МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Ижевский государственный технический университет

имени М. Т. Калашникова»

Институт непрерывного профессионального образования

Работа защищена с оценкой

«\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись \_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем»

МДК 01.05 «Веб-программирование»

на тему «система управления рабочими процессами сотрудников»

Выполнил:

Студент гр. Д22-191-2

А.А.Самохвалов Руководитель:

Е.А.Зылева

Ижевск 2024

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc202440222)

[Цели работы: 3](#_Toc202440223)

[Задачи работы: 4](#_Toc202440224)

[1 АНАЛИТИЧЕСКАЯ ЧАСТЬ 4](#_Toc202440225)

[1.1 Анализ предметной области 4](#_Toc202440226)

[1.2 Анализ систем с аналогичными функциями 8](#_Toc202440227)

[1.3 Выбор средств разработки 12](#_Toc202440228)

[1.4 Требования к системе 15](#_Toc202440229)

[1.5 Технические решения проекта программного продукта 18](#_Toc202440230)

[2 РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ 20](#_Toc202440231)

[2.1 Разработка интерфейса 20](#_Toc202440232)

[2.2 Разработка Базы Данных 26](#_Toc202440233)

[2.3 Описание разработанного функционала 36](#_Toc202440234)

[ЗАКЛЮЧЕНИЕ 50](#_Toc202440235)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 52](#_Toc202440236)

# ВВЕДЕНИЕ

Информационная система управления рабочими процессами сотрудников, или система управления задачами, играет ключевую роль в организации эффективной работы команды. Она позволяет оптимизировать рабочие процессы за счет автоматизации задач, что снижает нагрузку на менеджеров и повышает скорость выполнения работ. Кроме того, такие системы упрощают коммуникацию, обеспечивая централизованное место для обсуждений и минимизируя количество электронных писем.  
 Прозрачность и контроль также являются важными аспектами. Руководители могут легко отслеживать статус выполнения задач, что позволяет своевременно выявлять проблемы и принимать меры. Системы генерируют отчеты о выполнении задач, что помогает в оценке производительности как отдельных сотрудников, так и всей команды.  
 Гибкость и адаптивность систем управления задачами способствуют установлению приоритетов и сроков выполнения работ, что помогает командам адаптироваться к изменяющимся условиям и требованиям бизнеса. В случае возникновения непредвиденных обстоятельств задачи можно быстро перераспределить между сотрудниками.  
 Кроме того, такие системы повышают мотивацию работников. Ясно обозначенные задачи и цели помогают сотрудникам понимать, что от них ожидается, и повышают их вовлеченность. Возможность давать и получать обратную связь по выполнению задач способствует профессиональному росту сотрудников.  
 Таким образом, создание и внедрение информационной системы управления рабочими процессами не только оптимизирует внутренние процессы компании, но и способствует созданию более продуктивной и мотивированной команды. В условиях динамично меняющегося рынка такие системы становятся необходимым инструментом для достижения успеха.

Цели работы:

Цель информационной системы управления рабочими процессами сотрудников заключается в оптимизации процессов, что позволяет упростить и автоматизировать выполнение задач, повышая общую эффективность работы команды. Она также способствует улучшению коммуникации, создавая централизованное пространство для обсуждения задач и обмена информацией, что снижает количество недоразумений и увеличивает скорость принятия решений.  
 Кроме того, такая система повышает прозрачность, позволяя руководителям отслеживать статус выполнения задач и производительность сотрудников в реальном времени. Гибкость и адаптивность системы позволяют командам быстро реагировать на изменения в приоритетах и перераспределять задачи в зависимости от текущих потребностей бизнеса.  
 Система также стимулирует мотивацию сотрудников, создавая четкие цели и задачи, что способствует повышению их вовлеченности и удовлетворенности работой. Интеграция с другими системами, такими как CRM и ERP, позволяет создать комплексный подход к управлению бизнес-процессами. Наконец, поддержка удаленной работы делает взаимодействие команд, работающих в разных местах, более эффективным, что особенно актуально в современных хусловиях. Таким образом, основная цель системы — создать эффективную и продуктивную рабочую среду, способствующую достижению бизнес-целей организации.

Задачи работы:

1. Список сотрудников (должности, роли)
2. Список задач (работы)
3. Распределение задач по сотрудникам
4. Авторизация
5. Добавление новых сотрудников (доступна только администраторам)
6. Права доступа пользователей к каким-либо ресурсам (например обычный сотрудник не может изменять списки работы или распределения задач)
7. Учёт выполненной работы (вознаграждение сотрудника за перевыполненный план)

1 АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1 Анализ предметной области

Основной задачей системы является повышение общей производительности компании через оптимизацию управления сотрудниками и задачами. Это включает:

1. **Эффективное распределение ресурсов**: Правильное назначение задач с учетом компетенций сотрудников, их загруженности и времени.
2. **Мониторинг выполнения задач**: Возможность в реальном времени отслеживать прогресс выполнения задач, корректировать планы и определять проблемные области.
3. **Поддержка коммуникаций и сотрудничества**: Упрощение взаимодействия между сотрудниками и руководителями, улучшение процессов обмена информацией и обсуждения задач.
4. **Оценка результатов работы**: Объективная оценка вклада сотрудников в деятельность компании через систему KPI, показатели производительности и выполнение планов.
5. **Автоматизация кадровых процессов**: Управление не только задачами, но и данными о сотрудниках, графиками работы, временем отпусков и больничных.

Таким образом, основная цель системы заключается в автоматизации и улучшении всех аспектов управления персоналом, что способствует повышению производительности, снижению временных и человеческих затрат на административные задачи, а также формированию корпоративной культуры, ориентированной на результат.

### Ключевые функции продукта

**Планирование задач и ресурсов**:

* 1. Система должна предоставлять инструменты для планирования рабочих задач, проектов и мероприятий. Это может включать автоматическое назначение задач в зависимости от компетенций сотрудников, приоритетности проектов и их текущей загруженности.
  2. Важной функцией является возможность постановки задач с четкими сроками и этапами выполнения.

**Мониторинг выполнения и отчетность**:

* 1. Возможность отслеживания прогресса по каждой задаче и проекту в реальном времени. Руководитель может видеть, на каком этапе находится задача, кто ее выполняет, и сколько времени требуется для завершения.
  2. Автоматическая генерация отчетов на основе выполненных задач и временных затрат сотрудников для анализа производительности.

**Коммуникации и координация работы**:

* 1. Встроенные инструменты для эффективного общения сотрудников между собой и с руководством, такие как корпоративные чаты, видеоконференции, инструменты для совместной работы над документами и проектами.
  2. Важно, чтобы коммуникация по задачам не терялась и вся история обсуждений хранилась в системе.

### Категории пользователей системы

В системе управления работой сотрудников задействованы различные категории пользователей, каждая из которых имеет свои цели и задачи:

**Руководители**:

* 1. Их основная роль — это планирование задач, контроль за выполнением работ и оценка эффективности сотрудников. Руководители должны иметь доступ к аналитике и отчетам, чтобы на основе этих данных принимать управленческие решения.
  2. Система должна давать возможность гибкого управления ресурсами, регулировать нагрузку сотрудников, изменять приоритеты задач.

**Сотрудники**:

* 1. Основные пользователи системы, которые получают и выполняют задачи. Они должны иметь удобные инструменты для доступа к своим задачам, общения с коллегами, предоставления отчетов и получения обратной связи.
  2. Важно, чтобы система поддерживала мотивационные механизмы, такие как уведомления о выполнении задач, премирование за достижения, прозрачность карьерного роста.

**Администраторы системы**:

* 1. Технические специалисты, отвечающие за обеспечение бесперебойной работы системы, управление правами доступа и настройку интеграций с другими корпоративными системами.
  2. Они обеспечивают техническую поддержку и решение проблем, связанных с безопасностью данных и обновлениями системы.

### Бизнес-процессы, поддерживаемые системой

Система управления работой сотрудников поддерживает следующие основные бизнес-процессы:

**Планирование и постановка задач**:

Руководители могут планировать и назначать задачи сотрудникам на основе их квалификаций и приоритетов компании. Система должна учитывать текущую загруженность сотрудников и автоматически предлагать оптимальные варианты распределения задач.

**Мониторинг выполнения**:

В реальном времени отслеживается выполнение задач, а также соблюдение сроков. Сотрудники могут отмечать этапы выполнения работы, а руководители — контролировать прогресс.

**Оценка эффективности сотрудников**:

Через систему KPI, отчетов и аналитики оцениваются результаты работы каждого сотрудника. Это помогает принимать решения о награждениях, обучении или корректировке обязанностей.

**Автоматизация кадровых операций**:

Все аспекты кадрового учета, от управления графиками работы до учета отпусков и больничных, должны быть автоматизированы. Это снижает нагрузку на HR-отдел и минимизирует ошибки в учете данных.

### Технические требования

**Безопасность данных**:

Поскольку в системе обрабатываются персональные данные сотрудников, защита информации является критически важной. Это включает шифрование данных, многофакторную аутентификацию и управление правами доступа.

**Масштабируемость и производительность**:

Система должна быть готова к увеличению числа пользователей и объема данных, сохраняя высокую производительность. Это особенно важно для крупных организаций с большим количеством сотрудников.

**Доступность и мобильность:**

В современных условиях система должна поддерживать работу с мобильных устройств, предоставляя доступ к задачам и информации в любое время и в любом месте.

### Заключение

Анализ предметной области системы управления работой сотрудников демонстрирует важность комплексного подхода к разработке и внедрению подобных решений. Внедрение СУРС помогает не только автоматизировать процесс управления сотрудниками, но и значительно повысить их эффективность, улучшить взаимодействие внутри команды и усилить контроль за выполнением задач.

1.2 Анализ систем с аналогичными функциями

В любой сфере, особенно в бизнесе, нужно учитывать конкурентов и их аналоги. После некоторого времени, проведенными в поисках информации о конкурентах, я выделил для себя сайты, представленные ниже:

AmoCRM

AmoCRM (Рисунок 1 - Главный экран AmoCRM) — система управления взаимоотношениями с клиентами, позволяющая автоматизировать продажи и организовать работу отдела маркетинга внутри небольшого предприятия. Внедрение AmoCRM позволяет оптимизировать внутренние процессы компании, и улучшить качество работы менеджеров по продажам и руководителя отдела благодаря встроенному инструментарию.

Плюсы:

* Доступная цена — стоимость программы, в зависимости от функциональности тарифа, составляет от 599 до 1599 рублей в месяц. Для ознакомления с продуктом также действует бесплатный период.
* Удобный интерфейс — интуитивно понятное меню и настройка прав доступа упрощает работу в CRM и сокращает время, необходимое для обучения персонала.
* Интеграция с телефонией, почтой и т. д. — Amo CRM предусматривает возможность быстрой интеграции с сервисами email и sms-рассылки, IP-телефонии благодаря готовым модулям во внутреннем магазине.
* Регистрация всех клиентских данных — сервис ведет полный учет контактных данных и истории взаимодействия клиента с компанией: от записи разговоров с менеджерами до истории покупок на сайте.

