МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Системы автоматического управления»

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой

_____/ В.И. Ширяев

«_____» _____ 2018 г.

Разработка web-based системы управления учетными записями, применяемой при тестировании федерального государственного портала

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ – 09.03.01.2018.122.00 ПЗ ВКР

Руководитель работы

доц. каф. САУ, к.т.н.

_____/ *Алешин Е.А.*

«____» _____ 2018 г.

Автор работы

студент группы **КЭ-483**

_____/ *Непряхин А.В.*

«____» _____ 2018 г.

Нормоконтролер

доц. каф. САУ, к.т.н.

_____/ *Алешин Е.А.*

«____» _____ 2018 г.

Челябинск 2018

АННОТАЦИЯ

Непряхин А.В. Разработка web-based системы управления учетными записями, применяемой при тестировании федерального государственного портала: ЮУрГУ (НИУ), ВШ ЭКН; 2018, 62 с. 29 ил., библиогр. список – 10 наим., 14 листов слайдов презентации ф. А4.

Целью данной работы была разработка web-based системы управления учетными записями, применяемой при тестировании федерального государственного портала. Для этого был проведен анализ подходящих языков программирования, существующих баз данных, средств программирования и фреймворков для создания веб приложений. Разработано техническое задание, соответствующее требованием заказчика. Результатом работы была создана программа, отвечающая требованиям ТЗ.

Пояснительная записка отражает все этапы разработки системы, включая анализ существующего метода создания учетных записей, формулировку требований и результатов работы.

Оглавление

ВВЕДЕНИЕ

Развитие информационных технологий дало возможность автоматизировать обработку данных в разных сферах деятельности человека. На текущий момент трудно представить производственные, экономические, научные и образовательные процессы без информационных систем обработки информации. Любой вид человеческой деятельности основывается на процессах реального мира, с которыми связана эта деятельность. Часть реального мира, моделируемая информационной системой, называется предметной областью. Информационная система (далее ИС) разрабатывается для удовлетворения информационных потребностей пользователей.

Абсолютно любая информация имеет «время жизни». Она может существовать кратковременно (в памяти калькулятора в процессе вычислений), некоторое время (при подготовке какой-либо справки) или очень долго (при хранении личных, коммерческих, государственных данных). Эти временные периоды определяют жизненный цикл информации.

Жизненный цикл ИС является производной жизненного цикла информации, информационных продуктов, услуг и технических средств, а также представляет собой модель ее создания и использования. Различные состояния информационной системы отражает ее модель, начиная с момента возникновения необходимости в данной системе и заканчивая моментом ее полного выхода из употребления у последнего пользователя системы. Каждая информационная система представляет собой программное обеспечение. Одним из этапов жизненного цикла ПО является процесс тестирования и отладки продукта, что и будет рассматриваться в данном проекте.

Актуальность проекта обусловлена тем, что разработанный программный продукт автоматизирует процесс создания данных, а также позволяет ускорить процесс тестирования федерального государственного портала, соответственно уменьшить расходы на трудозатраты, и облегчить труд тестировщиков проекта.

Целью настоящей работы является разработка web-based системы управления учетными записями пользователей, применяемой при тестировании федерального государственного портала.

В соответствии с поставленной целью необходимо решить следующие задачи:

- Анализ процесса создания учетных записей;
- Выбор стека технологий;
- Составление технического задания;
- Разработка системы;
- Проведение опытной эксплуатации и внедрение системы.

Для достижения поставленной цели предлагается выполнить ряд мероприятий по комплексному пред проектному обследованию предприятия и, на основании полученных данных, провести проектирование и внедрение системы управления учетными записями пользователей. Проектируемая система позволит автоматизировать процесс создания данных, а также ускорить процесс тестирования федерального государственного портала.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Характеристика предприятия

«ЛАНИТ» — одна из крупнейших групп компаний на российском рынке информационных технологий. Предоставляет широкий комплекс ИТ-услуг, обеспечивающих этапы выполнения проекта от разработки и внедрения до обучения персонала и сервисной поддержки. Основана в 1989 году. Штаб-квартира расположена в Москве. Название компании происходит от «ЛАборатория Новых Информационных Технологий».

Компания специализируется на реализации крупных проектов федерального масштаба “под ключ”, выполняя полный комплекс работ от создания концепции и проектирования до сопровождения и эксплуатации внедренных решений. Помимо этого, компания успешно решает задачи, связанные с информационно-аналитическим сопровождением бизнеса и реализует портальные и интеграционные проекты, разрабатывает программное обеспечение на заказ.

Организационная структура департамента представлена на рисунке 1.1.

Рисунок 1.1 – Организационная структура «Департамента корпоративных систем» компании «Ланит»

1.2 Общая характеристика предметной области

Жизненный цикл программного обеспечения (также называемый циклом разработки) – это условная схема, включающая отдельные этапы, которые представляют стадии процесса создания ПО. При этом на каждом этапе выполняются разные действия.

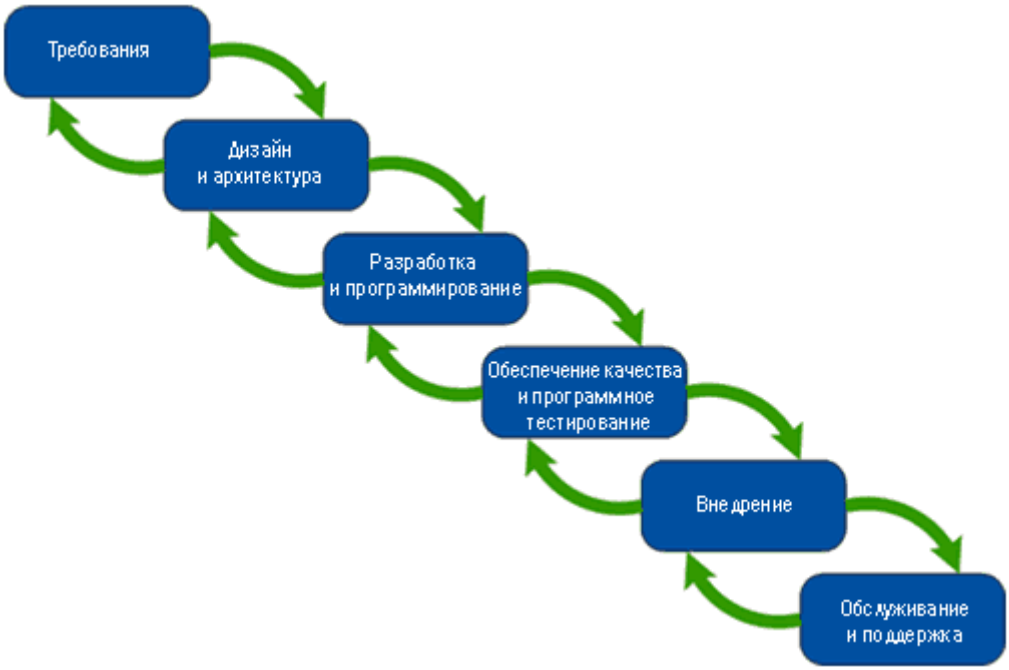

Рисунок 1.2 – Жизненный цикл ПО

1. Анализ требований

Жизненный цикл разработки ПО начинается со стадии анализа, во время которого участники процесса обсуждают требования, предъявляемые к конечному продукту. Цель этой стадии – определение детальных требований к системе. Кроме этого, необходимо убедиться в том, что все участники правильно поняли поставленные задачи и то, как именно каждое требование будет реализовано на практике.

Зачастую, в обсуждении участвуют также и специалисты по тестированию, которые уже на стадии разработки требований могут вносить собственные пожелания и, при необходимости, корректировать процесс.

В зависимости от выбранной модели разработки, могут отличаться подходы к определению момента перехода с одной стадии на другую. К примеру, в

каскадной или V-модели стадия анализа требований закрепляется в документе – спецификации требований к программному обеспечению (Software Requirement Specification, SRS), оформление которого должно быть закончено до перехода на следующую стадию.

Таким образом, этот этап предполагает сбор требований к разрабатываемому программному обеспечению, их систематизацию, документирование, анализ, а также выявление и разрешение противоречий.

2. Проектирование

На стадии проектирования (называемой также стадией дизайна и архитектуры) программисты и системные архитекторы, руководствуясь требованиями, разрабатывают высокоуровневый дизайн системы.

Разнообразные технические вопросы, возникающие в процессе проектирования, обсуждаются со всеми заинтересованными сторонами, включая заказчика. Определяются технологии, которые будут использоваться в проекте, загрузка команды, ограничения, временные рамки и бюджет. В соответствии с уточненными требованиями выбираются наиболее подходящие проектные решения.

Утвержденный дизайн системы определяет перечень разрабатываемых программных компонентов, взаимодействие с третьими сторонами, функциональные характеристики программы, используемые базы данных и многое другое. Дизайн, как правило, закрепляется отдельным документом – дизайн-спецификацией (Design Specification Document, DSD).

На этом этапе для упрощения визуализации процесса проектирования используются так называемые нотации – схематическое выражение характеристик разрабатываемой системы.

Основные используемые нотации:

– Блок-схемы;

– ER-диаграммы;

– UML-диаграммы;

– Макеты – например, нарисованный в фотошопе прототип сайта.

3. Разработка и программирование

После того как требования и дизайн продукта утверждены, происходит переход к следующей стадии жизненного цикла – непосредственно разработке. Здесь начинается написание программистами кода программы в соответствии с ранее определенными требованиями.

Системные администраторы настраивают программное окружение, front-end программисты разрабатывают пользовательский интерфейс программы и логику ее взаимодействия с сервером.

Кроме того, программисты пишут Unit-тесты для проверки правильности работы кода каждого компонента системы, проводят ревью написанного кода, создают билды и разворачивают готовое ПО в программной среде. Этот цикл повторяется до тех пор, пока все требования не будут реализованы.

Программирование предполагает четыре основных стадии:

1) Разработка алгоритмов – фактически, создание логики работы программы;

2) Написание исходного кода;

3) Компиляция – преобразование в машинный код;

4) Тестирование и отладка – речь, главным образом, о юнит-тестировании.

4. Тестирование

Тестировщики занимаются поиском дефектов в программном обеспечении и сравнивают описанное в требованиях поведение системы с реальным.

В фазе тестирования обнаруживаются пропущенные при разработке баги. При обнаружении дефекта, тестировщик составляет отчет об ошибке, который передается разработчикам. Последние его исправляют, после чего тестирование повторяется – но на этот раз для того, чтобы убедиться, что проблема была исправлена, и само исправление не стало причиной появления новых дефектов в продукте.

Тестирование повторяется до тех пор, пока не будут достигнуты критерии его окончания.

5. Внедрение и сопровождение

Когда программа протестирована и в ней больше не осталось серьезных дефектов, приходит время релиза и передачи ее конечным пользователям.

После выпуска новой версии программы в работу включается отдел технической поддержки. Его сотрудники обеспечивают обратную связь с пользователями, их консультирование и поддержку.

В случае обнаружения пользователями тех или иных пост-релизных багов, информация о них передается в виде отчетов об ошибках команде разработки, которая, в зависимости от серьезности проблемы, либо немедленно выпускает исправление (т.н. hot-fix), либо откладывает его до следующей версии программы.

1.3 Модели жизненного цикла программного обеспечения

- **Итеративная модель**

Рисунок 1.3 – Итеративная модель разработки ПО

Не все модели жизненного цикла последовательны.

Существуют также итеративные (или инкрементальные) модели, в которых используется другой подход. Вместо одной продолжительной последовательности действий здесь весь жизненный цикл продукта разбит на ряд отдельных мини-циклов. Причем каждый из них состоит из все тех же базовых стадий модели жизненного цикла. Эти мини-циклы называются итерациями. В каждой из итераций происходит разработка отдельного компонента системы, после чего этот компонент добавляется к уже ранее разработанному функционалу.

Итеративная модель не предполагает полного объема требований для начала работ над продуктом. Разработка программы может начинаться с требований к части функционала, которые могут впоследствии дополняться и изменяться. Процесс повторяется, обеспечивая создание новой версии продукта для каждого цикла.

В несколько упрощенном виде, итеративная модель состоит из четырех основных стадий, которые повторяются в каждой из итераций (plan-do-check-act):

– определение и анализ требований;

– дизайн и проектирование – согласно требованиями. Причем дизайн может как разрабатываться отдельно для данной функциональности, так и дополнять уже существующий;

– разработка и тестирование – кодирование, интеграция и тестирование нового компонента;

– фаза ревью – оценка, пересмотр текущих требований и предложения дополнений к ним.

По результатам каждой итерации принимается решение – будут ли использованы ее результаты для дополнения существующей функциональности в качестве входной точки для начала следующей итерации (т.н. инкрементальное прототипирование). В конечном итоге, достигается точка, в которой все требования были воплощены в продукте – происходит релиз.

В математических терминах, итеративная модель представляет реализацию методики последовательной аппроксимации – то есть, постепенное приближение к образу готового продукта:

Ключ к успешному использованию этой модели – строгая валидация требований и тщательная верификация разрабатываемой функциональности в каждой из итераций.

Основные стадии процесса разработки в итеративной модели фактически повторяют модель водопада. В каждой итерации создается программное обеспечение, требующее тестирования на всех уровнях.

Плюсы и минусы итеративной модели:

+ раннее создание работающего ПО;

+гибкость – готовность к изменению требований на любом этапе разработки;

+ каждая итерация – маленький этап, для которого тестирование и анализ рисков обеспечить проще, чем для всего жизненного цикла продукта.

— каждая фаза – самостоятельна, отдельные итерации не накладываются;

— могут возникнуть проблемы с реализацией общей архитектуры системы, поскольку не все требования известны к началу проектирования.

Когда использовать итеративную модель:

– для крупных проектов;

– когда известны, по крайней мере, ключевые требования;

– когда требования к проекту могут меняться в процессе разработки.

Итеративная модель является ключевым элементом так называемых «гибких» (Agile) подходов к разработке программного обеспечения.

