|  |  |
| --- | --- |
| **­­** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОЙ РАБОТЕ***

***НА ТЕМУ:***

***Определение индекса тяжести панкреатита по Рэнсону***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент \_\_ИУ5-56Б\_\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_В. Мажитов\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_А.И. Канев\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2025 г.*

# АННОТАЦИЯ

Расчётно-пояснительная содержит 21 страницу. С приложениями объем составляет 47 страниц. Работа включает в себя 8 диаграмм и 16 изображений системы. В процессе выполнения было использовано 9 источников.

Объектом разработки является система учета банковских договоров, позволяющая клиентам банка оформлять заявки на счета, а сотрудникам банка – принимать решение о одобрении или отклонении заявок.

Цель работы заключается в создании набора программного обеспечения и сопутствующих веб сервисов для учета банковских договоров.

В работе была разработана архитектура веб сервиса, рассчитанного на учет банковских договоров и оформление заявок на счета, разработан интерфейс для взаимодействия с данным веб сервисом, разработан и развернут веб сервер, нативное приложение и прогрессивное веб приложение, способные общаться с данным веб сервисом.

Пояснительная записка содержит 2 приложения.

**СОДЕРЖАНИЕ**

[АННОТАЦИЯ 2](#_Toc185524646)

[ВВЕДЕНИЕ 4](#_Toc185524647)

[ПРЕДМЕТНАЯ ОБЛАСТЬ 6](#_Toc185524648)

[АРХИТЕКТУРА 11](#_Toc185524649)

[АЛГОРИТМЫ 15](#_Toc185524650)

[ОПИСАНИЕ ИНТЕРФЕЙСА 17](#_Toc185524651)

[ЗАКЛЮЧЕНИЕ 25](#_Toc185524652)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 26](#_Toc185524653)

[ПРИЛОЖЕНИЕ А ТЕХНИЧЕСКОЕ ЗАДАНИЕ 27](#_Toc185524654)

[1. Цель 28](#_Toc185524655)

[2. Назначение разработки 28](#_Toc185524656)

[3. Стадии и этапы разработки 28](#_Toc185524657)

[4. Требования к функциональным характеристикам 28](#_Toc185524658)

[5. Требования к составу и параметрам технических средств 33](#_Toc185524659)

[6. Требования к информационной и программной совместимости 33](#_Toc185524660)

[ПРИЛОЖЕНИЕ Б СПИСОК HTTP МЕТОДОВ 35](#_Toc185524661)

# ВВЕДЕНИЕ

В современном банковском секторе важное значение имеет скорость и удобство предоставления услуг клиентам. В условиях высокой конкуренции и цифровой трансформации, внедрение автоматизированных систем подачи заявок становится неотъемлемой частью эффективной работы банка. Одной из таких услуг является открытие счетов в рамках различных договоров. Этот процесс должен быть максимально прозрачным, удобным для клиентов и оперативным для сотрудников банка.

Целью данной работы является разработка системы, предоставляющей интерфейс для учета заявок на оформление банковских счетов, и включающей в себя веб-сервис, веб-приложение и десктопное приложение Tauri.

Эта система предназначена для клиентов банка, желающих подать заявки на открытие счетов в рамках выбранных договоров, а также для сотрудников банка, занимающихся обработкой таких заявок. Клиенты могут оформлять заявки, выбрав нужные договора. Система обеспечивает автоматизированный процесс создания, отслеживания и управления заявками на открытие счетов, предоставляя сотрудникам банка возможность подтверждать или отклонять поступившие заявки.

Система автоматизирует процесс подачи и обработки заявок, обеспечивая удобство для клиентов и эффективность работы сотрудников банка. Реализован доступ к подробной информации о заявках на счета, что позволяет оперативно принимать решения и вести учет.

Нефункциональные требования к разрабатываемой системе:

1. Должна поддерживаться кроссплатформенность.
2. Интерфейс системы и текст ошибок должны быть русифицированы.

