Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____О.В. Непомнящий
«___»_____2024 г.

БАКАЛАВРСКАЯ РАБОТА

090301 Информатика и вычислительная техника

Автоматизация процесса массового набора персонала

Руководитель	_____	_____	старший преподаватель	А.Г. Хантимиров
	подпись	*дата*	*должность, ученая степень*	
Выпускник	_____	_____		А.М. Джумаев
	подпись	*дата*		
Нормоконтролёр	_____	_____	старший преподаватель	А.Г. Хантимиров
	подпись	*дата*	*должность, ученая степень*	

Красноярск 2024

РЕФЕРАТ

Выпускная квалификационная работа по теме «Автоматизация процесса массового набора персонала», 50 страниц, 27 рисунков, 12 использованных источников, 6 приложений.

ВЕБ-ПРИЛОЖЕНИЕ, НАБОР ПЕРСОНАЛА, АНАЛИЗ.

Цель работы - разработка клиент-серверной платформы для массового набора персонала. Эта платформа будет способна отслеживать эффективность работы специалиста по найму. В дополнение к этому, работа исследует перспективы использования подобных систем в современной бизнес-сфере.

Структура работы включает в себя: введение, основную часть из 3-х глав, заключение, список использованных источников.

В первой главе работы производится анализ предметной области, охватывающий существующие сервисы для автоматизации процесса массового набора персонала.

Во второй главе производится выбор инструментов разработки, проектирование системы, выявлению прецедентов и их описанию, проектированию моделей для четырех акторов: гость, работник, руководитель, администратор, проектирование базы данных, а также проектирование интерфейса программы.

В третьей главе описана реализация системы и описание функциональной части системы.

В заключении подводятся итоги по выполненной работе.

Данная работа подразумевает разработку как клиентской, так и серверной части. Итогом выполнения ВКР является полностью рабочее веб приложение для автоматизации процесса массового набора персонала.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В мире, который быстро развивается, особенно в бизнес-сфере, компетентный и эффективный подбор персонала становится ключевым фактором успеха. Отбор квалифицированных специалистов — это сложная и трудоемкая задача, которая требует точности и значительного количества времени. Это время, которое не всегда учитывается в работе специалиста по отбору кандидатов.

Целью работы является разработка веб-приложения для автоматизации процесса массового набора персонала, которое будет способно оценивать эффективность специалистов по найму работников.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Провести анализ предметной области;
2. Выполнить проектирование системы для автоматизации процесса массового набора персонала;
3. Разработать программную реализацию системы для автоматизации процесса массового набора персонала.

Результатом работы является реализация указанного выше веб-приложения для автоматизации процесса массового набора персонала

1 Анализ предметной области

1.1 Основные понятия и определения

Автоматизация процесса массового набора персонала на сегодняшний день играет ключевую роль в сфере HR, включая использование современных информационных технологий для оптимизации и улучшения процедур подбора кандидатов на рабочие позиции. Основой этого подхода является создание функциональной онлайн-платформы с централизованной базой данных, где систематизированно и безопасно хранятся анкеты и документы всех соискателей, а также различные инструменты для автоматизированной обработки этих данных для эффективного подбора кадров.

Онлайн-платформа, которую можно охарактеризовать как веб-сервис или систему, предназначена для сбора, хранения и обработки информации о потенциальных кандидатах на вакансии. Эта платформа предоставляет быстрый и удобный доступ к базе данных соискателей для специалистов по подбору персонала и других уполномоченных лиц, которым необходимо принимать решения о найме. Также она даёт доступ к информации об эффективности специалистов, которые принимают участие в процессе, редактируют данные и вкладывают документы.

Решение о том, стоит ли принять или отклонить кандидата на рабочую позицию, специалисты по отбору принимают на основе тщательного анализа данных, содержащихся в анкетах соискателей. Для этого они используют предоставленные им инструменты и системы, что значительно упрощает и ускоряет процесс принятия решений о приеме на работу.

1.2 Постановка требования при выполнении задачи

Целью данной бакалаврской работы является разработка системы автоматизации процесса массового подбора персонала. Разработанная система должна

обеспечивать хранение резюме соискателей, удобный в использовании интерфейс, функцию отправки резюме на следующий этап для дальнейшего согласования с руководителем, инструменты для анализа руководителю чтобы оценить эффективность специалиста по найму, аутентификация пользователей.

Система должна удовлетворять следующим требованиям**:**

1. Система должна быть подключена к базе данных, где хранятся резюме соискателей;
2. Интерфейс, в котором специалист может просмотреть резюме и отправить ее на следующий этап или отклонить;
3. Наличие инструментов для анализа эффективности работы специалистов по подбору персонала;
4. Аутентификация пользователей.

Система должная выполнять следующие задачи:

1. Обеспечивать удобный и безопасный доступ к резюме соискателей, которые сохраняются в базе данных;
2. Ограничивать доступ обычного работника к функциям руководителя, включая инструменты для анализа и согласования заявок;
3. Предоставлять удобный и эффективный интерфейс для хранения документов и данных соискателя;
4. Отслеживать выполненную работу специалиста для последующего анализа.

1.3 Цель создания системы

Цель создания этой системы - определить ее преимущества и ограничения, а также проанализировать, как система улучшает процесс подбора персонала по сравнению с традиционными методами.

1.4 Функциональные возможности

Система должна обладать следующими функциональными возможностями:

1. Создание карточки для хранения данных о кандидатах;
2. Аутентификация пользователей, чтобы каждый имел свою доску с текущими резюме;
3. Удобное редактирование данных для дополнения информации о карточке;
4. Инструменты для анализа эффективности специалиста по найму руководителем;
5. Изменение статуса карточки для отслеживания их текущего этапа.

1.5 Перечень ограничений

Система обязана соблюдать следующие ограничения:

1. Нельзя хранить данные в облачном сервисе;
2. Все инструменты для разработки должны быть в открытом доступе;
3. Все файлы, которые загружают пользователи должны храниться в локальной папке;

1.6 Анализ существующих решений

1.6.1 E-Staff

Главная страница E-Staff представлена на рисунке 1.

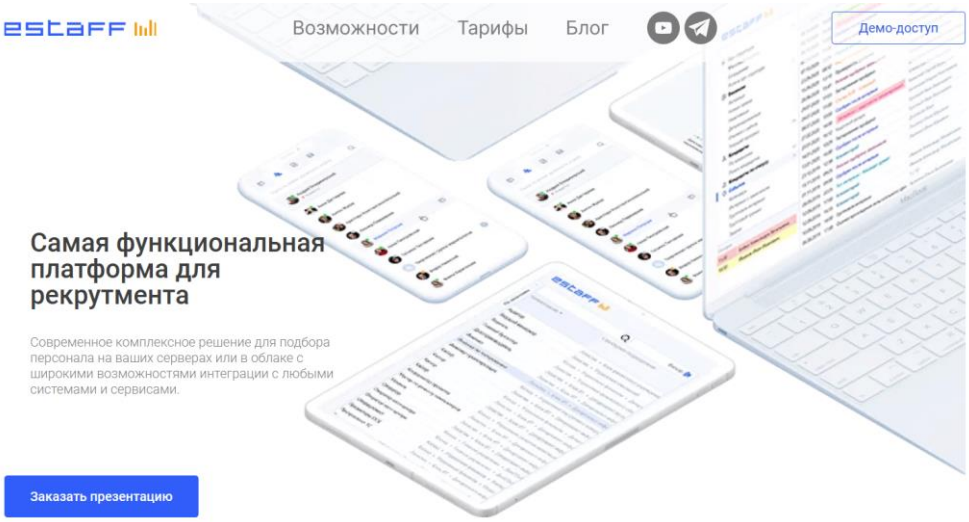


Рисунок 1 - Главная страница E-Staff

E-Staff[5] представляет собой онлайн-платформу, специально разработанную для управления и автоматизации различных процессов подбора персонала. Эта уникальная система предоставляет внушительный набор инструментов для эффективного управления вакансиями, кандидатами, организации интервью, а также для проведения глубокого анализа данных и управления другими ключевыми аспектами, которые напрямую связаны с рекрутингом.

Несмотря на множество преимуществ, которые может предложить E-Staff, у этой платформы также есть и определенные недостатки.

– стоимость: В зависимости от конкретного объема услуг и функций, которые выбираются пользователем, стоимость пользования данной платформой может оказаться достаточно высокой;

– безопасность: несмотря на широкий спектр возможностей, которые предлагает E-Staff, он не подходит по причине того, что те или иные файлы хранятся в облачном сервисе, из-за чего есть риск взлома системы с последующей утечкой документов.

1.6.2 Skillaz

Главная страница Skillaz представлена на рисунке 2.

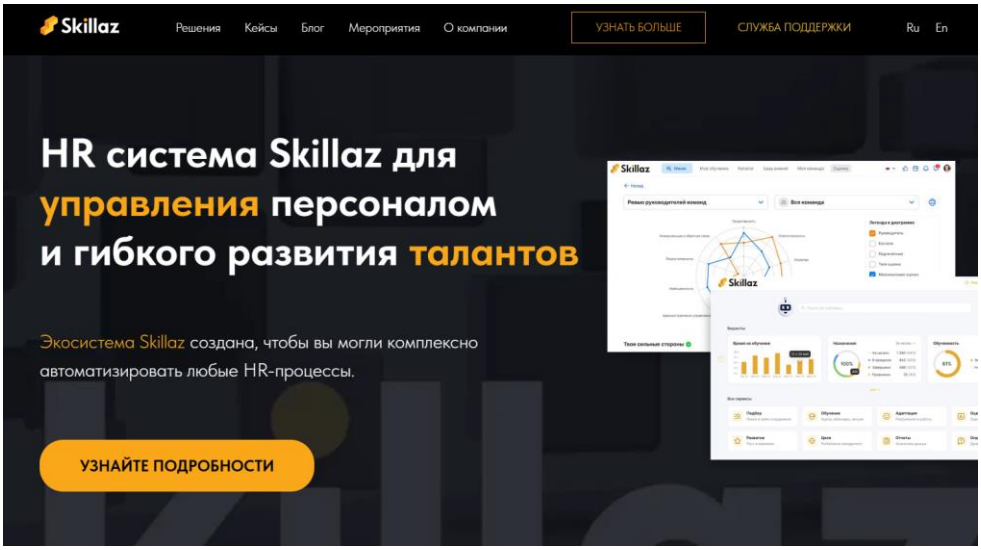


Рисунок 2 – Главная страница Skillaz

Skillaz[6] представляет собой инновационную онлайн-платформу для поиска работы и подбора персонала, которая активно использует принципы искусственного интеллекта и аналитики данных. Пользователи этой уникальной платформы имеют возможность создавать собственные профили, активно искать подходящие вакансии, проходить различные тестирования и участвовать в процессе подбора потенциальных кандидатов на работу.