- **Спиральная модель**

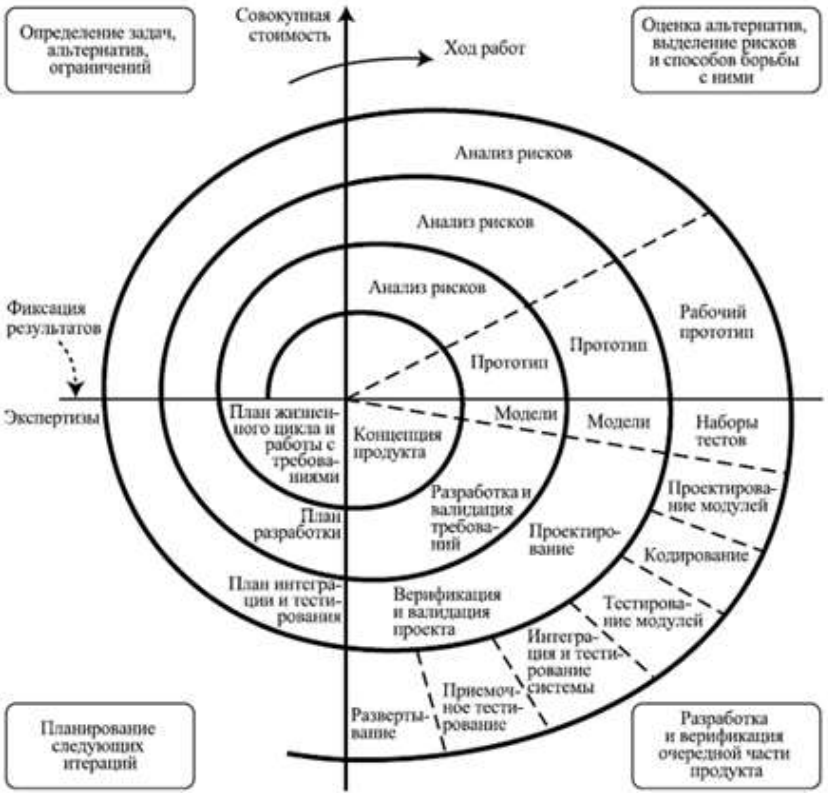

Рисунок 1.4 – Спиральная модель разработки ПО

Спиральная модель представляет шаблон процесса разработки ПО, который сочетает идеи итеративной и каскадной моделей. Суть ее в том, что весь процесс создания конечного продукта представлен в виде условной плоскости, разбитой на 4 сектора, каждый из которых представляет отдельные этапы его разработки: определение целей, оценка рисков, разработка и тестирование, планирование новой итерации.

В спиральной модели жизненный путь разрабатываемого продукта изображается в виде спирали, которая, начавшись на этапе планирования, раскручивается с прохождением каждого следующего шага. Таким образом, на выходе из очередного витка мы должны получить готовый протестированный прототип, который дополняет существующий билд. Прототип, удовлетворяющий всем требованиям – готов к релизу.

Главная особенность спиральной модели – концентрация на возможных рисках. Для их оценки даже выделена соответствующая стадия. Основные типы рисков, которые могут возникнуть в процессе разработки ПО:

— Нереалистичный бюджет и сроки;

— Дефицит специалистов;

— Частые изменения требований;

— Чрезмерная оптимизация;

— Низкая производительность системы;

— Несоответствие уровня квалификации специалистов разных отделов.

Плюсы и минусы спиральной модели:

+ улучшенный анализ рисков;

+ хорошая документация процесса разработки;

+гибкость – возможность внесения изменений и добавления новой функциональности даже на относительно поздних этапах;

+ раннее создание рабочих прототипов.

— может быть достаточно дорогой в использовании;

—управление рисками требует привлечения высококлассных специалистов;

— успех процесса в большой степени зависит от стадии анализа рисков;

— не подходит для небольших проектов.

Когда использовать спиральную модель:

– когда важен анализ рисков и затрат;

– крупные долгосрочные проекты с отсутствием четких требований или вероятностью их динамического изменения;

– при разработке новой линейки продуктов.

- **V-модель**

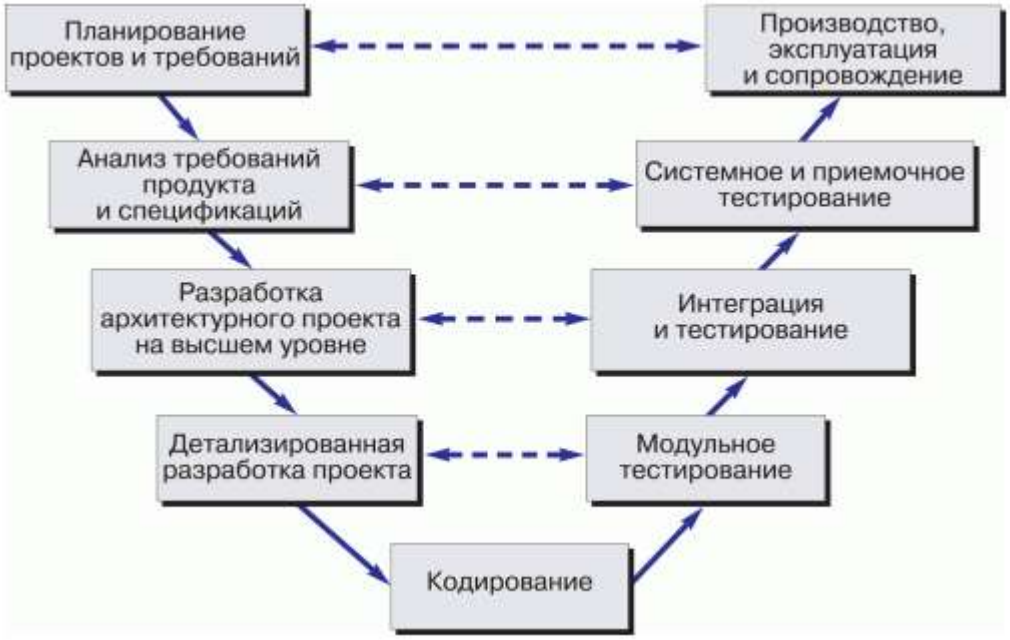

Рисунок 1.5 – V-модель разработки ПО

V-модель – это улучшенная версия классической каскадной модели. Здесь на каждом этапе происходит контроль текущего процесса, для того чтобы убедится в возможности перехода на следующий уровень. В этой модели тестирование начинается еще со стадии написания требований, причем для каждого последующего этапа предусмотрен свой уровень тестового покрытия.

Для каждого уровня тестирования разрабатывается отдельный тест-план, то есть во время тестирования текущего уровня, мы также занимаемся разработкой стратегии тестирования следующего. Создавая тест-планы, мы также определяем ожидаемые результаты тестирования и указываем критерии входа и выхода для каждого этапа.

В V-модели каждому этапу проектирования и разработки системы соответствует отдельный уровень тестирования. Здесь процесс разработки представлен нисходящей последовательностью в левой части условной буквы V, а стадии тестирования – на ее правом ребре. Соответствие этапов разработки и тестирования показано горизонтальными линиями.

Плюсы и минусы V-модели:

+ строгая этапизация;

+планирование тестирования и верификация системы производятся на ранних этапах;

+ улучшенный, по сравнению с каскадной моделью, тайм-менеджмент;

+ промежуточное тестирование.

— недостаточная гибкость модели;

— собственно создание программы происходит на этапе написания кода, то есть уже в середине процесса разработки;

— недостаточный анализ рисков;

— нет работы с параллельными событиями и возможности динамического внесения изменений.

Когда использовать V-модель:

– В проектах, в которых существуют временные и финансовые ограничения;

– Для задач, которые предполагают более широкое, по сравнению с каскадной моделью, тестовое покрытие.

- **Каскадная модель**

Рисунок 1.6 – Каскадная модель разработки ПО

Современные системы настолько большие и сложные, что над их созданием трудятся целые команды специалистов разного профиля: программисты, аналитики, системные администраторы, тестировщики и конечные пользователи. Все они работают вместе для разработки программ, содержащих миллионы строк кода.

Самой старой и известной моделью построения многоуровневого процесса разработки является каскадная (или попросту водопадная) модель: в ней каждый этап разработки, соответствующий стадии жизненного цикла ПО, продолжает предыдущий. То есть, для того, чтобы перейти на новый этап, мы полностью должны завершить текущий.

Каскадная модель проста и понятна, но не так практична как раньше. В условиях динамично изменяющихся требований, строго структурированный процесс может из преимущества превратиться в помеху на пути успешного завершения разработки системы. Поэтому сегодня водопадная модель применяется преимущественно крупными компаниями для больших и сложных проектов, которые предполагают всеобъемлющий контроль рисков.

Плюсы и минусы каскадной модели:

+ Полное документирование каждого этапа;

+ Четкое планирование сроков и затрат;

+ Прозрачность процессов для заказчика;

— Необходимость утверждения полного объема требований к системе еще на первом этапе;

— В случае необходимости внесения изменений требований позднее – возврат к первой стадии и переделка заново всей проделанной работы;

— Увеличение затрат средств и времени в случае необходимости изменения требований.

Несмотря на то, что каскадная модель все еще используется, она уже утратила былые позиции. Сегодня ей на смену приходят более продвинутые модели и методологии разработки программного обеспечения.

Когда использовать каскадную модель:

– В проектах с четко определенными требованиями, для которых не предусматривается их изменений в процессе разработки;

– Для проектов, которые мигрируют с одной платформы на другую. То есть, требования остаются те же, меняется только системное окружение и/или язык программирования;

– Когда от компании-разработчика не требуется проводить тестирования – к примеру, его обеспечением займется сам заказчик или сторонняя фирма.

1.4 Процесс тестирования на предприятии

Любой большой и качественный продукт не может существовать без процесса тестирования. Цель тестирования – предоставление актуальной информации о соответствии производимого продукта требованиям. Компания разрабатывает федеральный государственный портал, тем самым качество продукта должно быть на высоте. Для этого существуют отделы тестирования и поддержки пользователей. Специалисты поддержки дают обратную связь поьзователям и консультируют их. Тестировщики проекта занимаются следующими задачами:

- Тестирование доработок(новых функций)
- Написание тестовой документации
- Поиск причин проблем пользователей
- Регресионное и смоук тестирование(проверка основного функционала системы)

При нахождении ошибки(bug) в системе, специалист заводит отчет об ошибке(bug-report), который в дальнейшем отправляется на анализ в отдел разработки. После исправления, разработчик отправляет исправленную ошибку на тестирование, если она более не воспроизводится, то отчет об ошибке закрывается.

Так как в системе зарегистрировано большое количество пользователей, причем у каждого пользователя существует еще и роль в системе, а иногда и не одна (то есть возможных конфигураций учетных записей огромное количество),

то для проверки функционала системы требуется большое количество тестовых учетных записей. Можно сказать что мы имитируем работу пользователя на тестовых данных, тем самым ищем причины ошибок, которые у них могут возникнуть или уже возникли.

1.5 Единая система идентификации и аутентификации (ЕСИА)

Единая система идентификации и аутентификации (ЕСИА) предназначена для формирования единых методов регистрации, идентификации и аутентификации пользователей во всех государственных информационных системах.

- Обеспечение санкционированного доступа участников информационного взаимодействия, заявителей и иных пользователей к информации, содержащейся в государственных информационных системах, муниципальных информационных системах и иных информационных системах;
- Предоставление доступа к государственным и муниципальным услугам, в том числе услугам, предоставляемым государственными и муниципальными учреждениями и другими организациями, в которых размещается государственное задание (заказ) или муниципальное задание (заказ);
- Исполнение государственных и муниципальных функций;
- Формирование базовых государственных информационных ресурсов, определяемых Правительством Российской Федерации;
- Межведомственное электронное взаимодействие.

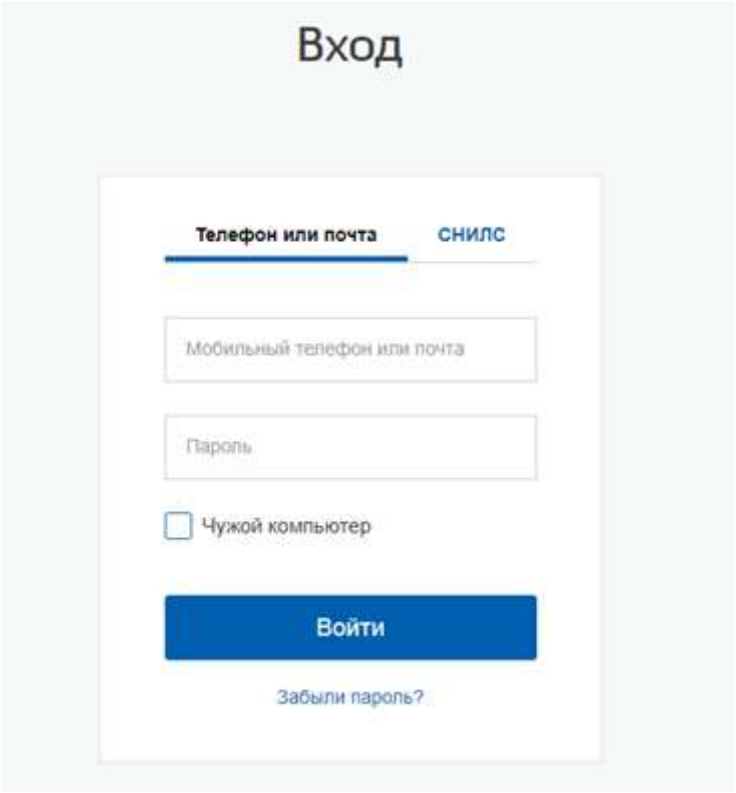

Рисунок 1.7 – Форма входа ЕСИА

Наша система не оперирует пользователями, она наделяет пользователя, который хранится в базе ЕСИА, определенными правами и обязанностями. Рядовой пользователь для входа в систему проходит авторизацию в ЕСИА. Для процессов тестирования в компании используется тестовый ЕСИА, в котором находятся тестовые пользователи, а не реальные. Не следует путать ЕСИА и тестовый ЕСИА.

На рисунке 1.4 можно увидеть процесс взаимодействия систем

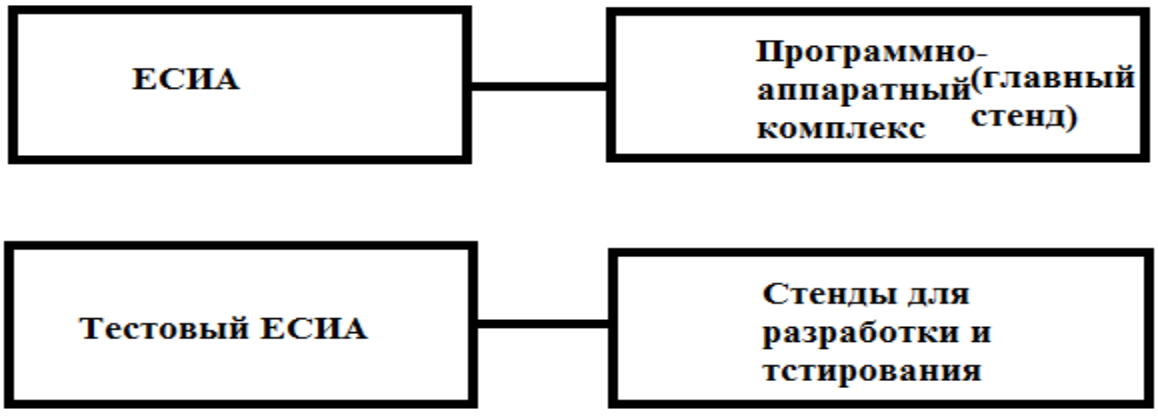

Рисунок 1.8 – Процесс взаимодействия ЕСИА и тестовой ЕСИА со стендами

Использование стендов обусловлено принципами разработки кода при использовании нескольких веток, иными словами используется система управления версиями. Как правило, системы управления версиями хранят историю изменений в виде линии (централизованные) или графа (распределенные). Ветка (бранч) – это просто линия разработки кода, которая имеет общую историю с другими ветками и существует параллельно с ними. Jeff Atwood в своем блоге сравнивает ветки с параллельными вселенными. В такой вселенной в какой-то момент история пошла по-другому относительно других. Это дает нам безграничные возможности, которые уравновешиваются безграничной сложностью наших вселенных.