Минусы:

* В первую очередь, это слабый функционал вне продаж. AmoCRM рассчитана сугубо на работу с клиентом и продажами, поэтому другие бизнес-процессы в ней развиты слабо. Эту проблему можно решить добавлением виджетов и интеграцией со сторонними сервисами. В стандартной системе иные функции не предусмотрены.
* Второй недостаток – аналитика. В самой системе есть 6 элементов подсчета: анализ продаж, сводный отчет, отчет по сотрудникам, список событий в CRM, звонки и цели. Данные пункты достаточно хорошо настроены и показывают достоверную информацию, но недостаток в том, что добавить свои элементы или редактировать имеющиеся нельзя.

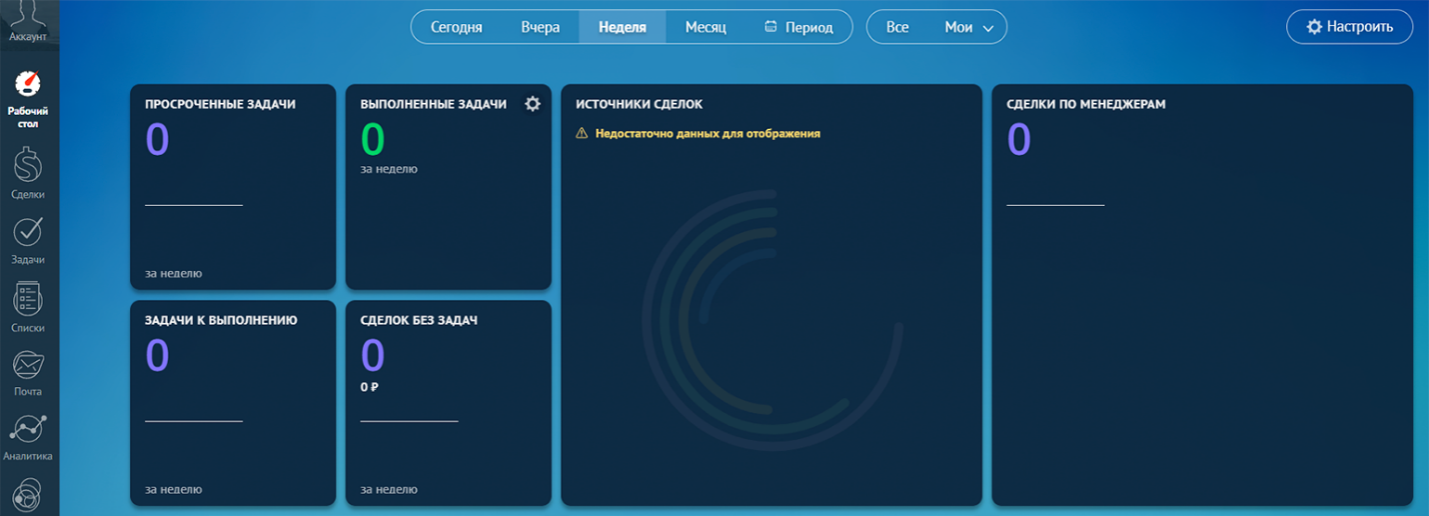


Рисунок 1 - Главный экран AmoCRM

Битрикс24

Битрикс24 (Рисунок 2 - Главный экран БитриксCRM) — один из продуктов компании «1С», лидера российского рынка разработки программного обеспечения для бизнеса. Система работает по принципу Saas: клиенты приобретают ПО, которое обслуживается на стороне провайдера. Понять, что это — CRM Битрикс, можно абсолютно бесплатно: зарегистрироваться в сервисе может каждый желающий. Правда, набор функций будет очень сильно ограничен: чтобы расширить его, нужно выбрать один из платных тарифов или «коробочную» версию.

Изначально Bitrix24 был частью CMS «1С-Битрикс: Управление сайтом». В наследство системе досталось много опций по взаимодействию с сайтом компании и даже полноценный конструктор веб-ресурсов. С 2012 года CRM Битрикс 24 — это самостоятельный продукт.

В 2019 году исследование Института проблем предпринимательства и J’son & Partners Consulting показало, что около 44 % предпринимателей России используют CRM.Битрикс24. По версии международного журнала PC Magazine, система входит в десятку лучших CRM мира.

Разобраться, что это — СРМ Битрикс, будет несложно: система интуитивно понятна и предлагает множество подсказок внутри личного кабинета. На помощь крупному бизнесу, который хочет выстроить собственную экосистему, придут специалисты по настройке и оптимизации CRM — например, программисты компании «Авиант».

Плюсы:

* возможность без проблем сменить подрядчика, благодаря большой партнерской сети сертифицированных битрикс-разработчиков
* собственная техническая поддержка сайта от 1С-Битрикс, которая реагирует на заявки в течение дня
* отсутствие скрытых платежей

Минусы:

* Лицензия стоит дорого
* Bitrix-разработчики стоят дорого
* Bitrix часто глючит
* Bitrix сильно тормозит и не выдерживает нагрузки

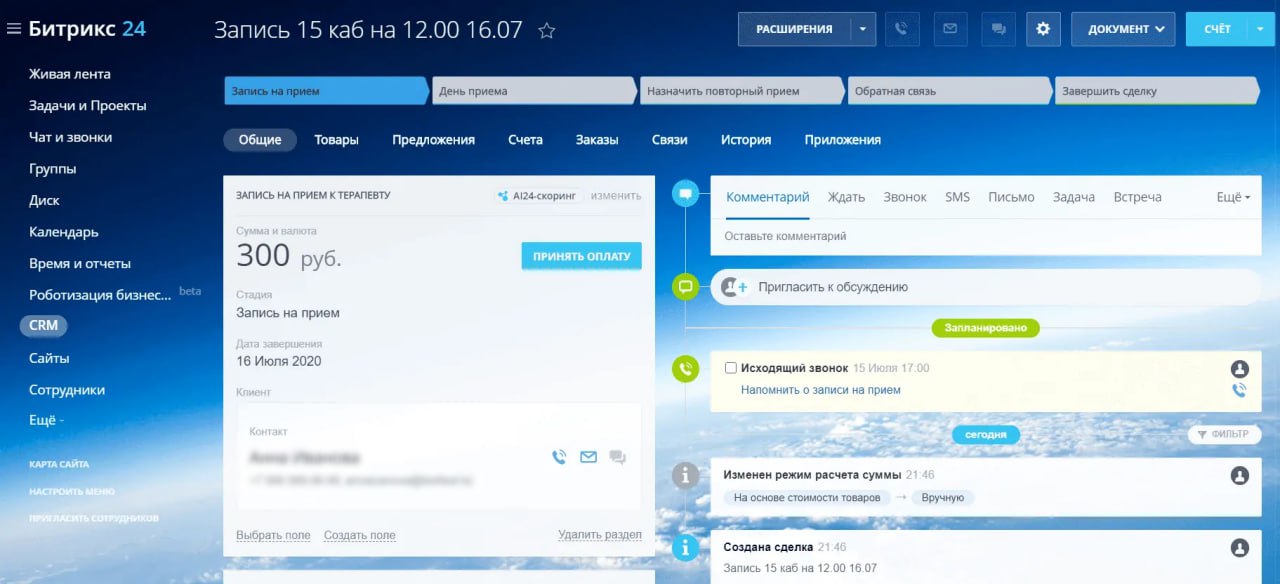


Рисунок 2 - Главный экран БитриксCRM

RetailCRM

RetailCRM (Рисунок 3 - Главный экран RetailCRM) — это решение для eCommerce и ритейла, которое помогает управлять заказами, клиентами и всеми коммуникациями в едином окне.

Система автоматизирует и упростит процессы так, что ваши менеджеры смогут обрабатывать больше заказов с лучшей эффективностью. Позволит за 30 минут запустить встроенную Программу лояльности и заняться CRM-маркетингом, чтобы удерживать и возвращать покупателей. Службы доставки, телефония, платёжные системы, мессенджеры, маркетплейсы и прочее — для интеграции со сторонними сервисами есть больше 100 готовых модулей. 14 500 магазинов по всему миру уже работают с RetailCRM и больше не сталкиваются с этими трудностями. Через платформу проходит более 25 млн заказов в год. С её помощью человеческий фактор сводится к нулю, риск потери клиентов уменьшается, а лояльность увеличивается.

Плюсы:

* Возможности для работы
* Цена
* Интеграторы

Минусы:

