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

Институт Кибернетики
Специальность 09.03.03. Прикладная информатика
Кафедра Оптимизации систем управления

БАКАЛАВРСКАЯ РАБОТА

Тема работы
«Разработка проекта основных функций, состава и блок-схемы национального стандарта передачи данных для нефтегазовых предприятий России.»

УДК ___ 004.67:006:622.32.013 _____

Студент

Группа	ФИО	Подпись	Дата
8К21	Гончаров Аркадий Сергеевич Забродин Иван Евгеньевич		

Руководитель

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Заведующий лабораторией каф. ОСУ	Марчуков А.В.			

КОНСУЛЬТАНТЫ:

По разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент	Банова К.А.	к.э.н.		

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент каф. ЭБЖ	Мезенцева И.Л.			

ДОПУСТИТЬ К ЗАЩИТЕ:

Зав. кафедрой	ФИО	Ученая степень, звание	Подпись	Дата
ОСУ	Иванов М.А.	к.т.н.		

Томск – 2016 г.

Запланированные результаты изучения по основной образовательной программе подготовки бакалавров 09.03.03. «Прикладная информатика»

Код результатов	Результат обучения (выпускник должен быть готов)
Профессиональные компетенции	
Р1	Применять глубокие естественнонаучные, математические и инженерные знания для создания и обработки новых материалов
Р2	Применять глубокие знания в области современных технологий машиностроительного производства для решения междисциплинарных инженерных задач
Р3	Ставить и решать инновационные задачи инженерного анализа, связанные с созданием и обработкой материалов и изделий, с использованием системного анализа и моделирования объектов и процессов машиностроения
Р4	Разрабатывать технологические процессы, проектировать и использовать новое оборудование и инструменты для обработки материалов и изделий, конкурентоспособных на мировом рынке машиностроительного производства
Р5	Проводить теоретические и экспериментальные исследования в области современных технологий обработки материалов, нанотехнологий, создания новых материалов в сложных и неопределенных условиях
Р6	Внедрять, эксплуатировать и обслуживать современные высокотехнологичные линии автоматизированного производства, обеспечивать их высокую эффективность, соблюдать правила охраны здоровья и безопасности труда на машиностроительном производстве, выполнять требования по защите окружающей среды

Универсальные компетенции	
Р7	Использовать глубокие знания по проектному менеджменту для ведения инновационной инженерной деятельности с учетом юридических аспектов защиты интеллектуальной собственности
Р8	Активно владеть иностранным языком на уровне, позволяющем работать в иноязычной среде, разрабатывать документацию, презентовать и защищать результаты инновационной инженерной деятельности
Р9	Эффективно работать индивидуально, в качестве члена и руководителя группы, состоящей из специалистов различных направлений и квалификаций, демонстрировать ответственность за результаты работы и готовность следовать корпоративной культуре организации
Р10	Демонстрировать глубокие знания социальных, этических и культурных аспектов инновационной инженерной деятельности, компетентность в вопросах устойчивого развития
Р11	Самостоятельно учиться и непрерывно повышать квалификацию в течение всего периода профессиональной деятельности

Форма задания на выполнение выпускной квалификационной работы

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

Институт киберенетики_____
Направление подготовки (специальность) прикладная информатика_____
Кафедра ОСУ _____

УТВЕРЖДАЮ:
Зав. кафедрой
_____ _____ _____
(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

Бакалаврской работы (в составе команды)

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

Студенту:

Группа	**ФИО**
8К21	1. Гончаров А.С. 2. Забродин И.Е.

Тема работы:

. «Разработка проекта основных функций, состава и блок-схемы национального стандарта передачи данных для нефтегазовых предприятий России.»

Утверждена приказом директора (дата, номер)	

Срок сдачи студентами выполненной работы:	

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.).*	1. Объект исследования – современные стандарты передачи данных и их роль в добычи углеводородов. История разработки современных стандартов передачи данных. Блок схема программной реализации и схема данных стандарта WITSML- 2. Составляющие стандарта WITSML-2. (RESQML, PRODML, WITSML 1.4.1) 2. Архитектура схемы данных должна поддерживать номенклатуру не менее 50000 значений 3. Международный проект: схема данных стандарта WITSML- 2.0, с возможностью добавления отечетвенных компонентов схемы данных 4. Должен включать в себя основные данные о процессе добычи, бурения и состоянии подземного резервуара. 5. Систему обозначений и размерностей сделать максимально совместимой с международными, но учитывающей особенности Российских технологий добычи, бурения и представления о состоянии резервуара. 6. Стандарт должен быть разработан на основе XML и веб-технологий. 7. Стандарт должен поддерживать современные системы измерений (датчиков), в том числе оптоволоконные. Стандарт должен поддерживать функцию добавлени и изменения схемы данных.
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. Разработать основные принципы создания схемы данных Национального стандарта передачи данных. 2. Разработать структурную схему схемы данных. 3. Разработать полную структурную схему программной поддержки стандарта (схема данных, серверная платформа, протокол передачи данных). 4. Разработать структуру базы данных Национального стандарта. 5. Разработать структурную схему блока обмена данных с агентами и моделями. 6. Разработать перспективную схему единого

	информационного пространства нефтегазовой корпорации, на основе стандарта передачи данных. 7. Обоснование выбора базовой технологии - XML
Перечень графического материала *(с точным указанием обязательных чертежей)*	1. Блок схема схемы данных; 2. Блок схема программного комплекса поддержки стандарта (сервера); 3. Схему базы данных (фрагмент для бурения); 4. Алгоритмы формирования схемы данных; 5. Алгоритм обмена данными с агентами и моделями; 6. Схема применения стандарта- обзорная;

Консультанты по разделам выпускной квалификационной работы

(с указанием разделов)

Раздел	Консультант

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

1. Анализ стандарта WITSML-2
2. Анализ проблем внедрения стандарта WITSML в отечественной нефтегазовой промышленности
3. Анализ преимуществ использования стандарта
4. Проектирование модели процесса взаимодействия с клиентом

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	

Задание выдал руководитель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Заведующий лабораторией кафедры ОСУ	Марчуков Артур Викторович			

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата

8К21	1.Гончаров А.С. 2. Забродин И.Е.		

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

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

Форма представления работы:

бакалаврская работа

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

КАЛЕНДАРНЫЙ РЕЙТИНГ-ПЛАН
выполнения выпускной квалификационной работы

Срок сдачи студентом выполненной работы:	15 июня 2016 г.

Дата контроля	Название раздела (модуля) / вид работы (исследования)	Максимальный балл раздела (модуля)
11.03.2016	*Раздел 1. Роль и современное состояние стандартизации передачи данных при бурении, добычи углеводородов и анализе состояния подземного резервуара в России и за рубежом.*	*10*
13.04.2016	*Раздел 2. Стандарт WITSML 2.0, как основа стандарта передачи данных для отечественной нефтегазодобывающей промышленности*	*15*
18.05.2016	*Раздел 3. Анализ преимуществ использования стандарта*	*10*
05.06.2016	*Раздел 4. Проектирование модели процесса взаимодействия с клиентом*	*30*
08.06.2016	*Раздел 5. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение*	*10*
08.06.2016	*Раздел 6. Социальная ответственность*	*10*
14.06.2016	*Заключение*	*15*

Составил преподаватель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Заведующий лабораторией каф. ОСУ	Марчуков А.В.			

СОГЛАСОВАНО:

Зав. кафедрой	Иванов М.А.	Ученая степень, звание	Подпись	Дата
ОСУ	Иванов М.А.	к.т.н.		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«ФИНАНСОВЫЙ МЕНЕДЖМЕНТ, РЕСУРСОЭФФЕКТИВНОСТЬ И РЕСУРСОСБЕРЕЖЕНИЕ»

Студентам:

Группа	ФИО
8К21	Гончарову Аркадию Сергеевичу Забродину Ивану Евгеньевичу

Институт	**Институт кибернетики**	**Кафедра**	**Оптимизации систем управления**
Уровень образования	бакалавриат	**Направление/специальность**	Прикладная информатика

Исходные данные к разделу «Финансовый менеджмент, ресурсоэффективность и ресурсосбережение»:	
1. Стоимость ресурсов научного исследования (НИ): материально-технических, энергетических, финансовых, информационных и человеческих	Работа с клиентами WITSML сервера, разработка веб-интерфейса для предоставления услуг сервера. Работа с клиентской информацией, отображаемой в личных кабинетах, входящая в положения договора, включающая личные данные клиента, информацию о тарифных планах и скважинах.
2. Нормы и нормативы расходования ресурсов	
3. Используемая система налогообложения, ставки налогов, отчислений, дисконтирования и кредитования	
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Оценка коммерческого потенциала, перспективности и альтернатив проведения НИ с позиции ресурсоэффективности и ресурсосбережения	Оценка конкурентоспособности, рассмотрение альтернатив проведения НИ.
2. Планирование и формирование бюджета научных исследований	Планирование этапов разработки программы, определение трудоемкости, формирование бюджета НТИ.
3. Определение ресурсной (ресурсосберегающей), финансовой, бюджетной, социальной и экономической эффективности исследования	Сравнительный анализ интегральных показателей эффективности.
Перечень графического материала (с точным указанием обязательных чертежей):	
1. Оценка конкурентоспособности технических решений 2. SWOT анализ, QuaD технология 3. График проведения и бюджет НИ 4. Оценка ресурсной, финансовой и экономической эффективности НИ	

Дата выдачи задания для раздела по линейному графику	15.04.2016

Задание выдал консультант:

Должность	**ФИО**	**Ученая степень, звание**	**Подпись**	**Дата**
Ассистент	Банова К.А.	к.э.н.		

Задание приняли к исполнению студенты:

Группа	**ФИО**
8К21	Гончаров Аркадий Сергеевич
8К21	Забродин Иван Евгеньевич

ЗАДАНИЕ ДЛЯ РАЗДЕЛА «СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студентам:

Группа	ФИО
8К21	Гончарову Аркадию Сергеевичу Забродину Ивану Евгеньевичу

Институт	Институт кибернетики	**Кафедра**	Оптимизации систем управления
Уровень образования	бакалавриат	**Направление/специальность**	Прикладная информатика

Исходные данные к разделу «Социальная ответственность»:	
1. Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика, рабочая зона) и области его применения	Объект исследования – национальный стандарт передачи данных для нефтяной и газовой промышленности, совместимый с международным стандартом WITSML 2.0; Интерфейс взаимодействия пользователя с сервером хранения и обработки данных, использующим национальный стандарт.
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Производственная безопасность. 1.1. Анализ выявленных вредных факторов при разработке и эксплуатации проектируемого решения. 1.2. Анализ выявленных опасных факторов при разработке и эксплуатации проектируемого решения.	Анализ выявленных вредных факторов: • Недостаточная освещенность рабочей зоны • Отклонение показателей микроклимата • Повышенный уровень шума на рабочем месте • Повышенный уровень электромагнитных излучений Анализ выявленных опасных факторов: - Статическое электричество - Короткое замыкание - Пожароопасность
2. Экологическая безопасность.	- Анализ негативного воздействия на окружающую природную среду: утилизация компьютеров и другой оргтехники
3. Безопасность в чрезвычайных ситуациях.	Возможные чрезвычайные ситуации: - Пожар
4. Правовые и организационные вопросы обеспечения безопасности.	- Рабочее место при выполнении работ сидя регулируется ГОСТом 12.2.032 – 78 - Организация рабочих мест с электронно-вычислительными машинами регулируется СанПиНом 2.2.2/2.4.1340 – 03

Дата выдачи задания для раздела по линейному графику	15.04.2016

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Ассистент	Мезенцева И.Л.			

Задание приняли к исполнению студенты:

Группа	ФИО	Подпись	Дата
8К21	Гончаров Аркадий Сергеевич Забродин Иван Евгеньевич		

Оглавление

Реферат 17

Обозначения и сокращения 19

Введение 20

Раздел 1. Роль и современное состояние стандартизации передачи данных при бурении, добычи углеводородов и анализе состояния подземного резервуара в России и за рубежом 22

1.1. Формирование данных о процессе бурения на станции управления бурением и передача до центра моделирования месторождения 22

1.2. Современные стандарты передачи данных, их предшественники и перспективы их развития 23

1.2.1. Предшественники современных стандартов 24

1.3. Типовая схема работы сервера поддержки стандарта, основные составляющие стандарта 30

1.4. Проблемы создания национального интегрированного стандарта передачи данных для нефтегазовой промышленности, совместимого с перспективным стандартом WITSML2.0 42

Раздел 2 Стандарт WITSML 2.0, как компонент стандарта передачи данных для отечественной нефтегазодобывающей промышленности 48

2.1. Вопросы интеграции национального стандарта и WITSML2.0 48

2.2. Основные функции национального стандарта 49

2.3. Алгоритм формирования схемы данных стандарта 51

2.4. Структура и схема данных WITSML 2.0 с включением национального стандарта 54

Раздел 3 Анализ преимуществ использования информационной инфраструктуры, интегрированной с национальным стандартом 63

3.1. Обоснование использования технологии XML 63

3.2. Определение потребности в оборудовании и ресурсах 66

3.3. Обзорная схема применения стандарта 70

Раздел 4.Проектирование модели процесса взаимодействия с клиентом 72

4.1. Описание требований к интерфейсу пользователя 72

4.2. Проектирование веб-приложения 76

4.3. Проектирование базы данных для веб-приложения 80
4.4. Дизайн веб-приложения 82
4.5. Разработка английской версии веб-приложения 83
4.6. Разработка веб-форм приложения 84
Раздел 5. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение. 95
5.1. Введение 95
5.2. Оценка коммерческого потенциала и перспективности проведения научных исследований с позиции ресурсоэффективности и ресурсосбережения 95
5.2.1. Потенциальные потребители результатов исследования 95
5.2.2. Анализ конкурентных технических решений 96
5.2.3. Технология QuaD 97
5.2.4. SWOT-анализ 99
5.3. Нахождение альтернативных способов проведения научных исследований 102
5.4. Создание научно-исследовательских работ 103
5.4.1. Этапы планирования работ в рамках научного исследования 103
5.4.2. Расчет трудоемкости исполнения работ 105
5.4.3. Бюджет научно-технического исследования 110
5.4.3.1. Расчет материальных затрат НТИ 110
5.4.3.2. Расчет амортизационных отчислений 112
5.4.3.3. Расчет основной заработной платы исполнителей темы 113
5.4.3.4. Дополнительная заработная плата 115
5.4.3.5. Отчисления во внебюджетные фонды (страховые отчисления) 115
5.4.3.6. Формирование бюджета затрат научно-исследовательского проекта 116
Раздел 6. Социальная ответственность 118
6.1. Введение 118
6.2. Производственная безопасность 119
6.2.1. Недостаточная освещенность рабочей зоны 119
6.2.2. Отклонение показателей микроклимата 120

6.2.3. Повышенный уровень шума на рабочем месте 122
6.2.4. Повышенный уровень электромагнитных излучений 122
6.2.5. Электробезопасность 123
6.3. Экологическая безопасность 123
6.4. Правовые и организационные вопросы обеспечения безопасности 124
6.5. Безопасность в чрезвычайных ситуациях 125
Заключение 127
Список публикаций, достижений и интеллектуальная собственность студентов: 129
Список использованных источников: 130
Приложения 132
Приложение 1. Программный код контроллеров 132
Приложение 2. Программный код представлений 163

Реферат

Ключевые слова: стандарт передачи данных, WITSML сервер, WITSML стандарт, PRODML, RESQML.

Объектом исследования выпускной квалификационной работы является: состав и функционал проектируемого национального стандарта передачи данных для российских нефтегазовых предприятий, совместимого с международным стандартом WITSML 2.0.

Целью работы является: разработка основных функций, состава и блок схемы национального стандарта передачи данных для нефтегазодобывающей отрасли России; разработка программного продукта, основанного на web-технологиях, позволяющего администрировать сервер для хранения и обработки стандартизованных данных.

В процессе исследования был изучен стандарт WITSML 2.0, и его применения, возможности программных платформ на основе использования стандарта, проведен анализ существующих средств разработки аналогичных программных продуктов.

С точки зрения экономики, разработка национального стандарта передачи данных снизит затраты на обслуживание нефте- и газодобывающей отрасли России, реализуя импортозамещение дорогостоящих систем передачи данных, используемых на территории страны на настоящий момент.

В результате исследования были формализованы знания об основных функциях и свойствах стандарта передачи данных, используемых на нефтегазовых предприятиях; Разработано коммерческое приложение, позволяющее клиентам взаимодействовать с администрацией сервера, получать информацию о стоимости услуг и потребности в технических ресурсах.

Разработка приложения велась с помощью Microsoft Visual Studio 2015, язык реализации C#.

Областью применения данного исследования является разработка и запуск в эксплуатацию отечественного сервера WITSML, работающего в

режиме реального времени, для передачи и хранения данных в формализованном виде, отвечающем международным требованиям.

В будущем планируется разработка программных модулей, координирующих бизнес-процессы работы сервера.

Обозначения и сокращения

ИМ (интеллектуальное месторождение) - Интеллектуальное нефтегазовое месторождение — система автоматического управления операциями по добыче нефти и газа, предусматривающая непрерывную оптимизацию интегральной модели месторождения и модели управления добычей.

WITSML- стандартный язык разметки передачи информации с буровой.

PRODML – стандартный язык разметки передачи информации о процессе добычи углеводородов.

RESQML - стандартный язык разметки передачи информации для построения геологической модели месторождния.

ОВД - оптоволоконный датчик.

LAS – Log ASCII Standart.

УИ – уникальный идентификатор.

ОС – операционная система сервера, контроллера и пр.

СУБД - система управления базой данных.

SCADA-система - (аббр. от англ. supervisory control and data acquisition, диспетчерское управление и сбор данных) — программный пакет, предназначенный для разработки или обеспечения работы в реальном времени систем сбора, обработки, отображения и архивирования информации об объекте мониторинга или управления. SCADA может являться частью АСУ ТП, АСКУЭ, системы экологического мониторинга, научного эксперимента, автоматизации здания и т. д.

XML- спецификация XML описывает XML-документы и частично описывает поведение XML-процессоров (программ, читающих XML-документы и обеспечивающих доступ к их содержимому).

LTE- стандарт мобильной связи Long Term Evolution.

SOAP – протокол обмена структурированными сообщениями.

Введение

В настоящее время вследствие освоения новых районов бурения и осложнения добычи затраты на разработку и эксплуатацию нефтяных и газовых месторождений увеличиваются при сохранении или же уменьшении объема извлекаемых из недр полезных ископаемых. Данная ситуация требует внедрения новых методов добычи, которые позволили бы увеличить ресурсоотдачу пластов и уменьшить сопутствующие расходы.

На текущий момент все ведущие мировые корпорации по добыче и сервисному обслуживанию месторождений, так или иначе, используют цифровые технологии при разработке и мониторинге месторождений, что позволяет заметно сократить издержки в некоторых условиях, а самое главное увеличить количество рентабельных скважин и этим увеличить общий объем добычи. Одной из ключевых технологий, позволяющих реализовать поставленную задачу, является внедрение современных протоколов обмена информацией и систем моделирования, позволяющих получать и обрабатывать данные о бурении в реальном времени, составлять математические, гидрологические и гидродинамические модели пласта, осуществлять обратную передачу данных от модели к скважине для оперативного управления процессом бурения.

Россия занимает одно из ведущих мест в мире по добыче и экспорту углеводородов. При этом до сих пор актуальна проблема несоответствия форматов передачи данных от разных вычислительных устройств, используемых на нефтегазовых предприятиях, друг другу. Это приводит как к информационным потерям и уменьшению скорости передачи данных вследствие множественных конвертаций, так и увеличению затрат на обслуживание таких информационных систем. Поэтому отечественные нефтегазовые компании предпочитают пользоваться услугами зарубежных серверов, аккумулирующих данные со скважин и представляющие их в виде

готовых моделей. В большинстве своем, они используют передовой международный стандарт передачи данных WITSML, реализованный на основе языка XML. Данный стандарт позволяет снимать около 7500 параметров, а также в нем нет ограничений на увеличение изначальной номенклатуры, что расширяет область его применения до месторождений любой сложности и структуры. Разработкой стандарта WITSML занимается американский консорциум Энерджистикс (Energistics) с 1990 года. На текущий момент перспективной версией является разрабатываемая версия WITSML 2.0, в которую предлагается включить предыдущие версии стандартов- WITSML, PRODML, RESQML.

На данный момент не существует отечественных серверов, поддерживающих международный стандарт передачи данных. Исходя из этого, актуальной становится задача создания национального стандарта передачи данных и сопутствующего программного обеспечения для внедрения на предприятия нашей страны. Это позволит сохранить положительные качества использования цифровых технологий в нефте- и газодобыче, сокращая при этом расходы на реализацию проектов с использованием таких технологий. Решение данной задачи заключается в разработке и реализации проекта по созданию национального стандарта передачи данных о нефтедобыче, совместимого с международным стандартом WITSML 2.0, но учитывающего специфику работы нефтегазовых предприятий России.

Раздел 1. Роль и современное состояние стандартизации передачи данных при бурении, добычи углеводородов и анализе состояния подземного резервуара в России и за рубежом

1.1. Формирование данных о процессе бурения на станции управления бурением и передача до центра моделирования месторождения

Современное состояние нефтеразведки и добычи характеризуется массовой разработкой высокоэффективных горизонтальных скважин, при прокладке которых важно выдерживать оптимальную траекторию бурения горизонтального участка. Для этого используются системы мониторинга бурения на основе оптоволоконных датчиков (кабелей). При данном методе мониторинга формируется огромное количество данных по состоянию всего ствола скважины.

В России в 90-х годах прошлого века было разработано несколько систем контроля процесса бурения. Обычно они представляли собой набор технологических датчиков, компьютер (устройство сбора информации) и информационное табло (пульт бурильщика). Хотя система и позволяла сохранять регистрируемые данные, основным её назначением был контроль параметров бурения на месте. Компьютер анализировал данные датчиков и сигнализировал об аномальных изменениях в показаниях, кроме того, оператор мог распечатать графический отчёт по собранным данным.

Однако, для оперативного контроля процесса необходима быстрая передача данных со скважины в буровое предприятие. Поэтому следующим шагом стала реализация передачи данных на региональные (локальные) уровни иерархии управления производством, при этом в качестве канала передачи данных использовалась телефонная, сотовая или спутниковая связь. Таким образом, у специалистов, находящихся зачастую в сотнях километрах от скважины, появилась возможность просматривать данные с буровой, собранные всего несколько часов назад.

В настоящее время появилась и активно развивается концепция «интеллектуального» месторождения, которая предполагает создание высокоэффективных систем мониторинга, управления разработкой и передачи данных, учитывающих структурные особенности нефтяных и газовых месторождений и обеспечивающих значительное повышение уровня добычи нефти и газа, а также сокращающих производственные издержки. Вместе с тем, проведенный анализ существующих программных комплексов и систем управления разработкой свидетельствует о том, что они пока не ориентированы на интегрирование в единую технологическую цепочку.

Одной из основных функций «интеллектуальных» месторождений является оперативное моделирование состояния подземного резервуара. Отечественные системы управления процессом бурения формируют большой массив информации, но отправить данные в геологическую модель месторождения не всегда представляется возможным. Для обеспечения полного взаимодействия и взаимной интеграции всех компонент «интеллектуального месторождения» в единое целое необходима стандартизация форматов передачи данных, унификация взаимодействий между отечественными системами управления разработкой, зарубежными системами моделирования и инфраструктурой "интеллектуального" месторождения.

1.2. Современные стандарты передачи данных, их предшественники и перспективы их развития

В 1990 году была создана Международная организация Petrotechnical Open Standards Consortium (POSC, с 2006 года - Energistics). Energistics – это международный некоммерческий консорциум, целью которого является предоставление специалистам нефтегазовых компании, занимающихся геологоразведкой и добычей, нейтральной площадки для совместной работы, продвижения открытых стандартов по обмену данными и решения вопросов, касающихся разведки и разработки.

Консорциум Energistics разрабатывает три направления промышленных стандартов в области обмена данных:

- WITSML для бурения, заканчивания и ремонта
- PRODML для работ по добыче, оптимизации и передачи отчетов
- RESQML для моделей геологической среды и пласта

Большая часть зарубежных добывающих компаний либо уже используют у себя технические решения WITSML для передачи скважинных данных в режиме реального времени или планируют это осуществить в ближайшее время. Количество продуктов, совместимых с WITSML, стремительно растет, и сама совместимость с WITSML становится требованием договоров, в особенности на новых месторождениях. WITSML позволяет обеспечить интегрирование данных и интеллектуальный сбор данных и является основополагающей технологией.

К сожалению, в России WITSML-стандарт не получил должного распространения. Причина заключалась в отсутствии доступных WITSML-серверов или возможности организации корпоративных WITSML-серверов. Их программная реализация оказалась слишком затратной для изготовителей станций ГТИ, а мощные ИТ-компании пока не вышли на этот рынок (в силу ряда причин, одна из которых - увязка с клиентскими приложениями на буровых - WITSML-клиентами на станциях ГТИ разных производителей, интерфейсы к которым закрыты и известны только их разработчикам).

1.2.1. Предшественники современных стандартов

В интерпретация геологических исследований предполагает интенсивный обмен информацией между программными системами различных производителей. Для успешного использования форматы разных систем должны поддерживаться каждой из этих систем, либо сводиться к формату, который всеми поддерживается. Учитывая постоянное расширение номенклатуры геофизических методов, внедрением новой аппаратуры и технологий, в геофизике постоянно актуальна задача обмена данными. В

данный момент наиболее популярным способом обмена геологическими данными являются отформатированные текстовые файлы. В отдельных случаях применяются средства импорта / экспорта, позволяющие синхронизировать реляционные схемы данных. Но во всех случаях для успешного обмена необходимо соотнести схему данных источника со схемой данных получателя. И так – для каждой пары взаимодействующих систем. Для решения проблемы связи между системами, работающими с геологическими данными, а также для организации информационного хранилища удобно использовать XML. Это позволяет создать мобильный независимый от операционной системы, СУБД и приложений формат данных, который удобно проверять и интерпретировать стандартными средствами. Промысловая геофизика, связанная с разведкой и эксплуатацией месторождений полезных ископаемых, традиционно сильно зависит от прогресса вычислительной техники и программного обеспечения, оказывая не меньшее влияние на информационно-коммуникационные технологии, со своей стороны. Цена получения информации о локальной структуре земных недр очень высока, стоимость ошибки при бурении одной скважины – несколько миллионов долларов, поэтому постоянно совершенствуется аппаратная база, методика измерений, обработки и интерпретации данных, программное обеспечение. В результате сервисным компаниям, предоставляющим геофизическую информацию, и ее потребителям, добывающим предприятиям, не хватает одной интегрированной системы. Они одновременно используют разные программные продукты.

Начиная с 1980-х удачные инициативы в достижении интероперабельности, такие, как всемирно признанная система стратиграфической классификации отложений или обменный формат данных бурения скважин WITS, фокусировались на определении как можно более широкого круга уникально и единообразно интерпретируемых понятий.

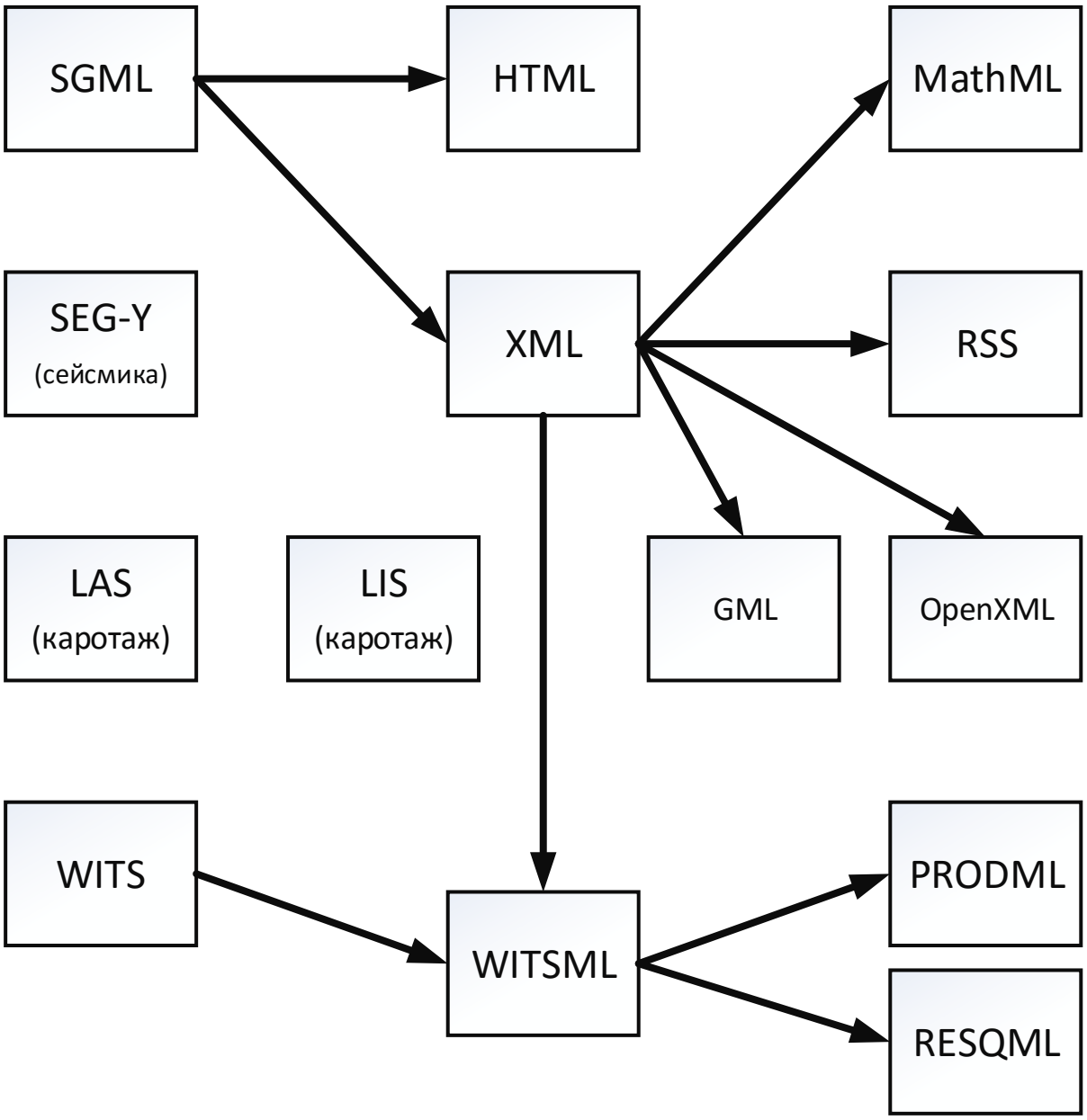

Рис. 1. Основные обменные форматы геофизических данных

На структурном уровне форматы, подобные WITS, оставались неудовлетворительными. Они поддерживали очень ограниченную номенклатуру объектов и свойств. Первые усилия по развитию выразительных возможностей отраслевых форматов, такие как версия 3.0 формата LAS, не смогли сдвинуть старые простые форматы из-за переусложненности и отсутствия стандартных программных интерфейсов. Только с распространением XML у нового поколения обменных форматов появились возможности представлять сложные структуры геофизических данных разведки

и эксплуатации месторождений полезных ископаемых. WITSML был первоначально разработан как новая, рассчитанная на интернет- технологии версия широко используемого формата WITS. Но эта разработка, основанная на стандартах W3C, применении XML и сервис-ориентированном программном интерфейсе SOAP быстро преодолела ограничения, свойственные WITS. PRODML отпочковался от WITSML как формат для поддержки оптимизации добычи нефти и газа.

WITS (Wellsite Information Transfer Specification) - телекоммуникационный протокол передачи данных бурения. Был разработан в середине 1980-х годов для обеспечения обмена информацией на буровой, и предоставления данных станциям удалённого мониторинга бурения.

WITS является многоуровневым протоколом, который предлагает простоту на низких уровнях и возрастающую гибкость формирования более высоких уровней репрезентации данных. На низких уровнях используется заранее определённый текстовой поток формата ASCII, в то же время на самом высшем уровне поток данных может быть настроен для любого вида представления информации.

Уровни WITS:

Уровень 0 - также известен как IRTS (Intra Rig Transfer Specification), базовый уровень обмена текстовой информацией между сервисными компаниями. (Основной формат передачи ASCII)

Уровень 1 - на уровне 1 и выше, поток данных превращается в двоичный формат LIS, значения конвертируются согласно типам данных LIS. В дальнейшем данные инкапсулируются в WITS пакет, который помещается между физическими и логическими заголовками и концовками записей, что превращает его в пакет LIS. Введен для компактности.

Уровень 2 - уровень 2 надстраивается на уровень 1 добавлением двусторонней коммуникации через записи определённые под комментарий. Это даёт возможность обмена мгновенными текстовыми сообщениями между пользователями.

Уровень 2b - уровень 2б добавляет опцию буферизации переданных данных, чтобы записи не терялись при разрыве коммуникационного канала.

Уровень 4 - WITS четвертого уровня работает по совершенно другому формату, чем предыдущие уровни начиная с того, что он основан на формировании передачи данных стандарта API RP66. Концепция предопределенных записей и двустороннего обмена остаются.

Несмотря на долгую историю и широкое распространение WITS обладает некоторыми ограничениями, свойственными большинству традиционных EDI-форматов. Эти ограничения можно объединить в две группы.

1. Недостаточная поддержка новых решений в инженерии скважин:
   - Устаревшие форматы записей (особенно относящихся к каротажу во время бурения);
   - Ограниченное число разделов под бурильные колонны;
   - Отсутствие гибкости в использовании различных единиц измерения;
   - Почти полное отсутствие поддержки статической (справочной) информации о скважине;
   - Ограниченная возможность двунаправленного диалога для передачи управленческих решений на скважину.
2. Недостаточная поддержка современных Web-ориентированных программных решений:
   - Бинарный формат (уровень 1) остается платформо-зависимым;
   - Расширения данных (уровень 4) трудны в реализации, так как описывают только синтаксис, без механизмов описания семантики;
   - Не стандартизирован транспортный уровень передачи сообщений. Используется устаревший способ коммуникации точка-точка (RS232);
   - Отсутствует стандартный прикладной программный интерфейс;
   - Слишком много вопросов оставлено для решения при реализации (сложность).
   - Также WITS создает проблемы интерпретации:

- Любые данные необходимо записывать в каналы, что естественно для каротажных кривых, но неудобно для других типов информации;
- Каналы WITS иногда сложно идентифицировать;
- Некоторые типы кривых не определены в спецификациях WITS;
- Назначение каналов WITS иногда изменяется по ошибке или в результате индивидуальной договоренности предприятий, нуждающихся в передаче дополнительных типов данных, и не имеющих такой возможности в рамках существующей жесткой спецификации.

Причины жизнеспособности WITS, как типичного «старого» стандарта:

- WITS хорошо работает и прост в обработке;
- Существующее оборудование поддерживает WITS;
- WITS-2b более надежен в условиях плохой связи: предыстория может быть восстановлена в случае обрыва линии из буфера. WITSML требует передачи пакета целиком;
- WITSML все еще плохо известен и более сложен для использования в полевой обработке;
- Гибкость WITSML в силу отсутствия общепризнанного индустриального образца потенциально чревата дублированием и путаницей в типах данных.

Последние два утверждения требуют пояснений. Теоретически, работа со схемой данных WITSML должна осуществляться достаточно просто, по взаимосвязям объектов: Скважина -> Ствол скважины -> Объект инклинометрии (траектории ствола) -> Объект каротажа. Но каждая сервисная компания реализует эту схему по-своему. Одна хранит и каротажные данные, и информацию об инклинометрии в одном объекте, следовательно, стандартный запрос ML не будет срабатывать. Другая не следует обычной для WITS процедуре авторизации подключения типа точка-точка и т.п. В отличие от многократно растиражированного и давно настроенного WITS практический опыт реализации и применения WITSML еще не устоялся. Помимо

перечисленных проблем общего характера распространению WITSML препятствуют специфические обстоятельства функционирования отрасли.

1.3. Типовая схема работы сервера поддержки стандарта, основные составляющие стандарта

Структурная схема серверной платформы стандарта представлена на рисунке 2. Платформа работает по принципу классической модели «клиент-сервер». Клиентами являются прикладные программы и приложения, станции управления бурением, SCADA-системы. Сервером является система сбора и хранения данных, или WITSML-север. Клиенты всегда выступают инициаторами обмена с сервером. Обмен данными осуществляется сообщениями по протоколу SOAP в виде документов xml, сформированных согласно стандарту WITSML.

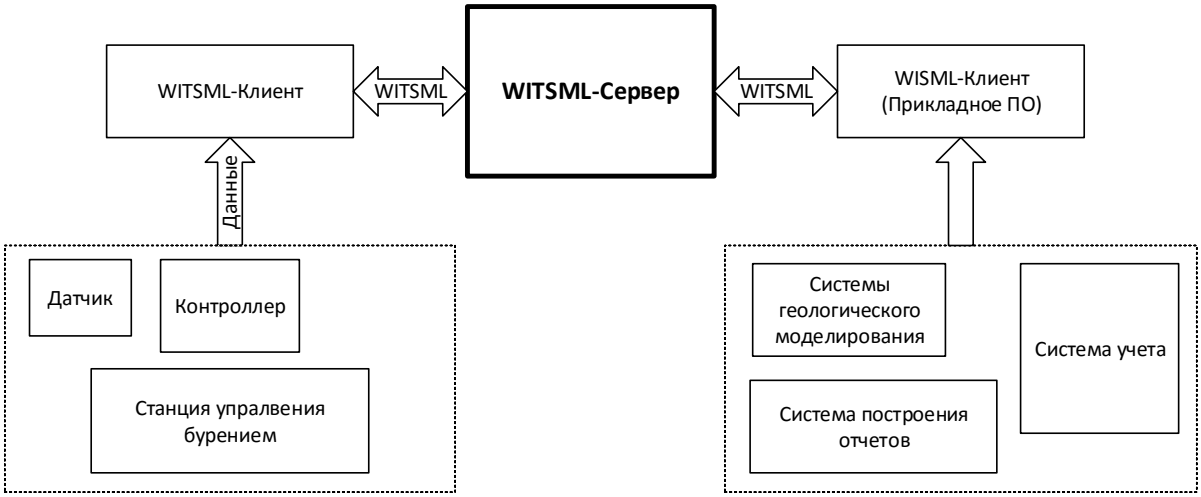

Рис. 2. Структурная схема серверной платформы стандарта WITSML

Общая схема процесса передачи в информационной инфраструктуре «интеллектуального» месторождения, в том числе по унификации технического взаимодействия отечественных кустовых контроллеров управления нефте- и газодобычей, а также станций управления бурением на основе разрабатываемых стандартов представлена на рисунке 3. В качестве основного

стандарта выступает WITSML как основа для всех производных от него стандартов.

Функционально общий алгоритм передачи данных является совокупностью следующих частей:

- Алгоритм работы WITSML-агента (программный модуль передачи данных о состоянии нефте- и газодобычи, предназначенный для обмена данными по стандарту WITSML);
- Алгоритм обработки запросов WITSML-сервером (хранения данных в формате WITSML).