Как правило, одна из наших историй является основной и носит гордое имя trunk или mainline. По аналогии с деревом, от нее отходят другие ветки. В эту ветку рано или поздно попадает готовый (или не совсем) функционал и исправления ошибок.

Рассмотрим один из этих случаев, когда отдельная ветка создается под каждый релиз. Это делается для того, чтобы исправлять дефекты, найденные после выпуска релиза или во время его тестирования. Этот процесс обычно называется стабилизацией. При этом сами исправления (багфиксы) не остаются только в релизных ветках, а переносятся в mainline (если история релиза и mainline не слишком разошлись), делая ее стабильнее. Код в релизной ветке изолирован от дестабилизирующего влияния разработки нового функционала и при этом не блокирует ее. Сама по себе релизная ветка предоставляет легкую возможность осуществлять поддержку релизной версии. Когда прекращается поддержка релиза, его ветка замораживается. А пока идет проект, mainline продолжает свое развитие, являясь точкой, в которой накапливается новый функционал для следующих релизов. Таким же образом можно осуществлять поддержку релизов для разных заказчиков, выделяя по ветке для каждого, если по каким-то причинам им нельзя поставить одну и ту же версию. Хочу отметить, что поддержка разных вариаций одной и той же версии — задача трудоемкая и ее

следует избегать по мере возможности. На рисунке 1.5 можно наглядно увидеть процесс использования системы управления версиями.

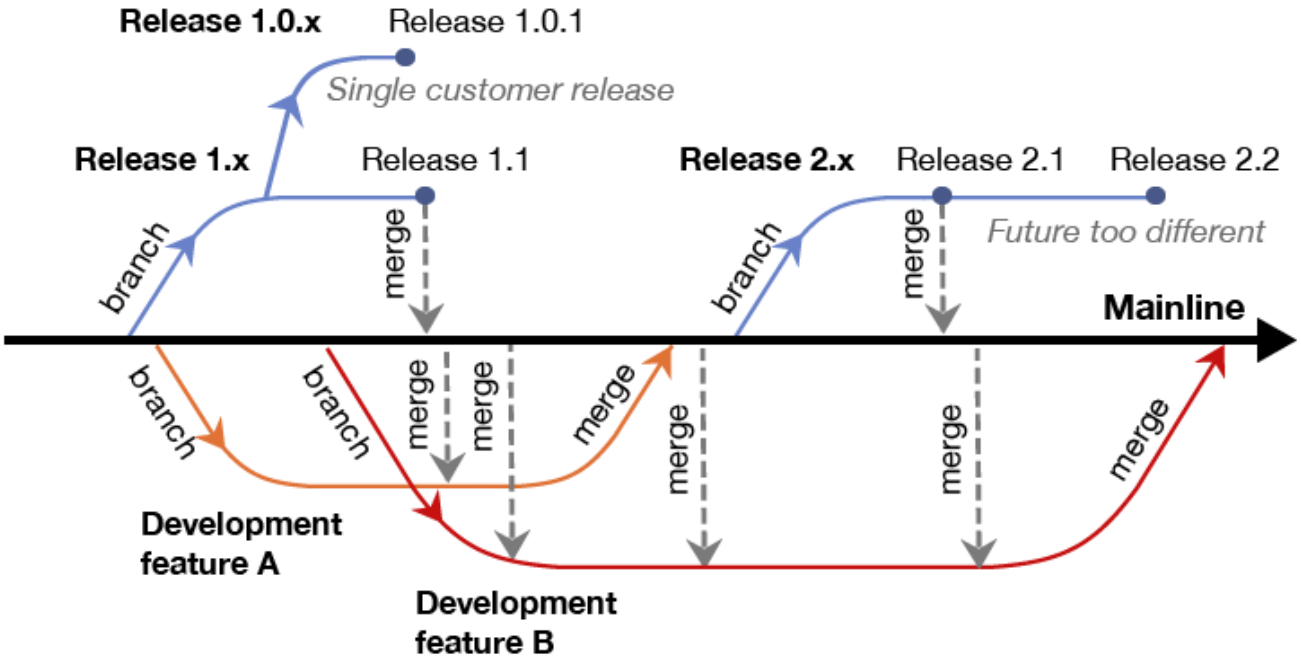

Рисунок 1.9 – Процесс использования нескольких веток кода

У такой модели есть градация стабильности, где самыми стабильными являются релизные ветки, менее стабильной является mainline, и самыми нестабильными являются ветки для разработки. Как правило, на диаграммах самые стабильные ветки отображаются выше всех, а нестабильные – ниже всех.

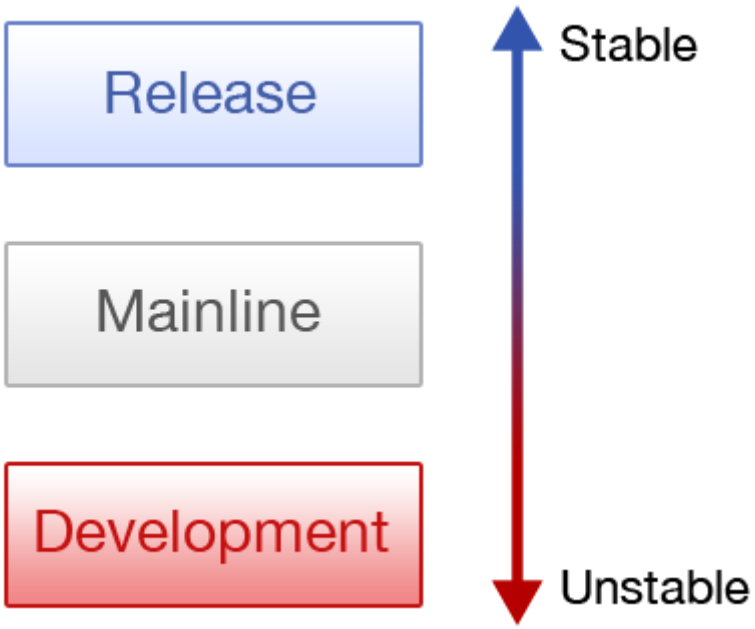

Рисунок 1.10 – Схема градации стабильности веток

2 ПРОЕКТНАЯ ЧАСТЬ

2.1 Описание постановки комплекса задач

Целью выпускной квалификационной работы считается разработка web-based системы управления учетными записями, применяемой при тестировании федерального государственного портала. Для выполнения поставленной цели решены следующие задачи:

– анализ существующего метода создания учетных записей;

– выбор инструментов для разработки системы;

– разработка технического задания;

– разработка блок-схемы алгоритма программы;

– программная реализация.

Для достижения поставленной цели предлагается выполнить ряд мероприятий по комплексному пред проектному обследованию предприятия и, на основании полученных данных, провести проектирование и внедрение управляющей системы. Проектируемая автоматизированная система позволит автоматизировать процесс создания данных, а также позволяет ускорить процесс тестирования федерального государственного портала, соответственно и уменьшить расходы на трудозатраты. Данная система предназначена для использования сотрудниками отдела тестирования и отдела поддержки пользователей. На стадии внедрения системы предполагается проведение ряда мероприятий, обеспечивающих мгновенный переход пользователей от существующего к внедряемому способу решения задачи создания учетных записей пользователей в фирме «ЛАНИТ».

В частности планируется:

– проведение опытной эксплуатации и обучения группы пользователей;

– разработка инструкций и рекомендаций по работе с системой;

– комплексное документирование проекта;

– переход к эксплуатации спроектированной системы.

2.2 Анализ существующего метода создания учетных записей

Для начала требуется разобраться как устроен процесс создания учетных записей. Рассмотрим существующий алгоритм, который указан на рисунке 2.1:

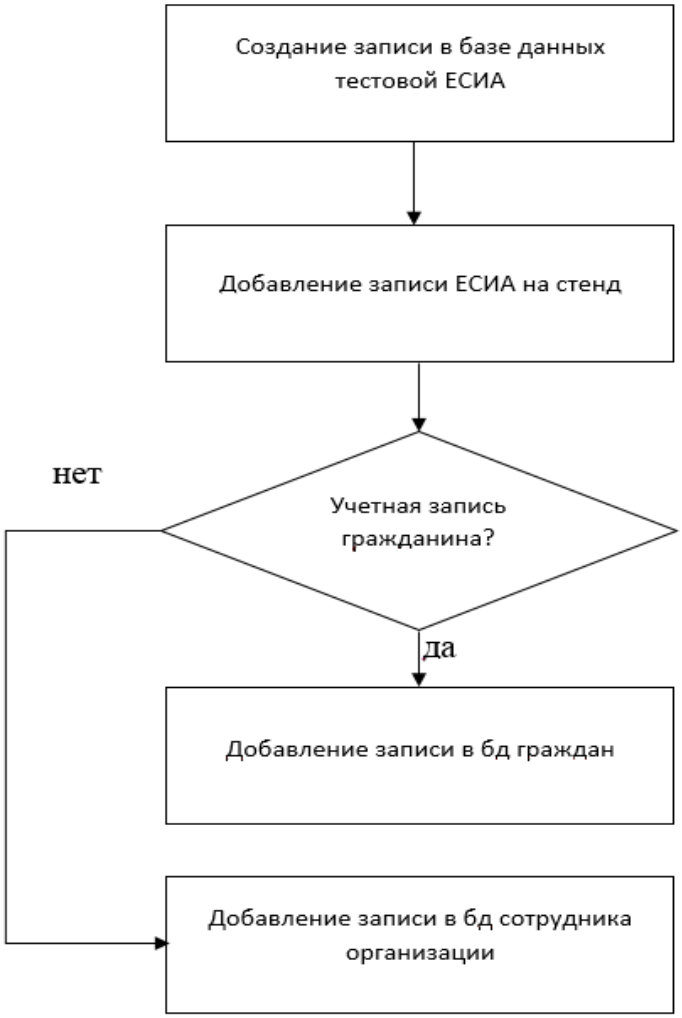

Рисунок 2.1 – Алгоритм создания учетных записей

Рассмотрим отдельно каждый этап:

1) Создание записи в бд тестовой ЕСИА. Для этого нужно выполнить операцию INSERT в бд пользователей ЕСИА, пример запроса:

INSERT INTO пользователи.есиа_пользователи

(уникальный идентификатор пользователя, имя пользователя, имя, фамилия, отчество, инн, снилс, огрн, e-mail, метод аутентификации, токен аутентификации, тип, роль записи, принадлежность к другим организациям, адрес организации, контактные данные организации, уникальный идентификатор организации, кпп организации, правовая форма организации, инн организации, имя организации,

сокращенное название организации, огрн организации, должность, тип организации, пароль, пол, дата рождения)

VALUES('75ead602-188f-42a5-875c-ce0f36d73430', 'uo1Admin', 'Алексей', 'Рыбаков', 'Кириллович', NULL, NULL, NULL, 'rybakov.ukpartner@list.ru', 'DS', '2a81a024-4e1c-49e7-b06c-c79f2a441afc', NULL, 'E', 'ADMIN', '<?xml version="1.0" encoding="UTF-8" standalone="yes"?><orgaddresses><address><addresstype>ORG_POSTAL</addresstype><contrychar3code>RUS</contrychar3code><index>300041</index><region>Тульская Область</region><district>г. Тула, ул. Вересаева</district><house>2</house><corpus></corpus><structure></structure><flat>1</flat></address><address><addresstype>ORG_LEGAL</addresstype><contrychar3code>RUS</contrychar3code><index>300041</index><kladrcode>7100000100003310004</kladrcode><russianregioncode>71</russianregioncode><region>Тульская обл.</region><district></district><settlement>г. Тула</settlement><street>ул. Вересаева</street><house>2</house><corpus></corpus><structure></structure><flat></flat></address></orgaddresses>', '<?xml version="1.0" encoding="UTF-8" standalone="yes"?><orgcontacts><contact><contacttype>PHN</contacttype><value>+7(872)563432</value><verificationstatus>N</verificationstatus></contact><contact><contacttype>EML</contacttype><value>ukpartner@list.ru</value><verificationstatus>N</verificationstatus></contact></orgcontacts>', '8fb740fe-ff5a-4d6a-b43f-2f1130717cd1', '710601001', 'Закрытые акционерные общества', '7106522137', 'Закрытое акционерное общество "Партнер"', 'ЗАО "Партнер"', '1127154013249', 'Специалист', 'L', 'RuRKoLwh2Kgm15NE3zi+Sw==', 'MALE', '01-01-1984');

Все данные вводятся вручную, при этом следует учитывать, что некоторые параметры имеют уникальный идентификатор.

2) Добавление записи ЕСИА на стенд.

После того, как у нас есть учетная запись в тестовой ЕСИА, смотрим: если добавляется сотрудник организации, нужно проверить, добавлена ли организация в базу данных, для положительного результата в бд организаций должна присутствовать запись об организации сотрудника.

- Если мы хотим добавить на стенд учетную запись гражданина, мы производим операцию INSERT, пример запроса:

INSERT INTO пользователи.гражданин

(уникальный идентификатор пользователя, фамилия, имя, отчество, e-mail,логин , статус, снилс, дата рождения, идентификатор соглашения пользователя)

VALUES (соответствующие значения)

- Если нам нужно добавить сотрудника организации:

INSERT INTO пользователи.сотрудник

(уникальный идентификатор пользователя, логин, идентификатор организации, фамилия, имя, отчество, e-mail, должность, e-mail организации, статус записи, снилс, дата рождения, идентификатор соглашения пользователя, признак владельца организации)

VALUES (соответствующие значения)

- Добавление организации:

INSERT INTO ppa_organizations

(уникальный идентификатор организации, тип организации, полное название организации, короткое название организации, огрн, инн, кпп, , часовой пояс, e-mail организации, сайт, дата создания, статус, адрес организации, телефон, факс, численность персонала , дата регистрации в системе, имя начальника, отчество начальника, фамилия начальника, родительский гуид)

VALUES (соответствующие значения)

Примерное время создания одной учетной записи составляет 5 минут.

Для сокращения времени сотрудников на создание учетных записей, было принято решение сделать этот процесс простым и интуитивно понятным, создав визуализированную утилиту, с некоторым функционалом, который будет описан далее.

2.3 Сравнение web и desktop приложений

Для начала нужно было определиться какое приложение нам нужно: web или же desktop? Для этого была проведена сравнительная характеристика, которую можно увидеть в таблице 2.1.

Таблица 2.1 – Сравнение web и desktop приложений

	Desktop приложение	**Web приложение**
1	**2**	**3**
Доступ к сети Internet	Не требуется.	Необходим, исключение: некоторые приложения могут временно работать автономно.
Установка/обновление	Должно быть развёрнуто или установлено.	Единовременная настройка. Одна установка для всех пользователей. Благодаря централизованности моментально обновление.
Интерфейс взаимодействия	Стандартные интерфейсы, стандартное взаимодействие.	Разнообразный интерфейс взаимодействия. Плюсы — разнообразие реализации, минусы, сложности — кроссбраузерная совместимость. Решается применением библиотек на JavaScritp.
Совместимость с устройствами	Зависимость от платформы. Исключение — кроссплатформенные приложения.	В большинстве случаем — платформо-независимое.