Вместе с тем, несмотря на множество преимуществ, которые предлагает Skillaz, у данной платформы также есть определенные недостатки. В частности:

– безопасность: несмотря на широкий спектр возможностей, которые предлагает Skillaz, он не подходит по причине того, что те или иные файлы хранятся в облачном сервисе из-за чего есть риск взлома системы с последующей утечкой документов.

– недостаток персонального подхода: поскольку платформа основана на алгоритмах, она иногда может упускать важные человеческие аспекты, которые могут быть ключевыми при подборе персонала или поиске работы;

– цена: Базовые тарифные планы могут начинаться от нескольких сотен до нескольких тысяч долларов в месяц или за год. Однако конкретные цены обычно обсуждаются индивидуально с представителями компании Skillaz и могут быть

скорректированы в соответствии с потребностями и запросами конкретного бизнеса.

1.7 Вывод по разделу

В ходе анализа выяснилось, что рассмотренные варианты автоматизации процесса массового подбора персонала не идеально подходят для данной задачи. В них могут быть неточности из-за использования искусственного интеллекта и ограниченности функционала. К тому же, эти системы обходятся в большую сумму, что не отвечает запросам заказчика.

Анализ предметной области позволил определить основные требования к разрабатываемой системе автоматизации процесса массового подбора персонала. Необходимо создать онлайн-платформу с удобным интерфейсом, позволяющим хранить и согласовывать резюме кандидатов, а также хранить все данные, с которыми работает специалист по найму. Система должна также иметь инструменты для анализа работы специалиста по найму кандидатов и самой системы, чтобы оценить эффективность работы как специалиста, так и системы.

2 Проектирование

2.1 Общая структура разрабатываемой системы

Общая структура разрабатываемой системы представляет собой клиент-сервер архитектуру. Была выбрана трехуровневая архитектура, состоящая из клиента, сервера и базы данных.

Визуализация трехуровневой архитектуры представлена на рисунке 3.

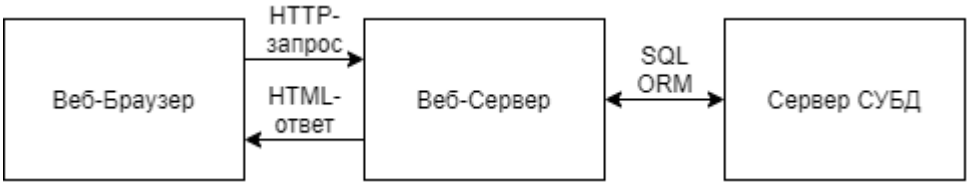


Рисунок 3 – Визуализация трехуровневой архитектуры

В этой архитектуре пользователь взаимодействуя с интерфейсом посылает на сервер запросы, после чего сервер их обрабатывает и позволяет взаимодействовать с базой данных без прямого доступа к ней, отображая при этом на клиенте ответ который он получил от сервера.

Ключевая идея здесь заключается в том, что запросы от потенциально неограниченного числа клиентов направляются на специально подготовленные служебные машины, известные как серверы. В нашем случае, для упрощения управления и обслуживания, мы будем использовать один сервер.

Преимущества такого подхода многочисленны:

– он предлагает централизованное управление данными через один сервер, что облегчает мониторинг и обслуживание;

– он обеспечивает высокий уровень информационной безопасности, поддерживая четкое разделение между пользовательскими и системными данными;

– он повышает производительность работы с общими ресурсами, благодаря оптимизации обработки запросов;

– наконец, он обеспечивает масштабируемость, позволяя свободно увеличивать количество клиентов и серверов по мере роста системы.

Однако стоит отметить и некоторые недостатки клиент-серверной архитектуры.

– риск отказа единственного сервера, что может оказать серьезное влияние на доступность системы;

– существует вероятность перегрузки сервера при слишком большом количестве клиентов, что может привести к снижению производительности.

2.2 Выбор средств и технологий разработки

2.2.1 Nest.js

Логотип Nest.js представлен на рисунке 4.

Рисунок 4 – Логотип Nest.js

Nest.js[7] представляет собой современный фреймворк для создания серверных приложений на базе TypeScript и Node.js. Nest.js стремится обеспечить разработчикам интуитивно понятные инструменты для создания масштабируемых и надежных серверных приложений.

Фреймворк также поддерживает TypeScript, что обеспечивает статическую типизацию и помогает избежать множества распространенных ошибок на этапе разработки.

2.2.2 TypeScript

Логотип Typescript представлен на рисунке 5.

Рисунок 5 –Логотип Typescript

TypeScript[8] представляет собой язык программирования, созданный на базе JavaScript и предоставляющий разработчикам возможность добавления строгой типизации в код. Он становится все более популярным благодаря нескольким ключевым аспектам:

Совместимость с экосистемой JavaScript позволяет использовать существующий код JavaScript в проектах на TypeScript и наоборот. Это облегчает переход на новый язык и обеспечивает гибкость в выборе подхода к разработке.

2.2.3 PostgreSQL

Логотип PostgreSQL представлен на рисунке 6.

Рисунок 6 – Логотип PostgreSQL

PostgreSQL[9] — это мощная и расширяемая объектно-реляционная система управления базами данных (СУБД), которая известна своей надежностью, функциональностью и возможностью обработки больших объемов данных.

PostgreSQL доступен абсолютно бесплатно и с открытым исходным кодом, что позволяет пользователям использовать его без лицензионных ограничений. Более того, он поддерживает различные операционные системы, что обеспечивает широкую доступность для разработчиков и пользователей.

2.2.4 Prisma

Логотип Prisma представлен на рисунке 7.

Рисунок 7 – Логотип Prisma

Prisma[10] — это современный инструмент для работы с базами данных, с открытым исходным кодом предназначенный для языков Javascript и Typescript. Оно позволяет работать с реляционными базами данных без использования SQL, что значительно упрощает работу.

Prisma имеет при себе удобный интерфейс в котором администратору очень удобно взаимодействовать с базой данных не привязываясь к приложениям и дополняя стек.

2.2.5 Vue.js

Логотип Vue.js представлен на рисунке 8.

Рисунок 8 – Логотип Vue.js

Vue.js[11] - это популярный JavaScript-фреймворк для построения интерфейсов, который позволяет создавать как простые, так и сложные веб-приложения. Его простой и гибкий API делает разработку удобной как для новичков, так и для опытных разработчиков.

В целом, Vue.js представляет собой мощный инструмент для создания современных веб-приложений, который отличается простотой использования и гибкостью.

2.2.6 Docker

Логотип Docker представлен на рисунке 9.

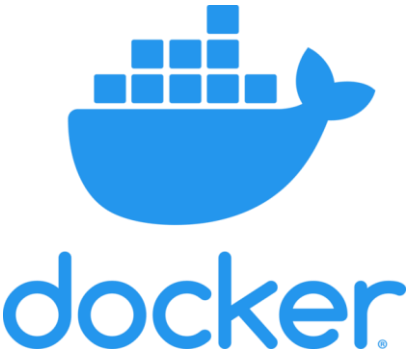


Рисунок 9 – Логотип Docker

Для того чтобы управлять сервером, клиентом, база данных и ORM, следует использовать Docker[12] – платформу которая позволяет упаковать все эти сервисы в контейнеры, для меньшего потребления ресурсов компьютера. Контейнер – это формат пакетирования, который состоит из операционной системы и кода, который запускается при включении контейнера.

Использование Docker контейнеров удобно тем, что это гарантирует одинаковую работоспособность в любой системе, независимо от различий в среде.

2.3 Диаграммы прецедентов

На основании анализа функционального требования к системе, были выделены следующие акторы:

- Гость;
- Пользователь;
- Руководитель;
- Администратор.

Диаграммы прецедентов показывают, какие функции доступны каждому из акторов.

2.3.1 Диаграммы прецедентов актора «гость»

Описание для диаграммы прецедентов актора «гость» (Рисунок 10).

Рисунок 10 – диаграмма прецедентов для «гостя»

Название прецедента: Войти в систему.

Цель сценария: Войти в систему под своим логином и паролем.

Предусловие: Открыто окно веб-сайта и «Администратор» выдал данные учетной записи пользователя.

Основной сценарий: Пользователь вводит логин и пароль, после чего нажимает кнопку войти.

Постусловие: Выполнен вход в систему.

2.3.2 Диаграмма прецедентов актора «пользователь»

У пользователя появляются новые прецеденты. Описание для диаграммы прецедентов актора «пользователь» (Рисунок 11):

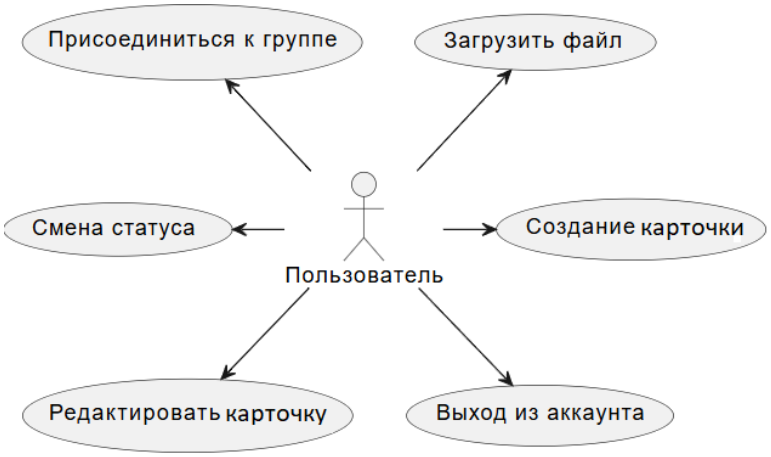


Рисунок 11 – Диаграмма прецедентов для «работника»

Название прецедента: Создание карточки.

Цель сценария: Создание карточки для резюме соискателя.

Предусловие: Открыт главное окно веб-сайта и выполнен вход в систему.

Основной сценарий: Пользователь нажимает на кнопку создания карточки и вводит данные, после чего нажимает на кнопку «Создать».

Постусловие: В базе данных и у клиента появляется новая запись с этой карточкой. Пользователь подключается к рабочей группе, только что созданной

карточки, в записях истории изменений и базе данных появляется запись о создании карточки и подключении к группе пользователем.

Название прецедента: Редактировать карточку.

Цель сценария: Редактирование данных карточки.

Предусловие: Карточка просматривается.

Основной сценарий: Пользователь нажимает на кнопку для переключения в режим редактирования, далее он выбирает поле, которое нужно изменить и пишет новые данные, после чего нажимает на кнопку «Сохранить».

Постусловие: В базе данных и у клиента редактируется данные карточки. В записях истории изменений и базе данных появляется запись о редактировании полей карточки пользователем.

Название прецедента: Смена статуса.

Цель сценария: Смена статуса карточки для согласования с руководителем.

Предусловие: Открыта карточка.