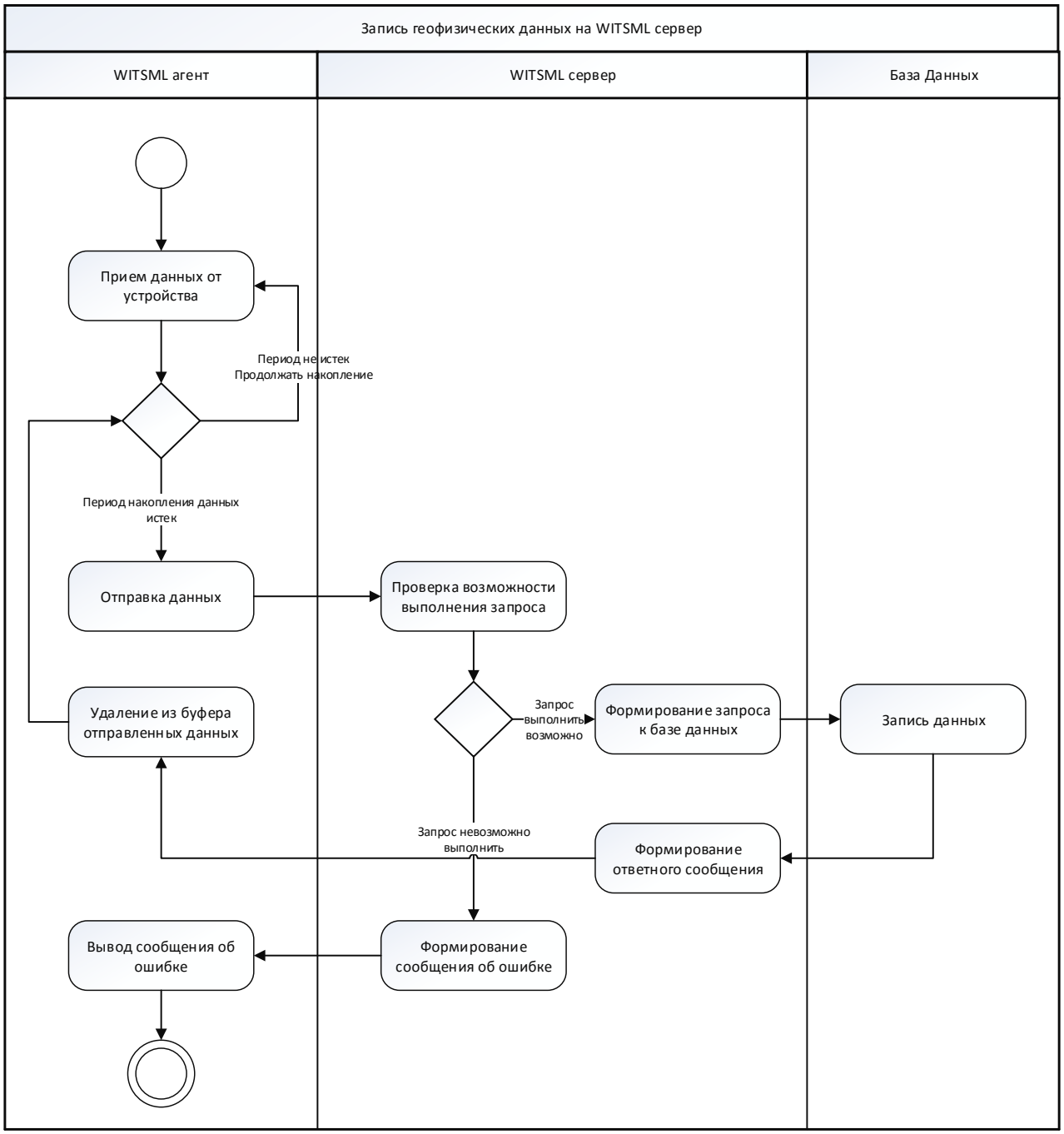

Рис. 3. Схема взаимодействия компонентов передачи геофизических данных в формате WITSML

Для обращения к WITSML-серверу необходимо сформировать сообщение в формате SOAP в соответствии с описанием службы WITSML-сервера. Обязательным параметром данного сообщения является WITSML-документ.

WITSML-документ – это XML-документ, сформированный в соответствии со схемой WITSML. Схема объектов WITSML, описание службы

WITSML-сервера и определенные условия взаимодействия клиента и сервера – в целом есть стандарт WITSML.

Для обеспечения взаимодействия с WITSML-сервером требуется реализовать множество компонентов и алгоритмов, специфичных только для данного стандарта. Таким образом, их можно объединить в отдельный модуль и назвать его WITSML-агентом. При этом данный модуль будет выступать как посредник между каким-либо прикладным программным обеспечением и WITSML-сервером. Основные функции этого модуля сводятся к двум сценариям.

Если программное обеспечение формирует данные (контроллер, станция управления бурением), то основными операциями являются:

- получение порции данных от прикладного ПО;
- конвертирование этих данных в формат WITSML;
- запись данных в формате WITSML на сервер.

Это основные операции, которые будут повторяться через заданные промежутки времени на протяжении всего периода времени работы комплекса бурения.

Если программное обеспечение в основном потребляет данные (моделирующие программные комплексы, программы мониторинга и т.д.), то основными операциями являются:

- получение команды от ПО на запрос данных с сервера;
- создание WITSML-документа;
- выполнение запроса, ожидание ответа от сервера;
- разбор полученного WITSML-документа;
- возврат извлеченных данных прикладному ПО.

Обобщенный алгоритм обработки запросов WITSML сервером представлен на рисунке 4.

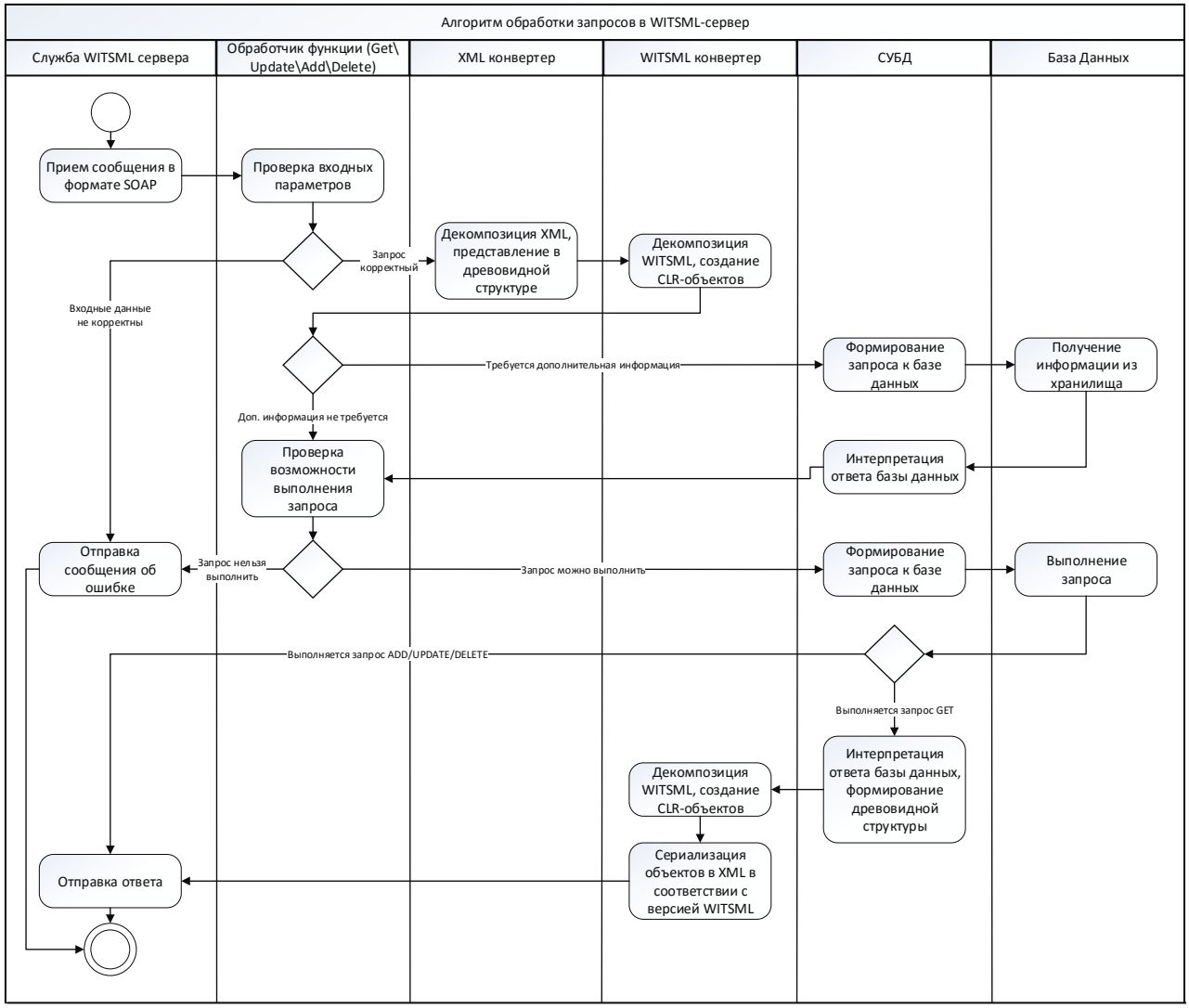

Рис. 4. Схема обработки запросов WITSML сервером

Архитектура программного обеспечения передачи данных в рамках инфраструктуры «интеллектуального» месторождения реализована по принципу модели клиент-сервер.

Инициатором обмена сообщениями и данными всегда выступает клиент. Клиент-WITSML представляет собой модуль, который встраивается в устройство или программное обеспечение и отправляет/получает данные. Типичные операции, которые выполняет WITSML-агент: накапливает некоторое количество данных, конвертирует эти данные в формат WITSML, передает на сервер-WITSML, и получает уведомление об успешном завершении операции, либо об ошибке.

Часть данных, в частности каротажные кривые, агент отправляет на сервер периодически и порционно, по ходу их производства буровой установкой. В случае если агент будет отправлять запросы слишком часто, возрастет нагрузка на сеть и оборудование, и это не приведет к сколько-нибудь значимому увеличению скорости поступления данных, поскольку компоненты инфраструктуры будут перегружены. Для устранения подобных проблем необходимо выбрать наиболее оптимальную частоту отправки данных на сервер.

Рекомендации о периодичности отправки данных агент может получить от сервера. С помощью специальной функции API сервера клиент запрашивает рекомендуемый сервером период отправки и максимальной объем данных, которые сервер готов принять. Исходя из этих параметров, клиент может вычислить периодичность отправки данных. Однако, это не обязательное требование, а рекомендация, и, если поступили важные данные, клиент может отправить их немедленно и сервер будет обязан обработать все поступающие запросы.

Безопасность передачи данных обеспечивается расширенным протоколом HTTP, поддерживающим шифрование – HTTPS. Для авторизации клиентов используется Basic HTTP Authentication метод. При использовании данного метода, пароль и логин передаются в заголовке HTTP пакета, закодированные при помощи base64. Очевидно, это самый ненадежный метод авторизации в комбинации с HTTP. Использование протокола HTTPS устраняет этот недостаток. В этом случае безопасность передачи данных зависит всецело от реализации HTTPS протокола.

Использование протокола HTTPS предпочтительнее, поскольку обеспечивает шифрование, как тела сообщения, так и его заголовков, что важно в связи с используемым методом авторизации. Для вызова функций API сервера и передачи структурированных сообщений используется протокол SOAP. Каждое сообщение между сервером-WITSML и клиентом-WITMSL является SOAP-пакетом. В каждом таком пакете содержится имя вызываемой

функции, WITSML-документ – основная часть пакета, и дополнительное сообщение. Дополнительное сообщение является не пустой строкой при ошибке или нестандартной ситуации, во всех остальных случаях – пустая строка.

Обработка сообщений всего сетевого стека до уровня SOAP протокола осуществляется средствами операционной системы и используемого веб-сервера Microsoft IIS. На вход службе WITSML сервера, запущенной в процессе под управлением веб-сервера, передается лишь содержимое HTTP сообщения виде текста и словарь заголовков. Содержимым является SOAP пакет. А словарь заголовков необходим для проведения авторизации.

Протокол SOAP с точки зрения высокоуровневых информационных систем, таких как WITSML сервер, можно отнести к сетевому стеку и разместить над прикладным уровнем, так как его роль сводится к вызову функций API и передаче сообщений. Такое поведение не является специфичным для разрабатываемой системы. Наоборот, данная функциональность типична для систем, имеющих в своем составе веб-службы, и одинаково реализована во многих программных продуктах. Поэтому платформа .Net включает в себя и предоставляет разработчику полный набор средств для работы с протоколом SOAP, а также средства для автоматической генерации интерфейсов SOAP служб, wsdl файлов и т.д.

Работа с SOAP-протоколом вынесена в отдельный модуль и реализована полностью на основе компонентов .Net Framework. Классы .Net Framework инкапсулируют весь механизм, предоставляя только интерфейс для обработки SOAP вызовов в виде скелета функций с входными аргументами, которые соответствуют полям SAOP пакета. Компоненты сериализации и десериализации сообщений, проверки корректности, генерации wsdl создаются автоматически средствами платформы .NET.

Сообщение SOAP, которым обмениваются клиент и сервер состоит из имени вызываемой функции и нескольких полей. Состав полей может меняться

в зависимости от ситуации и функции. В общем случае запрос от клиента к серверу может включать следующие поля:

- WMLtypeIn – тип объекта в запросе;
- XMLin или QueryIn – входящий документ в формате WITSML, основная часть запроса;
- OptionsIn – опциональные дополнительные параметры;
- CapabilitiesIn – список ограничений клиента, которые должен учесть сервер при формировании ответа.

Ответ сервера клиенту в общем случае может включать следующие поля:

- XMLout – документ в формате WITSML являющийся ответом;
- SuppMsgOut – дополнительная информация;
- Result – код возврата.

После того как было извлечено содержимое SOAP сообщения в виде набора строк, оно передается на вход службе WITSML Сервера. Если при разборе SOAP возникли ошибки, то за уведомление клиента об этом отвечают компоненты .Net и в таком случае запрос не доходит до службы WITSML сервера. Таким образом, на вход службе поступают всегда корректные данные.

Запрос всегда состоит из типа запрашиваемого объекта и текста запроса. Текст запроса – это XML документ, сформированный по стандарту WITSML. Он является основной частью запроса, с ним производится основная работа. Остальные параметры либо содержат дополнительную уточняющую информацию, либо являются вспомогательными (как тип объекта, хотя он и является обязательным, тип объекта всегда можно узнать из тела запроса). На этом этапе производится проверка входных аргументов. Тип объекта и тело запроса обязательные аргументы, и в случае их отсутствия, либо пустых строк, прекращается обработка запроса и возвращается клиенту сообщение с информацией о соответствующей ошибке.

Следующим этапом является разбор тела запроса. Для этого необходимо сначала разобрать текст как XML, а затем как WITSML. Т.е. провести

грамматический анализ WITSML запроса и проверку его синтаксической корректности сначала на соответствие правилам XML, а затем и структуре WITSML.

Разбор XML осуществляется с помощью встроенных средств платформы .Net. После разбора синтаксически правильного документа, создается дерево элементов исходного документа в виде взаимосвязанных объектов. При этом очень важно уделить особое внимание обработке ошибок, если документ был составлен неправильно.

На этапе разбора XML, если документ составлен синтаксически не корректно, дальнейшая обработка сообщения прекращается и клиенту возвращается код ошибки и сообщение, поясняющее и детализирующее ошибку, если таковое доступно для данного кода ошибки. Поскольку парсер стандартный, исключения, которые он выбрасывает, фильтруются по типу. Соответственно типу присваивается код и извлекается сообщения об ошибке.
После успешного разбора XML требуется проверять соответствие передаваемых данных схеме WITSML. В стандартной библиотеке также имеются валидаторы для проверки XML документа на соответствия схеме. Однако, в WITSML существуют несколько особенностей, которые не позволяют применять стандартный валидатор. Одна из таких особенностей заключается в различии интерпретация некоторых элементов в зависимости от контекста. В связи с этим возникла необходимость в разработке специальной реализации валидатора. Его принципиальное отличие от стандартного заключается в том, что он основывается на использовании средства рефлексии (Reflection), имеющейся в исполняющей среду платформы .Net Framework CLR (Common Language Runtime).

Для реализации данного валидатора генерируется набор классов CLR, соответствующих сущностям схемы WITSML, которые имеют набор свойств аналогичных набору элементов сущности. Кроме этого были созданы классы, представляющие типы данных стандарта WITSML и содержащие методы проверки значения на соответствие данному типу. Для некоторых свойств и

классов были заданы атрибуты для их декларативного описания. После построения DOM модели для WITSML-данных созданное дерево элементов передается на вход компонента проверки соответствия схеме.

Данный компонент начинает разбор с корневого элемента. По ходу спуска к листьям дерева, для каждого XML-элемента, исходя из названия тега и контекста создается соответствующий CLR-объект. Далее обрабатываются все дочерние XML-элементы, и если очередной элемент является простым тегом или атрибутом, то создается объект соответствующего типа данных, вызывается метод проверки корректности значения, затем этот объект устанавливается в качестве значения соответствующего свойства ранее созданного CLR-объекта. Если очередной элемент оказывается сложным элементом, т.е. содержащим в себе дочерние элементы, то создается новый объект и производится обработка его дочерних узлов. Проверка сложных объектов осуществляется с помощью атрибутов классов.

При создании объекта, с помощью механизмов рефлексии для него и всех его свойства извлекаются атрибуты, которые являются метаданными, на основе которых и делается вывод о соответствии схеме. Алгоритм повторяется до тех пор, пока не будет пройдено все дерево элементов. Если какому-либо из элементов присваивается значение неправильного типа, размера, или не соответствующее схеме, то генерируется исключение с кодом ошибки, сообщением, поясняющим ошибку, и краткой информацией о контексте ошибки, чтобы ее проще было локализовать. В случае отсутствия ошибок результатом работы компонента является полноценный CLR-объект.

Если объект успешно создан, то он содержит в своих полях гарантированно корректные данные. Следующий этап обработки – это определение типа запроса. Запросы могут быть четырех типов: выборка, добавление, обновление, удаление. Выборка самый просто тип запроса. Для его выполнения нужно определить ограничения выборки, составить запрос к базе данных, разобрать ответ базы данных, перевести его в корректный WITSML-документ, отправить ответ. Остальные три типа запросов включают также шаг

по выявлению контекста изменения данных – определению возможности изменения данных. Например, клиент может ошибочно запросить удаления уникального идентификатора скважины. Такой запрос невозможно выполнить, поскольку уникальный идентификатор является обязательным атрибутом объекта скважины согласно схеме данных стандарта WITSML. Также, как правило, в ответ на запрос об изменении данных клиенту возвращается только код успешного завершения операции. Таким образом, пропускается стадия формирования ответного WITSML документа.

На этапе определения контекста запроса на основе созданных объектов строится в памяти абстрактная иерархия таблиц реляционной баз данных. Таблицы заполняются атрибутами объектов. На этом этапе производится конвертация величин в существующие единицы измерения. Это необходимо в том случае, если данные в хранилище сохранены в одних единицах измерения, а их клиент прислал обновление в других. В таком случае нужно привести все величины к единой системе измерения. Для этого, перед форматированием запроса, в некоторых случаях до, а в некоторых после создания иерархии объектов производится уточняющий запрос к базе данных, выбирающий существующие единицы измерения. Если величина уже присутствует в хранилище, то новые данные конвертируются в единицы измерения присевающих данных и затем происходит обновление. Если же величина в хранилище на этот момент не была задана, то сервер сохраняет данные в тех единицах измерения, в которых были получены величины от клиента. Сервер всегда хранит единицы измерения для всех величин. Клиент или агент также всегда должны сопровождать величины и информацию о единицах их измерения.

После заполнения абстрактных таблиц данными, формируются набор ограничений, с помощью которых указывается контекст изменения или выборки данных. Ограничения формируются в зависимости от нескольких условий. Если тип запроса выборка, то любое значение любого элемента трактуется как предикат. Если тип запроса не выборка, то значение того или

иного элемента зависит от самого элемента. Если это элементы так называемой идентификационной группы, куда включаются идентификаторы и имена самого объекта, скважины и ствола скважины для всех объектов, и еще несколько специфичных для конкретных объектов, то такие значения трактуются тоже как предикат. Но значения всех остальных элементов, трактуются как новые данные, которые нужно сохранить в базе данных. При этом, если тип запроса обновление, и ограничение контекста захватывает слишком широкий диапазон, либо указывает на несуществующее множество объектов данных, то сервер выдаст ошибку, поскольку с большой вероятностью запрос составлен с ошибкой. Также сервер выдаст ошибку, если попытаться добавить объект, идентификаторы которого совпадают с одним из существующих объектов, так как это тоже с большой вероятностью ошибка составления запроса.

Исключение из данной логики обработки запросов составляют запросы типа удаления. В данном виде запросов значения всех элементов кроме идентификационной группы, служащей для ограничения, игнорируются. А информации несет в себе само наличие того или иного элемента. Если какой-либо элемент присутствует в запросе, то он помечается к удалению из базы данных.

После того, как абстрактные таблицы составлены, следует этап создания запроса к базе данных. На этом этапе первым шагом абстрактные таблицы и их содержимое переводится в SQL запрос. Параллельно выявляется возможность исполнения всего кода в одном запросе. В некоторых ситуациях весь запрос не представляется возможным выполнить за один раз, чтобы не нарушить логическую структуру данных. Тогда запросы разбиваются на несколько запросов. При этом, каждый запрос ставится в очередь выполнения, для него создается свой набор параметров и также создается парсер ответа от базы данных. Все запросы выполняются в строго определенном порядке. После того как запрос выполнен, созданный для данного конкретного запроса парсер ответа разбирает данные, которые были получены от базы данных. Если это

запрос типа выборка, то парсер строит также абстрактные реляционные таблицы и наполняет их данными.

После того как запрос выполнился и парсер разобрал ответ базы дынных, управление передается обратно главному обработчику, и он в зависимости от результата работы парсера либо возвращает сообщений только с кодом возврата, либо с документом в формате WITSML, если была выборка. Ответный WITSML документ формируется при помощи того же механизма, который позволяет создавать из иерархии элементов и их значений объект CLR. С помощью этого механизма сначала производится создание объекта из абстрактных таблиц, являющихся результатом работы парсера ответа, затем объект сериализуется в WITSML документ. Документ оборачивается в SOAP пакет и отправляется клиенту в качестве ответа.

1.4. Проблемы создания национального интегрированного стандарта передачи данных для нефтегазовой промышленности, совместимого с перспективным стандартом WITSML2.0

WITSML - стандарт передачи данных со скважины в процессе бурения. Стандартный язык разметки для передачи информации о буровой площадке (WITSML) позволяет эффективно обмениваться информацией о буровой площадке в нефтегазодобывающей промышленности, также в настоящее время возможности стандарта включают: заканчивание, каротажные работы при бурении, расчет и планирование траекторий скважин. В основе WITSML заложена технология XML, имеющая ценность для бизнеса за счет эффективных стандартных протоколов обмена данными. Многие холдинговые, независимые и национальные нефтяные компании уже приняли стандарт WITSML для обмена скважинными данными в режиме реального времени или планируют использовать его в ближайшем будущем. Число совместимого программного обеспечения со стандартом WITSML быстро растет, и на данном этапе стандарт включен в более чем 40 программных продуктов по всему миру. Совместимость со стандартом WITSML становится договорным

обязательством, в частности в отношении буровых работ, связанных с высокой стоимостью, высокой степенью риска или при высокой степени уплотнения скважин на единицу площади.

Развитие стандарта WITSML до версии 2.* предполагает включение в схему стандарта описание данных не только о бурении, но и о процессе нефтедобычи (PRODML) и построение геологической модели (RESQML).

Преимущества использования стандартизованного подхода к обмену информацией о бурении обычно характеризуются следующими аспектами:

- WITSML позволяет повысить отдачу от инвестиций в высокотехнологические области и открывает новые возможности автоматизации для энергетических компаний и оптимизации, что в противном случае будет невозможным или сложным.
- WITSML снижает затраты на обмен информацией между программными приложениями внутри компании-разработчика и между компаниями-разработчиками, совместными предприятиями, партнерами, подрядчиками и контролирующими органами.
- WITSML снижает затраты на замену или замещение программного обеспечения, что в результате усовершенствует функциональные возможности.
- PRODML позволяет оперативно создавать отчеты об объеме добываемых ресурсах, о температурном профиле скважины и в целом информация об отчетах внутренних, партнерских и государственных стандартов.
- RESQML позволяет принимать данные от буровой и строить геологическую модель в реальном времени, что повышает скорость принятия управленческих решений относительно управления создания скважины.

Современное состояние информационной инфраструктуры нефтяного предприятия можно описать, как систему, далекой от единого функционирования всех входящих в нее компонентов. В различных участках производственного процесса нефтедобычи и бурения используются

малосовместимые либо совсем не совместимые устройства и программные технологии, которые зачастую создают сложности и тормозят производственный процесс из-за несвоевременной или слишком долгой интерпретацией и передачи данных между объектами промысла. Преобладание иностранной техники и контроллеров (более 80% промысла) не располагает к развитию отечественного производства технических и программных решений для нефтяных предприятий. Так же современное состояние нефтеразведки и добычи характеризуется массовой разработкой высокоэффективных горизонтальных скважин, при прокладке которых важно выдерживать оптимальную траекторию бурения горизонтального участка.

Существует ряд проблем, с которым можно столкнуться при проектировании и внедрении системы информационной инфраструктуры нефтяного предприятия, совместимой с международным стандартом передачи данных WITSML.

Проблема: Различная номенклатура данных у отечественного программного обеспечения и зарубежного, под которое разрабатывается стандарт WITSML. Корни данной проблемы ведут к различным технологиям добычи нефти в России и за рубежом. И поэтому возникает необходимость расширять схему описания данных стандарта WITSML, учитывая отечественную специфику описания данных о нефтедобычи и бурении.

На примере структуры выходящих данных со станции Геофит выявим различия, которые нельзя найти в утвержденной схеме данных WITSML.

Таблица 1. Часть схемы описания данных со станции Геофит

Закладка	Параметр	Единица измерения
Парам. КУБ	Затяжка	кН
Парам. КУБ	Скорость ТБ	м/с
Парам. КУБ	Расход на вх.	л/с
Парам. КУБ	Плотность 1	г/см3
Парам. КУБ	Вес	кН*10
Парам. КУБ	Момент на кл.	Тс*м
Парам. КУБ	Расход/проявление	%
Парам. КУБ	Давление	атм
Парам. КУБ	Осевая нагрузка	кН
Парам. КУБ	Момент на рот.	Тс*м
Парам. КУБ	Талевый блок	м

Парам. КУБ	Уровень 1	м
Парам. КУБ	Уровень 2	м
Парам. КУБ	Температура бурового раствора на входе	◦C
Парам. КУБ	Уровень 3	м
Парам. КУБ	Уровень 4	м
Парам. КУБ	Нараб. ТК	т*км*100
Парам. КУБ	Температура бурового раствора на выходе	◦C
Парам. КУБ	Плотность 2	г/см3
Парам. КУБ	Вращ.мом.	кН*м
Парам. КУБ	Сигн. газа 1	%НКПР
Парам. КУБ	Расход/погл.	%
Парам. КУБ	Обороты ротора	об/мин
Парам. КУБ	Сигн. газа 2	%НКПР
Парам. КУБ	Сигн. газа 3	мг/м3
Парам. КУБ	Сигн. газа 4	мг/м3
Парам. КУБ	Давление 2	атм
Расчетные	Забой	м
Расчетные	Над забоем	м
Расчетные	Время цирк.	мин
Расчетные	Точка замера	м
Расчетные	Свечей на забое	шт
Расчетные	Расчетный забой	м
Расчетные	ДМК (по времени)	м/час

Теперь обращаясь к утвержденному на данный момент стандарту WITSML 1.4.1 попробуем соотнести названия параметров данных со станции Геофит с данными из схемы стандарта.

Таблица 2. Соотнесение данных

Закладка	Параметр	Единица измерения.
Парам. КУБ	Затяжка Tension	кН (килоДжоулей?)
Парам. КУБ	Скорость ТБ –не понятно чего-талевого блока? Velocity?	м/с m/s ft/s
Парам. КУБ	Расход на вх. Чего? Flowrate In?	л/с bbl/min (баррелей/мин) ft^3/h ($фут^3$/ час)
Парам. КУБ	Плотность 1 Density ECD (equivalent circulating density)?	г/см3 g/cm^3
Парам. КУБ	Вес Чего? Gravity	кН*10? m/s^2 ft/s^2
Парам. КУБ	Момент на кл. Tong torque?	Тс*м
Парам. КУБ	Расход/проявление чего? Flowrate / flow?	% ft^3/h ($фут^3$/ час) bbl/min (баррелей/мин)
Парам. КУБ	Давление Pressure где?	атм psi
Парам. КУБ	Осевая нагрузка Thrust load	кН
Парам. КУБ	Момент на рот. Rotor torque	Тс*м
Парам. КУБ	Талевый блок Casing string	м
Парам. КУБ	Уровень 1 Level (или depth)	м ft
Парам. КУБ	Уровень 2 Level (или depth)	м ft

Парам. КУБ	Температура бурового раствора на входе Mud Temperature In	◦C ◦ C ◦ F
Парам. КУБ	Уровень 3 Level (или depth)	м ft
Парам. КУБ	Уровень 4 Level (или depth)	м ft
Парам. КУБ	Нараб. ТК ? Duration (длительность)	т*км*100 h (в часах)
Парам. КУБ	Температура бурового раствора на выходе Mud Temperature Out	◦C ◦ C ◦ F
Парам. КУБ	Плотность 2 ? Density ECD (equivalent circulating density)	г/см3 g/cm^3
Парам. КУБ	Вращ.мом. Moment of rotation/ angular force/ torsional moment	кН*м
Парам. КУБ	Сигн. газа 1 Signal Gas	%НКПР LFL ????
Парам. КУБ	Расход/погл. Flowrate / lost circulation	% ????
Парам. КУБ	Обороты ротора RT (rotary table) revolutions	об/мин rpm
Парам. КУБ	Сигн. газа 2 Signal Gas	% НКПР LFL ????
Парам. КУБ	Сигн. газа 3 Signal Gas	мг/м3 % ??????
Парам. КУБ	Сигн. газа 4 Signal Gas	мг/м3 %
Парам. КУБ	Давление 2 Pressure	атм psi
Расчетные	Забой Bottom	м m
Расчетные	Над забоем Over bottom	м
Расчетные	Время цирк. Elapsed time	мин s (секунды)
Расчетные	Точка замера Survey point	м
Расчетные	Свечей на забое Stands down the hole	шт
Расчетные	Расчетный забой Reference depth	м
Расчетные	ДМК (по времени) Magnetic log (by time)	м/час

Оценивая результаты проделанной работы можно выявить, что часть параметров названия и единиц измерения можно соотнести с параметрами, описаными в схеме стандарта. Но часть параметров не соотносится с данными из стандарта либо по названию, либо по единцам измерения параметра, либо просто нет точного описания параметра, для того, чтобы можно было однозначно соотнести данные со станции с данными из стандарта. Что делать с теми параметрами, которые не имеют аналога в схеме стандарта? Решением этой проблемы является создание схемы данных национального стандарта передачи данных, учитывающего отечественную специфику нефтедобычи и бурения, и также совместимого с международным стандартом WITSML 2.0. Тем самым национальный стандарт передачи данных будет являтся расширенной схемой исходного WITSML, но с добавлением параметров

описания данных, специфичных для программного обеспечения и оборудования, используемых на реальных Российских промыслах.

Раздел 2 Стандарт WITSML 2.0, как компонент стандарта передачи данных для отечественной нефтегазодобывающей промышленности

2.1. Вопросы интеграции национального стандарта и WITSML2.0

Стандарт WITSML представляет важный вклад в управление данными по добыче и позволяет осуществлять обмен данными с программным обеспечением разного рода. С его помощью осуществляется информационный обмен в режиме реального времени между платформой и головным офисом, и автоматически происходит передача данных в приложения по геологии и геофизике. Таким образом, значительно улучшается поддержка и процесс принятия решений. Данные становятся доступными всем последующим бригадам, работающим на скважине, и результатом такого подхода передачи информации является более высокий конечный результат.

Все замечательно, если речь идет об зарубежном оборудовании и технологиях которые разрабатывают с пониманием стандарта WITSML, то есть используя номенклатуру данных, которая однозначно и определенно интегрирована с международным стандартом. Но как обстоят дела с описанием данных в отечественном нефтепромысле? Зачастую, как видно из практического опыта, на разных нефтепромыслах используют программные продукты от различных производителей: от отечественных до иностранных. И в большинстве своем данное программное обеспечение работает по своим стандартам и протоколам, обеспеченные разработчиками. Тем самым если мы возьмем и попробуем совместить два или более программных продукта от разных производителей, то они просто не смогут друг друга понять из-за различий в форматах данных.

Эта проблема решается простой стандартизацией всех данных, используемых при процессах бурения скважин и добычи углеводоров. Зарубежные компании, например Schlumberger, давно исследуют и разрабатывают программные продукты, которые можно интегрировать с сервером WITSML посредством Real-Time модуля. И возникает следующий

вопрос: а надо ли начинать разработку отечественных программных продуктов, которые будут полностью интегрироваться со стандартом WITSML? Это очень долгий и затратный процесс. Не проще ли будет взять и просто расширить схему международного стандарта под нужды отечественного программного обеспечения для нефтедобычи? Тем самым будем иметь схему стандарта WITSML 2.0 с интеграцией схемы национального стандарта. И это даст следующее преимущество: какое бы не использовали программное обеспечение, зарубежное или отечественное, оно будет понимать данную общую схему стандарта, так как в ней есть описания для всех специфичных данных.

2.2. Основные функции национального стандарта

Экспорт нефти и газа в России остается основным источником внешнего дохода станы. В такой ситуации естественно, что процесс добычи нефти обуславливает стремительное развитие научно-технического прогресса, в том числе процесс бурения давно автоматизирован. Степень вовлеченности человеческих ресурсов минимальна, однако, это приводит к потребности постоянно отслеживать процесс бурения, не только на случай непредвиденных ситуаций, но и с целью повысить экономическую эффективность.

Мониторинг бурения является неотъемлемой частью процесса бурения. Благодаря датчикам, устанавливаемых в каждой скважине и на буровых установках, компании получают своевременные данные обо всех происходящих процессах. Это позволяет принимать быстрые и наиболее правильные управленческие решения, а также эффективно распределять ресурсы.

Существует большое количество датчиков, используемых на буровых установках для мониторинга бурения, и ниже приведены виды подобных датчиков.

1. Датчик глубины (датчик оборотов вала буровой лебедки)
2. Датчик крутящего момента ротора

3. Датчик момента на ключе
4. Датчик оборотов ротора
5. Датчик ходов насоса
6. Датчик давления промывочной жидкости (ПЖ) на входе
7. Датчик потока (расхода) ПЖ на выходе
8. Датчик уровня ПЖ в приемной емкости
9. Датчик плотности ПЖ в приемной емкости
10. Датчик температуры ПЖ на входе (в емкости)
11. Датчик температуры ПЖ на выходе
12. Датчик нагрузки на крюке
13. Датчик электропроводности ПЖ на входе

Таким образом, огромное количество данных непрерывно поступают с буровых установок, что позволяет компаниям строить математические, геологические и гидрологические модели месторождений. Здесь возникает закономерный вопрос, где и как хранить и передавать информацию с буровых установок. Каждый отдельно взятый датчик измерять строго определенные величины, и более того передает их в командный центр в своем собственном формате. В этом заключается одна из основных проблем мониторинга бурения в отечественной промышленности. Исторически сложилось, что отечественные компании, производящие программные продукты для оборудования и непосредственного мониторинга, имеют собственные форматы передачи данных. При переходе от одного уровня обработки к другому происходят множественное преобразование данных, что неизменно приводит к появлению неточностей.

Следовательно, можно выделить главную функцию разработки схемы национального стндарта: обеспечить интеграцию всех программных компонентов информационной инфраструктуры нефтедобывающего предприятия. Данную функцию можно декомпозировать:

- Интеграция сервера национального стандарта и зарубежного оборудования;

- Интеграция сервера национального стандарта и отечественного оборудования;
- Обеспечение передачи данных от скважины до геологической модели в реальном времени.

2.3. Алгоритм формирования схемы данных стандарта

Нефтяная компания, имея у себя зарубежное оборудование и сервер WITSML 2.0, не знает проблем дезинтеграции данных между программными компонентами. Другая нефтяная компания, имеющая у себя отечественное оборудование и сервер WITSML 2.0, сталкивается ссо следующей проблемой: сервер WITSML не понимает структуру описания данных отечественной программы и не может передать информацию от SCADA-системы до базы данных и средств моделирования. Как уже выявили выше: наиболее простой путь решения данной проблемы – расширение номенклатуры данных стандарта WITSML 2.0 с учетом специфики отечественного оборудования и технологий бурения и добычи углеводородов. Естественно нельзя просто так взять и добавить все параметры из всех отечественных продуктов в стандарт WITSML, так как произойдет дублирование некоторых параметров. Следовательно, необходимо использовать определенный алгоритм для внесения параметра в схему стандарта. Пример алгоритма представлен на рисунке 5.

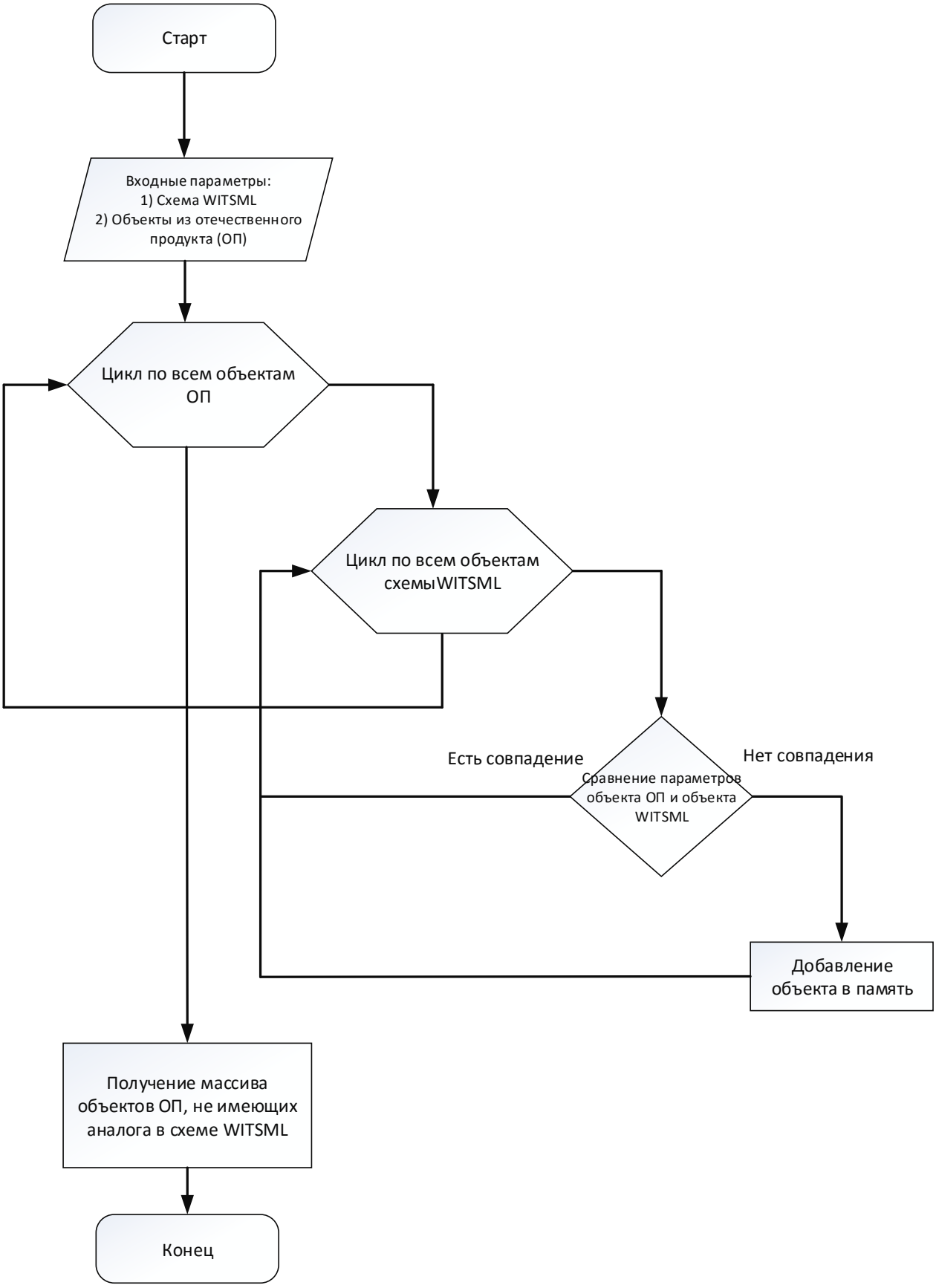

Рис. 5. Обобщенный алгоритм поиска параметров для расширения схемы стандарта

Чтобы расширить схему стандарта, необходимо точно знать, что должно входить в конечную структуру данных. Прежде всего необходимо иметь утвержденную схему международного стандарта и список параметров, собранных с отечественного оборудования. Далее следует каждый параметр соотнести с объектами из схемы данных WITSML. Если есть альтернатива либо полное соответствие объектов, то данные параметры соотносятся со стандартом. Если нет альтернативы объекта отечественного оборудования в стандарте WITSML, то следует данный параметр включить в список кандидатов на добавление в схему национального стандарта, который далее будет включен в международный стандарт WITSML 2.0.

Сформировавшийся список новых значений поступает на оценку эксперту в нефтяной отрасли. Данный эксперт проводит рецензирование на включение тех или иных параметров в общую схему данных национального стандарта. Данный процесс представлен на рисунке 6.