Продолжение таблицы 2.1 – Сравнение web и desktop приложений

1	2	3
Анимация, графика	Быстрая, быстрый отклик.	Относительно медленный.
Медиа	Незначительные проблемы с аудио и видео.	Проблемы. На данный момент всё реализуется через Flash. Но в разработке стандарт HTML5, который подразумевает поддержку аудио и видео на уровне браузера.
Шрифты	Присутствуют только те шрифты, которые установлены у пользователя.	Любые шрифты — есть возможность подгрузки необходимого шрифта через Internet.
Поиск по контенту	Нет, если только не реализовано на уровне приложения.	Да есть. Причём можно организовать свой поиск, но и воспользоваться сторонними сервисами.
Расшаривание	Если только дополнительно настроить.	Всё выполняется на сервере, пользователю не нужно знать процессы, происходящие на сервере. Кроссплаформенно, нужен только браузер. Инструменты, софт на сервере зачастую кроссплатформенны.

Продолжение таблицы 2.1 – Сравнение web и desktop приложений

1	2	3
Разработка	Под каждую платформу есть свои инструменты, зачастую под каждую платформу приходиться писать свою версию.	Всё выполняется на сервере, пользователя не волнует как там исполняется всё на сервере. Кроссплаформенно, нужен только браузер. Инструменты, софт на сервере зачастую кроссплатформенны.
Масштабы	Повсеместно.	Пока что web-приложения не столь популярны. Но темпы роста популярности(в куче с «облаками») велики. Уже сейчас многие переходят на хранение документов на Google Docs и прочие сервисы.
Тестирование	Производится тестером, группой тестеров. Для opensource происходит тестирования всеми, кому это интересно.	По сути всё так же. Только открытость(расположение в сети) данного рода приложений позволяет привлечь большее количество тестеров. В результате

Окончание таблицы 2.1 – Сравнение web и desktop приложений

1	2	3
		большее покрытие тестами и более быстрое обнаружение уязвимостей и некорректной работы софта.

Теперь рассмотрим ключевые отличия:

1) Архитектура. Клиент-серверная программа включает в себя 2 уровня: клиентскую машину и сервер; веб-приложение имеет 3 уровня: веб-браузер, сервер и сервер баз данных.
2) Количество одновременных пользователей. Программа клиент-сервер может поддерживать одновременную работу ограниченного числа пользователей; программа с веб архитектурой поддерживает неограниченное количество одновременных пользователей.
3) Клиент-серверное приложение управляется с помощью меню, в то время как веб-программа активизируется с помощью URL.
4) Использование куки. Куки нужны для веб-приложения, они не используются клиент-серверным ПО.
5) Количество дефектов. Тестирование производительности, тестирование безопасности, тестирование на совместимость показывают, что веб-программа, как правило, имеет меньше проблем, чем клиент-серверная.[6]

Основные требования, предъявляемые компанией :

- Поддержка легковесной интеграции
- Чтобы систему было достаточно просто поддерживать

Легковесная интеграция представлена протоколом SOAP. SOAP – это протокол на основе XML, который позволяет обмениваться информацией по определенному протоколу (например, HTTP или SMTP) между приложениями.

Это означает простой протокол доступа к объектам и использует XML для его формата обмена сообщениями для передачи информации. Для данного требования нам идеально подходит веб-приложение.

Веб-приложение имеет кроссбраузерность, это означает что нам придется поддерживать только 5 основных браузеров: Internet Explorer, Mozilla Firefox, Opera, Google Chrome и Safari.

Для desktop приложения существует огромное количество конфигураций компьютера и операционных системы, поэтому поддержка такого приложения будет довольно-таки трудной.

В результате анализа всего вышеперечисленного, было принято решение разрабатывать web-приложение.

2.4 Выбор инструментов для разработки программы

1) Выбор фреймворка.

Фреймворк - это программная оболочка, позволяющая упростить и ускорить решение типовых задач, характерных для данного языка программирования. Само слово framework означает «каркас» в переводе с английского. Действительно, фреймворки и призваны быть готовыми каркасами программ, на которые только и остаётся навесить стены и окна.

На данный момент самыми популярными фреймворками являются jQuery и AngularJS. Они работают на разных уровнях. Простейший способ просмотра разницы с точки зрения начинающего заключается в том, что jQuery по существу является аннотация JavaScript, поэтому способ проектирования страницы для JavaScript в значительной степени заключается в том, как мы это сделаем для jQuery. Процесс действительно начинается с нуля, поэтому конечным результатом является желаемый вид.

С jQuery вы выполняете манипуляции с dom, используя AngularJs вы создаете целые веб-приложения.

jQuery был создан, чтобы абстрагироваться от разных особенностей браузера и работать с DOM без необходимости добавлять проверки IE6 и т.д., Со временем он разработал хороший, надежный API, который позволил нам многое

сделать, но в его основе он предназначен для работы с DOM, поиска элементов, изменения пользовательского интерфейса и т.д.

AngularJS был создан как слой поверх jQuery, чтобы добавить концепции MVC для разработки интерфейса. Вместо того, чтобы давать вам API для работы с DOM, AngularJS дает вам привязку данных, шаблоны, пользовательские компоненты (похожие на jQuery UI, но декларативный вместо запуска через JS) и многое другое.

В результате был выбран Angular.js с его плюсами:

- Большое комьюнити
- Декларированный стиль кода
- Высокая скорость разработки
- Модульность
- Наличие готовых решений
- Простота тестирования

AngularJS позиционирует себя как фреймворк, улучшающий HTML. Он собрал концепции из разных языков программирования, как JavaScript, так и серверных, и делает из HTML также нечто динамическое. Мы получаем подход, основанный на данных, к разработке приложений. Нет нужды обновлять Модель, DOM или делать какие-то другие затратные по времени операции, например, исправлять ошибки браузеров. Мы концентрируемся на данных, данные же заботятся об HTML, а мы просто занимаемся программированием приложения.[3]

На рисунке 2.2 представлена сравнительная характеристика:

	jQuery	AngularJS
Abstracts the DOM	✓	✓
Unit Test Runner	✓	✓
Deferred Promises	✓	✓
Cross-Module Communication	✓	✓
Animation Support	✓	✓
AJAX / JSONP	✓	✓
RESTful API	✗	✓
Integration Test Runner	✗	✓
MVC Pattern Support	✗	✓
Templating	✗	✓
Two-way Data Binding	✗	✓
Dependency Management	✗	✓
Deep-Link Routing	✗	✓
Form Validation	✗	✓
Localization	✗	✓
File Size	32KB	38KB

Рисунок 2.2 – Сравнение jQuery и AngularJS

1) Выбор языка программирования:

На проекте используется язык программирования java, рассмотрим его основные преимущества:

- Объектно-ориентированный: в Java все является объектом. Дополнение может быть легко расширено, так как он основан на объектной модели.
- Платформонезависимый: в отличие от многих других языков, включая C и C++, Java, когда был создан, он не компилировался в платформе конкретной машины, а в независимом от платформы байт-коде. Этот байт код распространяется через интернет и интерпретируется в Java Virtual Machine (JVM), на которой он в настоящее время работает.
- Простой: процессы изучения и введение в язык программирования Java остаются простыми. Если Вы понимаете основные концепции объектно-ориентированного программирования, то он будет прост для Вас в освоении.
- Безопасный: методы проверки подлинности основаны на шифровании с открытым ключом.
- Архитектурно-нейтральный: компилятор генерирует архитектурно-нейтральные объекты формата файла, что делает скомпилированный код исполняемым на многих процессорах, с наличием системе Java Runtime.
- Портативный: архитектурно-нейтральный и не имеющий зависимости от реализации аспектов спецификаций — все это делает Java портативным. Компилятор в Java написан на ANSI C с чистой переносимостью, который является подмножеством POSIX.
- Прочный: выполняет усилия, чтобы устранить ошибки в различных ситуациях, делая упор в основном на время компиляции, проверку ошибок и проверку во время выполнения.
- Многопоточный: функции многопоточности, можно писать программы, которые могут выполнять множество задач одновременно. Введение в язык Java этой конструктивной особенности позволяет разработчикам создавать отлаженные интерактивные приложения.

- Интерпретированный: Java байт-код переводится на лету в машинные инструкции и нигде не сохраняется. Делая процесс более быстрым и аналитическим, поскольку связывание происходит как дополнительное с небольшим весом процесса.
- Высокопроизводительный: введение Just-In-Time компилятора, позволило получить высокую производительность.
- Распространенный: предназначен для распределенной среды интернета.
- Динамический: программирование на Java считается более динамичным, чем на C или C++, так как он предназначен для адаптации к меняющимся условиям. Программы могут выполнять обширное количество во время обработки информации, которая может быть использована для проверки и разрешения доступа к объектам на время выполнения.

2) Сервер jboss

JBossAS — J2EE сервер приложений с открытым исходным кодом. Сервер приложений — это программная платформа (software framework) предназначенная для эффективного исполнения процедур (программ, механических операций, скриптов) которые поддерживают построение приложений.

J2EE — Java Platform, Enterprise Edition. набор спецификаций и соответствующей документации для языка Java, описывающей архитектуру серверной платформы для задач средних и крупных предприятий. Основная цель спецификаций — обеспечить масштабируемость приложений и целостность данных во время работы системы. J2EE является промышленной технологией и в основном используется в высокопроизводительных проектах, в которых необходима надежность, масштабируемость, гибкость.

JMX — Java Management Extensions (JMX) — Java технология, которая поставляет инструменты для управления и мониторинга приложений, системных объектов, устройства (например, принтеров) и сервис-ориентированных сетей. Эти ресурсы представлены объектами, называемыми MBeans.

3) База данных Postgresql

PostgreSQL свободно распространяемая и максимально соответствует стандартам SQL. PostgreSQL или Postgres стараются полностью применять ANSI/ISO SQL стандарты своевременно с выходом новых версий.

От других СУБД PostgreSQL отличается поддержкой востребованного объектно-ориентированного и/или реляционного подхода к базам данных. Например, полная поддержка надежных транзакций, т.е. атомарность, последовательность, изоляционность, прочность (Atomicity, Consistency, Isolation, Durability (ACID).) Благодаря мощным технологиям Postgre очень производительна. Параллельность достигнута не за счет блокировки операций чтения, а благодаря реализации управления многовариантным параллелизмом (MVCC), что также обеспечивает соответствие ACID. PostgreSQL очень легко расширять своими процедурами, которые называются хранимые процедуры. Эти функции упрощают использование постоянно повторяемых операций.

Хотя PostgreSQL и не может похвастаться большой популярностью в отличии от MySQL, существует довольно большое число приложений облегчающих работу с PostgreSQL, несмотря на всю мощность функционала. Сейчас довольно легко установить эту СУБД используя стандартные менеджеры пакетов операционных систем.

Преимущества:

- Открытое ПО соответствующее стандарту SQL - PostgreSQL - бесплатное ПО с открытым исходным кодом. Эта СУБД является очень мощной системой.
- Большое сообщество - существует довольно большое сообщество в котором вы запросто найдёте ответы на свои вопросы
- Большое количество дополнений - несмотря на огромное количество встроенных функций, существует очень много дополнений, позволяющих разрабатывать данные для этой СУБД и управлять ими.
- Расширения - существует возможность расширения функционала за счет сохранения своих процедур.

- Объектность - PostrgreSQL это не только реляционная СУБД, но также и объектно-ориентированная с поддержкой наследования и много другого

Недостатки:

- Производительность - при простых операциях чтения PostgreSQL может значительно замедлить сервер и быть медленнее своих конкурентов, таких как MySQL
- Популярность - по своей природе, популярностью эта СУБД похвастаться не может, хотя и присутствует довольно большое сообщество.
- Хостинг - в силу выше перечисленных факторов иногда довольно сложно найти хостинг с поддержкой этой СУБД.[2]

2.5 Разработка технического задания

Техническое задание — исходный документ на проектирование технического объекта (изделия). ТЗ устанавливает основное назначение разрабатываемого объекта, его технические характеристики, показатели качества и технико-экономические требования, предписание по выполнению необходимых стадий создания документации (конструкторской, технологической, программной и т. д.) и её состав, а также специальные требования. Техническое задание является юридическим документом — как приложение включается в договор между заказчиком и исполнителем на проведение проектных работ и является его основой: определяет порядок и условия работ, в том числе цель, задачи, принципы, ожидаемые результаты и сроки выполнения. То есть должны быть объективные критерии, по которым можно определить, сделан ли тот или иной пункт работ или нет. Все изменения, дополнения и уточнения формулировок ТЗ обязательно согласуются с заказчиком и им утверждаются. Это необходимо и потому, что в случае обнаружения в процессе решения проектной задачи неточностей или ошибочности исходных данных возникает необходимость определения степени вины каждой из сторон-участниц разработки, распределения понесенных в связи с этим убытков. Техническое задание, как термин в области информационных технологий – это юридически значимый документ, содержащий

исчерпывающую информацию, необходимую для постановки задач исполнителям на разработку, внедрение или интеграцию программного продукта, информационной системы, сайта, портала либо прочего ИТ сервиса.[10]

Для реального внедрения системы, требуется описать ее в техническом задании и затем согласовать с руководством. Совместно с аналитиками проекта было разработано техническое задание.