* Не самый удобный и понятный сайт
* Сложная настройка
* Дорогая цена

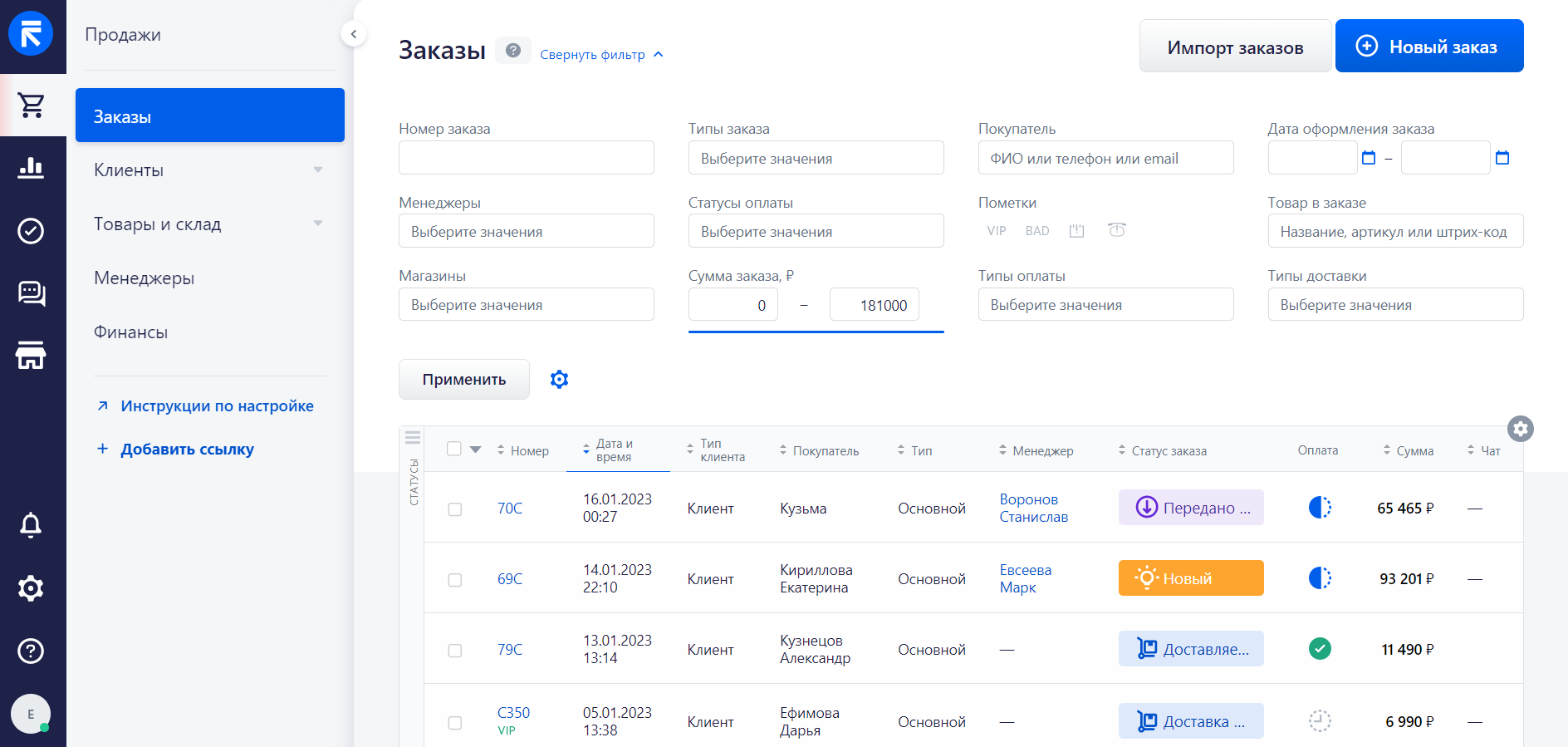


Рисунок 3 - Главный экран RetailCRM

1.3 Выбор средств разработки

**Spring Framework (Java)** – для серверной части:

**Почему Spring?**

1. **Модульность и гибкость:** Spring предоставляет широкий спектр модулей, таких как Spring Boot, Spring Data, Spring Security и Spring MVC, которые позволяют гибко настроить серверную часть системы.
2. **Высокая производительность:** Spring Boot, как часть экосистемы, упрощает разработку высокопроизводительных REST API и других веб-сервисов.
3. **Поддержка масштабируемости:** Фреймворк легко адаптируется под системы с высокой нагрузкой благодаря встроенной поддержке многопоточности и асинхронного программирования.
4. **Широкие возможности интеграции:** Spring поддерживает интеграцию с различными базами данных, такими как PostgreSQL, MySQL, а также с внешними сервисами через REST, SOAP и Kafka.
5. **Сообщество и документация:** Spring – один из самых популярных фреймворков для разработки серверных приложений, с активным сообществом и обширной документацией, что упрощает решение возникающих проблем.

**React (JavaScript)** – для клиентской части:

**Почему React?**

1. **Одностраничные приложения (SPA):** React идеально подходит для разработки современных интерфейсов с динамическими элементами, обеспечивая плавное и быстрое взаимодействие пользователя с системой.
2. **Компонентная архитектура:** React предлагает удобный подход к разработке интерфейсов через переиспользуемые компоненты, что ускоряет процесс разработки и упрощает поддержку кода.
3. **Высокая производительность:** Использование виртуального DOM делает интерфейс быстродействующим даже при большом объеме данных.
4. **Мобильная адаптивность:** React позволяет создавать адаптивные интерфейсы для работы на разных устройствах, что важно для обеспечения доступности системы.

#### Аргументы за выбор сайта как платформы

* **Доступность:** Веб-приложение доступно через браузер без необходимости установки дополнительных приложений, что удобно для сотрудников и администрации.
* **Мобильность:** Современные веб-технологии (например, адаптивный дизайн) обеспечивают удобную работу на настольных компьютерах, планшетах и смартфонах.
* **Обновление:** Обновление сайта происходит централизованно, без необходимости установки новых версий клиентского ПО пользователями.
* **Масштабируемость:** Веб-сайты легко адаптируются под изменяющиеся нагрузки, а выбранный стек React + Spring обеспечивает возможность горизонтального масштабирования.

#### Структура системы

**Клиентская часть (Frontend):**

* + React используется для создания пользовательского интерфейса, с акцентом на интуитивность и удобство.
  + Взаимодействие с серверной частью осуществляется через REST API.
  + Для стилизации интерфейса используется библиотека TailwindCSS.

**Серверная часть (Backend):**

* + Spring Boot выступает в качестве основного инструмента для реализации бизнес-логики.
  + Поддерживаются CRUD-операции для работы с задачами, сотрудниками, проектами и другими данными системы.
  + Реализуется REST API для обмена данными между клиентской и серверной частями.
  + Используется Spring Security для реализации механизмов аутентификации и авторизации.

**База данных:**

* + Для хранения данных будет использоваться база данных PostgreSQL, которая обеспечивает надежность и масштабируемость.
  + Для упрощения работы с базой данных применяется Spring Data JPA.

#### Преимущества выбранного подхода

**Быстрая разработка:**

* + Использование Spring Boot упрощает настройку серверной части и ускоряет запуск системы.
  + React позволяет быстро создавать прототипы пользовательского интерфейса.

**Гибкость и модульность:**

* + Взаимодействие через REST API обеспечивает четкое разделение клиентской и серверной частей, что упрощает их обновление и развитие.
  + Модульная структура Spring и React позволяет легко добавлять новые функции и модули.

**Поддержка современных стандартов:**

* + Используемые технологии соответствуют актуальным стандартам в разработке веб-приложений, обеспечивая долгосрочную поддержку и совместимость.

**Масштабируемость и производительность:**

* + Технологии React и Spring поддерживают горизонтальное и вертикальное масштабирование, что позволяет адаптировать систему под рост числа пользователей.

Выбор Spring и React для разработки системы управления рабочими процессами обоснован их высокой производительностью, гибкостью и активной поддержкой сообществ. Этот стек позволяет создать масштабируемое, надежное и современное веб-приложение, которое будет соответствовать потребностям бизнеса и обеспечивать удобство для пользователей.

1.4 Требования к системе

Система управления работой сотрудников должна обеспечивать выполнение нескольких ключевых функций, которые позволяют автоматизировать и упростить многие процессы, а также соответствовать современным требованиям к пользовательскому интерфейсу и техническим характеристикам.

#### Ключевые функциональные требования

**Мониторинг и отчетность**:

* 1. Система должна предоставлять руководителям возможность отслеживать прогресс выполнения задач в реальном времени. Это позволяет оперативно выявлять задержки, анализировать производительность сотрудников и принимать обоснованные решения.
  2. Необходимо обеспечить генерацию отчетов по выполнению задач, динамике проектов и эффективности команд.

**Улучшение планирования**:

* 1. Система должна предоставлять инструменты для долгосрочного и краткосрочного планирования задач, что позволяет видеть общую картину загрузки сотрудников.
  2. Предусмотрено автоматическое распределение задач на основе данных о загруженности и компетенциях команды.

**Повышение удовлетворенности сотрудников**:

* 1. Четкое распределение задач, доступность информации о целях и сроках, а также прозрачность процесса выполнения задач снижают уровень стресса сотрудников и способствуют их вовлеченности.
  2. Возможность сотрудникам самостоятельно отслеживать свои достижения и прогресс.

#### Обучение и развитие персонала

**Поддержка профессионального развития**:

* 1. Модуль для управления обучением должен предоставлять функционал для организации тренингов, семинаров, курсов, а также для отслеживания их прохождения сотрудниками.
  2. Должны поддерживаться индивидуальные планы обучения и карьерного роста для каждого сотрудника, что способствует их мотивации и удержанию в компании.

**Оценка навыков и знаний**:

* 1. Реализована возможность оценки полученных навыков через тестирования, оценки руководителей или других сотрудников.
  2. Интеграция системы с внешними платформами обучения для расширения образовательных возможностей.

#### Оценка эффективности и контроль результатов

* Оценка работы сотрудников осуществляется через систему ключевых показателей эффективности (KPI). Это позволяет руководителям получать объективные данные о результатах работы каждого сотрудника.
* Система должна предоставлять возможность сравнения результатов сотрудников, что помогает выявлять сильные стороны и зоны для улучшения.

**Технические требования**

**Кроссбраузерная совместимость**:

* + Сайт должен корректно работать на всех популярных браузерах, включая Google Chrome, Mozilla Firefox, Microsoft Edge, Safari и другие.
  + Для обеспечения совместимости используются стандарты HTML5, CSS3 и ES6+.

**Мобильная адаптивность**:

* + Система должна быть оптимизирована для работы на мобильных устройствах.
  + Реализуется адаптивная верстка, позволяющая корректно отображать интерфейс на экранах разных размеров (настольные компьютеры, планшеты, смартфоны).
  + Особое внимание уделяется удобству использования на устройствах с сенсорным управлением, включая крупные кнопки и адаптированное меню.

**Высокая производительность**:

* + Система должна быстро загружаться и реагировать на действия пользователей. Для этого используются методы оптимизации, такие как кэширование, минимизация ресурсов и асинхронная загрузка данных.

Таким образом, СУРП должна не только соответствовать высоким функциональным требованиям, но и предоставлять современный, удобный и надежный интерфейс, доступный с любого устройства.

## 1.5 Технические решения проекта программного продукта

На Use Case диаграмме (Рисунок 4) представлены три исполнителя: Работник, администратор и Менеджер. Администратор добавляет и назначает должности сотрудникам компании, менджер следит за успеваемостью работников, а работники выполняют задачи, которые создал администратор или менджер

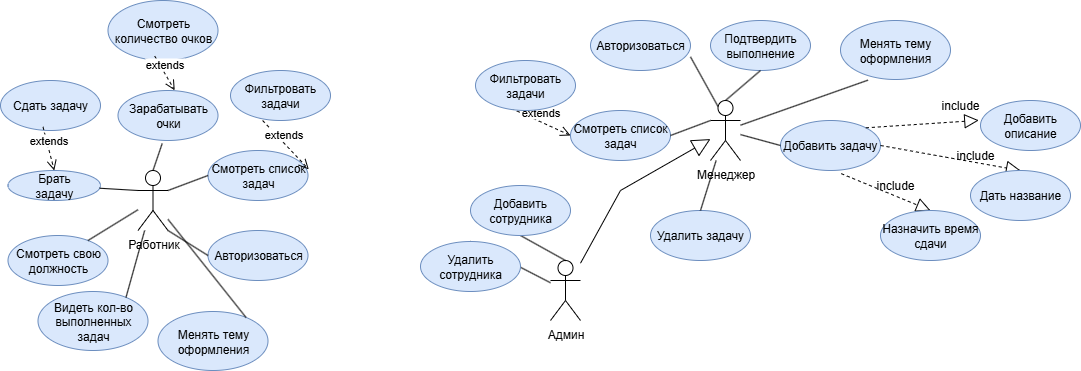


Рисунок 4 - UML-диаграмма

ER-диаграмма для описанного веб-приложения представлена на рисунке 19 (Рисунок 5).