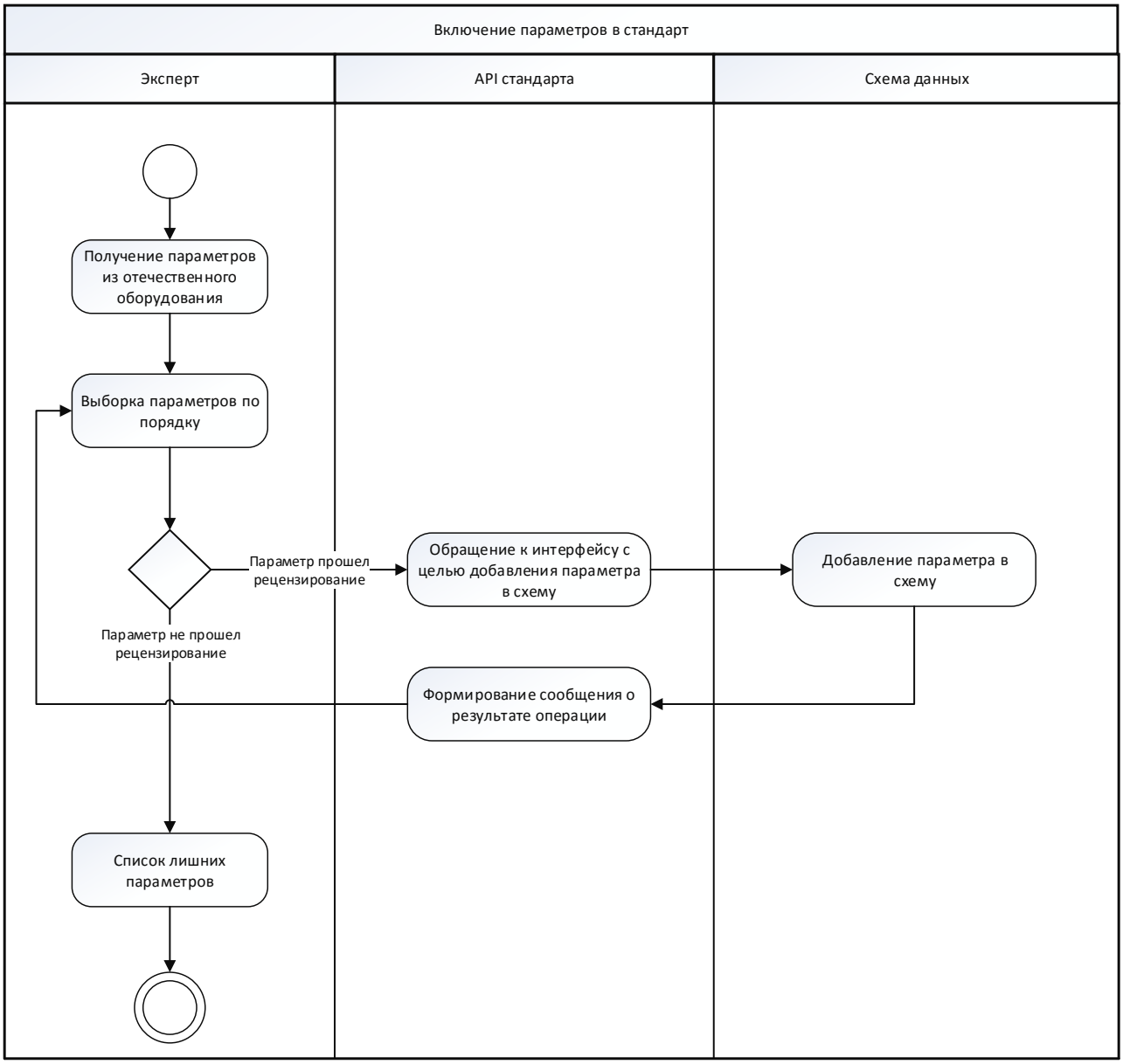

Рис. 6. Процесс ключения экспертом нового объекта в схему описания данных стандарта

2.4. Структура и схема данных WITSML 2.0 с включением национального стандарта

Используя вышепомянутый алгоритм поиска параметров для добавления в схему национального стандарта можно визуализировать обобщенную схему, получающегося результата (рис. 7). Данные от буровой поступают в различных форматах агентам. Агенты передают информацию на хранение в базу данных по протоколу ETP. Схема WITSML, и схема национального стандарта могут работать параллельно, не вмешиваясь в алгоритмы работы программного обеспечения: зарубежное работает с WITSML, отечественное работает со

схемой национального стандарта. Протокол ETP необходим для корректного и точного выполнения прочтения схемы стандарта (проверка версии стандарта, соответствие описание данных со схемой). Инициатором получения информации для построения геологической модели являются программные комплексы для разведки и добычи углеводородов (например, программная платформа Petrel 2015).

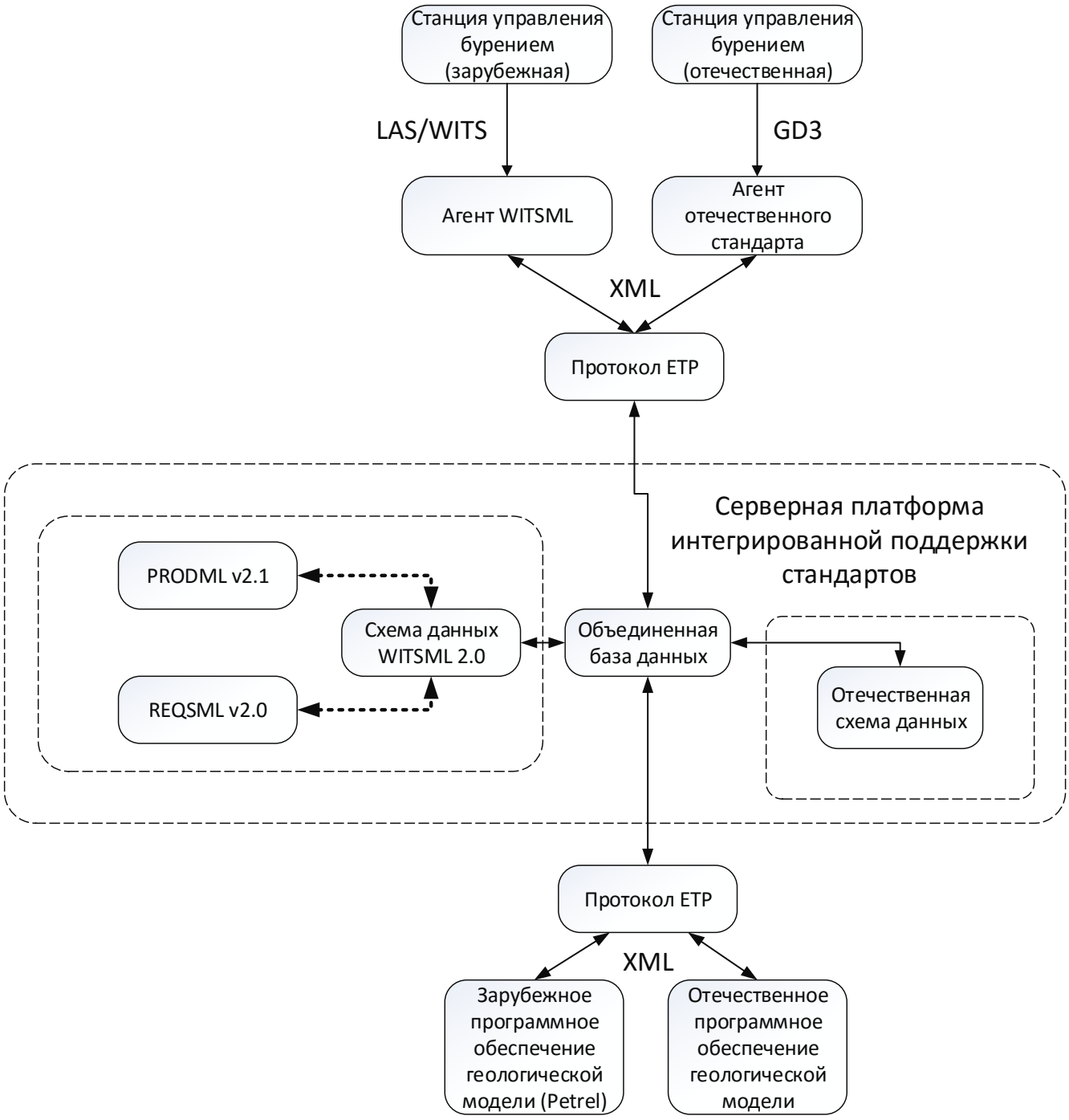

Рис. 7. Обобщенная схема работы и структура интегрированного стандарта, включающего поддержку национального стандарта.

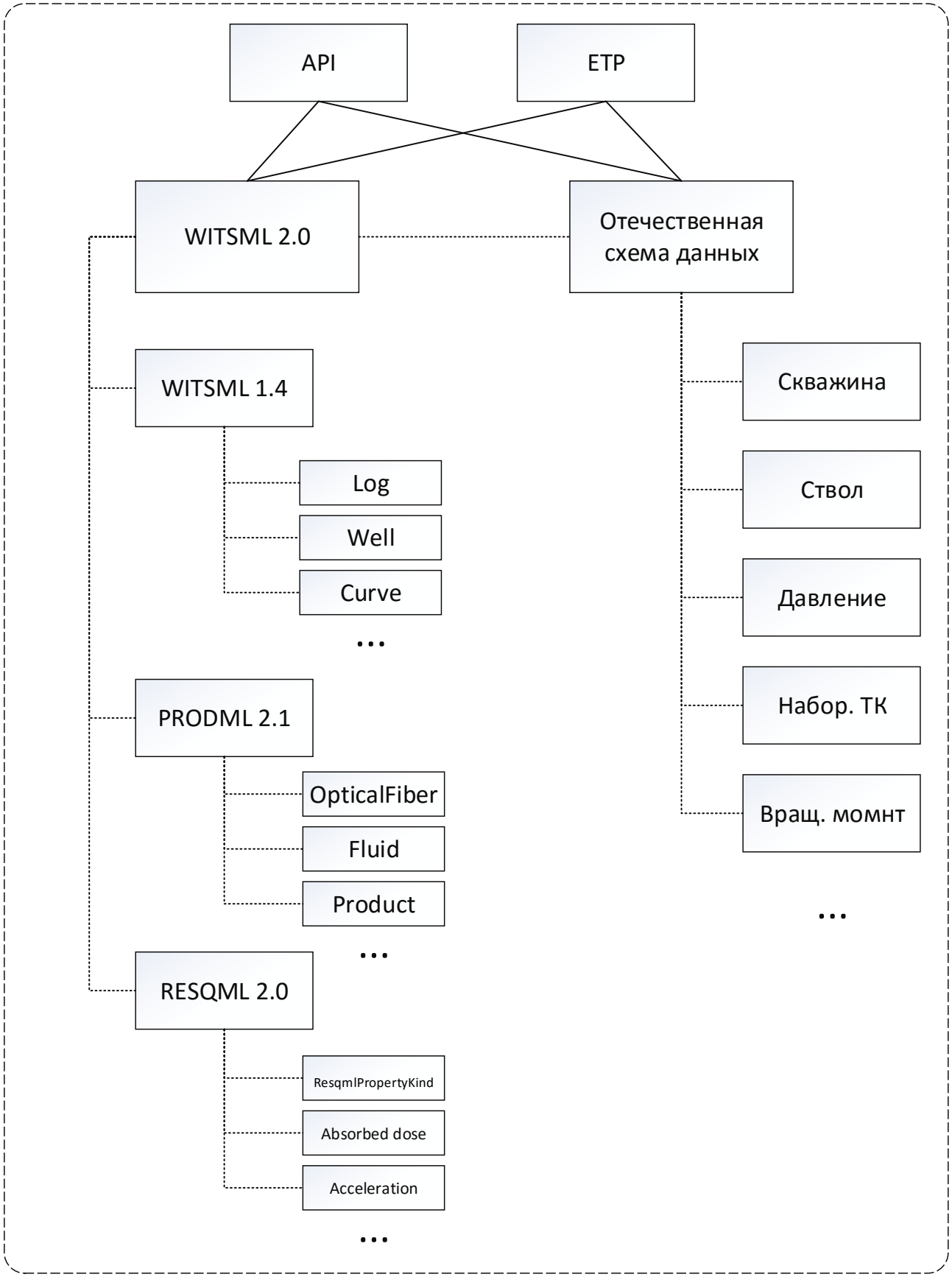

Рис. 8. Обобщенная диаграмма структуры национального стандарта

Одним из важных аспектов является хранение данных о нефтедобычи и бурении. Так как WITSML является простым языком разметки документов, то

никакая компания не будет хранить все данные в текстовых файлах формата *xml или *.csv. Следовательно, необходимо использовать продвинутые средства хранения – это базы данных. Приведем простой пример, как данные собранные в формате WITSML хранятся в базе данных.

Создадим файлы WITSML, PRODML, RESQML со следующим содержанием:

WITSML файл

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="./stylesheets/generic.xsl" type="text/xsl"
media="screen"?>
<wells
      xmlns="http://www.witsml.org/schemas/1series"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.witsml.org/schemas/1series
../xsd_schemas/obj_well.xsd"
      version="1.4.1.1">
      <documentInfo>
            <documentName>well</documentName>
            <fileCreationInformation>
                  <fileCreationDate>2001-10-31T08:15:00.000Z</fileCreationDate>
                  <fileCreator>John Smith</fileCreator>
            </fileCreationInformation>
      </documentInfo>
      <well uid="w-12">
            <name>6507/7-A-42</name>
            <nameLegal>Company Legal Name</nameLegal>
            <numLicense>Company License Number</numLicense>
            <numGovt>Govt-Number</numGovt>
            <dTimLicense>2001-05-15T13:20:00.000Z</dTimLicense>
            <field>Big Field</field>
            <country>US</country>
            <wellDatum uid="KB">
                  <name>Kelly Bushing</name>
                  <code>KB</code>
                  <elevation uom="ft" datum="SL">78.5</elevation>
            </wellDatum>
            <wellDatum uid="SL">
                  <name>Sea Level</name>
                  <code>SL</code>
                  <datumName namingSystem="EPSG" code="5106">Caspian
Sea</datumName>
            </wellDatum>
      </well>
</wells>
```

PRODML файл

```
<?xml version="1.0" encoding="UTF-8"?>
<prodml:fiberOpticalPaths xmlns:prodml="http://www.prodml.org/schemas/1series"
xmlns:witsml="http://www.witsml.org/schemas/1series"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.prodml.org/schemas/1series
../xsd_schemas/obj_fiberOpticalPath.xsd">
      <prodml:documentInfo>
      <prodml:documentName>Optical Path example</prodml:documentName>
</prodml:documentInfo>
      <prodml:facilityIdentifier>
            <prodml:name>Sample Well</prodml:name>
      </prodml:facilityIdentifier>
      <prodml:fiberOpticalPath uid="OP1">
```

```
            <prodml:name>Optical Fiber 1</prodml:name>
                  <prodml:inventory>
                        <prodml:segment uid="10">
                              <prodml:name>Surface Fiber</prodml:name>
                              <prodml:type>fiber</prodml:type>
                              <prodml:fiberLength
uom="ft">127</prodml:fiberLength>
                              <prodml:overStuffing
uom="ft">3</prodml:overStuffing>
                              <prodml:fiberConveyance>
                                    <prodml:permanent>

      <prodml:permanentCableInstallationType>buried parallel to
tubular</prodml:permanentCableInstallationType>
                                    </prodml:permanent>
                              </prodml:fiberConveyance>
                        </prodml:segment>
                        <prodml:segment uid="30">
                              <prodml:name>Downhole Fiber</prodml:name>
                              <prodml:type>fiber</prodml:type>
                              <prodml:fiberLength
uom="ft">1500</prodml:fiberLength>
                              <prodml:overStuffing
uom="ft">50</prodml:overStuffing>
                              <prodml:fiberConveyance>
                                    <prodml:intervention>

      <prodml:interventionConveyanceType>coiled
tubing</prodml:interventionConveyanceType>
                                    </prodml:intervention>
                              </prodml:fiberConveyance>
                        </prodml:segment>
                  </prodml:inventory>
      </prodml:fiberOpticalPath>
</prodml:fiberOpticalPaths>
```

RESQML файл

```
<?xml version="1.0"?>
<enumListSet
      xmlns="http://www.energistics.org/energyml/data/resqmlv2"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.energistics.org/energyml/data/resqmlv2
                  enumValuesResqml.xsd"
      version="2.0.1">
      <enumList>
            <name>ResqmlPropertyKind</name>
            <description>Defines classes of properties. A property inherents the
underlying characteristis of its parent but specializes its
meaning.</description>
            <namingSystem>Energistics</namingSystem>
            <value>
                  <name>absorbed dose</name>
                  <description>The amount of energy absorbed per
mass.</description>
                  <isAbstract>false</isAbstract>
                  <parentKind>quantity</parentKind>
            </value>
            <value>
                  <name>acceleration linear</name>
                  <isAbstract>false</isAbstract>
                  <parentKind>quantity</parentKind>
            </value>
      </enumList>
</enumListSet>
```

В файле WITSML описываются данные о скважине: уникальный идентификатор, ее название, локация. У скажины также есть вложенные объекты, содержащие некоторую информацию (вкладыш трубы, высота над уровнем моря и др.). В файле PRODML описываются данные об оптоволоконном датчике. Далее, приведем пример хранения информации об этой скважине во фрагменте базе данных (рис. 9).

Well			
uid	name	field	country
w-12	6507/7-A-42	Big Field	US

WellDatum			
uid	**wellUid**	name	code
KB	**w-12**	Kelly Bushing	KB
SL	**w-12**	Sea Level	SL

Рис. 9. Пример хранения данных из XML файла

Далее приведем фрагмент схемы предполагаемой структуры базы данных с включением стандартизированных параметров из отечественных систем (рис. 10).

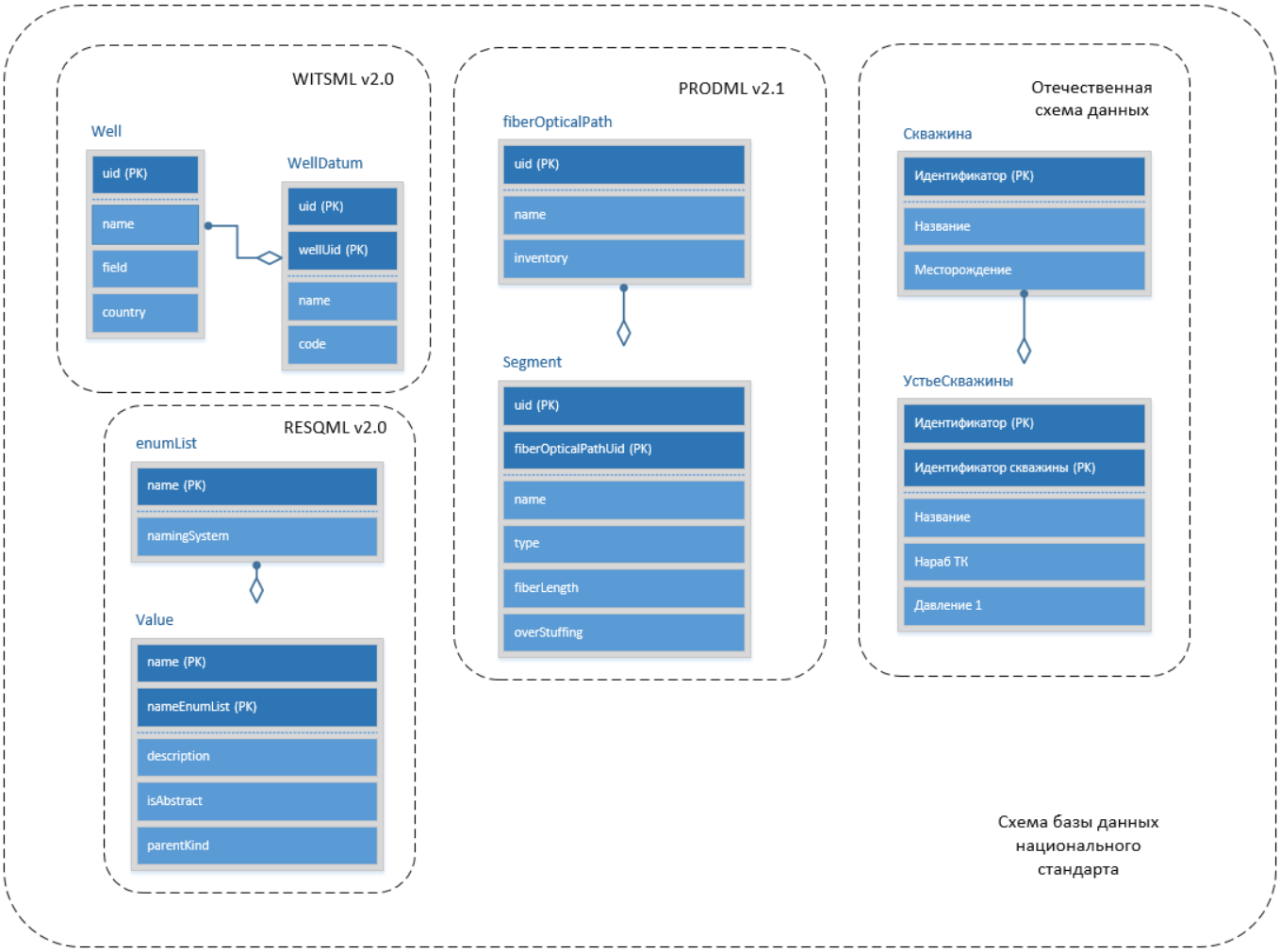

Рис. 10. Фрагмент схемы фрагмента базы данных национального стандарта

Следовательно, можно сделать вывод о том, что включение дополнительных параметров в схему данных стандарта повлечет за собой изменение структуры базы данных. И изменение структуры базы данных поведет за собой следующее неприятное событие – программы, настроенные на интегрированность с утвержденным международным стандартом просто перестанут понимать измененную схему базы данных. Тем самым напрашивается вывод о том, что если рассматривать изменение структуры стандарта и схемы данных, то следует учитывать следующую деталь – ни один объект или параметр из утвержденного стандарта не должен быть изменен в угоду разрабатываемому национальному стандарту.

Говоря иными словами, не следует включать в параметры объектов утвержденной схемы данных новые параметры из объектов отечественных

программных продуктов. На уровне представления базы данных это будет выглядеть как добавление новых сущностей и внешними ключами (если они потребуются) от сущностей утвержденного стандарта. Но ни в одной сущности исходного стандарта не следует добавлять внешний ключ к новой сущности, определенной из отечественного оборудования.

Раздел 3 Анализ преимуществ использования информационной инфраструктуры, интегрированной с национальным стандартом

3.1. Обоснование использования технологии XML

Язык разметки документов - это набор специальных инструкций, называемых тэгами, предназначенных для формирования в документах какой-либо структуры и определения отношений между различными элементами этой структуры. Тэги языка, или, как их иногда называют, управляющие дескрипторы, в таких документах каким-то образом кодируются, выделяются относительно основного содержимого документа и служат в качестве инструкций для программы, производящей показ содержимого документа на стороне клиента. В самых первых системах для обозначения этих команд использовались символы <и>, внутри которых помещались названия инструкций и их параметры. Сейчас такой способ обозначения тэгов является стандартным.

Использование гипертекстовой разбивки текстового документа в современных информационных системах во многом связано с тем, что гипертекст позволяет создавать механизм нелинейного просмотра информации. В таких системах данные представляются не в виде непрерывного потока текстовой информации, а набором взаимосвязанных компонентов, переход по которым осуществляется при помощи гиперссылок.

XML (Extensible Markup Language) - это язык разметки, описывающий целый класс объектов данных, называемых XML- документами. Этот язык используется в качестве средства для описания грамматики других языков и контроля за правильностью составления документов. Т.е. сам по себе XML не содержит никаких тэгов, предназначенных для разметки, он просто определяет порядок их создания. Таким образом, если, например, мы считаем, что для обозначения элемента *скважина* в документе необходимо использовать тэг <well>, то XML позволяет свободно использовать определяемый нами тэг и мы можем включать в документ фрагменты, подобные следующему:

```
<well>Скважина</well>
```

Набор тэгов может быть легко расширен. Если, предположим, мы хотим также указать, что описание скважины должно по смыслу идти внутри описания набора скважин из месторождения, то просто задаем новые тэги и выбираем порядок их следования:

```
<wells>
    <well>Скважина №1</well>
    <well>Скважина №2</well>
</wells>
```

И далее, можно так же создать более подробное описание данных объектов.