В ходе работы необходимо выполнить следующие задачи:

* 1. Создать MVP и базового дизайн на основе sberbank.ru
  2. Создать базу данных для хранения информации о договорах и заявках на открытие счетов
  3. Создать веб-сервис на Golang
  4. Реализовать авторизацию и хранение отозванных JWT в Redis
  5. Разработать базовый SPA на React для гостя
  6. Внедрить адаптивность, менеджер состояний Redux Toolkit, PWА, разработать Tauri приложение
  7. Завершить разработку интерфейса клиента банка в React, использовать для обращений к методам веб-сервиса Axios
  8. Реализовать интерфейс сотрудника банка в React
  9. Разработать десктопное приложение Tauri
  10. Развернуть приложение при помощи GitHub Pages
  11. Подготовить набор документации, включающий РПЗ, ТЗ и набор диаграмм
  12. Оформить git-репозиторий на сервисе GitHub, содержащий исходный код проекта.

# ПРЕДМЕТНАЯ ОБЛАСТЬ

Цифровые технологии играют ключевую роль в трансформации банковской сферы, создавая новые подходы к предоставлению финансовых услуг и взаимодействию с клиентами [1]. В последние годы банки стремятся интегрировать инновации, чтобы оставаться конкурентоспособными на фоне активного развития финтех-компаний и открытого банкинга. Одним из центральных элементов этой трансформации является использование интерфейсов программирования приложений, которые поддерживают цифровизацию продуктов и услуг [2].

В своей статье «Цифровые финансовые услуги и инновации открытого банкинга: становятся ли банки невидимыми?» Валерия Стефанелли, Франческо Манта и Пьерлуиджи Тома исследуют стратегическое позиционирование европейских банков в процессе цифровой трансформации. Авторы сосредоточились на анализе партнерств с финтех-компаниями и внедрении технологий API в период с 2015 по 2019 годы. Основное внимание уделено тому, каким образом цифровые стратегии банков соответствуют рыночным требованиям [2]. В исследовании использовались такие метрики, как количество разработанных API и число заключенных стратегических партнерств с финтех-компаниями. Эти параметры оценивались как индикаторы уровня технологической открытости и инновационности банков.

Результаты показали, что банки можно разделить на четыре группы: последователи, инноваторы, технологические центры и невидимые банки [2].

Идеальным классом банков авторы называют «невидимые банки». Эти банки сочетают высокую технологическую зрелость с активным использованием стратегических партнерств. Примерами являются банки, которые разработали более 15 API и заключили от 20 до 35 стратегических соглашений. Такие банки активно внедряют инновации, что позволяет им опережать конкурентов.