Были описаны следующие варианты использования:

- ЕСИА_1: Просмотр реестра записей ЕСИА
- ЕСИА_2: Просмотр записи ЕСИА
- ЕСИА_3: Создание записи ЕСИА
- ЕСИА_4: Редактирование записи ЕСИА
- ЕСИА_5: Поиск записи ЕСИА
- ПЛ_1: Просмотр реестра записей площадки
- ПЛ_2: Поиск записи в реестре площадки
- ПЛ_3: Добавление записи в реестр записей площадки
- ПЛ_4: Просмотр записи в реестре площадки
- АВТ_1: Авторизация в системе

Экранные формы:

- ЭФ_ЕСИА_Реестр: Реестр записей ЕСИА
- ЭФ_ЕСИА_Созд: Форма создания записи ЕСИА
- ЭФ_ЕСИА_Карт: Просмотр карточки записи ЕСИА
- ЭФ_ПЛ_Реестр: Реестр записей площадки
- ЭФ_ПЛ_Доб: Форма добавления записи на площадку
- ЭФ_ПЛ_Карт: Просмотр карточки записи площадки

Диаграммы вариантов использования:

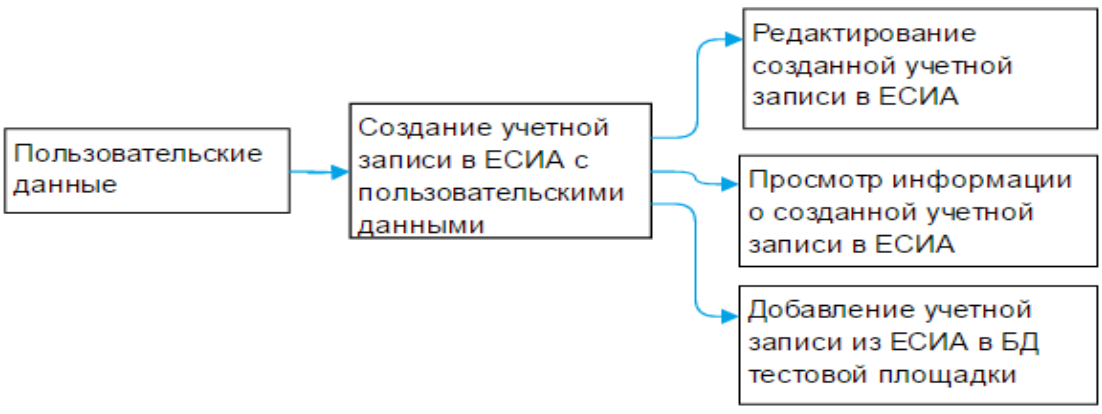

Рисунок 2.3 – Варианты использования с базой данных ЕСИА

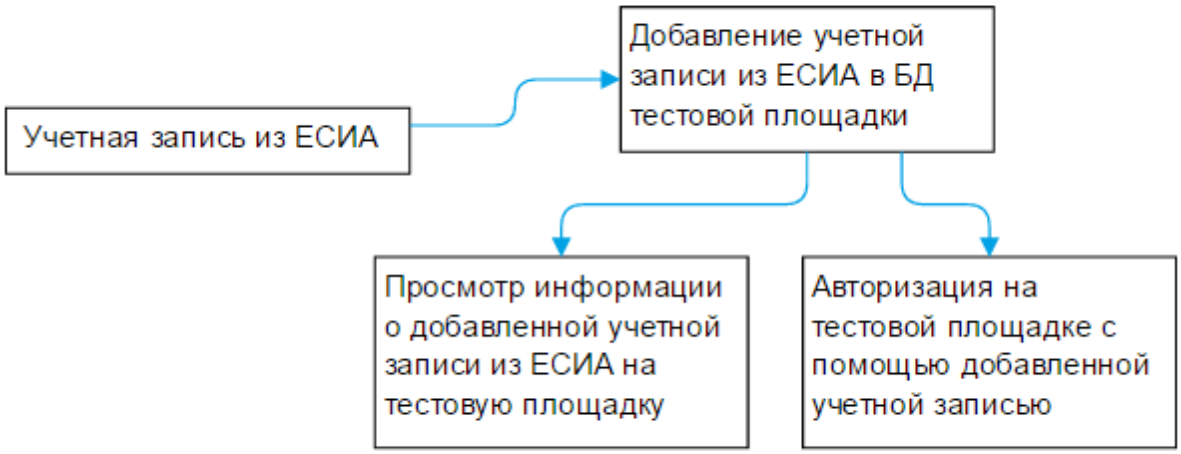

Рисунок 2.4 – Варианты использования с базой данных тестовых площадок.

Приведем пару примеров вариантов использования системы:

- ПЛ_4: Просмотр записи в реестре площадки
- ПЛ_1: Просмотр реестра записей площадки

ПЛ_4: Просмотр записи в реестре площадки:

Название сценария	**Просмотр записи реестра площадки**
Описание	Сценарий просмотра информации об учетной записи, ранее добавленной в базу данных площадки
Предусловия	Пользователь находится на странице ЭФ_ПЛ_Реестр: Реестр записей площадки Пользователь осуществил поиск записей по критериям
Инициирующее событие	Пользователь нажал в выпадающем меню шеврона «Просмотреть» на записи реестра
Результат	• Открывается карточка записи площадки (ЭФ_ПЛ_Карт) • Предоставлено сообщение о системной ошибке

Ограничения	Нет.			
№	**Действие**	**АС**	**М/ЭМ**	**С/ВИ /А**
Базовый сценарий. Система отображает форму реестра				
1.	Пользователь в выпадающем списке шеврона записи выбирает «Просмотреть»			
2.	Открывается на просмотр карточка записи площадки		ЭФ_ПЛ_Карт	
3.	Сценарий завершен.			
Альтернативный сценарий 1. Предоставлено сообщение о системной ошибке Если на шаге 1 возникла системная ошибка, то				
1.1	Система предоставляет сообщение об ошибке с описанием причин ошибки.			
1.2	Сценарий завершен.			

ПЛ_1: Просмотр реестра записей площадки

Название сценария	**Просмотр реестра записей площадки**
Описание	Сценарий просмотра информации об учетной записи, ранее добавленной в базу данных ЕСИА на определённой площадке
Предусловия	Пользователь зашел на любую страницу системы

Инициирующее событие	Пользователь из головного меню перешел в реестр записей площадки (ЭФ_МЕН)
Результат	• Отображается форма ЭФ_: Реестр записей площадки • Предоставлено сообщение о системной ошибке
Ограничения	Нет.

№	**Действие**	**АС**	**М/ЭМ**	**С/ВИ /А**
Базовый сценарий. Система отображает форму реестра				
4.	Система отображает форму ЭФ_ПЛ_Реестр: Реестр записей площадки		ЭФ_ПЛ_Реестр	
5.	Пользователь просматривает реестр записей ЕСИА			
6.	Сценарий завершен.			
Альтернативный сценарий 1. Предоставлено сообщение о системной ошибке Если на шаге 1 возникла системная ошибка, то				
1.3	Система предоставляет сообщение об ошибке с описанием причин ошибки.			
1.4	Сценарий завершен.			
Альтернативный сценарий 3. Пользователь выбрал пункт «Просмотреть» выпадающего меню записи площадки				
3.1	Система выполняет **Ошибка! Источник ссылки не найден.**ПЛ_2: Просмотр записи площадки			
3.2	Сценарий завершен.			

Альтернативный сценарий 4. Пользователь нажал кнопку «Найти» в блоке поиска				
4.1	Система выполняет ПЛ_2			
4.2	Сценарий завершен.			

Примеры экранных форм:

- ЭФ_ПЛ_Реестр: Реестр записей площадки
- ЭФ_ПЛ_Карт: Просмотр карточки записи площадки
- ЭФ_АВТ: Авторция

ЭФ_ПЛ_Реестр: Реестр записей площадки

Код эл-та	Название	Тип	Описание	По умолчанию	Ред.	Об.	Код ERD
ЭФ_ПЛ_Реестр.1	Список стендов	Выпадающий список	Список стендов проекта, доступный для выбора (критерий поиска записей)	STAND	+	+	
ЭФ_ПЛ_Реестр.2	Кнопка «Найти»	Кнопка	Кнопка, инициирующая поиск по заданным критериям				
ЭФ_ПЛ_Реестр.3	Шеврон записи площадки	Шеврон/Выпадающий список	Шеврон/выпадающий список со следующими пунктами: • Просмотр (инициирует ВИ_ПЛ_1)				

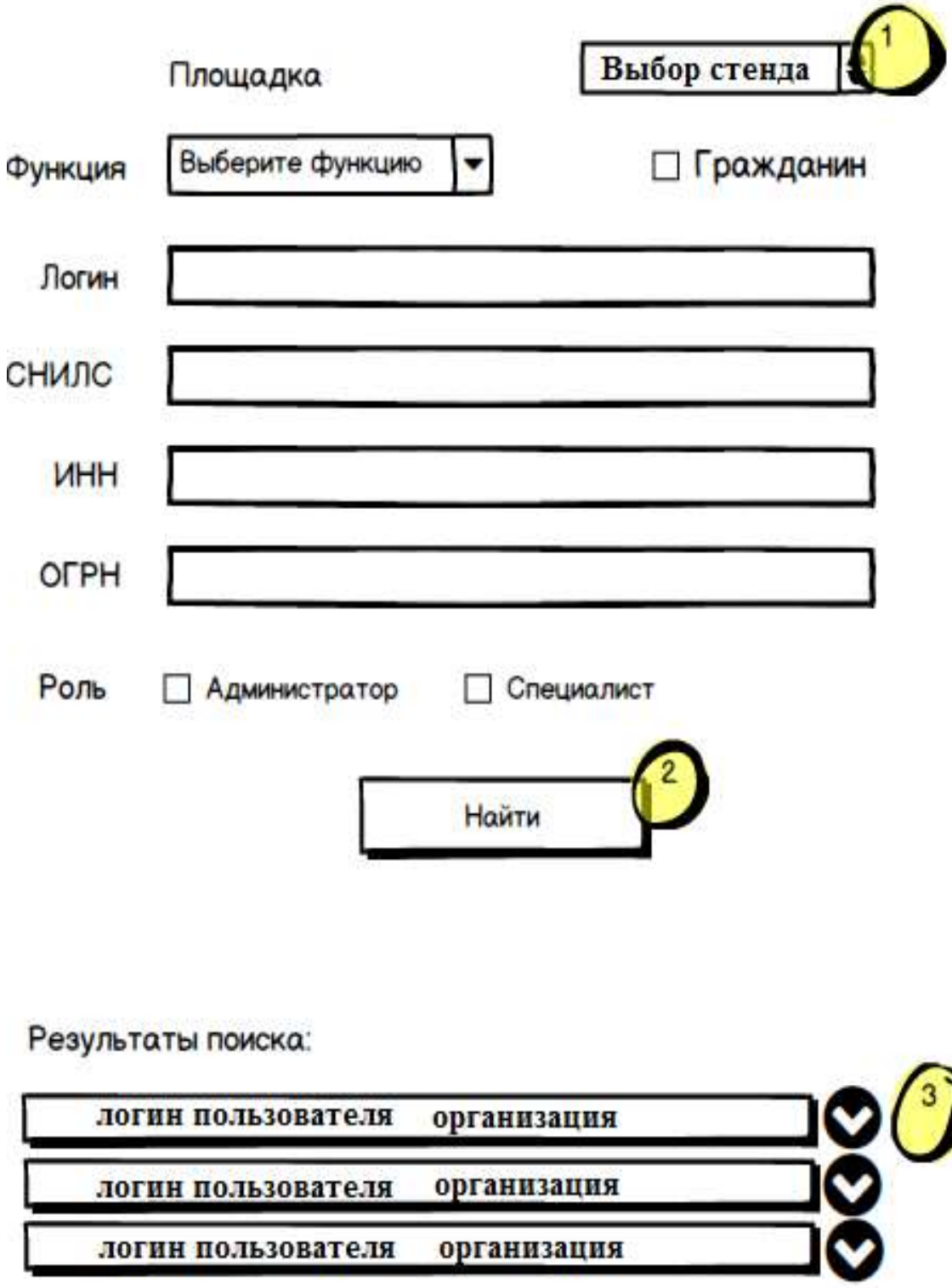

Рисунок 2.4 – Экранная форма ЭФ_ПЛ_Реестр: Реестр записей площадки

ЭФ_ПЛ_Карт: Просмотр карточки записи площадки

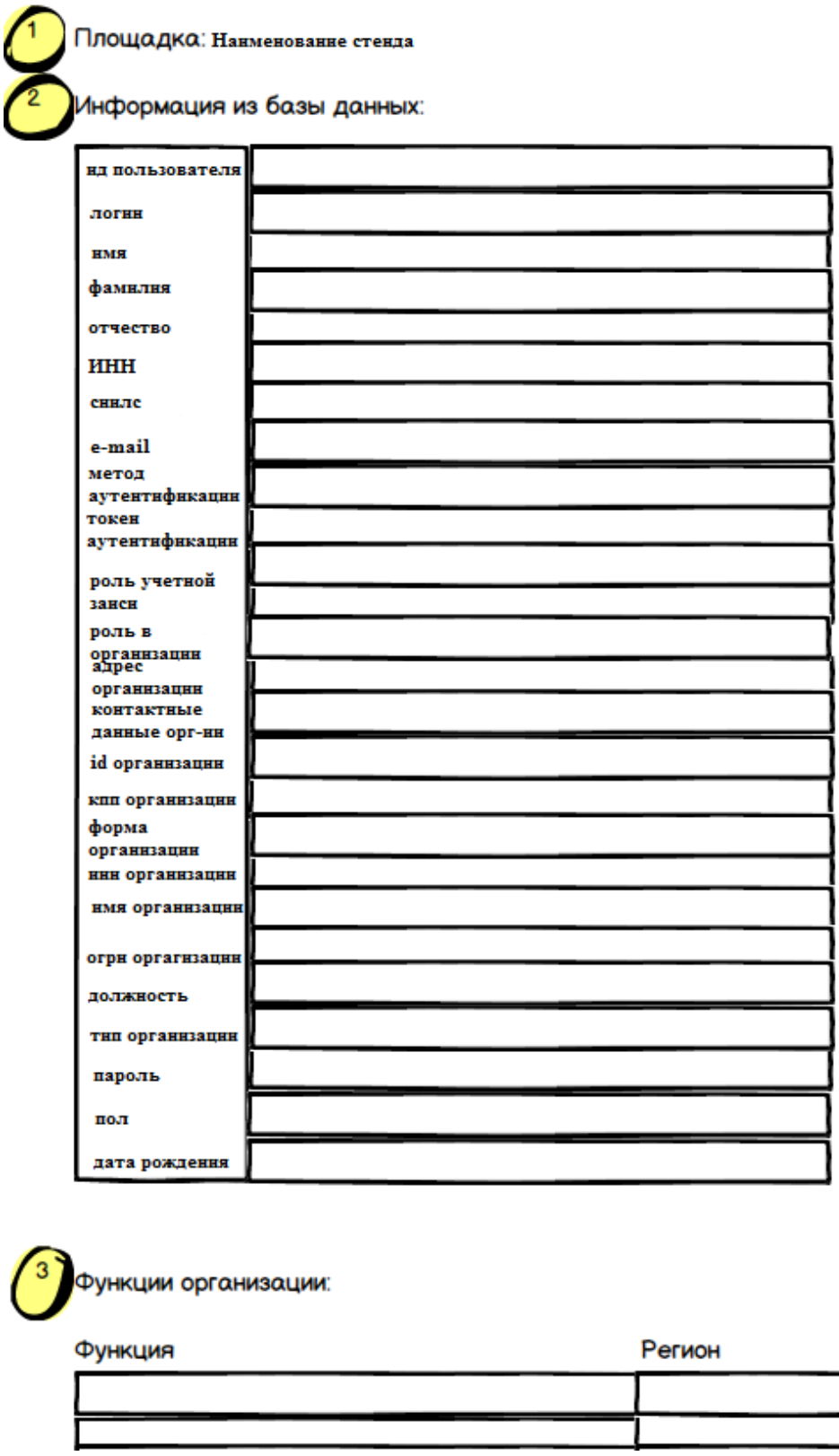

Рисунок 2.5 – Экранная форма ЭФ_ПЛ_Карт: Просмотр карточки записи площадки

Код эл-та	Название	Тип	Описание	По умолчанию	Ред.	Об.	Код ERD
ЭФ_ПЛ_Карт.1	Площадка	Текст	Информация о стенде, на котором взята информация	STAND			
ЭФ_ПЛ_Карт.2	Информация о записи	Текст	Информация из БД по записи, выбранной на просмотр				
ЭФ_ПЛ_Карт.3	Функции организации	Таблица	Функция и регион действия функции организации. В регионе действия функции указывается статус функции.				