```
<wells>
    <well uid=”1”>
    <name>Well_1</name>
    <field>Field_1</field>
    </well>
    <well uid=”2”>
    <name>Well_2</name>
    <field>Field_2</field>
    </well>
</wells>
```

Как видно, сам процесс создания XML документа очень прост и требует от нас лишь базовых знаний HTML и понимания тех задач, которые мы хотим выполнить, используя XML в качестве языка разметки. Таким образом, у разработчиков появляется уникальная возможность определять собственные команды, позволяющие им наиболее эффективно определять данные, содержащиеся в документе. Автор документа создает его структуру, строит

необходимые связи между элементами, используя те команды, которые удовлетворяют его требованиям и добивается такого типа разметки, которое необходимо ему для выполнения операций просмотра, поиска, анализа документа.

XML Schema — язык описания структуры XML-документа. Спецификация XML Schema является рекомендацией W3C.

Как большинство языков описания XML, XML Schema была задумана для определения правил, которым должен подчиняться документ. Но, в отличие от других языков, XML Schema была разработана так, чтобы её можно было использовать в создании программного обеспечения для обработки документов XML.

После проверки документа на соответствие XML Schema читающая программа может создать модель данных документа, которая включает:

- словарь (названия элементов и атрибутов);
- модель содержания (отношения между элементами и атрибутами и их структура);
- типы данных.

Каждый элемент в этой модели ассоциируется с определённым типом данных, позволяя строить в памяти объект, соответствующий структуре XML-документа. Языкам объектно-ориентированного программирования гораздо легче иметь дело с таким объектом, чем с текстовым файлом.

Другим удобством XML Schema является то, что один словарь может ссылаться на другой, и, таким образом, разработчик может использовать уже существующие словари и легче устанавливать и распространять стандарты XML структуры для определённых задач (например, словарь протокола SOAP).

Файл, содержащий XML Schema, обычно имеет расширение «.xsd» (XML Schema definition).

В частном случае, если имеется утвержденная международным консорциумом XML схема, не нужно придумывать параметры и объекты для описания данных, так как можно взять данные описания из схемы.

3.2. Определение потребности в оборудовании и ресурсах

Прежде всего, клиенту важно знать какое количество параметров позволяет передавать стандарт WITSML, и соответственно, хранить сервер, а также существующие ограничения для хранения данных.

Ниже представлен пример информационной таблицы об услугах сервера.

Таблица 3. Пример информационной таблицы об услугах сервера

	Активные скважины	Неактивные скважины
Максимальное количество параметров, снимаемых с одной скважины	7500	-
Максимальное количество скважин	Неограниченно (определяется индивидуальными запросами)	Неограниченно (определяется индивидуальными запросами)
Максимальная частота съема данных (в сутки)	86400	-
Доступ к данным	В режиме реального времени	В течение 10 минут

Возможности для активных и неактивных скважин различаются, в первую очередь, потому что для неактивных скважин не происходит пополнение данных, и соответственно, они хранятся не на основном сервере, а в системе резервного хранения.

Таким образом, для активных скважин максимальное количество параметров, которое позволяет хранить сервер, определено исходя из возможностей стандарта WITSML.

Максимальное количество скважин для каждого пользователя будет определяться индивидуального, согласно его потребностям и возможностями

сервера, и отражаться в договоре, как для активных, так и для неактивных скважин.

Также предположительно датчики на буровых установках могут снимать параметры ежесекундно, поэтому максимальная частота съема данных представляет собой количество секунд в сутках.

Клиенту предоставляется возможность работать со своими данными с активных скважин в режиме реального времени, и в пределах приблизительно десяти минут для неактивных скважин. При этом, время ожидания может разниться в зависимости от местонахождения скважин, типа соединения и буферного времени работы сервера.

Такая таблица позволит клиенту изначально понять условия хранения данных на сервере, и определить целесообразность собственных запросов и потребностей в ключе покупки услуг сервера.

Расчет потребности в ресурсах памяти имеет первостепенную важность для формирования экономической модели оказания услуг сервера. Корректно рассчитанное количество памяти необходимое для работы сервера позволит определить расходы на оборудование и персонал, а соответственно рассчитать оптимальную стоимость услуги.

Необходимо различать оперативную и дисковую память. Оперативная память обеспечивает быстрый доступ к данным, а дисковая - долговременное хранение. Здесь будет рассмотрен алгоритм расчета потребности в ресурсах оперативной памяти.

Данные о процессе бурения скважин поступают на сервер с периодичностью в одну секунду. Каждый пакет данных представляет собой файл с 7000 параметров, снятых с датчиков на буровой установке.

Предположим, каждый такой файл занимает n байт памяти, тогда мы можем рассчитать, что с одной скважины на сервер поступает Vчас=3600*n байт данных за один час.

Если предположить, что разработка ведется круглосуточно, и без остановки в течение года, то одна скважина будет занимать $V_{год}$ =31536000*n байт. Это приблизительно $V_{год}$ =30,075*n Мбайт.

Теперь следует указать, что общее число скважин равно k, тогда общий объем памяти, который будет использован за год, составляет:

$V_{сум/год}$ = 30,075*n*k.

Предположим, что один файл данных со скважины имеет объем 8 байт, и максимальное количество скважин, обслуживаемых сервером, составляет 200. Исходя из выше описанной формулы, рассчитаем необходимость в ресурсах памяти:

$V_{сум/год}$ = 30,075*n*k = 30,075*8*200 = 48120 Мбайт = 46,99 Гбайт.

Таким образом, была определена потребность в ресурсах оперативной памяти на один год при таких исходных данных. Можно сказать, что на один год для функционирования работы сервера потребуется приблизительно 50 Гбайт памяти.

Подобные расчеты позволяют определить количество и функциональные характеристики оборудования, обеспечивающего работу сервера.

Выше были определены потребности в ресурсах памяти, из которых можно сделать вывод о необходимом оборудовании.

Прежде всего, оказание услуг заключается в предоставлении серверного места для хранения бурящихся скважин. Для обеспечения безопасности и бесперебойного доступа к данным необходимо обеспечить сервер резервного хранения, который будет в состоянии заменить основной сервер в случае неполадок.

Так как после завершения бурения скважины, данные на сервер поступать не будут, скважины переходит в категорию неактивных, и хранить ее на основном сервере не целесообразно. Поэтому для хранения данных с неактивных скважин необходима отдельная система хранения, доступ к которой будет осуществляться по запросу клиента.

Для обеспечения доступа к данным в режиме реального времени важно обеспечить сервер бесперебойным питанием и постоянным подключением к сети. Поэтому существует потребность в источниках бесперебойного питания и в услугах двух провайдеров.

Подводя итоги всего вышесказанного можно определить потребность в оборудовании для работы сервера WITSML:

1. Сервер для хранения данных с минимальным объемом памяти 30,075*k*n;
2. Сервер резервного хранения;
3. Система хранения с минимальным объемом памяти не менее 50Тбайт;
4. Источники бесперебойного питания.

Исходя из определенной потребности в оборудовании, можно рассчитать потребность в трудовых ресурсах, необходимых для обслуживания этого оборудования.

Учитывая потребности в оборудовании можно выделить определенное количество сотрудников, обеспечивающих работу сервера.

Прежде всего, возникает потребность в сотрудниках, обслуживающих оборудование. Для основного сервера хранения, резервного сервера хранения и системы хранения неактивных скважин целесообразно нанять двух взаимозаменяемых сотрудников – системный администратор системы хранения и системный администратор резервного хранения.

Также для корректной работы портала и в целом оказания услуги, возникает необходимость в системном администраторе баз данных.

Подводя итог, можно определить приблизительный состав персонала:

1. Руководитель проекта;
2. Системный администратор БД;
3. Системный администратор системы хранения;
4. Системный администратор резервного хранения.

Определение штата сотрудников позволит рассчитать фонд заработной платы и сопутствующие расходы на содержание персонала.

3.3. Обзорная схема применения стандарта

Стандарт реализован на платформе XML, и так как данный язык разметки является расширяемым, в него могут быть добавлены самые различные объекты. На рисунке 11 представлена типовая схема использования расширяемого стандарта WITSML в составе информационной инфраструктуры нефтедобывающего предприятия.

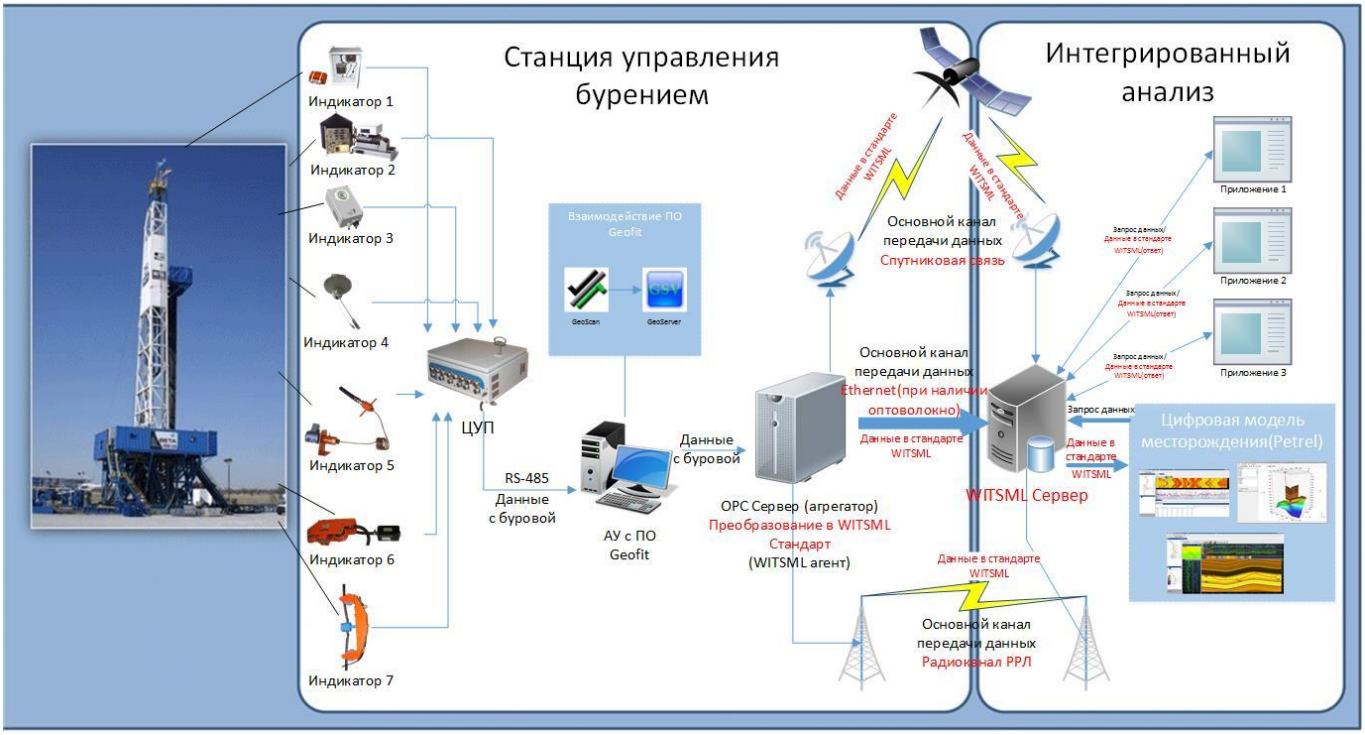

Рис. 11. Информационная структура нефтяного предприятия с использованием стандарта WITSML

Пошаговое описание процесса сбора данных о бурении в реальном времени:

1) Сбор данных датчиками при бурения и отправление их в центр управления процессами;
2) Центр управления процессами аггрегирует данные и отправляет их в программное обеспечение регистрации технологических процессов;

3) Далее, данные с буровой отправляются на агрегатор (OPC-сервер или WITSML агент), который преобразовывает данные в формат, сответствующий стандарту WITSML;
4) Агрегатор отправляет данные на WITSML сервер, который в свою очередь передает информацию на хранение в базу данных и на построение цифровой модели месторождения.

Из всего вышесказанного можно выделить следующие преимущества применения стандарта:

- Так как данные приходят от буровой до построения модели в реальном времени, появляется возможность принимать своевременные управленческие решения по поводу манипулированием процессом бурения;
- Стандартизированный обмен данными позволяет интегрировать оборудование и программные компоненты с различными типами описания данных между собой;
- Расширемый язык разметки позволяет гибко настраивать и добавлять новые объекты в описание схемы данных.

Раздел 4.Проектирование модели процесса взаимодействия с клиентом

4.1. Описание требований к интерфейсу пользователя

В рамках исследуемой области требуется создать приложение, с помощью которого конечный пользователь сможет взаимодействовать с WITSML сервером и получать актуальную агрегированную информацию о состоянии скважин, месторождения и самого сервера.

Цели создания приложения:

1) Предоставить пользователям интерфейс взаимодействия с сервером WITSML;
2) Предоставить описание и другую информацию о WITSML-сервере.

Задачи, решаемые при помощи приложения:

1) Предоставление необходимой информации заинтересованным лицам о WITSML сервере;
2) Обеспечение обратной связи пользователей с разработчиками WITSML-сервера;
3) Управление WITSML-сервером.

Программное обеспечение, используемое для разработки данного web-приложения:

1) Microsoft Server 2012 – операционная система сервера;
2) Microsoft Visual Studio 2015 – среда разработки программы;
3) Team Foundation Server 2013, Git.Lab212 – серверы для коллективной разработки и контроля версий программы;
4) Microsoft SQL Server Express – система управления базой данных для данного приложения;
5) Дополнительные расширения для Visual Studio:
   - Twitter Bootstrap 3 – набор графических решений для сайтостроения;
   - Knockout.js - библиотека скриптов;

- Resharper 9 – система контроля кода программы;
- Git – система контроля исходного кода и версий приложения;
- Entity Framework 6 – технология для связи с базой данных.

Общая схема взаимодействия интерфейса с сервером и базой данных (рисунок 12).

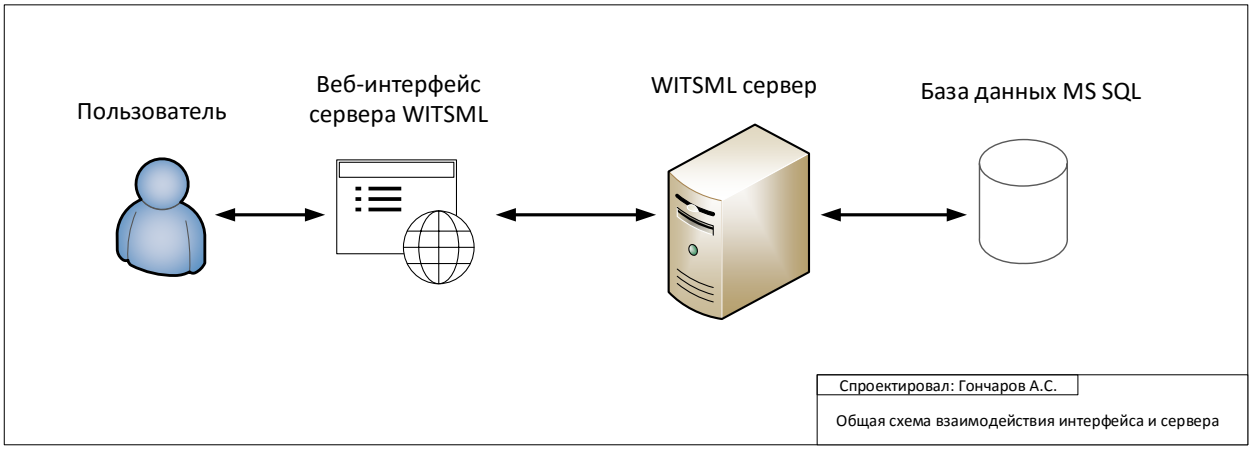

Рис. 12. Схема взаимодействия интерфейса с сервером

Предусмотренные роли пользователей:

1. Анонимные пользователи (гости);
2. Пользователи с тестовым доступом;
3. Активные авторизованные пользователи;
4. Администраторы web-приложения.

Структура приложения:

1. Раздел анонимного пользователя. Это корневая страница, на которую попадают все пользователи при наборе адреса приложения в браузере. Данный раздел содержит следующие страницы:
   - Главная страница. Данная страница должна содержать общую информацию;
   - Регистрация нового пользователя;
   - Авторизация пользователя.

2. Раздел для вновь зарегистрированных пользователей с руководством по использованию и призывом активации тестового доступа. Данный раздел содержит следующие страницы:
    - Страница с руководством по использованию, различными призывами, дополнительной информацией и т.д.;
    - Страница создания и настройки тестовой скважины.
3. Основной интерфейс пользователя. Данный раздел должен содержать следующие страницы:
    - Страница, на которую пользователь попадает после авторизации, содержит краткий свод важной информации и элементы для быстрого доступа к разделам;
    - Раздел «скважины».
4. Интерфейс администратора:
    - Список пользователей, их группировка и управлении;
    - Редактирования политик.

Функционал сайта:

1. Гостевая страница для анонимных пользователей (Landing Page), с краткой информацией о сервере WITSML, тарифных планах, контактах.
2. Предоставление вновь зарегистрированным пользователям пробного периода использования севера на срок и количество скважин, заданных в соответствующей политике;
3. Регистрация/аутентификация пользователей;
4. Личный кабинет пользователя, в котором должна присутствовать следующая функциональность:
    - Создать скважину;
    - Удалить, переименовать скважину;
    - Заморозить/активировать скважину;
    - Группировка скважин по группам (т.е. создание сущности – группа скважин);

- Объединение скважин/групп скважин в месторождения;
- Просмотр скважин: какая из них активна (на основе последнего обновления), дата последнего обновления;
- Просмотр списка траекторий и кривых, вывод логов кривых и экспорт в Excel;
- Просмотр параметров текущего тарифного плана, срок, варианты действий;
- Переход на другой тарифный план.

5. Удалённый запуск тестов Energistics, для авторизованных пользователей и тестового раздела сервера.
   - Просмотр содержимого теста;
   - Просмотр результатов теста;
   - Просмотр успешности прохождения теста;
   - Запуск теста на выполнение, с перезаписью результатов для авторизованного пользователя.
6. Интерфейс администратора сервера, в котором реализованы следующие функции:
   - Создание, удаление, блокировка, разблокировка пользователей;
   - Создание, удаление, активирование, деактивация скважин;
   - Создание/изменение и назначение политик пользователей;
   - Просмотр всех пользователей, групп, компаний, их удаление/редактирование.

   Требования к системе администрирования:

Общее требование - система администрирования должна быть построена на основе пользовательских политик.

Управление правами доступа основывается на параметрах политики пользователя.

Пользовательская политика представляет из себя набор параметров, которые в свою очередь определяют возможности пользователя. К примеру,

если применить к пользователю политику тестового доступа, в которой задано, что срок доступа 10 дней, и количество скважин 1 шт., то пользователь, будет иметь доступ к одной скважине в течении 10 дней с момента применения политики.

Механизм политики пользователей должен быть реализован со следующими особенностями:

1. Администратор может создавать/удалять/редактировать политики;
2. При редактировании политики, прежний вариант политики сохраняется в архиве;
3. Политики должны разделятся на актуальные и не актуальные:
4. Каждому пользователю одновременно можно применить только одну политику;
5. Можно применить к пользователю политику из архива;

У пользователя сохраняется история применявшихся к нему политик.

4.2. Проектирование веб-приложения

Веб-интерфейс – средство взаимодействия пользователя с веб-сайтом или другим приложением через браузер. Популярность веб-интерфейса растет прямо пропорционально росту всемирной паутины. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется на сервере, обмен информацией происходит по сети. Преимущества веб-интерфейса перед любым другим интерфейсом, сделанным под какую-либо операционную систему, является его кроссплатформенность, так как для доступа к веб-интерфейсу необходим браузер. И, так как браузеры разработаны под большинство современных используемых операционных систем, для взаимодействия с сервером через данный интерфейс не составляет проблем. Преимущества такого подхода:

1) Пользователь может добраться до сервера с любого устройства (от настольных компьютеров до сотовых телефонов), которое подключено к глобальной сети;

2) Простота – минимум управляющих элементов в одном окне;
3) Привычность интерфейса – большинство современных людей систематически пользуются браузерами;
4) Простота программирования – 3-х уровневая структура приложения: Представление, Контроллер, Модель.

Для того, чтобы понять, что может пользователь делать на сайте и для чего сайт ему нужен, необходимо создать диаграмму вариантов использования (Use Case Diagram). На диаграмме будут показаны действия, которые предусмотрены для выполнения пользователю, находящегося в определённой роли (рисунок 13).

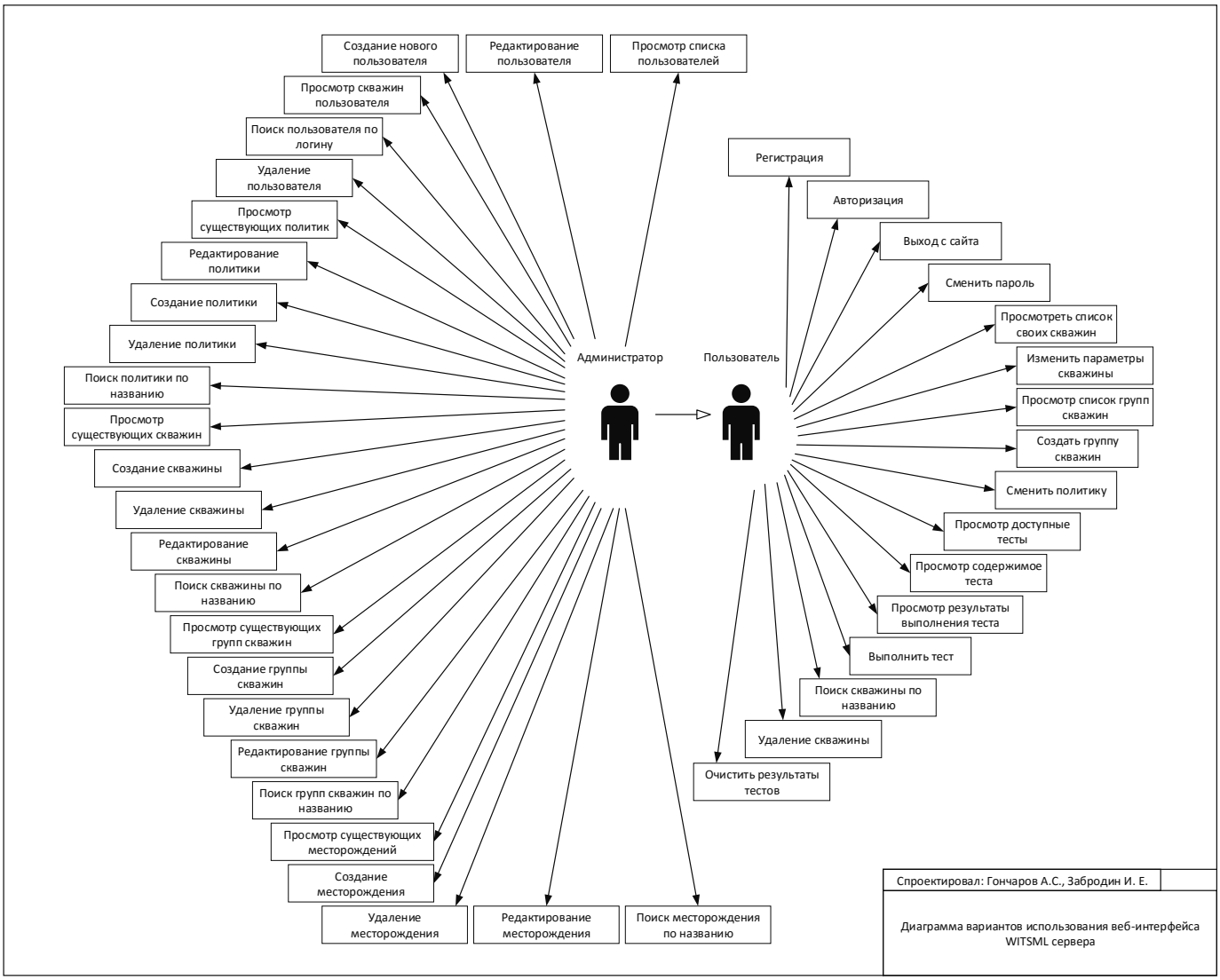

Рис. 13. Диаграмма вариантов использования веб-интерфейса WITSML сервера

Создается приложение на платформе .NET Framework 4.5 со встроенным функционалом авторизации и аутентификации. Для каждой страницы пользовательского интерфейса создается представление *.cshtml. Так как приложение является клиент-серверным, можно представить структуру передачи данных от интерфейса пользователя до сервера и обратно. Схема обращений интерфейса по функциям регистрации, авторизации, создания пользователя к серверу представлена на рисунке 14.

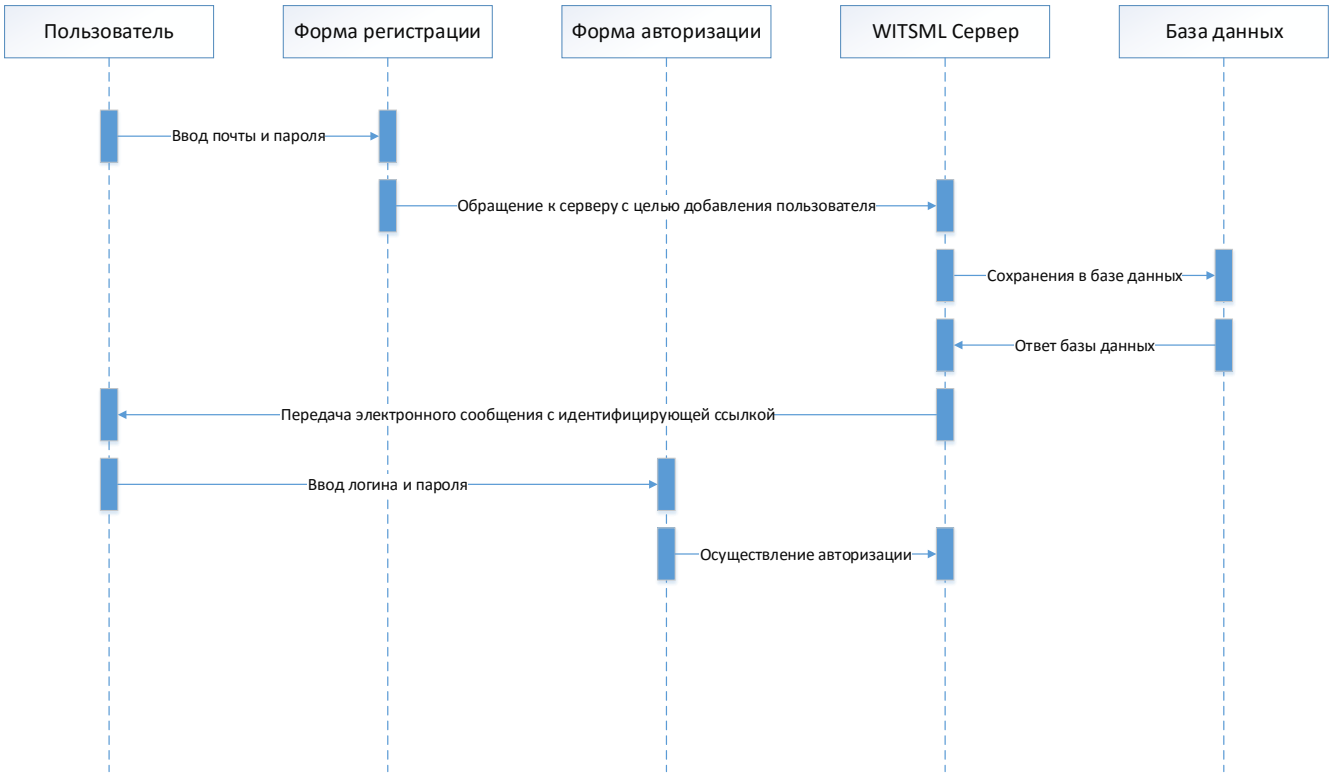

Рис. 14. Диаграмма последовательности при регистрации и авторизации

Данная схема наглядно показывает, какая информация передается по каналам данных и какой компонент системы ее принимает и обрабатывает. При авторизации пользователя, сервер делает запрос в базу данных с целью получения информации о пользователе. Если эта информация присутствует в базе, то происходит авторизация и аутентификация пользователя в соответствии с его ролью в системе. Иначе, если пользователь не был зарегистрирован, появляется сообщение об ошибке авторизации с предложением зарегистрироваться на сайте.

Пользователь имеет возможность создать тестовую скважину и при наличии агрегатора данных или WITSML агента подключиться к WITSML серверу и передать на хранение данные о скважине и бурении. Схема взаимодействия пользователя и сервера при создании новой скважины представлена на рисунке 15.

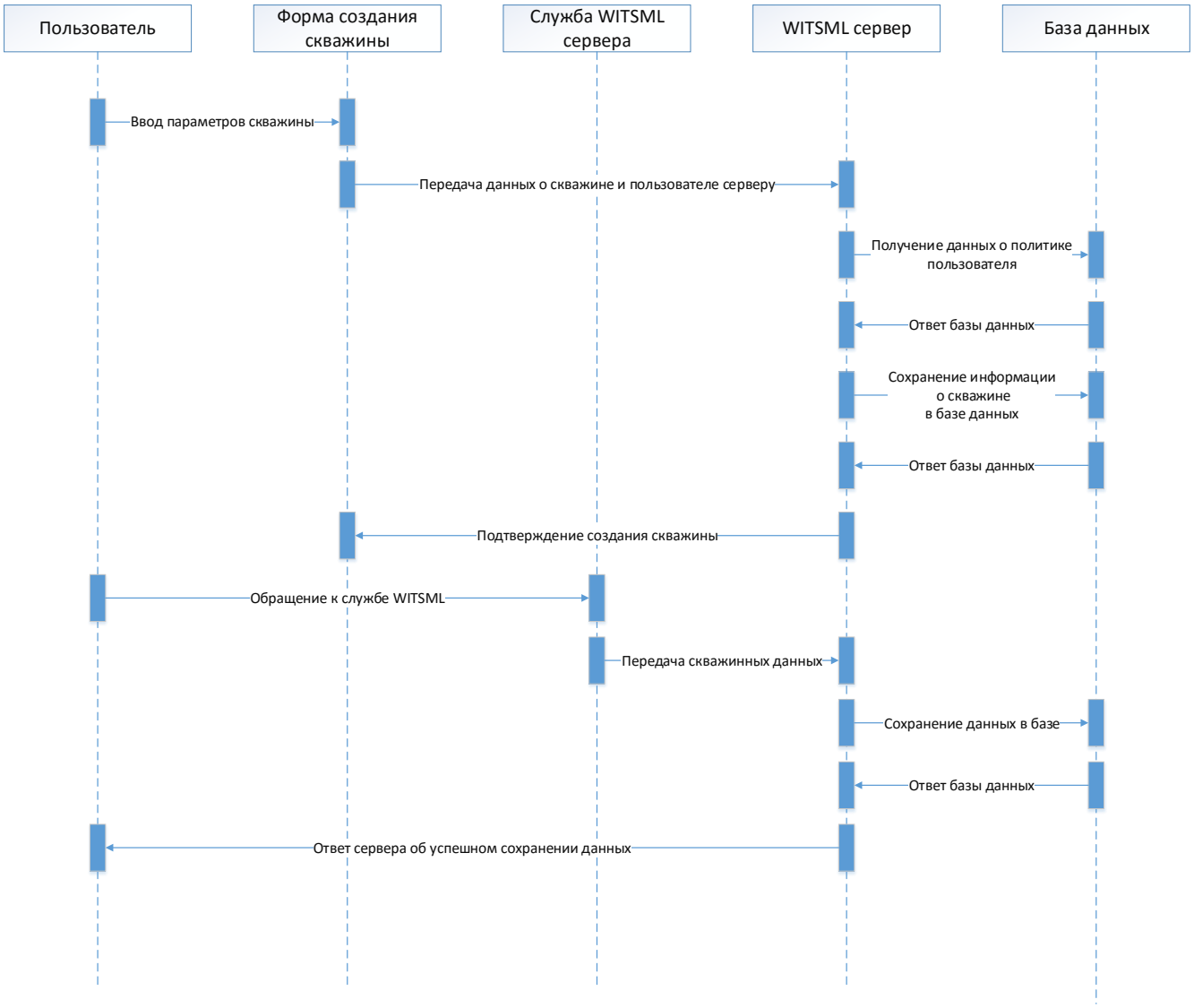

Рис. 15. Диаграмма последовательности при создании скважины и передачи даных о бурении

4.3. Проектирование базы данных для веб-приложения

Для управления объектами и получения информации о них необходима база данных, где будут храниться данные о пользователях, о скважинах, о политиках и т.д.

Схема базы данных представлена на рисунке 16 (из схемы исключены несущественные поля).

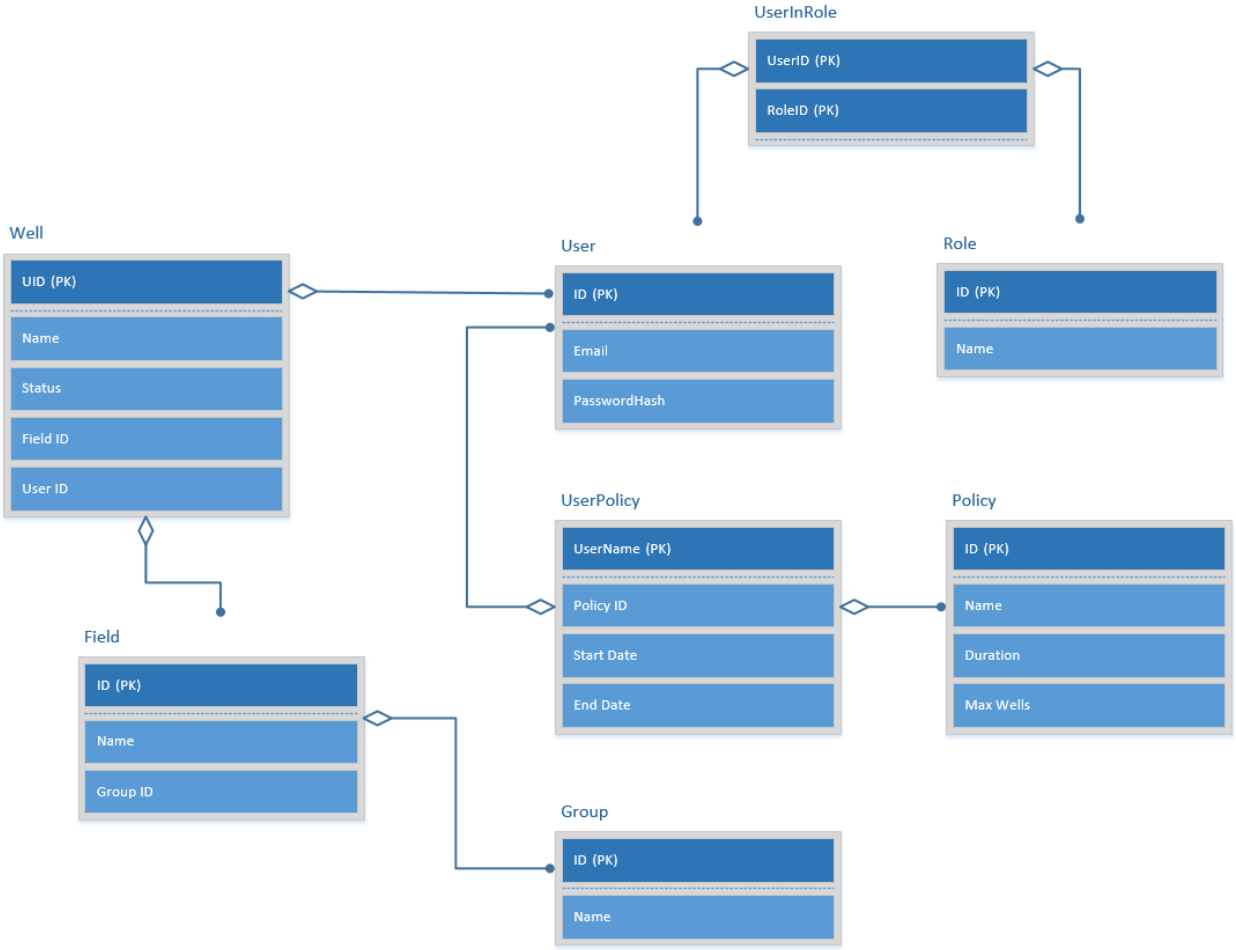

Рис. 16. Структура базы данных

Следует пояснить взаимосвязи между сущностями базы данных:

- Пользователь имеет определенную политику и роль;
- Скважина находится на определенном месторождении;
- Месторождения объединены в логические группы скважин;
- Скважина может принадлежать только одному пользователю;
- Пользователь может иметь много скважин в распоряжении.

Связь с базой данных происходит через объектно-реляционный модуль Entity Framework (рисунок 17). Для обеспечения связи с базой в проекте приложения необходимо создать объект DataBase Context, отвечающий за связь с таблицами базы.

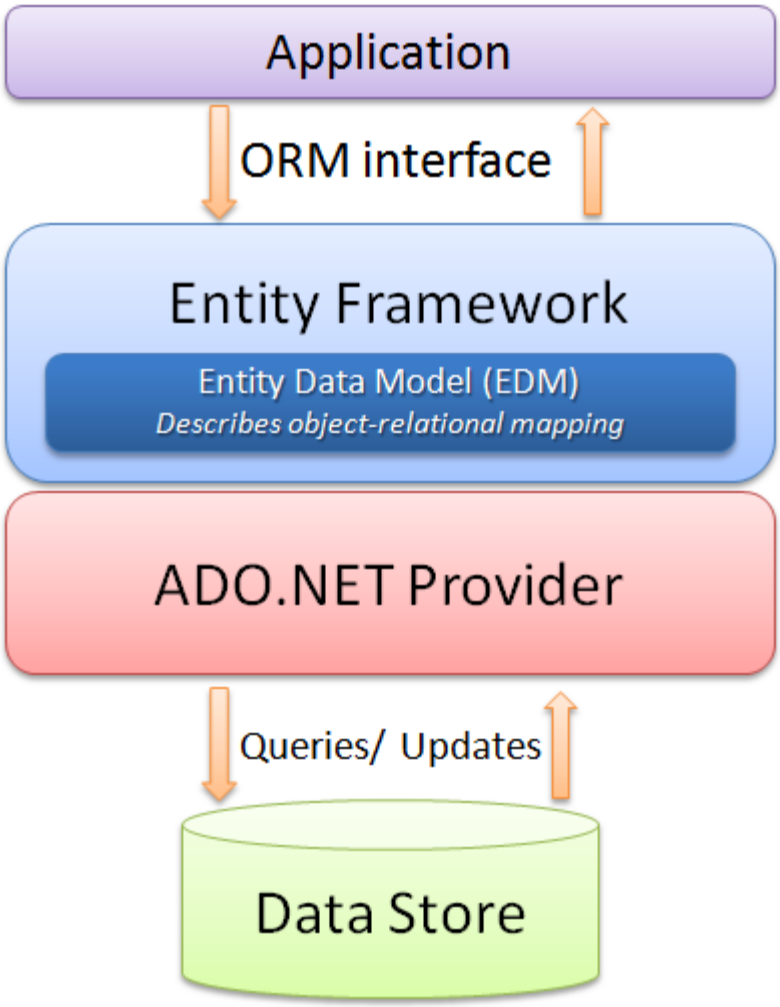

Рис. 17. Схема взаимодействия приложения и базы данных через Entity Framework [Источник: MSDN]

4.4. Дизайн веб-приложения

Дизайн страниц на сайте выдержан в одном стиле и в одной цветовой гамме. Для этого была использована готовая библиотека стилей Bootstrap и скриптов jQuery, предоставляющая готовые решения для любых дизайнерских решений в сайтостроении. В данном приложении дизайн выдержан в сине-зелено-серых оттенках, не напрягающих глаза, но дающих достаточный контраст для обозначения управляющих элементов и в целом для работы с веб-интерфейсом.

4.5. Разработка английской версии веб-приложения

Английский язык один из самых распространненых в мире. И сайт, который опубликован в сети интернет так же должен поддерживать универсальный язык, связывающий народы мира.

Для создания данной возможности необходимы 2 вещи:

1) Перключатель языка непосреднественно на сайте
2) Библиотека языковых ресурсов в приложении

Для наглядности схема использования языковых ресурсов приведена на рисунке 18.

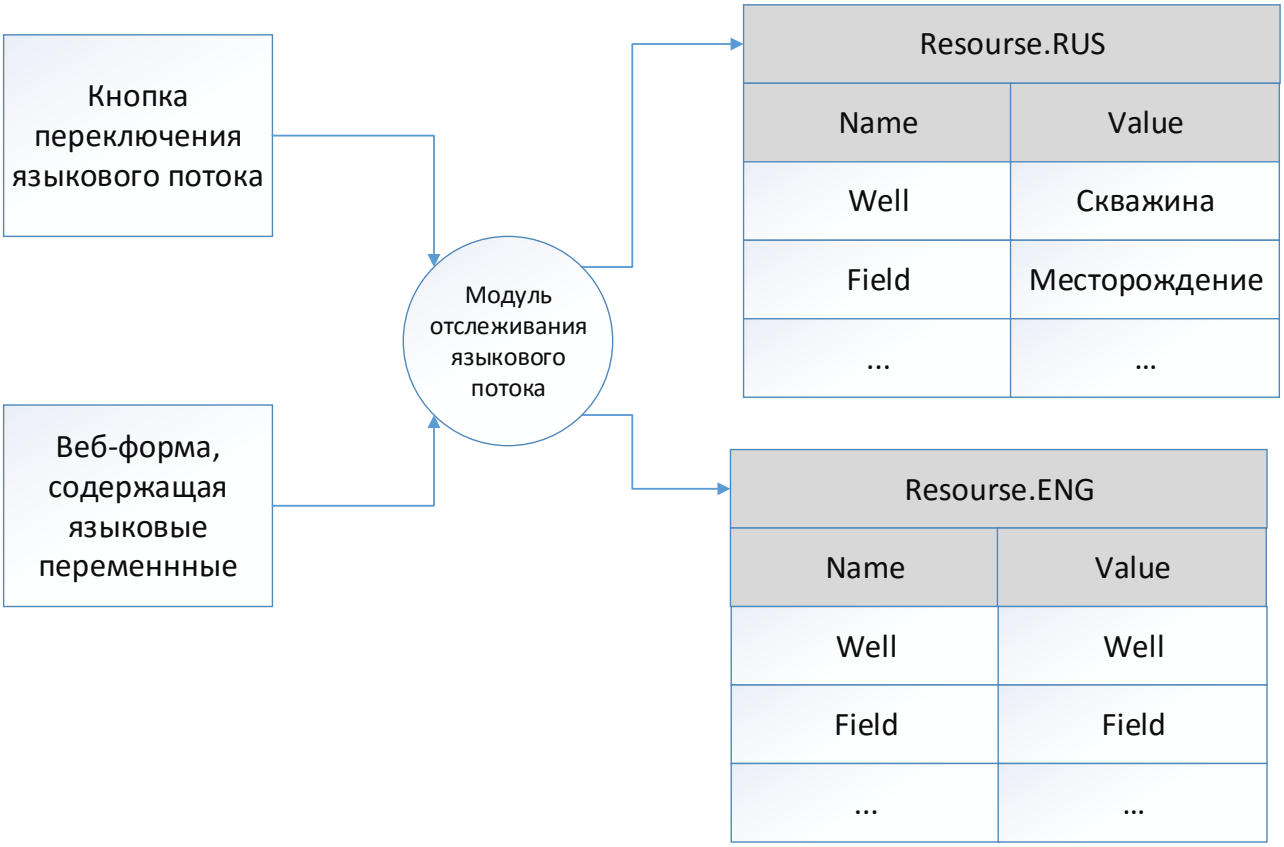

Рис. 18. Языковые ресурсы приложения

Кнопка переключения языкового потока реализована в виде RadioButton: когда необходимо иметь активным только одно значение, в данном случае – язык. Ресурс – это своеобразная таблица, имеющая два атрибута: имя параметра и его значение. Так как, английский язык – универсальный, имя атрибута применяется на английском языке, а значения – соответсвенно те, которые

необходимы для мультиязычности приложения. Пример таблицы ресурсов представлен на рисунке 19,20.

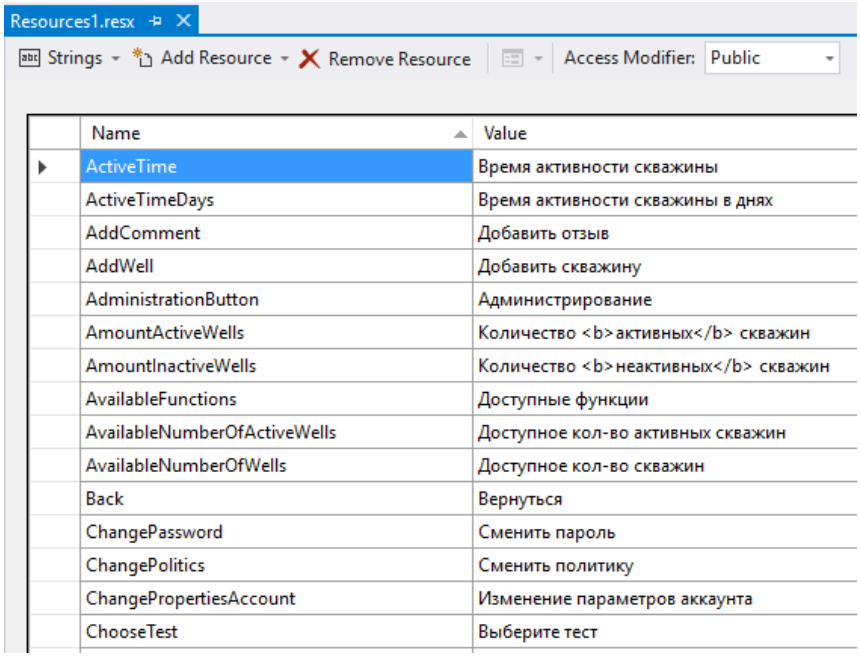

Рис. 19. Таблица ресурсов для русского языка

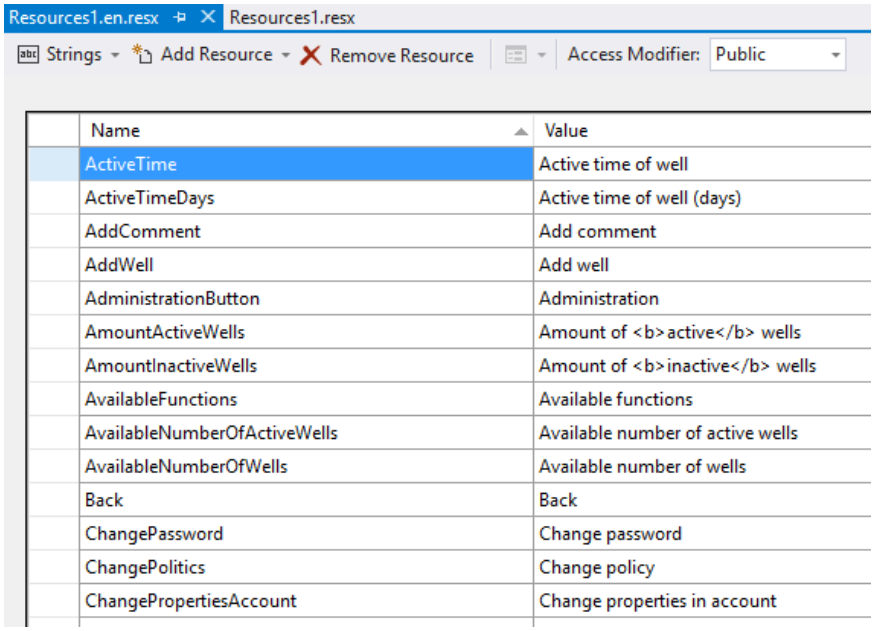

Рис. 20. Таблица ресурсов для английского языка

4.6. Разработка веб-форм приложения

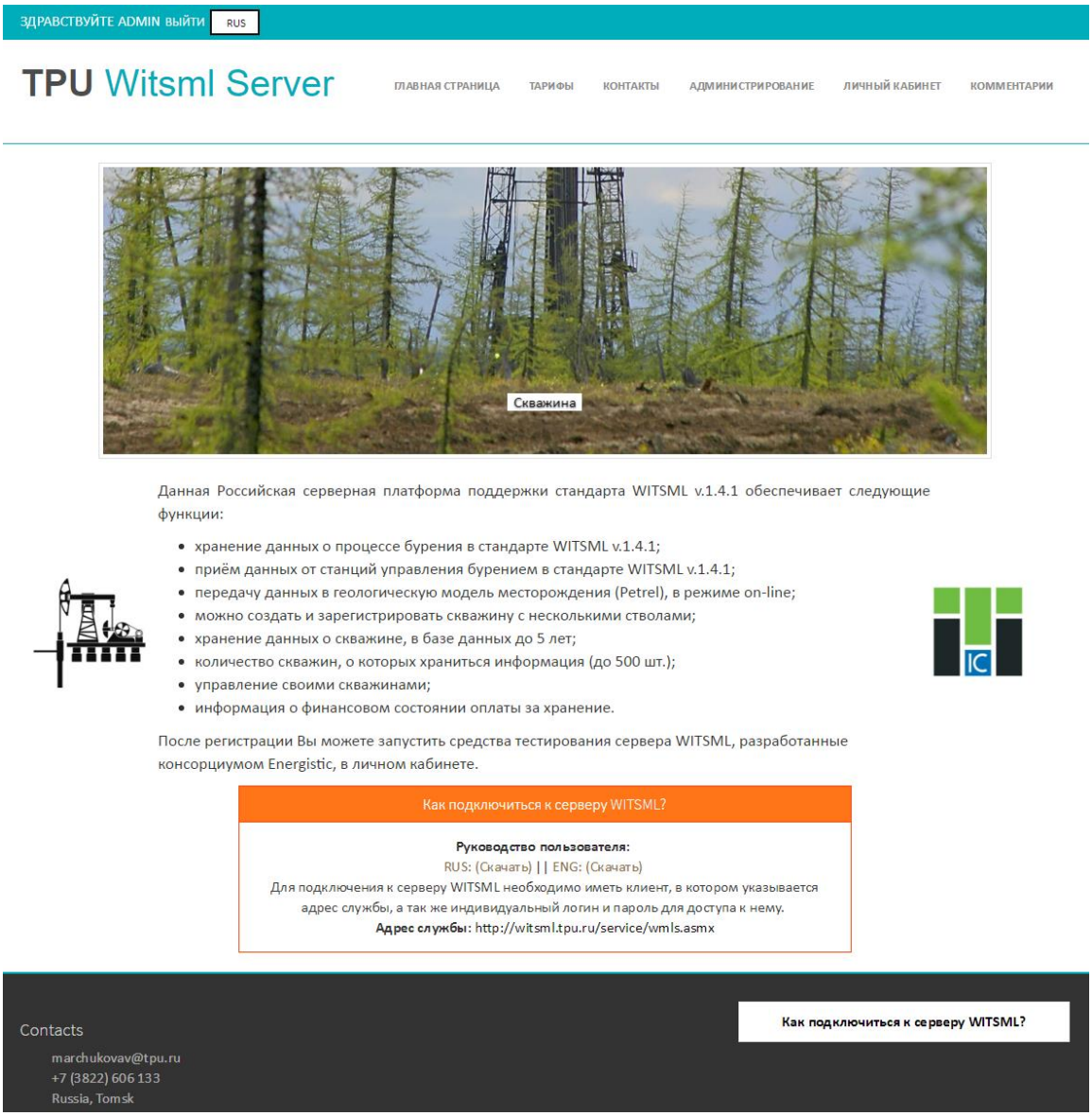

Рис. 21. Главная страница

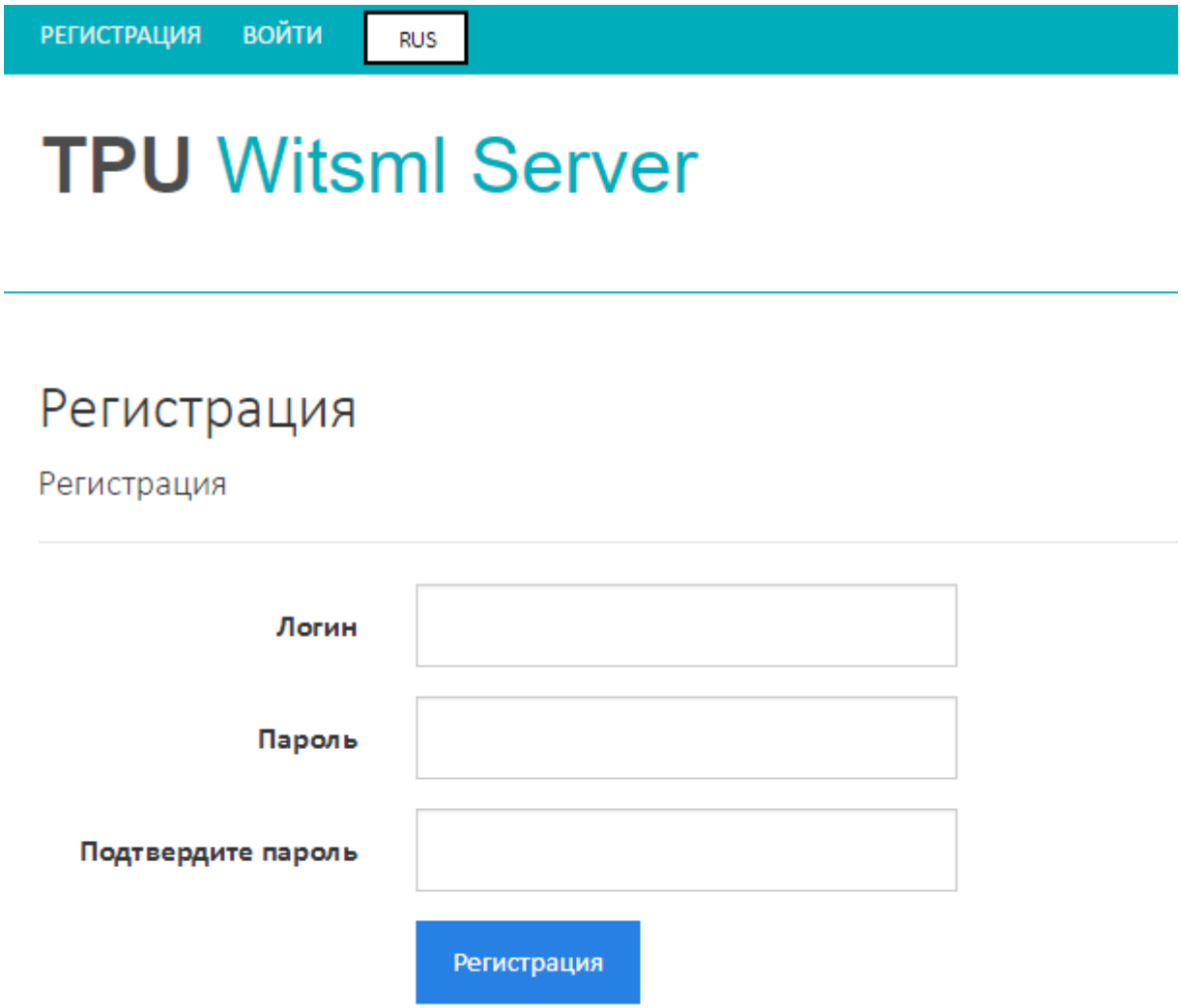

Рис. 22. Форма регистрации

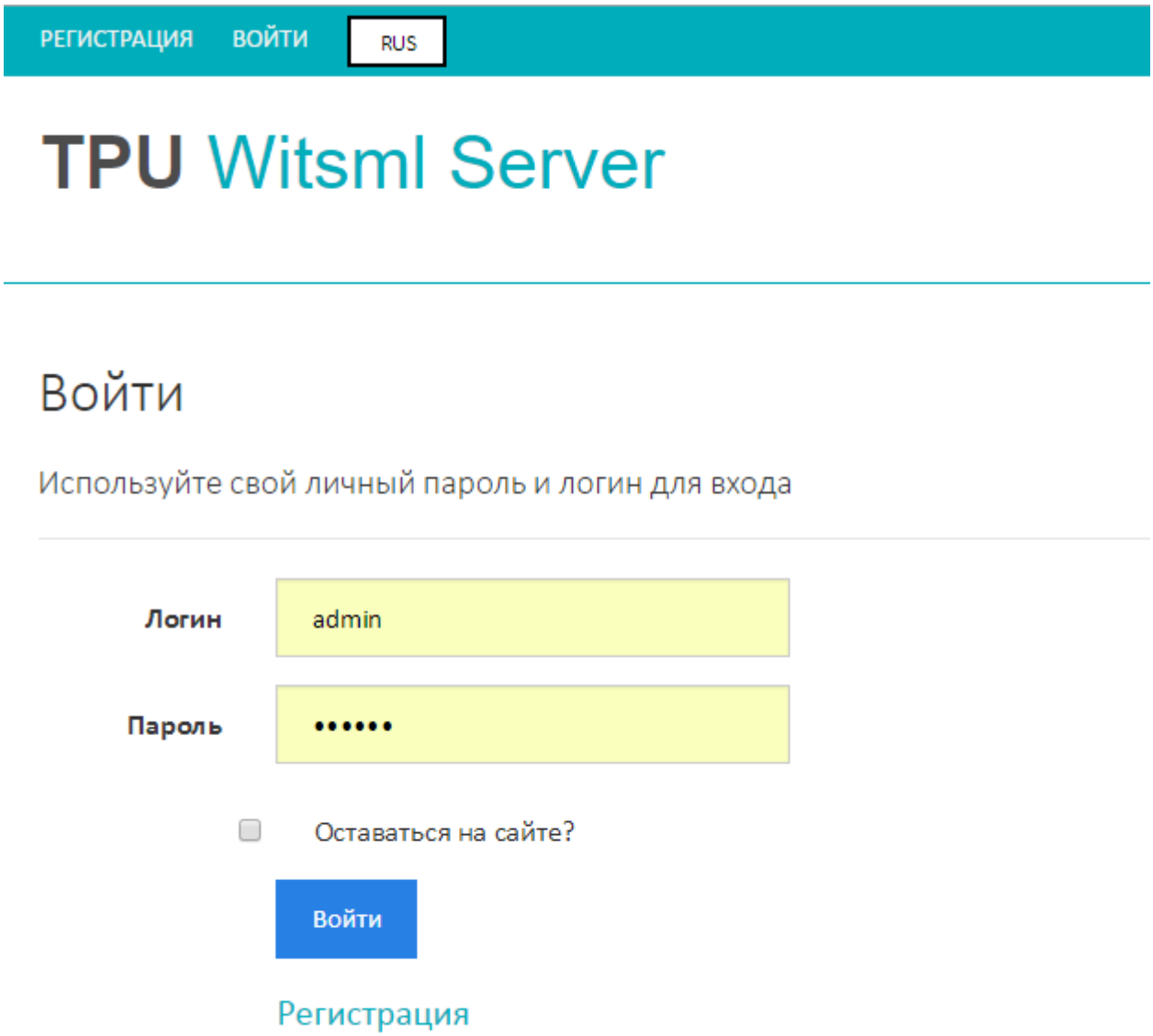

Рис. 23. Форма авторизации

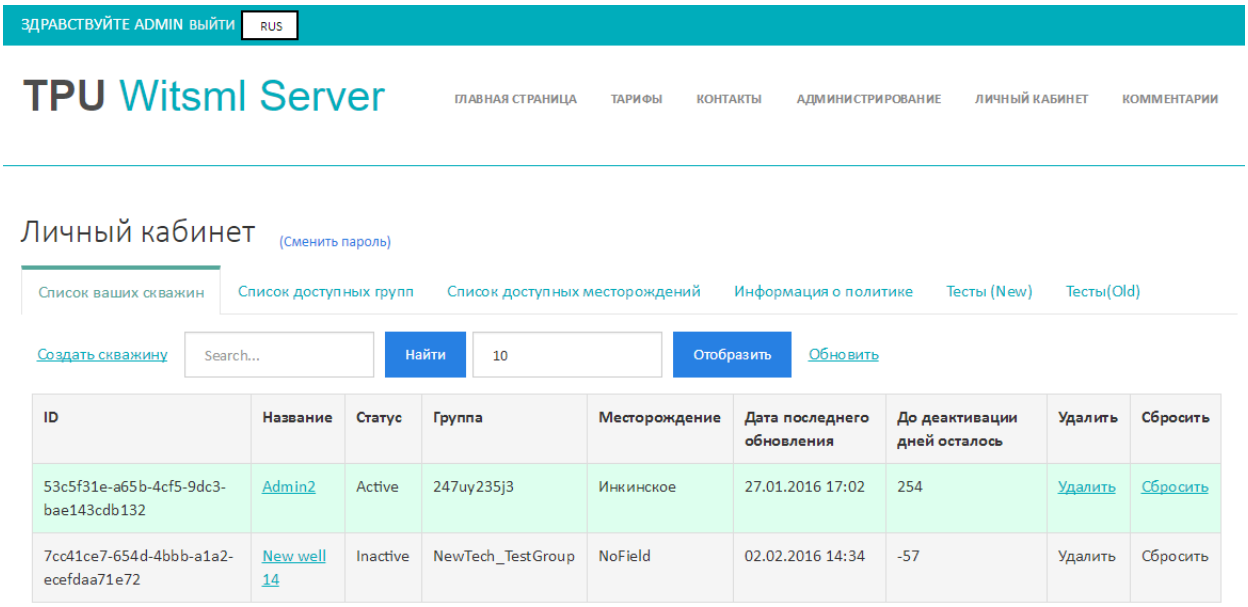

Рис. 24. Личный кабинет/Список скважин пользователя

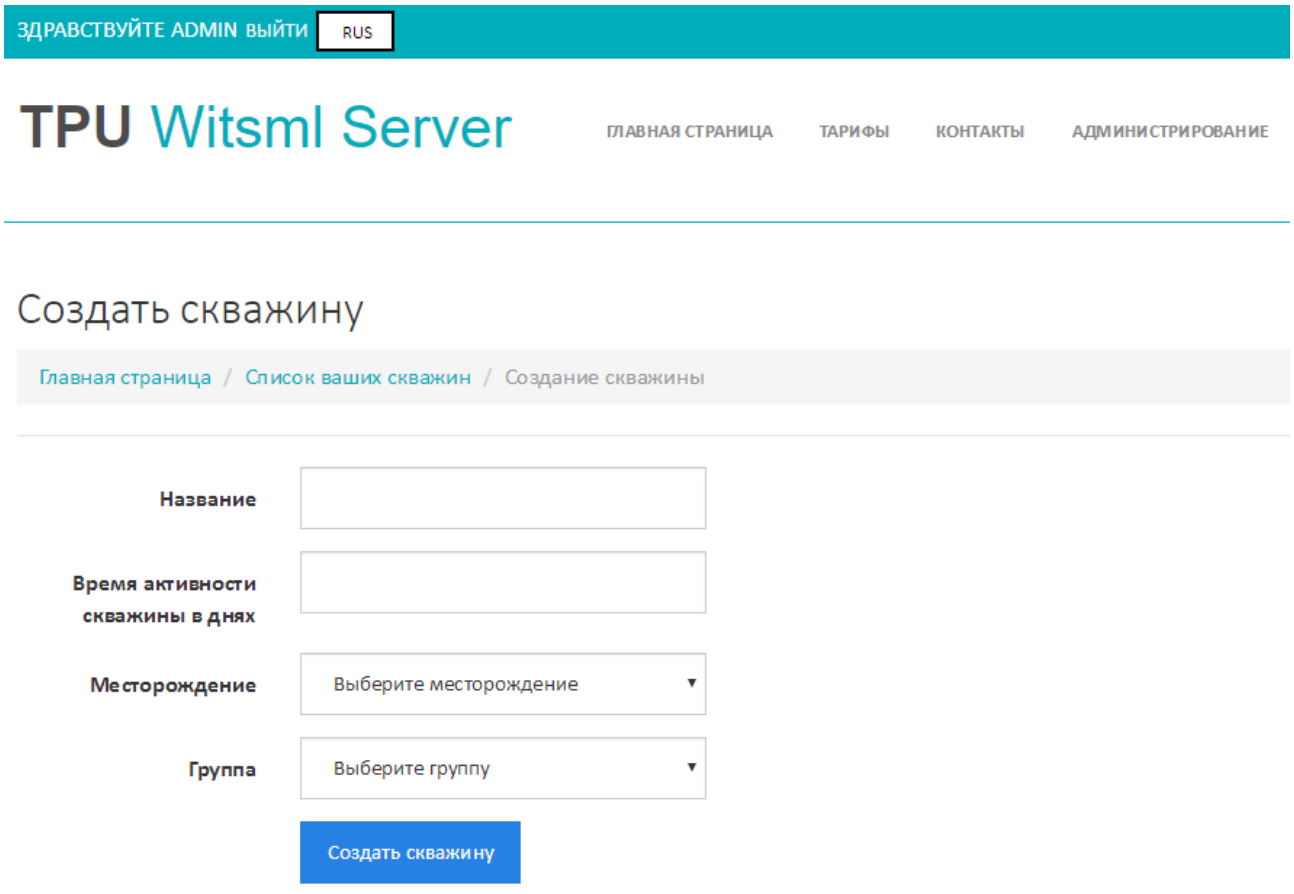

Рис. 25. Форма создания скважины

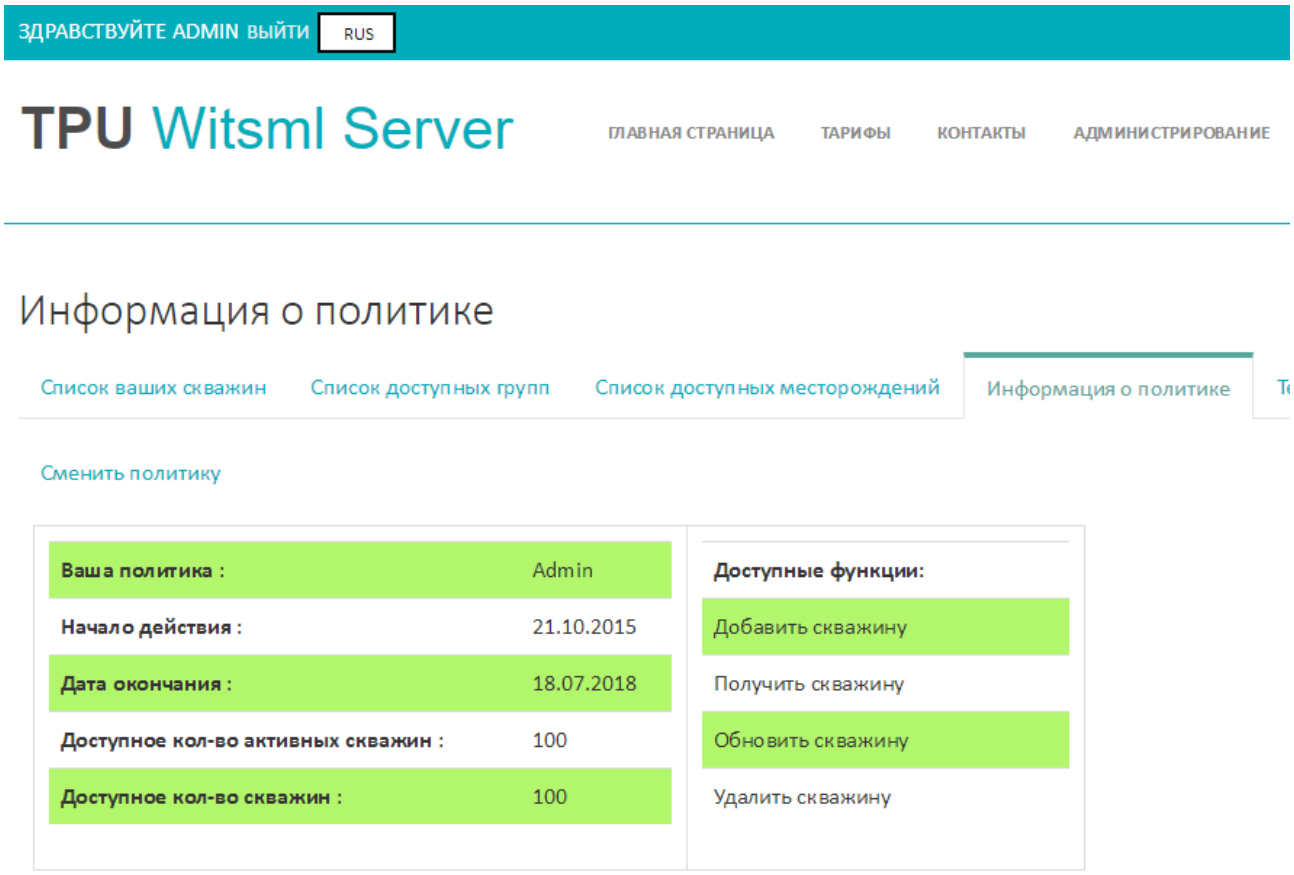

Рис. 26. Форма информации о политике

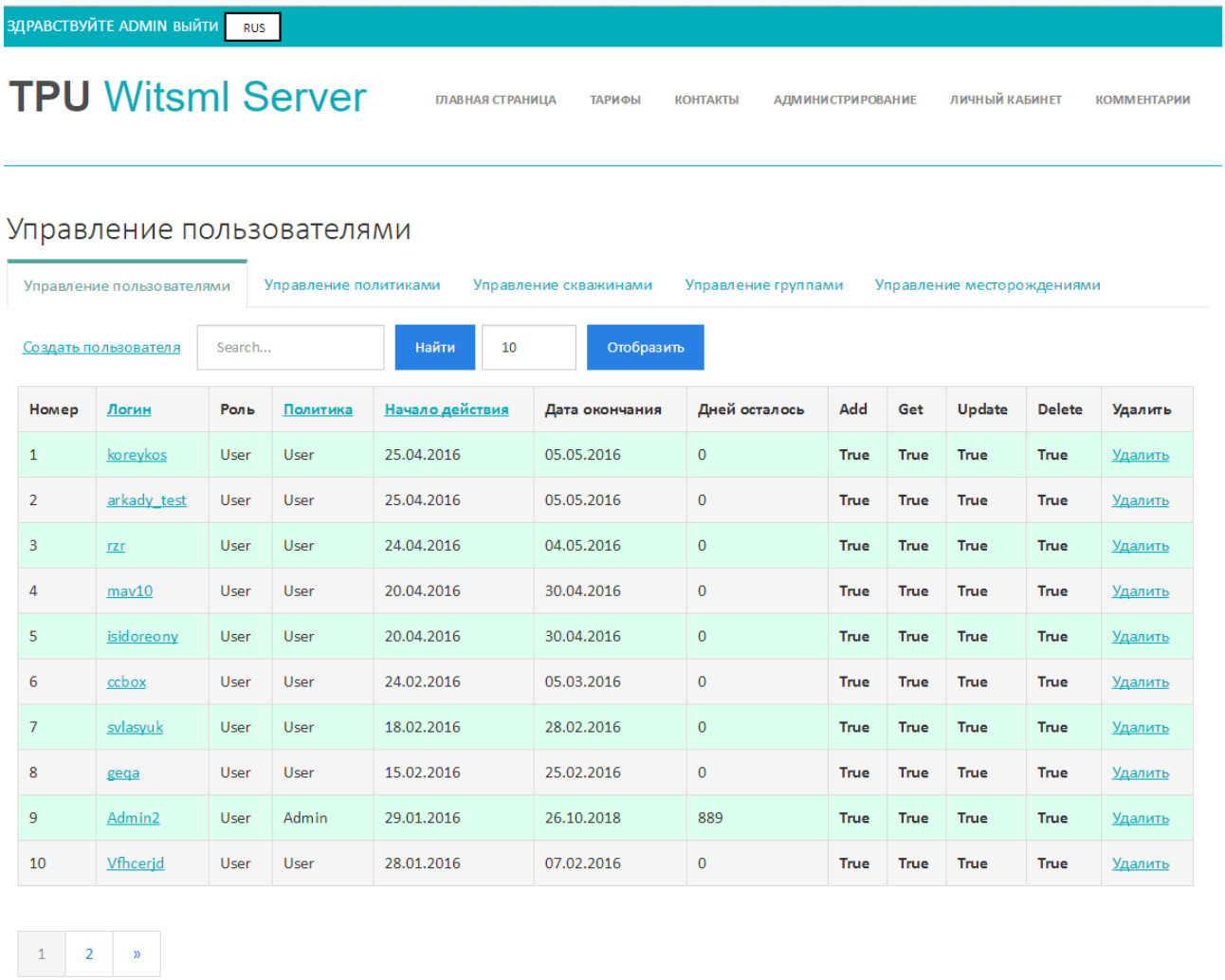

Рис. 27. Управление пользователями администратором

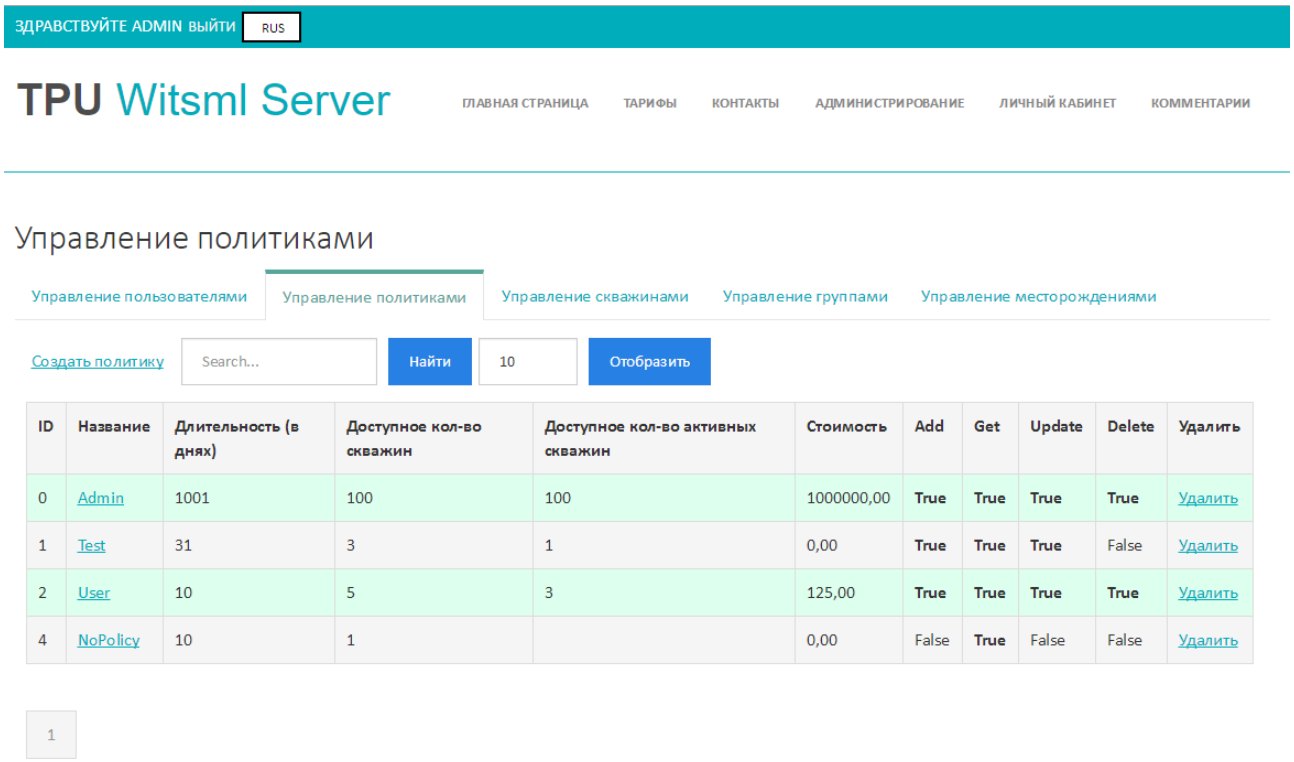

Рис. 28. Управление политиками пользователей администратором

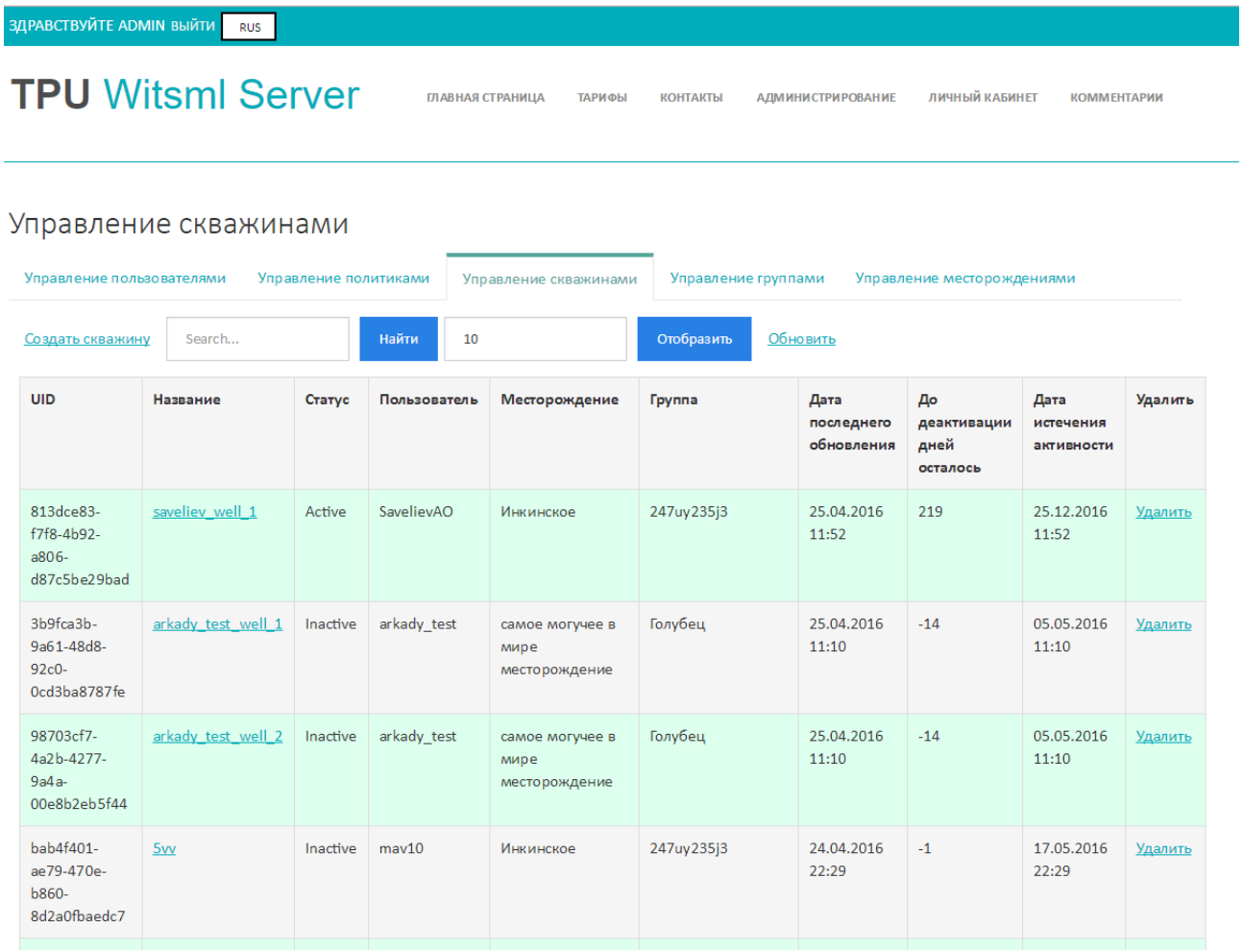

Рис. 29. Управление всеми скважинами на сервере

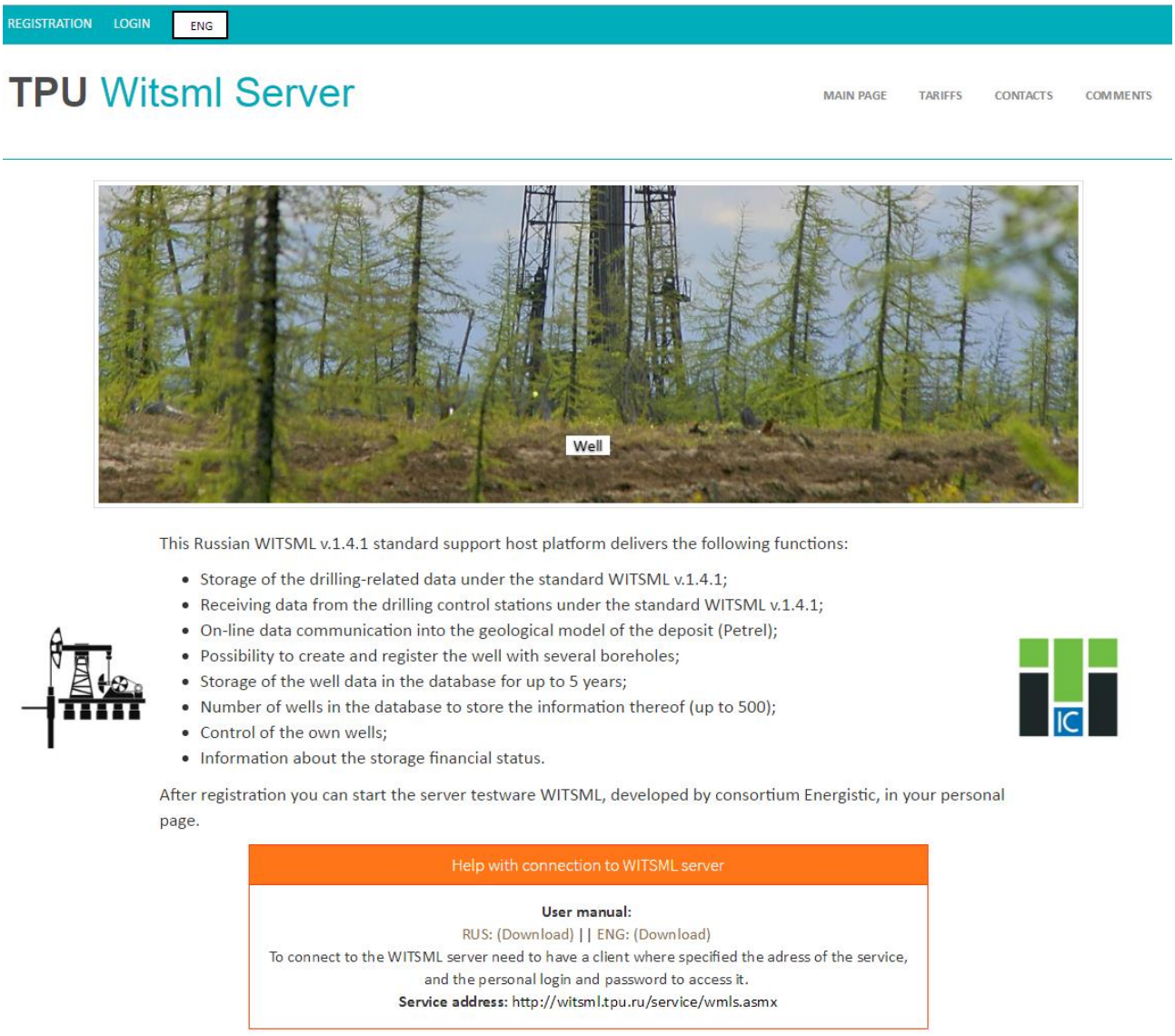

Рис. 30. Английская версия сайта

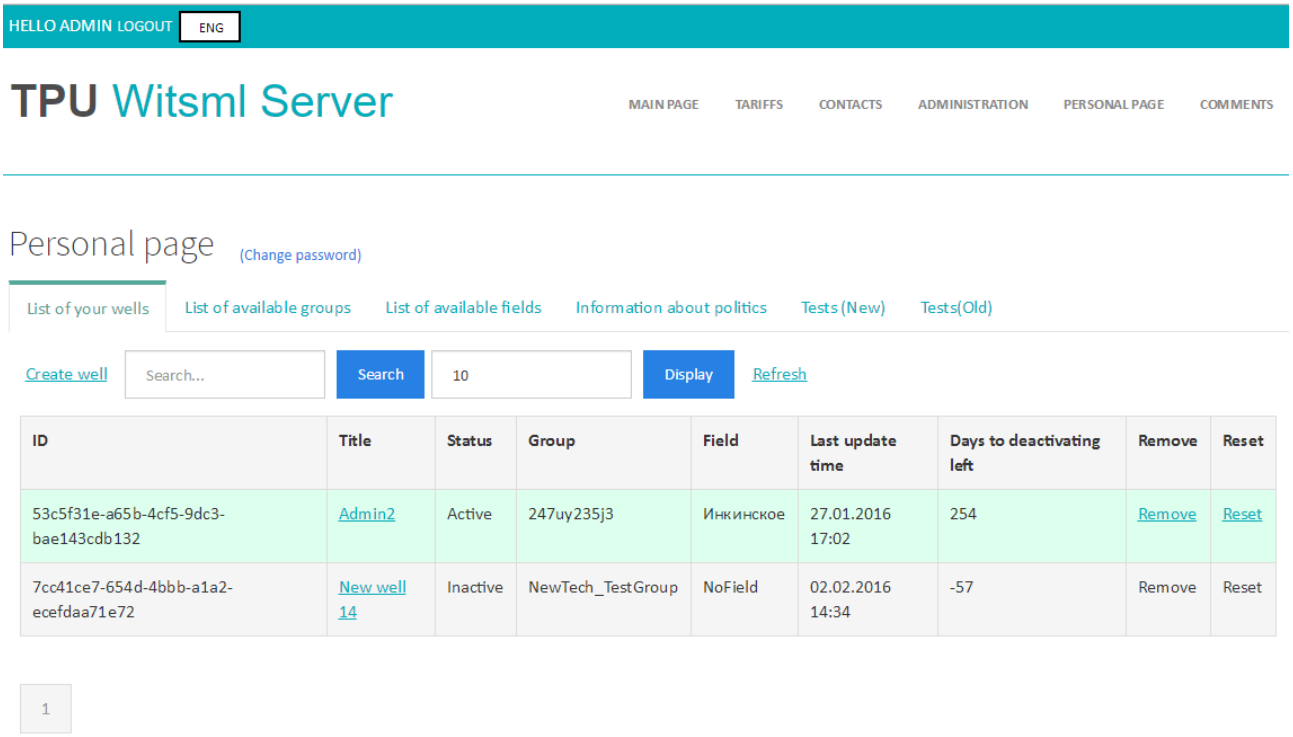

Рис. 31. Английская версия личного кабинета

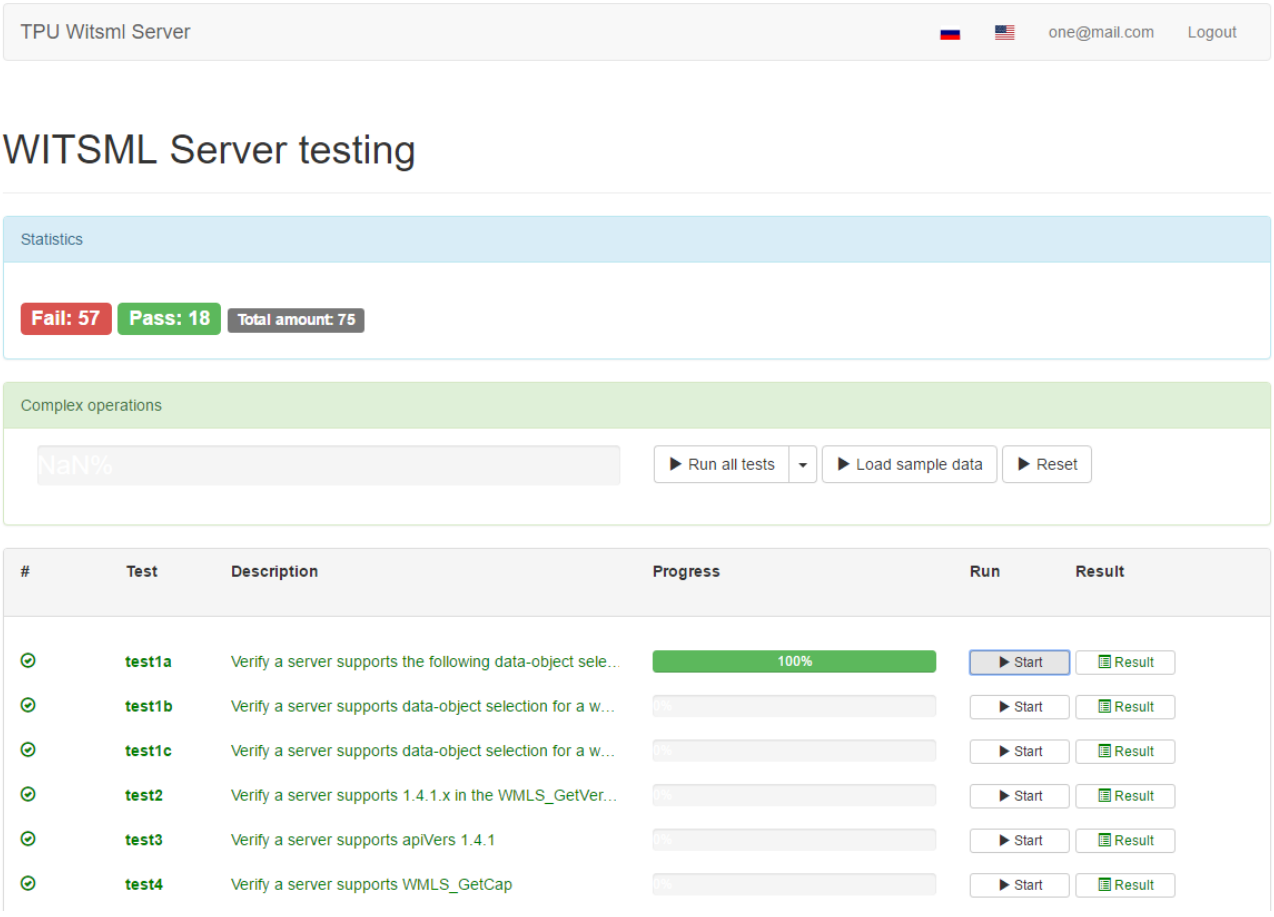

Рис. 32. Страница для тестирования сервера WITSML

Раздел 5. Финансовый менеджмент, ресурсоэффективность и ресурсосбережение.

5.1. Введение

Коммерческая привлекательность научного исследования определяется не только превышением технических параметров над предыдущими разработками, но и тем, насколько быстро разработчик сумеет найти ответы на такие вопросы – будет ли продукт востребован рынком, какова будет его цена, каков бюджет научного проекта, какой срок потребуется для выхода на рынок и т.д.

В рамках данной научной работы исследуется коммерческая составляющая разработки проекта основных функций, состава и блок-схемы национального стандарта передачи данных для нефтегазовых предприятий России, совместимого с международным стандартом WITSML-2.

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

5.2.1. Потенциальные потребители результатов исследования

Основными потребителями услуг WITSML-сервера являются компании, занимающиеся бурением скважин, разработкой месторождений, а также компании-владельцы буровых установок. Сведенья о сегментах рынка продаж услуг WITSML-сервера представлены в таблице 4.

Таблица 4. Карта сегментирования рынка продаж услуг WITSML-сервера

		Месторасположение сервера	
		Зарубежные сервера	Отечественные сервера
Месторасположение компании заказчика	Зарубежные компании		
	Отечественные компании		

Petrolink ЛУКОЙЛ

Согласно карте сегментирования рынка можно сделать вывод о полностью свободном сегменте. Целевым сегментом является продажа услуг отечественным компаниям с размещением и разработкой отечественных серверов.

5.2.2. Анализ конкурентных технических решений

Для анализа конкурирующих разработок была построена оценочная карта для сравнения конкурентных разработок. Таким образом, рассмотрена конкурентоспособность разработки относительно уже существующих компаний, предоставляющих услуги сервера WITSML: Petrolink (к1) и Petrodaq (к2).

Для определения анализа конкурентных технических решений использовалась формула:

$$К = \sum В_i \cdot Б_i, \quad (1)$$

где К – способность научной разработки или конкурента выдержать конкуренцию ;

$В_i$ – вес критерия (в долях единицы);

$Б_i$ – балл *i*-го показателя.

Таблица 5. Оценочная карта для сравнения конкурентных разработок

Показатели	**Вес критерия**	**Баллы**			**Конкуренто-способность**		
		$Б_ф$	$Б_{к1}$	$Б_{к2}$	$К_ф$	$К_{к1}$	$К_{к2}$
1	2	3	4	5	6	7	8
Технические критерии оценки ресурсоэффективности							
1. Ограниченность ресурсов памяти	0,1	2	3	1	0,2	0,3	0,1
2. Удобство в эксплуатации	0,1	4	5	4	0,4	0,5	0,4

(соответствующее требованиям потребителей)							
3. Устойчивость к помехам	0,1	4	4	3	0,4	0,4	0,3
4. Энергоэкономичность	0,05	2	2	1	0,1	0,1	0,05
5. Устойчивость	0,05	5	5	4	0,25	0,25	0,2
6. Отсутствие ошибок доступа	0,1	5	5	3	0,5	0,5	0,3
7. Удобство представления данных	0,05	5	4	4	0,25	0,2	0,2
8. Функциональная мощность (многообразие предоставляемых возможностей)	0,1	4	5	4	0,4	0,5	0,4
9. Легкость эксплуатации	0,05	5	5	4	0,25	0,25	0,2
10. Качественность интерфейса	0,05	4	5	3	0,2	0,25	0,15
Экономические критерии оценки эффективности							
1. Уровень конкурентоспособности продукта	0,1	4	5	3	0,4	0,5	0,3
2. Способность продукта проникнуть на рынок	0,05	3	5	4	0,15	0,25	0,2
3. Стоимость	0,1	5	3	4	0,5	0,3	0,4
Итого	**1**				**4**	**4,3**	**3,2**

Коэффициент конкурентоспособности предприятия:

$$k_{кс} = К_{ф} / К_{фко} = (4/4,3+4/3,2)/2 = 1,09.$$

$k_{кс} > 1$, следовательно предприятие конкурентоспособно.

5.2.3. Технология QuaD

В данном разделе для оценки конкурентоспособности исследования была построена оценочная карта по технологии QuaD.

Таблица 6. Оценочная карта для сравнения конкурентных разработок

Показатели	Вес критерия	Баллы	Макси-мальный балл	Относительное значение	Средневзвешенное значение
1	2	3	4	5	
Показатели оценки качества разработки					
1. Ограниченность ресурсов памяти	0,1	30	100	0,3	0,03

2. Удобство в эксплуатации (соответствующее требованиям потребителей)	0,1	70	100	0,7	0,07
3. Устойчивость к помехам	0,1	80	100	0,8	0,08
4. Энергоэкономичность	0,05	70	100	0,7	0,035
5. Устойчивость	0,05	85	100	0,85	0,0425
6. Отсутствие ошибок доступа	0,1	90	100	0,9	0,09
7. Удобство представления данных	0,05	90	100	0,9	0,045
8. Функциональная мощность (многообразие предоставляемых возможностей)	0,1	70	100	0,7	0,07
9. Легкость эксплуатации	0,05	80	100	0,8	0,04
10. Качественность интерфейса	0,05	75	100	0,75	0,0375
Показатели оценки коммерческого потенциала разработки					
11. Уровень конкурентоспособности продукта	0,1	70	100	0,7	0,07
12 Способность продукта проникнуть на рынок	0,05	35	100	0,35	0,0175
13. Стоимость	0,1	95	100	0,95	0,095
Итого	**1**				**0,7225**

$$П_{ср} = \sum В_i \cdot Б_i , \qquad (2)$$

где $П_{ср}$ – средневзвешенное значение оценки качества и перспектив научной разработки;

$В_i$ – вес критерия (в долях единицы);

$Б_i$ – средневзвешенное значение *i*-го показателя.

Значение $П_{ср}$ составило 72,25, что показывает перспективность разработки выше среднего.

5.2.4. SWOT-анализ

Посредством SWOT-анализа выявим внешние и внутренние факторы среды проекта, и выявим степень соответствия сильных и слабых сторон проекта внешним условиям окружающей среды.

Таблица 7. Матрица SWOT

		Сильные стороны	Слабые стороны
		С1.Низкая стоимость услуги С2.Близкое расположение серверов С3.Квалиффицированный персонал С4.Безопасность хранения данных С5.Отсутствие аналогов в РФ	Сл1.Недостаточная конкурентоспособность Сл2.Недостаточная аппаратно-техническая база Сл3.Отсутсвие опыта Сл4.Возможные сбои программного продукта Сл5.Отсутствие долговременных соглашений с клиентами

Возможности	В1.Привлечение крупных клиентов В2.Расиширение аппаратно-технической базы В3.Привлечение зарубежных клиентов В4.Получение дополнительного финансирования В5.Выход на международные рынки	В1С1С2С3С4С5 – Привлечение большего количества клиентов, увеличение доходов В3С1С3- Индивидуальный подход к клиенту В4С1С2С4С5- Участие в конкурсах финансирования проектов В5С1С3-Расширение бизнеса для международных рынков.	В2Сл4- Приобретение нового оборудования В4Сл1Сл3Сл5 – Участие в конкурсах финансирования проектов.
Угрозы	У1.Появление новых конкурентов У2.Изменение законодательства У3.Потеря платежеспособности У4.Потеря клиентов У5.Сбои аппаратно-технического обеспечения	У1С2С5- Снижение стоимости услуги У2С2С4С5- Изменение политики компании У3С1 – увеличение стоимости услуги У5С4С5 – Улучшение обслуживания оборудования	У1Сл1- Доработка программного продукта У3Сл1Сл5- Проведение рекламной политики У4Сл1Сл2Сл3Сл4Сл5- Привлечение клиентов посредством скидок У5Сл2Сл4-Расширение аппаратно-технической базы

Таблица 8. Интерактивная матрица проекта

Сильные стороны						
Возможности		С1	С2	С3	С4	С5
	В1	+	+	+	+	+
	В2	-	0	-	-	-

	В3	+	-	+	-	-
	В4	+	+	-	+	+
	В5	+	-	+	-	-

Направления реализации проекта: В1С1С2С3С4С5, В3С1С3, В4С1С2С4С5, В5С1С3.

Таблица 9. Интерактивная матрица проекта

Слабые стороны						
Возможности		Сл1	Сл2	Сл3	Сл4	Сл5
	В1	-	-	-	-	-
	В2	0	-	0	+	-
	В3	-	-	-	-	-
	В4	+	-	+	-	+
	В5	-	-	-	-	-

Направление реализации проекта: В2Сл4, В4Сл1Сл3Сл5.

Таблица 10. Интерактивная матрица проекта

Сильные стороны						
Угрозы		С1	С2	С3	С4	С5
	У1	-	+	-	-	+
	У2	-	+	-	+	+
	У3	+	0	-	-	-
	У4	-	0	-	-	-
	У5	0	-	-	+	+

Направления реализации проекта: У1С2С5, У2С2С4С5, У3С1, У5С4С5.

Таблица 11. Интерактивная матрица проекта

Слабые стороны

		Сл1	Сл2	Сл3	Сл4	Сл5
Угрозы	У1	+	-	-	-	-
	У2	-	-	-	-	-
	У3	+	-	-	-	+
	У4	+	+	+	+	+
	У5	-	+	-	+	-

Направления реализации проекта: У1Сл1, У3Сл1Сл5, У4Сл1Сл2Сл3Сл4Сл5, У5Сл2Сл4.

5.3. Нахождение альтернативных способов проведения научных исследований

Для нахождения возможных альтернативных способов проведения научных исследований воспользуемся морфологическим методом. Для этого построим морфологическую матрицу.

Таблица 12. Морфологическая матрица для веб-интерфейса WITSML-сервера

	1	**2**	**3**	**4**	**5**
А. Тип клиентского интерфейса	Веб-портал	Веб - страница	Веб-приложение	Стационарное приложение	Нет интерфейса
Б. Тип хранения данных	Сервер хранения	Система хранения	Локальные базы данных		
В. Доступ к данным	В режиме реального времени	По запросу клиента	Нет доступа		
Г. Процесс составления договора	Полностью автоматизированный	Частично автоматизированный	Полностью ручной	Отсутствие договора	
Д. Составление тарифного плана	Индивидуальное для каждого клиента	Общее для всех клиентов			
Е. Расчет стоимости услуги	Индивидуальный для каждого клиента	Строго определенная сумма для всех клиентов			
Ж. Способ	Банковски	Электронны	Личная		

оплаты услуги	м переводом	е системы оплаты	оплата в кассу		

С помощью морфологического анализа были выделены следующие преимущественные варианты:

1) А1Б2В1Г2Д1Е1Ж1 – разработка веб-портала для сервера WITSML, с использованием глобальной системы хранения и предоставлением доступа клиентам к данным скважин в режиме реального времени, а также с обеспечением, частично автоматизированного документооборота, и индивидуальными расчетами тарифных планов и стоимости услуг для каждого клиента, оплачиваемых посредством банковского перевода.