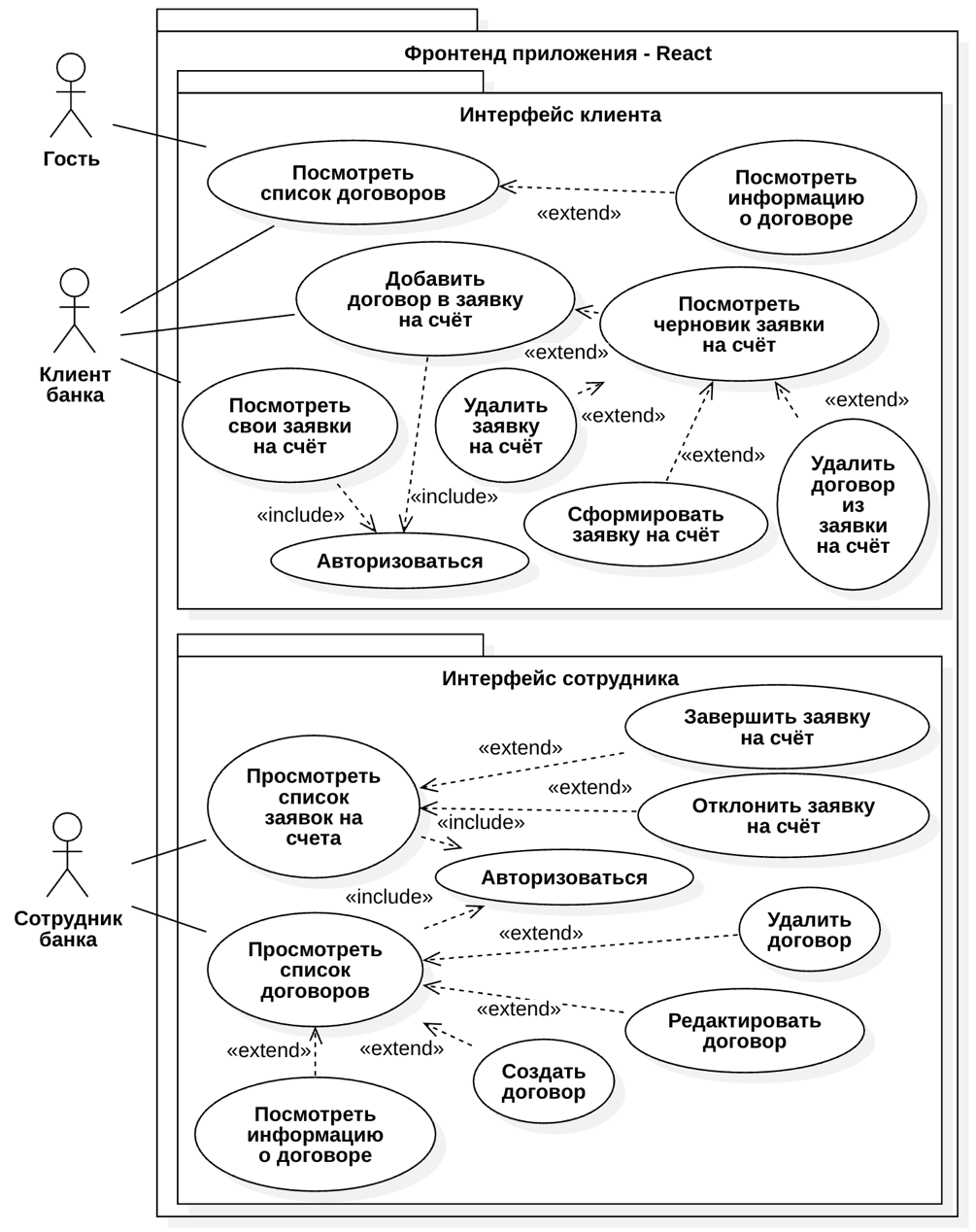
Выводы авторов подчеркивают важность проактивного подхода к цифровизации для сохранения конкурентоспособности. Банки, активно инвестирующие в разработку API и стратегические партнерства, обеспечивают себе лучшее положение на рынке. Также авторы подчёркивают влияние региональных факторов на уровень цифровизации банков. К примеру, в странах с высокой цифровой зрелостью, таких как Нидерланды и Великобритания, проникновение финтеха среди пользователей превышает 70%, а рост использования финтех-услуг с 2014 по 2019 год составил более 400%. Однако отсутствие цифровой грамотности у значительной части клиентов, особенно в странах с низким уровнем цифровизации (Италия, Франция), представляет вызов для широкого внедрения инноваций [2].

Авторы предоставили системный анализ цифровой трансформации в банковской отрасли, опираясь на количественные и качественные показатели. Однако исследование ограничено небольшим размером выборки и отсутствием временного ряда для анализа динамики. Перспективным направлением могло бы стать изучение аналогичных процессов в других регионах с более развитым финтех-сектором, что позволило бы выявить лучшие практики для европейских банков.

Клиент банка (bank customer) может посмотреть список всех созданных им заявок. В случае, если заявка на счёт составлена корректно, сотрудник банка (bank employee) утверждает (approve) её. В противном случае – отклоняет (reject). Сотрудник банка может только либо согласовать, либо отклонить заявку.

Сотрудники банка, могут редактировать список договоров (banking contract), в рамках которых возможно открытие счёта (account): добавлять новые, удалять неактуальные, а также редактировать информацию о каждом отдельно взятом договоре. Сотрудники банка имеют возможность просматривать все заявки на счета (account request), созданные клиентами банка, а также фильтровать их по имени клиента, дате формирования и статусу. После того, как заявка на счёт была согласована (accept) или отклонена (reject), сотрудник банка не может изменять её статус.