ЭФ_АВТ

Рисунок 2.6 – Экранная форма ЭФ_АВТ: форма авторизации

Код эл-та	Название	Тип	Описание	По умолчанию	Ред.	Об.	Код ERD
ЭФ_АВТ.1	Поля ввода логина и пароля	Поля ввода	• Поля ввода логина и пароля для авторизации в системе				
ЭФ_АВТ.2	Кнопка «Войти»	Кнопка	Кнопка инициирующая автормзацию по веденым пользовательским данным				

3 РЕАЛИЗАЦИЯ И ВНЕДРЕНИЕ

3.1 Реализация разработанной системы

Руководствуясь требованиями безопасности, была разработана форма входа в утилиту. Каждый сотрудник может войти в систему, используя личный логин и пароль. Так же это обеспечивает контроль за системой, так как каждое действие с базой данных: добавление, редактирование записей отслеживается для каждого пользователя в соответствующей бд. На рисунке 3.1 приведет приме записи в бд:

id integer	actiondate timestamp without time zone	actiontype character varying(255)	current_state text	description text
21393	2018-06-11 11:29:46.194	STAND MANAGER CREATE		Создание организации на стенде на стенде STAND на основе данных записи ЕСИА:

esia_username character varying(255)	inn character varying(255)	kpp character varying(255)	ogrn character varying(255)	previous_state text	success boolean	userlogin character varying(255)
trekalo UL 25	, 1442264325, 2224132555	, 222401001,	, , 1095555502331		t	a.trekalo

Риснок 3.1 – пример записи в бд при добавлении организации

На рисунке 3.2 изображена форма входа в утилиту.

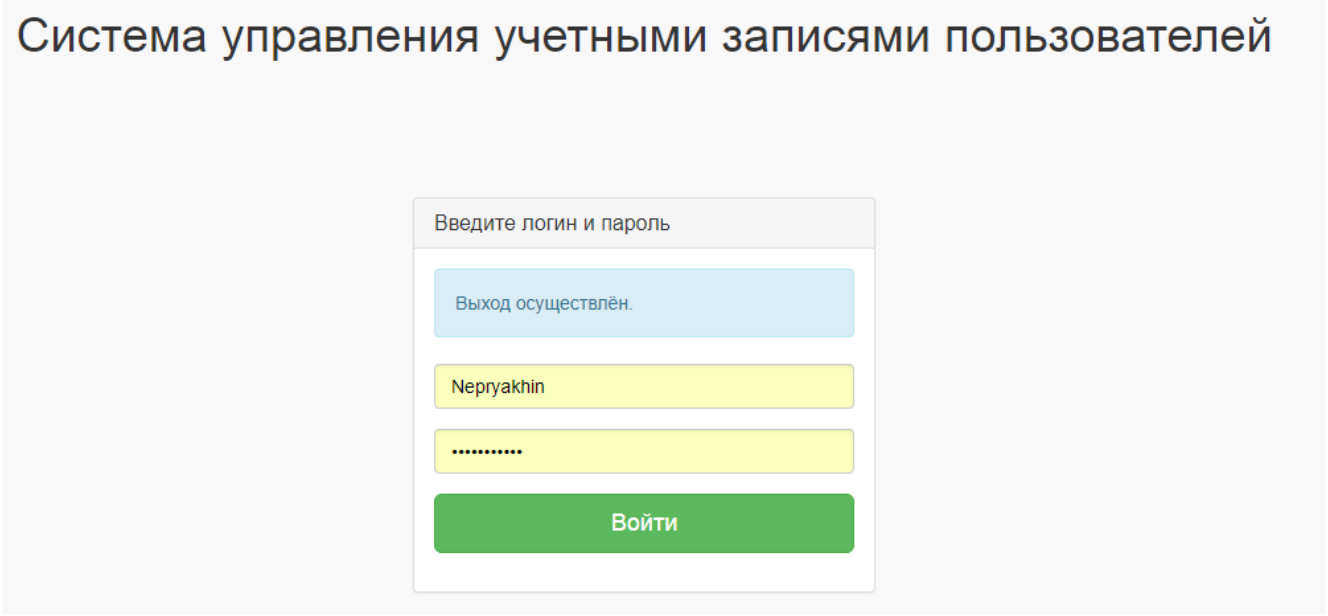

Рисунок 3.2 – Форма входа

После успешного входа в систему мы попадаем на главную страницу, где мы можем:

- Создать запись в тестовую ЕСИА
- Просмотреть запись ЕСИА и там же изменить ее
- Добавить запись ЕСИА на площадку(стенд)
- Просмотреть запись на площадке

- Выйти из учетной записи

На рисунке 3.3 изображено главное меню системы.

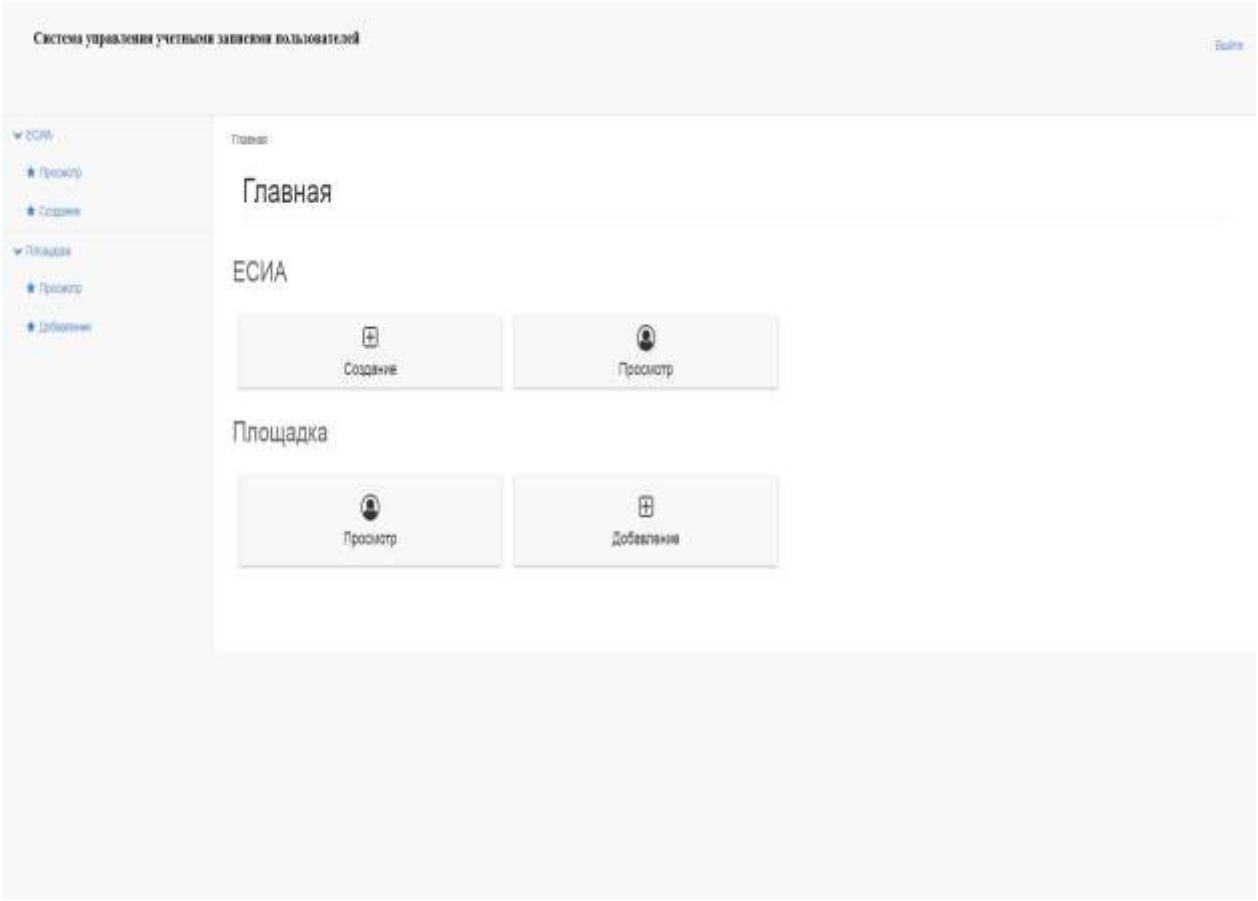

Рисунок 3.3 – Главное меню системы

Рассмотрим отдельно каждый пункт:

1) Создание записи в тестовой ЕСИА

Для начала необходимо выбрать тип записи, затем система покажет какие поля обязательные, а какие нет. Для добавления записи необходимо заполнить как минимум все обязательные поля. После заполнения полей необходимо нажать на кнопку «Создать». После этого запись попадает в баз данных тестовой ЕСИА. На рисунках 3.3 и 3.4 можно увидеть страницу добавления записи ЕСИА

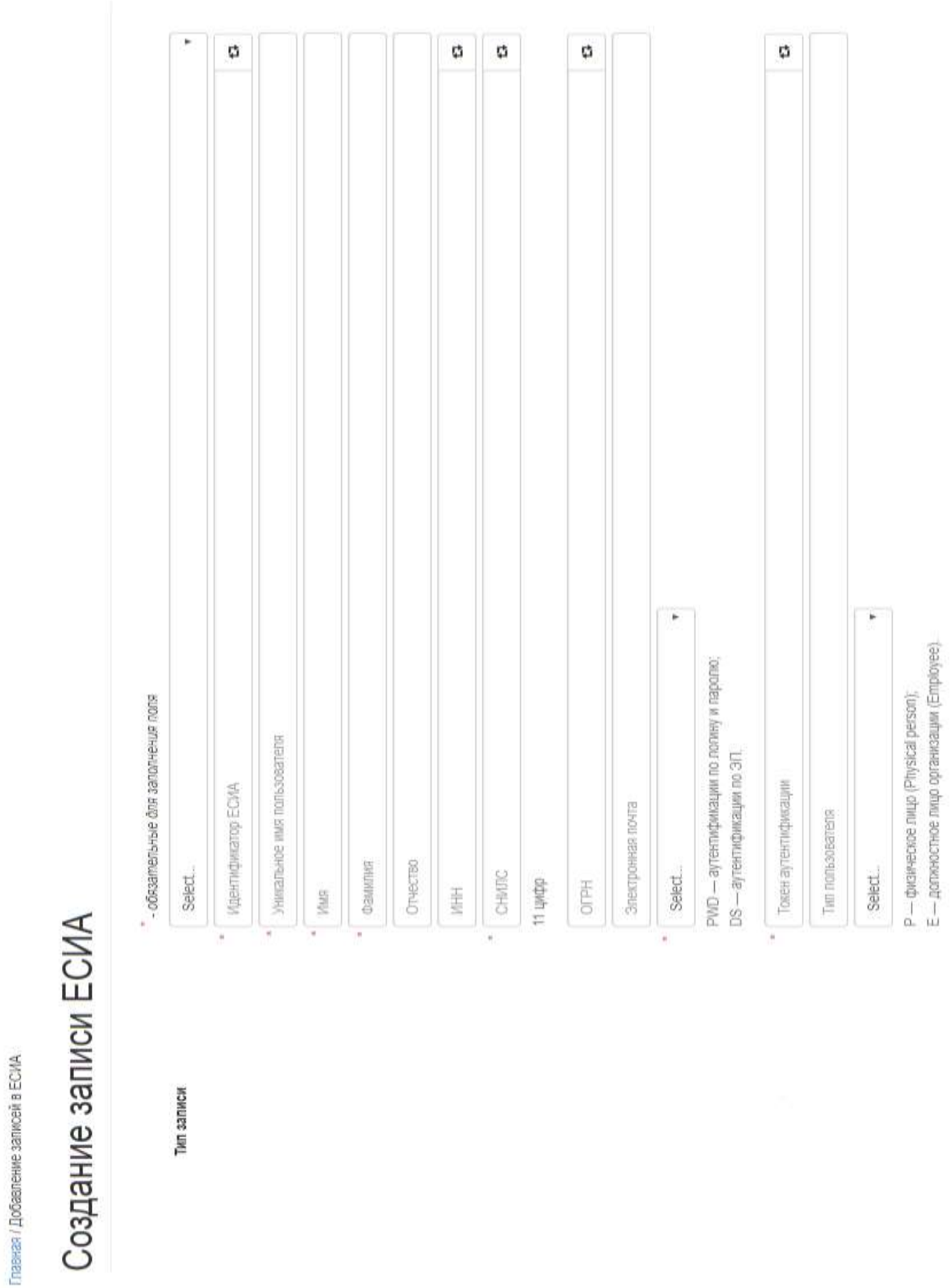

Рисунок 3.3 – Страница добавления записи ЕСИА 1 часть

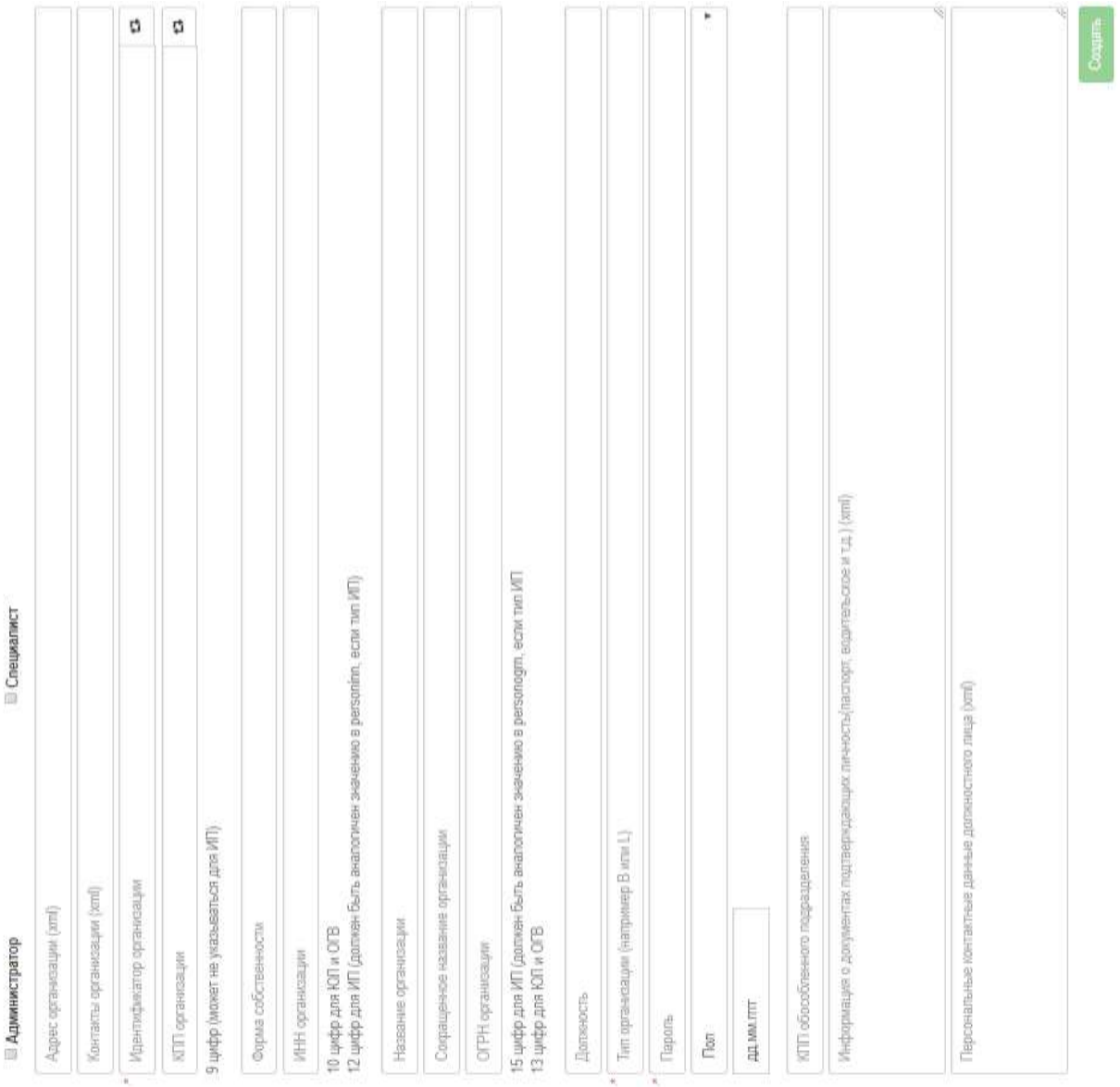

Рисунок 3.4 – Страница добавления записи ЕСИА 2 часть

2) Просмотр записи ЕСИА

Для просмотра записи следует ввести критерий поиска, который возможен по 6 различным атрибутам. После ввода нужных нам атрибутов нужно нажать на кнопку «Найти». Система выводит все записи, удовлетворяющие критериям поиска. Страница просмотра записи ЕСИА – рисунок 3.5.

