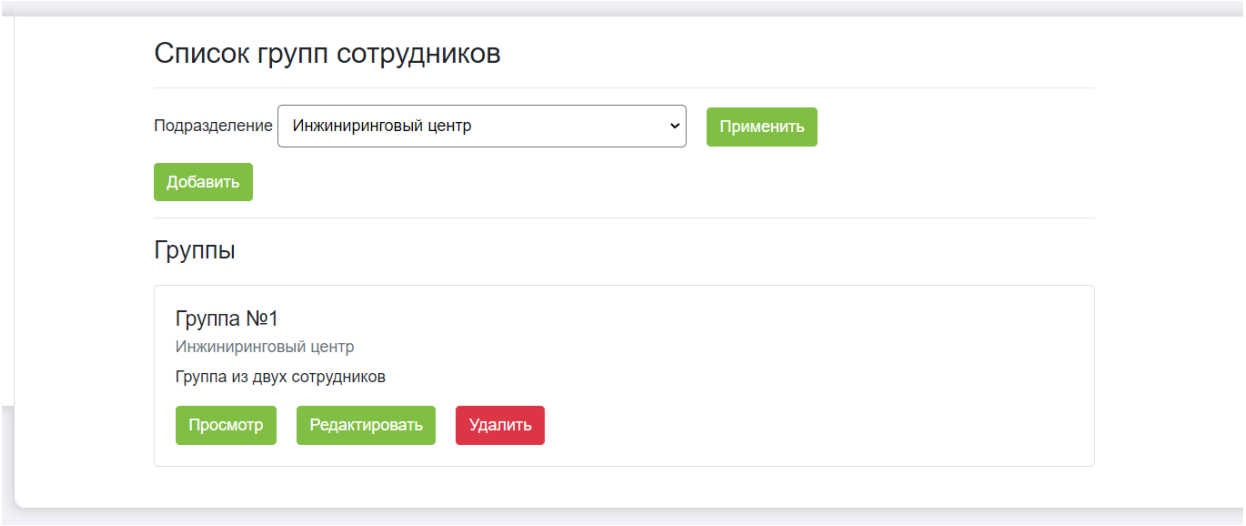


Рисунок 3.30 – Список групп сотрудников подразделения

При нажатии на кнопку «Добавить» можно создать новую группу, для чего в открывающейся форме нужно выбрать подразделение, сотрудников этого подразделения, ввести название и описание группы (рисунок 3.31).

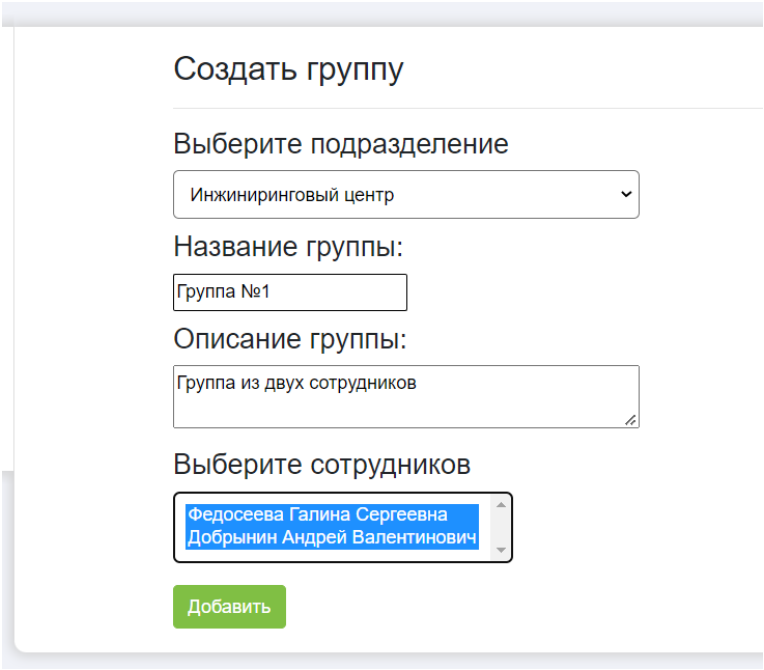


Рисунок 3.31 – Создание группы

Во вкладке «Правила отпусков» можно просмотреть список добавленных для подразделений руководителя внутренних правил выбора отпусков, как показано на рисунке 3.32.

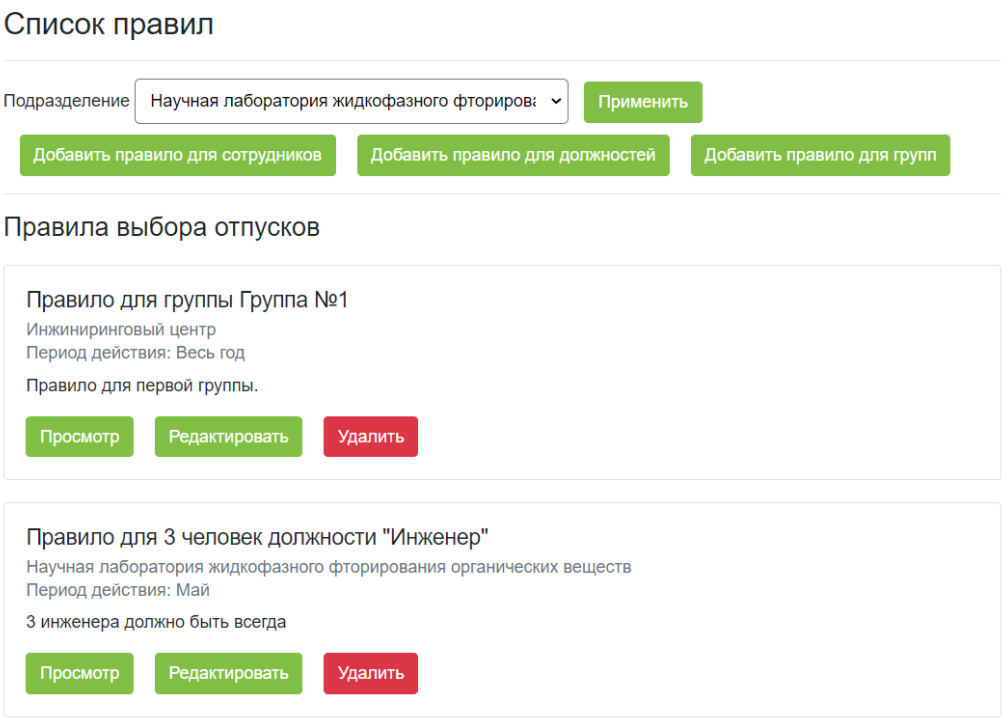


Рисунок 3.32 – Список правил выбора отпусков в подразделениях

При добавлении правила для сотрудников (рисунок 3.33) нужно выбрать подразделение, тип правила (сотрудники должны уходить в отпуск одновременно либо их отпуска не должны пересекаться), описание, период действия и список сотрудников, затронутых правилом.

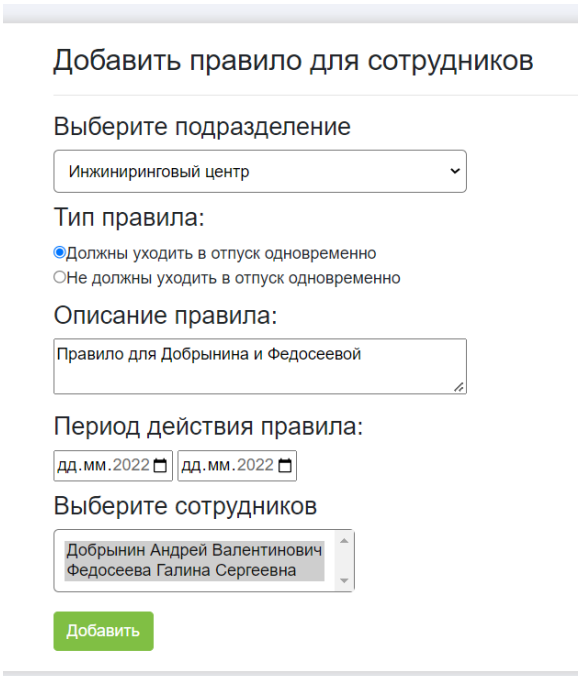


Рисунок 3.33 – Добавление правила для сотрудников

Для правил должностей (рисунок 3.34) тоже нужно выбрать подразделение, период и ввести описание, но вместо сотрудников требуется выбрать должность и ввести число – минимальное число сотрудников данной должности, которые должны быть на рабочем месте.

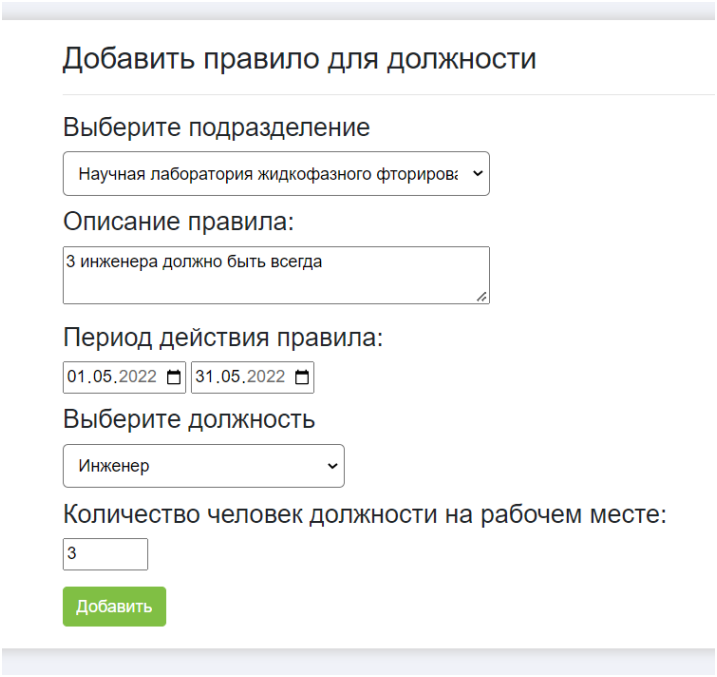


Рисунок 3.34 – Добавление правила для должности

Форма добавления правила для группы аналогична форме для сотрудников, но вместо списка сотрудников требуется выбрать созданную ранее группу в подразделении (рисунок 3.35).

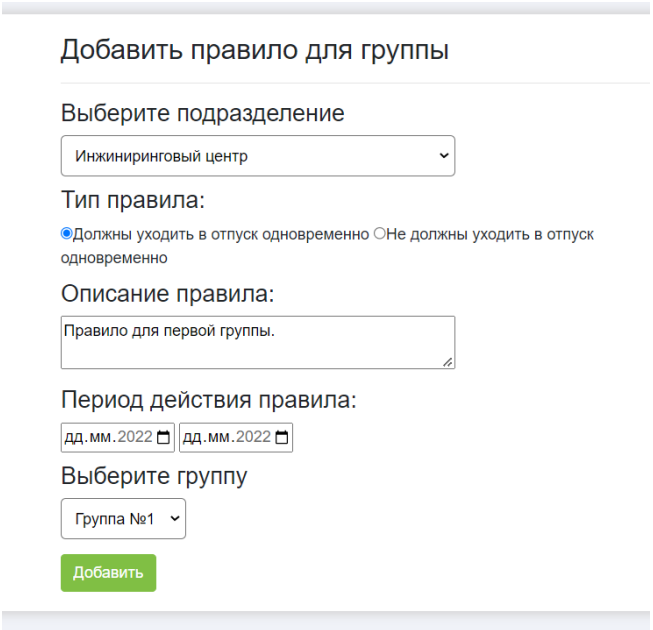


Рисунок 3.35 – Добавление правила для группы

Во вкладке «Подразделения» руководитель может просмотреть список подчиненных подразделений, как показано на рисунке 3.36.

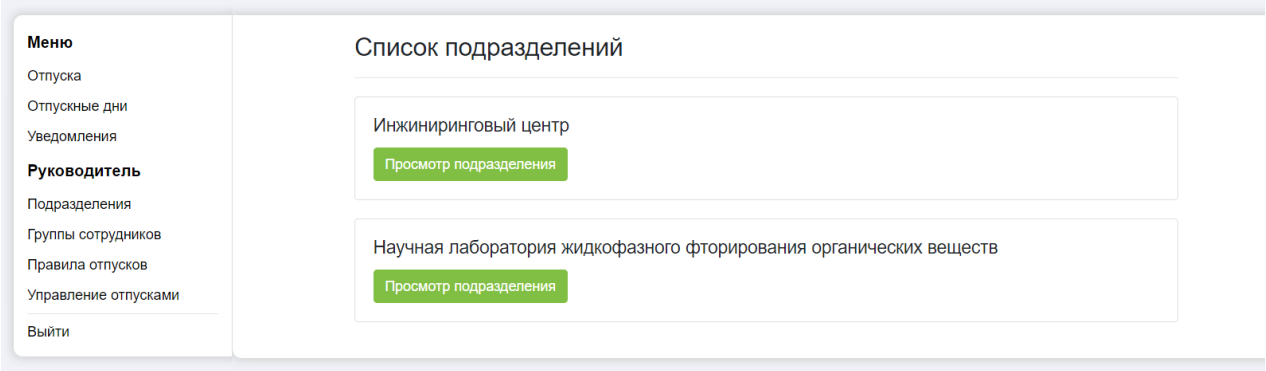


Рисунок 3.36 – Список подчиненных подразделений руководителя

При выборе подразделения руководитель может перейти к списку сотрудников и календарю отпусков (рисунок 3.37).

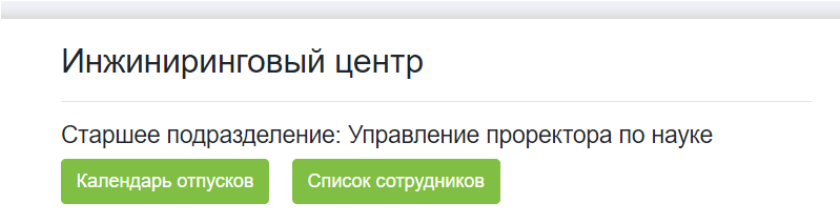


Рисунок 3.37 – Просмотр подразделения

В списке сотрудников возможно выполнять поиск по ФИО и фильтрацию по должности, как показано на рисунке 3.38.

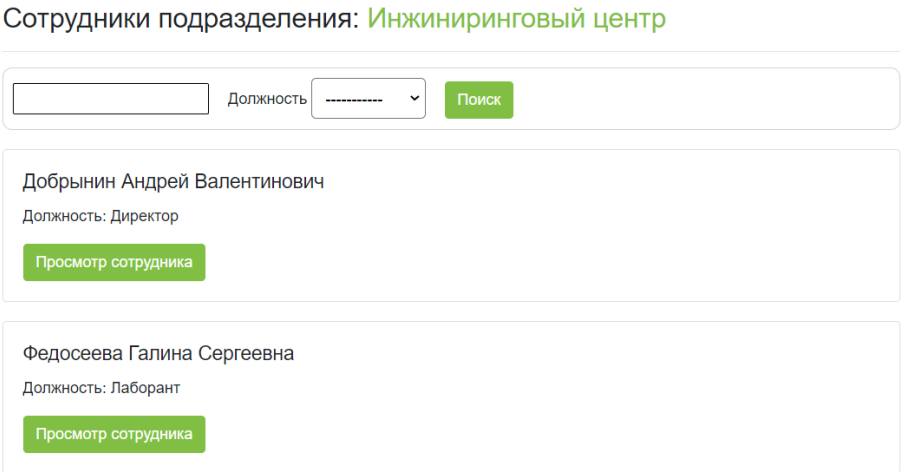


Рисунок 3.38 – Просмотр списка сотрудников из подчиненного подразделения

В свою очередь, календарь отпусков показан на рисунке 3.39. В календаре возможно указывать период, отображаемый в нем (год, начальную и конечную

даты, как можно видеть на рисунке 3.40); тип отпусков (запланированные, утвержденные, либо и те, и другие). Сам календарь представляет собой таблицу, где строки – это сотрудники, а столбцы – даты. Отдельные дни в календаре могут быть выделены разными цветами: будние дни белые, выходные серые, а праздники – блекло-розовые.