Функции пользователей с различными ролями описаны на диаграмме прецедентов (рисунок 1).

Рисунок 1 – Диаграмма прецедентов

Гостям доступен просмотр списка доступных договоров. Гости, прошедшие этап регистрации, являются клиентами банка. Клиенты банка могут добавлять договоры в заявку на открытие счёта, формировать её, а также просматривать список своих заявок.

Клиент выбирает договоры, в рамках которых намеревается открыть счёт, затем на основе выбранного списка формирует заявку. После чего сотрудник банка может просмотреть её, а также согласовать или отклонить. Возможные состояния статуса заявки на открытие счёта отображены на диаграмме состояний (рисунок 2).

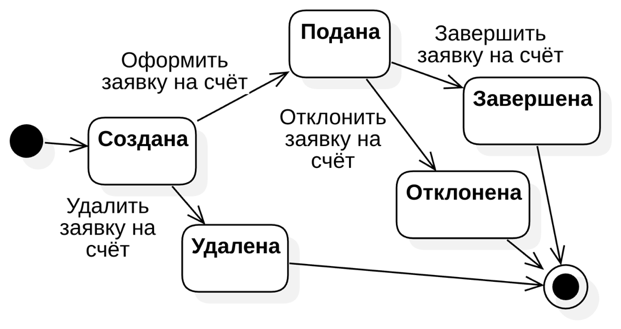


Рисунок 2 – Диаграмма состояний заявок

Заявки обрабатываются сотрудниками банка. В результате обработки заявки её либо одобряют, либо отклоняют. Сотруднику также доступны операции для работы с договорами: просмотр всех договоров, редактирование, создание и удаление договоров, а также просмотр списка всех договоров в табличном виде.

Процесс оформления заявки на счёт отражен на диаграмме деятельности (рисунок 3).

В начале взаимодействия с системой оформления заявок на банковские счета клиент банка запрашивает список доступных для открытия счета договоров. При необходимости клиент может добавить договоры в текущую черновую заявку. После чего может продолжить выбор договоров из каталога или перейти к оформлению заявки. В случае, если клиент выбрал оформление заявки на открытие банковского счёта, ему необходимо выбрать договор, который будет привязан к счёту после открытия. После чего указать номер счёта. По нажатии кнопки «Оформить» заявка на счёт формируется и становится доступна в интерфейсе сотрудника банка.

Сотрудник банка имеет возможность просматривать заявки клиентов, а также принимать решение об одобрении или отклонении заявки.