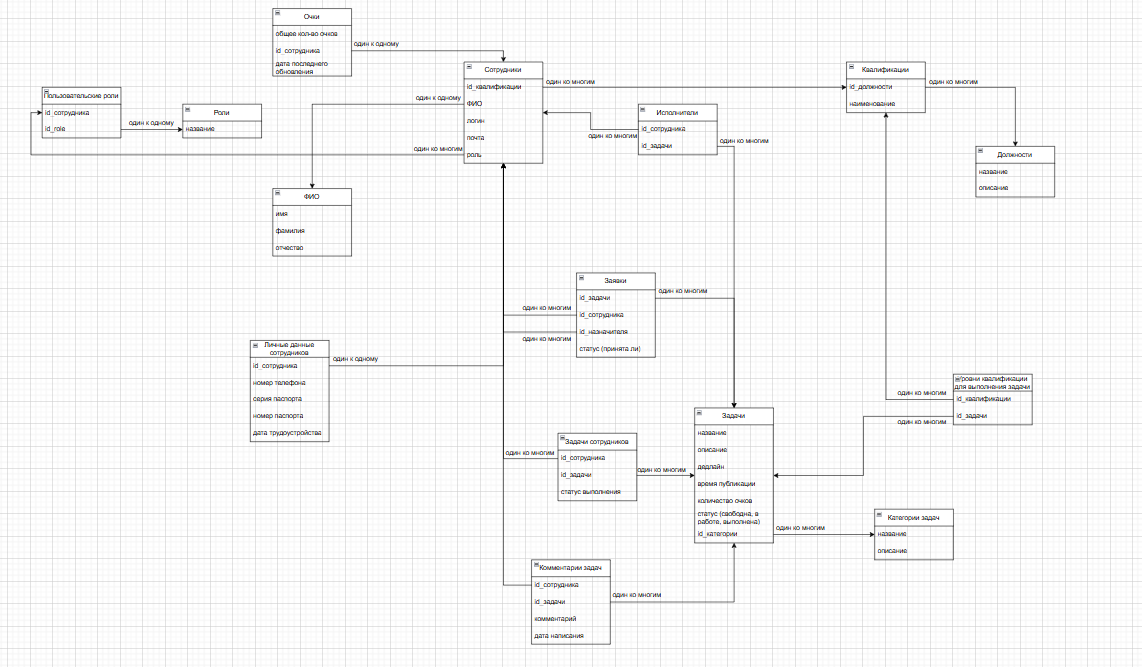
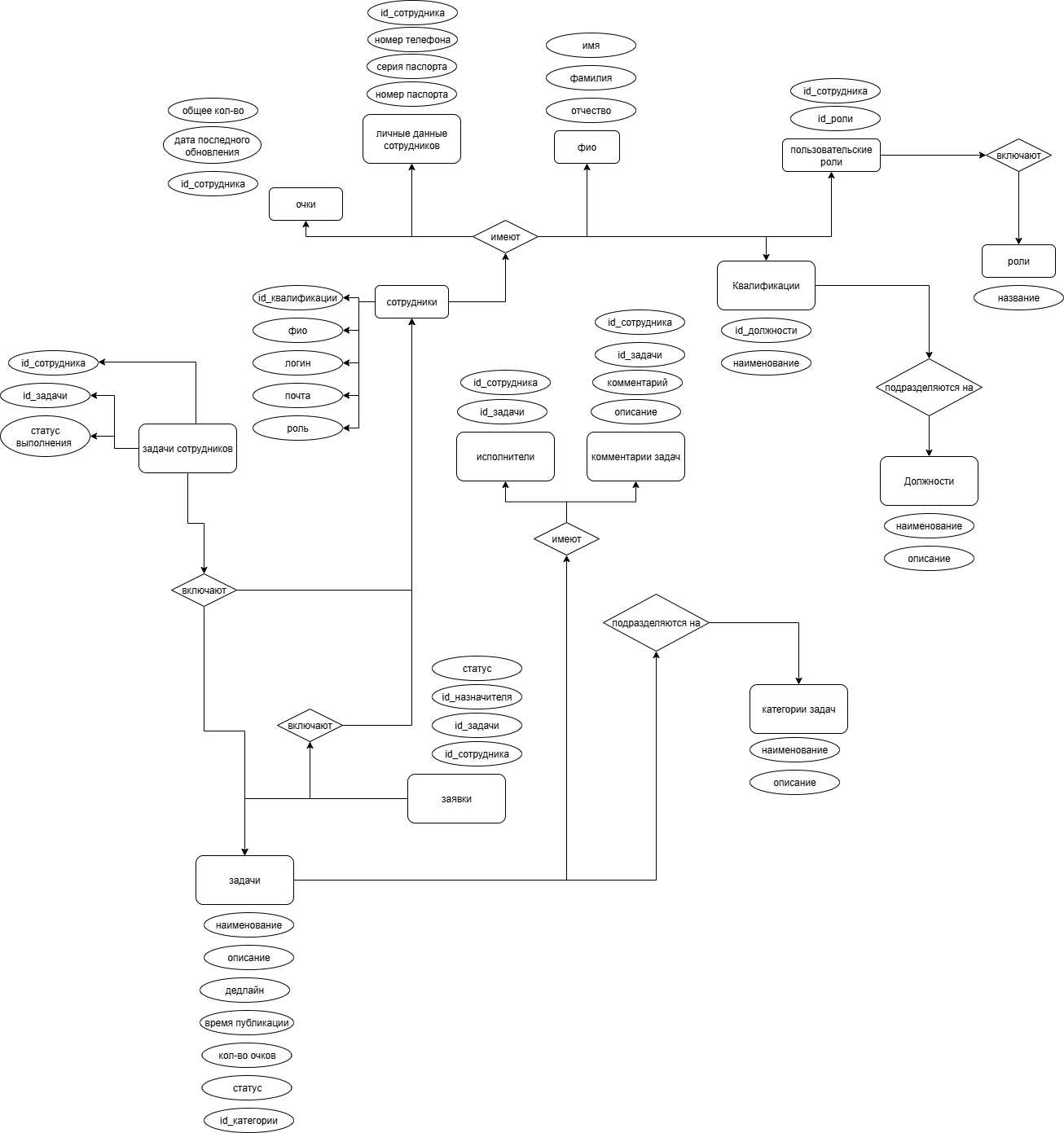


Рисунок 5 – ER-диаграмма

В данной диаграмме присутствуют следующие сущности: user\_roles(пользовательские роли), roles(роли), points(очки), workers(сотрудники), FIO(ФИО), workers\_data(личные данные сотрудников), workers\_tasks(задачи сотрудников), tasks\_description(комментарии к задачам), tasks(задачи), tasks\_categories(категории задач), qualification\_level(уровни квалификации для выполнения задач), qualification(квалификация), tikets(заявки), positions(должности), performers(исполнители).

Концептуальная модель базы данных (Концептуальная модель 1) представляет взаимосвязи между таблицами, а также поля таблиц. Например, квалификации содержат поля id\_должности и наименование и подразделяются на Должности.



Концептуальная модель 1

2 РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ

2.1 Разработка интерфейса

При переходе на страницу сайта пользователь попадает на окно авторизации для дальнейшей работы. Так как это внутренне-корпоративное приложение, создать нового пользователя может только администратор, назначенный компанией, что исключает возможность различных атак на сайт.

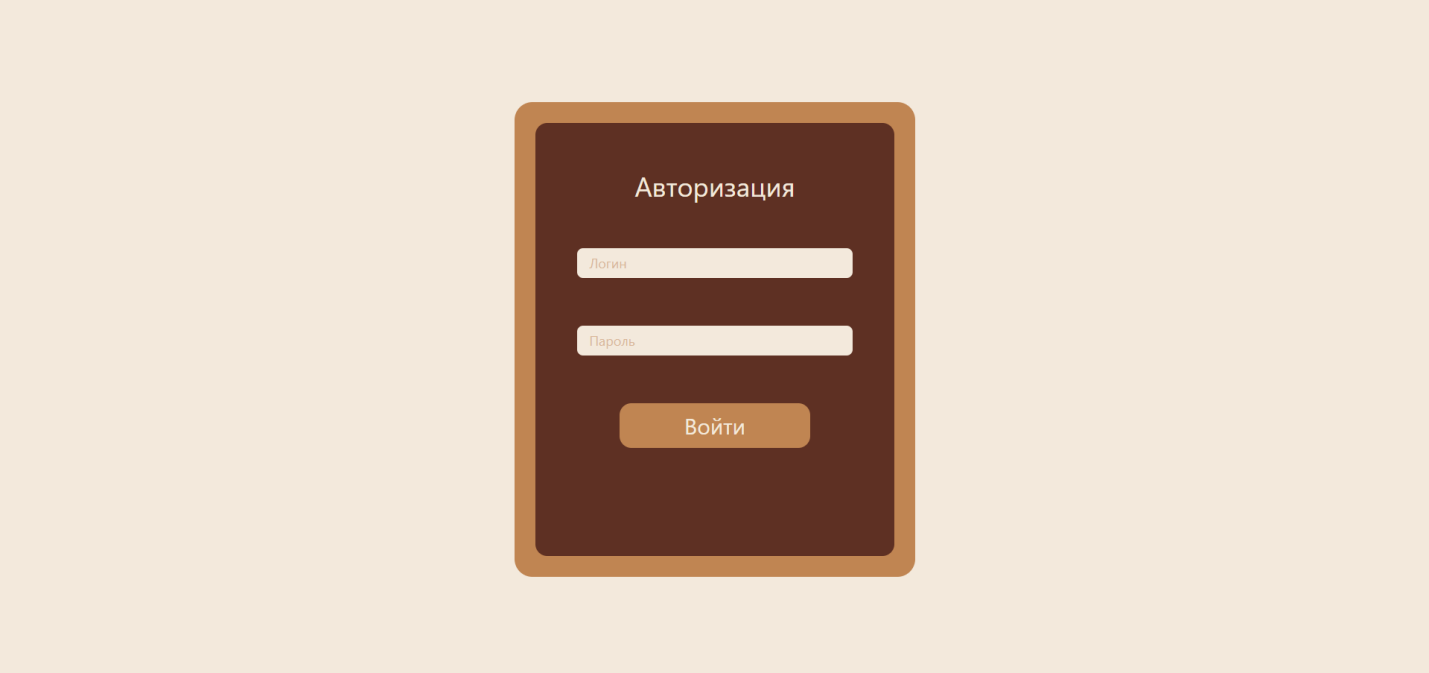


Рисунок 6 - страница авторизации

После входа в аккаунт происходит редирект на главную страницу, где находятся общие сведения о пользователе и задачи, назначенные ему. В случае, если задачи на данный момент отсутствуют, сообщается о том, что пользователь не взял задачи.