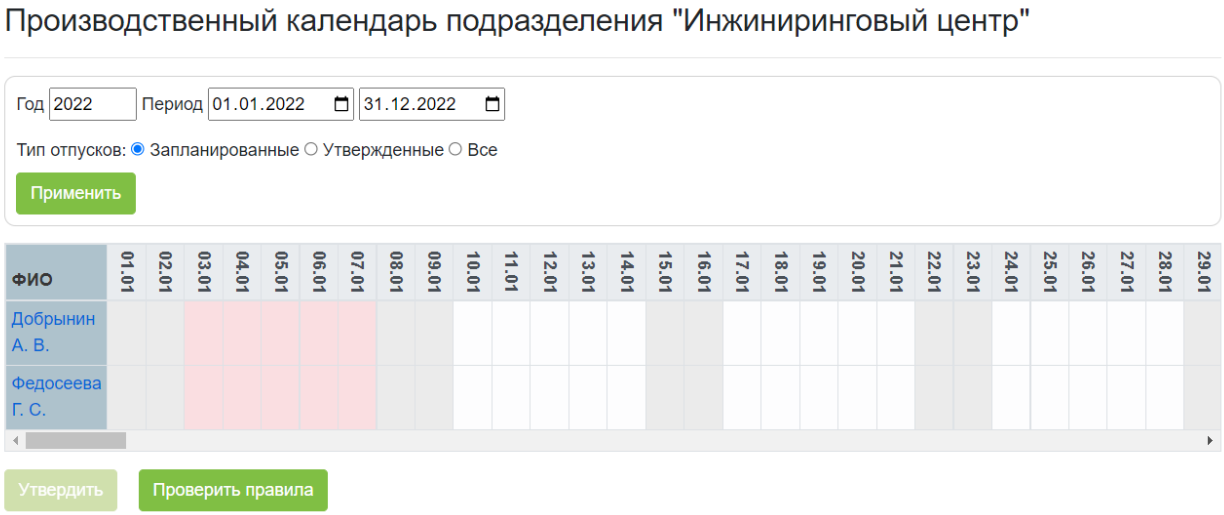


Рисунок 3.39 – Календарь отпусков подразделения

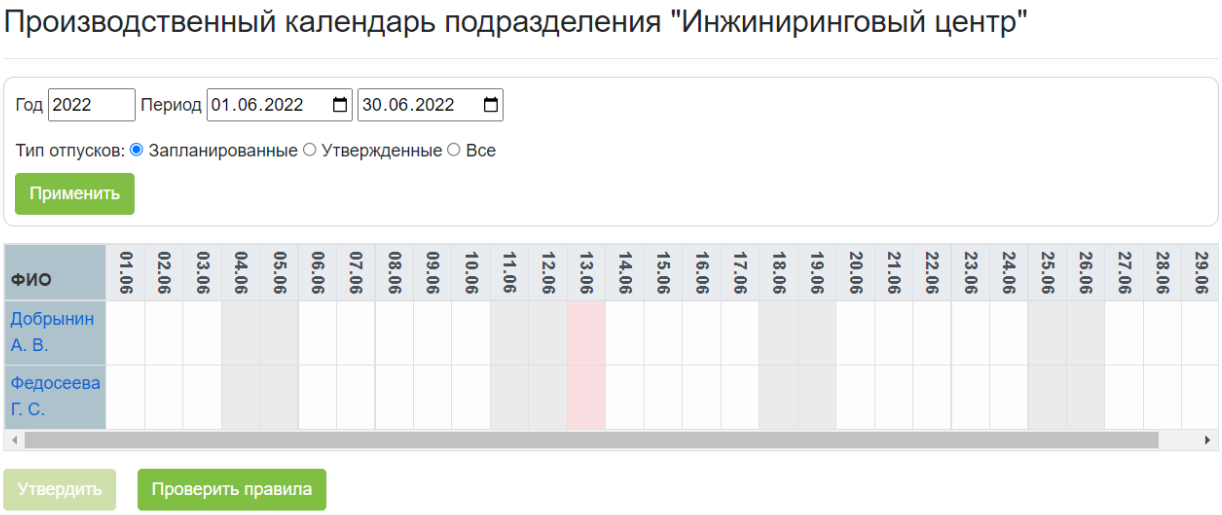


Рисунок 3.40 – Календарь, ограниченный июнем 2022-го года

Для примера сотруднице подразделения Федосеевой Г. С. был добавлен запланированный отпуск, состоящий из двух периодов: один длится весь январь, другой - с 3 по 21-е марта. Теперь на календаре можно увидеть появление оранжевых полос (рисунок 3.41), соответствующих выбранным периодам – это запланированные, но ещё не утвержденные отпуска.

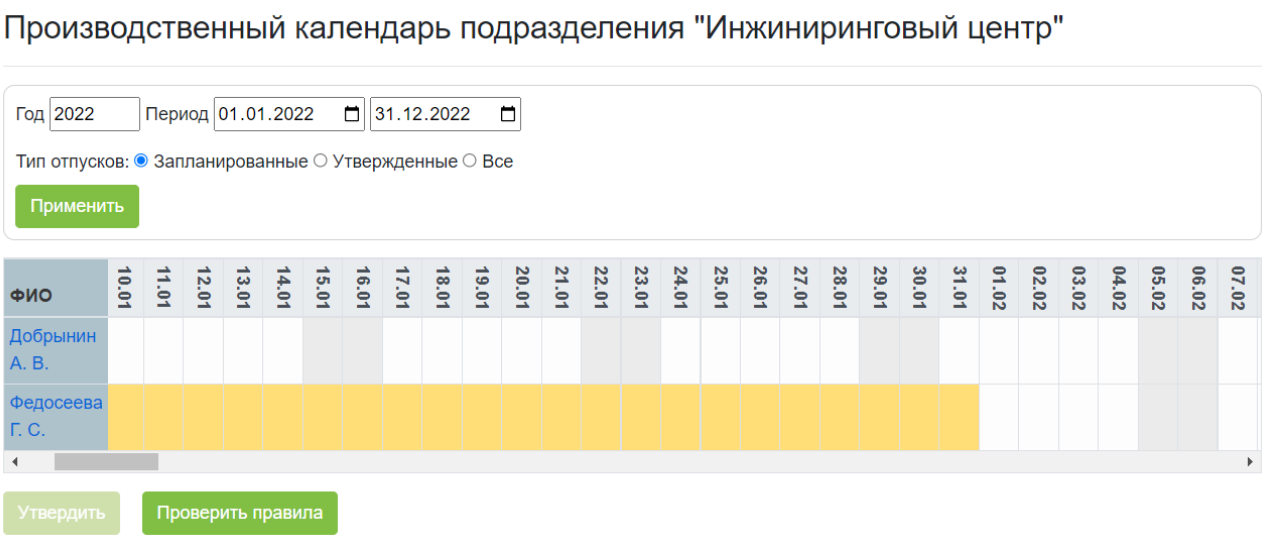


Рисунок 3.41 – Отображение на календаре запланированного отпуска

По умолчанию кнопка утверждения отпуска недоступна: для начала нужно проверить выбранные запланированные отпуска на соблюдение правил подразделения. Если какие-то правила окажутся нарушены, то появится соответствующее предупреждение, в котором можно пройти по ссылкам на правила, которые не были соблюдены. Так, ранее были созданы два правила (для сотрудников и для групп) для Добрынина и Федосеевой, согласно которым они должны уходить в отпуск одновременно. Эти правила были нарушены, что вызвало соответствующую ошибку (рисунок 3.42).

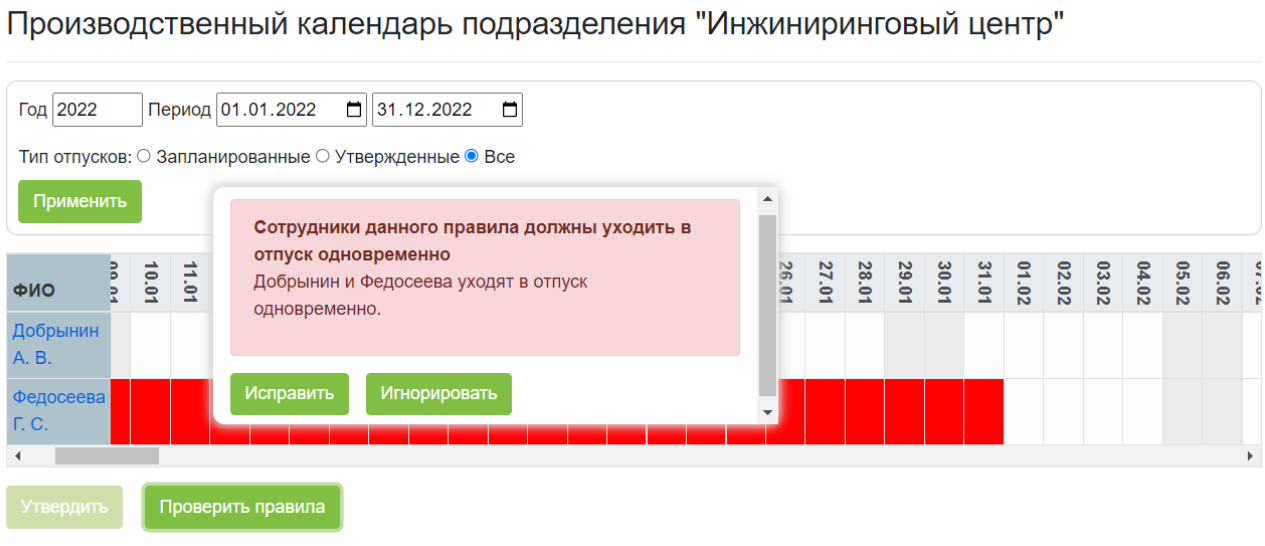


Рисунок 3.42 – Отображение сообщения о нарушении правил

Сообщение имеет две кнопки «Игнорировать» и «Исправить». Игнорирование сообщения подразумевает, что руководитель ознакомился с

нарушениями установленных им правил и согласен с ними, поэтому в этом случае кнопка утверждения отпусков окажется разблокированной.

Было решено исправить ошибку, поэтому Добрынину и Федосеевой были назначены совпадающие отпуска. Теперь появляется сообщение об успешной соблюдении правил, и кнопка утверждения отпусков становится доступной, как показано на рисунке 3.43.

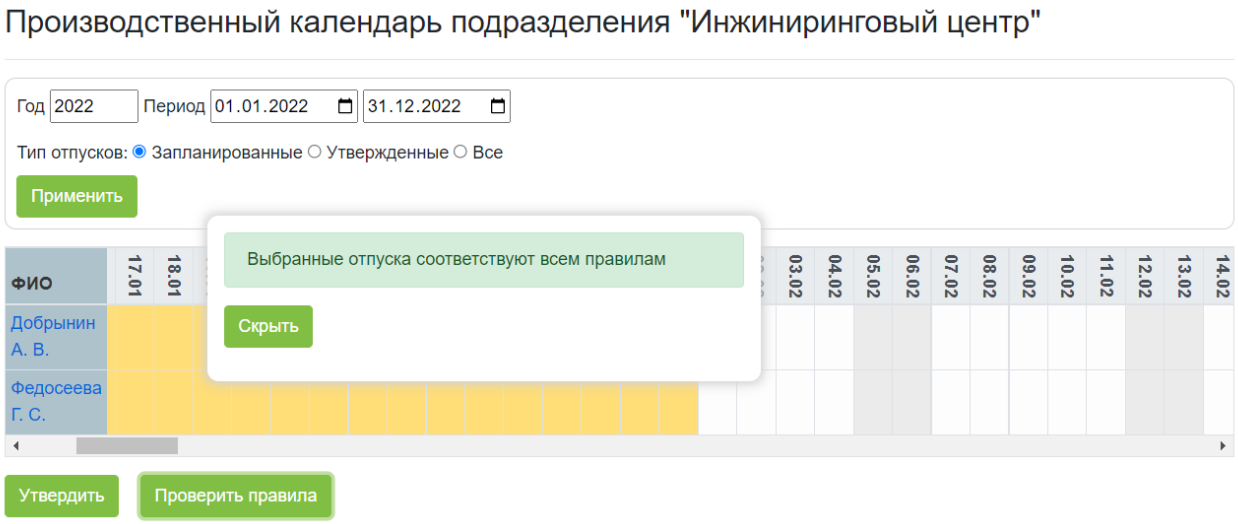


Рисунок 3.43 – Успешная проверка соблюдения правил отпусков

После успешной процедуры утверждения отпуска окрашиваются в ярко-зеленый цвет, как можно видеть на рисунке 3.44.

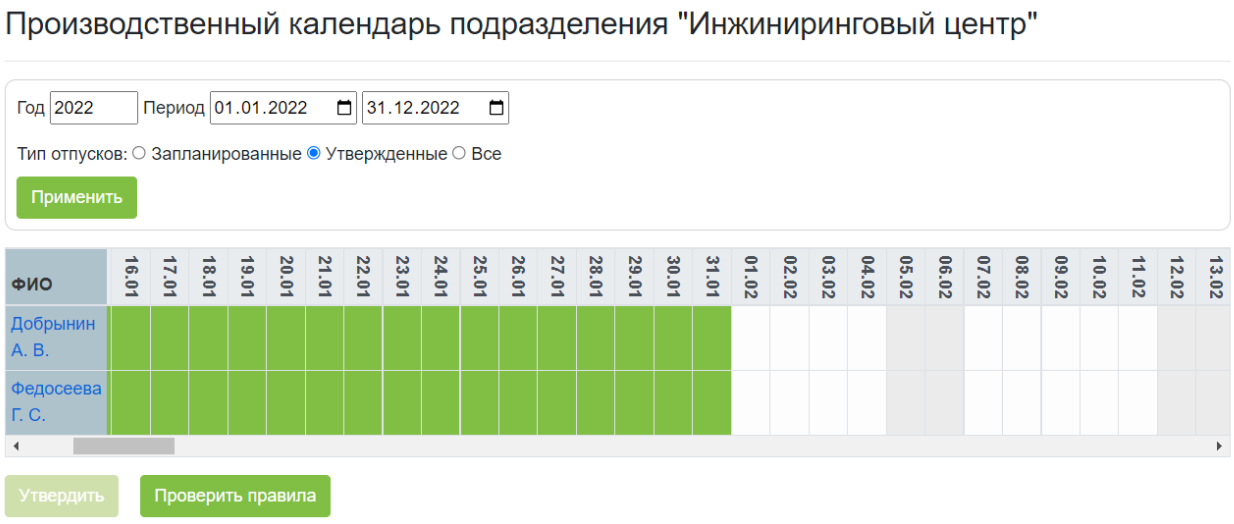


Рисунок 3.44 – Утвержденные отпуска в календаре

На этой же странице возможно загрузить календарь утвержденных отпусков в двух форматах: JSON и Excel.

Пример файла с календарем в формате JSON показан на рисунке 3.45. Как можно видеть, в этот файл записываются идентификатор и наименование подразделения, год календаря отпусков, а также список сотрудников с их идентификаторами, ФИО и периодами утвержденных отпусков.


```
{
  "DepartmentId": "246",
  "DepartmentName": "Инжиниринговый центр",
  "Year": 2022,
  "Employees": [
    {
      "Id": "2565",
      "Name": "Федосеева Галина Сергеевна",
      "VacationPeriods": [
        {
          "StartDate": "2022-01-01T00:00:00",
          "EndDate": "2022-01-31T00:00:00"
        },
        {
          "StartDate": "2022-03-02T00:00:00",
          "EndDate": "2022-03-07T00:00:00"
        }
      ]
    },
    {
      "Id": "3293",
      "Name": "Добрынин Андрей Валентинович",
      "VacationPeriods": [
        {
          "StartDate": "2022-01-01T00:00:00",
          "EndDate": "2022-01-31T00:00:00"
        },
        {
          "StartDate": "2022-03-03T00:00:00",
          "EndDate": "2022-03-04T00:00:00"
        }
      ]
    }
  ]
}
```


Рисунок 3.45 – Выгрузка файла с утвержденными отпусками в формате JSON

При выгрузке данных в формате Excel выполняется построение файла с простым календарем отпусков, как можно видеть на рисунке 3.46.

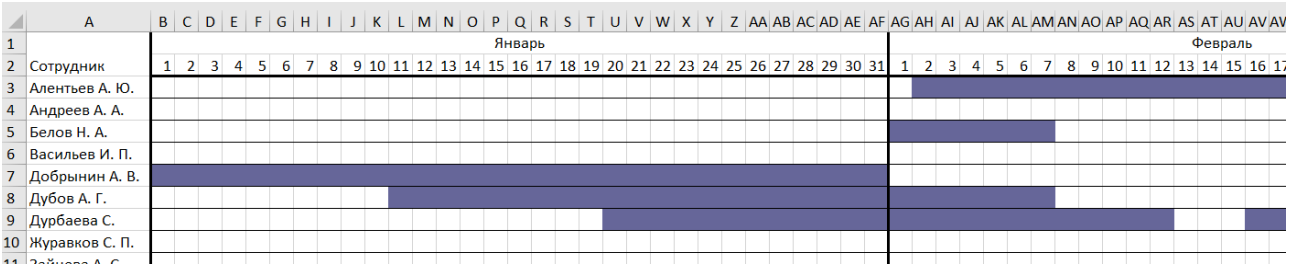


Рисунок 3.46 – Выгрузка файла с утвержденными отпусками в формате Excel

Выводы по разделу

В результате работы, описанной в данном разделе, было создано веб-приложение на основе выделенных ранее требований к системе. Так, программа поддерживает правила, установленные в Трудовом кодексе РФ:

1. Всем сотрудникам автоматически начисляются 28 дней основного оплачиваемого отпуска.
2. При разбиении отпуска как минимум один период должен длиться не менее 14 дней (в противном случае система не разрешит сохранить такой отпуск).
3. При наличии неистраченных отпусков за прошлый год они начисляются в счет следующего и обязательно должны быть потрачены.
4. При выборе сотрудником периодов отпуска праздничные дни не вычитаются из имеющегося у сотрудника количества отпускных дней.

При этом сотрудники подразделений могут выбирать периоды своих отпусков, редактировать или удалять их, пока они ещё не были утверждены. В свою очередь, руководители могут добавлять сотрудникам количество их отпускных дней по причинам, указанным в Трудовом кодексе, управлять отпусками сотрудников, создавать правила для их отпусков и выгружать календарь утвержденных отпусков в форматах JSON и Excel-таблицы.

Таким образом, в данном приложении были реализованы возможности, облегчающие работу руководителя подразделения: теперь при расстановке и утверждении отпусков сотрудников ему не нужно самостоятельно проверять соответствие этих отпусков ТК РФ и особенностям подразделения, ведь вместо этого возможно создать соответствующие правила (для сотрудников, групп или должностей), которые будут проверяться системой автоматически.

С другой стороны, все ещё есть большое количество функционала, который необходимо или желательно добавить в будущем. В первую очередь, это подключение к системе авторизации и API ТПУ, чтобы приложение могло

работать с реальными данными и использоваться настоящими сотрудниками ТПУ. С другой стороны, добавление некоторых функций также было бы полезным и удобным: например, автоматическая расстановка отпусков сотрудников таким образом, чтобы они не нарушали правила выбора отпусков.

4 Финансовый менеджмент, ресурсоэффективность и ресурсосбережение

4.1 Предпроектный анализ

4.1.1 Потенциальные потребители результатов исследования

Для выявления и анализа потребителей результатов исследования необходимо произвести изучение целевого рынка, на котором будет продаваться разработка, и его сегментирование – разделение потенциальных клиентов на группы по ряду критериев [23].

В данном случае ведется разработка веб-приложения, поэтому в качестве целевого рынка следует рассматривать рынок услуг по разработке интернет-ресурсов для внутреннего использования предприятиями. Такие интернет-ресурсы по своему наполнению и назначению могут быть различными: корпоративные сайты, веб-приложения для управления проектами, отпусками, автоматизации процессов. В свою очередь, компании-клиенты, заказывающие или пользующиеся подобными ресурсами, могут быть сегментированы по размеру: крупные, средние и мелкие.

Полученная карта сегментирования рынка представлена в виде таблицы 4.1.