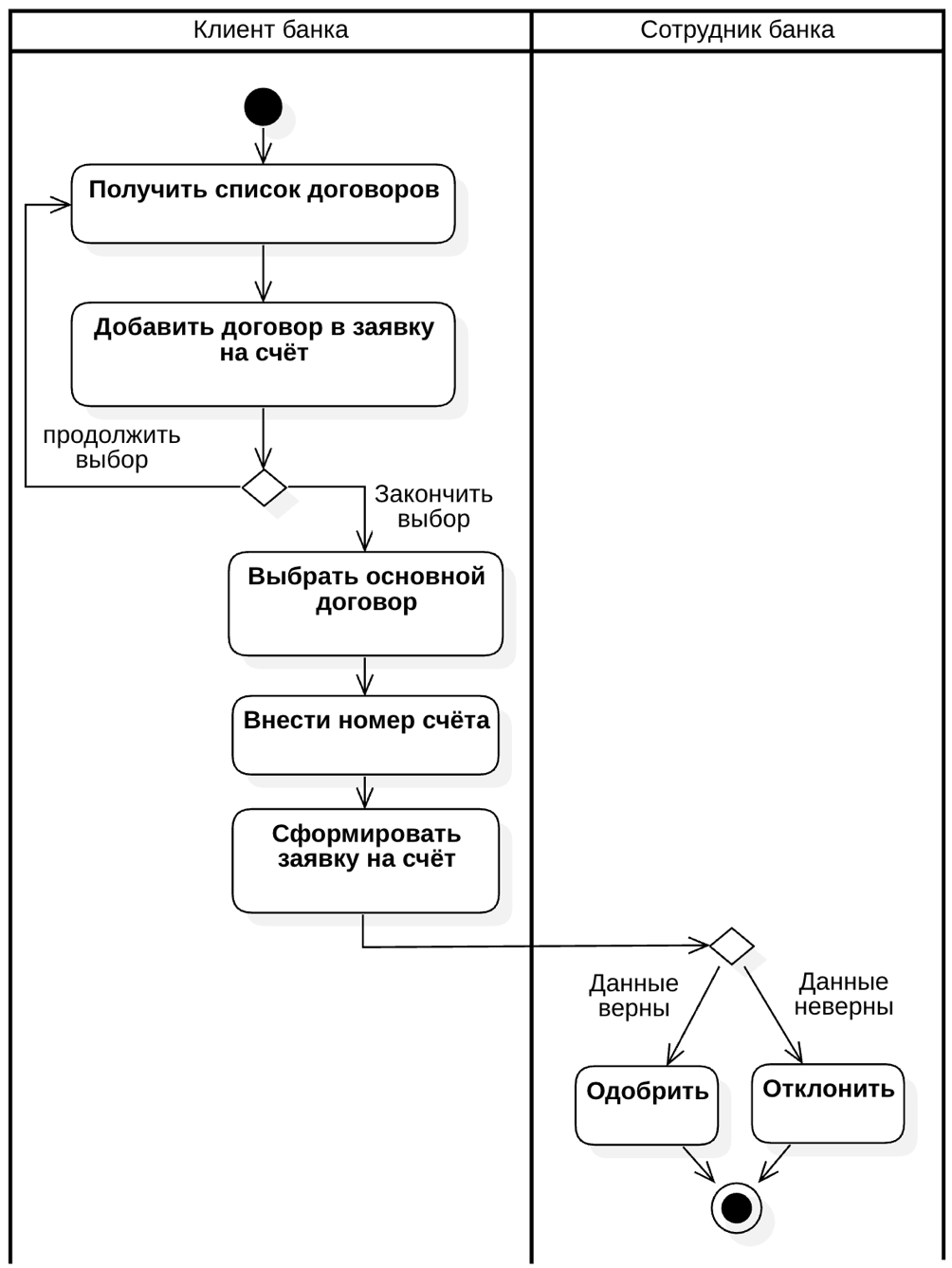
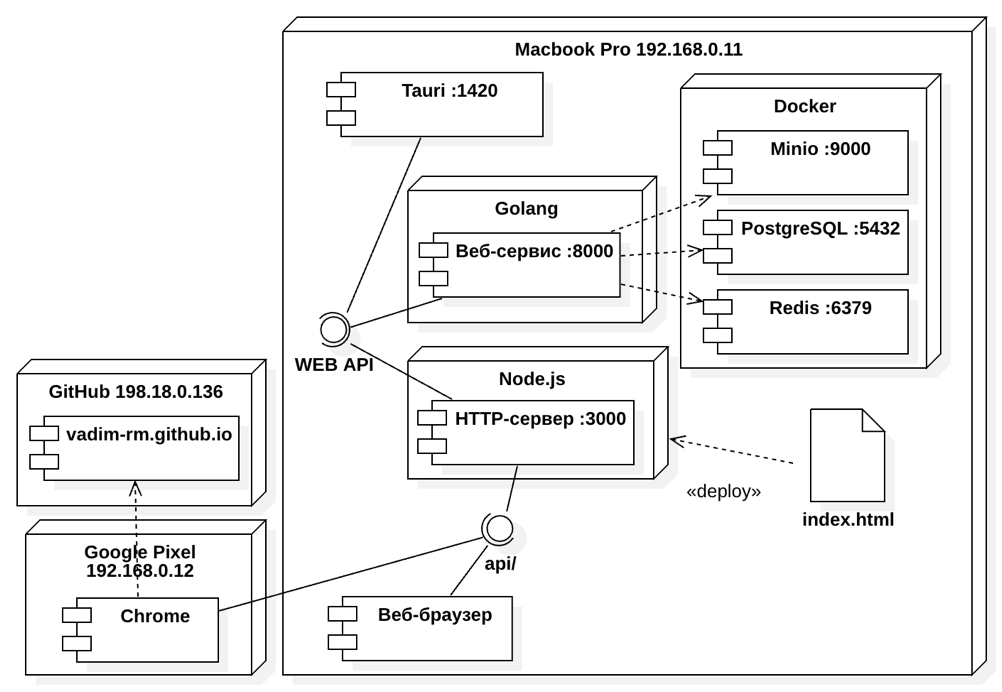