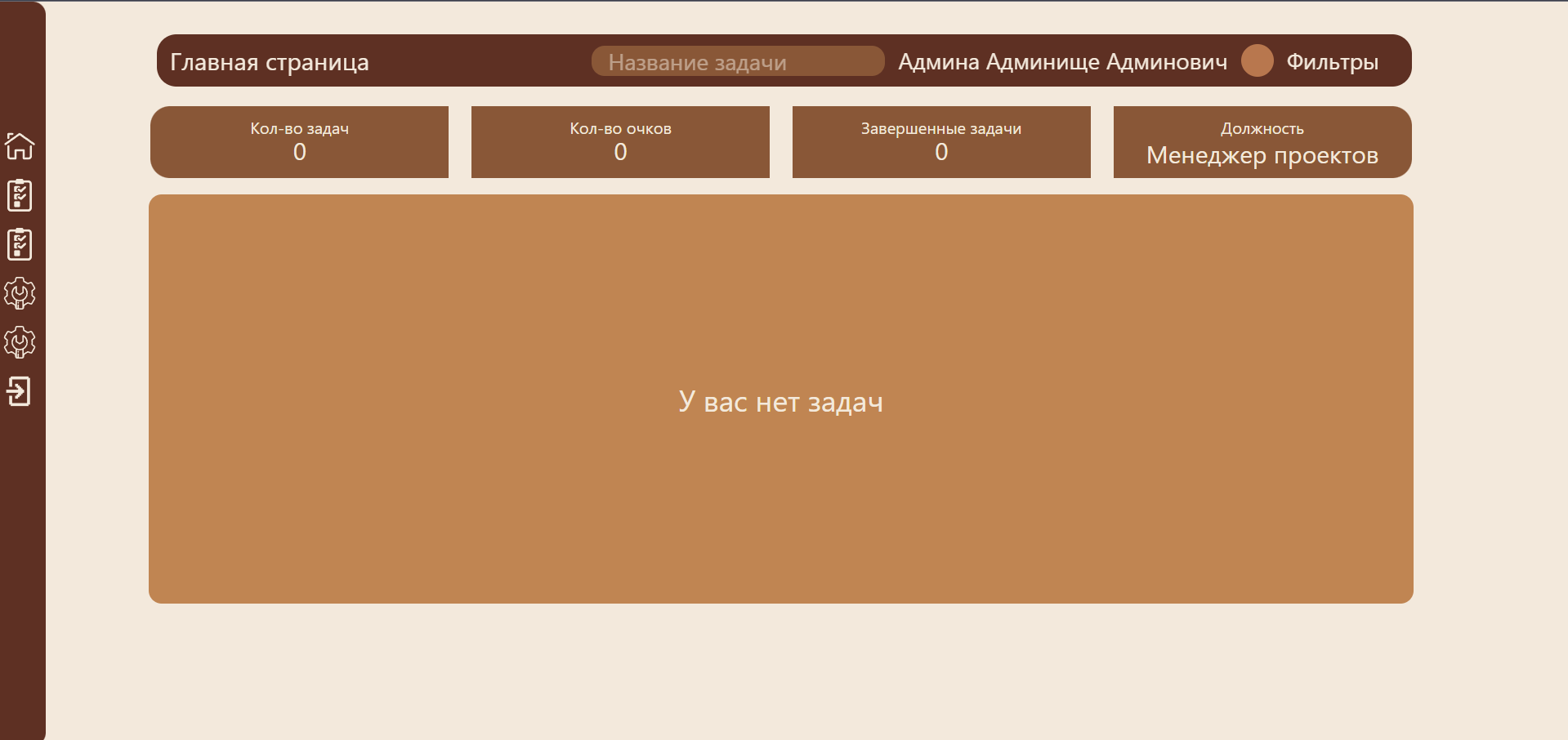


Рисунок 7 - Главная страница

Чтобы взять задачу нужно перейти на панель задач (Рисунок 8) и выбрать задачу из нужной категории, сложности и названию, после чего нажать на кнопку взять задачу. Если список задач просматривает администратор или менджер, ему видна кнопка для удаления задачи. Перед тем как попасть к определенному пользователю, задачи модерируются и, в зависимости от квалификации работника, одобряются или отклоняются.

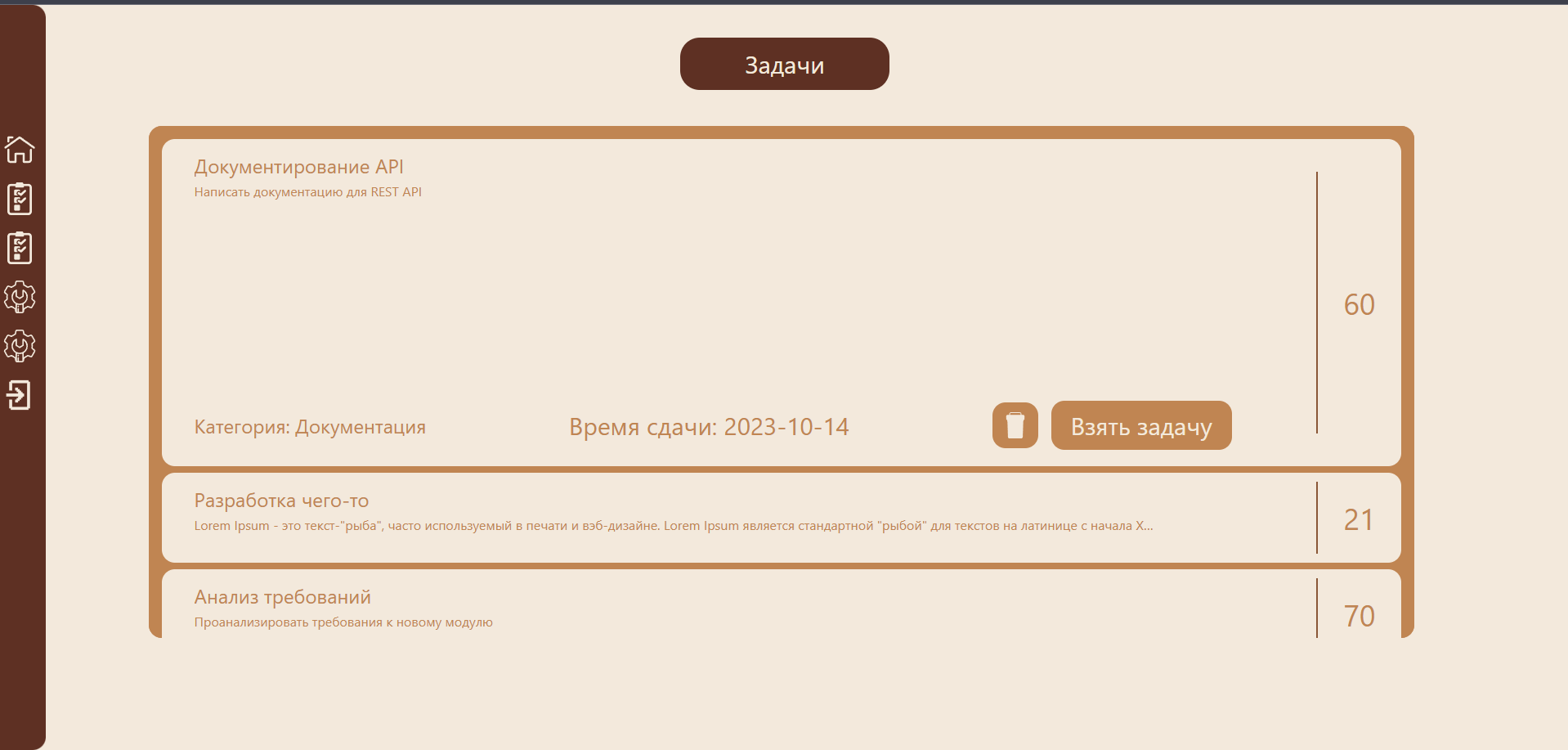


Рисунок - страница задач

На сайте присутствует темная тема (Рисунок 9) – сменяется с помощью нажатия на круг цвета противоположной темы, а также на рисунке продемонстрирована страница с заявками, где можно одобрить/отклонить заявку на задачу и фильтры в header, в зависимости от которых отображается список задач.

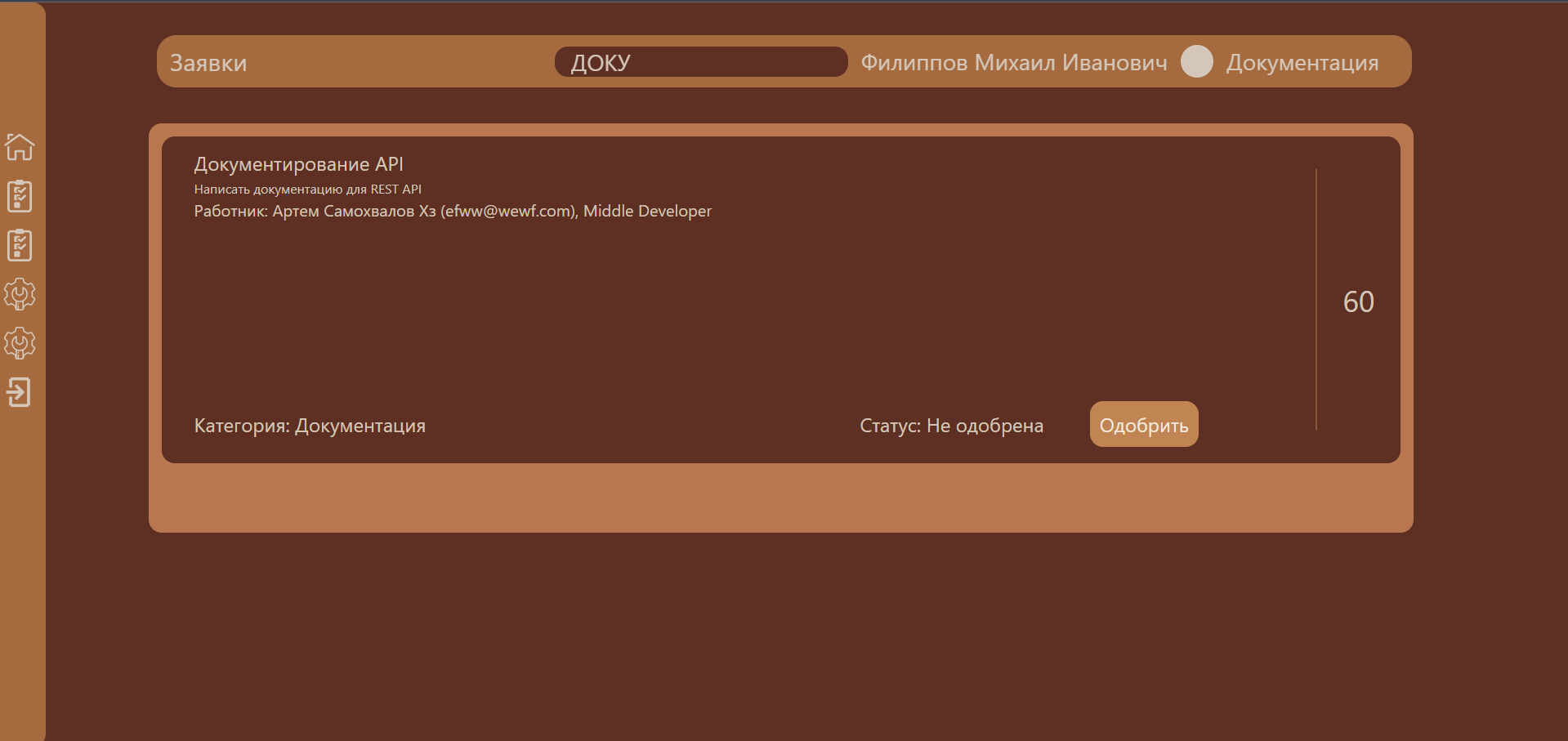


Рисунок - Темная тема и пример заявки

На рисунке ниже (Рисунок 10) представлены навигационный компонент со всеми доступными пользователю ссылками, а также админ панель, доступная только администратору и менджеру (менджеру доступны только функции просмотра и удаления квалификации, должности и категории задач).