Для найденной записи доступны сценарии просмотра и редактирования учетной записи. При вызове сценария просмотра записи, загружается информация из базы данных и отображается на экране. Пример такой записи можно увидеть на рисунке 3.6.

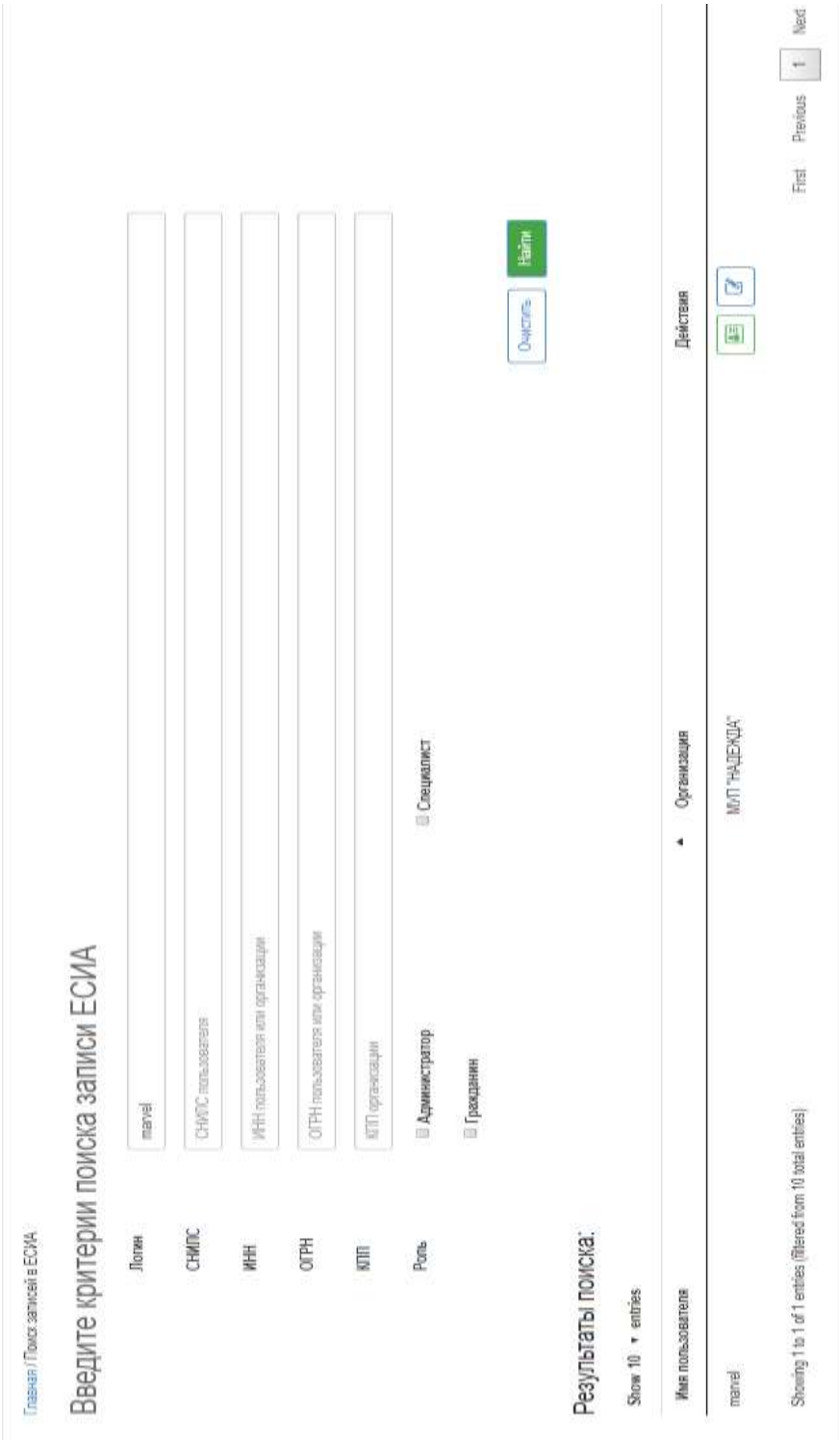

Рисунок 3.5 – Страница поиска записи ЕСИА

Рисунок 3.6 – Страница записи ЕСИА

Так же найденную запись можно редактировать, после редактирования записи, в базе данных значения полей заменятся на измененные данные. Процесс редактирования записи ЕСИА показан на рисунке 3.7

Рисунок 3.7 – Страница редактирования записи ЕСИА

3) Добавление записи на площадку

После создания записи в бд ЕСИА можно спокойно добавлять учетную запись на любой тестовый стенд. Для этого выбираем стенд, на который хотим добавить запись и логин пользователя ЕСИА, после чего поля автоматически заполняются значениями атрибутов учетной записи ЕСИА, далее нажимаем на кнопку «Добавить» и соответствующая запись создается в базе данных указанного стенда. Страница добавления записи на стенд показана на рисунке 3.8

Рисунок 3.8 – Страница добавления записи ЕСИА на площадку

4) Просмотр записи на площадке

Для просмотра записи первоначально нужно выбрать стенд, на котором требуется найти запись, затем следует ввести критерий поиска, который возможен по 6 различным атрибутам. После ввода нужных нам атрибутов нужно нажать на кнопку «Найти». Система выводит все записи, удовлетворяющие критериям поиска. Страница просмотра записей на площадке отображена на рисунке 3.9

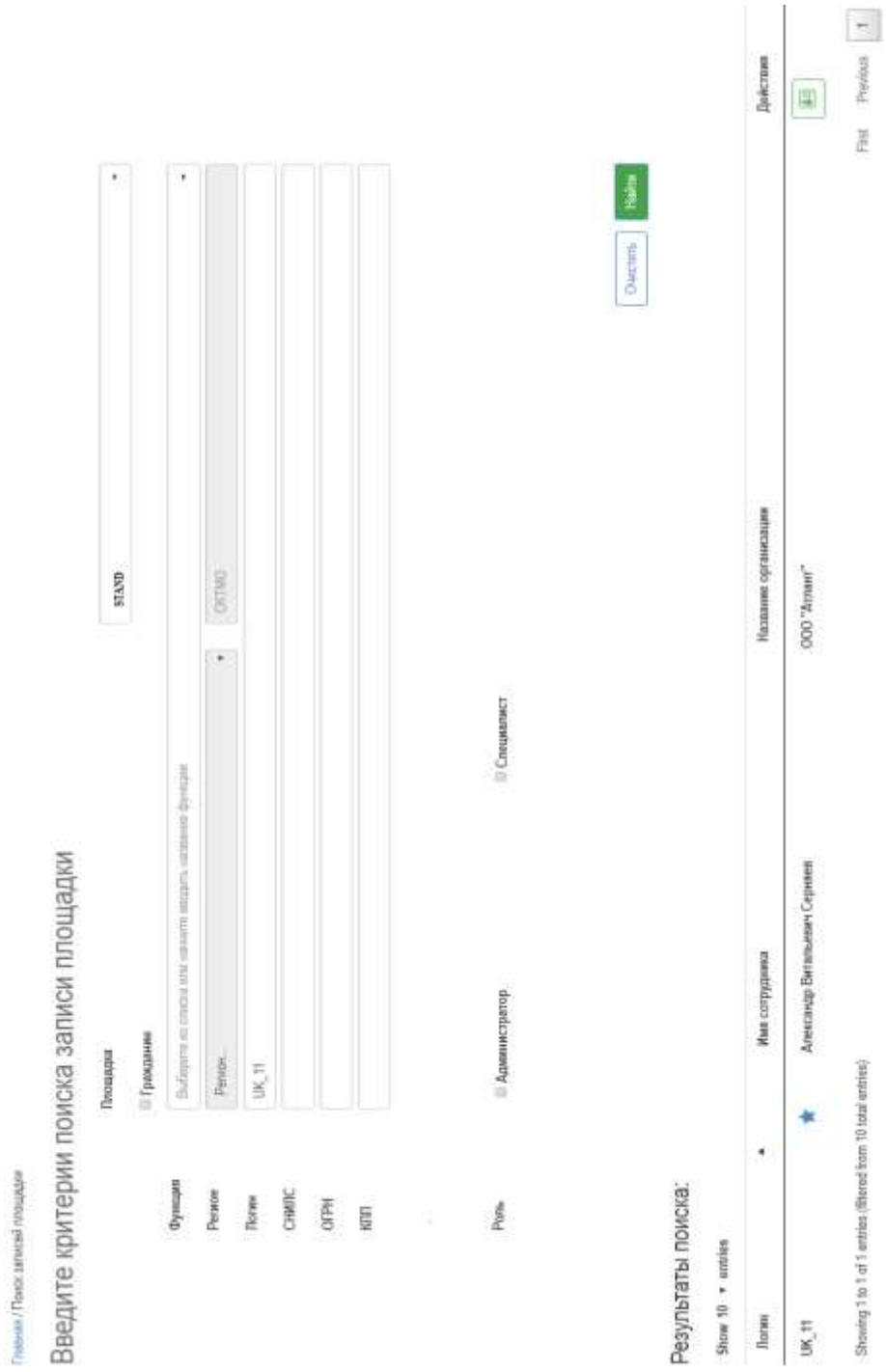

Рисунок 3.9 – Страница просмотра записей на площадке

3.2 Внедрение в производственный процесс

После разработки системы было произведено внедрение. В домене портала для утилиты был выделен отдельный адрес.

Запуску системы предшествует тестирование исполнителем системы. После получения положительных результатов начинается работа по реальному внедрению и запуску системы. Если опытная эксплуатация делается только на опытных примерах без участия рядовых исполнителей заказчика, без использования реальных задач, она не достигнет поставленной цели. Целью же является сбор замечаний, которые необходимо устранить для перевода в промышленную эксплуатацию. Данный этап правильнее было бы назвать расширенным тестированием. Реальная опытная эксплуатация начинается после внедрения системы при участии, как минимум, 50-70% процентах рабочих мест.

В процессе внедрения:

- Проведена опытная эксплуатация системы
- Была создана обучающая страница, в которой отображены основной функционал системы и инструкция по использованию.
- Проведено обучение сотрудников по использованию системы управления учетными записями пользователей.
- Произведен переход к эксплуатации спроектированной системы.

3.3 Инструкция по эксплуатации

1. Поиск пользователей.

Для осуществления поиска выбираем кнопку "Просмотр". Рисунок 3.3.

Для поиска на стенде ("Площадка" -> "Просмотр") организации или гражданина необходимо выбрать название стенда, после чего становятся доступными поля для задания параметров поиска пользователей:

а) по функции организации или выбор гражданина (отмечаем чек-бокс "Гражданин");

б) по определенному логину пользователя или значению СНИЛСа;

в) по конкретному ОГРН или КПП организации;

г) по коду региона или коду ОКТМО территории действия полномочий организации (в разработке);

д) по роли пользователя - администратор и/или уполномоченный специалист (если конкретная роль не важна, то отмечаем оба чек-бокса).

В результатах поиска отображаются все пользователи, удовлетворяющие заданным критериям. При нажатии на иконку в столбце "Действия" можно просмотреть подробную информацию по данному пользователю:

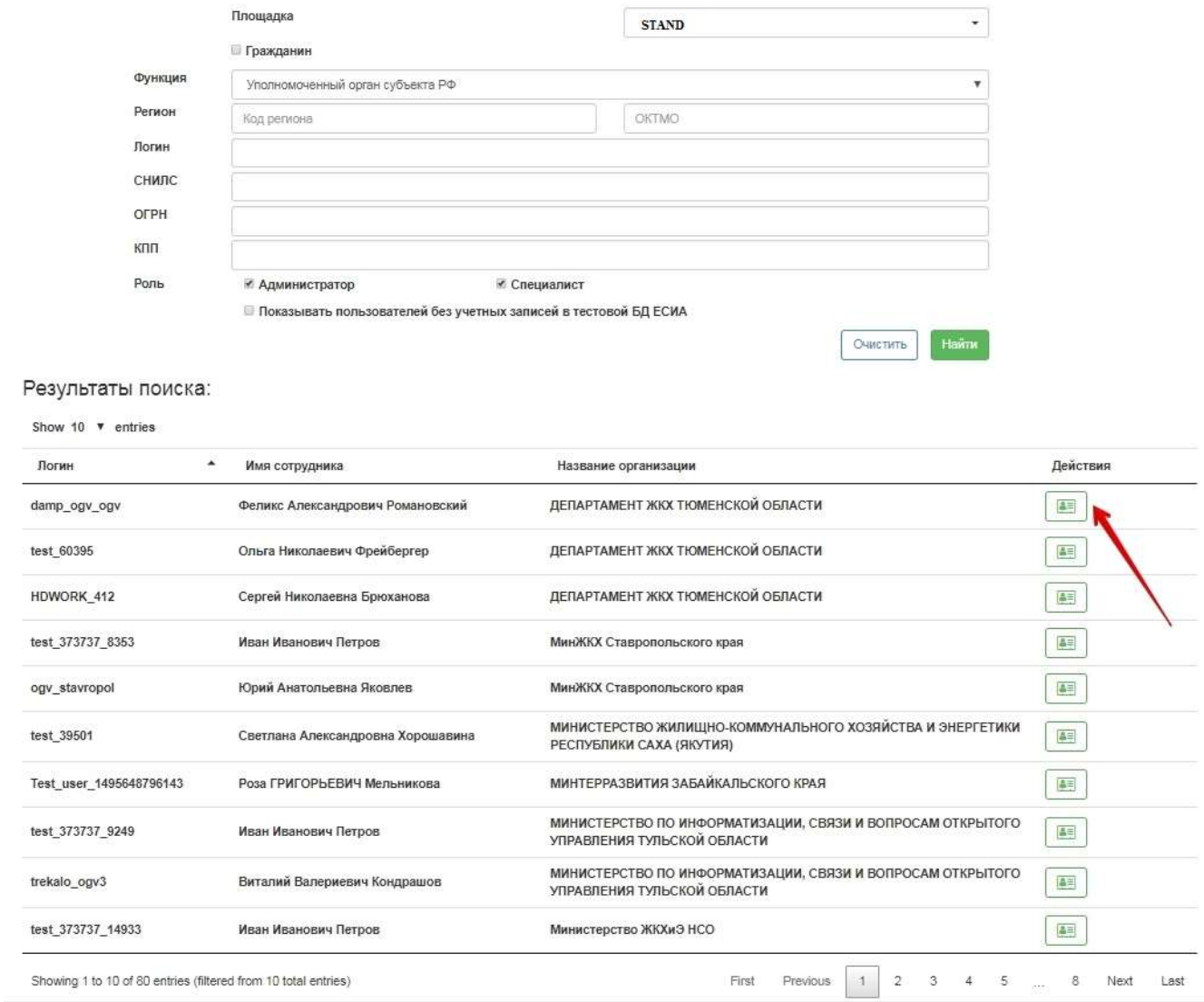

Рисунок 3.10 – Страница реестра записей на стенде

Также дополнительно можно выбрать отображение пользователей, не добавленных пока в БД ЕСИА (т.е. для работы под ними, необходимо будет добавить данные в эту БД). Таких пользователей можно найти на стендах с