Основной сценарий: Пользователь нажимает на кнопку «Сменить статус» и из выпадающего списка выбирает подходящий статус, после чего пользователь подтверждает свой выбор.

Постусловие: В базе данных у карточки меняется статус, эта карточка появляется у руководителя для согласования. В записях истории изменений и базе данных появляется запись о смене статуса карточки пользователем.

Название прецедента: Загрузить файл.

Цель сценария: Загрузить документ в карточку.

Предусловие: Карточка редактируется.

Основной сценарий: Пользователь нажимает на кнопку для загрузки документа и выбирает файл, который нужно загрузить, после чего нажимает кнопку «Сохранить».

Постусловие: В базе данных и у клиента появляется документ в списке документов карточки. В записях истории изменений и базе данных появляется

запись о загрузке файла пользователем. Создаётся запись в таблице документов, с ссылкой на файл на сервере, которая привязывается к карточке.

Название прецедента: Присоединиться к группе.

Цель сценария: Присоединиться к группе, для отметки о том, что пользователем работает над карточкой.

Предусловие: Открыта карточка.

Основной сценарий: Пользователь нажимает на кнопку «Добавиться в группу»

Постусловие: В записях истории изменений и базе данных появляется запись о добавлении в рабочую группу карточки пользователем.

Название прецедента: Выход из аккаунта.

Цель сценария: Выход из аккаунта.

Предусловие: Выполнен вход в систему

Основной сценарий: Пользователь нажимает на кнопку «Выход», после чего подтверждает свой выбор.

Постусловие: Пользователь выходит из системы, теперь он гость.

2.3.3 Диаграмма прецедентов актора «руководитель»

Новые прецеденты для актора «руководитель» (Рисунок 12):

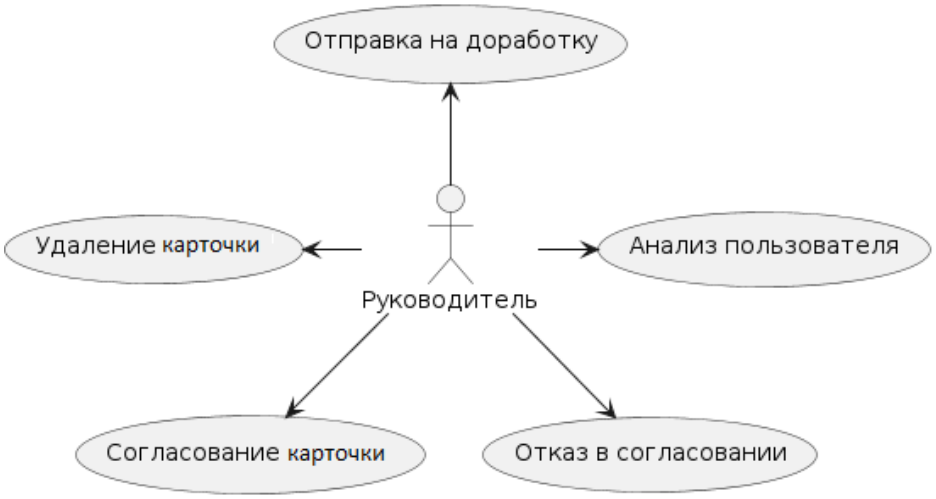


Рисунок 11 – Диаграмма прецедентов для «руководитель»

Название прецедента: Анализ пользователя.

Цель прецедента: Просмотреть эффективность пользователя по графикам.

Предусловия: Выполнен вход в систему под руководителя.

Основной сценарий: Руководитель нажимает на вкладку «Анализ» в главном окне веб-сайта и выбирает нужного для анализа пользователя.

Постусловия: Руководитель просматривает отчетность о работе пользователя.

Название прецедента: Удаление карточки.

Цель прецедента: Удаление карточки.

Предусловия: Карточка которую надо удалить просматривается.

Основной сценарий: Руководитель нажимает на кнопку «Сменить статус» и во всплывающем окне выбирает статус «Удалён», после чего подтверждает свой выбор.

Постусловие: В базе данных и у всех пользователей удаляется запись с карточкой, а также все данные, связанные с этой карточкой.

Название прецедента: Согласование карточки.

Цель прецедента: Согласовать резюме для смены статуса в последующий этап.

Предусловия: Карточка которую надо согласовать просматривается.

Основной сценарий: Руководитель нажимает на кнопку «Сменить статус» и во всплывающем окне выбирает следующий статус, после чего подтверждает свой выбор.

Постусловие: В базе данных и у всех пользователей, Карточка меняет свой статус, на следующий статус. В записях истории изменений и базе данных появляется запись о смене статуса карточки руководителем.

Название прецедента: Отказ в согласовании.

Цель прецедента: Отказать в согласовании резюме, для последующего переноса карточки во вкладку отказано.

Предусловия: Карточка которую надо согласовать просматривается.

Основной сценарий: Руководитель решает, что резюме не подходит нуждам компании. Руководитель нажимает на кнопку «Сменить статус» и во всплывающем окне выбирает статус «Отказано», после чего подтверждает свой выбор.

Постусловие: В базе данных и у всех пользователей, карточка меняет свой статус, на «Отказано». В записях истории изменений и базе данных появляется запись о смене статуса карточки руководителем, а также появляется запись о дате отказа данной карточки, которая будет удалена через месяц, или же восстановлена в случае необходимости.

Название прецедента: Отправка на доработку.

Цель прецедента: Отправить карточку на доработку, на предыдущий этап.

Предусловия: Карточка которую надо согласовать просматривается.

Основной сценарий: Руководитель нажимает на кнопку «Сменить статус» и во всплывающем окне выбирает «Доработать». Руководитель также оставляет комментарий о том, что же следует доработать, после чего подтверждает свой выбор.

Постусловие: в базе данных и у всех пользователей, карточка меняет свой статус, на «доработать». В записях истории изменений и базе данных появляется запись о смене статуса карточки руководителем, а также о добавлении комментария.

2.3.4 Диаграмма прецедентов актора «администратор»

Новые прецеденты для актора «администратор» (Рисунок 13):

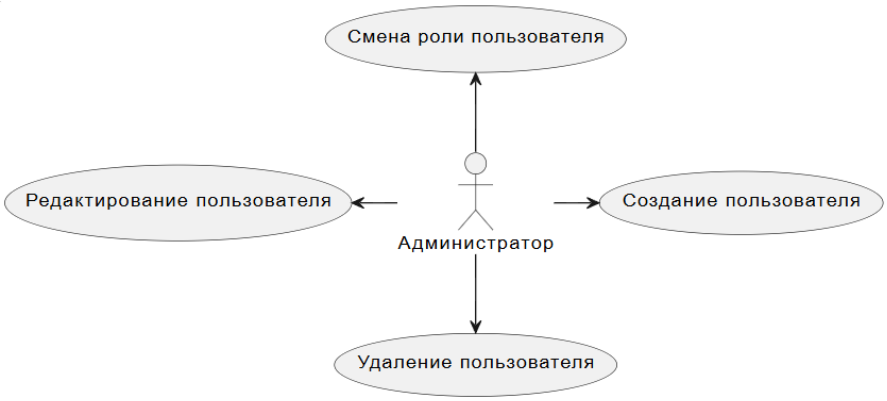


Рисунок 13 – Диаграмма прецедентов для «Администратор»

Название прецедента: Создание пользователя.

Цель прецедента: Создание учетной записи для предоставления работнику.

Предусловия: Выполнен вход в систему под администратора.

Основной сценарий: Администратор заходит на вкладку «Анализ» и нажимает на кнопку создания пользователя, после чего вводит данные пользователя в соответствующие поля. Генерирует логин и пароль пользователя и нажимает на кнопку «Создать».

Постусловия: В базе данных добавляется запись о новом пользователе, теперь ему нужно сообщить его логин и пароль, чтобы он мог войти в систему.

Название прецедента: Удаление пользователя.

Цель прецедента: Удаление пользователя, по причине ухода пользователя.

Предусловия: Открыта вкладка «Анализ».

Основной сценарий: Администратор выбирает пользователя и просматривает его страницу и нажимает на кнопку удаления пользователя. Администратор должен ввести полное имя, фамилию и отчество, чтобы была доступна кнопка удаления пользователя, после нажатия которой подтверждает свой выбор.

Постусловие: В базе данных удаляется запись с карточкой, а также все данные, связанные с этим пользователем.

Название прецедента: Редактирование пользователя.

Цель прецедента: Редактирование данных пользователя.

Предусловия: Открыта вкладка «Анализ».

Основной сценарий: Администратор выбирает пользователя и просматривает его страницу, после чего нажимает на кнопку редактирования пользователя. Администратор вбивает новые данные о пользователе и нажимает на кнопку «Сохранить» с подтверждением.

Постусловие: В базе данных редактируется запись с пользователем, а также все данные, связанные с редактированием пользователя.

Название прецедента: Смена роли пользователя.

Цель прецедента: Смена роли пользователя.

Предусловия: Открыта вкладка «Анализ».

Основной сценарий: Администратор выбирает пользователя и нажимает на кнопку «Сменить роль». Во всплывающем окне Администратор выбирает роль, после чего нажимает на кнопку «Сохранить» с подтверждением.

Постусловие: В базе данных редактируется запись о роли пользователя.

2.4 Разработка базы данных

Для реализации была разработана схема базы данных.

В таблице «User» содержится информация о пользователе, такая как: его ID, имя, отдел в котором он работает, электронная почта, пароль в зашифрованном виде, его роль.

Таблица «Task» представляет собой запись о карточке, которую может создать пользователь для заполнения резюме. В данной таблице хранится: его ID, статус его этапа, имя соискателя, его почта, номер телефона, место на которое он претендует, ссылка на источник с которого было взято резюме, наименование источника с которого было взято резюме, комментарий к карточке.

Таблица «WorkGroup» представляет собой пометку пользователей о том что они работают над данной карточкой, поэтому в нём хранится ID пользователя и ID карточки, а также здесь хранится информация о дате создания

карточки, когда она была закончена, когда она была удалена и её статус об активности.

Таблица «Document» представляет собой документы, которые могут быть прикреплены к карточке. В данной таблице хранится: его ID, идентификационный номер документа, а также название документа которое дал пользователь, ссылка на файл документа, а также ID карточки которой он принадлежит.

Таблица «History» представляет собой историю изменений карточки, которые привнёс определенный пользователь, поэтому в нём хранится ID группы которая работает над данной карточкой и ID пользователя который отредактировал карточку, в ней также хранится какое действие было сделано, старые и новые данные, а также дата изменения.

Таблица «Users Group» представляет собой таблицу «посредника» для построения отношений многие-ко-многим между таблицами «User» и «WorkGroup».

Схема базы данных представлена на рисунке 14.