2) А2Б1В2Г1Д2Е2Ж2 – разработка веб-страницы для сервера WITSML, с использованием серверов хранения данных и предоставления доступа к данным по запросу клиента, а также с обеспечением полностью автоматизированного документооборота, общие тарифные планы и стоимости услуг для всех клиентов, и оплата посредством электронных платежных систем.

3) А3Б2В1Г1Д1Е1Ж1 – разработка веб-приложения для сервера WITSML, с использованием глобальной системы хранения и предоставлением доступа клиентам к данным скважин в режиме реального времени, а также с обеспечением, полностью автоматизированным документооборотом, и индивидуальными расчетами тарифных планов и стоимости услуг для каждого клиента, оплачиваемых посредством банковского перевода.

5.4. Создание научно-исследовательских работ

5.4.1. Этапы планирования работ в рамках научного исследования

Таблица 13 Список этапов, содержание работ и распределение ролей

Этапы	№ раб	Содержание работ	Должность исполнителя
Разработка ТЗ	1	Выбор темы	И1, И2 Руководитель

	2	Составление и утверждение ТЗ	Руководитель
Анализ предметной области	3	Создание календарного плана работ	И1, И2
	4	Выбор и изучение теоретических материалов по теме	И1, И2
	5	Анализ изученного материала	И1, И2 Руководитель
	6	Проработка литературных источников и периодической литературы	И1, И2
Основная	7	Анализ современного состояния стандартизации данных в нефтеной и газовой промышленности	И2
	8	Разработка состава номенклатуры национального стандарта передачи данных	И1, И2
	9	Проектирование основных функций и блок схемы национального стандарта передачи данных	И1
	10	Проектирование и разработка средства мониторинга и предоставления продукта конечному пользователю сервера национального стандарта	И2
	11	Разработка системы формирования финансовой документации	И1
	12	Тестирование и доработка системы	И1, И2
Заключительная	13	Написание пояснительной записки	И1, И2
	14	Создание и оформление презентации дипломного проекта	И1, И2

Для выполнения научного исследования формируется рабочая группа, в состав которой входят 2 студента-дипломника и один руководитель. Порядок этапов и работ, распределение исполнителей по данным видам работ приведен в таблице.

5.4.2. Расчет трудоемкости исполнения работ

Чтобы определить ожидаемое значение продолжительности работ $t_{ож}$ применяются две оценки: t_{min} и t_{max} (метод двух оценок).

$$t_{ожс} = \frac{3 \cdot t_{min} + 2 \cdot t_{max}}{5}, \qquad (3)$$

где t_{min} – наименьшая трудоемкость работ, чел/дн.;

t_{max} – наибольшая трудоемкость работ, чел/дн.

После нахождения значения ожидаемой трудоемкости работ, необходимо определить длительность каждой работы в рабочих днях $T_р$, учитывая при этом возможность параллельности работ сразу несколькими исполнителями.

$$T_{рi} = \frac{t_{ожi}}{Ч_i}, \qquad (4)$$

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

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

$Ч_i$ – количество исполнителей, которые выполняют одну и ту же работу параллельно, чел.

Для удобства построения графика, длительность каждого из этапов работ из рабочих дней следует перевести в календарные дни.

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

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

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

$k_{кал}$ – коэффициент календарности.

Коэффициент календарности определяется по следующей формуле:

$$k_{кал} = \frac{T_{кал}}{T_{кал} - T_{вых} - T_{пр}}, \qquad (6)$$

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

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

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

Для выполнения перечисленных в таблице 10 работ требуются специалисты: студент, научный руководитель. Результаты расчетов представлены в таблице 14.

Таблица 14. Временные показатели проведения научного исследования

№ Работы	Трудоёмкость работ									Исполнители			Длительность работ в рабочих днях T_{pi}			Длительность работ в календарных днях T_{ki}		
	t_{min}, чел-дни			t_{max}, чел-дни			$t_{ожi}$, чел-дни											
	Исп.1	Исп.2	Исп.3	Исп.1	Исп.2	Исп.3	Исп.1	Исп.2	Исп.3	Исп.1	Исп.2	Исп.3	Исп.1	Исп.2	Исп.3	Исп.1	Исп.2	Исп.3
Выбор темы	2	2	2	3	3	3	2,4	2,4	2,4	3	3	3	0,6	0,6	0,6	1	1	1
Составление и утверждение ТЗ	4	6	5	7	9	8	5,2	7,2	6,2	1	1	1	5,2	7,2	6,2	8	11	9
Создание календарного плана работ	1	1	1	2	2	2	1,4	1,4	1,4	2	2	2	0,3	0,35	0,35	1	1	1
Выбор и изучение теоретических материалов по теме	10	9	11	14	13	15	11,6	10,6	12,6	2	2	2	2,9	2,65	3,15	4	4	5
Анализ изученного материала	7	8	10	10	11	13	8,2	9,2	11,2	3	3	3	1,6	1,84	2,24	2	3	3
Проработка литературных источников и периодической литературы	3	3	3	6	6	6	4,2	4,2	4,2	2	2	2	1,0	1,05	1,05	2	2	2
Анализ современного состояния стандартизации данных	7	10	8	14	18	13	9,8	13,2	10	1	1	1	2,4	3,3	2,5	4	5	4

в нефтеной и газовой промышленности																		
Разработка состава номенклатуры национального стандарта передачи данных	20	23	26	30	36	31	24	28,2	28	2	2	2	6	7,05	7	9	10	10
Проектирование основных функций и блок схемы национального стандарта передачи данных	31	31	31	50	50	50	38,6	38,6	38,6	1	1	1	38,6	38,6	38,6	57	57	57
Проектирование и разработка средства мониторинга и предоставления продукта конечному пользователю сервера национального стандарта	31	31	31	50	50	50	38,6	38,6	38,6	1	1	1	38,6	38,6	38,6	57	57	57
Разработка системы формирования финансовой документации	31	31	31	50	50	50	38,6	38,6	38,6	1	1	1	38,6	38,6	38,6	57	57	57
Тестирование и доработка системы	31	31	31	50	50	50	38,6	38,6	38,6	2	2	2	38,6	38,6	38,6	57	57	57
Написание пояснительной записки	8	11	16	14	20	24	10,4	14,6	19,2	2	2	2	2,6	3,65	4,8	4	5	7

Создание и оформление презентации научного исследования	7	10	11	10	14	18	8,2	11,6	13,8	2	2	2	2,05	2,9	3,45	3	4	5
Итого	197	211	221	317	339	340	245,2	262,2	269	-	-	-	180,3	186,3	187	268	276	277

Ккал = 365/ (365-118) = 1, 48

По данным расчетам, программа будет разработана:

- в первом исполнении 268 дней
- во втором исполнении 276 дней
- в третьем исполнении 277 дней

Следовательно, можно сделать вывод, что в первом исполнении работы будет выполнена быстрее.

На основе таблицы 11 построен календарный план-график, представленный в таблице 12. График строился для максимального по длительности исполнения работ в рамках научно-исследовательского проекта с разбивкой по месяцам и декадам (10 дней) за период времени дипломирования.

5.4.3. Бюджет научно-технического исследования

При распределении бюджета НТИ необходимо обеспечить полное и достоверное отражение всех видов расходов, связанных с его выполнением. В процессе формирования бюджета НТИ используется следующая группировка затрат по статьям:

- материальные затраты НТИ;
- расчет амортизации;
- основная заработная плата исполнителей темы;
- дополнительная заработная плата исполнителей темы;
- отчисления во внебюджетные фонды (страховые отчисления);
- накладные расходы.

5.4.3.1. Расчет материальных затрат НТИ

Расчет материальных затрат осуществляется по формуле 6:

$$З_{м} = (1 + k_{Т}) \cdot \sum_{i=1}^{m} Ц_{i} \cdot N_{расх i}\ , \qquad (7)$$

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

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

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

k_T – коэффициент, учитывающий транспортно-заготовительные расходы.

Для разработки данного продукта необходимы следующие материальные ресурсы:

- Системный блок
- Принтер
- Монитор
- Система хранения данных
- Два сервера

Расчет материальных затрат представлен в таблице 15.

Таблица 15. Материальные затраты

Наименование	**Единица измерения**	**Количество**			**Цена за ед., руб.**			**Затраты на материалы, ($З_м$), руб.**		
		Исп.1	Исп.2	Исп.3	Исп.1	Исп.2	Исп.3	Исп.1	Исп.2	Исп.3
Системный блок	Шт.	4	4	4	34000	35000	34900	136000	140000	139600
Принтер	Шт.	1	1	1	8500	8000	7000	8500	8000	7000
Монитор	Шт.	6	4	5	7600	8000	7800	45600	32000	39000
Система хранения данных DEPO Storage 3436	Шт.	1	1	1	400522	400522	400522	400522	400522	400522
Сервер Depo Storm 4355T2	Шт.	2	1	0	459591	459591	0	919182	459591	0
Сервер Depo Storm 2350V2	Шт.	0	1	3	0	109810	109810	0	109810	329430
Итого								**1509804**	**1149923**	**915552**

5.4.3.2. Расчет амортизационных отчислений

Таблица 16. Величина амортизационных отчислений

Наименование	Количество			Спервон., руб.			Т п.и.	На, %	А в мес., руб.			А за период, руб.		
	Исп. 1	Исп. 2	Исп. 3	Исп. 1	Исп. 2	Исп. 3	Исп. 1,2,3	Исп. 1,2,3	Исп. 1	Исп. 2	Исп. 3	Исп. 1	Исп. 2	Исп. 3
Монитор	6	4	5	45600	-	-	730	0,137	62,5	-	-	562,5	-	-
Системный блок	4			136000	140000	139600			186,3	191,8	191,3	1676,7	1726,2	1721,7
Система хранения данных DEPO Storage 3436	1			400522			1460	0,068	272,4			2451,6		
Сервер Depo Storm 4355T2	2	1	-	919182	459591	-			625	312,5	-	5625	2812,5	-
Сервер Depo Storm 2350V2	-	1	3	-	109810	329430			-	74,7	224	-	672,3	2016
											Всего	**10315,8**	**7662,6**	**6189,3**

5.4.3.3. Расчет основной заработной платы исполнителей темы

Для расчета основной заработной платы используется формула 8:

$$C_{осн/зп} = \sum_{i=1}^{n} t_i \cdot C_{зп_i} \qquad (8)$$

где n – количество работ;

t_i – затраты труда на выполнение i-го вида работ, в днях;

$C_{зп_i}$ – средняя заработная плата работника, который выполняет i-ый вид работы, руб/день.

Средняя заработная плата в день рассчитывается по формуле 9:

$$C_{зп_i} = \frac{D \cdot K \cdot M_p}{F_o} \qquad (9)$$

где D – должностной оклад работника в месяц;

K – коэффициент, в который входят коэффициент по премиям и районный коэффициент (K=1,3);

M_p – количество месяцев работы без отпуска в течение года;

F_0 – действительный годовой фонд рабочего времени работника, в днях.

При отпуске 28 дня M_p=11,08.

Результаты расчета действительного годового фонда проведены в таблице 15.

Таблица 17. Баланс рабочего времени

Показатели рабочего времени	**Руководитель**	**Студент1**	**Студент 2**
Календарное число дней	365	365	365
Количество нерабочих дней - выходные дни - праздничные дни	118	118	118
Потери рабочего времени - отпуск - невыходы по болезни	28	228	228
Действительный годовой фонд рабочего времени	219	219	219

Расчет затрат на основную заработную плату приведен в таблице 18. При этом затраты на оплату труда первого студента-дипломника определяются как оклад студента 1 (D = 6976,22), второго – как оклад студента 2 с окладом в размере 6976,22 рублей. А оклад руководителя проекта составляет 23264,86. Коэффициент К, учитывающий коэффициент по премиям и районный коэффициент равен 1,3, а $M_р$ равно 11,08.

Таблица 18. Затраты на основную заработную плату

Исполнители	**Среднедневная заработная плата $С_{зп}$ (руб.)**			**Трудоемкость (t_i), чел-дни**			**Затраты на основную зарплату (руб.)**		
	Исп. 1	**Исп. 2**	**Исп. 3**	**Исп. 1**	**Исп. 2**	**Исп. 3**	**Исп. 1**	**Исп. 2**	**Исп. 3**
Руководитель	1530,2			7	9	8	10711,4	13771,8	12241,6
Студент 1	458,8			43	44	45	19728,4	20187,2	20646
Студент 2	458,8			43	44	45	19728,4	20187,2	20646
Итого							39456,8	54146,2	53533,6

5.4.3.4. Дополнительная заработная плата

Дополнительная заработная плата включает заработную плату за не отработанное рабочее время, но гарантированную действующим законодательством.

Расчет дополнительной заработной платы ведется по формуле 10:

$$З_{доп} = k_{доп} \cdot З_{осн} \quad (10)$$

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

$k_{доп}$ равен 0,12. Результаты по расчетам дополнительной заработной платы сведены в таблицу 19.

Таблица 19. Затраты на дополнительную заработную плату

Исполнители	**Основная зарплата(руб.)**			**Коэффициент дополнительной заработной платы ($k_{доп}$)**	**Дополнительная зарплата(руб.)**		
	Исп.1	**Исп.2**	**Исп.3**		**Исп.1**	**Исп.2**	**Исп.3**
Руководитель	10711,4	13771,8	12241,6	0,12	1285,4	1652,6	1469
Студент 1	19728,4	20187,2	20646	0,12	2367,4	2422,5	2477,5
Студент 2	19728,4	20187,2	20646	0,12	2367,4	2422,5	2477,5
Итого					4734,8	6497,6	6424

5.4.3.5. Отчисления во внебюджетные фонды (страховые отчисления)

Величина отчислений во внебюджетные фонды определяется исходя из формулы 11:

$$З_{внеб} = k_{внеб} \cdot (З_{осн} + З_{доп}), \quad (11)$$

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

На 2014 г. В соответствии с Федерального закона от 24.07.2009 №212-ФЗ установлен размер страховых взносов равный 30%. На основании пункта 1

ст.58 закона №212-ФЗ для учреждений осуществляющих образовательную и научную деятельность в 2014 году водится пониженная ставка – 27,1%.

Отчисления во внебюджетные фонды представлены в таблице 20.

Таблица 20 Отчисления во внебюджетные фонды

Исполнитель	**Основная заработная плата, руб.**			**Дополнительная заработная плата, руб.**		
	Исп .1	**Исп .2**	**Исп .3**	**Исп .1**	**Исп .2**	**Исп .3**
Руководитель проекта	10711,4	13771,8	12241,6	1285,4	1652,6	1469
Студент 1	19728,4	20187,2	20646	2367,4	2422,5	2477,5
Студент 2	19728,4	20187,2	20646	2367,4	2422,5	2477,5
Коэффициент отчислений во внебюджетные фонды	27,1%					
Итого						
Исполнение 1	**15227,06**					
Исполнение 2	**16434,47**					
Исполнение 3	**16248,51**					

5.4.3.6. Формирование бюджета затрат научно-исследовательского проекта

Рассчитанная величина затрат научно-исследовательской работы (темы) является основой для формирования бюджета затрат проекта, который при формировании договора с заказчиком защищается научной организацией в качестве нижнего предела затрат на разработку научно-технической продукции. Определение бюджета затрат на научно-исследовательский проект по каждому варианту исполнения приведен в таблице 21.

Таблица 21. Расчет бюджета затрат НТИ

Наименование статьи	**Сумма, руб.**		
	Исп.1	**Исп.2**	**Исп.3**
1. Материальные затраты НТИ	1509804	1149923	915552

2. Амортизационные отчисления	10315,8	7662,6	6189,3
3. Затраты по основной заработной плате исполнителей темы	39456,8	54146,2	53533,6
4. Затраты по дополнительной заработной плате исполнителей темы	4734,8	6497,6	6424
5. Отчисления во внебюджетные фонды	15227,06	16434,47	16248,51
6. Бюджет затрат НТИ	**1579368**	**1233286**	**996755**

Вывод: Основываясь на данных, полученных в пунктах 3.4.1 – 3.4.6, был рассчитан бюджет затрат научно-исследовательской работы для трех исполнителей. Наиболее низким по себестоимости оказался проект третьего исполнителя, затраты на его полную реализацию составляют **996755** рублей.

Раздел 6. Социальная ответственность

6.1. Введение

При работе над проектной и научно-исследовательской деятельностью значительную роль играет обеспечение безопасности охраны труда и окружающей среды. Понятие «социальная ответственность» включает в себя: охрану здоровья работников, обеспечение безопасных условий работы, защита от профессиональных заболеваний и производственного травматизма.

Выше перечисленные аспекты регламентируются в соответствии с международным стандартом ICCSR26000:2011 «Социальная ответственность организации». Целью данного стандарта является принятие проектных решений, исключающих несчастные случаи на производстве и снижение негативных воздействий на окружающую среду. На основе стандарта написана данная часть работы.

Раздел, посвященный социальной ответственности организации, является обязательной частью дипломной работы и включает в себя следующие составляющие: техногенная безопасность, региональная безопасность, организационные мероприятия обеспечения безопасности, особенности законодательного регулирования проектных решений и безопасность в чрезвычайных ситуациях.

Научно-исследовательская работа представляет собой разработку проекта национального стандарта передачи данных по сети и предполагает непосредственную работу с персональным компьютером. Поэтому определение оптимальных условий труда является неотъемлемым критерием безопасности.

6.2. Производственная безопасность

6.2.1. Недостаточная освещенность рабочей зоны

Освещение в недостаточной степени может привести к напряжению зрения, ослаблению внимания и наступлению преждевременной утомленности. Ослепление, резь в глазах и раздражение могут быть вызваны чрезмерно ярким освещением. Неверное направление света на месте труда может создать резкие тени или блики, а также дезориентировать работающего. Перечисленные причины могут привести к профзаболеваниям.

При выполнении работ категории высокой зрительной точности величина коэффициента естественного освещения (КЕО) должна быть не ниже 1,5 %. При зрительной работе средней точности - не ниже 1,0 %. В качестве источников искусственного освещения обычно используются люминесцентные лампы типа ЛБ или ДРЛ, которые попарно объединяются в светильники, равномерно располагающиеся над рабочими поверхностями.

Требования к освещенности в помещениях с компьютерами: при выполнении зрительных работ высокой точности общая освещенность должна составлять 300 лк, а совмещенная – 750 лк; аналогичные требования при выполнении работ средней точности – 200 лк и 300 лк соответственно. Коэффициент пульсации освещенности в кабинетах информатики и вычислительной техники высших учебных заведений не должен превосходить 10 %.

Кроме того все поле зрения должно быть освещено достаточно равномерно – это основные требования.

6.2.2. Отклонение показателей микроклимата

Компьютеры могут привести к повышению температуры и снижению относительной влажности в помещении. В таких помещениях должны соблюдаться определенные параметры микроклимата. В санитарных нормах - СанПиН 2.2.4.548 – 96 установлены величины параметров микроклимата, создающие комфортные условия. Отклонения параметров микроклимата от установленных норм способствуют в первую очередь нарушению физиологической функции человека сохранять тепловой баланс организма, что может повлиять на состояние здоровья и общую производительность труда. В обычных климатических условиях теплоотдача осуществляется в основном за счет излучения (примерно 45 % всей удаляемой организмом теплоты), конвекции (30 %) и испарения (25 %).

При пониженной температуре окружающей среды возрастает удельный вес конвекционно-радиационных теплопотерь. В условиях повышенной температуры среды теплопотери уменьшаются за счет конвекции и излучения, но увеличиваются за счет испарения. При температуре воздуха и ограждений, равной температуре тела, теплоотдача за счет излучения и конвекции практически исчезает и единственным путем теплоотдачи становится испарение пота.

Низкая температура и усиление подвижности воздуха способствуют увеличению теплопотерь конвекцией и испарением.

Роль влажности при пониженных температурах воздуха значительно меньше. В то же время считается, что при низких температурах среды повышенная влажность увеличивает теплопотери организма в результате интенсивного поглощения водяными парами энергии излучения человека. Однако большее увеличение теплопотерь происходит при непосредственном смачивании поверхности тела и одежды.

Таблица 22. Параметры микроклимата для помещений с компьютерами

Период года	Категория работ по уровню энергозатрат, Вт	Параметр микроклимата	Оптимальная величина	Допустимая величина
Холодный	Ia (до 139)	Температура воздуха в помещении	22…24 °С	20…25 °С
		Относительная влажность	40…60 %	15…75 %
		Скорость движения воздуха	до 0,1 м/с	До 0,1 м/с
Теплый	Ia (до 139)	Температура воздуха в помещении	23…25 °С	21…28 °С
		Относительная влажность	40…60 %	15…75 %
		Скорость движения воздуха	до 0,1м/с	0,1 – 0,2 м/с

Таблица 23. Нормы подачи свежего воздуха в помещения с компьютерами

Характеристика помещения	Объемный расход подаваемого в помещение свежего
Объем до 20 м^3 на человека 20…40 м^3 на человека Более 40 м^3 на человека	Не менее 30 Не менее 20 Естественная вентиляция

Объем рабочих помещений не должен быть меньше 19,5 $м^3$/человека с учетом максимального числа одновременно работающих в смену.

6.2.3. Повышенный уровень шума на рабочем месте

Работающие в условиях длительного шумового воздействия испытывают раздражительность, головные боли, головокружение, снижение памяти, повышенную утомляемость, понижение аппетита, боли в ушах и т. д. Все это снижает работоспособность человека и его производительность, качество и безопасность труда.

Для снижения уровня шума стены и потолок помещений, где установлены компьютеры, могут быть облицованы звукопоглощающими материалами.

Таблица 24. Предельные уровни звука, дБ, на рабочих местах.

Категория напряженности труда	Категория тяжести труда	Предельный уровень звука
I. Мало напряженный	I. Легкая	80 дБ

6.2.4. Повышенный уровень электромагнитных излучений

Максимальный уровень рентгеновского излучения на рабочем месте оператора компьютера обычно не превышает 10 мкбэр/ч, а интенсивность ультрафиолетового и инфракрасного излучений от экрана монитора лежит в пределах 10…100 $мВт/м^2$.

Для снижения воздействия этих видов излучения рекомендуется применять мониторы с пониженным уровнем излучения (MPR-II, TCO-92, TCO-99), устанавливать защитные экраны, а также соблюдать регламентированные режимы труда и отдыха.

Таблица 25. Допустимые значения параметров неионизирующих электромагнитных излучений (в соответствии с СанПиН 2.2.2.542-96)

Наименование параметра	Допустимые

Напряженность электрической составляющей электромагнитного поля на расстоянии 50см от поверхности видеомонитора	10 В/м
Напряженность магнитной составляющей электромагнитного поля на расстоянии 50см от поверхности видеомонитора	0,3А/м
Напряженность электростатического поля не должна превышать: для взрослых пользователей	20 кВ/м

6.2.5. Электробезопасность

На рабочем месте располагаются монитор, клавиатура, мышь и системный блок.

Токи статического электричества, на корпусах мониторов, системного блока, мыши и клавиатуры, могут привести к возникновению разрядов при прикосновении к данным элементам. Такие разряды опасности для человека не представляют. Но при возникновении заряда с довольно большим электрическим потенциалом порождает электрическое поле с повышенной напряженностью, которое может нанести вред человеку.

При продолжительном пребывании человека в таком поле наблюдаются изменения центральной нервной и сердечно-сосудистой системы. Также избыточный электрический заряд может способствовать к выходу компьютера из строя.

Для уменьшения величин токов статического электричества на производстве применяют нейтрализаторы статического электричества, которые создают вблизи диэлектрического наэлектризованного объекта положительные и отрицательные ионы.

6.3. Экологическая безопасность

В настоящее время проблема экологической безопасности является приоритетной. Это стало поводом для принятия жестких законов,

ограничивающих обычную утилизацию компьютерной техники. В большей мере это обусловливается тем, что в производстве такой техники используется множество различных материалов, которые способны нанести непоправимый вред окружающей среде и, соответственно, здоровью человека. Утилизация компьютерного оборудования является достаточно сложной. Непосредственная переработка большей части компонентов включает в себя их сортировку, последующую гомогенизацию и отправку для повторного использования, т.е. с предварительным помолом или переплавкой.

Люминесцентные лампы представляют собой «чрезвычайно опасные» виды отходов. Все люминесцентные лампы содержат от 3 до 5 мг ртути. Так как в производстве энергосберегающих ламп используется ртуть (в небольших дозах, в виде амальгамы), их утилизация не должна представлять простой выброс в контейнер с мусором. Лампы должны утилизироваться коммунальными службами, занимающимися вывозом специальных отходов. Транспортировка ламп на полигоны складирования должна выполняться организациями, которые специализируются на утилизации опасных отходов.

6.4. Правовые и организационные вопросы обеспечения безопасности

Площадь на одно рабочее место пользователей ПЭВМ должна составлять не менее 6 м^2. При размещении рабочих мест с персональными компьютерами должны учитываться расстояние между рабочими столами с мониторами.

Не следует размещать рабочие места с ПЭВМ вблизи силовых кабелей и вводов, высоковольтных трансформаторов, технологического оборудования, создающего помехи в работе ПЭВМ.

Температура, относительная влажность и скорость движения воздуха на рабочих местах должны соответствовать действующим санитарным нормам микроклимата производственных помещений.

Внутренняя отделка интерьера помещений с компьютерами должна быть сделана при использовании диффузно-отражающих материалов с

коэффициентами отражения для потолка от 0,7 до 0,8; для стен от 0,5 до 0,6; для пола от 0,3 до 0,5. В помещениях с эксплуатацией компьютеров поверхность пола должна быть нескользкой, ровной и удобной влажной уборки, а также иметь антистатические свойства.

В обязательном порядке в помещении должны находиться углекислотный огнетушитель для тушения пожара и аптечка первой медицинской помощи.

Согласно СанПиНу 2.2.2.542-96 при 8-ми часовой рабочей смене на ВДТ и ПЭВМ перерывы в работе должны составлять от 10 до 20 минут каждые два часа работы.

Согласно ст. 91 ТК РФ нормальная продолжительность рабочего времени в неделю не может превышать 40 часов. Для работников, которые являются инвалидами I или II группы устанавливается сокращенная продолжительность рабочего времени – не более 35 часов в неделю, в соответствии с ст. 92 ТК РФ. Возможно, применение гибкого режима рабочего времени по согласованию с руководством. В соответствии с медицинским заключением беременным женщинам по их заявлению снижаются нормы выработки, при этом сохраняется среднего заработка по прежней работе (ст. 254 ТК РФ).

6.5. Безопасность в чрезвычайных ситуациях

Пожарная безопасность — состояние защищённости личности, имущества, общества и государства от пожаров. Пожарная безопасность обеспечивается системой пожарной защиты и системой предотвращения пожара.

Опасный фактор пожара (ОФП):

- пламя и искры;
- тепловой поток;
- повышенная температура окружающей среды;

- повышенная концентрация токсичных продуктов горения и термического разложения;
- пониженная концентрация кислорода;
- снижение видимости в дыму.

Действия при пожаре в здании:

- При наличии телефона, «112» или «01» сообщить о пожаре и своем местоположении;
- Не входить в места с высокой концентрацией дыма и видимостью менее, чем 10 метров.
- Покинуть помещение, используя запасные и основные пути эвакуации;
- Попутно отключить электроэнергию;
- Передвигаться к выходу на четвереньках, при этом закрывая рот и нос подручными средствами защиты;
- Плотно закрыть дверь при выходе;

Если дым и пламя в соседних помещениях не позволяет выйти наружу:

- Стараться не поддаваться панике;
- Проверить возможности спуститься по пожарной лестнице или выйти на крышу;
- При отсутствии возможности эвакуироваться для защиты от дыма и тепла необходимо как можно надёжней загерметизировать своё помещение:
- Плотно закрыть двери, окна и форточки, заткнуть щели изнутри, используя при этом любую, желательно мокрую, ткань;
- При наличии воды, постоянно смачивать двери и пол.

Заключение

На основании исследовательской работы можно сделать вывод о значимости процесса мониторинга бурения, как важного этапа всей нефтегазовой промышленности. Мониторинг бурения доказал свою эффективность на практике неоднократно. При текущем уровне научно-технического прогресса методы, технологии и средства мониторинга становятся все более сложными и эффективными. Применение единого стандарта, позволяющего максимально точно и своевременно передавать информацию от буровых установок в головные офисы и на рабочие места технологов, значительно повышает успех мониторинга и принятие управленческих решений в целом. Таким образом, внедрение стандарта WITSML в отечественной промышленности является закономерным явлением. Данный стандарт поддерживает съем большого количества параметров, а также позволяет программно расширять номенклатуру этих параметров. Это, а также его надежность, которая была проверена годами исследований и практического использования, позволяет говорить о преимуществах его внедрения в отечественных серверах. Объединение всех систем в единой информационное пространство положительно скажется на точности построения прогнозов, на качестве анализа, на скорости и качестве принятия решений. Разработка агента и сервера, интегрированных с национальным стандартом, позволит значительно повысить эффективность бурения и хранения данных в отечественной промышленности.

В рамках дипломной работы было разработано приложение, отражающее в себе систему оценки доходности сервера. Приложение было реализовано на основании таких web-технологий, как MVC. Результатом работы приложения стала возможность взаимодействия с клиентами, формирования контрактов, сбор информации о клиенте и представляемых им данных, расчет стоимости услуг индивидуально для каждого клиента в зависимости от типа скважин,

количества параметров, частоте съема данных и срока хранения, а также расчет потребности в оперативной памяти сервера, удовлетворяющей потребностям клиента. Для определения рентабельности и эффективности расчетной системы, а также для формирования ценовой политики был проведен предварительный анализ расходов на содержание и эксплуатацию сервера.

Список публикаций, достижений и интеллектуальная собственность студентов:

1. Свидетельство о регистрации программы для ЭВМ "Агент WITSML Адениум для станции управления бурением", Черкашин А.Ю., Марчуков А.В., Гончаров А.С., Савельев А.О.
2. Свидетельство о регистрации программы для ЭВМ "Сервер WITSML Адонис для обработки данных бурения", Черкашин А.Ю., Марчуков А.В., Гончаров А.С.
3. Всероссийская конференция "Современные технологии поддержки принятия решений в экономике ис-24/2015", тема"Проблемы создания единого информационного простарнства нефтяного предприятия на основне языка XML", диплом, Гончаров А.С.
4. Всероссийский конкурс проектов и разработок в области it-технологий "IT-прорыв", тема "WITSML Server", диплом 1 степени, Гончаров А.С.
5. 54-ая международная научная студенческая конференция МНСК-2016, Гончаров А.С.
6. Международная конференция"Молодежь и современные технологии", тема "Проблемы создания единого информационного пространства нефтяного предприятия на основе языка XML" №82, Гончаров А.С.

Список использованных источников:

1. Сэм Улиарис, «Что такое экономические модели?», Финансы и развитие, июнь 2011.
2. Н.М. Абдикеев, «Интернет технологии в экономике знаний: Учебник»- М.:ИНФРА-М, 2012.
3. Ю.М. Осипова, Е.Е. Смирнова, «Основы предпринимательского дела: учебник» 2-е издание, Издательство БЭК, 1996.
4. Пикеринг, Д., Грёвик, Л., Франссенс, Д., Дикс, Н., Донигер, А., Шей, Д.: “WITSML достигает совершеннолетия в глобальной индустрии бурения и освоения”. США, 2009.
5. [Электронный ресурс] - Оптимизация добычи и интеллектуальные скважины, http://www.weatherford.ru/ru/service/production/53.
6. [Электронный ресурс] - Оптоволоконные датчики в бурении, http://stud24.ru/geology/optovolokonnye-datchiki-vburenii/476130-1811034.html
7. [Электронный ресурс] - портал Energistics, http://www.energistics.org/
8. [Электронный ресурс] - портал XML, http://www.codenet.ru/webmast/xml/part1.php
9. [Электронный ресурс] - статья XML схемы, https://ru.wikipedia.org/wiki/XML_Schema
10. [Электронный ресурс] - Промышленные стандарты обмена данными по бурению и заканчиванию скважин, http://www.energistics.org/Assets/witsmlflyerrussian.pdf
11. Майо Д. Самоучитель Microsoft Visual Studio 2010, С.: «БХВ-Петербург», 2010. — С. 464.
12. Дж. Рихтер, Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд.
13. Александр Климов, советы программистам, «БХВ-Петербург», 2008. - С. 544.

14. Beeming G. Team Foundation Server 2013 Customization, Packt Publishing, 2014. — С. 102.
15. Мейер Дж.Д., Тейлор Джейсон и др. Командная разработка с использованием Visual Studio Team Foundation Server: Справочник, 2007. - С. 575
16. Виктор Гольцман , SQL и реляционная теория.
17. [Электронный ресурс] - Microsoft SQL Server, https://ru.wikipedia.org/wiki/Microsoft_SQL_Server/
18. [Электронный ресурс] – Rogtec, http://rogtecmagazine.com/
19. [Электронный ресурс] Положение о ВКР ТПУ, http://tpu.ru/f/2039/vkr.pdf#2

Приложения

Приложение 1. Программный код контроллеров

Контроллер «Admin»