Рисунок 3 – Диаграмма деятельности

# АРХИТЕКТУРА

Архитектура системы отображена на диаграмме развертывания (рисунок 4). Рисунок 4 – Диаграмма развертывания

Веб-сервис, реализованный на языке Golang [3] связан с серверами Minio [6], Redis [5] и СУБД PostgreSQL [7]. В Redis хранятся отозванные JWT [4] пользователей, добавление которых происходит при выходе пользователя из системы.

Язык программирования Golang был выбран благодаря его высокой производительности и простоте синтаксиса. Использование стандартной библиотеки Go и встроенных инструментов для работы с сетевыми запросами позволяет создавать надежные и эффективные веб-сервисы. Веб-сервис на Golang является общим для веб-сервера и десктопного приложения Tauri [8].

Данные хранятся в СУБД PostgreSQL. Она была выбрана, является стандартом современной индустрии разработки.

Структура данных отражена на ER диаграмме (рисунок 5). Модель договоров представляет собой набор полей, отражающих параметры договора. Данные о договоре хранятся в таблице contracts. Для хранения в одной заявке на счёт нескольких договоров используется промежуточная таблица связи М-М account\_contracts. Таблица accounts представляет собой список заявок на открытие счетов. Данные о пользователях системы (клиентах и сотрудниках) хранятся в таблице users.