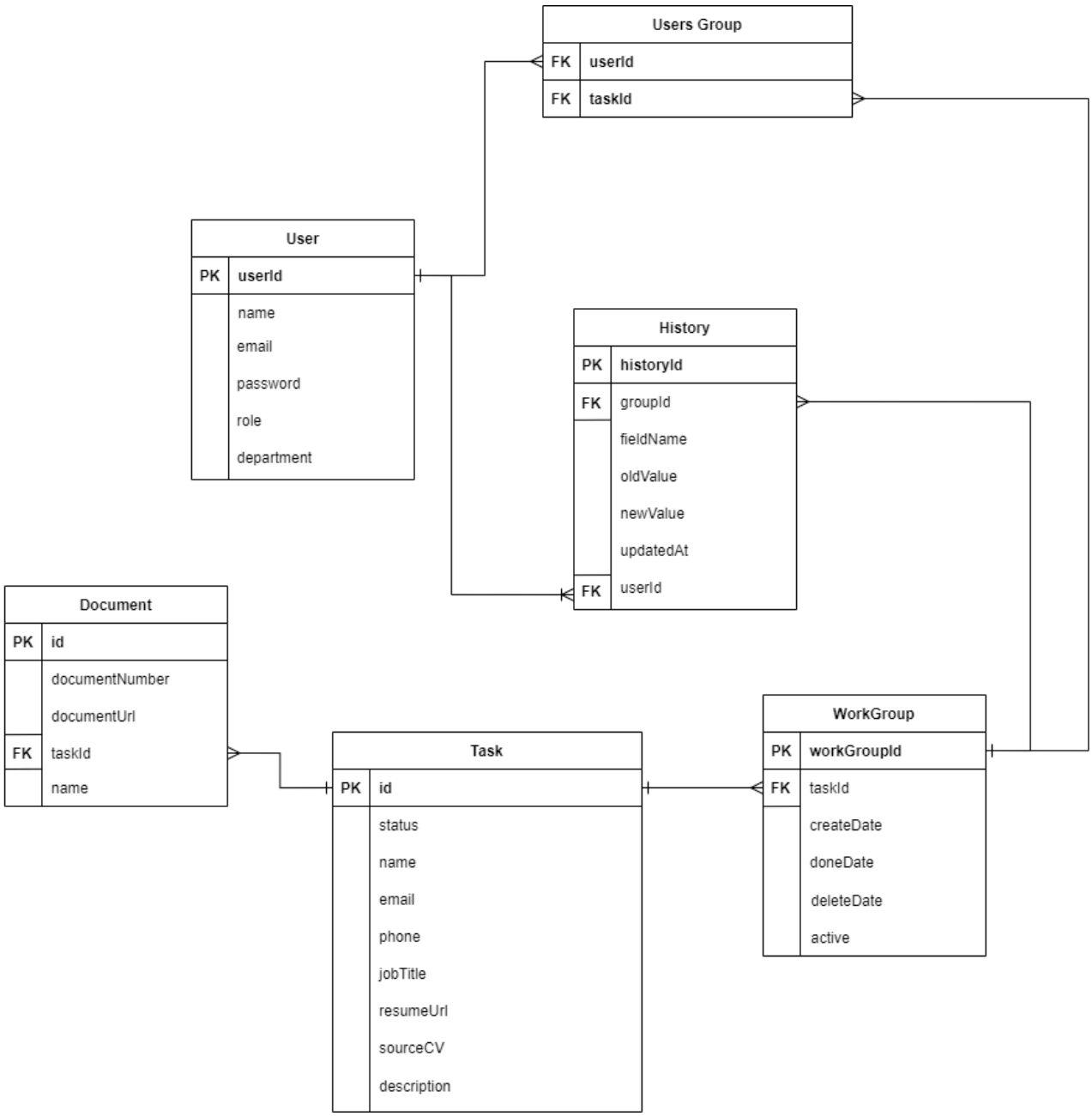


Рисунок 14 – Схема базы данных

2.5 Проектирование интерфейса

Исходя из функционала приложения можно выделить список необходимых страниц:

- страница авторизации;
- главная страница со списком всех карточек;
- страница со списком пользователей.

Страница авторизации содержит форму с логином и паролем.

Главная страница – это страница, на которую переходит после авторизации в системе. На этой странице отображаются все карточки, которые есть в системе, вверху страницы находится навигационная панель с помощью, которой можно перейти на страницу со списком всех пользователей. На этой странице создаются и редактируются все карточки с резюме. Сбоку также есть фильтрация по статусу карточки и поисковая строка, фильтрующая по любой информации, которая находится внутри карточки.

Страница со списком пользователей содержит в себе список всех пользователей, работающих в системе, перейти на эту страницу возможно при наличии роли «руководитель» или же «администратор». Внутри каждой записи о пользователе хранится информация о его эффективности работы в системе. Также здесь при наличии роли «администратор» можно создать или удалить пользователя, также можно редактировать информацию и роль в системе определенного пользователя.

2.6 Выводы ко второй главе

В результате выполнения главы была выбрана общая структура разрабатываемой системы. Был выбран список средств и технологий с помощью которых предстоит реализовать клиент, сервер и базу данных. Была разработана диаграмма прецедентов, которая показывает функциональные возможности пользователя системы. Была разработана структура базы данных, а также был выявлен список страниц, который предстоит реализовать.

3 Программная реализация

3.1 Реализация сервера

3.1.1 Авторизация

Для того чтобы работать в самой системе необходима авторизация, это нужно чтобы неавторизованный пользователь не мог подключиться к системе и вносить данные. Авторизация будет работать на JWT токенах, при отправке запроса с клиента, браузер будет отправлять его вместе с запросом и если JWT-токен верный, то выполняется действие. Регистрация пользователей будет только на стороне Администратора, который будет создавать учетные записи по просьбе и выдавать аккаунт. Также были созданы запросы для выхода из учетной записи и проверка на то, что авторизован ли пользователь с получением данных кроме пароля.

В приложении А представлен контроллер класса auth.

3.1.2 Взаимодействие с карточками

Для того чтобы работать с карточками, в которых хранится основная информация о соискателях, был создан класс tasks. В этом классе реализованы контроллер и сервис, с помощью которых создано взаимодействие с карточками. В данном контроллере представлены действия, которые может совершить пользователь, а именно: создание карточки, загрузка документа в данную карточку, смена статуса карточки и её редактирование. Каждый из этих запросов получает также данные о пользователе для последующего отслеживания действий специалистов по найму.

В приложении Б представлен контроллер tasks.

3.1.3 Отслеживание действий специалистов

Для того чтобы правильно оценивать эффективность специалистов по найму, был создан класс history. В нём будут разработаны сервисы для того, чтобы после каждого выполненного запроса, связанного с карточками, сохранять в базе данных действия пользователя. Храниться будут данные такие как: рабочая группа в которой были выполнены действия, наименование действия, старые и новые данные и пользователь, который взаимодействовал с карточкой. При редактировании полей карточки или же смене статуса в базе данных сохраняются старые данные, которые были до этого, а также новые данные, которые ввёл пользователь.

В приложении В представлен сервис для редактирования карточки пользователем, в котором сразу же создаётся таблица с данными, о его действиях.

3.1.4 Рабочая группа

Для взаимодействия с рабочей группой был создан класс workgroup, к которому пользователи могут присоединиться, чтобы отметиться о том, что они ведут работу над данной карточкой. Это нужно для того, чтобы специалист по найму мог отметить карточку чтобы воспользоваться фильтрацией и быстрее найти её из всего списка карточек. Рекомендуется чтобы специалист по найму перед тем, как вносить изменения присоединился к рабочей группе, но даже без этого можно отредактировать карточку, это действие всё равно сохранится. С помощью сервиса history, информация о присоединении к рабочей группе определенного пользователя, тоже будет храниться в базе данных.

В приложении Г представлен сервис для добавления пользователя в рабочую группу.

3.1.5 Работа с документами

Для того чтобы пользователи могли загружать файлы был создан класс documents, основной запрос отправляется с контроллера tasks, а в сервисе documents реализовано само отправление и хранение с переименованием файла, для того чтобы не было ошибок при загрузке файлов с одинаковым названием, потому как в системе будет храниться множество файлов с одинаковым названием. При загрузке документа можно сохранить имя и номер документа, но путь к нему в локальной папке сервера будет прописан уже самим сервисом. Также, при загрузке файла создаётся запись об этом с помощью сервиса history.

В приложении Д представлен сервис для отправления и хранения файлов.

3.1.6 Взаимодействие с пользователями

Чтобы руководители и администратор могли работать с пользователями, а также чтобы подключить авторизацию, был создан класс users. Реализованные контроллер и сервис users нужны для того, чтобы: редактировать данные о пользователе, удалить пользователя, получить информацию о том какие пользователи отмечены в данной рабочей группе, а также чтобы сменить роль пользователя в системе. Большинство запросов в этом контроллере недоступны пользователям с ролью Работник, так как в них либо хранится важная информация о пользователях, либо с помощью этих запросов можно непосредственно редактировать информацию о пользователях.

В приложении Е представлен контроллер users.

3.2 Реализация интерфейса

3.2.1 Авторизация

Чтобы пользователь мог войти в свою учетную запись было создано окно, на которое попадает любой, кто подключиться к системе. В данном окне

необходимо ввести свою почту и пароль, которые выдал администратор. Без авторизации работа в системе не предоставляется возможным.

На рисунке 15 представлено окно для авторизации.

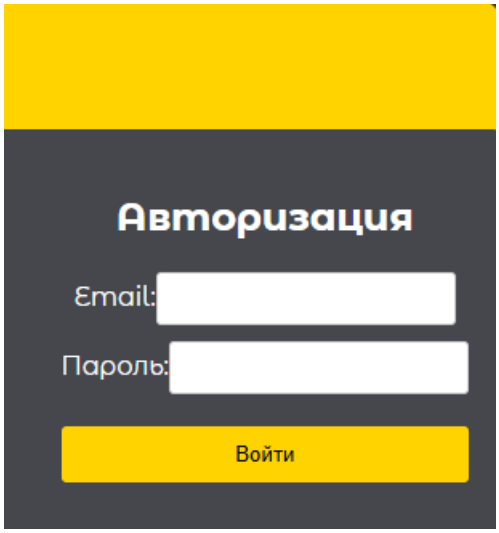


Рисунок 15 – Окно для авторизации

3.2.2 Главная страница

После авторизации пользователь попадает на главную страницу, на которой размещены все резюме созданными пользователями. На странице предусмотрена фильтрация по статусу работы карточки, а также поисковая строка которая фильтрует информацию по: имени соискателя, почте, телефону, должности на которую претендует человек, ссылке и источнику с которого было взято резюме.

На рисунке 16 представлена главная страница.