```
using System;
using System.Linq;
using System.Threading.Tasks;
using PagedList;
using System.Web.Mvc;
using WitsmlServer.Web.DbModels.Error;
using WitsmlServer.Web.DbModels.Field;
using WitsmlServer.Web.DbModels.F_G_W_Model;
using WitsmlServer.Web.DbModels.Group;
using WitsmlServer.Web.DbModels.Policy;
using WitsmlServer.Web.DbModels.UserPolicy;
using WitsmlServer.Web.DbModels.well;
using Field = WitsmlServer.Web.DbModels.Field.Field;
using Group = WitsmlServer.Web.DbModels.Group.Group;
using Well = WitsmlServer.Web.DbModels.well.Well1;
using Policy = WitsmlServer.Web.DbModels.Policy.Policy;

namespace witsml_app.Controllers
{
    using System.Linq.Expressions;

    using NLog;

    using witsml_app.ServiceClient;

    public class AdminController : BaseController
    {
        private static readonly Logger Logger = LogManager.GetCurrentClassLogger();


        [Authorize(Roles = "Admin")]
        public ActionResult Administration(int? page, string searchString, int? pageSize,
string sortBy)
        {
            //главная страница администратора, она же - страница управления пользователями

            var pagenumber = page ?? 1;
            string SearchString = searchString ?? "";
            var PageSize = pageSize ?? 10;

            ViewBag.SortBy = sortBy;
            ViewBag.SearchString = SearchString;
            ViewBag.PageSize = PageSize;
            ViewBag.PageNumber = pagenumber;
            return View();
        }


        [Authorize(Roles = "Admin")]
        public ActionResult AdministrationPolitics(int? page, string searchString, int?
pageSize)
        {
            //страница управления политиками
```

```
            var pagenumber = page ?? 1;
            var PageSize = pageSize ?? 10;
            var db = new PolicyModel();

            if (!String.IsNullOrEmpty(searchString))
            {
                var query =
                    from policy in db.Policy
                    where policy.Name.ToUpper().Contains(searchString.ToUpper())
                    orderby policy.Id
                    select policy;
                var politics = query.ToPagedList(pagenumber, PageSize);
                ViewBag.query = politics;
            }
            else
            {
                var query =
                    from policy in db.Policy
                    orderby policy.Id
                    select policy;
                var politics = query.ToPagedList(pagenumber, PageSize);
                ViewBag.query = politics;
            }

            ViewBag.PageSize = PageSize;
            ViewBag.SearchString = searchString;
            ViewBag.PageNumber = pagenumber;

            return View();
        }
        [Authorize(Roles = "Admin")]
        public ActionResult AdministrationWells(int? page, string searchString, int?
pageSize)
        {
            //страница управления скважинами

            var pagenumber = page ?? 1;
            string SearchString = searchString ?? "";
            var PageSize = pageSize ?? 10;

            ViewBag.SearchString = SearchString;
            ViewBag.PageSize = PageSize;
            ViewBag.PageNumber = pagenumber;

            RefreshWells();
            return View();
        }
        [Authorize(Roles = "Admin")]
        public ActionResult AdministrationGroups(int? page, string searchString, int?
pageSize)
        {
            //страница управления логических групп скважин

            var pagenumber = page ?? 1;
            var PageSize = pageSize ?? 10;
            var db = new GroupModel();
            if (!String.IsNullOrEmpty(searchString))
            {
                var query =
                    from _group in db.Group
                    where _group.Name.ToUpper().Contains(searchString.ToUpper())
                    orderby _group.Id
                    select _group;
```

```
                var groups = query.ToPagedList(pagenumber, PageSize);
                ViewBag.query = groups;
            }
            else
            {
                var query =
                    from _group in db.Group
                    orderby _group.Id
                    select _group;
                var groups = query.ToPagedList(pagenumber, PageSize);
                ViewBag.query = groups;
            }
            ViewBag.PageSize = PageSize;
            ViewBag.SearchString = searchString;
            ViewBag.PageNumber = pagenumber;
            return View();
        }
        [Authorize(Roles = "Admin")]
        public ActionResult AdministrationFields(int? page, string searchString, int?
pageSize)
        {
            //страница управления месторождениями

            var pagenumber = page ?? 1;
            var PageSize = pageSize ?? 10;
            var db = new FieldModel();
            if (!String.IsNullOrEmpty(searchString))
            {
                var query =
                    from field in db.Field
                    where field.Name.ToUpper().Contains(searchString.ToUpper())
                    orderby field.Id
                    select field;
                var fields = query.ToPagedList(pagenumber, PageSize);
                ViewBag.query = fields;
            }
            else
            {
                    var query =
                        from field in db.Field
                        orderby field.Id
                        select field;
                    var fields = query.ToPagedList(pagenumber, PageSize);
                    ViewBag.query = fields;
            }
            ViewBag.PageSize = PageSize;
            ViewBag.PageNumber = pagenumber;
            ViewBag.SearchString = searchString;
            return View();
        }
         [Authorize(Roles = "Admin")]
        public ActionResult AddNewUser()
        {
            return View();
        }
         [Authorize(Roles = "Admin")]
        public ActionResult AddNewPolicy()
        {
            return View();
        }

        [Authorize(Roles = "Admin")]
        [HttpPost]
```

```
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> AddNewPolicy(Policy model)
        {
            //метод добавления новой политики

            if (ModelState.IsValid)
            {
                try
                {
                    var db = new PolicyModel();

                    int i = db.Policy.Count();

                    var query = from p in db.Policy select p;

                    foreach (var p in query)
                    {
                        if (p.Id == i + 1) i++;
                    }

                    var policy = new Policy
                                    {
                                        Id = i + 1,
                                        Name = model.Name,
                                        Duration = model.Duration,
                                        MaxWells = model.MaxWells,
                                        MaxActiveWells = model.MaxActiveWells,
                                        Actuality = model.Actuality,
                                        Cost = model.Cost,
                                        Add = model.Add,
                                        Get = model.Get,
                                        Update = model.Update,
                                        Delete = model.Delete
                                    };

                    db.Policy.Add(policy);
                    db.SaveChanges();
                    return RedirectToAction("AdministrationPolitics", "Admin", new { page =
1 });
                }
                catch (Exception eee)
                {
                    ErrorModel erm = new ErrorModel();
                    erm.Name = model.Name;
                    return View("Error", erm);
                }
            }
            return RedirectToAction("AdministrationPolitics", "Admin", new { page = 1 });
        }

        [Authorize(Roles = "Admin")]
        public ActionResult AddNewWell()
        {
            return View();
        }