Таблица 4.1 – Карта сегментирования рынка услуг по разработке приложений для предприятий

		Вид интернет-ресурса			
		Корпоративный сайт	Управление отпусками	Управление проектами	Автоматизация процессов
Размер компании	Крупные				
	Средние				
	Мелкие				

 Фирма А  Фирма Б  Фирма В

Как можно видеть из таблицы, сегмент управления отпусками для средних компаний является незанятым, поэтому возможно сконцентрироваться именно на нем, но с возможным масштабированием системы для работы в крупных и мелких компаниях с обеспечением конкурентного преимущества за счет более углубленного и специализированного функционала веб-приложения.

4.1.2 Анализ конкурентных технических решений с позиции ресурсоэффективности и ресурсосбережения

Существуют различные инструменты, которые могут быть как специализированы на задачах, для решения которых разрабатывается данное веб-приложение, так и адаптированы под них. В обоих случаях такие разработки являются конкурентными, и необходимо провести их анализ.

В качестве конкурентных решений были выбраны следующие продукты:

1. Microsoft Excel – пример табличного процессора, функционал которого достаточен для создания календаря отпусков, записи и отслеживания отпускных дней сотрудников и т.д.
2. Jira с приложением Calendar for Jira – программный инструмент для управления проектами от компании Atlassian, используемый для отслеживания задач и прогресса выполнения проектов [24]. С подключенным календарем позволяет наглядно визуализировать отпуска сотрудников в виде цветного графика, отслеживать их в контексте ключевых дат проектов, а также отправлять уведомления о выходе сотрудников в отпуск [25].
3. 1С: Зарплата и управление персоналом – система для автоматизации задач, связанных с управлением персоналом, расчетом заработной платы, кадровым учетом и многим другим [26]. Помимо прочего, в данной системе поддерживается и управление отпусками с

возможностью расчета отпускных, формирования графиков и приказов на отпуск.

На основе анализа конкурентных решений была сформирована таблица 4.2.

Таблица 4.2 – Оценочная карта для сравнения конкурентных технических решений (разработок)

Критерии оценки	**Вес крите-рия**	**Баллы**				**Конкурентоспособность**			
		$Б_ф$	**$Б_{к1}$**	**$Б_{к2}$**	**$Б_{к3}$**	**$К_ф$**	**$К_{ф1}$**	**$К_{ф2}$**	**$К_{ф3}$**
1	2	3	4	5	6	7	8	9	10
Технические критерии оценки ресурсоэффективности									
1. Повышение производительности труда пользователя	0,2	5	2	4	5	1	0,4	0,8	1
2. Удобство в эксплуатации	0,05	4	1	3	4	0,2	0,05	0,15	0,2
3. Помехоустойчивость	0,01	3	5	4	3	0,03	0,05	0,04	0,03
4. Масштабируемость	0,02	5	2	3	5	0,1	0,04	0,06	0,1
5. Энергоэкономичность	0,02	4	5	4	3	0,08	0,1	0,08	0,06
6. Надежность	0,05	3	5	4	3	0,15	0,25	0,2	0,15
7. Потребность в ресурсах памяти	0,02	4	5	4	3	0,08	0,1	0,08	0,06
8. Функциональная мощность (предоставляемые возможности)	0,2	5	1	3	4	1	0,2	0,6	0,8
9. Простота эксплуатации	0,05	5	4	5	2	0,25	0,2	0,25	0,1
10. Качество пользовательского интерфейса	0,1	4	3	5	4	0,4	0,3	0,5	0,4
11. Возможность интеграции	0,05	5	1	4	5	0,25	0,05	0,2	0,25
Экономические критерии оценки эффективности									
1. Конкурентоспособность продукта	0,04	5	2	4	4	0,2	0,08	0,16	0,16
2. Уровень проникновения на рынок	0,01	1	5	3	5	0,01	0,05	0,03	0,05
3. Цена	0,09	5	3	4	3	0,45	0,27	0,36	0,27
4. Послепродажное обслуживание	0,07	5	3	3	5	0,35	0,21	0,21	0,35
5. Финансирование научной разработки	0,01	5	3	4	4	0,05	0,03	0,04	0,04
6. Срок выхода на рынок	0,01	1	5	5	5	0,01	0,05	0,05	0,05
Итого	1	69	55	66	67	**4,61**	**2,43**	**3,81**	**4,07**

По полученной таблице можно видеть, что наиболее серьезным конкурентом является система от 1С, так как, в отличие от других инструментов,

её функционал изначально включает в себя возможность управления отпусками, что обеспечивает высокие значения таких показателей, как повышение производительности труда пользователей, функциональная мощность и удобство в эксплуатации; кроме того, как и разрабатываемая система, 1С: ЗУП предполагает возможное использование на предприятиях разного размера с интеграцией их данных в системе. Следует также учитывать, что системы 1С уже долгое время занимают прочное положение на рынке и обладают высокой популярностью в различных компаниях и организациях. Соответственно, для обеспечения достаточной конкурентной способности разрабатываемая система должна не уступать 1С в удобстве и простоте использования и при этом предоставлять также дополнительные, уникальные возможности для управления отпусками.

Наиболее низкие баллы получила Microsoft Excel (либо любой аналогичный табличный процессор). Несмотря на высокие показатели энергоэкономичности, надежности, простоты в эксплуатации, подобные программы не обладают никаким специализированным функционалом для работы с отпусками (по сути, руководителю или секретарю подразделения приходится адаптировать имеющийся функционал таблиц под эту задачу), из-за чего управление отпусками посредством таблиц может в результате оказаться неудобным и сложным.

Использование Jira представляет собой промежуточный вариант между Excel и 1С: с одной стороны, как и в таблицах, функционал все ещё ограничен созданием календаря с выделением на нем периодов отпусков сотрудников; с другой стороны, здесь есть некоторые специальные возможности, повышающие удобство работы при планировании отпусков, а также интеграция в том случае, если предприятие уже пользуется Jira.

4.1.3 SWOT-анализ

SWOT-анализ – это метод планирования и инструмент для оценки внешних и внутренних факторов, от которых зависит, как компания или отдельный продукт будут развиваться на рынке [27]. Данный анализ строится на основе 4 основных параметров: S (streangths) – сильные стороны, W (weaknesses) – слабые стороны, O (opportunities) – возможности, T (threats) – угрозы.

На первом этапе анализа выявляются и описываются сильные и слабые стороны, возможности и угрозы проекта, которые либо уже проявились в ходе или в результате работы, либо могут потенциально проявиться в будущем.

На втором этапе необходимо сопоставить между собой сильные и слабые стороны с возможностями и угрозами и описать, как их сочетания могут повлиять на разрабатываемый проект.

Полученная в результате проведения анализа матрица SWOT представлена в таблице 4.3.

Таблица 4.3 – Матрица SWOT-анализа

	Сильные стороны	**Слабые стороны**
	С1. Наглядное отображение отпусков сотрудников. С2. Простой и удобный web-интерфейс, доступный из браузера пользователя. С3. Богатый функционал с множеством возможностей управления отпусками и поддержкой соблюдения Трудового кодекса РФ. С4. Подробное документирование кода программы. С5 Постоянная поддержка разработчика. С6. Использование кроссплатформенных технологий, позволяющих развертывание веб-приложения на различных ОС.	Сл1. Отсутствие короткого и запоминающегося доменного имени. Сл2. Недоработки и ошибки работы системы. Сл3. Необходимость проведения работ по обеспечению адаптивности интерфейса. Сл4. При использовании веб-приложения на разных предприятиях необходима корректировка в зависимости от места хранения определенных данных: в БД приложения или на сервере предприятия. Сл5. Отсутствие на данный момент реального выхода на рынок.
Возможности	В1С3. Благодаря поддержке со стороны организации возможно узнать в	В5Сл1. Покупка доменного имени означает необходимость регулярной его оплаты.
В1. Получение поддержки при		

разработке со стороны ТПУ. В2. Уход с рынка конкурентных разработок. В3. Появление спроса со стороны других организаций или предприятий. В4. Повышение стоимости конкурентных разработок. В5. Покупка доменного имени.	подробностях обо всех нюансах и пожеланиях к работе подобной системы. В1С5. В результате взаимодействия разработчика и организации возможно более быстро и качественно реализовать готовый проект. В1С6. Возможность размещения приложения на серверах ТПУ. В2В4С1С2С3. При труднодоступности разработок конкурентов данный проект способен привлечь новых клиентов за счет его достоинств. В5С2. При покупке запоминающего доменного имени пользователи получают возможность простого доступа к системе через их браузер.	В3Сл4. Необходимость доработки приложения перед началом его работы может оттолкнуть потенциальных клиентов. В3Сл2Сл3. Имеющиеся недостатки и недоработки системы не способствуют успешному выходу на рынок. В1Сл4. Так как разработка велась на основе данных из ТПУ, есть вероятность, что система не будет полностью пригодна для использования в других университетах или предприятиях, не связанных со сферой высшего образования.
Угрозы У1. Потеря актуальности. У2. Медленная работа системы. У3. Отсутствие интереса к системе со стороны реальных пользователей. У4. Появление новых конкурентных разработок. У5. Повышение цены на аренду или прекращение работы VDS-сервера.	У3С1С2С3. Широкие и уникальные (по сравнению с конкурентными разработками) возможности системы способны привлечь внимание к системе со стороны пользователей. У2С5С6. Постоянная поддержка разработчика и работа над совершенствованием системы означают поиск способов увеличения скорости программы, в том числе использование более быстрого сервера для развертывания веб-приложения. У4С3С5. Система может оставаться конкурентоспособной даже при появлении новых конкурентов за счет постоянного обновления и уже имеющегося более широкого и специализированного функционала, чем у существующих конкурентных разработок.	У2У3Сл2Сл3. Существенные недостатки системы способны привести к оттоку пользователей и отказу их от использования данной разработки. У4Сл2Сл3. Конкурентные разработки могут оказаться лучше данного проекта не только за счет функционала, но и благодаря более надежной и приятной работе системы. У1У4Сл5. Промедление с выходом на рынок может привести к тому, что подобная системе уже окажется неактуальной либо на рынке уже будет распространена некоторая система с аналогичным или лучшим функционалом. У3Сл1. Без короткого и запоминающегося доменного имени пользователи не смогут получать доступ к системе, не имея при себе прямой ссылки.

4.1.4 Оценка готовности проекта к коммерциализации

При реализации научной разработки необходимо оценить степень её готовности к коммерциализации и определить уровень собственных знаний для её проведения либо завершения. Таким образом можно узнать, насколько проработанным является проект с точки зрения коммерциализации и компетенций разработчика. Для этого была заполнена таблица 4.4.

Таблица 4.4 – Степень готовности научного проекта к коммерциализации

№ п/п	Наименование	Степень проработанности научного проекта	Уровень имеющихся знаний у разработчика
1.	Определен имеющийся научно-технический задел	4	4
2.	Определены перспективные направления коммерциализации научно-технического задела	5	4
3.	Определены отрасли и технологии (товары, услуги) для предложения на рынке	4	4
4.	Определена товарная форма научно-технического задела для представления на рынок	4	3
5.	Определены авторы и осуществлена охрана их прав	5	4
6.	Проведена оценка стоимости интеллектуальной собственности	5	5
7.	Проведены маркетинговые исследования рынков сбыта	4	4
8.	Разработан бизнес-план коммерциализации научной разработки	2	4
9.	Определены пути продвижения научной разработки на рынок	5	3
10.	Разработана стратегия (форма) реализации научной разработки	5	4
11.	Проработаны вопросы международного сотрудничества и выхода на зарубежный рынок	1	1
12.	Проработаны вопросы использования услуг инфраструктуры поддержки, получения льгот	2	2

13.	Проработаны вопросы финансирования коммерциализации научной разработки	2	2
14.	Имеется команда для коммерциализации научной разработки	4	3
15.	Проработан механизм реализации научного проекта	5	5
	ИТОГО БАЛЛОВ	57	52

Итоговые баллы за степень проработанности научного проекта и уровень имеющихся знаний у разработчика лежат в пределах от 50 до 60 баллов, что означает перспективность разработки выше среднего. Тем не менее, это также означает, что для успешной коммерциализации продукта следует произвести дополнительную серьезную работу как над проработкой проекта в этом плане, так и над соответствующими знаниями разработчика.

4.1.5 Методы коммерциализации результатов научно-технического исследования

Из методов коммерциализации разработки наиболее потенциально успешным представляется инжиниринг. Так, разработчик данной системы может предоставлять услуги по её развертыванию и доработке под условия конкретного предприятия, в результате чего в такой организации будут усовершенствованы процессы, связанные с планированием отпусков в её подразделениях.

С другой стороны, для данного проекта подходит и метод передачи интеллектуальной собственности в уставной капитал предприятия. Изначально данная система разрабатывается для ТПУ на основе данных и структуры этого университета, поэтому в случае, если разработчик не сможет или не будет заинтересован в дальнейшем продвижении разработки в других организациях, то наилучшим решением может оказаться передача данной системы и связанной с ней интеллектуальной собственности ТПУ.

4.2 Инициация проекта

Инициация проекта представляет собой группу процессов, которые выполняются для определения нового проекта (или новой фазы уже существующего), его целей, содержания, изначальных финансовых ресурсов. Также выделяются заинтересованные стороны проекта, оказывающие влияние на его конечный результат.

4.2.1 Цели и результаты проекта

Для определения целей проекта сначала необходимо перечислить заинтересованные стороны, которые были записаны в таблицу 4.5.

Таблица 4.5 – Заинтересованные стороны проекта

Заинтересованные стороны проекта	**Ожидание заинтересованных сторон**
Университет	Программа с функционалом управления отпусками
Пользователь	Удобство и простота использования системы
Разработчик	Получение прибыли с разработанного продукта
Научный руководитель, студент	Выполненная выпускная квалификационная работа

В свою очередь, цели и результат проекта представлены в таблице 4.6.

Таблица 4.6 – Цели и результат проекта

Цели проекта:	• Сбор требований со стороны университета для разрабатываемой системы; • Проектирование вариантов использования системы, базы данных, карты веб-приложения, ключевых алгоритмов, эскизирование интерфейса; • Осуществление расчета стоимости проекта; • Реализация требуемого функционала системы в виде веб-приложения; • Развертывание приложения на облачном сервере с операционной системой Linux;

	• Осуществление тестирования системы; • Доработка системы в соответствии с замечаниями со стороны университета.
Ожидаемые результаты проекта:	Успешное создание и развертывание на облачном сервере информационной системы для управления отпусками.
Критерии приемки результата проекта:	Соответствие функционала требованиям университета; бесперебойная работа системы.
Требования к результату проекта:	**Требование**:
	Разработанная система полностью удовлетворяет функциональным требованиям.
	При работе с системой не возникают непредвиденные либо мешающие дальнейшей работе ошибки.
	Разработанное приложение доступно удаленно из любого современного веб-браузера.

4.2.2 Организационная структура проекта

В рабочую группу проекта входят те люди, которые так или иначе связаны с реализацией данного проекта. Их роли и функции были записаны в таблицу 4.7.

Таблица 4.7 – Рабочая группа проекта

№ п/п	ФИО, основное место работы, должность	Роль в проекте	Функции	Трудозатраты, час.
1	Шкулов Никита Петрович, ТПУ, магистрант ОИТ ИШИТР ТПУ	Исполнитель по проекту	Реализация проекта	800
2	Фадеев Александр Сергеевич, ТПУ, доцент ОИТ ИШИТР ТПУ	Руководитель и заказчик проекта	Постановка задачи, проверка и корректировка результатов	20
ИТОГО:				820

4.2.3 Ограничения и допущения проекта

Ограничения проекта – это все факторы, которые могут ограничивать степень свободы участников проекта и задавать границы проекта – те параметры

проекта, которые не будут реализованы в его рамках. Такие ограничения для данной разработки были внесены в таблицу 4.8.

Таблица 4.8 – Ограничения проекта

Фактор	Ограничения
2.3.1 Бюджет проекта	200 000 рублей
2.3.1.1 Источник финансирования	НИ ТПУ
2.3.2 Сроки проекта	01.02.2021 – 31.05.2022
2.3.2.1 Дата утверждения плана управления проектом	28.12.2021
2.3.2.2 Дата завершения проекта	31.05.2022

4.3 Планирование управления научно-техническим проектом

Группа процессов планирования включает в себя различные процессы, осуществляемые для определения содержания работ, целей, которые следует выполнить в ходе работы над проектом, а также разработки последовательности действий, необходимых для достижения выделенных целей.

План управления научно-техническим проектом должен включать в себя такие элементы, как иерархическая структура работ проекта, контрольные события, план проекта и бюджет научного исследования.

4.3.1 Иерархическая структура работ проекта

Иерархическая структура работ – это декомпозиция проекта на более мелкие и измеримые части. ИСР описывает все результаты и работы, которые должны быть выполнены для завершения проекта, а также их место в структуре проекта [28].

Иерархическая структура работ по проекту представлена на рисунке 4.1.