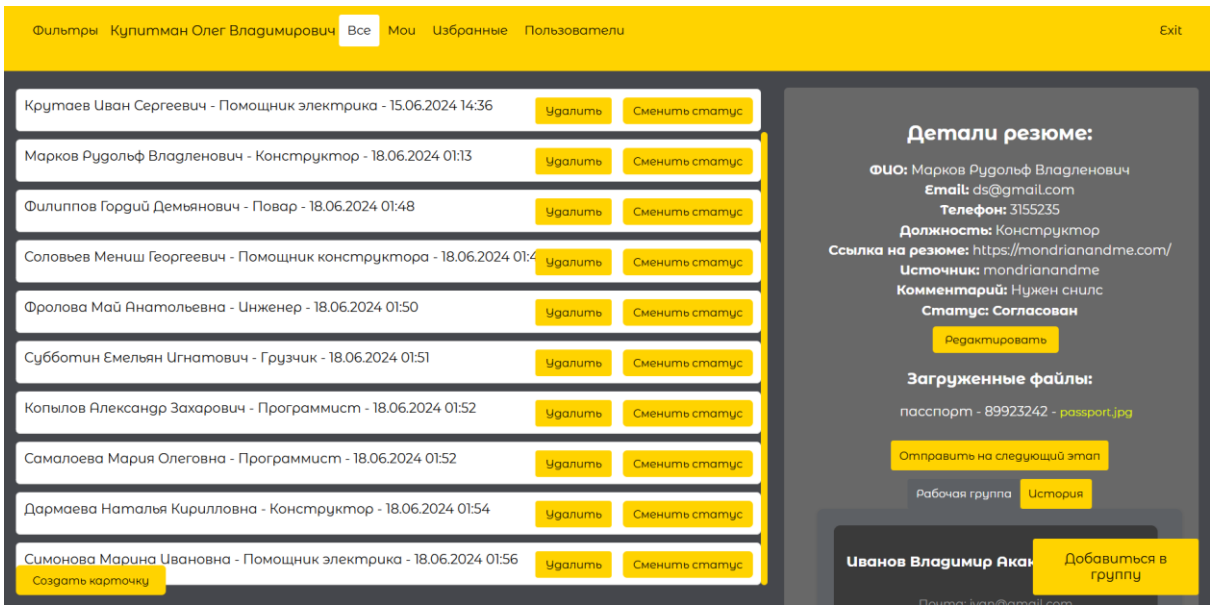


Рисунок 16 – Главная страница

3.2.3 Отображение карточки и её редактирование

Для того чтобы просмотреть информацию о уже созданной карточке, необходимо выбрать её из списка и тогда откроется окно, в котором представлена вся информация о соискателе. Загруженные ранее файлы можно при желании скачать чтобы ознакомиться с ними.

На рисунке 17 представлена окно с информацией о карточке.

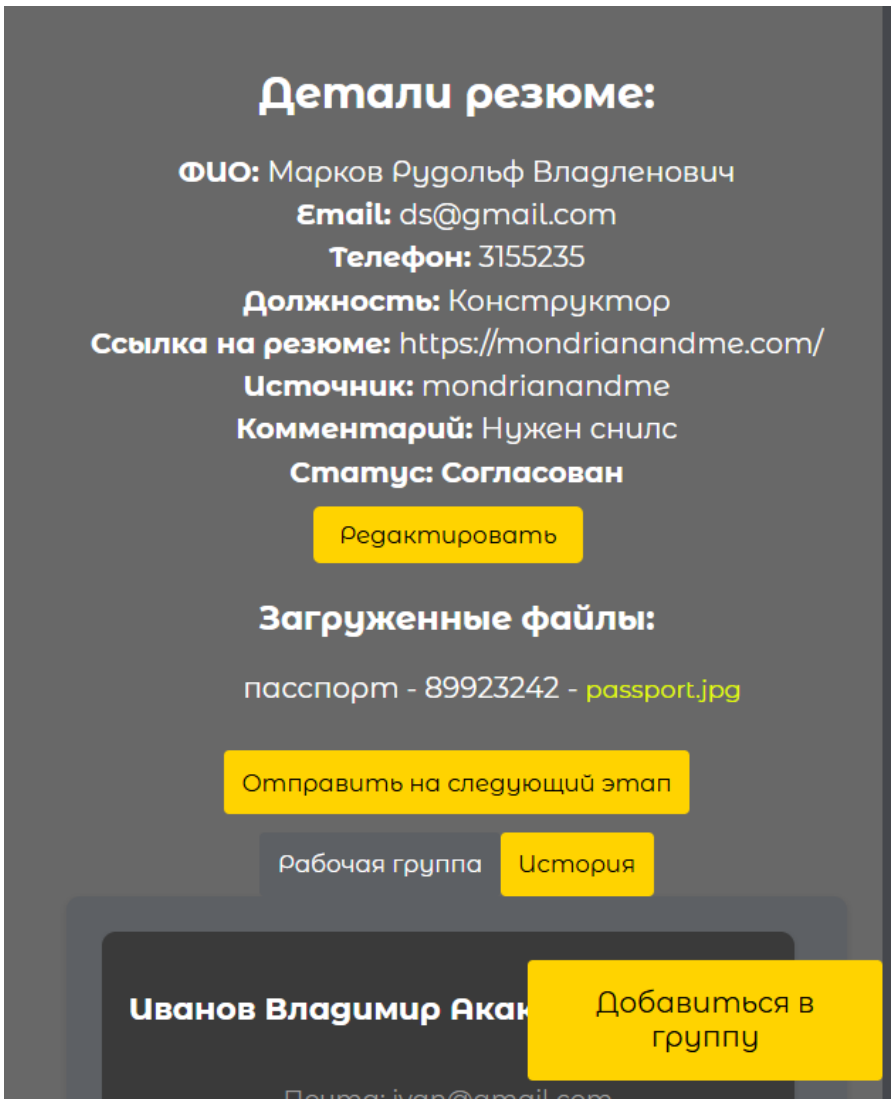


Рисунок 17 – Окно с информацией о карточке

После нажатия на кнопку «Редактировать», окно перейдет в режим редактирования, в котором можно изменить данные карточки, после чего необходимо нажать на кнопку «Сохранить», иначе вся введённая информация не будет загружена в базу данных. Любое изменение в карточке будет отображено в специальном окне «История».

На рисунке 18 представлено окно редактирования карточки.

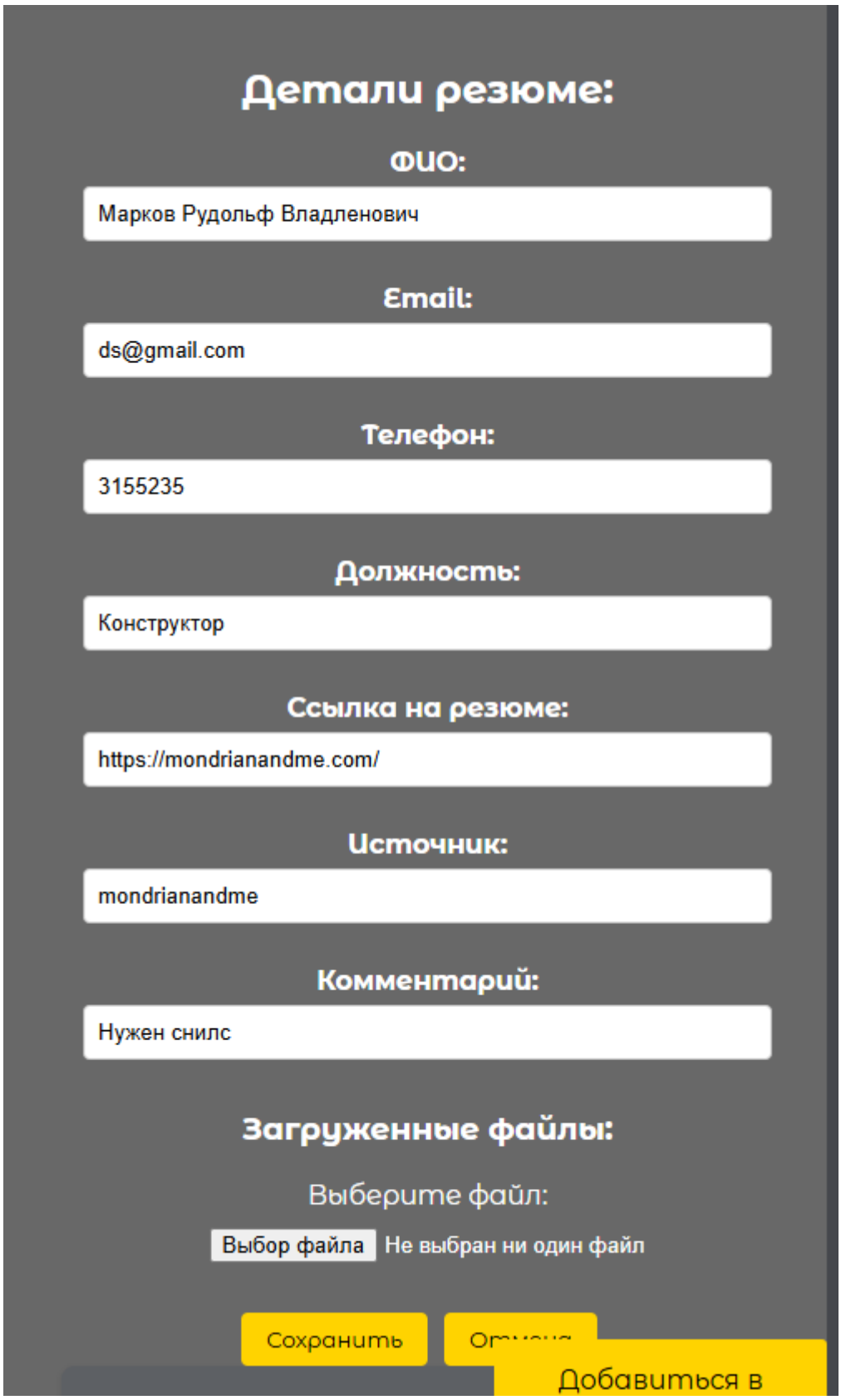


Рисунок 19 – Окно редактирования карточки

3.2.4 Смена статуса карточки

Для того чтобы сменить статус карточки для согласования, доработке или любого действия работника или же руководителя, необходимо у представленной карточки нажать кнопку «Сменить статус» и во всплывающем окне, выбрать подходящую строку и нажать «Подтвердить» иначе смена статуса не произойдет.

На рисунке 20 представлено модальное окно для смены статуса.