дампом (PAK1,PAK2). Причём добавление этих пользователей можно произвести, выбрав соответствующие кнопки в результатах поиска:

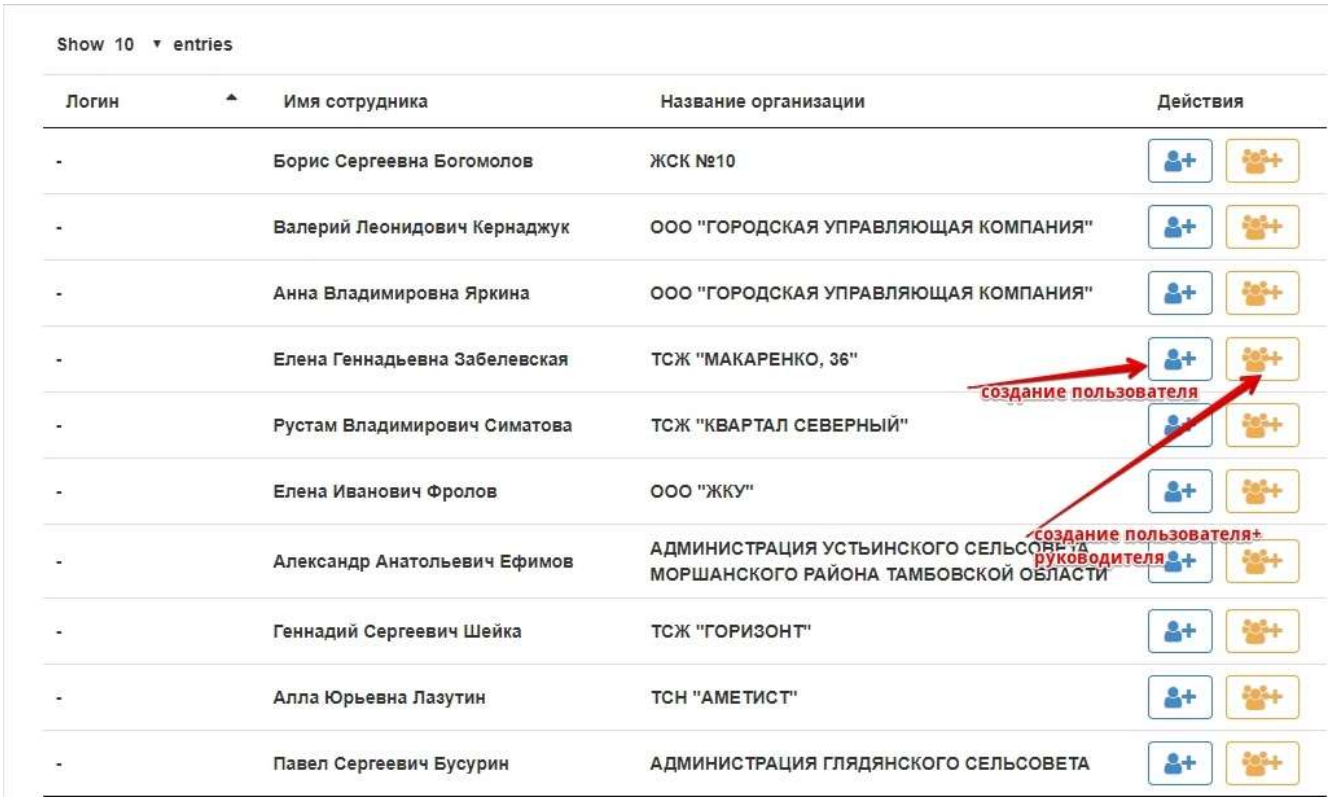

Рисунок 3.11 – Страница реестра записей на стенде

Использование этих кнопок позволяет добавить пользователя за один клик, в утилите появляется сообщение, содержащее логин созданного пользователя:

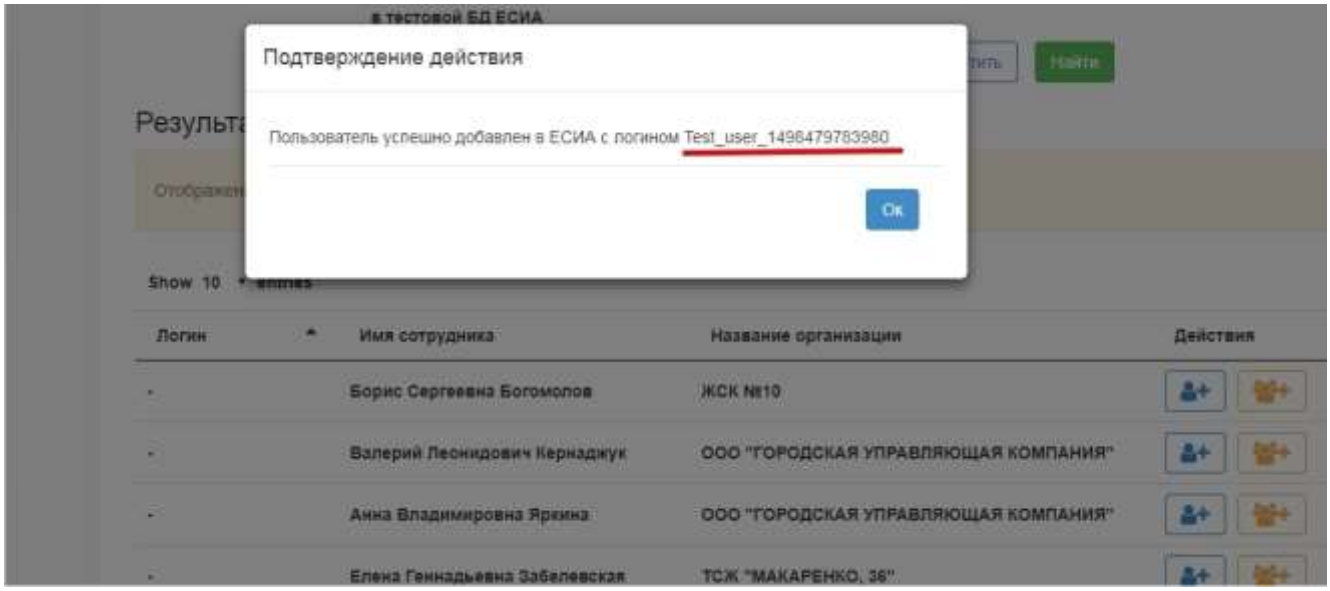

Рисунок 3.12 – Восстановление пользователя со стенда с дампом

после чего можно работать от имени этого пользователем на стенде с дампом.

Аналогично работает поиск и по БД ЕСИА ("ЕСИА" -> "Просмотр") с учетом того, что там нет сведений о функциях организаций.

Для найденных записей в БД ЕСИА кроме просмотра доступно также редактирование записи (к данной функции следует относиться внимательно и вносить изменения только, если Вы уверены в их необходимости).

2. Добавление пользователей.

Для добавления нового пользователя на стенд необходимо выбрать на главной странице утилиты плашку "Создание". Перед добавлением пользователя на стенд сперва его необходимо добавить в ЕСИА ("ЕСИА" -> "Создание").

а) Добавление в ЕСИА.

В первую очередь необходимо выбрать тип создаваемой записи. При этом существуют шаблоны для гражданина, ЮЛ, ОГВ, ИП, выбрав который автоматически сгенерируются значения для большинства полей, после чего останется только заполнить данные по логину, ФИО пользователя, наименованию, ОГРН, ИНН, КПП организации пользователя и выбрать роль (чаще всего стоит отметить обе роли - администратор+специалист). При этом стоит помнить, что следует создавать пользователей с уникальным логином и СНИЛС (при этом логин должен быть указан без использвания кириллицы), а добавляемые организации должны иметь уникальный ОГРН и ИНН (т.е. по этим параметрам -логин и СНИЛС в ЕСИА, а ОГРН, ИНН на стенде - поиск должен возвращать пустой результат).

При заполнении данных вручную утилита также позволяет автоматически генерировать значения для ряда полей (гуиды и ряд других), для чего необходимо нажать на значок обновления в правой части поля.

Если нам необходимо создать пользователя для уже существующей на стенде организации, то начиная с поля orgaddresses поля заполняем значениями этой организации, которые можно взять в результатах поиска соответствующей организации на стенде (необходимо скопировать значение этих полей).

б) Добавление на стенд.

После добавления записи в ЕСИА, можно переходить к добавлению пользователя на конкретный стенд. На главной странице для раздела "Площадка" нажимаем "Добавление". На открывшейся странице указываем стенд, на который мы хотим добавить пользователя и логин пользователя, после чего нажимаем "Загрузить":

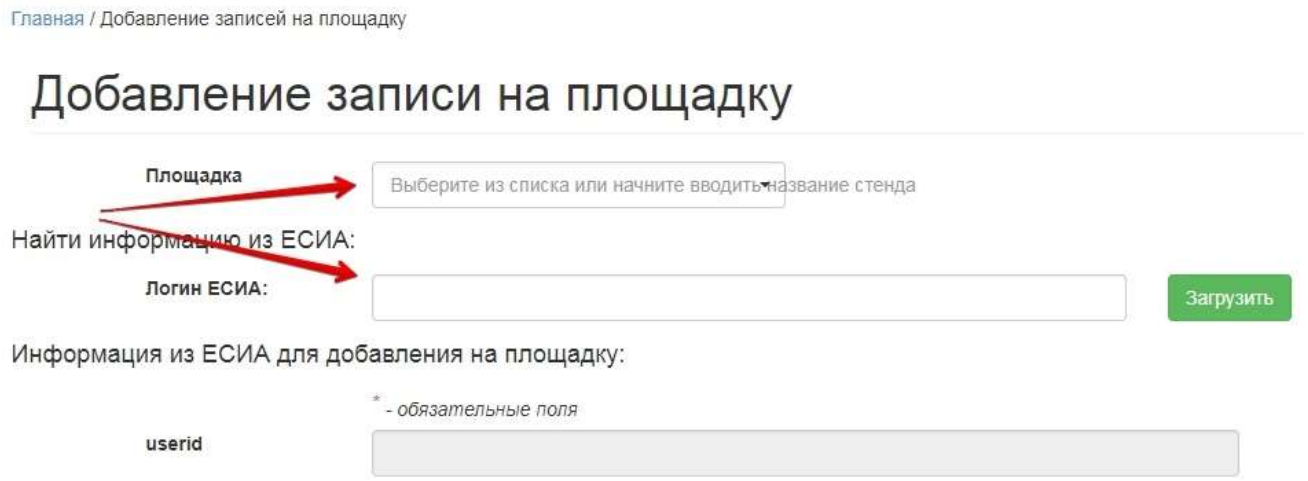

Рисунок 3.13 – Страница добавления записи ЕСИА на площадку

После этого поля таблицы автоматически заполнятся данными пользователя и организации из БД ЕСИА. Нам остаётся только нажать "Добавить", после чего появляется сообщение "Назначить текущего пользователя руководителем организации?", на которое, как правило, надо дать утвердительный ответ. После этого можно переходить к авторизации пользователя на соответствующем стенде.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работе были решены следующие задачи:

Произведен анализ существующего метода создания учетных записей пользователей, изучены связи таблиц в базе данных, определены недостатки и преимущества существующего метода.

Рассмотрены средства программирования, фреймворки для создания веб приложений, выбран стек технологий.

По требованиям компании разработано техническое задание, в котором описаны все экранные формы, варианты использования и алгоритмы системы.

Разработано веб приложение для управления учетными записями пользователя в соответствии с техническим заданием.

Проведена опытная эксплуатация системы, написано руководство по использованию утилиты, обучены специалисты отдела тестирования и поддержки, произведен переход к разработанной системе.

До внедрения системы создание учетной записи пользователя занимало у специалиста около 4-х минут, в день среднее количество создаваемых учетных записей равняется 8 штукам. Предположим, что в среднем рабочий час специалиста отдела тестирования стоит 200 рублей, на проекте 50 тестировщиков, получаем что 1 человек тратит на создание учетных записей около 32 минут за рабочий день, что приблизительно равняется 100 рублей на создание учетных записей для человека за 1 день. Посчитаем расходы за 1 день для всего отдела тестирования 100*50 = 5000 рублей в день. В месяце 22 рабочих дня, 22*5000 = 110000 рублей в месяц тратится на создание учетных записей. После внедрения системы создание учетной записи стало занимать 2 минуты, то есть в 2 раза меньше, экономия составляет 55000 рублей в месяц.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Выскорко, М.С. Регулярные выражения в Javascript / М.С. Выскорко – http://www.opennet.ru/base/dev/pcre_javascript.txt.html .

2 Голосов, А.О. Схемы реляционных баз данных: теория нормализации и построение нормальных форм: учебное пособие / А.О. Голосов, М.Ш. Цаленко – 2-е изд. – М.: Финансы и статистика, 1983. – с. 92-119.

3 Документация и информация о фреймворке angular.js. – https://angularjs.org

4 Зыль, С.А. Проектирование, разработка и анализ программного обеспечения: учебное пособие / С.А. Зыль – БХВ-Петербург, 2010 г. – 336 с.

5 Кантор, И. DOM: работа с HTML-страницей / И.Кантор – http://javascript.ru/tutorial/dom .

6 Конноли, Т. Базы данных: проектирование, реализация и сопровождение. Теория и практика: учебное пособие / Т. Конноли – 2-е издание. – М.: Издательский дом "Вильямс", 2000. – 1120 с.

7 Копейкин, М.В. Базы данных. Объектно-реляционный подход: учебное пособие / М.В. Копейкин, В.В. Спиридонов, Е.О. Шумаков – СПб.: СЗПИ, 1998. – 96 с.

8 Курняван, Б. Создание Web-приложений на языке Java с помощью сервлетов, JSP и EJB: методическое пособие / Б. Курнавян — М.: Лори, 2005. — 880 с.

9 Мартин, Р. Быстрая разработка программ. Принципы, примеры, практика: учебное пособие / Р. Мартин, Дж. Ньюкирк, Р. Косс — М.: Издательский дом «Вильямс», 2004. — 752 с.

10 Скотт, Ф.У. Принципы проектирования и разработки программного обеспечения: Сертификационный экзамен 70-100 / Ф.У. Скотт, А. Брюс – СПб.: Русская Редакция, 2002 г. – 736 с.