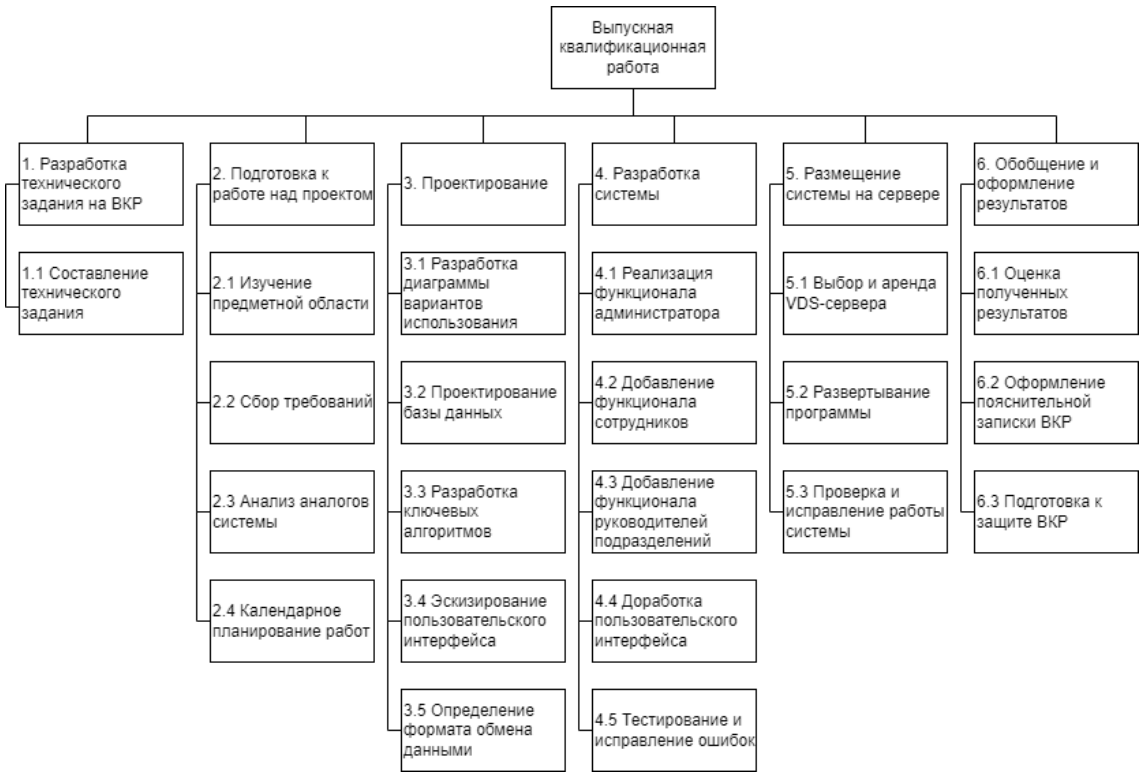


Рисунок 4.1 – Иерархическая структура работ по проекту

5.3.2 План проекта

Для иллюстрации плана проекта может использоваться диаграмма Ганта – инструмент управления проектами, в котором план проекта состоит из двух частей: в левой части приводится список заданий, а в правой – временная шкала с полосами, изображающими выполняемую исполнителями заданий работу [29]. Диаграмма Ганта, построенная для данного проекта, приведена на рисунке 4.2.

Код работы	Вид работ	Исполнители	Тк кал, дн.	Продолжительность выполнения работ
				2021: 2 3 4 5 6 7 8 9 10 11 12 — 2022: 1 2 3 4 5
1.1	Составление ТЗ	Разработчик, руководитель	62	
2.1	Изучение предметной области	Разработчик	49	
2.2	Сбор требований	Разработчик, руководитель	21	
2.3	Анализ аналогов системы	Разработчик	48	
2.4	Календарное планирование работ	Разработчик	29	
3.1	Создание диаграммы предецентов	Разработчик	7	
3.2	Проектирование БД	Разработчик	12	
3.3	Разработка ключевых алгоритмов	Разработчик	11	
3.4	Эскизирование пользовательского интерфейса	Разработчик	5	
3.5	Определение формата обмена данными	Разработчик, руководитель	14	
4.1	Создание панели администратора	Разработчик	14	
4.2	Реализация функционала для сотрудников	Разработчик	35	
4.3	Добавление функционала для руководителей	Разработчик	63	
4.4	Доработка пользовательского интерфейса	Разработчик	35	
4.5	Тестирование	Разработчик, руководитель	30	
5.1	Выбор и аренда VDS-сервера	Разработчик	7	
5.2	Развертывание	Разработчик	11	
5.3	Проверка работы	Разработчик, руководитель	8	
6.2	Оформление ВКР	Разработчик, руководитель	23	

Разработчик Руководитель

Рисунок 4.2 – Календарный план-график проекта

4.3.3 Бюджет научного исследования

В бюджете научного исследования должны быть наиболее полно и достоверно отображены все виды планируемых расходов, необходимых для его выполнения. Сами планируемые затраты могут быть сгруппированы по

нескольким статьям: сырье и материалы, специальное оборудование, основная и дополнительная заработная плата, отчисления на социальные нужны, научные и производственные командировки, оплата работ, выполняемых сторонними организациями и предприятиями и т.д.

В таблицу 4.9 были внесены затраты на специальное оборудование для научных (экспериментальных) работ. Так как данная работа связана с программированием, то в число спецоборудования входят различная компьютерная техника (которая предоставляется бесплатно при работе над проектом в университете) и специализированное ПО (имеющее бесплатные версии). Кроме того, сюда была внесена оплата за VDS-сервер с операционной системой Ubuntu, на котором планируется размещать веб-приложение для работы.

Таблица 4.9 – Расчет затрат по статье «Спецоборудование для научных работ»

№ п/п	Наименование оборудования	Кол-во единиц оборудования	Цена единицы оборудования, тыс. руб.	Общая стоимость оборудования, тыс. руб.
1.	Персональный компьютер	1	0	0
2.	Мышь компьютерная	1	0	0
3.	Клавиатура	1	0	0
4.	VDS-сервер с ОС Ubuntu	3	1,4	4,2
5.	Среда разработки Visual Studio Community	1	0	0
6.	СУБД MySQL	1	0	0

Основная заработная плата – это тот размер зарплаты, которую будет необходимо выплатить участникам проекта в процессе её выполнения; величина расходов по заработной плате зависит от трудоемкости выполняемых работ и действующей системы оплаты труда.

Результаты расчетов по заработной плате приведены в таблице 4.10. Для руководителя (доцента со степенью кандидата наук) месячный должностной оклад без районного коэффициента составляет 26 300 рублей; для разработчика в качестве заработной платы рассматривалась месячная стипендия магистранта ТПУ.

Таблица 4.10 – Расчет основной заработной платы

Исполнители	**$З_б$, руб.**	**$k_р$**	**$З_м$, руб**	**$З_{дн}$, руб.**	**$Т_р$, раб. дн.**	**$З_{осн}$, руб.**
Руководитель	26300	1,3	34190	1449,2	40	57968
Разработчик	3100		3100	160,8	484	77827
					Итого по статье $З_{осн}$:	135795

Для расчета отчислений на социальные нужды необходимо умножить размер основной заработной платы на коэффициент отчислений на уплату во внебюджетные фонды, равный 30%. Соответственно, размер отчислений для зарплаты руководителя равен 17390 рублей.

Накладные расходы – это расходы по содержанию, эксплуатации и ремонту оборудования, зданий и т.д. При расчетах величина накладных расходов может приниматься в размере 70-90% от суммы основной заработной платы работников, участвующих в процессе выполнения проекта. Следовательно, при величине коэффициента 0,7 размер накладных расходов равен 95 056 рублей.

Полученные значения затрат, дающие представление о необходимых денежных средствах для реализации проекта, были внесены в таблицу 4.11. Таким образом, общие затраты на проект составляют 252 441 рубль.

Таблица 4.11 – Бюджет затрат НТИ

№	Затраты по статьям				
	Специальное оборудование для научных работ	Основная заработная плата	Накладные расходы	Отчисления на социальные нужны	Итого плановая себестоимость
1	4200	135795	95056	17390	252 441

4.3.4 Организационная структура проекта

Для определения наиболее подходящей организационной структуры проекта была использована таблица 4.12.

Таблица 4.12 – Выбор организационной структуры научного проекта

Критерии выбора	**Функциональная**	**Матричная**	**Проектная**
Степень неопределенности условий реализации проекта	Низкая	Высокая	Высокая
Технология проекта	Стандартная	Сложная	Новая
Сложность проекта	Низкая	Средняя	Высокая

Взаимозависимость между отдельными частями проекта	Низкая	Средняя	Высокая
Критичность фактора времени (обязательства по срокам завершения работ)	Низкая	Средняя	Высокая
Взаимосвязь и взаимозависимость проекта от организаций более высокого уровня	Высокая	Средняя	Низкая

Степень неопределенности условий реализации проекта низкая, так как изначально определены сроки к разрабатываемой системе, используемые технология и самые необходимые требования к ней. Технология проекта стандартная: пусть сам проект и предполагает создание ряда функций, которых нет у конкурентных разработок, сами выбранные технологии для разработки веб-приложения и работы с базой данных являются распространёнными, широко используемыми и хорошо документированными. Сложность проекта высокая за счет большого количества функционала, который следует реализовать, и множества сложностей в предметной области управления отпусками. Взаимозависимость между отдельными частями проекта высокая: вся система и весь её функционал строится вокруг одной главной, объединяющей функции выбора отпусков сотрудниками и утверждения их руководителем. Критичность фактора времени также является высокой, так как проект обязательно необходимо завершить в срок для возможности защиты ВКР. Взаимосвязь проекта с организациями более высокого уровня средняя: такая зависимость есть, но она не оказывает влияния на большую часть времени разработки программы.

В результате проведенного анализа критериев можно сделать вывод, что для данной разработки лучше всего подходит именно проектная структура научного проекта в связи с высокими сложностью, взаимозависимостью между его отдельными частями и критичным фактором времени.

4.3.5 План управления коммуникациями проекта

План управления коммуникациями показывает, как именно осуществляется коммуникации между участниками проекта в ходе работы над ним. Этот план для данной работы представлен в таблице 4.13.

Таблица 4.13 – План управления коммуникациями проекта

№ п/п	Какая информация передается	Кто передает информацию	Кому передается информация	Когда передается информация
1.	Документы и информация по проекту	Ответственное лицо по направлению	Руководителю проекта	Не позже сроков графиков и контрольных точек
2.	Обмен информацией о текущем состоянии проекта	Разработчик	Руководителю проекта	Еженедельно (понедельник)
3.	Выполнение контрольной точки проекта	Разработчик	Руководителю проекта	По завершении очередного контрольного события по плану проекта

4.3.6 Реестр рисков проекта

Риски проекта – это все те возможные события, наступление которых возможно в ходе работы над проектом и способно вызвать негативные последствия, ведущие к нежелательным эффектам, снижению качества и эффективности работы, проекта и т.д. Учет рисков помогает либо избежать опасных факторов, либо подготовиться к ним.

Риски были внесены в таблицу 4.14.

Таблица 4.14 – Реестр рисков проекта

№	Риск	Потенциальное воздействие	Вероят-ность наступ-ления (1-5)	Влия-ние (1-5)	Уровень риска	Способы смягчения риска	Условия наступления
1	Потеря актуаль-ности	Ненужность системы конечным	3	5	Сред-ний	Постоянное взаимодействие с заказчиком,	Затянувшаяся разработка, несоответствие

		пользователям, отсутствие возможности продажи продукта				уточнение актуальных потребностей пользователей	конечного продукта требованиям
2	Отсутствие спроса	Невозможность продажи уже готового продукта	4	4	Высо-кий	Обеспечение привлекательного для потенциальных клиентов функционала, продвижение продукта	Несоответствие продукта потребностям пользователей, слишком слабое продвижение на рынке
3	Появление новых конкурент-ных разработок	Снижение спроса данной разработки	2	3	Низкий	Мониторинг конкурентных разработок и их функционала, своевременное реагирование и обновление программы	Появление новых инструментов для управления отпусками
4	Повыше-ние цены на аренду сервера	Увеличение расходов на разработку и работу системы	3	2	Низкий	Учет возможного повышения цены при планировании расходов	Изменение экономической ситуации, потребности арендодателя сервера
5	Ошибки работы программы	Снижение удовлетворен-ности пользователей системой либо невозможность работы с ней	4	5	Высо-кий	Внимание к качеству системы при её разработке, подробное тестирование и устранение всех найденных недостатков	Недостаточное тестирование системы, неисправлен-ные недоработки

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

4.4.1 Оценка абсолютной эффективности исследования

Для оценки общей экономической эффективности инвестиций применяется ряд показателей, такие как чистый доход, чистый дисконтированный доход, внутренняя норма доходности, потребность в

дополнительном финансировании, срок окупаемости, индексы доходности затрат и инвестиций и т.д.

Чистый дисконтированный доход (NPV, Net Present Value) – это накопленный дисконтированный эффект за расчетный период. Этот показатель характеризует превышение суммарных денежных поступлений над суммарными затратами для данного проекта с учетом неравноценности затрат и результатов, относящихся с различным моментам времени. Для признания проекта эффективным с точки зрения инвестора или предприятия необходимо, чтобы ЧДД был положительным.

В таблице 4.15 приведены расчеты NPV для данного проекта.

Таблица 4.15 – Расчет чистого дисконтированного дохода для проекта

№	Наименование показателей	Шаг расчета				
		0	1	2	3	4
1.	Выручка от реализации, тыс. руб.	0	152,460	152,460	152,460	152,460
2.	Итого приток, тыс. руб.	0	152,460	152,460	152,460	152,460
3.	Инвестиционные издержки, тыс. руб.	-252,441	0	0	0	0
4.	Операционные затраты, тыс. руб. (С+Ам+ФОТ)	0	25,500	25,500	25,500	25,500
5.	Налогооблагаемая прибыль	0	126,960	126,960	126,960	126,960
6.	Налоги, тыс. руб. ($Выр_{опер}$ = $приб_{донал}$*20%)	0	25,392	25,392	25,392	25,392
7.	Итого отток, тыс. руб. ($Затр_{опер}$+налоги)	-252,441	50,892	50,892	50,892	50,892
8.	Чистый денежный поток, тыс. руб. ЧДП=$П_{чист}$+Ам ($П_{чист}$=$П_{донал.}$-налог)	-252,441	101,568	101,568	101,568	101,568
9.	Коэффициент дисконтирования (приведена при i=20%)	1	0,833	0,694	0,579	0,482

10.	Дисконтированный чистый денежный поток, тыс. руб. (c8*c9)	-252,441	84,640	70,533	58,778	48,982
11.	То же нарастающим итогом, тыс. руб.	-252,441	-167,801	-97,268	-38,490	10,492

В итоге можно видеть, что чистый дисконтированный доход на четвертом шаге расчета для проекта положителен и равен 10 492 рубля, так что можно сделать вывод об его эффективности.

Дисконтированный срок окупаемости (DPP) – это показатель, характеризующий временной промежуток, за который удастся вернуть инвестируемые деньги с учетом стоимости вкладываемой единицы [30]. Расчеты этого показателя представлены в таблице 4.16.

Таблица 4.16 – Дисконтированный срок окупаемости

№	Наименование показателя	Шаг расчета				
		0	1	2	3	4
1.	Дисконтированный чистый денежный поток (i=0,2)	-252,441	84,640	70,533	58,778	48,982
2.	То же нарастающим итогом	-252,441	-167,801	-97,268	-38,490	10,492
3.	Дисконтированный срок окупаемости	DPP = 1 + 38,490 / 48,982 = 0,78 года				

Индекс доходности (рентабельности) инвестиций показывает, сколько приходится дисконтированных денежных поступлений на рубль инвестиций.

Расчет осуществляется по формуле (4.1).

$$PI = \sum_{t=1}^{n} \frac{ЧПД_t}{(1+i)^t} / I_0, \quad (4.1)$$

где t – шаг расчета,

$ЧПД_t$ – дисконтированный чистый денежный поток, рублей,

i – ставка дисконтирования,

I_0 – первоначальные инвестиции, рублей.

Соответственно, значение индекса доходности возможно рассчитать следующим образом:

$$PI = \frac{\left(\frac{84640}{1,2} + \frac{70533}{1,44} + \frac{58778}{1,728} + \frac{48982}{2,0736}\right)}{252441} = 0,7$$

Полученное значение меньше единицы, из чего можно сделать вывод, что проект неэффективен при i = 0,2. Следовательно, этот результат нужно принять во внимание и внести некоторые изменения в проект, чтобы на каждый рубль инвестиций приходилось больше единицы дисконтированных денежных поступлений.

Социальная эффективность научного проекта отображает социально-экономические последствия реализации проекта для общества и отдельных категорий населения, причем учитывать следует не только непосредственные результаты проекта, но и результаты в смежных секторах экономики.

Критерии социальной эффективности данного проекта были приведены в таблице 4.17.

Таблица 4.17 – Критерии социальной эффективности

ДО	ПОСЛЕ
Нехватка специализированных инструментов для планирования отпусков	Появление веб-приложения, обеспечивающего весь необходимый функционал для управления отпусками в организации
Необходимость ручного отслеживания пересечений отпусков сотрудников и соблюдения Трудового кодекса РФ	Программа берет на себя все необходимые проверки корректности выбора отпусков сотрудников
Недостаточная цифровизация университета	Появление в электронной среде университета дополнительного сервиса, облегчающего работу его сотрудникам

4.4.2 Оценка сравнительной эффективности исследования

Оценить эффективность проекта возможно на основе расчета интегрального показателя эффективности научного исследования, для нахождения которого необходимо определить две величины: финансовую эффективность и ресурсоэффективность.

Для расчета интегрального показателя ресурсоэффективности была составлена таблица 4.18, в которой было проведено сравнение разрабатываемого проекта и двух аналогов исполнения.

Таблица 4.18 – Сравнительная оценка характеристик вариантов исполнения проекта

Критерии	Весовой коэффициент параметра	Текущий проект	Аналог 1	Аналог 2
1. Способствует росту производительности труда	0,1	5	2	4
2. Удобство в эксплуатации (соответствует требования потребителей)	0,15	5	1	4
3. Помехоустойчивость	0,15	3	4	3
4. Энергосбережение	0,2	4	4	3
5. Надежность	0,25	4	5	4
6. Материалоемкость	0,15	4	4	4
ИТОГО	1	25	20	22

Полученные значения интегрального показателя ресурсоэффективности: для текущего проекта 4,1; для первого аналога 3,6; для второго аналога – 3,65.

Результаты сравнения эффективности разработки были внесены в таблицу 4.19.

Таблица 4.19 – Сравнительная эффективность разработки

№ п/п	Показатели	Аналог	Разработка
1	Интегральный финансовый показатель разработки	0,98	0,94
2	Интегральный показатель ресурсоэффективности разработки	3,65	4,1
3	Интегральный показатель эффективности	3,72	4,36
4	Сравнительная эффективность вариантов исполнения	0,85	1

Выводы по разделу

В результате проведенной работы можно сделать вывод, что данный проект является эффективным. Так, для него был выбран сегмент рынка, который ещё не занят конкурентными разработками, при этом по сравнению с ними разрабатываемая система обладает более широким и специализированным

функционалом. Кроме того, в результате расчета чистого дисконтированного дохода и дисконтированного срока окупаемости можно было видеть, что проект способен приносить прибыль с возвращением инвестированных денег. Помимо этого, была проведена работа по инициации проекта, описанию его целей, результатов, организационной структуры и т.д., были выполнены SWOT-анализ и оценка рисков.