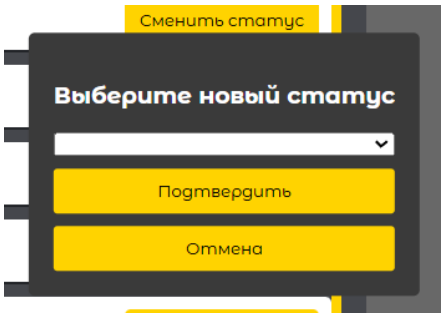


Рисунок 20 – Модальное окно для смены статуса

3.2.5 Создание карточки

Для того чтобы самолично создать карточку, необходимо нажать на кнопку, которая прикреплена к углу экрана на главной странице. Во всплывающем окне необходимо вписать хотя бы имя человека, иначе система оповестит что создание карточки невозможно, остальные данные не имеют такой строгой проверки, их можно ввести в последствии. После нажатия «Сохранить» в базе данных появится новая запись о карточке, в случае же нажатия «Отменить» система попросит подтверждения, для того чтобы не создавать карточку в базе данных.

На рисунке 21 представлено модальное окно для создания карточки.

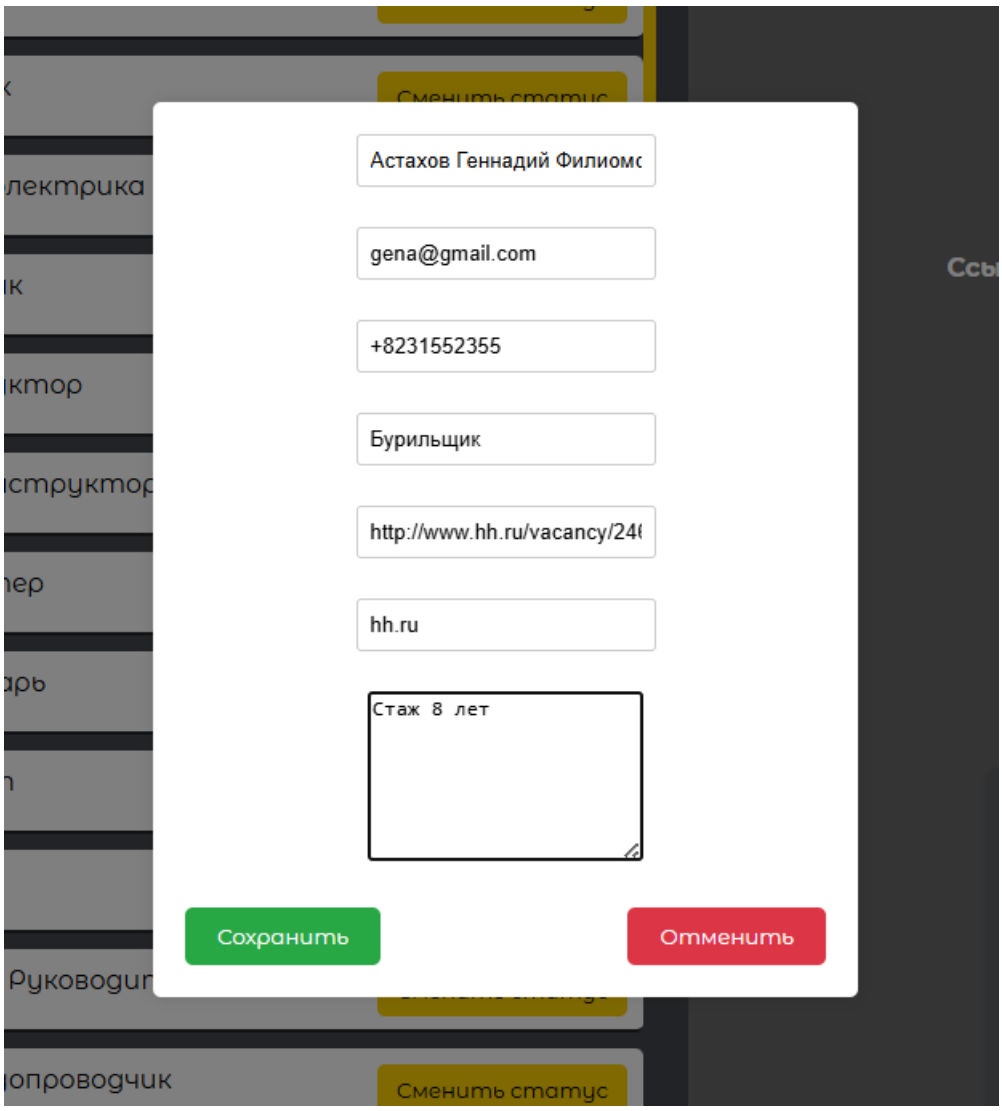


Рисунок 21 – Модальное окно для создания карточки

3.2.6 Информация об изменениях карточки

В описании карточки предложены 2 окна, а именно «История» и «Рабочая группа», в случае нажатия на вкладку истории, раскроется информация: кто привнёс изменения, время изменения, действие которые было выполнено. Если же изменялись данные такие как статус и редактирование, то будет отображено как их старое значение, так и новое. В случае загрузки документа будет показано наименование документа, а при присоединении к рабочей группе будет отображено сообщение об этом.

На рисунке 22 представлена открытая вкладка с историей изменений карточки.

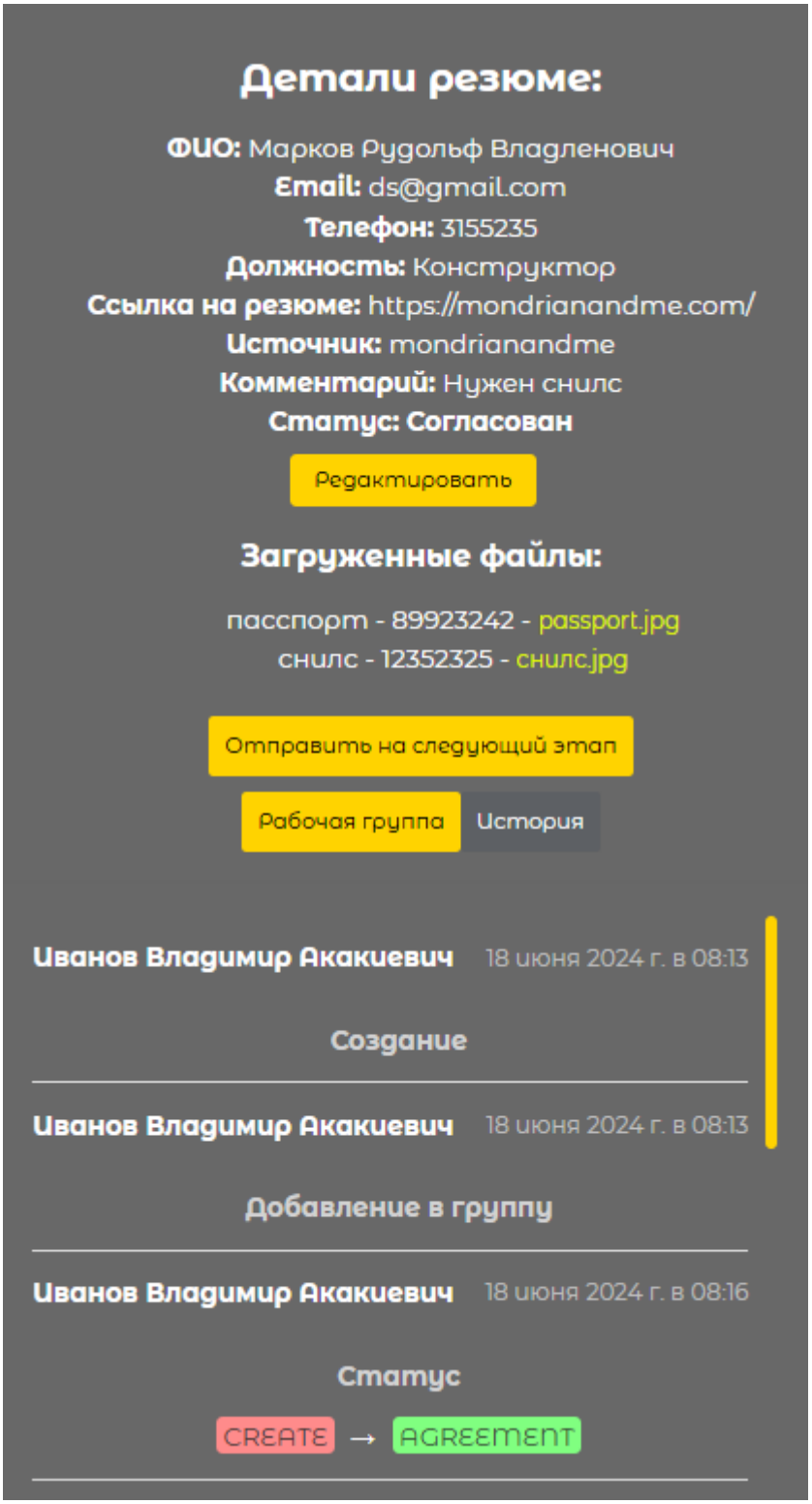


Рисунок 22 – Открытая вкладка с историей изменений карточки

3.2.7 Подключение и отображение рабочей группы

Если же из предложенных вкладок выбрать «Рабочая группа», то будет отображён список пользователей, отмеченных в данной группе, с их почтой и отделом в котором они работают. Для того чтобы подключиться к группе, необходимо нажать на соответствующую кнопку, после чего пользователь появится в списке рабочей группы. Если же пользователь самолично создавал карточку, то он автоматически будет подключен к только что созданной карточке.

На рисунке 23 представлена открытая вкладка с рабочей группой карточки.