        [Authorize(Roles = "Admin")]
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> AddNewWell(EditWellModel model)
        {
            // метод добавления новой скважины
            if (!this.ModelState.IsValid)
```

```
            {
                return this.View(model);
            }

            try
            {
                var userPolicyDb = new UserPolicyDbContext();

                var maxWells = (from userPolicy in userPolicyDb.User_Policy
                                from policy in userPolicyDb.Policy
                                where userPolicy.UserName == model.Id_User && policy.Id ==
userPolicy.Id_Policy
                                select policy.MaxWells).First();

                var wellDb = new WellDbContext();
                var amountOfWellsAlreadyExists =
                    (from well in wellDb.Well1 where well.Id_User == model.Id_User select
well).Count();

                var wellUidExists = wellDb.Well1.Any(x => x.WellUid == model.WellUid);

                if (wellUidExists)
                {
                    return this.View("ErrorWellUid");
                }

                if (amountOfWellsAlreadyExists >= maxWells)
                {
                    ErrorCountWells erm = new ErrorCountWells();
                    erm.Name = model.Id_User;
                    return this.View("ErrorCountWells", erm);
                }

                var db = new WellDbContext();

                var db1 = new FieldGroupModel();
                var field = db1.Field.SingleOrDefault(s => s.Id == model.Id_Field);

                // высчитывание времени активности скважины
                var lastTimeUpdate = DateTime.Now;
                var lastActiveDate =
                    DateTime.Now.AddYears(model.Years ?? 0)
                            .AddMonths(model.Months ?? 0)
                            .AddDays(model.Days ?? 0)
                            .AddHours(model.Hours ?? 0);

                var newWell = new Well
                    {
                        WellUid = Guid.NewGuid().ToString(),
                        Name = model.Name,
                        Status = "Active",
                        Id_User = model.Id_User,
                        Id_Field = field == null ? (int?)null : field.Id,
                        LastTimeUpdate = lastTimeUpdate,
                        LastActiveDate = lastActiveDate,
                        ActiveTime = (lastActiveDate - lastTimeUpdate).Days
                    };

                /* Добавление в базу сервера */
                int resultCode;
                string errorMessage;
                WitsmlService<IInternalService>.Use(
                    service =>
```

```
                        {
                            if ((resultCode = service.CreateWell(model.Id_User,
newWell.WellUid, model.Name, out errorMessage)) != 1)
                            {
                                throw new Exception(string.Format("Code: {0}. {1}",
resultCode, errorMessage));
                            }
                        });

                db.Well1.Add(newWell);
                db.SaveChanges();

                return this.RedirectToAction("AdministrationWells", "Admin", new {page =
1});
            }
            catch (Exception ex)
            {
                ErrorModel erm = new ErrorModel();
                erm.Name = model.Name;
                Logger.Error(ex);
                return this.View("Error", erm);
            }
        }

        [Authorize(Roles = "Admin")]
        public ActionResult AddNewGroup()
        {
            return View();
        }

        [Authorize(Roles = "Admin")]
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> AddNewGroup(Group model)
        {
            //метод добавления новой группы скважин

            if (ModelState.IsValid)
            {
                try
                {
                    var db = new GroupModel();
                    var group = new Group
                    {
                        Name = model.Name
                    };
                    db.Group.Add(group);
                    db.SaveChanges();
                    return RedirectToAction("AdministrationGroups", "Admin", new { page = 1
});

                }
                catch (Exception eee)
                {
                    ErrorModel erm = new ErrorModel();
                    erm.Name = model.Name;
                    return View("Error",erm);
                }

            }
            return RedirectToAction("AdministrationGroups");
        }
```

```
        [Authorize(Roles = "Admin")]
        public ActionResult AddNewField()
        {
            return View();
        }

        [Authorize(Roles = "Admin")]
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> AddNewField(Field model)
        {
            //метод добавления нового месторождения

            if (ModelState.IsValid)
            {

                try
                {
                    var db = new FieldModel();

                    var db1 = new GroupModel();
                    var idNoGroup = (from gr in db1.Group where gr.Name.Contains("NoGroup")
select gr).First().Id;

                    var field = new Field
                    {
                        Name = model.Name,
                        Id_Group = idNoGroup
                    };

                    db.Field.Add(field);
                    db.SaveChanges();

                    return RedirectToAction("AdministrationFields", "Admin", new { page = 1
});
                }
                catch (Exception eee)
                {
                    ErrorModel erm = new ErrorModel();
                    erm.Name = model.Name;
                    return View("Error", erm);
                }
            }

            return RedirectToAction("AdministrationFields", "Admin", new { page = 1 });
        }

        [Authorize(Roles = "Admin, User, User_Test")]
        public ActionResult RefreshWells()
        {
            //обновление страницы управления скважинами, для пересчета дней активности ...

            var db = new FieldGroupModel();
            var link =
                from well in db.Well1
                select well;

            foreach (var w in link)
            {
                if ((w.LastTimeUpdate >= w.LastActiveDate) || (w.ActiveTime == 0) ||
(w.LastActiveDate<DateTime.Now))
                {
                    w.Status = "Inactive";
```

```
                }
            }
            db.SaveChanges();
            return RedirectToAction("AdministrationWells", "Admin");
        }

    }
}
```

Контроллер «Base»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace witsml_app.Controllers
{
    using System.Threading;

    public class BaseController : Controller
    {
        // GET: Base
        protected override IAsyncResult BeginExecuteCore(AsyncCallback callback, object
state)
        {
            string cultureName = null;

            // Attempt to read the culture cookie from Request
            HttpCookie cultureCookie = Request.Cookies["_culture"];
            if (cultureCookie != null)
                cultureName = cultureCookie.Value;
            else
                cultureName = Request.UserLanguages != null && Request.UserLanguages.Length
> 0 ?
                        Request.UserLanguages[0] :  // obtain it from HTTP header
AcceptLanguages
                        null;
            // Validate culture name
            cultureName = CultureHelper.GetImplementedCulture(cultureName); // This is safe

            // Modify current thread's cultures
            Thread.CurrentThread.CurrentCulture = new
System.Globalization.CultureInfo(cultureName);
            Thread.CurrentThread.CurrentUICulture = Thread.CurrentThread.CurrentCulture;

            return base.BeginExecuteCore(callback, state);
        }
    }
}
```

Контроллер «Edit»

```
using System;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Threading.Tasks;
using System.Web.Mvc;
```

```
using WitsmlServer.Web.DbModels.EditModel;
using WitsmlServer.Web.DbModels.Field;
using WitsmlServer.Web.DbModels.F_G_W_Model;
using WitsmlServer.Web.DbModels.Group;
using WitsmlServer.Web.DbModels.Policy;
using WitsmlServer.Web.DbModels.UserPolicy;
using WitsmlServer.Web.DbModels.UserRolePolicy;
using Field = WitsmlServer.Web.DbModels.Field.Field;
using Group = WitsmlServer.Web.DbModels.Group.Group;
using Policy = WitsmlServer.Web.DbModels.Policy.Policy;

namespace witsml_app.Controllers
{
    using System.Windows.Forms;

    public class EditController : BaseController
    {

        // GET: Edit
        [Authorize(Roles = "Admin")]
        public ActionResult EditUser(string userName, int? pageNumber)
        {
            //страница редактирования пользователя
            try {
            if (userName == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            var db = new UserRolePolicyModel();
            var query =
                from user_p in db.User_Policy
                from user in db.AspNetUsers
                from policy in db.Policy
                where user_p.UserName == user.UserName && policy.Id == user_p.Policy.Id &&
user.UserName == userName
                select new
                {
                    user,
                    policy,
                    user_p
                };
            EditUserModel e = new EditUserModel();
            foreach (var u in query)
            {
               // var role = db.AspNetUsers.Find(u.user.Id).AspNetRoles;
                e.HLogin = u.user.UserName;
                e.Login = u.user.UserName;
                e.Role = u.user_p.Policy.Name;
                e.Policy = u.policy.Name;
                e.Date_Start = u.user_p.Start_Date;
            }

            var pagenumber = pageNumber ?? 1;

            ViewBag.PageNumber = pagenumber;
            ViewBag.userName = userName;

            var db2 = new FieldGroupModel();
            var query2 =
                from well in db2.Well1
                from field in db2.Field
                from group_ in db2.Group
```

```
                where well.Id_Field == field.Id && field.Id_Group == group_.Id &&
well.Id_User == userName
                select well;
            ViewBag.Wells = query2;
            return View(e);
            }
            catch (Exception ee)
            {
                return View("Error");
            }
        }


        [HttpPost]
        [ValidateAntiForgeryToken]
        [Authorize(Roles = "Admin")]
        public async Task<ActionResult> EditUser(EditUserModel model, bool changeStartDate)
        {
            //метод редактирования пользователя
            try {
            if (!ModelState.IsValid)
            {
                return View(model);
            }
            var db = new UserRolePolicyModel();
            var query =
                from user_p in db.User_Policy
                from user in db.AspNetUsers
                from policy in db.Policy
                from role in db.AspNetRoles
                where user_p.UserName == user.UserName && policy.Id == user_p.Policy.Id &&
user.UserName == model.HLogin && role.Name==model.Role
                select new
                {
                    role,
                    user,
                    policy,
                    user_p
                };
            foreach (var u in query.ToList())
            {
                u.user.AspNetRoles.Clear();
                u.user.AspNetRoles.Add(u.role);
                var query1 =
                    from policy1 in db.Policy
                    where policy1.Name == model.Policy
                    select policy1;
                foreach (var p in query1)
                {
                    u.user_p.Id_Policy = p.Id;
                    u.user_p.End_Date = u.user_p.Start_Date.AddDays(u.policy.Duration);
                    if (changeStartDate)
                    {
                        u.user_p.Start_Date = DateTime.Now;
                    }
                }

            }
            db.SaveChanges();
            return RedirectToAction("Administration", "Admin", new { page = 1 });
            }
            catch (Exception ee)
            {
```

```
                return View("Error");
            }
        }

        [Authorize(Roles = "Admin")]
        public ActionResult EditPolicy(int id)
        {
            //страница редактирования политики
            try {
            var db = new PolicyModel();
            Policy policy = db.Policy.Find(id);
            if (policy == null)
            {
                return HttpNotFound();
            }
            return View(policy);
            }
            catch (Exception ee)
            {
                return this.View();
            }
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        [Authorize(Roles = "Admin")]
        public ActionResult EditPolicy([Bind(Include = "Id, Name, Duration,
MaxWells,MaxActiveWells, Cost, Add, Get, Update, Delete")]Policy model)
        {
            //метод редактирования политики
            try {
            if (!ModelState.IsValid)
            {
                return View(model);
            }
            var db = new PolicyModel();
            decimal a = model.Cost ?? 0;
            db.Entry(model).State = EntityState.Modified;
            db.SaveChanges();
            return RedirectToAction("AdministrationPolitics", "Admin", new { page = 1 });
            }
            catch (Exception ee)
            {
                return View("Error");
            }
        }

        [Authorize(Roles = "Admin")]
        public ActionResult EditWell(string id)
        {
            //страница редактирования скважины
            try {
            EditWellModel model = new EditWellModel();
            var db = new FieldGroupModel();
            var query =
                from well in db.Well1
                from _group in db.Group
                from field in db.Field
                where well.WellUid == id && field.Id == well.Id_Field && _group.Id ==
field.Id_Group
                select new
                {
                    well,
                    field
```

```
                };
            foreach (var w in query)
            {
                model.Name = w.well.Name;
                model.Status = w.well.Status;
                model.Id_User = w.well.Id_User;
                model.Id_Field = w.field.Id;
                model.HName = w.well.Name;
                model.WellUid = w.well.WellUid;
                model.ActiveTime = w.well.ActiveTime;
                model.LastTimeUpdate = w.well.LastTimeUpdate;
                model.LastActiveDate = w.well.LastActiveDate;
            }
            return View(model);
            }
            catch (Exception ee)
            {
                return this.View();
            }
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        [Authorize(Roles = "Admin")]
        public async Task<ActionResult> EditWell(EditWellModel model)
        {
            //метод редактирования скважины
            try {
            if (!ModelState.IsValid)
            {
                return View(model);
            }
            var db = new FieldGroupModel();
            var query =
                from _group in db.Group
                from field in db.Field
                from well in db.Well1
                where field.Id == well.Id_Field && _group.Id == field.Id_Group && well.Name
== model.HName
                select well;
            foreach (var w in query)
            {
                w.WellUid = model.WellUid;
                w.Name = model.Name;
                w.Id_User = model.Id_User;
                w.Id_Field = model.Id_Field;
                w.LastTimeUpdate = DateTime.Now;
                w.LastActiveDate = w.LastTimeUpdate;
                w.LastActiveDate = w.LastActiveDate.Value.AddYears(model.Years ?? 0);
                w.LastActiveDate = w.LastActiveDate.Value.AddMonths(model.Months ?? 0);
                w.LastActiveDate = w.LastActiveDate.Value.AddDays(model.Days ?? 0);


                if ((model.Years==null) &&
                    (model.Months == null) &&
                    (model.Days == null))
                {

                    w.ActiveTime = model.ActiveTime;
                    w.LastActiveDate = w.LastActiveDate.Value.AddDays(w.ActiveTime);
                }
                else
                {
```

```
                    w.ActiveTime = (w.LastActiveDate - w.LastTimeUpdate).Value.Days;
                }
                var user_name = model.Id_User;
               /* var count_active_wells =
               ( from well in db.Well1
                where well.Id_User == user_name && well.Status == "Active"
                select well).Count();*/


               /* var db2 = new UserPolicyDbContext();
                var link2 =
                    from user_policy in db2.User_Policy
                    from policy in db2.Policy
                    where user_policy.UserName == user_name && policy.Id ==
user_policy.Id_Policy
                    select policy;
                int max_active_wells = 0;
                foreach (var p in link2)
                {
                    max_active_wells = p.MaxActiveWells ?? 0;
                }
                if (model.Status == "Inactive")
                {
                    count_active_wells++;
                }
                if ((w.LastActiveDate > w.LastTimeUpdate) && (w.LastActiveDate >=
DateTime.Now) && (max_active_wells >= count_active_wells))
                {*/
                    w.Status = "Active";
              /*  }
                else w.Status = "Inactive";*/
            }
            db.SaveChanges();
            return RedirectToAction("AdministrationWells", "Admin", new { page = 1 });
            }
            catch (Exception ee)
            {
                return View("Error");
            }
        }

        [Authorize(Roles = "Admin")]
        public ActionResult EditGroup(int id)
        {
            //страница редактирования группы скважин
            try {
            var db = new GroupModel();
            Group group = db.Group.Find(id);
            if (group == null)
            {
                return HttpNotFound();
            }
            return View(group);
            }
            catch (Exception ee)
            {
                return this.View();
            }
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        [Authorize(Roles = "Admin")]
        public ActionResult EditGroup([Bind(Include = "Id, Name")]Group model)
```

```
{
    //метод редактирования группы скважин
    try {
    if (!ModelState.IsValid)
    {
        return View(model);
    }
    var db = new GroupModel();
    db.Entry(model).State = EntityState.Modified;
    db.SaveChanges();
    return RedirectToAction("AdministrationGroups", "Admin", new { page = 1 });
    }
    catch (Exception ee)
    {
        return View("Error");
    }
}
[Authorize(Roles = "Admin")]
public ActionResult EditField(int id)
{
    //страница редактирования месторождения
    try {
    var db = new FieldModel();
    Field field = db.Field.Find(id);
    if (field == null)
    {
        return HttpNotFound();
    }
    return View(field);
    }
    catch (Exception ee)
    {
        return this.View();
    }
}
[HttpPost]
[ValidateAntiForgeryToken]
[Authorize(Roles = "Admin")]
public ActionResult EditField([Bind(Include = "Id, Name, Id_Group")]Field model)
{
    //метод редактирования месторождения
    try {
    if (!ModelState.IsValid)
    {
        return View(model);
    }
    var db = new FieldModel();
    db.Entry(model).State = EntityState.Modified;

    db.SaveChanges();
    return RedirectToAction("AdministrationFields", "Admin", new { page = 1 });
    }
    catch (Exception ee)
    {
        return View("Error");
    }
}

[Authorize(Roles = "Admin")]
public ActionResult EditGroupWells(int id)
{
    //страница редактирования группы скважин
    try {
```

```
            var db = new FieldGroupModel();
            var link = from well in db.Well1
                       from field in db.Field
                       where well.Id_Field == field.Id && field.Id_Group == id
                       select well;
            if (id == null)
            {
                return HttpNotFound();
            }

            var group = db.Group.Find(id).Name;

            ViewBag.Wells = link;
            ViewBag.Group = group;
            }
            catch (Exception ee)
            {
                return View("Error");
            }
            return View();
        }
    }
}
```

Контроллер «EditUser»

```
using System;
using System.Linq;
using System.Threading.Tasks;
using System.Web.Mvc;
using Microsoft.AspNet.Identity;
using WitsmlServer.Web.DbModels.F_G_W_Model;
using WitsmlServer.Web.DbModels.UserPolicy;

namespace witsml_app.Controllers
{
    using WitsmlServer.Web.DbModels.UserRolePolicy;

    public class EditUserController : BaseController
    {
        // GET: EditUser
        [Authorize(Roles = "Admin,User,User_Test")]
        public ActionResult EditUserWell(string id)
        {
            //страница редактирования скважины со стороны пользователя (в личном кабинете)
            try {
            EditWellModel model = new EditWellModel();
            var db = new FieldGroupModel();
            var query =
                from well in db.Well1
                from _group in db.Group
                from field in db.Field
                where well.WellUid == id && field.Id == well.Id_Field && _group.Id ==
field.Id_Group
                select new
                {
                    well,
                    _group,
                    field
                };
            foreach (var w in query)
            {
```

```
                model.Name = w.well.Name;
                model.WellUid = w.well.WellUid;
                model.Status = w.well.Status;
                model.Id_User = w.well.Id_User;
                model.Id_Group = w._group.Id;
                model.Id_Field = w.field.Id;
                model.HName = w.well.Name;
                model.ActiveTime = w.well.ActiveTime;
            }
            return View(model);
            }
            catch (Exception ee)
            {
                return View("Error");
            }
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        [Authorize(Roles = "Admin,User,User_Test")]
        public async Task<ActionResult> EditUserWell(EditWellModel model)
        {
            //метод редактирования скважины со стороны пользователя
            try {
            if (!ModelState.IsValid)
            {
                return RedirectToAction("Index", "PersonalPage");
            }
            var db = new FieldGroupModel();
            var query =
                from _group in db.Group
                from field in db.Field
                from well in db.Well1
                where field.Id == well.Id_Field && _group.Id == field.Id_Group && well.Name
== model.HName
                select new
                {
                    well,
                    _group,
                    field
                };
            string user_name = User.Identity.GetUserName();
            var active_well =
                from well in db.Well1
                where well.Id_User == user_name && well.Status == "Active"
                select well;
            var active_wells = active_well.Count();
            //var db1 = new UserPolicyDbContext();
                var db1 = new UserRolePolicyModel();
          //  var max_active_wells = db1.User_Policy.Where(x => x.UserName ==
user_name).First().Policy.MaxActiveWells;
            var max_active_wells = db1.User_Policy.Where(x => x.UserName ==
user_name).First().Policy.MaxActiveWells;


            foreach (var w in query)
            {
               // w.well.WellUid = model.WellUid;
                w.well.Name = model.Name;
                w.well.Id_User = User.Identity.GetUserName();
                w.well.Id_Field = model.Id_Field;
                w.well.LastTimeUpdate = DateTime.Now;
```

```
                w.well.LastActiveDate =
w.well.LastTimeUpdate.Value.AddDays(model.ActiveTime);
                w.field.Id_Group = model.Id_Group;
                if ((w.well.LastTimeUpdate < w.well.LastActiveDate) && (max_active_wells >
active_wells))
                {
                    w.well.Status = "Active";
                }
                else
                {
                    w.well.Status = "Inactive";
                }
            }
            db.SaveChanges();
            return RedirectToAction("Index", "PersonalPage");
            }
            catch (Exception ee)
            {
                return View("Error");
            }
        }

    }
}
```

Контроллер «Home»

```
using System;
using System.Web;
using System.Web.Mvc;
using witsml_app.Models;
using System.Resources;

namespace witsml_app.Controllers
{
    using System.Data.Entity.Migrations.History;
    using System.Linq;
    using System.Windows.Forms;

    using WitsmlServer.Web.DbModels.Comment;

    public class HomeController : BaseController
    {
        [HttpGet]
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult HelpPage()
        {
            return this.View("_HelpMessage");
        }

        [HttpPost]
        public ActionResult Index(ContentLanguage language)
        {
            return View();
        }
        public ActionResult SetCulture(string culture)
        {
            // Validate input
```

```
            culture = CultureHelper.GetImplementedCulture(culture);
            // Save culture in a cookie
            HttpCookie cookie = Request.Cookies["_culture"];
            if (cookie != null)
                cookie.Value = culture;   // update cookie value
            else
            {
                cookie = new HttpCookie("_culture");
                cookie.Value = culture;
                cookie.Expires = DateTime.Now.AddYears(1);
            }
            string url = this.Request.UrlReferrer.AbsolutePath;
            Response.Cookies.Add(cookie);
            return Redirect(url);
        }

        public ActionResult About()
        {
            return View();
        }

        public ActionResult Contact()
        {
            return View();
        }

        public ActionResult Comments()
        {
            var db = new CommentModel();
            var comments =
                from c in db.Comment
                orderby c.Date descending
                select c;
            ViewBag.Comments = comments;
            return View();
        }

        [Authorize(Roles = "Admin, User, User_Test")]
        [HttpPost]
        public ActionResult AddComment(string comment, string remark)
        {
            var db = new CommentModel();
            Comment newComment = new Comment()
                                    {
                                        Comment1 = comment,
                                        Date = DateTime.Now,
                                        Remark = remark,
                                        UserName = User.Identity.Name
                                    };
            db.Comment.Add(newComment);
            db.SaveChanges();
            return RedirectToAction("Comments");
        }

    }
}
```

Контроллер «Personal Page»

```
namespace witsml_app.Controllers
{
    using System;
```

```
    using System.Collections.Generic;
    using System.IO;
    using System.Linq;
    using System.Threading.Tasks;
    using System.Web.Mvc;
    using Microsoft.AspNet.Identity;
    using PagedList;
    using WitsmlServer.Web.DbModels.TestModel;
    using witsml_app.PythonTestScripts;
    using witsml_app.ServiceClient;
    using WitsmlServer.Web.DbModels.F_G_W_Model;
    using WitsmlServer.Web.DbModels.Group;
    using WitsmlServer.Web.DbModels.UserPolicy;
    using WitsmlServer.Web.DbModels.well;

    using Group = WitsmlServer.Web.DbModels.Group.Group;

    public class PersonalPageController : BaseController
    {
        // главная страница в личном кабинете - она же и является списком скважин
пользователя
        [Authorize(Roles = "Admin,User,User_Test")]
        public ActionResult Index(int? page, string searchString, int? pageSize)
        {

            searchString = searchString ?? "";

            this.ViewBag.SearchString = searchString;
            this.ViewBag.PageSize = pageSize ?? 10;
            this.ViewBag.PageNumber = page ?? 1;

            return this.View();
        }

        // страница списка доступных групп
        [Authorize(Roles = "Admin,User,User_Test")]
        public ActionResult IndexGroup(int? page, string searchString, int? pageSize)
        {
            var db = new GroupModel();
            if (!string.IsNullOrEmpty(searchString))
            {
                var query =
                    from @group in db.Group
                    where @group.Name.ToUpper().Contains(searchString.ToUpper())
                    orderby @group.Id
                    select @group;
                var groups = query.ToPagedList(page ?? 1, pageSize ?? 10);
                this.ViewBag.query = groups;
            }
            else
            {
                var query =
                    from @group in db.Group
                    orderby @group.Id
                    select @group;
                var groups = query.ToPagedList(page ?? 1, pageSize ?? 10);
                this.ViewBag.query = groups;
            }

            this.ViewBag.PageSize = pageSize ?? 10;
            this.ViewBag.SearchString = searchString;
            this.ViewBag.PageNumber = page ?? 1;
            return this.View();
```

```
        }

        // страница списка доступных месторождений
        [Authorize(Roles = "Admin,User,User_Test")]
        public ActionResult IndexField(int? page, string searchString, int? pageSize)
        {
            var db = new FieldGroupModel();
            if (!string.IsNullOrEmpty(searchString))
            {
                var query =
                    from @field in db.Field
                    where @field.Name.ToUpper().Contains(searchString.ToUpper())
                    orderby @field.Id
                    select @field;
                var fields = query.ToPagedList(page ?? 1, pageSize ?? 10);
                this.ViewBag.query = fields;
            }
            else
            {
                var query =
                    from @field in db.Field
                    orderby @field.Id
                    select @field;
                var fields = query.ToPagedList(page ?? 1, pageSize ?? 10);
                this.ViewBag.query = fields;
            }

            this.ViewBag.PageSize = pageSize ?? 10;
            this.ViewBag.SearchString = searchString;
            this.ViewBag.PageNumber = page ?? 1;
            return this.View();
        }

        [Authorize(Roles = "Admin,User,User_Test")]
        public ActionResult IndexPolicy()
        {
            // страница с информацией о политике
            return View();
        }

        /// <summary>
        /// страница с тестами
        /// </summary>
        [Authorize(Roles = "Admin,User,User_Test")]
        public ActionResult IndexQuery(string fileName)
        {
            var directoryTests = new DirectoryInfo(PythonTests.TestAcceptedScriptsPath);
            var files = directoryTests.GetFiles("test*.py").OrderBy(x => x.Name, new
NaturalComparer()).Where(x=>!x.Name.Contains("suit")).ToList();
            var db = new TestShowModel();
            var tests = (from test in db.Tests where test.Show == true select
test.Title).ToList();



            if (!User.IsInRole("Admin"))
            {
                var show = false;
                var filesCleared = new List<FileInfo>();

                foreach (var file in files)
                {
                    foreach (var test in tests)
```

```
            {
                if (file.Name == test)
                {
                    show = true;
                }
            }

            if (show == true)
            {
                filesCleared.Add(file);
            }
            show = false;
        }

        files = filesCleared;

    }
    this.ViewBag.Tests = tests;
    this.ViewBag.Files = files;
    this.ViewBag.DirectoryTests = directoryTests;
    this.ViewBag.FileName = fileName;
    return this.View();
}

// хз чо, еще не продумано
[Authorize(Roles = "Admin,User,User_Test")]
public ActionResult IndexLogs()
{
    return this.View();
}

// страница создания скважины пользователем
[Authorize(Roles = "Admin,User,User_Test")]
public ActionResult CreateWell()
{
    return View();
}

// метод создания скважины со стороны пользователя
[HttpPost]
[ValidateAntiForgeryToken]
[Authorize(Roles = "Admin, User, User_Test")]
public async Task<ActionResult> CreateWell(EditWellModel model)
{
    if (this.ModelState.IsValid)
    {
        try
        {
            var db = new WellDbContext();

            var db1 = new FieldGroupModel();
            var field = db1.Field.Single(s => s.Id == model.Id_Field);
            var login = this.User.Identity.GetUserName();
            var guid = Guid.NewGuid().ToString();
            var wellUidExists = db.Well1.Any(x => x.WellUid == guid);

            if (wellUidExists)
            {
                return this.View("ErrorWellUid");
            }

            var well = new WitsmlServer.Web.DbModels.well.Well1
                        {
```

```
                                    WellUid = guid,
                                    Name = model.Name,
                                    // Status = "Active",
                                    Id_User = login,
                                    Id_Field = field.Id,
                                    ActiveTime = model.ActiveTime,
                                    LastTimeUpdate = DateTime.Now,
                                    LastActiveDate =
                                        DateTime.Now.AddDays(model.ActiveTime)
                                };
                    field.Id_Group = model.Id_Group;
                    db1.SaveChanges();

                    var db3 = new UserPolicyDbContext();
                    var query = from policy in db3.Policy
                                from userp in db3.User_Policy
                                where policy.Id == userp.Id_Policy && userp.UserName ==
login
                                select policy.MaxActiveWells;

                    var query1 = from well1 in db.Well1
                                 where well1.Status == "Active" && well1.Id_User == login
                                 select well1;
                    var countWells = query1.Count();

                    if (countWells < query.First())
                    {
                        well.Status = "Active";
                    }
                    else
                    {
                        well.Status = "Inactive";
                    }

                    /* Добавление в базу сервера */
                    int resultCode;
                    string errorMessage;
                    WitsmlService<IInternalService>.Use(
                        service =>
                            {
                                if ((resultCode = service.CreateWell(login, guid,
model.Name, out errorMessage)) != 1)
                                {
                                    throw new Exception(string.Format("Code: {0}. {1}",
resultCode, errorMessage));
                                }
                            });

                    db.Well1.Add(well);
                    db.SaveChanges();
                    return this.RedirectToAction("Index", "PersonalPage", new { page = 1 });
                }
                catch (Exception ee)
                {
                    return this.View("Error");
                }
            }
            else
            {
                // If we got this far, something failed, redisplay form
                return this.View(model);
            }
        }
```

```
        // создание нового месторождения
        [Authorize(Roles = "Admin, User, User_Test")]
        public ActionResult CreateField()
        {
            return this.View();
        }

        // метод создания месторождения
        [HttpPost]
        [ValidateAntiForgeryToken]
        [Authorize(Roles = "Admin, User, User_Test")]
        public async Task<ActionResult> CreateField(Field model)
        {
            if (this.ModelState.IsValid)
            {
                var db = new FieldGroupModel();

                var idNoGroup = db.Group.First(x => x.Name.Contains("NoGroup")).Id;

                var field = new Field
                {
                    Name = model.Name,
                    Id_Group = idNoGroup
                };

                db.Field.Add(field);
                db.SaveChanges();

                return this.RedirectToAction("IndexField", "PersonalPage", new { page = 1
});
            }
            else
            {
                // If we got this far, something failed, redisplay form
                return this.View(model);
            }
        }

        // создание новой группы скважин
        [Authorize(Roles = "Admin, User, User_Test")]
        public ActionResult CreateGroup()
        {
            return this.View();
        }

        // метод создания группы скважин
        [HttpPost]
        [ValidateAntiForgeryToken]
        [Authorize(Roles = "Admin, User, User_Test")]
        public async Task<ActionResult> CreateGroup(Group model)
        {
            if (this.ModelState.IsValid)
            {
                var db = new GroupModel();
                var group = new Group
                {
                    Name = model.Name
                };

                db.Group.Add(group);
                db.SaveChanges();
```

```
                return this.RedirectToAction("IndexGroup", "PersonalPage", new { page = 1
});
            }
            else
            {
                // If we got this far, something failed, redisplay form
                return this.View(model);
            }
        }

        // обновление списка скважин для пересчета дней активности
        public ActionResult RefreshUserWells()
        {
            var db = new FieldGroupModel();
            string login = this.User.Identity.Name;
            var link =
                from well in db.Well1
                where well.Id_User == login
                select well;

            foreach (var w in link)
            {
                if ((w.LastTimeUpdate >= w.LastActiveDate) || (w.ActiveTime == 0))
                {
                    w.Status = "Inactive";
                }
            }

            return this.RedirectToAction("Index", "PersonalPage");
        }

        public ActionResult ShowContent(string fileName)
        {
            try
            {
                var directoryTests = new DirectoryInfo(PythonTests.TestAcceptedScriptsPath);
                var file = directoryTests.GetFiles("test*.py").Where(x =>
x.Name.Contains(fileName)).First();
                var fileResult1 = file.Open(FileMode.Open);
                StreamReader sr2 = new StreamReader(fileResult1);
                String line = sr2.ReadToEnd();
                sr2.Close();
                ViewBag.Content = line;
                ViewBag.FileName = fileName;
                return this.View();
            }
            catch (Exception ee)
            {
                return this.View("Error");
            }
        }
        public ActionResult ShowResult(string fileName)
        {
            try
            {
                var userName = this.User.Identity.Name;
                var pythonTests = new PythonTests(userName);
                var directoryResults = new DirectoryInfo(pythonTests.TestResultsPath);
                var file = (from result in directoryResults.GetFiles()
                            where (result.Name.Contains(fileName) &&
!result.Name.Contains("soap"))
                            select result).First();
                var fileResult1 = file.Open(FileMode.Open);
```

```
                StreamReader sr2 = new StreamReader(fileResult1);
                String line = sr2.ReadToEnd();
                sr2.Close();
                ViewBag.Content = line;
                ViewBag.FileName = fileName;
                return this.View();
            }
            catch (Exception ee)
            {
                return this.View("Error");
            }
        }
    }
}
```

Контроллер «Remove»

```
using System.Linq;
using System.Web.Mvc;
using WitsmlServer.Web.DbModels.Field;
using WitsmlServer.Web.DbModels.Group;
using WitsmlServer.Web.DbModels.Policy;
using WitsmlServer.Web.DbModels.UserPolicy;
using WitsmlServer.Web.DbModels.well;

namespace witsml_app.Controllers
{
    using System;

    using WitsmlServer.Web.DbModels.Comment;
    using WitsmlServer.Web.DbModels.F_G_W_Model;
    using WitsmlServer.Web.DbModels.UserRolePolicy;

    using witsml_app.ServiceClient;

    using Well1 = WitsmlServer.Web.DbModels.well.Well1;

    public class RemoveController : BaseController
    {
        // GET: Remove
        [Authorize(Roles = "Admin")]
        public ActionResult RemoveWell(string id)
        {
            //метод удаления скважины

            var db = new WellDbContext();
            var query =
                from well in db.Well1
                where well.WellUid == id
                select well;
            foreach (var w in query)
            {
                db.Well1.Remove(w);
            }

            var login = query.First().Id_User;

            /* Удаление из базы сервера */
            int resultCode;
            string errorMessage;
            WitsmlService<IInternalService>.Use(
                service =>
```

```
                    {
                        if ((resultCode = service.DeleteWell(login, id, out errorMessage))
!= 1)
                        {
                            throw new Exception(string.Format("Code: {0}. {1}", resultCode,
errorMessage));
                        }
                    });

            db.SaveChanges();

            return RedirectToAction("AdministrationWells", "Admin", new { page = 1 });
        }

        [Authorize(Roles = "Admin, User, User_Test")]
        public ActionResult RemoveActivityWell(string id)
        {
            //метод удаления активного состояния скважины (т.е. сброс активности) со стороны
пользователя

            var db = new WellDbContext();
            var query =
                from well in db.Well1
                where well.WellUid == id
                select well;

            Well1 well1 = new Well1();

            foreach (var w in query)
            {
                db.Well1.Remove(w);
                w.Status = "Inactive";
                w.LastTimeUpdate = w.LastTimeUpdate.Value.AddDays(w.ActiveTime - 1);
                w.LastActiveDate = w.LastTimeUpdate.Value.AddDays(w.ActiveTime);
                well1 = w;
            }
            db.SaveChanges();
            db.Well1.Add(well1);
            db.SaveChanges();

            return RedirectToAction("Index", "PersonalPage");
        }

        [Authorize(Roles = "Admin")]
        public ActionResult RemoveUserWell(string id)
        {
            //удаление скважины пользователем

            var db = new WellDbContext();
            var login = this.User.Identity.Name;
            var query =
                (from well in db.Well1
                where
                    well.WellUid == id &&
                    well.Id_User == login
                select well).First();

            db.Well1.Remove(query);

            /* Удаление из базы сервера */
            int resultCode;
            string errorMessage;
            WitsmlService<IInternalService>.Use(
```

```
                service =>
                    {
                        if ((resultCode = service.DeleteWell(login, id, out errorMessage))
!= 1)
                        {
                            throw new Exception(string.Format("Code: {0}. {1}", resultCode,
errorMessage));
                        }
                    });

            db.SaveChanges();
            return RedirectToAction("Index", "PersonalPage");
        }

        [Authorize(Roles = "Admin")]
        public ActionResult RemoveField(int id)
        {
            //удаление месторождения

            var db = new FieldModel();
            var query =
                from field in db.Field
                where field.Id == id
                select field;
            foreach (var w in query)
            {
                db.Field.Remove(w);
            }
            db.SaveChanges();
            return RedirectToAction("AdministrationFields", "Admin", new { page = 1 });
        }

        [Authorize(Roles = "Admin")]
        public ActionResult RemovePolicy(int id)
        {
            //удаление политики

            var db = new PolicyModel();
            var query =
                from policy in db.Policy
                where policy.Id == id
                select policy;
            foreach (var w in query)
            {
                db.Policy.Remove(w);
            }
            db.SaveChanges();
            return RedirectToAction("AdministrationPolitics", "Admin", new { page = 1 });
        }
        [Authorize(Roles = "Admin")]
        public ActionResult RemoveGroup(int id)
        {
            //удаление группы скважин

            var db = new GroupModel();
            var query =
                from group_ in db.Group
                where group_.Id == id
                select group_;
            foreach (var w in query)
            {
                db.Group.Remove(w);
            }
```

```
            db.SaveChanges();
            return RedirectToAction("AdministrationGroups", "Admin", new { page = 1 });
        }
        [Authorize(Roles = "Admin")]
        public ActionResult RemoveUser(string id)
        {
            //удаление юзера

            var db = new UserPolicyDbContext();
            var users = (from user in db.AspNetUsers where user.Id == id select
user).ToList();
            foreach (var w in users)
            {
                db.AspNetUsers.Remove(w);
            }
            var userName = users.First().UserName;
            var db1 = new UserRolePolicyModel();
            var link = from up in db1.User_Policy from u in db1.AspNetUsers
                       where /*u.UserName == up.UserName &&*/ up.UserName == userName
                       select up;
            foreach (var up in link)
            {
                db1.User_Policy.Remove(up);
            }

            var db2 = new FieldGroupModel();
            var wells = (from w in db2.Well1 where w.Id_User == userName select w);
            foreach (var w in wells)
            {
                db2.Well1.Remove(w);
            }

            db.SaveChanges();
            db1.SaveChanges();
            db2.SaveChanges();
            return RedirectToAction("Administration", "Admin", new { page = 1 });
        }

        [Authorize(Roles = "Admin")]
        public ActionResult RemoveComment(int id)
        {
            var db = new CommentModel();
            Comment comment = db.Comment.First(x => x.Id == id);
            db.Comment.Remove(comment);
            db.SaveChanges();
            return RedirectToAction("Comments", "Home");
        }
    }
}
```