Тем не менее, были обнаружены и те моменты, на которые следует обратить внимание. Во-первых, при анализе степени готовности научного проекта к коммерциализации были выявлены недостаточные степень проработанности проекта и уровень имеющихся знаний у разработчика. Во-вторых, планируемый бюджет превышает значение бюджета из раздела ограничений проекта, что может быть связано со слишком большой растянутостью проекта во времени, влекущую увеличению затрат на заработную плату разработчика.

5 Социальная ответственность

Введение

Темой работы является разработка программы, позволяющей выполнять управление и синхронизацию отпусков для сотрудников и руководителей подразделений университета.

Соответственно, разрабатываемое веб-приложение предполагается использовать в университетах (в первую очередь – в ТПУ, но возможна адаптация проекта для применения в других вузах), а основные пользователи приложения – это сотрудники университетов, которые могут выбирать и просматривать свои отпуска, и их руководители, которые занимаются проверкой и утверждением выбранных отпусков.

Место выполнения проекта – город Томск. В свою очередь, расположение потенциальных пользователей может быть любым за счет использования облачных серверов для работы приложения.

Использование разрабатываемого проекта позволяет обеспечивать соблюдение норм Трудового кодекса РФ, так как при выборе отпусков сотрудниками и утверждении их руководителями производится автоматическая проверка на соответствие 19-й главы кодекса, посвященной отпускам. Благодаря этому возможно как соблюдать права сотрудников при назначении отпусков, так и облегчить работу руководителям, так как им больше не нужно проверять планируемые отпуска сотрудников вручную.

5.1 Правовые и организационные вопросы обеспечения безопасности

5.1.1 Правовые нормы трудового законодательства

В проектируемом приложении затрагиваются нормы, заложенные в главе 19 Трудового кодекса РФ «Отпуска». Так, согласно статье 115 все сотрудники

должны иметь ежегодный оплачиваемый отпуск продолжительностью 28 календарных дней. Количество дней может быть увеличено как дополнительный оплачиваемый отпуск за работу сотрудников с вредными или опасными условиями труда, за особый характер их работы, ненормированный рабочий день и в ряде других случаев, предусмотренных ТК РФ и иными федеральными законами [1].

При формировании календаря отпусков также следует придерживаться ряда правил, закрепленных в статьях Трудового кодекса. Например, отпуска сотрудника могут быть разделены на несколько отдельных периодов, но как минимум один из этих периодов не должен иметь продолжительность менее 14 календарных дней (статья 125). Кроме того, при вычислении количества использованных сотрудником отпускных дней необходимо учитывать праздничные дни, так как в число календарных дней отпуска они не включаются.

Согласно статьям 124, 125 и 126, отпуск сотрудника может быть перенесен, продлен, разделен, прекращен либо заменен денежной компенсацией, но в этих случаях необходимо получение согласия сотрудника. Кроме того, если в течение года сотрудником не была потрачена часть его отпускных дней, то она должна быть включена в число дней отпуска следующего года и обязательно использована.

5.1.2 Эргономические требования к правильному расположению и компоновке рабочей зоны

При разработке проекта необходимо выполнять требования ГОСТ 12.2.032-78 «ССБТ. Рабочее место при выполнении работ сидя. Общие эргономические требования». Согласно этому стандарту, компоновка рабочей зоны должна обеспечивать возможность легкой досягаемости наиболее часто выполняемых операций, как и их расположение в пределах оптимальной зоны моторного поля. Указан ряд значений, которые должны быть достигнуты на рабочем месте: например, высота сиденья должны быть в пределах от 400 мм

(женщины) до 430 мм (мужчины), расстояние от сиденья до нижнего края рабочей поверхности не менее 150 мм, высота пространства для ног не менее 600 мм.

Также согласно стандарту ГОСТ Р ИСО 9241-5-2009 «Эргономические требования к проведению офисных работ с использованием видеодисплейных терминалов (VDT). Часть 5» на рабочем месте должны быть обеспечена поза, при которой позвоночник расположен вертикально, отсутствует скручивание верхней части туловища, линия зрения заключена между горизонталью и 60 градусами ниже горизонтали, а плечи сотрудника расположены вертикально, а предплечья – горизонтально.

Офисная мебель должна быть удобной и иметь средства регулировки, позволяющие достигнуть оптимальной рабочей позы, причем эти средства должны приводиться в действие из обычной рабочей позы, они не должны требовать приложения чрезмерных усилий или предварительного обучения и специальных инструментов.

5.2 Производственная безопасность

5.2.1 Анализ выявленных вредных и опасных факторов

Перечень выявленных вредных и опасных факторов, характерных для среды, в которой выполняется разработка проектируемого приложения, представлен в таблице 5.1.

Таблица 5.1 – Возможные опасные и вредные производственные факторы на рабочем месте при разработке проекта

Факторы (ГОСТ 12.0.003-2015)	Нормативные документы
Повышенный уровень шума	ГОСТ 12.1.003-2014 ССБТ. Шум. Общие требования безопасности [41]
Отсутствие или недостаток необходимого естественного освещения	СП 52.13330.2016 Естественное и искусственное освещение. Актуализированная редакция СНиП 23-05-95 [33]

Отсутствие или недостатки необходимого искусственного освещения	СП 52.13330.2016 Естественное и искусственное освещение. Актуализированная редакция СНиП 23-05-95 [33]
Повышенная яркость света	ГОСТ 12.0.003-2015 ССБТ Опасные и вредные производственные факторы. Классификация [34]
Умственное напряжение, в том числе вызванное информационной нагрузкой	МР 2.2.9.2311 – 07 «Профилактика стрессового состояния работников при различных видах профессиональной деятельности» [35]
Эмоциональные перегрузки	МР 2.2.9.2311 – 07 «Профилактика стрессового состояния работников при различных видах профессиональной деятельности» [35]
Опасность поражения электрическим током	ГОСТ 12.1.038-82 ССБТ. Электробезопасность. Предельно допустимые значения напряжений прикосновения и токов [36]

5.2.1.1 Повышенный уровень шума

Уровень шума при работе над проектом может быть повышен из-за ряда причин, таких как шум проезжающих мимо помещения машин, уличные мероприятия с использованием громкой музыки, разговоры людей, находящихся в помещении, где ведется разработка, и так далее.

Данный фактор является вредным и оказывает раздражающее влияние, приводящее к повышению утомляемости сотрудника, раздражительности, увеличивает вероятность возникновения ошибок при работе над проектом, что может привести к замедлению работы и негативным последствиям в качестве итоговой разработки, так как её создание требует повышенной сосредоточенности и внимания. Кроме того, длительное воздействие шума способно привести к тугоухости работника и увеличению риска заболеваний нервной и сердечно-сосудистой системы.

Нормативные значения звукового давления и уровней звука на рабочих местах устанавливаются СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания», где допустимые значения определяются типом

помещения, видом источника шума (постоянный либо непостоянный шум) и частотой звука (для источников постоянного шума). Так, для данной разработки подходят типы помещений «учебные кабинеты» и «жилые комнаты квартир». Для них допустимые уровень звука и эквивалентный уровень звука равны 40 дБА (при условии времени суток от 7 до 23 ч. для жилых комнат), а уровни звукового давления лежат в диапазоне от 79 дБ (31,5 Гц) до 28 дБ (8000 Гц) [32].

Для защиты от шумового воздействия могут применяться средства индивидуальной защиты (наушники, беруши). Возможна защита от уличного шума, если она была изначально заложена при строительстве зданий, в которых находится помещение для разработки, посредством использования материалов, уменьшающих распространение шума.

5.2.1.2 Отсутствие или недостаток необходимого естественного и искусственного освещения, повышенная яркость света

Возникновение недостатка необходимого освещения либо повышенной его яркости может быть во время работы над проектом как постоянным фактором (например, из-за планировки помещения и недостаточной обеспеченности светильниками для искусственного освещения), так и временным, связанным с периодическим отсутствием источника света либо изменением его положения. Например, при работе в пасмурную погоду естественное освещение значительно уменьшается, тогда как ночью оно практически отсутствует; с другой стороны, при определенном положении рабочего места и окон в какое-то время суток солнце может находиться напротив глаз сотрудника, тем самым ослепляя его.

Недостаток освещения и повышенная яркость света являются вредным фактором, способным нанести вред зрению сотрудника и повысить утомляемость при выполнении работы, так как сотрудник вынужден напрягать глаза, чтобы видеть объекты на рабочем месте либо на экране.

В СП 52.13330.2016 «Естественное и искусственное освещение. Актуализированная редакция СНиП 23-05-95» приведены показатели освещения для различных видов помещений; в данном случае может быть использована категория «Кабинеты информатики, компьютерные кабинеты», для которой указаны следующие значения [33]:

1. Высота плоскости над полом: 0,8 м, горизонтальная.
2. Разряд и подразряд зрительной работы: А-2.
3. Средняя освещенность рабочих поверхностей при общем освещении: 400 лк.
4. Равномерность распределения освещенности U_0: не менее 0,6.
5. Объединенный показатель дискомфорта R_{UG}: не более 14.
6. Коэффициент пульсации освещенности $К_п$: не более 10%.
7. Индекс цветопередачи источников света R_a: 90.
8. КЕО при совмещенном боковом освещении: 0,7.

Само помещение, используемое для разработки проекта, и его элементы освещения имеют следующие параметры: длина 8м, ширина 4м, площадь 32 м^2, высота потолка 2,75 м, высота рабочей поверхности 0,75 м, тип светильников – шар молочного стекла (ШМ) с одной лампой накаливания.

Произведем расчет системы освещения для этого помещения, при котором она будет удовлетворять требованиям стандартов, а именно обеспечивать освещенность в 400 лк.

В первую очередь, необходимо рассчитать количество светильников, которые возможно расположить в помещении. Для этого, во-первых, рассчитывается высота светильника над рабочей поверхностью согласно формуле (5.1):

$$h = H - h_p - h_c, \quad (5.1)$$

где:

H – высота потолка в помещении, м;

h_p – расстояние от пола до рабочей поверхности стола, м;

h_c – расстояние от потолка до светильника, м.

Следовательно, высота светильника над рабочей поверхностью для рассматриваемого помещения равна:

$$h = 2{,}75 - 0{,}75 - 0{,}5 = 1{,}5 \text{ м}$$

Во-вторых, требуется найти расстояние между рядами светильников L по формуле (5.2):

$$L = \lambda \cdot h, \quad (5.2)$$

Величина λ для выбранного типа светильника (ШМ) равна 2,3, поэтому L равно 3,45 м.

Количество рядов светильников можно вычислить по формуле (5.3).

$$n_{\text{ряд}} = \frac{B - \frac{2L}{3}}{L} + 1, \quad (5.3)$$

где:

B – ширина помещения, м.

Ширина помещения – 4 м. Соответственно, количество рядов светильников равно:

$$n_{\text{ряд}} = \frac{4 - \frac{2 \cdot 3{,}45}{3}}{3{,}45} + 1 = 1{,}49$$

Из полученного значения следует, что в помещении возможно разместить один ряд светильников.

Количество светильников в одном ряду возможно рассчитать по формуле (5.4).

$$n_{\text{св}} = \frac{A - \frac{2L}{3}}{l_{\text{св}} + 0.5} + 1, \quad (5.4)$$

где:

A – длина помещения, м;

$l_{\text{св}}$ – длина светильника, м.

Если размер плафонов светильников имеет диаметр 0,35 м, а расстояние между ними по длине помещения принять 0,5 м (как указано в формуле), то в помещении с длиной 8 м можно разместить 8 таких светильников в одном ряду:

$$n_{\text{св}} = \frac{8 - \frac{2 \cdot 3{,}45}{3}}{0{,}35 + 0{,}5} + 1 = \frac{5{,}7}{0{,}85} + 1 = 7{,}7$$

Так как ряд всего один, и в каждом светильнике есть только одна лампа, то общее количество светильников и ламп также равно восьми.

Световой поток лампы определяется по формуле (5.5):

$$\Phi = \frac{E_н \cdot S \cdot К_з \cdot Z}{N \cdot \eta}, \quad (5.5)$$

где:

$E_н$ – нормируемая минимальная освещенность по СНиП 23-05-95, лк;

S – площадь освещаемого помещения, м^2;

$К_з$ – коэффициент запаса;

Z – коэффициент неравномерности освещения;

N – число ламп в помещении;

μ – коэффициент использования светового потока.

Для нахождения коэффициента использования светового потока необходимо рассчитать индекс помещения, для чего используется формула (5.6):

$$i = \frac{S}{h(A+B)}, \quad (5.6)$$

где:

S – площадь помещения, м^2;

h – высота светильника над рабочей поверхностью, м;

A – длина помещения, м;

B – ширина помещения, м.

Таким образом, становится возможным вычислить индекс помещения:

$$i = \frac{32}{1{,}5 \cdot (8+4)} = 1{,}78$$

Значения коэффициентов отражения потолка и стен были приняты равными 50% и 30% соответственно, так как потолок – побеленный, но уже потерявший чистый белый цвет, а на стены наклеены обои светло-бежевого цвета. Для типа светильника ШМ сочетание полученных значений индекса помещения и коэффициентов отражения потолка и стен дает значение коэффициента использования светового потока, равное 0,25.

Теперь возможно рассчитать световой поток лампы:

$$Ф = \frac{400 \cdot 32 \cdot 1{,}2 \cdot 1{,}15}{8 \cdot 0{,}25} = 8832 \text{ (лм)}$$

Для расчета электрической мощности всей осветительной системы нужно взять стандартную лампу (в данном случае – лампу накаливания) с ближайшим значением светового потока. Так, подходит лампа Г215-225-500-1 номинальной мощностью 500 Вт и световым потоком 8400 лм.

Теперь необходимо проверить световой поток лампы по формуле (5.7).

$$-10\% \leq \frac{Ф_{л.станд} - Ф_{л.расч}}{Ф_{л.станд}} \cdot 100\% \leq +20\%, \quad (5.7)$$

где:

$Ф_{л.станд}$ – световой поток стандартной лампы, лм;

$Ф_{л.расч}$ – полученное расчетное значение светового потока для помещения, лм.

В результате расчета было получено значение -5%, не выходящее за указанные пределы. Так как ламп 8, и каждая из них имеет мощность в 500 Вт, то электрическая мощность осветительной установки, требуемой для рассматриваемого помещения при использовании светильников ШМ, равна 4000 Вт.

5.2.1.3 Умственное напряжение

Согласно ГОСТ 12.0.003-2015 «Опасные и вредные производственные факторы», умственное напряжение является одним из видов нервно-психической перегрузкой, который может быть вызван информационной нагрузкой [34].

При разработке проекта работнику приходится решать большое количество сложных задач, связанных с проектированием будущей программы, созданием и реализацией различных алгоритмов. В связи с этим, а также с внутренней сложностью программы и предметной области, разработчик вынужден постоянно держать в голове большое количество информации и предусматривать разнообразные ситуации при использовании программы, что в конечном итоге приводит к умственному перенапряжению.

Негативным последствиями данного вредного фактора могут быть как эффекты, возникающие в момент или после напряжения (например, головная боль, усталость, снижение внимания и концентрации, раздражительность), так и заболевания: гипертоническая болезнь, ишемическая болезнь, невротические расстройства.

Для того, чтобы избежать умственного перенапряжения либо снизить его негативное влияние следует делать перерывы в работе. Так, в МР 2.2.9.2311-07 «Профилактика стрессового состояния работников при различных видах профессиональной деятельности» указывается, что в течение рабочего дня должны быть предусмотрены три перерыва: обеденный перерыв продолжительностью 30 минут и два перерыва продолжительностью 5-7 минут каждый, которые следует проводить через 2 часа после начала рабочей смены и за 2 часа до её конца [35]. В первый из таких перерывов целесообразно повышение двигательной активности за счет выполнения физических упражнений; второй перерыв следует использовать для психологической разгрузки.

5.2.1.4 Эмоциональные перегрузки

Вероятность возникновения эмоциональных перегрузок связана с такими особенностями работы, как высокий уровень ответственности за её результат, наличие жестких крайних сроков её выполнения, большой объём работы, различные факторы неопределенности и т.д.

Так же, как и умственное напряжение, эмоциональные перегрузки оказывают влияние как на самочувствие сотрудника и результат его работы, так и на его здоровье в целом. Согласно МР 2.2.9.2311-07 «Профилактика стрессового состояния работников при различных видах профессиональной деятельности», чем выше напряженность труда работников, тем выше вероятность возникновения ишемической и гипертонической болезней и невротических расстройств: например, при малой напряженности труда

вероятность возникновения невротических расстройств у мужчин равен 0, а у женщин меньше 20%, тогда как при изнурительно напряженной работе (V категория напряженности труда) у мужчин этот показатель вырастает до 35-43,9%, а у женщин – до 61,4-70,3% [35].

Для снижения эмоционального напряжения рекомендуется включение в распорядок дня индивидуальных сеансов психологической разгрузки, которым сотрудник должен быть обучен под руководством инструктора или психоневролога. Сами сеансы должны иметь продолжительность 10-20 минут и проводиться в одни и те же время и месте (в специальной комнате, собственном рабочем кабинете).

В качестве психологической разгрузки могут использоваться психическая саморегуляция, аутогенная тренировка, функциональная музыка, коррекция функционального состояния с помощью фильмов, дыхательные упражнения.

5.2.1.5 Опасность поражения электрическим током

Опасность поражения электрическим током может возникнуть в процессе работы над проектом, так как он тесно связан с использованием компьютера: именно на компьютере (ноутбуке) ведется вся разработка программы, и он также должен быть периодически подключен к электрической сети для зарядки аккумулятора.

Поражение электрическим током является опасным фактором, имеющим различные виды воздействия: термическое (появление ожогов, нагрев до высокой температуры кровеносных сосудов, нервов и внутренних органов на пути тока), электролитическое (разложение органической жидкости, в том числе крови), механическое (разрыв, расслоение и другие повреждения тканей организма), биологическое (нарушение внутренних биоэлектрических процессов). Все эти воздействия по отдельности и в совокупности способны нанести вред здоровью человека или к его смерти.