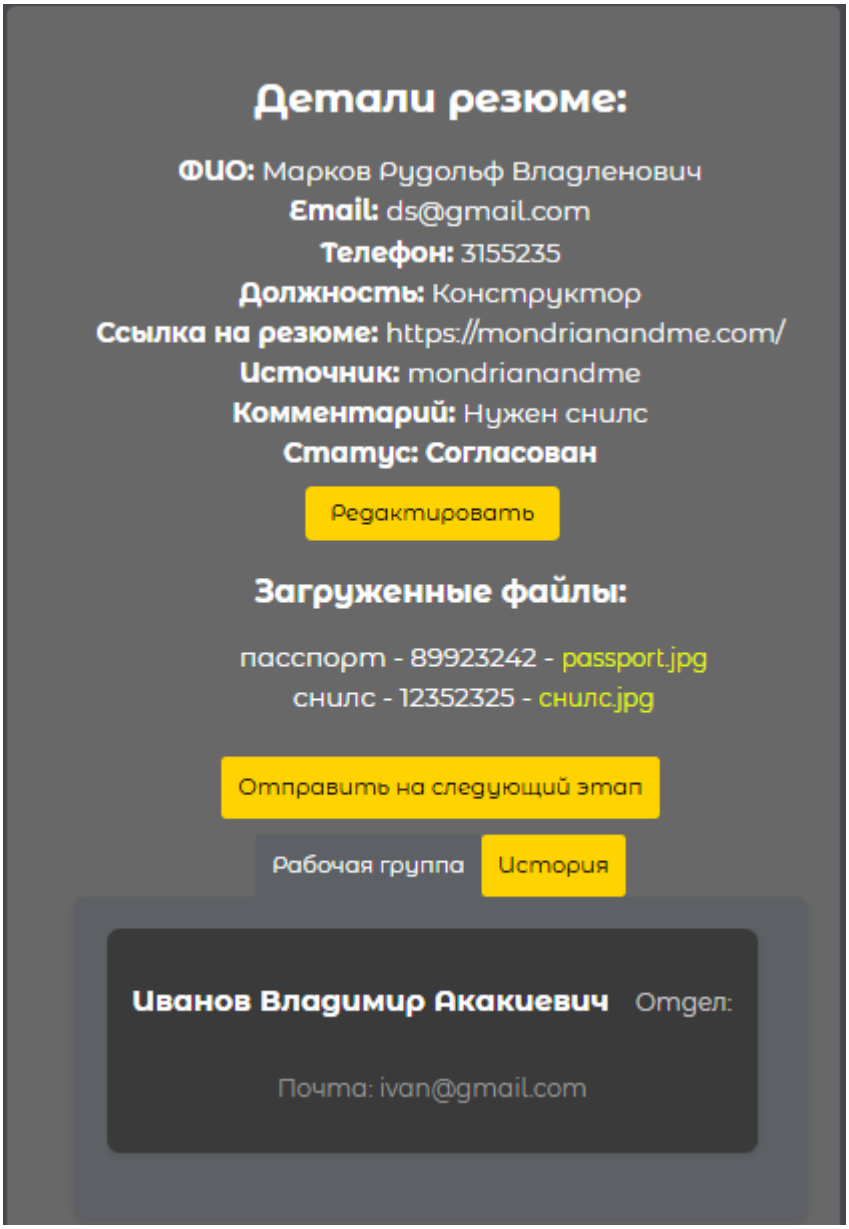


Рисунок 23 – Открытая вкладка с рабочей группой карточки

3.2.8 Страница с пользователями

Пользователь имеющий роль «Руководитель» или же «Администратор», может перейти на страницу с пользователями, в которой представлен список всех пользователей в системе. На странице предусмотрена фильтрация по роли в системе, а также поисковая строка, фильтрующая список по таким данным как: имя пользователя, почта и отдел, в котором пользователь работает.

При выборе пользователя отобразится информация о его работе в системе, но редактировать данные о пользователе, а также сменить роль в системе возможно только при наличии роли «Администратор».

На рисунке 24 представлена страница со списком пользователей в системе.

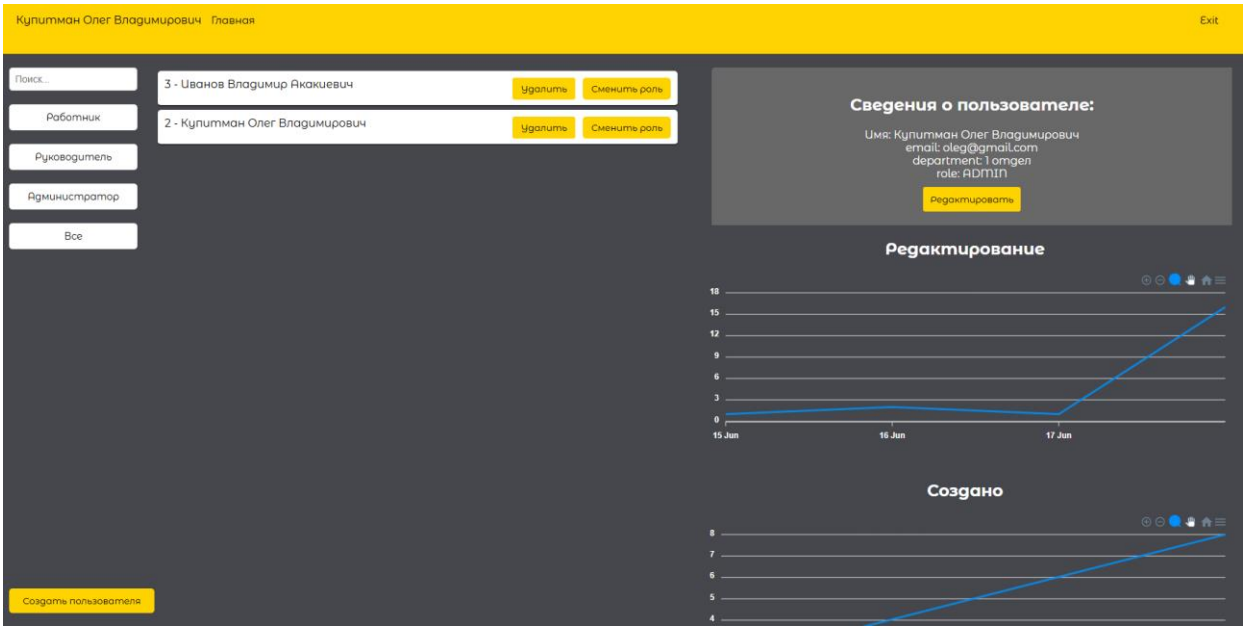
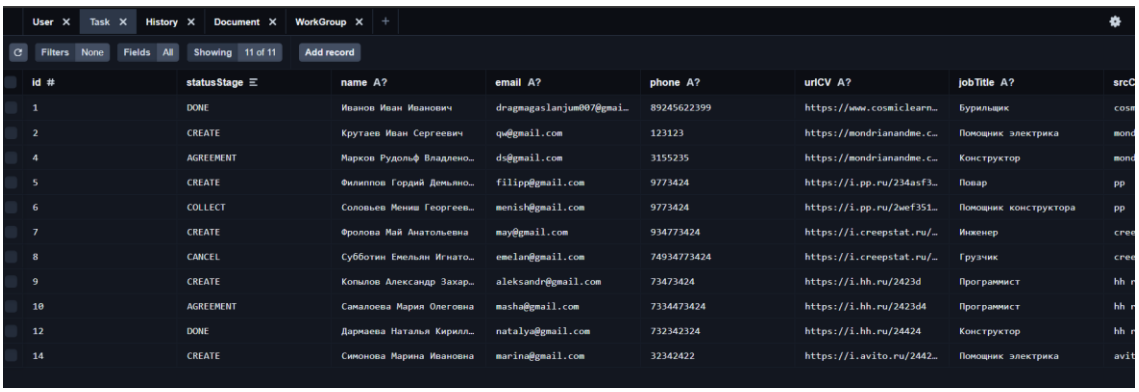


Рисунок 24 – Страница со списком пользователей в системе

3.3 Prisma Studio

Для того чтобы взаимодействовать напрямую с базой данных была подключена Prisma Studio.

На рисунке 25 продемонстрирована тестовая база данных в Prisma studio.



User | Task | History | Document | WorkGroup

Filters: None | Fields: All | Showing: 11 of 11 | Add record

id #	statusStage	name A?	email A?	phone A?	urlCV A?	jobTitle A?	srcC
1	DONE	Иванов Иван Иванович	dragmagaslanjum007@gmai…	89245622399	https://www.cosmiclearn…	Бурильщик	cosm
2	CREATE	Крутаев Иван Сергеевич	qw@gmail.com	123123	https://mondrianandme.c…	Помощник электрика	mond
4	AGREEMENT	Марков Рудольф Владлено…	ds@gmail.com	3155235	https://mondrianandme.c…	Конструктор	mond
5	CREATE	Филиппов Гордий Демьяно…	filipp@gmail.com	9773424	https://i.pp.ru/234asf3…	Повар	pp
6	COLLECT	Соловьев Мениш Георгеев…	menish@gmail.com	9773424	https://i.pp.ru/2wef351…	Помощник конструктора	pp
7	CREATE	Фролова Май Анатольевна	may@gmail.com	934773424	https://i.creepstat.ru/…	Инженер	cree
8	CANCEL	Субботин Емельян Игнато…	emelan@gmail.com	74934773424	https://i.creepstat.ru/…	Грузчик	cree
9	CREATE	Копылов Александр Захар…	aleksandr@gmail.com	73473424	https://i.hh.ru/2423d	Программист	hh r
10	AGREEMENT	Самалоева Мария Олеговна	masha@gmail.com	7334473424	https://i.hh.ru/2423d4	Программист	hh r
12	DONE	Дармаева Наталья Кирилл…	natalya@gmail.com	732342324	https://i.hh.ru/24424	Конструктор	hh r
14	CREATE	Симонова Марина Ивановна	marina@gmail.com	32342422	https://i.avito.ru/2442…	Помощник электрика	avit

Рисунок 25 – Заполненная база данных

3.4 Настройка Docker Compose

Запуск нескольких контейнеров стоит делать с помощью инструмента Docker Compose, он позволяет расположить несколько изолированных сред на одном хосте и управлять ими.

Для этого был составлен docker-compose.yml файл, который и будет запускать все эти контейнеры, а также созданы Dockerfile на сервере и клиенте, чтобы упаковать код.

На рисунке 26 представлена частичная конфигурация docker-compose.yml

```yaml
backend:
  build:
    context: ./cv_board
    dockerfile: Dockerfile
  container_name: nestjs_backend
  env_file:
    - .env
  environment:
    DATABASE_URL: "postgresql://admin:admin@postgres:5432/nestjs-postgres?schema=public"
    JWT_SECRET: "cat"
    JWT_EXPIRATION_TIME: "168000000"
    POSTGRES_HOST_AUTH_METHOD: md5
  depends_on:
    - postgres
  ports:
    - "3000:3000"
  networks:
    nestjs-network:
      ipv4_address: 172.18.0.3
  command: >
    sh -c "npx prisma generate &&
           npx prisma migrate deploy &&
           npm run start:prod"

frontend:
  build:
    context: ./cv_board_client
    dockerfile: Dockerfile
  container_name: vuejs_frontend
  ports:
    - "8080:80"
  networks:
```

Рисунок 26 – Частичная конфигурация docker-compose.yml

В конфигурацию были также включены контейнеры базы данных PostgreSQL и Prisma Studio, чтобы управлять и взаимодействовать с базой данных. Для каждого контейнера, были подключены порты, сделано это для того чтобы обращаться к этим сервисам по этим же портам.

После создания всех файлов и настройки конфигурации, можно запустить все контейнеры одной командой.

На рисунке 27 представлены включенные контейнеры в Docker Desktop.

Рисунок 27 – Включенные контейнеры

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы было проведено анализ предметной области, поставлены требования при выполнении задачи, определены цели создания системы, проанализированы существующие решения.

Далее было осуществлено проектирование общей структуры разрабатываемой системы, были выбраны средства и технологии разработки для клиента, сервера и базы данных, также были составлены диаграммы прецедентов для всех акторов, проектирование интерфейса программы и базы данных.

В результате разработано веб-приложение для автоматизации процесса массового набора персонала, которое помогает специалистам быстрее нанимать людей на работу, а также предоставляет руководству инструменты для анализа эффективности работы специалистов по найму.