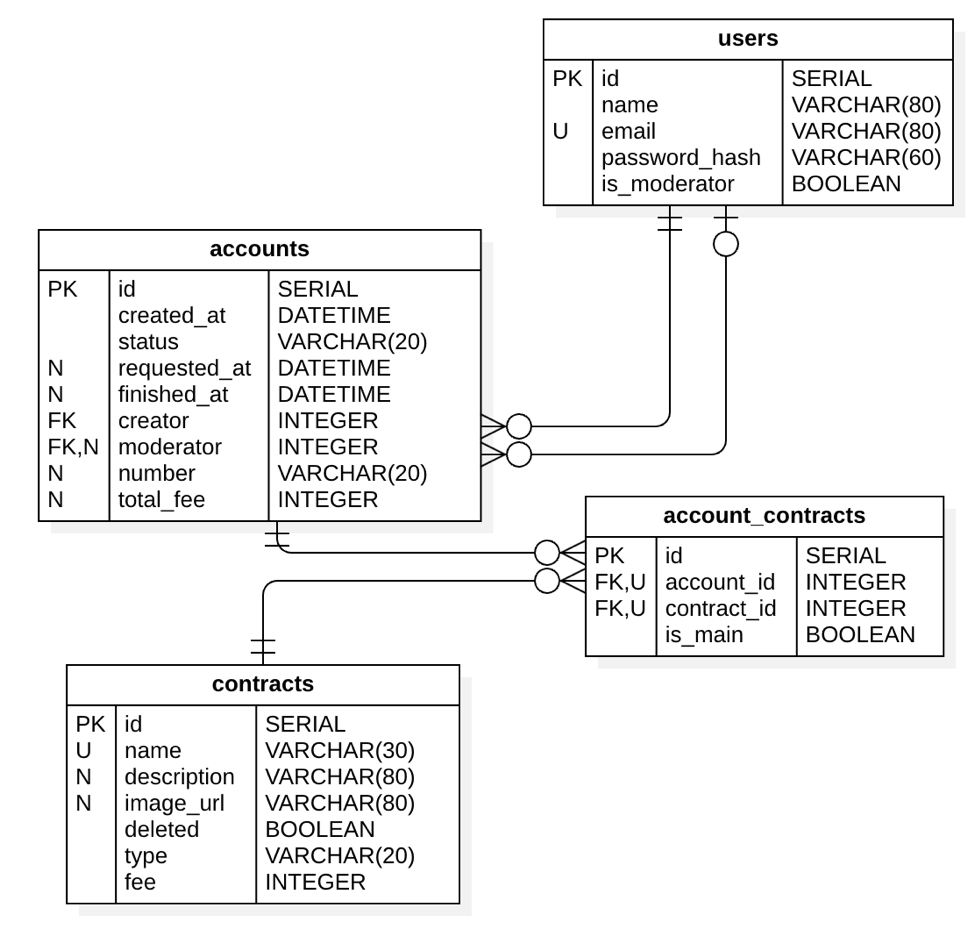


Рисунок 5 – ER диаграмма.

Веб-сервер реализован с использованием языка TypeScript и фреймворка React [7]. Устройство бэкенда приложения разработанной системы приведено на диаграмме классов бэкенда (рисунок 6). Пользователи взаимодействуют с доменами. Домены связаны с моделями. Модели имеют связи с таблицами в базе данных.