Предельно допустимые значения напряжений прикосновения и токов регламентируются стандартом ГОСТ 12.1.038-82 ССБТ. «Электробезопасность. Предельно допустимые значения напряжений прикосновения и токов». Например, напряжения прикосновения и токи, протекающие через тело человека, не должны превышать значений: 2 В и 0,3 мА для переменного тока частотой 50 Гц, 3 В и 0,4 мА для переменного тока 400 Гц и 8 В и 1 мА для постоянного тока.

Для того, чтобы избежать поражения электрическим током, нельзя касаться электроприборов или проводов мокрыми руками, пользоваться неисправными устройствами либо пробовать их починить самостоятельно, а также трогать провода с нарушениями изоляции.

5.3 Экологическая безопасность

Данная разработка является программным обеспечением и не связана с процессом производства, поэтому большинство возможных негативных воздействий на окружающую среду, а именно воздействия на атмосферу, гидросферу и селитебную зону отсутствуют.

С другой стороны, возможно воздействие на литосферу в виде загрязнения почты, связанное с утилизацией компьютера и периферийных устройств, таких как принтеры, МФУ, веб-камеры, наушники, колонки, телефоны. При этом компьютерная техника, утратившая потребительские свойства, относится к IV классу опасности (малоопасные отходы).

Согласно ГОСТ Р 56397-2015 «Техническая экспертиза работоспособности радиоэлектронной аппаратуры, оборудования информационных технологий, электрических машин и приборов», компьютерное оборудование подлежит списанию и утилизации, если оно было признано в результате технической экспертизы не ремонтпригодным и неработоспособным; аналогичный вывод выносится в случае деградационного отказа оборудования и нецелесообразности его ремонта и модернизации.

Обращение с отходами регламентируется ГОСТ Р 53692-2009 «Ресурсосбережение. Обращение с отходами». Согласно этому стандарту, при ликвидации любого изделия (в том числе и рассматриваемых отходов электронной промышленности) необходимо осуществить ряд мероприятий: снятие с эксплуатации, обезвреживание (при необходимости) и списание с передачей его на утилизацию или удаление. При этом является важным обеспечение максимального возвращения материальных и энергетических ресурсов от ликвидируемых объектов или отходов в сферу хозяйственной деятельности, которое может осуществляться в виде повторного использования отдельных элементов или как сырье для другого или аналогичного производства [37]. Таким образом возможно не только сэкономить ресурсы, но и сохранить окружающую среду от дополнительного загрязнения.

Помимо компьютерной техники, при выполнении проекта возможно использование бумаги (а также различных упаковочных материалов, сделанных также из бумаги либо картона), которое означает появление соответствующих твердых бытовых отходов – макулатуры. Данный вид отходов затрагивается в стандарте ГОСТ Р 55090-2012 «Ресурсосбережение. Обращение с отходами. Рекомендации по утилизации отходов бумаги», где перечислены различные меры для повторного использования бумаги и предотвращения неблагоприятного экологического воздействия при производстве бумаги и утилизации макулатуры [38].

Еще один возможный источник загрязнения литосферы – это люминесцентные лампы, опасность которых связана с содержанием в них ртути. Подобные отходы перерабатываются в соответствии с ГОСТ Р 52105-2003 «Ресурсосбережение. Обращение с отходами. Классификация и методы переработки ртутьсодержащих отходов». Так, они проходят переработку на специализированных предприятиях, где с помощью специальных методов удается получить из отходов металлическую ртуть, так что она может быть использована повторно, не загрязняя окружающую среду. В этом же стандарте можно увидеть, что ртутьсодержащие элементы от переработки

люминесцентных ламп (люминофор, ступа) перерабатываются с помощью термических методов, заключающихся в прогревании или прокаливании в установке, приспособленной для испарения и конденсации паров ртути [39].

5.4 Безопасность в чрезвычайных ситуациях

К числу ЧС, которые могут возникнуть в ходе работы над данным проектом, относятся природные катастрофы (ураганы, наводнения), геологические воздействия (землетрясения) и техногенные аварии (пожары, ЧС с выбросом радиоактивных веществ на СХК в городе Северск).

ЧС, возникновение которой представляется наиболее вероятной, является пожар. Источниками такого пожара могут быть электронные устройства (например, рабочий компьютер), пожаровзрывоопасные вещества в помещении либо здании, где ведется разработка, а также неисправная электропроводка.

Меры по предупреждению такой ЧС возможно найти в ГОСТ 12.1.004-91 ССБТ. «Пожарная безопасность. Общие требования». Так, требуется максимально ограничить количество пожаровзрывоопасных веществ в помещении, причем размещены они также должны наиболее безопасным способом; поддерживать температуру и давление среды, при которых исключается распространение пламени; применять электрооборудование, соответствующие требованиям пожарной безопасности [40].

В случае обнаружения признаков возникновения пожара (задымление, запах гари, появление открытого пламени) в первую очередь необходимо вызвать противопожарную службу и отключить электроэнергию в помещении. Если очаг возгорания небольшой и в помещении имеется огнетушитель, то возможно попробовать самостоятельно потушить возгорание; в противном случае (либо если потушить пожар не удалось) необходимо покинуть помещение и здание согласно плану эвакуации. Пример плана эвакуации для использованной аудитории 113Б в 10-м корпусе ТПУ приведен на рисунке 5.1.

Рисунок 5.1 – План эвакуации из помещения при пожаре и других ЧС

Выводы по разделу

В результате выполнения раздела социальной ответственности были найдены и изучены нормы и стандарты, регулирующие вопросы обеспечения безопасности при разработке проекта; выявлены вредные и опасные факторы, которые могут воздействовать на работника в процессе разработки, а также соответствующие меры по защите от действия этих факторов; определено негативное воздействие на окружающую среду, а также рассмотрены наиболее вероятные ЧС. Кроме того, в процессе изучения фактора недостаточности освещения были произведены расчеты такого освещения для рабочего помещения, которое будет удовлетворять стандартам.

Само рабочее помещение относится к категории помещений без повышенной опасности, так как в нем отсутствуют условия, создающие повышенную или особую опасность. Группа персонала по электробезопасности – I, так как разработчик проекта не относится к электротехническому и электротехнологическому персоналу. По уровню энергозатрат организма программирование (как в данном проекте) относится к категории работ Iа, так как в его процессе не требуется работать стоя, ходить или совершать физические нагрузки. По взрывопожарной и пожарной опасности категория рабочего помещения – Д (пониженная пожароопасность), так как в нем находятся только негорючие вещества и материалы в холодном состоянии; само здание, в котором расположено рабочее помещение, относится к IV категории объектов, оказывающих негативное воздействие на окружающую среду, так как отсутствуют те загрязняющие факторы, которые указаны для других категорий.

Заключение

В результате выполнения выпускной квалификационной работы была создана система для управления и согласованного планирования отпусков сотрудников подразделений университета (ТПУ). Функционал разработанного веб-приложения включает в себя такие возможности, как:

1. Выбор желаемых периодов отпусков для сотрудников, их редактирование, оставление заявок на перенос уже утвержденных отпусков.
2. Просмотр числа назначенных отпускных дней для сотрудников, функция их назначения для руководителя с выбором причины.
3. Создание правил выбора отпусков внутри подразделений для возможности автоматической проверки пересечений отпусков сотрудников и числа сотрудников определенных должностей на рабочем месте в каждый день года.
4. Визуализация отпусков в виде цветного календаря с фильтрацией по году, периоду, типу отпусков (желаемые и уже утвержденные).
5. Поддержка Трудового кодекса РФ: автоматическое назначение ежегодного основного оплачиваемого отпуска, проверка длительности периодов отпусков (при разбиении как минимум один период должен длиться не менее 14 дней), учет неиспользованных отпускных дней с прошлого года.
6. Получение руководителями уведомлений о приближающемся начале и конце отпуска подчиненных сотрудников.
7. Панель администратора для проверки корректности обработки данных из JSON-сообщений.

В процессе проектирования и разработки системы нужно было решить множество задач. Например, одна из таких задач была связана с потенциальным практическим применением веб-приложения в университете, что означает поддержку возможности загрузки данных из API. Следовательно, при

реализации системы было предусмотрено, что при подключении к API (а не использовании локальных файлов в формате JSON, как было сделано в процессе разработки) нужно будет внести правки в код не всей программы, а только нескольких отдельных методов.

Ещё одна важная задача, которая была решена – изучение предметной области и отражение связанных с ней требований в функционале программы. Так, с одной стороны, планирование отпусков тесно связано с нормами Трудового кодекса РФ. С другой стороны, есть также другие особенности, выходящие за рамки кодекса, которые стоило учесть в работе веб-приложения: например, именно для этого были созданы правила выбора отпусков, благодаря заданию которых руководителю не нужно вручную проверять, какие пересечения отпусков недопустимы для работы его подразделения.

Таким образом, применение разработанного веб-приложения действительно способно облегчить процесс согласованного планирования отпусков в подразделениях университета за счет реализованного функционала, специализирующегося на управлении отпусками, а также потенциальной возможности подключения к API ТПУ для использования в приложении реальных данных, хранящихся на сервере университета.

Conclusion

It was necessary to develop an information system for managing and planning vacation leaves of employees of departments of the university (TPU), and this goal has been successfully achieved during the accomplishment of this graduation qualification work. The functionality of the web application includes:

1. Choosing desired vacation leave periods for employees, as well as editing them and making the requests for shifting or cancelling the leaves, which have been already approved.
2. Viewing an amount of vacation days of employees. Also, the heads can add or remove those days for their workers.
3. Creating the rules for departments, which are used for limiting the employees' leaves and are automatically checked by the system, when a head of a department tries to approve a leave.
4. Visualization of the leaves in form of a colored calendar, where filters by a year, a period or a type of leaves can be used.
5. Supporting the laws of Labor Code of the Russian Federation: adding 28 leave days to all the employees automatically, checking the duration of the leaves, remembering leave days which have not been used past year.
6. Sending the notifications to the heads of departments about an upcoming start or an end of the leaves of their workers.
7. Administration panel for checking if all the data received from the API of TPU have been handled correctly.

There were a lot of various tasks to solve during the process of application design and development. For example, this web application is supposed to be used in TPU, what means that it must be able to get the needed data from API of TPU. Therefore, the architecture of the program allows adding this feature: in order to connect to the API, it is needed only to edit the code of several methods, not the whole program.

Another important task, which has been solved, is studying domain knowledge and reflecting it in the software requirements. On the one hand, the process of planning the leaves is connected with the Labor Code of the Russian Federation. On the other hand, there are features of departments and the university in a whole, which were needed to be considered during the process of creating the application. For example, the rules for vacation leaves were created for this purpose, so that a head of a department does not need to check what intersections of their workers' leaves are not allowed.

In conclusion, this web application is actually able to make the process of planning and managing the leaves in the departments of the university easier and more comfortable thanks to the implemented functionality, which is specialized on the leaves, as well as because of potential ability to be connected to the API of TPU, so it will be able to use the relevant data from the server of the university.

Список публикаций и научных достижений

1. Диплом за победу в хакатоне Счетной палаты Российской Федерации Audithon 2021 в специальной номинации «Нестандартное решение» 25-28 марта 2021 г. (Приложение Г).

Список использованных источников

1. Трудовой кодекс Российской Федерации от 30.12.2001 N 197-ФЗ (ред. от 09.03.2021).
2. Использование диаграммы вариантов использования UML при проектировании программного обеспечения / Хабр [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/566218/, свободный (дата обращения: 25.05.2022 г.).
3. What is Web Application (Web Apps) and its Benefits [Электронный ресурс] – Режим доступа: https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app, свободный (дата обращения: 25.05.2022 г.).
4. API простым языком: что это и зачем нужен | РБК Тренды [Электронный ресурс] – Режим доступа: https://trends.rbc.ru/trends/industry/614b2abe9a79476f5b552e0e, свободный (дата обращения: 25.05.2022 г.).
5. What is an API? (Application Programming Interface) | MuleSoft [Электронный ресурс] – Режим доступа: https://www.mulesoft.com/resources/api/what-is-an-api, свободный (дата обращения: 25.05.2022 г.).
6. Introducing JSON | JSON [Электронный ресурс] – Режим доступа: https://www.json.org/json-en.html, свободный (дата обращения: 25.05.2022 г.).
7. JSON vs XML [Электронный ресурс] – Режим доступа: https://www.w3schools.com/js/js_json_xml.asp, свободный (дата обращения: 25.05.2022 г.).
8. Что такое XML / Хабр [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/524288, свободный (дата обращения: 30.05.2022 г.).

9. JSON и XML. Что лучше? / Хабр [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/31225, свободный (дата обращения: 30.05.2022г.).
10. Most used web frameworks among developers 2021 | Statista [Электронный ресурс] – Режим доступа: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web, свободный (дата обращения: 25.05.2022 г.).
11. The web framework for perfectionists with deadlines | Django [Электронный ресурс] – Режим доступа: https://www.djangoproject.com/, свободный (дата обращения: 25.05.2022 г.).
12. Django Advantages and Disadvantages – Why You Should Choose Django? – DataFlair [Электронный ресурс] – Режим доступа: https://data-flair.training/blogs/django-advantages-and-disadvantages/, свободный (дата обращения: 25.05.2022 г.).
13. Spring Framework | Spring по-русски! [Электронный ресурс] – Режим доступа: http://spring-projects.ru/projects/spring-framework/, свободный (дата обращения: 25.05.2022 г.).
14. Java Spring Pros and Cons – Javatpoint [Электронный ресурс] – Режим доступа: https://www.javatpoint.com/java-spring-pros-and-cons, свободный (дата обращения: 25.05.2022 г.).
15. Общие сведения об ASP.NET Core | Microsoft Docs [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0, свободный (дата обращения: 25.05.2022 г.).
16. Spring Boot vs ASP.NET Core: A Showdown | Medium [Электронный ресурс] – Режим доступа: https://medium.com/@putuprema/spring-boot-vs-asp-net-core-a-showdown-1d38b89c6c2d, свободный (дата обращения: 30.05.2022 г.).

17. Язык программирования Java в 2021 году. Сферы применения и как стать Java-разработчиком [Электронный ресурс] – Режим доступа: https://brunoyam.com/blog/programmirovanie/vse-pro-java-v-2021-godu, свободный (дата обращения: 30.05.2022 г.).
18. ASP.NET MVC 5 | Введение в создание сайтов [Электронный ресурс] – Режим доступа: https://metanit.com/sharp/mvc5/1.1.php, свободный (дата обращения: 30.05.2022 г.).
19. Difference between MVC and MVT architecture [Электронный ресурс] – Режим доступа: https://www.geekinsta.com/difference-between-mvc-and-mvt, свободный (дата обращения: 30.05.2022 г.).
20. ASP.NET Core | Введение [Электронный ресурс] – Режим доступа: https://metanit.com/sharp/aspnet5/1.1.php, свободный (дата обращения: 25.05.2022 г.).
21. Язык программирования C#: история, специфика, место на рынке | GeekBrains – образовательный портал [Электронный ресурс] – Режим доступа: https://gb.ru/posts/yazyk-programmirovaniya-c-sharp-istoriya-specifika-mesto-na-rynke, свободный (дата обращения: 25.05.2022 г.).
22. Система управления базами данных MySQL [Электронный ресурс] – Режим доступа: https://depix.ru/articles/sistema_upravleniya_bazami_dannyh_mysql, свободный (дата обращения: 25.05.2022 г.).
23. Сегментация рынка - критерии определения ЦА - Сбербанк [Электронный ресурс] – Режим доступа: https://www.sberbank.ru/ru/s_m_business/pro_business/segmentaciya-rynka-kriterii-opredeleniya-celevoj-auditorii/, свободный (дата обращения: 07.05.2022 г.).
24. Что такое Jira и как с ней работать - База Знаний Timeweb Coomunity [Электронный ресурс] – Режим доступа: https://timeweb.com/ru/community/articles/kak-rabotat-v-jira, свободный (дата обращения: 07.05.2022 г.).

25. Планируем отпуска и отгулы не выходя из Jira [Электронный ресурс] – Режим доступа: https://www.teamlead.ru/pages/viewpage.action?pageId=74186779, свободный (дата обращения: 07.05.2022 г.).
26. 1С:Зарплата и управление персоналом 8 | О продукте [Электронный ресурс] – Режим доступа: https://v8.1c.ru/hrm/, свободный (дата обращения: 07.05.2022 г.).
27. Что такое SWOT-анализ. Объясняем простыми словами — Секрет фирмы [Электронный ресурс] – Режим доступа: https://secretmag.ru/enciklopediya/chto-takoe-swot-analiz-obyasnyaem-prostymi-slovami.htm, свободный (дата обращения: 07.05.2022 г.).
28. Иерархическая структура работ (ИСР) - Управление проектами [Электронный ресурс] – Режим доступа: https://forpm.ru/иерархическая-структура-работ-иср/, свободный (дата обращения: 07.05.2022 г.).
29. Что такое диаграмма Ганта? | Atlassian [Электронный ресурс] – Режим доступа: https://www.atlassian.com/ru/agile/project-management/gantt-chart, свободный (дата обращения: 07.05.2022 г.).
30. Дисконтированный (дисконтируемый) срок окупаемости: что это такое, как рассчитать величину периода DPP по формуле, простой расчет для проекта с учетом дисконтирования и без, как посчитать показатели [Электронный ресурс] – Режим доступа: https://www.cleverence.ru/articles/finansy/diskontirovannyy-srok-okupaemosti-chto-eto-takoe-i-kak-pravilno-rasschitat-pokazateli-metod-i-formul/, свободный (дата обращения: 07.05.2022 г.).
31. ГОСТ Р ИСО 9241-5-2009 Эргономические требования к проведению офисных работ с использованием видеодисплейных терминалов (VDT). Часть 5.

32. СанПиН 1.2.3685-21 Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания.
33. СП 52.13330.2016 Естественное и искусственное освещение. Актуализированная редакция СНиП 23-05-95.
34. ГОСТ 12.0.003-2015 Опасные и вредные производственные факторы.
35. МР 2.2.9.2311 – 07 Профилактика стрессового состояния работников при различных видах профессиональной деятельности.
36. ГОСТ 12.1.038-82 ССБТ. Электробезопасность. Предельно допустимые значения напряжений прикосновения и токов.
37. ГОСТ Р 53692-2009 Ресурсосбережение. Обращение с отходами.
38. ГОСТ Р 55090-2012 Ресурсосбережение. Обращение с отходами. Рекомендации по утилизации отходов бумаги.
39. ГОСТ Р 52105-2003 Ресурсосбережение. Обращение с отходами. Классификация и методы переработки ртутьсодержащих отходов.
40. ГОСТ 12.1.004-91 ССБТ. Пожарная безопасность. Общие требования.
41. ГОСТ 12.1.003-2014 ССБТ. Шум. Общие требования безопасности.