Контроллер «Test»

```
namespace witsml_app.Controllers
{
    using System;
    using System.Diagnostics;
    using System.IO;
    using System.Linq;
    using System.Threading.Tasks;
    using System.Web;
    using System.Web.Mvc;
```

```
    using System.Web.Routing;

    using NLog;

    using WitsmlServer.Web.DbModels.TestModel;

    using witsml_app.PythonTestScripts;

    public class TestController : AsyncController
    {
        private static readonly Logger Logger = LogManager.GetCurrentClassLogger();
        private string executingTestScriptName;

        [AsyncTimeout(6000000)]
        public void ClearResultsAsync()
        {
            var userName = this.User.Identity.Name;
            var processRunning = "ProcessRunning1" + userName;
            var pythonTests = new PythonTests(userName);

            try
            {
                if (this.HttpContext.Application[processRunning] != null)
                {
                    return;
                }

                this.HttpContext.Application[processRunning] = "Run1";
                this.AsyncManager.OutstandingOperations.Increment();
                Task.Factory.StartNew(
                    _ =>
                        {
                            Logger.Trace(string.Format("deleting all test results for user:
{0}", userName));

                            pythonTests.DeleteAllTestResults();

                            Logger.Trace(string.Format("start executing deleteWells.py
script for user: {0}", userName));
                            var pi = new ProcessStartInfo
                                {
                                    WorkingDirectory = pythonTests.WorkingDirectory,
                                    FileName = pythonTests.PythonInterpreter,
                                    Arguments = pythonTests.DeleteScriptFullPath
                                };

                            var process = Process.Start(pi);
                            if (process == null)
                            {
                                Logger.Error("Can't start process Python27.exe with
deleteWells.py script");
                                return;
                            }

                            process.WaitForExit();
                            Logger.Trace(string.Format("deleteWells.py successfully executed
for user: {0}", userName));
                        },
                    null).ContinueWith(
                        _ =>
                            {
                                Logger.Trace(string.Format("start executing load_data_set.py
script for user: {0}", userName));
```

```
                                var pi = new ProcessStartInfo
                                    {
                                        WorkingDirectory = pythonTests.WorkingDirectory,
                                        FileName = pythonTests.PythonInterpreter,
                                        Arguments = pythonTests.LoadDataScriptFullPath
                                    };
                                pi.CreateNoWindow = true;
                                pi.WindowStyle = ProcessWindowStyle.Hidden;
                                var process = Process.Start(pi);
                                if (process == null)
                                {
                                    Logger.Error("Can't start process Python27.exe with
load_data_set.py script");
                                }
                                else
                                {
                                    process.WaitForExit();
                                }
                                Logger.Trace(string.Format("load_data_set.py successfully
executed for user: {0}", userName));

                                this.AsyncManager.OutstandingOperations.Decrement();
                                this.HttpContext.Application[processRunning] = null;

                            });
                }
                catch (Exception ex)
                {
                    Logger.Error(ex);
                    throw;
                }
            }

            // выполнение теста
            [AsyncTimeout(6000000)]
            public void SomeTaskAsync(int id, string fileName)
            {
                this.executingTestScriptName = fileName;
                var processRuning = "ProcessRunning" + this.User.Identity.Name;
                var userName = this.User.Identity.Name;
                var pythonTests = new PythonTests(userName);
                this.HttpContext.Application[processRuning] = null;

                if (this.HttpContext.Application[processRuning] != null)
                {
                    return;
                }

                this.HttpContext.Application[processRuning] = "Run";
                this.AsyncManager.OutstandingOperations.Increment();
                Task.Factory.StartNew(
                    _ =>
                        {
                            var pi = new ProcessStartInfo
                                {
                                    CreateNoWindow = true,
                                    WorkingDirectory = pythonTests.WorkingDirectory,
                                    FileName = pythonTests.PythonInterpreter,
                                    Arguments = pythonTests.TestScriptFullPath(fileName)
                                };

                            var process = Process.Start(pi);
```

```
                        if (process == null)
                            Logger.Error(string.Format("Can't execute {0} script",
fileName));
                        else
                            process.WaitForExit();

                        this.HttpContext.Application[processRuning] = null;

                        this.AsyncManager.OutstandingOperations.Decrement();
                    },
                null);
        }

        [AsyncTimeout(6000000)]
        public void ExecuteAllTestsAsync()
        {
            var userName = this.User.Identity.Name;
            var directoryTests = new DirectoryInfo(PythonTests.TestAcceptedScriptsPath);
            var files = directoryTests.GetFiles("test*.py").Where(x =>
x.Name.Contains("test") && !x.Name.Contains("suit")).OrderBy(f => f.Name);
            var pythonTests = new PythonTests(userName);

            foreach (var test in files)
            {
                this.AsyncManager.OutstandingOperations.Increment();
                Task.Factory.StartNew(
                    _ =>
                        {
                            var pi = new ProcessStartInfo
                            {
                                WorkingDirectory = pythonTests.WorkingDirectory,
                                FileName = pythonTests.PythonInterpreter,
                                Arguments = pythonTests.TestScriptFullPath(test.Name)
                            };

                            var process = Process.Start(pi);
                            if (process == null)
                            {
                                Logger.Error(string.Format("Can't execute {0} script",
test.Name));
                            }
                            else
                            {
                                process.WaitForExit();
                            }

                            this.AsyncManager.OutstandingOperations.Decrement();
                        },
                    null);
            }
            var a = 1;
        }
        public ActionResult ClearResultsCompleted()
        {

            return this.RedirectToAction("IndexQuery", "PersonalPage");
        }

        public ActionResult SomeTaskCompleted(string result)
        {
            return this.RedirectToAction("IndexQuery", "PersonalPage", new { fileName =
this.executingTestScriptName });
        }
```

```
        public ActionResult ExecuteAllTestsCompleted()
        {
            return this.RedirectToAction("IndexQuery", "PersonalPage");
        }

        public ActionResult ExecuteAllTestsProgress()
        {
            return Json(new
                            {
                                Progress = HttpContext.Application["Test"]
                            }, JsonRequestBehavior.AllowGet);
        }

        public void ShowTest(string fileName)
        {
            var db = new TestShowModel();

            var test = (from tests in db.Tests
                        where tests.Title == fileName
                        select tests).First();

            if (test.Show == true)
            {
                test.Show = false;
            }
            else
            {
                test.Show = true;
            }
            db.SaveChangesAsync();
        }

    }
}
```

Приложение 2. Программный код представлений

Представление «Login»

```
@using LangResources
@using witsml_app.Models
@model LoginViewModel
@{
    ViewBag.Title = Resources.Login;
}

<h2>@ViewBag.Title</h2>
<div class="row">
    <div class="col-md-8">
        <section id="loginForm">
            @using (Html.BeginForm("Login", "Account", new { ReturnUrl = ViewBag.ReturnUrl
}, FormMethod.Post, new { @class = "form-horizontal", role = "form" }))
            {
                @Html.AntiForgeryToken()
                <h4>@Resources.UseLoginAndPass</h4>
                <hr />
                @Html.ValidationSummary(true, "", new { @class = "text-danger" })
                <div class="form-group">
                    @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label" })
                    <div class="col-md-10">
```

```
                        @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })
                        @Html.ValidationMessageFor(m => m.Email, "", new { @class = "text-
danger" })
                    </div>
                </div>
                <div class="form-group">
                    @Html.LabelFor(m => m.Password, new { @class = "col-md-2 control-label"
})
                    <div class="col-md-10">
                        @Html.PasswordFor(m => m.Password, new { @class = "form-control" })
                        @Html.ValidationMessageFor(m => m.Password, "", new { @class =
"text-danger" })
                    </div>
                </div>
                <div class="form-group">
                    <div class="col-md-offset-2 col-md-10">
                        <div class="checkbox">
                            @Html.CheckBoxFor(m => m.RememberMe)
                            @Html.LabelFor(m => m.RememberMe)
                        </div>
                    </div>
                </div>
                <div class="form-group">
                    <div class="col-md-offset-2 col-md-10">
                        <input type="submit" value="@Resources1.Login" class="btn btn-
primary" />
                    </div>
                </div>
                <div class="form-group">
                    <div class="col-md-offset-2 col-md-10">
                        <p>
                            @Html.ActionLink(Resources1.Register, "Register", null, new {
@style = "font-size: 20px" })
                        </p>
                    </div>
                    </div>

            }
</section>
    </div>
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Представление «Register»

```
@using LangResources
@model WitsmlServer.Web.DbModels.AddModel.AddUserModel
@{
    ViewBag.Title = Resources1.Register;
}

<h2>@ViewBag.Title</h2>

@using (Html.BeginForm("Register", "Account", FormMethod.Post, new { @class = "form-
horizontal", role = "form" }))
{
    @Html.AntiForgeryToken()
    <h4>@Resources1.Register</h4>
    <hr />
```

```
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.Login, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Login, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.Password, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.Password, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.ConfirmPassword, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.PasswordFor(m => m.ConfirmPassword, new { @class = "form-control" })
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-primary" value="@Resources1.Register" />
        </div>
    </div>

}
@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Представление «Administration»

```
@using System.Web.UI.WebControls
@using LangResources
@using PagedList
@using PagedList.Mvc
@using WitsmlServer.Web.DbModels.UserRolePolicy

@{
    ViewBag.Title = Resources1.ManagingUsers;
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>@ViewBag.Title</h2>

<ul class="nav nav-tabs">
    <li role="presentation" class="active">@Html.ActionLink(Resources1.ManagingUsers,
"Administration", "Admin", new { page = 1 }, null)</li>
    <li role="presentation">@Html.ActionLink(Resources1.ManagingPolitics,
"AdministrationPolitics", "Admin", new { page = 1 }, null)</li>
    <li role="presentation">@Html.ActionLink(Resources1.ManagingWells,
"AdministrationWells", "Admin", new { page = 1 }, null)</li>
    <li role="presentation">@Html.ActionLink(Resources1.ManagingGroups,
"AdministrationGroups", "Admin", new { page = 1 }, null)</li>
    <li role="presentation">@Html.ActionLink(Resources1.ManagingFields,
"AdministrationFields", "Admin", new { page = 1 }, null)</li>
</ul>


<table>
    <tr>
        <td>
```

```
            <ul class="nav nav-pills">
                <li role="presentation">@Html.ActionLink(Resources1.CreateUser,
"AddNewUser", "Admin", new { @style = "font-size: 16px" })</li>
            </ul>
        </td>
        <td>
            @using (Html.BeginForm(Url.Action("Administration", new { page =
ViewBag.PageNumber, searchString = ViewBag.SearchString, pageSize = ViewBag.PageSize })))
            {
                <table>
                    <tr>
                        <td>
                            <input type="text" class="form-control" placeholder="Search..."
name="SearchString" value=@ViewBag.SearchString>
                            <input type="hidden" name="PageSize" value=@ViewBag.PageSize
min="1">
                        </td>
                        <td style="padding-left:10px">
                            <button class="btn btn-primary"
type="submit">@Resources1.Search</button>
                        </td>
                    </tr>
                </table>
            }
        </td>
        <td style="padding-left: 5px">
            @using (Html.BeginForm(Url.Action("Administration", new { page =
ViewBag.PageNumber, searchString = ViewBag.SearchString, pageSize = ViewBag.PageSize })))
            {
                <table>
                    <tr>
                        <td>
                            <input type="number" class="form-control" name="PageSize"
value=@ViewBag.PageSize min="1" style="width: 90px">
                            <input type="hidden" name="SearchString"
value=@ViewBag.SearchString>
                        </td>
                        <td style="padding-left:10px">
                            <button class="btn btn-primary"
type="submit">@Resources1.Display</button>
                        </td>
                    </tr>
                </table>
            }
        </td>
    </tr>
</table>

<div class="panel-body" style="padding-left: 10px; padding-top: 15px; width: 100%">
    <table class="table table-bordered" style="background-color: whitesmoke">
        <tr>
            <td style="padding: 10px">
                <b> @Resources1.Number </b>
            </td>
            <td style="padding: 10px">
                <b>@Html.ActionLink(Resources1.LoginModel,"Administration","Admin", new
{sortBy = "sortByUserName"},null)</b>
            </td>
            <td style="padding: 10px">
                <b>@Resources1.Role</b>
            </td>
            <td style="padding: 10px">
```

```
                <b>@Html.ActionLink(Resources1.Policy, "Administration", "Admin", new {
sortBy = "sortByPolicy" }, null)</b>
            </td>
            <td style="padding: 10px">
                <b>@Html.ActionLink(Resources1.DateBegin, "Administration", "Admin", new {
sortBy = "sortByDateBegin" }, null)</b>
            </td>
            <td style="padding: 10px">
                <b>  @Resources1.EndDate</b>
            </td>
            <td style="padding: 10px">
                <b> @Resources1.DaysLeft</b>
            </td>
            <td style="padding: 10px">
                <b> Add</b>
            </td>
            <td style="padding: 10px">
                <b> Get</b>
            </td>
            <td style="padding: 10px">
                <b> Update</b>
            </td>
            <td style="padding: 10px">
                <b> Delete</b>
            </td>
            <td style="padding: 10px">
                <b> @Resources1.Delete</b>
            </td>
        </tr>
        @{
                int count = 1;
                int days = 0;
                string color1 = "whitesmoke";
                int page = ViewBag.PageNumber;
                string searchString = ViewBag.SearchString;
                int pagesize = ViewBag.PageSize;
            string sortBy = ViewBag.SortBy;

                var db = new UserRolePolicyModel();
                var query = from user_p in db.User_Policy
                            from user in db.AspNetUsers
                            from policy in db.Policy
                            where user_p.UserName == user.UserName && policy.Id ==
user_p.Policy.Id && (user.UserName.Contains(searchString) ||
user.UserName.ToUpper().Contains(searchString.ToUpper()))
                            orderby user_p.Start_Date descending
                            select new { user, policy, user_p };

            var users = query.ToPagedList(page, pagesize);
            if (sortBy == "sortByUserName")
            {
                users = query.OrderBy(x => x.user.UserName).ToPagedList(page, pagesize);
            }
                else
                    if (sortBy == "sortByPolicy")
                    {
                        users = query.OrderBy(x => x.user_p.Policy.Name).ToPagedList(page,
pagesize);
                    }
            else
            {
                users = query.OrderByDescending(x => x.user_p.Start_Date).ToPagedList(page,
pagesize);
```

```
            }


                foreach (var u in users)
                {
                    if (count % 2 == 0)
                    {
                        color1 = "whitesmoke";
                    }
                    else
                    {
                        color1 = "#ddffee";
                    }
                    <tr bgcolor=@color1>
                        <td style="padding: 10px">
                            @(count + pagesize * (page - 1))
                        </td>
                        <td style="padding: 10px">
                            @LinkExtensions.ActionLink(Html, u.user.UserName, "EditUser",
"Edit", new { userName = u.user.UserName }, null)
                        </td>

                        <td style="padding: 10px">
                            @u.user.AspNetRoles.First().Name
                        </td>
                        <td style="padding: 10px">
                            @u.policy.Name
                        </td>
                        <td style="padding: 10px">
                            @u.user_p.Start_Date.Date.ToShortDateString()
                        </td>
                        <td style="padding: 10px">

@u.user_p.Start_Date.Date.AddDays(u.policy.Duration).ToShortDateString()
                        </td>
                        <td style="padding: 10px">
                            @{
                                days =
Convert.ToInt32((u.user_p.Start_Date.Date.AddDays(u.policy.Duration) - DateTime.Now).Days);
                                if (days < 0)
                                {
                                    days = 0;
                                }
                            }
                            @days
                        </td>
                        <td style="padding: 10px">
                            @if (u.policy.Add == true)
                            {
                                <b>@u.policy.Add</b>
                            }
                            else
                            {
                                @u.policy.Add
                            }
                        </td>
                        <td style="padding: 10px">
                            @if (u.policy.Get == true)
                            {
                                <b>@u.policy.Get</b>
                            }
                            else
                            {
```

```
                                @u.policy.Get
                            }
                        </td>
                        <td style="padding: 10px">
                            @if (u.policy.Update == true)
                            {
                                <b>@u.policy.Update</b>
                            }
                            else
                            {
                                @u.policy.Update
                            }
                        </td>
                        <td style="padding: 10px">
                            @if (u.policy.Delete == true)
                            {
                                <b>@u.policy.Delete</b>
                            }
                            else
                            {
                                @u.policy.Delete
                            }
                        </td>
                        <td style="padding: 10px">
                            @if (u.user.AspNetRoles.First().Name != "Admin")
                            {
                                <div onclick=" return confirm('Confirm deleting?', null) ">
                                    @Html.ActionLink(Resources1.Delete, "RemoveUser",
"Remove", new { id = u.user.Id }, null)
                                </div>
                            }
                        </td>
                    </tr>
                    count++;
                }
        }
    </table>
    @Html.PagedListPager((IPagedList)users, page_ => Url.Action("Administration", new { page
= page_, searchString = ViewBag.SearchString, pageSize = ViewBag.PageSize }))
</div>
```

Представление «Administration Wells»

```
@using System.Globalization
@using LangResources
@using PagedList
@using PagedList.Mvc
@using WitsmlServer.Web.DbModels.F_G_W_Model
@{
    ViewBag.Title = Resources1.ManagingWells;
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>@ViewBag.Title</h2>

<ul class="nav nav-tabs">
    <li role="presentation">@Html.ActionLink(Resources1.ManagingUsers, "Administration",
"Admin", new { page = 1 }, null)</li>
    <li role="presentation">@Html.ActionLink(Resources1.ManagingPolitics,
"AdministrationPolitics", "Admin", new { page = 1 }, null)</li>
    <li role="presentation" class="active">@Html.ActionLink(Resources1.ManagingWells,
"AdministrationWells", "Admin", new { page = 1 }, null)</li>
```

```
    <li role="presentation">@Html.ActionLink(Resources1.ManagingGroups,
"AdministrationGroups", "Admin", new { page = 1 }, null)</li>
    <li role="presentation">@Html.ActionLink(Resources1.ManagingFields,
"AdministrationFields", "Admin", new { page = 1 }, null)</li>
</ul>


<table>
    <tr>
        <td>
            <ul class="nav nav-pills">
                <li role="presentation">@Html.ActionLink(Resources1.CreateWell,
"AddNewWell", "Admin", new { @style = "font-size: 16px", })</li>
            </ul>
        </td>
        <td>
            @using (Html.BeginForm(Url.Action("AdministrationWells", new { page =
ViewBag.PageNumber, searchString = ViewBag.SearchString, pageSize = ViewBag.PageSize })))
            {
                <table>
                    <tr>
                        <td>
                            <input type="text" class="form-control" placeholder="Search..."
name="SearchString" value=@ViewBag.SearchString>
                            <input type="hidden" name="PageSize" value=@ViewBag.PageSize
min="1">
                        </td>
                        <td style="padding-left:10px">
                            <button class="btn btn-primary"
type="submit">@Resources1.Search</button>
                        </td>
                    </tr>
                </table>
            }
        </td>
        <td style="padding-left: 5px">
            @using (Html.BeginForm(Url.Action("AdministrationWells", new { page =
ViewBag.PageNumber, searchString = ViewBag.SearchString, pageSize = ViewBag.PageSize })))
            {
                <table>
                    <tr>
                        <td>
                            <input type="number" class="form-control" name="PageSize"
value=@ViewBag.PageSize min="1">
                            <input type="hidden" name="SearchString"
value=@ViewBag.SearchString>
                        </td>
                        <td style="padding-left:10px">
                            <button class="btn btn-primary"
type="submit">@Resources1.Display</button>
                        </td>
                    </tr>
                </table>
            }
        </td>
        <td>
            <ul class="nav nav-pills">
                <li role="presentation">@Html.ActionLink(Resources1.Refresh, "RefreshWells",
"Admin", new { @style = "font-size: 16px" })</li>
            </ul>
        </td>
    </tr>
</table>
```

```
<div class="panel-body" style="padding-left: 10px; padding-top: 15px; width: 100%">

    <table class="table table-bordered" style="background-color: whitesmoke">
        <tr>
            <td style="padding: 10px">
                <b>UID</b>
            </td>
            <td style="padding: 10px">
                <b> @Resources1.Title</b>
            </td>
            <td style="padding: 10px">
                <b> @Resources1.Status</b>
            </td>
            <td style="padding: 10px">
                <b> @Resources1.User</b>
            </td>
            <td style="padding: 10px">
                <b>@Resources1.Field</b>
            </td>
            <td style="padding: 10px">
                <b>@Resources1.Group</b>
            </td>
            <td style="padding: 10px">
                <b>@Resources1.LastUpdateTime</b>
            </td>
            <td style="padding: 10px">
                <b>@Resources1.DeactivatingDaysLeft</b>
            </td>
            <td style="padding: 10px">
                <b>@Resources1.DeactivatingDate</b>
            </td>

            <td style="padding: 10px">
                <b> @Resources1.Delete</b>
            </td>
        </tr>
        @{
            int count = 1;
            string color1 = "whitesmoke";
            string searchString = ViewBag.SearchString;

            int pagenumber = ViewBag.PageNumber;
            int pagesize = ViewBag.PageSize;

            var db = new FieldGroupModel();
            var query =
                from well in db.Well1
                from field in db.Field
                from _group in db.Group
                where
                    (field.Id == well.Id_Field)
                    &&
                    (_group.Id == field.Id_Group)
                    &&
                    (well.Name.ToUpper().Contains(searchString.ToUpper()) ||
well.Name.Contains(searchString))
                orderby well.LastTimeUpdate descending
                select new
                {
                    well,
```

```
                    field,
                    _group
                };

            var well_Page = query.ToPagedList(pagenumber, pagesize);


            foreach (var w in well_Page)
            {
                if (count % 2 == 0)
                {
                    color1 = "whitesmoke";
                }
                else
                {
                    color1 = "#ddffee";
                }
                <tr bgcolor=@color1>
                    <td style="padding: 10px">
                        @w.well.WellUid
                    </td>
                    <td style="padding: 10px">
                        @LinkExtensions.ActionLink(Html, w.well.Name, "EditWell", "Edit",
new { id = w.well.WellUid }, null)
                    </td>
                    <td style="padding: 10px">
                        @w.well.Status
                    </td>
                    <td style="padding: 10px">
                        @w.well.Id_User
                    </td>
                    <td style="padding: 10px">
                        @w.field.Name
                    </td>
                    <td style="padding: 10px">
                        @w._group.Name
                    </td>
                    <td style="padding: 10px">
                        @try
                        {
                            @w.well.LastTimeUpdate.Value.ToString("dd.MM.yyyy HH:mm",
CultureInfo.InvariantCulture)//.Value//.ToShortDateString()
                        }
                        catch (Exception ee)
                        { }
                    </td>
                    <td style="padding: 10px">
                        @try
                        {
                            @((w.well.LastActiveDate - DateTime.Now).Value.Days)
                        }
                        catch (Exception ee)
                        { }
                    </td>
                    <td style="padding: 10px">
                        @try
                        {
                            @w.well.LastActiveDate.Value.ToString("dd.MM.yyyy HH:mm",
CultureInfo.InvariantCulture)//.Value.ToShortDateString()
                        }
                        catch (Exception ee)
                        { }
```

```
                    </td>
                    <td style="padding: 10px">
                        <div onclick=" return confirm('Confirm removing?', null) ">
                            @Html.ActionLink(Resources1.Delete, "RemoveWell", "Remove", new
{ id = w.well.WellUid }, null)
                        </div>
                    </td>
                </tr>
                        count++;
            }
        }
    </table>

    @Html.PagedListPager((IPagedList)well_Page, page_ => Url.Action("AdministrationWells",
new { page = page_, searchString = ViewBag.SearchString, pageSize = ViewBag.PageSize }))

</div>
```

Представление «Edit Well»

```
@using System.Globalization
@using LangResources
@using WitsmlServer.Web.DbModels.Field
@using WitsmlServer.Web.DbModels.UserPolicy
@model WitsmlServer.Web.DbModels.F_G_W_Model.EditWellModel
@{
    ViewBag.Title = Resources1.EditWell;
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>@ViewBag.Title</h2>

<ol class="breadcrumb">
    <li><a href="@Url.Action("Index", "Home")">@Resources1.HomePageButton</a></li>
    <li><a href="@Url.Action("AdministrationWells", "Admin", new { page = 1 },
null)">@Resources1.ManagingWells</a></li>
    <li class="active">@ViewBag.Title</li>
</ol>

@using (Html.BeginForm("EditWell", "Edit", FormMethod.Post, new { @class = "form-
horizontal", role = "form" }))
{
    @Html.AntiForgeryToken()
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })

    @Html.HiddenFor(m => m.HName, new { @class = "form-control" })

    @Html.HiddenFor(m => m.LastTimeUpdate, new { @class = "form-control" })
    @Html.HiddenFor(m => m.ActiveTime)
    @Html.HiddenFor(m => m.Status)
    @Html.HiddenFor(m=>m.WellUid)


    <div class="form-group">
        @Html.LabelFor(m => m.Name, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Name, new { @class = "form-control" })
        </div>
    </div>
```

```
    <div class="form-group">
        @Html.LabelFor(m => m.Status, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @{
                @Html.Label(Model.Status, new { @class = "control-label" })
            }
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(m => m.LastTimeUpdate, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">

            @if (Model.LastTimeUpdate != null)
            {
                var date = Model.LastTimeUpdate.Value;

                var day = date.Day.ToString();
                var month = date.Month.ToString();
                var year = date.Year.ToString();

                var hour = date.Hour.ToString();
                var minute = date.Minute.ToString();

                if ((day == "1") || (day == "2") || (day == "3") || (day == "4") || (day ==
"5") || (day == "6") || (day == "7") || (day == "8") || (day == "9"))
                {
                    day = "0" + day;
                }
                if ((month == "1") || (month == "2") || (month == "3") || (month == "4") ||
(month == "5") || (month == "6") || (month == "7") || (month == "8") || (month == "9"))
                {
                    month = "0" + month;
                }
                if ((minute == "1") || (minute == "2") || (minute == "3") || (minute == "4")
|| (minute == "5") || (minute == "6") || (minute == "7") || (minute == "8") || (minute ==
"9"))
                {
                    minute = "0" + minute;
                }

                <b>
                    <p>Дата: @day.@month.@year</p>
                    <p>Время: @hour:@minute</p>
                </b>
            }

        </div>
    </div>

    <table border="0" style="margin-left: 00px" width="600">
        <tr>
            <td colspan="1" style="width:195px;padding-left: 60px">
                @Html.LabelFor(m => m.ActiveTime, new { @class = "col-md-2 control-label" })
            </td>

            <td>

                <div class="form-group">
                    @Html.LabelFor(m => m.Years, new { @class = "col-md-2 control-label" })
                    <div class="col-md-10">
                        @{
```

```
                            @Html.TextBoxFor(m => m.Years, new { @class = "form-control",
@style = "width: 80px", Value = new
DateTime(TimeSpan.FromDays(Model.ActiveTime).Ticks).Year-1 })
                        }
                    </div>
                </div>
            </td>
            <td>
                <div class="form-group">
                    @Html.LabelFor(m => m.Months, new { @class = "col-md-2 control-label" })
                    <div class="col-md-10">
                        @{
                            @Html.TextBoxFor(m => m.Months, new { @class = "form-control",
@style = "width: 80px", Value = new
DateTime(TimeSpan.FromDays(Model.ActiveTime).Ticks).Month-1 })
                        }
                    </div>
                </div>
            </td>
            <td>
                <div class="form-group">
                    @Html.LabelFor(m => m.Days, new { @class = "col-md-2 control-label" })
                    <div class="col-md-10">
                        @{
                            @Html.TextBoxFor(m => m.Days, new { @class = "form-control",
@style = "width: 80px", Value = new DateTime(TimeSpan.FromDays(Model.ActiveTime).Ticks).Day-
1 })
                        }
                    </div>
                </div>
            </td>
            <td>
                <div class="form-group">
                    @Html.LabelFor(m => m.Hours, new { @class = "col-md-2 control-label" })
                    <div class="col-md-10">
                        @{
                            @Html.TextBoxFor(m => m.Hours, new { @class = "form-control",
@style = "width: 80px", Value = new DateTime(TimeSpan.FromDays(Model.ActiveTime).Ticks).Hour
})
                        }
                    </div>
                </div>
            </td>

        </tr>
    </table>

    <div class="form-group">
        @Html.LabelFor(m => m.Id_User, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @{
                var db = new UserPolicyDbContext();
                var query =
                    from user in db.AspNetUsers
                    select user;
                List<string> list = new List<string>();
                foreach (var e in query)
                {
                    list.Add(e.UserName);
                }
                SelectList select_list = new SelectList(list);
                @Html.DropDownListFor(m => m.Id_User, select_list, Resources1.SelectUser,
new { @class = "form-control" })
```

```
            }
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(m => m.Id_Field, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @{
                var db2 = new FieldModel();
                var query2 =
                    from field in db2.Field
                    select field;
                List<Field> list2 = new List<Field>();
                foreach (var e in query2)
                {
                    list2.Add(e);
                }
                SelectList select_list2 = new SelectList(list2,"Id","Name");
                @Html.DropDownListFor(m => m.Id_Field, select_list2, Resources1.SelectField,
new { @class = "form-control" })
            }
        </div>
    </div>


    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" value="@Resources1.SaveChanges" class="btn btn-primary" />
        </div>
    </div>

}
@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Представление «Index»

```
@using LangResources

@{
    ViewBag.Title = "TPU Witsml Server 1.4.1";
}



<center>


            <div id="carousel-example-generic" class="carousel slide" data-ride="carousel"
style="width: 970px">

                <!-- Wrapper for slides -->
                <div class="carousel-inner" role="listbox">
                    <div class="item active">
                        <img src="~/images/25-0g010.jpg" class="img-thumbnail">

                        <div class="carousel-caption">
                            <!--some text--><span style="background-color: white;
color:black">  @Resources1.Well  </span>
                        </div>
                    </div>
```

```
                    <div class="item">
                        <img src="~/images/IMGP1413.jpg" class="img-thumbnail">
                        <div class="carousel-caption">
                            <!--some text--><span style="background-color: white;
color:black">  @Resources1.VankorSwamp  </span>
                        </div>
                    </div>
                    <div class="item">
                        <img src="~/images/IMGP1264.JPG" class="img-thumbnail">
                        <div class="carousel-caption">
                            <!--some text--><span style="background-color: white;
color:black;">  @Resources1.DataProcessingCenter  </span>
                        </div>
                    </div>
                </div>
            </div>
        </center>

        <br/>

        <table border="0">
            <tr>
                <td style="width: 150px">
                    <img src="~/images/39660.jpeg" width="150px">
                </td>
                <td style="font-size: 18px">
                   @Html.Raw(Resources1.MainText)
                </td>
                <td style="width: 150px">
                    <img src="~/images/i2LpcVcDEEE.jpg" width="100">
                </td>
            </tr>
        </table>
@{
    Html.RenderPartial("_HelpMessage");
}
```

Представление «_Layout»

```
@using LangResources
<!DOCTYPE html>
<html style="height: 100vh">
    <head>
        <meta charset="utf-8"/>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <meta name="description" Content="Российский WITSML сервер, разработанный в Томском
Политехническом Университете, реализует версию 1.4.1 стандарта WITSML.">
        <meta name="keywords" Content="witsml, витсмл, tpu, тпу, tomsk, томск, 1.4.1, 141,
server, сервер, sdk, россия, russia, разработка, программа, application, product, продукт,
купить, buy, price, цена">
        <meta name="Publisher-Email" Content="marchukovav@tpu.ru">
        <meta name="robots" content="All">

        <link rel="shortcut icon" href="/images/i2LpcVcDEEE.jpg" type="image/jpg">
        <title>@ViewBag.Title</title>


        <!-- Google Analytics -->
        <script>
        (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
```

```
                    (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new
Date();a=s.createElement(o),

m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
        })(window,document,'script','//www.google-analytics.com/analytics.js','ga');
        ga('create', 'UA-72269001-1', 'auto');
        ga('send', 'pageview');
        </script>
        <!-- Google Analytics -->

        <!-- Yandex.Metrika counter -->
        <script src="https://mc.yandex.ru/metrika/watch.js" type="text/javascript"></script>
        <script type="text/javascript">
            try {
                var yaCounter34636980 = new Ya.Metrika({
                    id:34636980,
                    clickmap:true,
                    trackLinks:true,
                    accurateTrackBounce:true,
                    webvisor:true,
                    trackHash:true
                });
            } catch(e) { }
        </script>
        <noscript><div><img src="https://mc.yandex.ru/watch/34636980"
style="position:absolute; left:-9999px;" alt="" /></div></noscript>
        <!-- /Yandex.Metrika counter -->

        <!-- Piwik -->
        <script type="text/javascript">
            var _paq = _paq || [];
            _paq.push(['trackPageView']);
            _paq.push(['enableLinkTracking']);
            (function() {
                var u="//witsml.tpu.ru/analytics/";
                _paq.push(['setTrackerUrl', u+'piwik.php']);
                _paq.push(['setSiteId', 1]);
                var d=document, g=d.createElement('script'),
s=d.getElementsByTagName('script')[0];
                g.type='text/javascript'; g.async=true; g.defer=true; g.src=u+'piwik.js';
s.parentNode.insertBefore(g,s);
            })();
        </script>
        <noscript><p><img src="//witsml.tpu.ru/analytics/piwik.php?idsite=1"
style="border:0;" alt="" /></p></noscript>
        <!-- End Piwik Code -->

        <link href="/Content/colour-blue.css" rel="stylesheet" />
        <link href="/Content/custom-style.css" rel="stylesheet" />
        <link href="/Content/font-awesome.css" rel="stylesheet" />
        <link href="/Content/font-awesome.min.css" rel="stylesheet" />
        <link href="/Content/theme-style.css" rel="stylesheet" />
        <link href="/Content/theme-style.min.css" rel="stylesheet" />
        <link href="/Content/bootstrap.css" rel="stylesheet" />
        <link href="/Content/Site.css" rel="stylesheet" />

        <script src="/Scripts/jquery.js"></script>
        <script src="/Scripts/jquery.js"></script>
        <script src="/Scripts/jquery-1.10.2.js"></script>
        <script src="/Scripts/jquery.validate.js"></script>
        <script src="/Scripts/jquery.validate.unobtrusive.js"></script>
        <script src="/Scripts/modernizr-2.6.2.js"></script>
        <script src="/Scripts/respond.js"></script>
```

```
        <script src="/Scripts/script.js"></script>
        <script src="/Scripts/_references.js"></script>
        <script src="/Scripts/bootstrap.js"></script>
        <script src="~/Scripts/bootstrap.js"></script>
        <script src="/Scripts/highlight.pack.js"></script>
        <script src="/Scripts/script.js"></script>
        <script src="/Scripts/highlight.pack.js"></script>
        <script>hljs.initHighlightingOnLoad();</script>
        <link rel="stylesheet" href="/Content/styles/vs.css">
        <link rel="stylesheet" href="/Content/bootstrap.css">

    </head>
@{
    var culture =
System.Threading.Thread.CurrentThread.CurrentUICulture.Name.ToLowerInvariant();
}

@helper selected(string c, string culture)
{
    if (c == culture)
    {
        @:checked="checked"
    }
}
    <body style="font-family: 'Calibri'; font-size: 16px">
    <div class="wrapper">
        <!--Header upper region-->
        <div class="header-upper navbar-static-top js-clingify-wrapper">
            <div class="header-upper-inner container">
                <div class="row">
                    <div class="col-xs-8 col-xs-push-4">
                    </div>
                    <div class="col-xs-4 col-xs-pull-8">

                        <!--user menu-->
                        <div class="btn-group user-menu">
                            @Html.Partial("_LoginPartial")
                        </div>
                        @using (Html.BeginForm("SetCulture", "Home"))
                        {
                            <div class="btn-group language-menu" style="border: 2px solid
black; background-color: white">
                                <center>
                                    <!--language menu-->
                                    <a href="" class="btn btn-link dropdown-toggle" data-
toggle="dropdown"><span class="flag-icon flag-icon-gb" style="background-color: white;
color: black; width:110%">@Resources1.Language</span></a>

                                    <ul class="dropdown-menu dropdown-menu-mini dropdown-
menu-primary" style="background-color: black; width:70px">
                                        <li style="background-color: white;border: 1px solid
black;">
                                             <input name="culture" id="ru" value="ru"
type="radio" @selected("ru", culture) /> Russian
                                        </li>
                                        <li style="background-color: white;border: 1px solid
black;">
                                             <input name="culture" id="en" value="en"
type="radio" @selected("en", culture) /> English
                                        </li>
                                    </ul>
                                </center>
                            </div>
```

```
                    }
                </div>
            </div>
        </div>
    </div>

    <!--Header & Branding region-->
    <div class="header" data-toggle="clingify">
        <div class="header-inner container">
            <div class="navbar">
                <div class="pull-left">
                    <!--branding/logo-->
                    <a class="navbar-brand" href="@Url.Action("Index", "Home")"
title="Home">
                        <h1>
                            <!-- !!! Оставить пробел между словами и тегом "..span>
Witsml.." Игаче в гугле выглядит криво !!! -->
                            <span>TPU</span> Witsml Server<span></span>
                            <!-- !!! Оставить пробел между словами и тегом "..span>
Witsml.." Игаче в гугле выглядит криво !!! -->
                        </h1>
                    </a>
                </div>
                <!--everything within this div is collapsed on mobile-->
                <div class="navbar-collapse collapse">
                    <!--main navigation-->
                   <ul class="nav navbar-nav" style="font-size: 20px">
                        <li class="dropdown">
                            @Html.ActionLink(Resources1.HomePageButton, "Index", "Home",
null, new { @style = "font-size: 13px" })
                        </li>

                        <li class="dropdown">
                            @Html.ActionLink(Resources1.TarifsButton, "About", "Home",
null, new { @style = "font-size: 13px" })
                        </li>

                        <li class="dropdown">
                            @Html.ActionLink(Resources1.ContactsButton, "Contact",
"Home", null, new { @style = "font-size: 13px" })
                        </li>
                        @{
                            if (User.IsInRole("Admin"))
                            {
                                <li
role="presentation">@Html.ActionLink(Resources1.AdministrationButton, "Administration",
"Admin", new { page = 1 }, new { @style = "font-size: 13px" })</li>
                            }
                            if (User.IsInRole("Admin") || User.IsInRole("User") ||
User.IsInRole("User_Test"))
                            {
                                <li
role="presentation">@Html.ActionLink(Resources1.PersonalAreaButton, "Index", "PersonalPage",
new { page = 1 }, new { @style = "font-size: 13px" })</li>
                            }
                        }
                        <li class="dropdown">
                            @Html.ActionLink(Resources1.CommentButton, "Comments",
"Home", null, new { @style = "font-size: 13px" })
                        </li>

                    </ul>
```

```
                    </div>
                </div>

            </div>
        </div>

        <!--Body-->
        <div class="container body-content" style="padding-top: 20px; height:100%">
            @RenderBody()
            <div class="push">
            </div>

        </div>
    </div>

    <footer id="footer" style="height:30%">
        <div class="container" style="">
            <div class="row">
                <div class="col-md-8">
                    <div class="block contact-block">
                        <!--todo: replace with company contact details-->
                        <h3>
                            Contacts
                        </h3>
                        <address>
                            <ul class="fa-ul">
                                <li>
                                    <abbr title="Email"></abbr>
                                    marchukovav@tpu.ru
                                </li>
                                <li>
                                    <abbr title="Phone"></abbr>
                                    +7 (3822) 606 133
                                </li>
                                <li>
                                    <abbr title="Address"></abbr>
                                    Russia, Tomsk
                                </li>
                            </ul>
                        </address>
                    </div>
                </div>
                <div class="col-md-4" style="">
                    <center>
                        <div class="list-group">
                            <b> @Html.ActionLink(Resources1.HelpWithConnectionTitle,
"HelpPage", "Home", null, new { @class = "list-group-item list-group-item-primary", style =
"color: black; font-size: 16px" }) </b>
                        </div>
                    </center>
                </div>

            </div>

            <div class="row">
                <div id="toplink">
                    <a href="#top" class="top-link" title="Вернуться на верх">To top!</a>
                </div>
                <!--todo: replace with company copyright details-->
                <div class="subfooter">
                    <div class="col-md-6">
                        <p>Site template by <a href="http://ic.tpu.ru">TPU IC</a> |
Copyright @DateTime.Now.Year &copy; TPU IC</p>
```

```
                        <!--MadeByArkad-->
                    </div>
                </div>
            </div>
        </div>
    </footer>

    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @Scripts.Render("~/bundles/jqueryval")
    @RenderSection("scripts", required: false)

    <script>
        jQuery.extend(jQuery.validator.methods, {
            number: function (value, element) {
                return this.optional(element) || /^-
?(?:\d+|\d{1,3}(?:\.\d{3})+)(?:,\d+)?$/.test(value);
            }
        });
        $(".numericOnly").bind('keypress', function (e) {
            if (e.keyCode == '9' || e.keyCode == '16') {
                return;
            }
            var code;
            if (e.keyCode) code = e.keyCode;
            else if (e.which) code = e.which;
            if (e.which == 46)
                return false;
            if (code == 8 || code == 46)
                return true;
            if (code < 48 || code > 57)
                return false;
        });

        (function ($) {
            $("#en").click(function() {
                $(this).parents("form").submit(); // post form
            });
            $("#ru").click(function() {
                $(this).parents("form").submit(); // post form
            });
        })(jQuery);
    </script>
</body>

</html>
```

Представление «Create Well»

```
@using LangResources
@using WitsmlServer.Web.DbModels.Field
@using WitsmlServer.Web.DbModels.Group
@model WitsmlServer.Web.DbModels.F_G_W_Model.EditWellModel
@{
    ViewBag.Title = Resources1.CreateWell;
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>@ViewBag.Title</h2>

<ol class="breadcrumb">
    <li><a href="@Url.Action("Index", "Home")">@Resources1.HomePageButton</a></li>
```

```
    <li><a href="@Url.Action("Index","PersonalPage")">@Resources1.PersonalListWell</a></li>
    <li class="active">Создание скважины</li>
</ol>

<hr/>
@using (Html.BeginForm("CreateWell", "PersonalPage", FormMethod.Post, new { @class = "form-
horizontal", role = "form" }))
{
    @Html.AntiForgeryToken()
    @Html.ValidationSummary("", new { @class = "text-danger" })
   @* <div class="form-group">
        @Html.LabelFor(m => m.WellUid, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.WellUid, new { @class = "form-control" })
        </div>
    </div>*@


    <div class="form-group">
        @Html.LabelFor(m => m.Name, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.Name, new { @class = "form-control" })
        </div>
    </div>

    <div class="form-group">
        @Html.Label(Resources1.ActiveTimeDays, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.ActiveTime, new { @class = "form-control",
@type="number" })
        </div>
    </div>

    <div class="form-group">
        @Html.LabelFor(m => m.Id_Field, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @{
                var db2 = new FieldModel();
                var query2 =
                    from field in db2.Field
                    select field;
                List<Field> list2 = new List<Field>();
                foreach (var e in query2)
                {
                    list2.Add(e);
                }
                SelectList select_list2 = new SelectList(list2,"Id","Name");
                @Html.DropDownListFor(m => m.Id_Field, select_list2, Resources1.SelectField,
new { @class = "form-control" })
            }
        </div>
    </div>


    <div class="form-group">
        @Html.LabelFor(m => m.Id_Group, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @{
                var db3 = new GroupModel();
                var query3 =
                    from group_ in db3.Group
                    select group_;
                List<Group> list3 = new List<Group>();
```

```
                foreach (var e in query3)
                {
                    list3.Add(e);
                }
                SelectList select_list3 = new SelectList(list3,"Id","Name");
                @Html.DropDownListFor(m => m.Id_Group, select_list3, Resources1.SelectGroup,
new { @class = "form-control" })
            }
        </div>
    </div>



    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-primary" value="@Resources1.CreateWell" />
        </div>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```