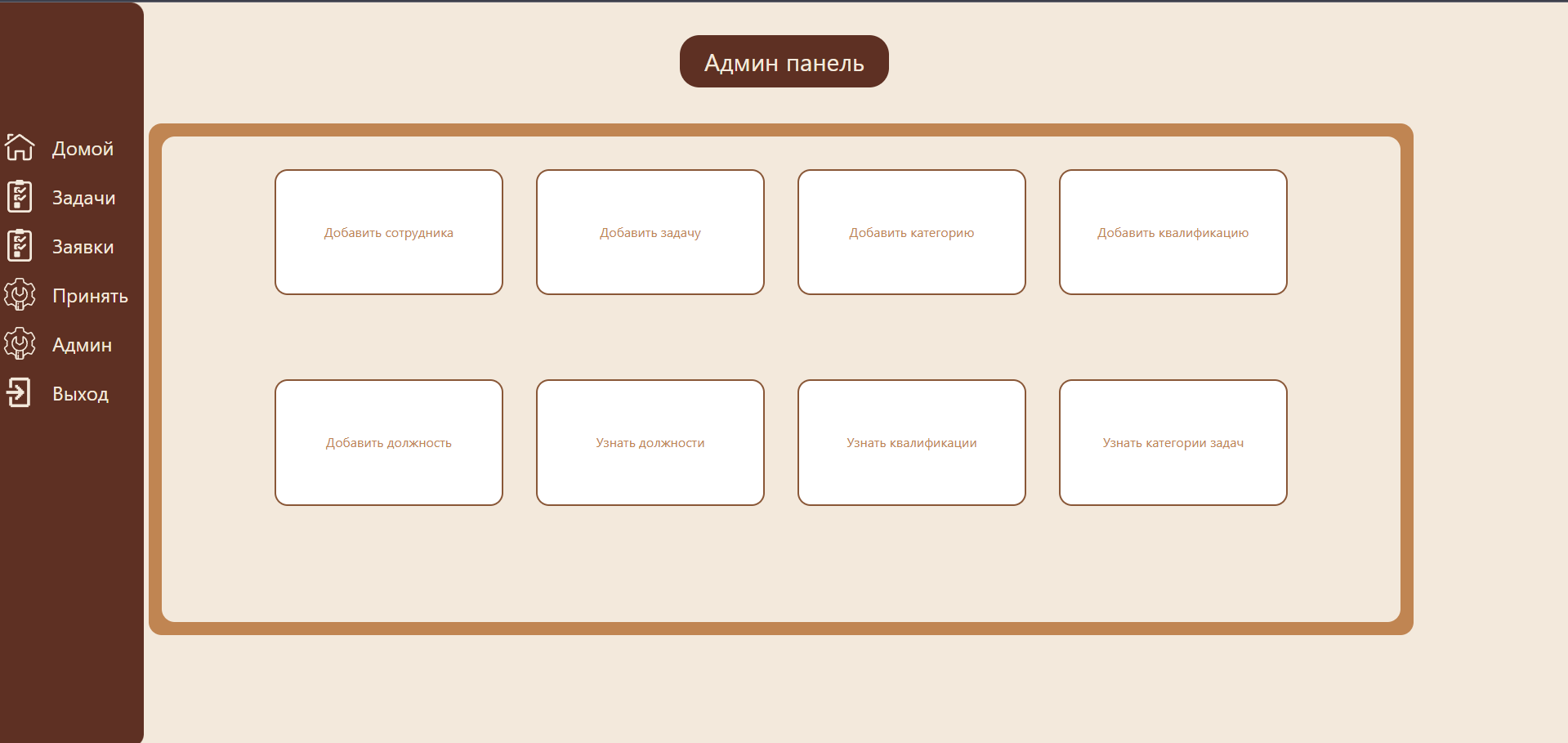


Рисунок - админ панель

При нажатии какого-либо пункта на админ панели появляется модальное окно с формой, которая при заполнении валидируется и отправляет POST-запрос на сервер с полученными данными. На рисунке (Рисунок 11) представлено модальное окно с добавлением сотрудника (поля: логин, почта, пароль, имя, фамилия, отчество, квалификация, роль, номер телефона для связи, номер паспорта, серия паспорта)

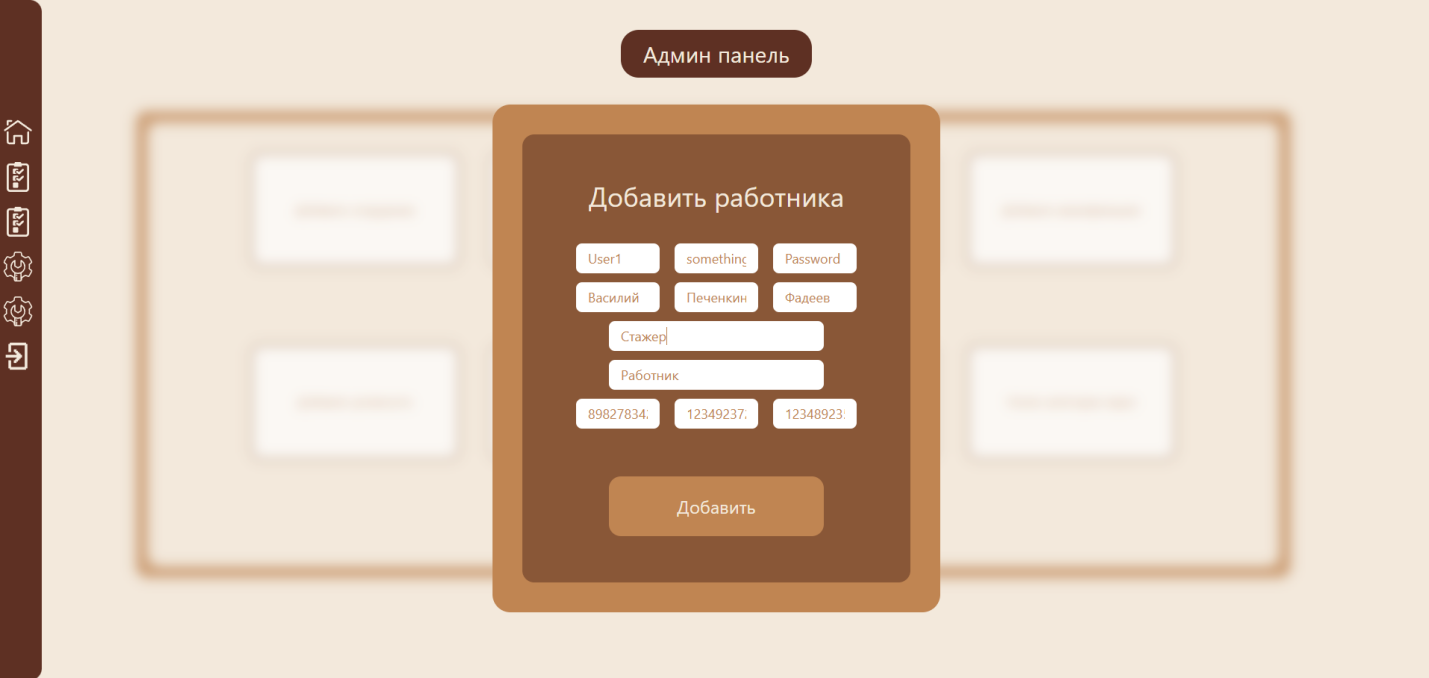


Рисунок - добавление сотрудника

Чтобы установить соединение с БД и связь с сервером было написано API, представленное в графическом виде с помощью библиотеки SwaggerUI (Рисунок 13). Все роуты разбиты на группы и представленны в виде url-путя, на который нужно отправить запрос с указанными в пункте параметрами (Рисунок 12).