Приложение А

Analysis of technologies. Software requirements and design

Студент:

Группа	ФИО	Подпись	Дата
8ВМ02	Шкулов Никита Петрович		

Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИТ	Фадеев А.С.	к.т.н		

Консультант – лингвист ОИЯ ШБИП:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОИЯ	Диденко А.В.	к.ф.н.		

1 Analysis of technologies

According to the software requirements, taken from potential users of the program, the program must be a web application interacting with API (Application Programming Interface) of TPU for getting the needed data and also having its own database for storing the data about the employees' vacation leaves. Moreover, another essential requirement for the program is being a multi-platform software. Both database and the web application itself must be able to be deployed and run on various operating systems like Linux.

A web application (web app) is an application program that is stored on a remote server and delivered over the Internet through a browser interface [1]. This type of programs is very comfortable and easily accessible for all the users. First of all, it is not needed to download and install such applications on a personal computer. Instead, the whole functionality of the web app can be accessed via any web browser, which even the users without much IT experience know how to use for internet surfing.

Generally, web applications have following advantages:

1. When the application gets a new version, it is not needed to solve the problem of delivering that version to all the users who installed the program on their PCs or smartphones, because they all use only one, the latest version, which is working on a remote server and serving the users' requests;
2. As it was said above, web apps are easily accessible for the users, because their functions are delivered via web browsers. On the other hand, it is an advantage for the developers as well, because, unlike the case of usual applications, they do not need to make different versions of the app for various existing and popular platforms and operating systems like Windows, macOS, Android etc. [2] Instead, the only thing the developers must care about is screen resolution of a user's device, but this problem is solved by many useful best practices and technologies of frontend development;

3. Since the web applications are running on remote servers, the process of their work mostly uses the resources of the server, not the user's device; in other words, such applications demand much lower technical requirements from the users' devices.

A workflow of a typical web application can be described by following steps [3]:

1. A user sends a request to a web server using web browser or the user interface of an app;
2. Then, the request is forwarded by the web server to the corresponding web application server;
3. The web application server processes the request: usually that means performing some data-related tasks (for example, adding, editing, deleting data objects) and then generating the appropriate results;
4. Results are sent from the web application server to the web server;
5. The web server responds back to the client, and the requested information is displayed to the user.

As it was said before, the interaction between the web application and the server of TPU is supposed to be done via API of TPU. API is a software consisting of methods, which forms an open interface used for providing interaction between programs [4].

API has to be used in this web application, because some of the needed data are already stored in the servers and databases of TPU. For example, these data include employees of the university, the personal and university-related data about them, departments, job positions, hierarchy of employees and departments. It is impossible to get all these data and save them in the inner database of the web application, because they may be updated in the university databases at any moment afterwards, so the data «inside» of the web app would always be considered outdated and unreliable.

There are many advantages of using API for getting the needed data from a remote server, for example:

1. API provides an additional layer of security. API is accessed only via a certain range of methods made for receiving or sending the data using predefined formats of messages. In other words, the application interacting with the API does not «see» the server, its database and inner algorithms. They are not exposed anyway, as well as how exactly the server transforms its own data into the messages that the client application receives. At the same time, the server knows nothing about the client with the exception of the request received earlier. Thereby, both sides of the interaction can be certain, that they do not reveal any important or confidential information to each other;
2. APIs are usually developed and treated not as a code, but as products. They are designed for specific users; they have their own documentation and also get new versions and updates. So, the users of API can have certain expectations of its maintenance and lifecycle [4];
3. APIs are standardized, so that they have much stronger discipline for security and governance, and they are monitored and managed for providing better performance and scale.

The communication between API and its clients is provided by messages of certain formats. The three most common formats are JSON, XML and YAML.

JSON (JavaScript Object Notation) and XML (EXtensible Markup Language) represent two opposite approaches of formatting the messages for software and machine communication. JSON messages are very plain and simple, and it is both strength and the weakness of this format.

On the one hand, they have light weight, so such messages are processed by programs faster; also, JSON is very easy to read and write. On the other hand, JSON messages provide much less functionality than XML. However, XML messages are heavy, slow and not understandable for a human [5]. Meanwhile, YAML («Yet Another Markup Language») is an attempt to create a format, which would have great functionality like XML and also would be lightweight and easy to read like JSON.

For the web application of this thesis the rich functionality of XML and YAML is not needed, so the advantages of JSON messages and the fact, that API of TPU uses exactly this format, were the reasons of decision to use JSON for the program.

JSON is built on two main structures [6]:

1. A collection of name/value pairs. In various languages, this data structure can be represented by dictionaries, objects, records, hash table, keyed lists etc.;
2. An ordered list of values. In most programming languages, this is realized as an array, vector, list, or sequence.

As it can be seen, both these structures are universal, because they are supported in all modern programming languages. Moreover, a lot of those languages also have special built-in or additional libraries designed for making the work with JSON even simpler and faster for developers.

When it comes to the choice of programming language for implementing the web app, it is mostly determined not by the language itself, but existing common frameworks for creating web applications. The three frameworks have been chosen for comparing: Spring, Django and ASP.NET Core. According to the statistical analysis, they are all among the most used web frameworks in 2021 [7].

Django is a high-level Python-based free and open-source web framework that encourages rapid development and clean and pragmatic design [8]. It has the following advantages [9]:

1. Django uses Python as a programming language. Python itself is an easy to use and read programming language, which has a big worldwide community of developers, wide range of educating materials and various libraries for extending its functionality;
2. This framework is made by experienced web developers for other web developers. In other words, people who made it know from their own experience, which parts or features of other existing web frameworks are comfortable and useful and which ones make the process of implementing

the applications harder. As a result, they prepared a lot of functionality in order to help developers to create the apps faster and more efficiently;

3. Django offers fast development. The architecture of Django projects is implemented with the philosophy of loosely coupled components. That means that it is possible to work on different modules of a web application in parallel (for example, by giving the task of implementing different modules to different team members), because they do not have strong connection between them and they can be integrated easily;
4. The previous point also means that modules of Django project can be used or edited independently. It is important in the world of IT, where re-using components is an effective way of improving the speed of creating applications and where sudden changes of requirements made by a customer are not that rare.

However, this framework has its own drawbacks. First of all, it is so-called «The Django way», a certain style of organizing and implementing web applications, which is obligatory to use and follow while making apps with Django. Any Django project has a certain set of files and pre-defined variables, and it is necessary to learn about them before creating any Django project. It must be said, that the logical file structure of Django is easy to learn, but it does not change the fact that developers cannot use their own file structures, because deploying anything using Django without following its rules is impossible.

Spring is one of the most popular Java frameworks for developing web applications. According to the official site of the framework, it has following features [10]:

1. It can use independently evolvable microservices;
2. Spring has asynchronous, nonblocking architecture, which allows a developer to use computing resources in a more efficient way;
3. It is possible to deploy an application to any cloud services;
4. The web applications made with Spring are fast, secure and responsible, and they can use any data store as well;

5. There is an opportunity to connect Spring web application to the business events of enterprise systems of a company;
6. Spring web applications are flexible, because they may be scaled up on demand and scaled to zero when there is no demand;
7. Spring supports automated tasks: offline processing of data at any suitable time.

On the other hand, Spring has several serious drawbacks as well [11]:

8. Spring is not an easy framework to use. It is needed to have a lot of programming experience before starting to use Spring, and learning the framework itself and its features is a long and difficult process;
9. Parallel mechanism of Spring is difficult to understand for developers who have never used this framework before, because it provides a lot of confusing options to choose. Meanwhile, wrong decisions may lead to serious negative consequences and delays;
10. There is a lot of XML, which is required for working with Spring, while this format, as it was mentioned before, is difficult to read and write.

ASP.NET Core is an open-source and cross-platform framework for building modern cloud-based internet connected applications, such as web apps, IoT apps and mobile backends [12]. This framework includes a part ASP.NET Core MVC implementing a design pattern Model-View-Controller, which is used for creating web applications.

MVC applications separate programs into three main groups of components: Models, Views and Controllers. The diagram of their interaction is shown in Figure 1 [13].

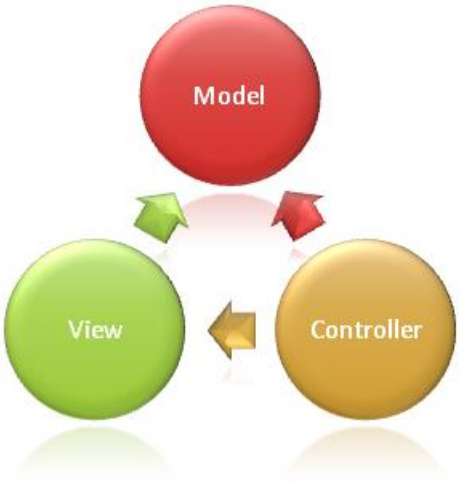


Figure 1 – Relationship between MVC components

The Model in an MVC application is used for representing the state of the application, the data and also any business logic or operations that should be performed by the application. For example, if the web application uses database as the source of the data, one of the ways of representing the tables in the program is directly creating the models for them. Besides, there is a special type of Models, which are called ViewModels and contain the data for views displaying.

Views are responsible for displaying content through the user interface. ASP.NET Core has a Razor view engine, which allows including C# code inside of plain HTML pages, giving a developer an opportunity to use constructions of that programming language like circles for creating web pages. Controllers handle user's requests, and they are responsible for selecting which model types to work with and which view to render.

Such a separation of responsibilities has an important advantage: having such a structure, it is easier to keep the layers (models, views and controllers) independent of each other, so in case of any changes of the code inside of them it is not needed to modify other layers. It helps to save time, prevents the need of re-testing the application after any minor change and makes the applications more flexible and scalable.

So, one of the advantages of ASP.NET Core framework is using MVC design pattern. Besides, there are even more of them:

1. Web applications created with ASP.NET Core are multi-platform. That means that they can be deployed not only on OS Windows, but also on macOS and Linux. It is important for this project as well, because one of software requirements is the support of OS Linux. Unlike paid and expensive operating systems such as Windows Server, Linux systems are mostly free, and many of them (for example, Ubuntu, Debian) are widely used as server operating systems because of their great performance and high level of security;
2. This framework provides an opportunity to use additional modules. ASP.NET Core has a lot of built-in libraries, which can be used for web application development, but, if it is not sufficient, it is possible to use the package manager NuGet, which includes even more libraries for solving various tasks. For example, there are Entity Framework Core, which adds the support of many data sources, and Newtonsoft.Json with useful classes and methods for parsing and creating JSON messages.

Finally, ASP.NET Core has been chosen for this project because of many reasons:

1. This framework supports MVC;
2. There is a big worldwide community of ASP.NET Core developers and a lot of educating materials, what means that in case of troubles it will be easier to find a solution in internet or in the books;
3. It is possible to deploy an application on Linux operating systems;
4. The developer of this project has more experience of using this framework in comparison with Spring and Django, what means better understanding of the framework and faster process of implementing the application with better quality.

2 Software Requirements

Requirements analysis is a phase of software development process, which is used to determine the needs and expectations of a program. This process involves frequent communication and interaction with the customer and potential users of the product in order to document all the key requirements, get complete understanding of the clients' needs, resolve current and potential conflicts and define expectations [14].

This process has a significant impact on the success of the ready software. Without knowing what the customer and the users expect from the program, there is a risk of creating a product, which nobody will actually need. For example, the lack of communication with the customer may result in developing the software, which does not solve the problems it was supposed to solve; in this case, the customers definitely will not be satisfied with the result they get for the money they paid.

On the other hand, the developers must understand the people who will use the program and their overall level of experience of using computer programs. Otherwise, the software may end up satisfying the client's needs, when it comes to its functionality, but it will be uncomfortable and hard to use for its potential users. This cannot be called a success either.

Therefore, several meetings with potential users (the heads of departments of TPU) have been organized, so that it was possible to find out their expectations and also get useful ideas about the desired functionality of the web application. Those requirements are represented in the list below:

1. The web application must provide a web interface with a calendar showing periods of vacation leaves of the employees of TPU departments. Also, that calendar must allow the heads of those departments to manage the leaves of their subordinates;
2. There must be an opportunity to connect the web application to the API of TPU for getting the current data about the departments, job positions, employees of the university, their mutual hierarchy and also the list of non-working days;

3. The web application must be able to handle correctly various types of employees and their mutual hierarchy. For example, there are different cases possible: a person may be an employee of a department of TPU or several departments; a person may be a head of one or more departments; a person may be a head of one department and an employee in another;
4. It must be possible to limit the combinations of the leave periods of employees. For example, a head of a department must have an opportunity to forbid two (or more) chosen employees to have their leaves simultaneously;
5. The web application must include notifications. Notifications are sent to the heads of departments, when the leaves of their subordinates are going to start or end soon. The employees receive notifications, when the heads of their departments apply any changes to their vacation leaves;
6. Employees must have an opportunity to make a request for changing (shifting, cancelling, interrupting) their leaves, which have been approved earlier;
7. It must be possible to create and manage groups of employees: named combinations of employees inside departments for making application for the vacation to them easier and more understandable.

The communication with potential users is important because of yet another reason. When developers create programs, they have to touch a certain domain knowledge which those programs belong to. There is a potential problem: a lot of things may be understandable and obvious for employees, who work using the domain knowledge of their job every day, but for developers they are not. So, when the program is created, there is a possibility of mistakes caused by the lack of knowledge of the developers about the domain knowledge.

Besides speaking with the users, it is possible to get information about the domain knowledge from various documents, books, sites etc. For example, the most important document for this project is the 19th chapter of Labor Code of the Russian Federation, «Leave». The articles of this chapter give better understanding about how

the leaves are granted in Russian companies and institutions. As a result, some of them may be treated as software requirements for the project:

1. Every employee is granted annual leaves, which lasts at least 28 calendar days (article 115);
2. The heads of departments must be able to grant employees additional days of leaves, when certain conditions (dangerous labor conditions; having irregular working hours etc.) are met (article 116);
3. The employees with irregular working hours are granted an additional annual paid leave with the duration of at least 3 calendar days (article 119);
4. When calculating the annual paid leave duration, non-working holidays falling on the period of the leave are not included into the number of calendar days (article 120);
5. The web application should give an opportunity to shift or prolong an already approved leave (article 124);
6. In case of a leave being cancelled or interrupted, the web application must save the fact that a certain number of leave days has not been used by the employee, so that they must be spent the next year (articles 124 and 125);
7. It is banned not to grant the annual paid leave for two successive years (article 124);
8. It must be possible to split the leave into several periods, but in this case at least one of them must have the duration of at least 14 days (article 125).

3 Software Design

The design phase of software development deals with transforming the customer requirements into an implementable form using a programming language [15]. It is essential to pay attention to this phase and not to skip it before starting to program. Information systems may be very complicated, and it is needed to find an architecture, which would provide developers an opportunity to create a program, that

would satisfy the client's needs and would be flexible enough for being changed if needed without much harm and waste of time.

First of all, the design phase lets developers look at an application as a whole, how it will work as a system and what main processes must be implemented. It is hard to think about it, when the programming phase has already started, because during this process developers are focused only on small parts of the program they are currently working on.

Secondly, this phase helps to organize a team's work. When the design phase has successfully finished, developers know what modules they will implement, how those modules are connected and work together. Also, if needed, the developers may get better understanding by looking at various diagrams, which are the result of the design phase as well. As a result, it is easier to plan and schedule the work, because all the needed parts of an upcoming application are already visible and understandable for everybody, and it is only needed to implement them using a programming language.

3.1 Use Case Diagram

Use case diagrams are used for summarizing the details of the system users (also called as actors) and their interactions with the system. To build one, a certain set of symbols and connectors are used. An effective use case diagram can help a team to discuss and present in visual and understandable form scenarios of interactions between the system and its users, goals that the system helps those users achieve, the scope of the system [16].

There are four actors in this project. The first one is a guest, a user who has not passed a procedure of authorization yet, so this person does not have any use cases except of signing in, using their login and password, as it is shown in Figure 2.

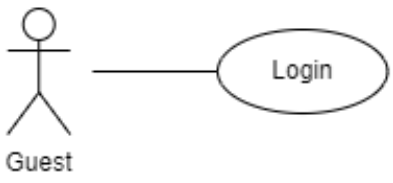


Figure 2 – The actor «Guest»

The second actor is an administrator shown in Figure 3. It is a special type of users, which has rights to view the lists and information about all the employees and departments, the data about which application receives from the API of TPU. This role is supposed to observe all the data that the system is working with, so that it is possible to find all the mistakes made by the system.