На текущий момент реализован минимальный функционал по ТЗ заказчика, впереди предстоит тестирование, по итогам которых будут вноситься изменения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 7.32-2001. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления // Консорциум кодекс: электронный фонд правовых и нормативно-технических документов : официальный сайт. – URL: https://docs.cntd.ru/document/1200026224 (дата обращения: 19.10.2023).

2. ГОСТ 7.9-95 (ИСО 214-76). Система стандартов по информации, библиотечному и издательскому делу. Реферат и аннотация. Общие требования // Научная периодика: проблемы и решения. – URL: https://nppir.ru/wp-content/uploads/22-gost-7.9-95.pdf (дата обращения: 19.10.2023).

3. ГОСТ 7.1-2003. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления // Российская государственная библиотека: официальный сайт. – URL: https://diss.rsl.ru/datadocs/doc_291wu.pdf (дата обращения: 19.10.2023).

4. СТУ 7.5-07-2021. Стандарт университета. Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности // Сибирский федеральный университет : официальный сайт. – URL: https://about.sfu-kras.ru/node/8127 (дата обращения19.10.2023).

5. Главный экран E-staff // E-staff – Программа для HR и кадровых агенств, автоматизация подбора персонала: сайт. – URL: https://e-staff.ru/estaff_home (дата обращения: 19.10.2023).

6. Главный экран Skillaz // HR-платформа для автоматизации управления персоналом: сайт. – URL: https://skillaz.ru/ (дата обращения: 19.10.2023).

7. Документация NestJS // NestJS – A progressive Node.js framework: сайт. – URL: https://docs.nestjs.com/ (дата обращения: 19.10.2023).

8. Современный учебник Typescript // Typescript – JavaScript With Syntax For Types: сайт. – URL: https://www.typescriptlang.org/ (дата обращения: 19.10.2023).

9. Документация PostgreSQL // PostgreSQL – The world’s most advanced open source database: сайт. – URL: https://www.postgresql.org/docss/ (дата обращения: 19.10.2023)

10. Руководство для Prisma ORM // Simlify working and interacting with database: сайт. – URL: https://www.prisma.io/ (дата обращения: 19.10.2023).

11. Документация Vue.js // The Progressive JavaScript Framework: сайт. – URL: https://vuejs.org/ (дата обращения: 19.10.2023).

12. Документация Docker // Docker docs: сайт. – URL: https://docs.docker.com (дата обращения: 19.10.2023).

ПРИЛОЖЕНИЕ А

Листинг кода auth.controller

```
@HttpCode(200)
@UseGuards(LocalAuthenticationGuard)
@Post('log-in') // Запрос для аутентификации
async logIn(@Req() request: RequestWithUser, @Res() response: Response) {
  const { user } = request
  const cookie = this.authenticationService.getCookieWithJwtToken(user.id)
  request.res.set('Set-Cookie', cookie)
  user.password = undefined
  return request.res.send(user)
}
```

```
@UseGuards(JwtAuthenticationGuard)
@Post('log-out') // Запрос для выхода из учетной записи
async logOut    (@Req()request:RequestWithUser,@Res() response: Response) {
  response.setHeader('Set-Cookie', this.authenticationService.getCookieForLog-
Out());
  return response.sendStatus(200);
}
```

```
@UseGuards(JwtAuthenticationGuard)
@Get('check') // Запрос предназначенный для проверки авторизации
authenticate(@Req() request: RequestWithUser) {
  const user = request.user;
  user.password = undefined;
  return user;
}
```

Полный код программы будет предоставлен на CD-диске.

ПРИЛОЖЕНИЕ Б

Листинг кода tasks.controller

```
  @Post('/documents') // Запрос предназначенный для загрузки документа на сервер
  @UseInterceptors(FileInterceptor('file'))
  @UseGuards(JwtAuthenticationGuard)
  async attachDocumentToTask(
    @Body() requestBody: { taskId: string, number: string, fileName: string, name:
string },
    @UploadedFile() file: Express.Multer.File,
    @Req() request: RequestWithUser
  ): Promise<void> {
    const { taskId, number, fileName, name } = requestBody;
    const document = await this.documentsService.uploadDocument(taskId, number,
fileName, file, request.user.id, name);
    return document;
  }


  @Patch('/status') // Запрос на смену статуса карточки
  @UseGuards(JwtAuthenticationGuard)
  async statusChange(
    @Body('id') taskid: number,
    @Body('statusStage') newStatus: Status,
    @Req() request: RequestWithUser
  ) {
    return this.tasksService.statuschange(taskid, newStatus, request.user.id);
  }

  @Patch('/update/:id') // Запрос для редактирования карточки
  @UseGuards(JwtAuthenticationGuard)
  async update(@Body('id') id: number, @Body() updateTaskDto: UpdateTaskDto, @Req()
request: RequestWithUser) {
    return this.tasksService.updateAndHistory(id, updateTaskDto, request.user.id);
  }
```

Полный код программы будет предоставлен на CD-диске.

ПРИЛОЖЕНИЕ В
Листинг кода tasks.service

```
// Сервис являющийся примером сохранения в истории изменений карточки
async updateAndHistory(id: number, updateTaskDto: UpdateTaskDto, userId: number) {
    const taskBeforeUpdate = await this.prismaService.task.findUnique({
      where: {
        id,
      },
    });
    const updatedTask = await this.prismaService.task.update({
      where: { id },
      data: updateTaskDto
    });

    const differences = this.findDifferences(taskBeforeUpdate, updatedTask)

    let oldValue = '';
    let newValue = '';

    if ((await differences).length === 0) {
    } else {
      (await differences).forEach((field) => {
        oldValue = field + ' ' + taskBeforeUpdate[field] + ' ' + oldValue;
        newValue = field + ' ' + updatedTask[field] + ' ' + newValue;
      });
    }

    const groupId = await this.workGroupService.findGroupId(id);

    if(oldValue !== newValue)
    {
    await this.historyService.create({
      userId: userId,
      groupId: groupId,
      fieldName: "редактирование",
      oldValue: oldValue,
      newValue: newValue,
    });
  }

    return updatedTask;
  }
```

Полный код программы будет предоставлен на CD-диске.

ПРИЛОЖЕНИЕ Г

Листинг кода workgroup.service

```
// Сервис для добавления в рабочую группу, с последующей записью в истории измене-
ний
async addUserToGroup(taskId: number, userId: number) {
    try {
      const group = await this.prismaService.workGroup.findUnique({
        where: { taskId: taskId },
        include: { users: true },
      });
      if (!group) {
        throw new Error(`Work group with ID ${taskId} not found`);
      }
      const userExistsInGroup = group.users.some((user) => user.id === userId);
      if (userExistsInGroup) {
        throw new Error(`User with ID ${userId} is already in the group`);
      }

      const groupId = await this.prismaService.workGroup.findUnique({
        where: { taskId: taskId, },
      });
      await this.prismaService.workGroup.update({
        where: { id: groupId.id },
        data: {
          users: {
            connect: [{ id: userId }],
          },
        },
      });
    await this.prismaService.history.create({
      data: {
        userId: userId,
        groupId: groupId.id,
        fieldName: "Добавление в группу",
        oldValue: "",
        newValue: "",
      }
    })
    } catch (error) {
      throw new Error(`Failed to add user to group: ${error.message}`);
    }
  }
```

Полный код программы будет предоставлен на CD-диске.

ПРИЛОЖЕНИЕ Д

Листинг кода documents.service

```
// Сервис предназначенный для загрузки файла на сервер, с последующей записью в ис-
тории изменений
async uploadDocument(taskId: string, number: string, fileName: string, file: Ex-
press.Multer.File, userId: number, name: string): Promise<void> {
    try {
      const uniqueFileName = new Date().getTime() + '_' + fileName;
      const uploadDir = 'upload';
      const filePath = path.join(uploadDir, uniqueFileName);
      if (!fs.existsSync(uploadDir)) {
        fs.mkdirSync(uploadDir, { recursive: true });
      }
      const fileStream = fs.createWriteStream(filePath);
      fileStream.on('error', (err) => {
        throw new Error(`Failed to write file: ${err.message}`);
      });
      fileStream.on('finish', async () => {
        const document = await this.prisma.document.create({
          data: {
            number,
            name,
            url: `${uniqueFileName}`,
            task: { connect: { id: parseInt(taskId) } }
          }
        });
        return document;
      });
      fileStream.write(file.buffer);
      fileStream.end();
      const workGroupId = await this.workGroupService.findGroupId(Number(taskId));
      await this.historyService.create({
        userId: userId,
        groupId: workGroupId,
        fieldName: "Загрузка",
        oldValue: "",
        newValue: file.originalname,
      });
    } catch (error) {
      throw new Error(`Failed to upload document: ${error.message}`);
    }
  }
```

Полный код программы будет предоставлен на CD-диске.

ПРИЛОЖЕНИЕ Е

Листинг кода users.controller

```
@UseGuards(UsersGuards)
  @Get('/all')// Запрос для получения полного списка всех пользователей в системе
  findAll() {
    return this.usersService.findAll();
  }

  @Get(':id')// Запрос для получения информации конкретного пользователя
  async findOne(@Param('id') id: string) {
    return await this.usersService.getById(Number(id));
  }

  @UseGuards(AdminGuards)
  @UseGuards(JwtAuthenticationGuard)
  @Patch('/update/:id')// Запрос для редактирования информации о пользователе
  async update(@Param('id') id: string, @Body() updateUserDto: UpdateUserDto) {
    const idInt = parseInt(id)
    return this.usersService.update(idInt, updateUserDto);
  }

  @UseGuards(AdminGuards)
  @Delete(':id')// Запрос для удаления пользователя из системы
  remove(@Param('id') id: string) {
    return this.usersService.remove(Number(id));
  }

  @Get('/find/group/:taskId')// Запрос для получения списка пользователей в рабочей
группе
  async findGroup(@Param('taskId') taskId: string){
    return this.usersService.getUsersInWorkGroup(Number(taskId));
  }

  @UseGuards(AdminGuards)
  @UseGuards(JwtAuthenticationGuard)
  @Patch('/role')
  async statusChange(
    @Body('id') userId: number,
    @Body('role') newRole: Role,
  ) {
    return this.usersService.roleChange(userId, newRole);
  }
```

Полный код программы будет предоставлен на CD-диске.

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
«17» 06 2024 г.

БАКАЛАВРСКАЯ РАБОТА

090301 Информатика и вычислительная техника

Автоматизация процесса массового набора персонала

Руководитель	_____ подпись	17.06.24 дата	старший преподаватель должность, ученая степень	А.Г. Хантимиров
Выпускник	_____ подпись	17.06.24 дата		А.М. Джумаев
Нормоконтролёр	_____ подпись	17.06.24 дата	старший преподаватель должность, ученая степень	А.Г. Хантимиров

Красноярск 2024