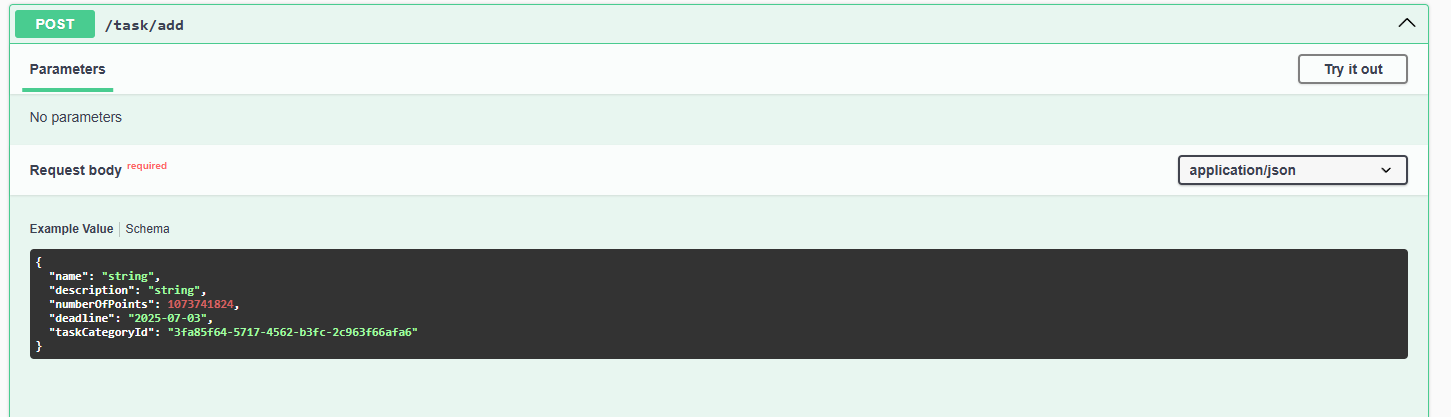


Рисунок - параметры запроса

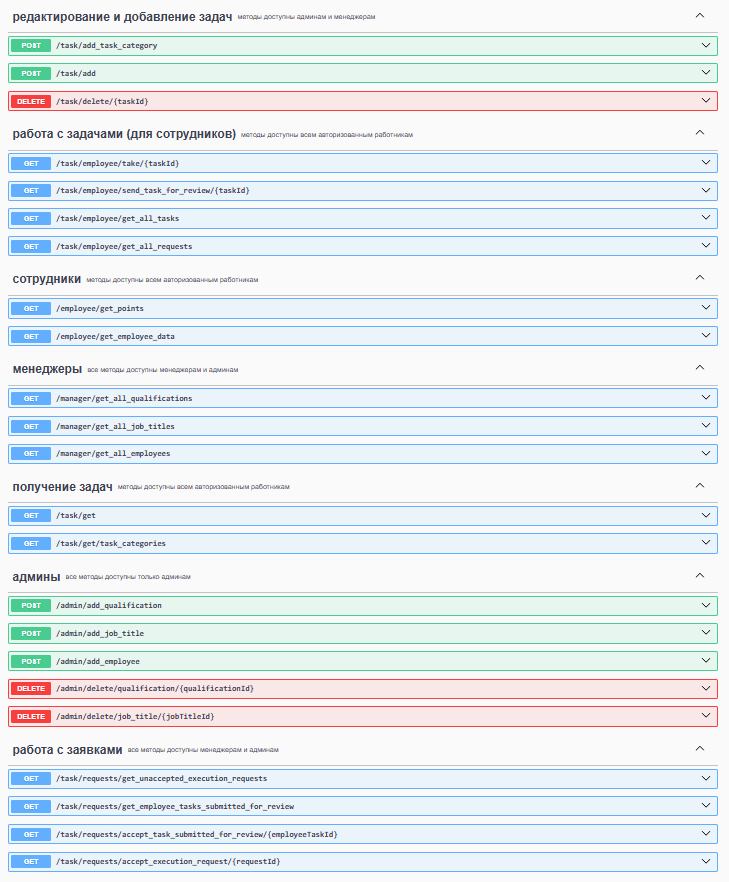


Рисунок 13 – API

2.2 Разработка Базы Данных

Для разработки приложения использовалась СУБД PostgreSQL, основанная на Postgre и содержит следующие таблицы:

* Сотрудники
* Исполнители
* Заявки
* Очки
* ФИО
* Личные данные сотрудников
* Задачи
* Комментарии задач
* Задачи сотрудников
* Запросы на выполнение задач
* Категории задач
* Уровни квалификации для выполнения задач
* Должности
* Квалификации
* Заявки на выполнение задач
* Роли

**Full\_names**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название поля | Тип данных | Ключ | Уникальность | Внешний ключ (Связь) | Значение по умолчанию |
| id | UUID | Primary key |  |  |  |
| first\_name | VARCHAR(255) |  |  |  |  |
| last\_name | VARCHAR(255) |  |  |  |  |
| patronymic | VARCHAR(255) |  |  |  |  |

Таблица 1 - ФИО

Таблица сотрудников в третьей нормальной форме (Таблица 1):

* id: Уникальный идентификатор.
* first\_name: Имя.
* last\_name: Фамилия.
* patronymic: Отчество.

**Job\_titles**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название поля | Тип данных | Ключ | Уникальность | Внешний ключ (Связь) | Значение по умолчанию |
| id | UUID | Primary key |  |  |  |
| name | VARCHAR(255) |  | + |  |  |
| description | TEXT |  |  |  |  |

Таблица 2 - названия работ

Таблица названия работ (Таблица 2):

* id: Уникальный идентификатор.
* name: Название должности.
* description: Описание должности.

**Qualifications**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название поля | Тип данных | Ключ | Уникальность | Внешний ключ (Связь) | Значение по умолчанию |
| id | UUID | Primary key |  |  |  |
| name | VARCHAR(255) |  |  |  |  |
| job\_title\_id | UUID |  |  | job\_titles |  |

Таблица 3 - квалификации

Таблица названия квалификации (Таблица 3):

* id: Уникальный идентификатор.
* name: Название квалификации.
* job\_title\_id: Связь с должностью.

**Roles**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название поля | Тип данных | Ключ | Уникальность | Внешний ключ (Связь) | Значение по умолчанию |
| id | UUID | Primary key |  |  |  |
| name | VARCHAR(255) |  | + |  |  |

Таблица 4 - роли

Таблица названия ролей (Таблица 4):

* id: Уникальный идентификатор.
* name: Название роли.

**Employees**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название поля | Тип данных | Ключ | Уникальность | Внешний ключ (Связь) | Значение по умолчанию |
| id | UUID | Primary key |  |  |  |
| login | VARCHAR(255) |  | + |  |  |
| email | VARCHAR(255) |  | + |  |  |
| password | VARCHAR(255) |  |  |  |  |
| full\_name\_id | UUID |  |  | full\_names |  |
| qualification\_id | UUID |  |  | qualifications |  |

Таблица 5 - сотрудники

Таблица сотрудников (Таблица 5):

* id: Уникальный идентификатор.
* login: Логин.
* email: Почта.
* password: Хеш пароля.
* full\_name\_id: Ссылка на ФИО.
* qualification\_id: Ссылка на квалификацию.

**Employee\_roles**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название поля | Тип данных | Ключ | Уникальность | Внешний ключ (Связь) | Значение по умолчанию |
| employee\_id | UUID | Primary key |  | employees |  |
| role\_id | UUID | Primary key |  | roles |  |

Таблица 6 - роли работников

Таблица ролей доступных пользователям (Таблица 6):

* employee\_id: Сотрудник.
* role\_id: Роль.

**Personal\_employees\_data**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название поля | Тип данных | Ключ | Уникальность | Внешний ключ (Связь) | Значение по умолчанию |
| id | UUID | Primary key |  |  |  |
| employee\_id | UUID |  | + | REFERENCES employees(id) |  |
| phone\_number | VARCHAR(255) |  |  |  |  |
| passport\_number | INT |  |  |  |  |
| passport\_series | INT |  |  |  |  |
| date\_of\_employment | DATE |  |  |  |  |

Таблица 7 - данные пользователя

Таблица информации о пользователе (Таблица 7):

* id: Уникальный идентификатор.
* employee\_id: Ссылка на сотрудника.
* phone\_number: Телефон.
* passport\_number: Номер паспорта.
* passport\_series: Серия паспорта.
* date\_of\_employment: Дата устройства на работу.

**Points**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название поля | Тип данных | Ключ | Уникальность | Внешний ключ (Связь) | Значение по умолчанию |
| id | UUID | Primary key |  |  |  |
| total | INT |  |  |  |  |
| last\_update | DATE |  |  |  |  |
| employee\_id | UUID |  | + | employees |  |

Таблица 8 - очки

Таблица очков пользователя (Таблица 8):

* id: Уникальный идентификатор.
* total: Кол-во баллов.
* last\_update: Дата последнего обновления.
* employee\_id: Ссылка на сотрудника.

**Tokens**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название поля | Тип данных | Ключ | Уникальность | Внешний ключ (Связь) | Значение по умолчанию |
| id | UUID | Primary key |  |  |  |
| employee\_id | UUID |  |  | employees |  |
| token | TEXT |  |  |  |  |

Таблица 9 - токены

Таблица токенов для авторизации (Таблица 9):

* id: Уникальный идентификатор.
* employee\_id: Ссылка на сотрудника.
* token: Сам токен.

**Task\_categories**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название поля | Тип данных | Ключ | Уникальность | Внешний ключ (Связь) | Значение по умолчанию |
| id | UUID | Primary key |  |  |  |
| name | VARCHAR(255) |  | + |  |  |
| description | VARCHAR(255) |  |  |  |  |

Таблица 10 - категории задач

Таблица категорий задач (Таблица 10):

* id: Уникальный идентификатор.
* name: Название категории.
* description: Описание категории.

**Tasks**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название поля | Тип данных | Ключ | Уникальность | Внешний ключ (Связь) | Значение по умолчанию |
| id | UUID | Primary key |  |  |  |
| name | VARCHAR(255) |  |  |  |  |
| description | TEXT |  |  |  |  |
| number\_of\_points | INT |  |  |  |  |
| deadline | DATE |  |  |  |  |
| publication\_time | TIMESTAMP |  |  |  |  |
| status | VARCHAR(50) |  |  |  |  |
| task\_category\_id | UUID |  |  | task\_categories |  |

Таблица 11 - задача

Таблица задачи (Таблица 11):

* id: Уникальный идентификатор.
* name: Название задачи.
* description: Описание.
* number\_of\_points: Кол-во баллов.
* deadline: Срок выполнения.
* publication\_time: Время публикации.
* status: Статус задачи.
* task\_category\_id: Категория задачи.

**Employee\_tasks**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название поля | Тип данных | Ключ | Уникальность | Внешний ключ (Связь) | Значение по умолчанию |
| id | UUID | Primary key |  |  |  |
| description | TEXT |  |  |  |  |
| execution\_status | VARCHAR(50) |  |  |  |  |

Таблица 12 - задачи сотрудника

Таблица задач пользователя (Таблица 12):

* id: Уникальный идентификатор.
* description: Описание выполнения.
* execution\_status: Статус выполнения.

**Qualification\_level\_for\_task**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название поля | Тип данных | Ключ | Уникальность | Внешний ключ (Связь) | Значение по умолчанию |
| id | UUID | Primary key |  |  |  |
| qualification\_id | UUID |  |  | qualifications |  |
| task\_id | UUID |  |  | tasks |  |

Таблица 13 - требуемая квалификация

Таблица требуемой квалификации для задачи (Таблица 13):

* id: Уникальный идентификатор.
* qualification\_id: Квалификация.
* task\_id: Задача.

**Requests**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название поля | Тип данных | Ключ | Уникальность | Внешний ключ (Связь) | Значение по умолчанию |
| id | UUID | Primary key |  |  |  |
| is\_accepted | BOOLEAN |  |  |  |  |
| task\_id | UUID |  |  | tasks |  |
| employee\_id | UUID |  |  | employees |  |

Таблица 14 - запросы

Таблица запрашиваемых задач (Таблица 14):

* id: Уникальный идентификатор.
* is\_accepted: Принята ли заявка.
* task\_id: Задача.
* employee\_id: Сотрудник.

**Task\_comments**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название поля | Тип данных | Ключ | Уникальность | Внешний ключ (Связь) | Значение по умолчанию |
| id | UUID | Primary key |  |  |  |
| comment | TEXT |  |  |  |  |
| date\_of\_writing | TIMESTAMP |  |  |  |  |
| employee\_id | UUID |  |  | employees |  |

Таблица 15 - комментарий к задаче

Таблица комментария к задаче (Таблица 15):

* id: Уникальный идентификатор.
* comment: Текст комментария.
* date\_of\_writing: Дата написания.
* employee\_id: Автор комментария.