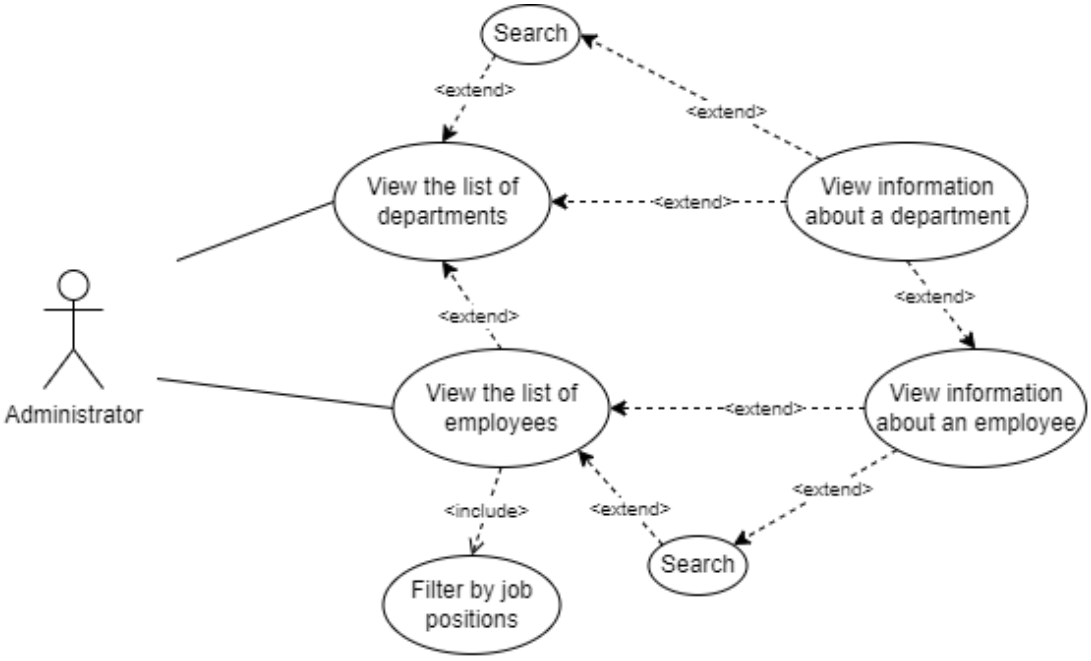


Figure 3 – The actor «Administrator»

The third actor, shown in Figure 4, is an employee of a department of TPU. All the users of the application (except for administrators) have this role. The main functionality of such users is an opportunity to plan and manage their leaves. They have a page with the list of leaves, where they can view the leaves, which they have added before, edit them (if those leaves have not been approved by the head of their department yet) and add new ones. Besides, if a leave has been approved already, the employee can send a message with a request to the head to change that leave. Also, the employees can view the notifications, which tell them about recent actions related to their leaves (for example, when the head of the department approves a leave).

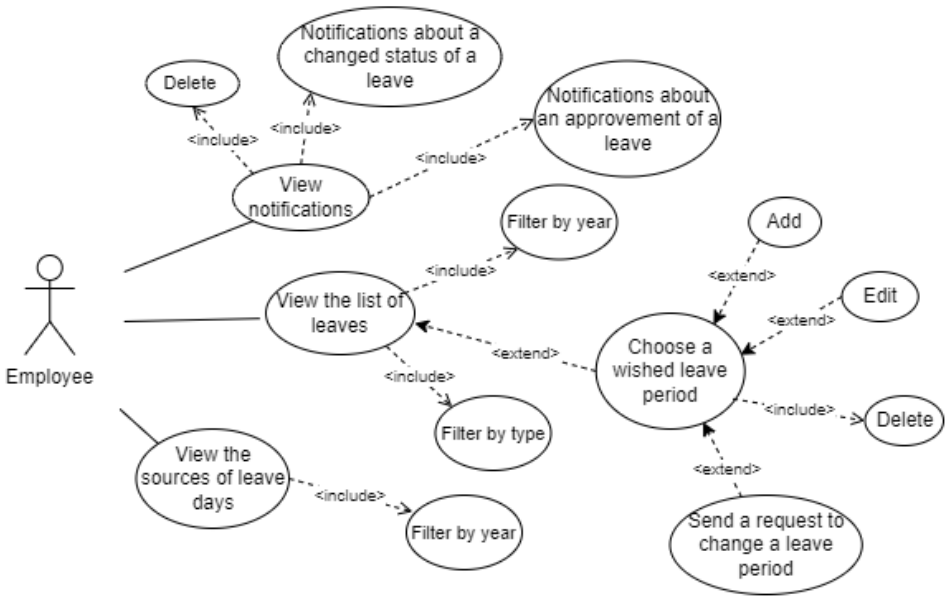


Figure 4 – The actor «Employee»

The last actor (Figure 5) is a head of a department. First of all, this actor inherits from the actor «Employee», so that it has the set of the employees' use cases. Besides, the heads have their own functionality related to managing the vacation leaves in the department (departments) they run.

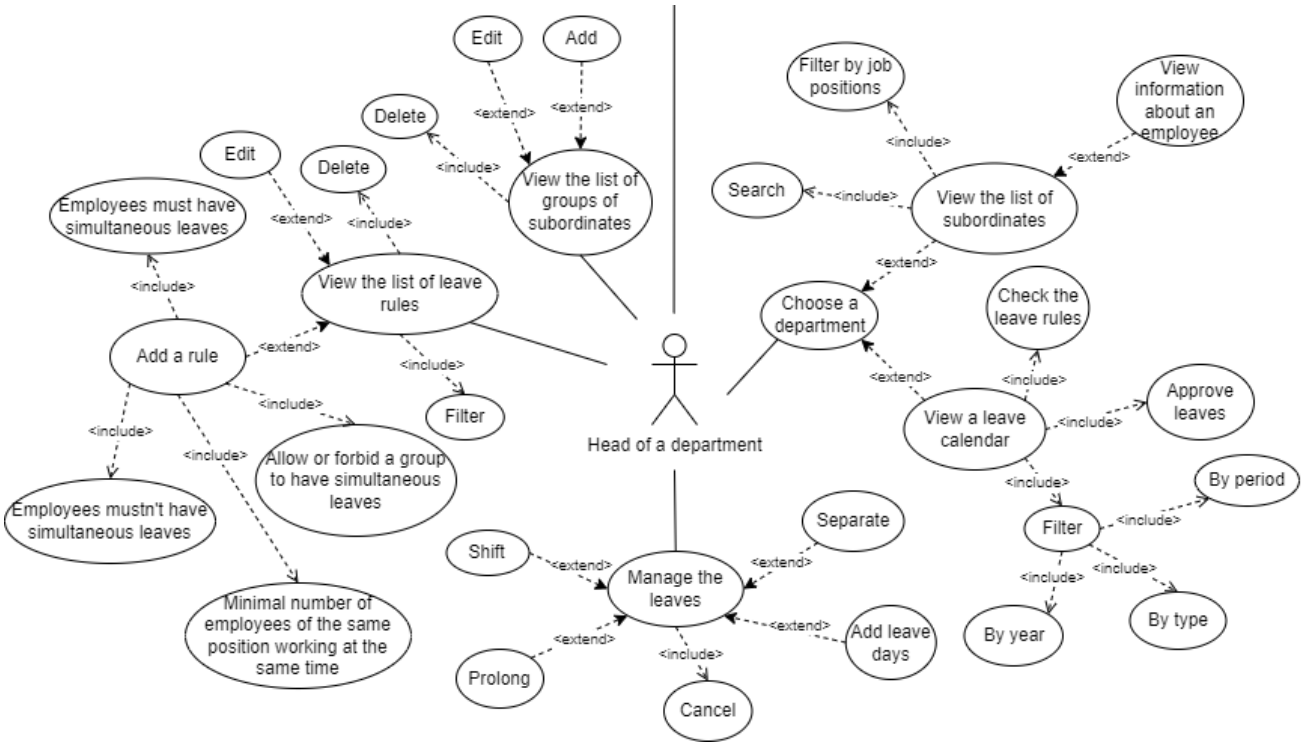


Figure 5 – The actor «Head of a department»

As it can be seen in the figure, the heads have several special functions:

1. They can view the lists of the vacation rules they have created earlier, also they can add, edit and delete them;

2. The heads can view the list of groups of their employees, add new groups, edit and delete existing ones;
3. They can view the lists of departments, which they manage, and the lists of employees working in those departments, as well as the information about a chosen employee or a department;
4. They can use the vacation calendars, which allow the heads to see the leaves of their workers in a table; besides, it is possible to check if all the created vacation rules for the leaves have been followed;
5. The heads can manage their employees' vacation leaves.

3.2 Site Map

The site map is a diagram representing the pages of the site (web application), their hierarchy and mutual links. This kind of diagrams is helpful for developers during a programming phase, because it lets them know what kinds of pages must be implemented and how they are connected, so it is possible to develop the logic of moving from one page to another. The Figure 6 shows the site map for this web application.

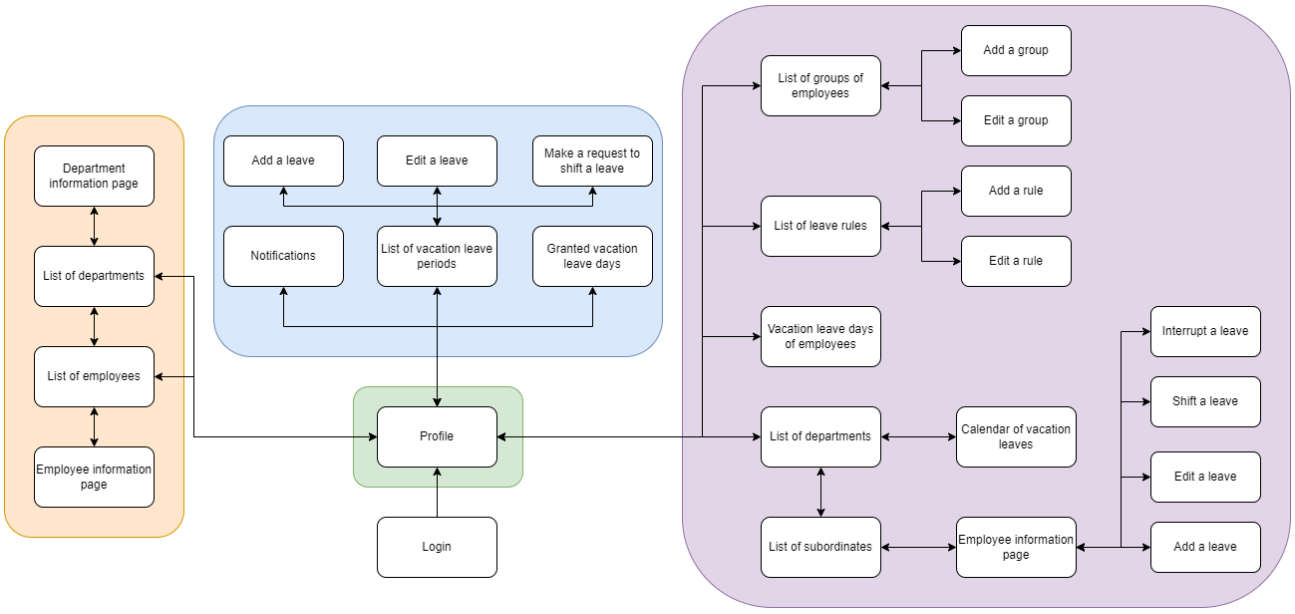


Figure 6 – The site map

As it can be seen, there are 5 zones in the map:

1. The zone of a guest, where there is only a login page, because the guests do not have any other functions available for them;
2. The zone of a user's profile, which is available for all the users passed through an authorization procedure;
3. The zone of an administrator, where such users can visit the pages with the lists of departments and employees of TPU;
4. The zone of an employee, where notifications, the leave periods of this employee and the list of granted leave days are available;
5. The zone of a head of a department, which includes the functionality related to managing the leave periods of their employees.

References

1. SearchSoftwareQuality. 2019. What is Web Application (Web Apps) and its Benefits. [online] Available at: https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app (Accessed 24 May 2022).
2. Indeed Career Guide. 2021. What Is a Web Application? How It Works, Benefits and Examples | Indeed.com. [online] Available at: https://www.indeed.com/career-advice/career-development/what-is-web-application (Accessed 24 May 2022).
3. RingCentral UK Blog. 2021. What is a Web Application? Definition, Benefits and How it Works | RingCentral UK Blog. [online] Available at: https://www.ringcentral.co.uk/gb/en/blog/definitions/web-application (Accessed 24 May 2022).
4. MuleSoft. 2022. What is an API? (Application Programming Interface) | MuleSoft. [online] Available at: https://www.mulesoft.com/resources/api/what-is-an-api (Accessed 24 May 2022).

5. Nordic APIs. 2016. What Data Formats Should My API Support? | Nordic APIs |. [online] Available at: https://nordicapis.com/what-data-formats-should-my-api-support (Accessed 24 May 2022).
6. Json.org. n.d. JSON. [online] Available at: https://www.json.org/json-en.html (Accessed 24 May 2022).
7. Statista. 2022. Most used web frameworks among developers 2021 | Statista. [online] Available at: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web (Accessed 24 May 2022).
8. Djangoproject.com. 2022. The web framework for perfectionists with deadlines | Django. [online] Available at: https://www.djangoproject.com (Accessed 24 May 2022).
9. DataFlair. 2020. Django Advantages and Disadvantages - Why You Should Choose Django? - DataFlair. [online] Available at: https://data-flair.training/blogs/django-advantages-and-disadvantages (Accessed 24 May 2022).
10. Spring.io. 2022. Why Spring?. [online] Available at: https://spring.io/why-spring (Accessed 24 May 2022).
11. www.javatpoint.com. n.d. Java Spring Pros and Cons - Javatpoint. [online] Available at: https://www.javatpoint.com/java-spring-pros-and-cons (Accessed 24 May 2022).
12. GitHub. 2022. GitHub - dotnet/aspnetcore: ASP.NET Core is a cross-platform .NET framework for building modern cloud-based web applications on Windows, Mac, or Linux.. [online] Available at: https://github.com/dotnet/aspnetcore (Accessed 24 May 2022).
13. Docs.microsoft.com. 2022. Overview of ASP.NET Core MVC. [online] Available at: https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-6.0 (Accessed 24 May 2022).

14. www.simplilearn.com. 2022. What Is Requirement Analysis: Applications, Techniques and Tools Used. [online] Available at: https://www.simplilearn.com/what-is-requirement-analysis-article (Accessed 24 May 2022).
15. GeeksforGeeks. 2021. Software Engineering | Software Design Process - GeeksforGeeks. [online] Available at: https://www.geeksforgeeks.org/software-engineering-software-design-process (Accessed 24 May 2022).
16. Lucidchart. n.d. UML Use Case Diagram Tutorial. [online] Available at: https://www.lucidchart.com/pages/uml-use-case-diagram (Accessed 24 May 2022).

Приложение Б

Метод GetDaysInfo()

Листинг Б.1 – Код метода GetDaysInfo()

```
/// <summary>
/// Получить распределение отпускных дней
/// </summary>
/// <param name="days">Список отпускных дней, назначенных
/// сотруднику</param>
/// <param name="year">Год, на который нужно просмотреть
/// отпускные дни</param>
/// <returns>Словарь "тип отпуска –
/// количество назначенных дней"</returns>
static private Dictionary<string, int> GetDaysInfo(List<VacationDay>
days, int year)
{
      Dictionary<string, int> result = new Dictionary<string, int>();

      days = days.OrderByDescending(d => d.Year).ToList();
      foreach (VacationDay day in days)
      {
          // проверка, принадлежат ли дни отпуска прошлому году
          if (day.Year == year - 1)
          {
               string previousYearText = "Оставшиеся за прошлый год";
               if (result.ContainsKey(previousYearText))
                    result[previousYearText] += day.NumberOfDays –
                                                day.TakenDays;
               else
                    result.Add(previousYearText, day.NumberOfDays –
                                                day.TakenDays);
          }
          // дни принадлежат текущему году
          else
          {
               // проверка, есть ли уже
               //запись о таком основании отпуска
               if (result.ContainsKey(day.VacationType.Name))
               {
                    // если уже есть - добавить количество
                    //полученных по этому основанию дней
                    result[day.VacationType.Name] += day.NumberOfDays;
               }
               // если нет записи о таком основании отпуска - создать
               else
                    result.Add(day.VacationType.Name,
                                   day.NumberOfDays);
```

```
            }
        }

        return result;
    }
```

Приложение В.

Мастер-страница _Common.cshtml

Листинг В.1 – Код разметки мастер-страницы _Common.cshtml

```
@using Microsoft.AspNetCore.Http;
@using VacationSystem.HtmlHelpers;

@{
  Layout = "_Layout";

  // является ли данный сотрудник руководителем
  //какого-нибудь подразделения
  bool isHead = false;

  if (Context.Session.GetString("head") != null)
    isHead = bool.Parse(Context.Session.GetString("head"));

  bool isAdmin = false;
  if (Context.Session.GetString("user_type") == "administrator")
    isAdmin = true;

  // сообщения со стороны сервера
  string success = null;
  if (TempData["Success"] != null)
    success = TempData["Success"].ToString();

  string error = null;
  if (TempData["Error"] != null)
    error = TempData["Error"].ToString();

  string message = null;
  if (TempData["Message"] != null)
    message = TempData["Message"].ToString();
}

<div class="popup-window">
</div>
<main role="main" id="main-block" class="flex-container flex-row flex-
start">
  <div id="left-menu" class="flex-container flex-column flex-start">
    @if (isAdmin)
    {
      <p class="menu-header menu-item">Меню</p>
      <a asp-controller="Admin" asp-action="Departments" class="menu-link
      menu-item">
        Подразделения
      </a>
```

```
      <a asp-controller="Admin" asp-action="Employees" class="menu-link
      menu-item">
        Сотрудники
      </a>
    }
    else
    {
      <p class="menu-header menu-item">Меню</p>
      <a asp-controller="Vacation" asp-action="Index" class="menu-link
      menu-item">Отпуска</a>
      <a asp-controller="VacationDays" asp-action="List" class="menu-link
      menu-item">Отпускные дни</a>
      <a asp-controller="Home" asp-action="Profile" class="menu-link
      menu-item">Уведомления</a>
      @if (isHead)
      {
        <p class="menu-header menu-item">Руководитель</p>
        <a asp-controller="Head" asp-action="Departments" class="menu-
        link menu-item">Подразделения</a>
        <a asp-controller="Groups" asp-action="Index" class="menu-link
        menu-item">Группы сотрудников</a>
        <a asp-controller="Rules" asp-action="Index" class="menu-link
        menu-item">Правила отпусков</a>
        <a asp-controller="VacationDays" asp-action="SetDays"
        class="menu-link menu-item">Управление отпусками</a>
      }
    }
    <hr class="menu-separator" />
    <a asp-controller="Login" asp-action="Logout" class="menu-link menu-
    item">
      Выйти
    </a>
  </div>
  <div id="right-space" class="flex-container flex-column flex-start">
    <div id="messages-block" class="container">
      <div class="row">
        <div class="col-md-12">
          @Html.CheckError(error)
          @Html.CheckMessage(message)
          @Html.CheckSuccess(success)
        </div>
      </div>
    </div>
    <div id="main-content main-small-size">
      @RenderBody()
    </div>
  </div>
</main>
@section scripts{
  @await RenderSectionAsync("Scripts", required: false)
}
```

Приложение Г

Рисунок Г.1 – Диплом за победу в номинации хакатона Audithon 2021