**Task\_execution\_requests**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название поля | Тип данных | Ключ | Уникальность | Внешний ключ (Связь) | Значение по умолчанию |
| id | UUID | Primary key |  |  |  |
| is\_accepted | BOOLEAN |  |  |  | false |
| task\_id | UUID |  |  | tasls |  |
| employee\_id | UUID |  |  | employees |  |
|  |  |  |  |  |  |

Таблица 16 - выполненная задача

Таблица выполнения задачи (Таблица 16):

* id: Уникальный идентификатор.
* is\_accepted: Принято ли выполнение.
* task\_id: Ссылка на задачу.
* employee\_id: Сотрудник.

2.3 Описание разработанного функционала

Backend:

1. Получение данных о пользователе:

Метод контроллера обрабатывающий запрос на получение данных о сотруднике доступный только авторизированным пользователям, определен в классе ‘EmployeeController’. Происходит вызов метода ‘getEmployeeData’ в интерфейсе EmployeeService.

@GetMapping("/get\_employee\_data")

public EmployeeDTO employeeData() {

return employeeService.getEmployeeData();

}

Метод получения данных о сотруднике из базы данных. Данные о сотруднике подгружаются с помощью класса ‘UserDetails’ из ‘Spring Security’, который хранит необходимые данные о текущем пользователе; далее по полученному логину производиться запрос к базе данных через ‘employeeRepository’; в результате возвращается объект класса о сотруднике.

@Override

public EmployeeDTO getEmployeeData() {

UserDetailsImpl userDetails = (UserDetailsImpl) SecurityContextHolder.getContext().getAuthentication().getPrincipal();

Employee employee = employeeRepository.findByLoginWithRolesAndQualificationAndJobTitle(userDetails.getUsername()).orElseThrow();

return toEmployeeDTO(employee);

}

Метод интерфейса ‘EmployeeRepository’ для получения полных данных о сотруднике

@Query("""

SELECT e FROM Employee e

JOIN FETCH e.roles

LEFT JOIN FETCH e.qualification eq

LEFT JOIN FETCH eq.jobTitle

WHERE login = :login

""")

Optional<Employee> findByLoginWithRolesAndQualificationAndJobTitle(String login);

1. Получение всех задач в приложении

Метод контроллера для получения задач

@Operation(description = "менеджеры и админы получают все задачи, при этом сотрудники только свободные")

@GetMapping

public List<TaskResponse> getAllTasks() {

return taskService.getAllTasks();

}

Метод сервиса подгружающий из бд задачи и их категории, если пользователь администратор или менеджер, то он получает абсолютно все задачи; если пользователь обычный работник, то получены будут только свободные задачи.

@Override

public List<TaskResponse> getAllTasks() {

return taskRepository.findAllWithCategory().stream()

.filter(this::isPrivateTask)

.map(t -> TaskResponse.builder()

.id(t.getId())

.numberOfPoints(t.getNumberOfPoints())

.name(t.getName())

.deadline(t.getDeadline())

.description(t.getDescription())

.category(TaskCategoryDTO.builder()

.id(t.getTaskCategory().getId())

.name(t.getTaskCategory().getName())

.description(t.getTaskCategory().getDescription()).build()

)

.status(t.getStatus()).build()

).toList();

}

private boolean isPrivateTask(Task t) {

UserDetailsImpl userDetails = (UserDetailsImpl) SecurityContextHolder.getContext()

.getAuthentication().getPrincipal();

if (t.getStatus() != TaskStatus.FREE) {

Set<Role> roles = userDetails.getEmployee().getRoles();

for (Role role : roles) {

if (role.getAuthority().equals("ROLE\_ADMIN") ||

role.getAuthority().equals("ROLE\_MANAGER")) {

return true;

}

}

return false;

}

return true;

}

Запрос на получение задачи и категории из бд

@Query("""

SELECT t FROM Task t

JOIN FETCH TaskCategory tc ON t.taskCategory.id = tc.id

""")

List<Task> findAllWithCategory();

1. Запрос на обновление токенов

Метод контроллера обновляющий токены

@PostMapping("/update\_tokens")

public ResponseEntity<?> updateTokens(@RequestBody Map<String, String> request) {

if (request.get("refreshToken") == null) {

return ResponseEntity.badRequest().body(

Map.of("refreshToken", "поле обязательно к заполнению")

);

}

return ResponseEntity.ok(authenticationService.updateTokens(request.get("refreshToken")));

}

Метод сервиса отвечающий за логику обновления токенов

@Override

@Transactional

public JwtDTO updateTokens(String refreshToken) {

if (!jwtService.validateRefreshToken(refreshToken)) {

throw new RefreshTokenValidationException("срок действия токена истек");

}

Token token = tokenRepository.findByToken(refreshToken)

.orElseThrow(() -> new RefreshTokenValidationException("неправильный токен"));

JwtDTO jwtDTO = new JwtDTO(

jwtService.generateAccessToken(token.getEmployee()),

jwtService.generateRefreshToken(token.getEmployee())

);

token.setToken(jwtDTO.getRefreshToken());

return jwtDTO;

}

Frontend:

1. Получение данных пользователя с сервера:

Функция получает router, хранилище и оператор загрузки, определяющий идет ли загрузка или нет. Данные с сервера попадают в setUserData, откуда отображаются на страницу.

async function getUserData({ setUserData, router, setIsLoading }) {

  const accessToken = localStorage.getItem("accessToken");

  try {

    const getData = await fetch(

      `${env.BACKEND\_API\_URL}/employee/get\_employee\_data`,

      {

        method: "GET",

        headers: {

          "Content-Type": "application/json;charset=utf-8",

          Authorization: `Bearer ${accessToken}`,

        },

      }

    );

    const DataJSON = await getData.json();

    setUserData(DataJSON);

    setIsLoading(false);

  } catch (e) {

    router.push("/login");

  }

}

2. Удаление задачи:

Функция получает id задачи и отправляет запрос на сервер для удаления этой задачи.

function onDelete(e, id) {

    e.stopPropagation();

    const accessToken = localStorage.getItem("accessToken");

    fetch(env.BACKEND\_API\_URL + "/task/delete/" + id, {

      method: "DELETE",

      headers: {

        "Content-Type": "application/json;charset=utf-8",

        Authorization: `Bearer ${accessToken}`,

      },

      body: {

        taskId: id,

      },

    });

    getTasks();

  }

3. Обработка клика по header:

При клике на блок header или какой-нибудь из его внутренних элементов, при всплытии события отрабатывает функция, отображающая скрытое содержимое этого блока.

function mousehandler(e) {

  if (document.documentElement.clientWidth <= 768) return;

  if (e.type == "mouseover")

    document

      .querySelectorAll("#ok")

      .forEach((el) => (el.style.display = "flex"));

  else

    document

      .querySelectorAll("#ok")

      .forEach((el) => (el.style.display = "none"));

}

ЗАКЛЮЧЕНИЕ

Разработка веб-приложения является сложным, но познавательным процессом, который научил меня многому, а также дал возможность понять и осознать те вещи, которые были мне непонятны до этого.

Данное веб-приложение предоставляет возможность отслеживать работу и производительность сотрудников, так как имеет функции добавлять очки за завершенные задачи.

Преимущества проекта:

- Отсутствие рекламы

- Легкое освоение функциональности

- Легко модернизировать и со временем расширять сайт за счет его хорошо

спроектированной структуры

- Понятный и приятный интерфейс

- Авторизация по jwt токенам, набирающая сейчас популярность

Планируется:

- Повышение количества функций, доступных для администраторов

- Добавление нового вида пользователя «Куратор»

- Добавление новых способов метрики, с использованием «Яндекс»

- Добавление проверки задач Куратором

В целом, данный сайт представляет собой хороший инструмент для работы, экономии времени на совещания и обобщает все задачи в одном месте. Однако, важно обеспечить контроль за содержанием сайта, его развитием и защиту от злоумышленников.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Абрамов, Г. В. Проектирование и разработка информационных систем :

учебное пособие для СПО / Г. В. Абрамов, И. Е. Медведкова, Л. А.

Коробова. — Саратов : Профобразование, 2020. — 169 c. — ISBN

2. Стасышин, В. М. Разработка информационных систем и баз данных :

учебное пособие для СПО / В. М. Стасышин. — Саратов :

Профобразование, 2020. — 100 c. — ISBN

3. Конституция РФ.

<https://www.consultant.ru/document/cons_doc_LAW_28399/>

4. Википедия.

https://ru.wikipedia.org/wiki

5. ФЗ-149

https://www.consultant.ru/document/cons\_doc\_LAW\_61798/

6. Зубкова, Т. М. Технология разработки программного обеспечения :

учебное пособие для СПО / Т. М. Зубкова. — Саратов : Профобразование,

2019. — 468 c. — ISBN

7. Синицын, С. В. Основы разработки программного обеспечения на примере

языка С : учебное пособие для СПО / С. В. Синицын, О. И. Хлытчиев. —

Саратов : Профобразование, 2019. — 212 c. — ISBN

8. Горелов, С. В. Современные технологии программирования: разработка

Windows-приложений на языке С#. В 2 томах. Т.I : учебник / С. В. Горелов

; под редакцией П. Б. Лукьянова. — Москва : Прометей, 2019. — 362 c. —

ISBN

9. Фролов, А. Б. Основы web-дизайна. Разработка, создание и сопровождение

web-сайтов : учебное пособие для СПО / А. Б. Фролов, И. А. Нагаева, И. А.

Кузнецов. — Саратов : Профобразование, 2020. — 244 c. — ISBN

10. Спицина, И. А. Разработка информационных систем. Пользовательский

интерфейс : учебное пособие для СПО / И. А. Спицина, К. А. Аксёнов ; под

44 редакцией Л. Г. Доросинского. — 2-е изд. — Саратов, Екатеринбург :

Профобразование, Уральский федеральный университет, 2020. — 98 c. —

ISBN

11. Дэн Абрaмов, Кайл Симпсон. Современный JavaScript. Руководство для разработчиков / пер. с англ. — Москва : БХВ-Петербург, 2022. — 336 c. — ISBN 978-5-9775-1295-5

12. Флэнаган, Д. JavaScript. Подробное руководство / Дэвид Флэнаган ; пер. с англ. — 7-е изд. — СПб. : Питер, 2021. — 1056 c. — ISBN 978-5-4461-1520-9

13. Эккель, Б. Философия Java / Брюс Эккель ; пер. с англ. — 4-е изд. — СПб. : Питер, 2021. — 1168 c. — ISBN 978-5-469-01708-6

14. Шилдт, Г. Java. Полное руководство / Герберт Шилдт ; пер. с англ. — 10-е изд. — М. : Вильямс, 2022. — 1344 c. — ISBN 978-5-8459-2080-1

15. Мартин, А. Практика SQL. Руководство для начинающих / Антон Мартин ; пер. с англ. — Москва : ДМК Пресс, 2021. — 304 c. — ISBN 978-5-97060-913-4