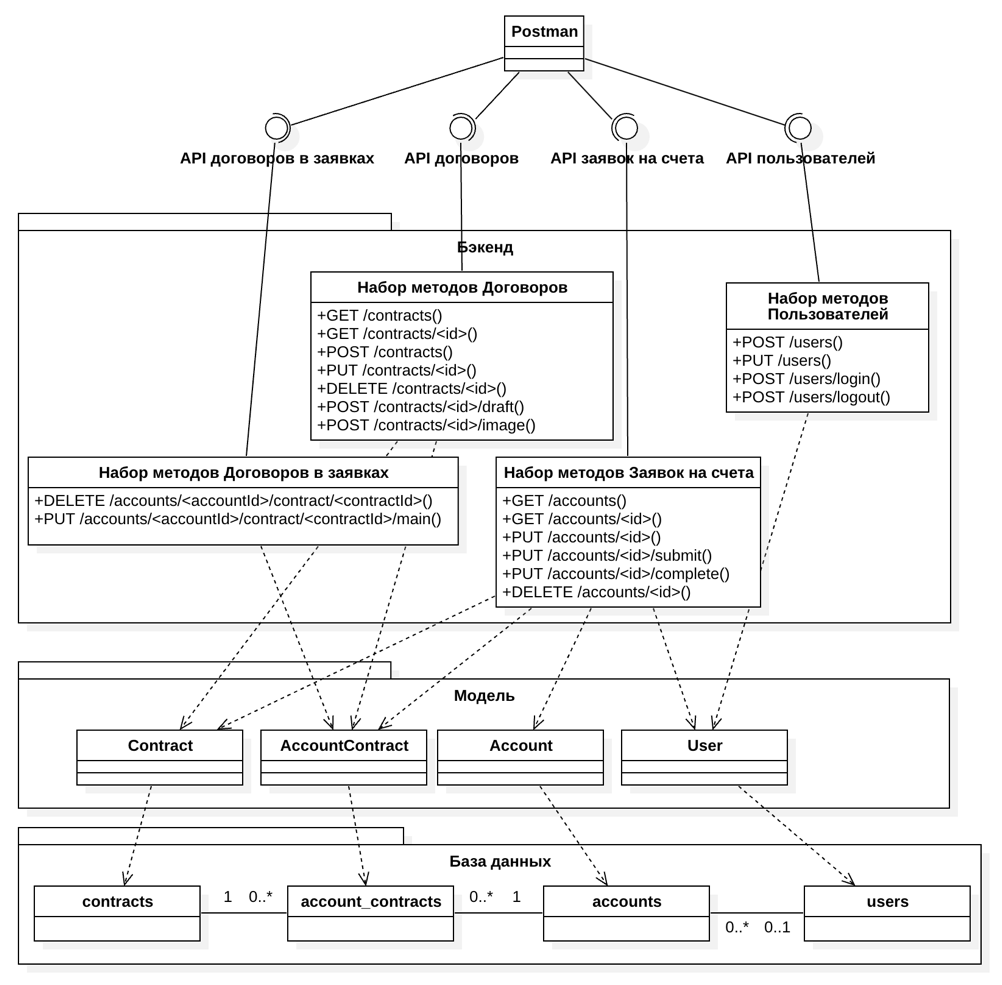


Рисунок 6 – Диаграмма классов бэкенда

Связь фронтенда и бэкенда отражена на диаграмме классов фронтенда (рисунок 7). Каждая страница связана с API, которое используется для взаимодействия с данными на соответствующей странице.

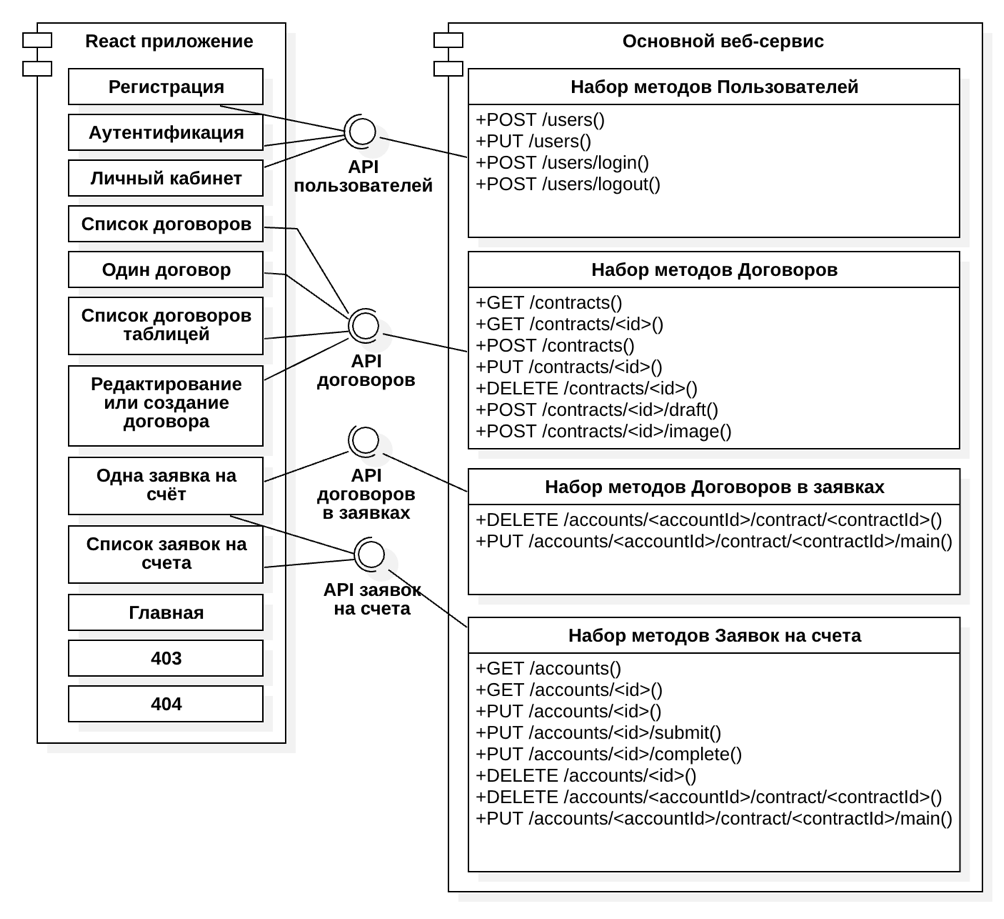


Рисунок 7 – Диаграмма классов фронтенда

# АЛГОРИТМЫ

Алгоритм работы разработанной системы отображен на диаграмме последовательности (рисунок 8). В основе системы лежит веб-сервис, реализующий внутри себя всю бизнес-логику. Он предоставляет доступ к методам из следующих доменов: пользователи, договоры, заявки на счета, договоры в заявках на счета. Методы следуют правилам REST API.

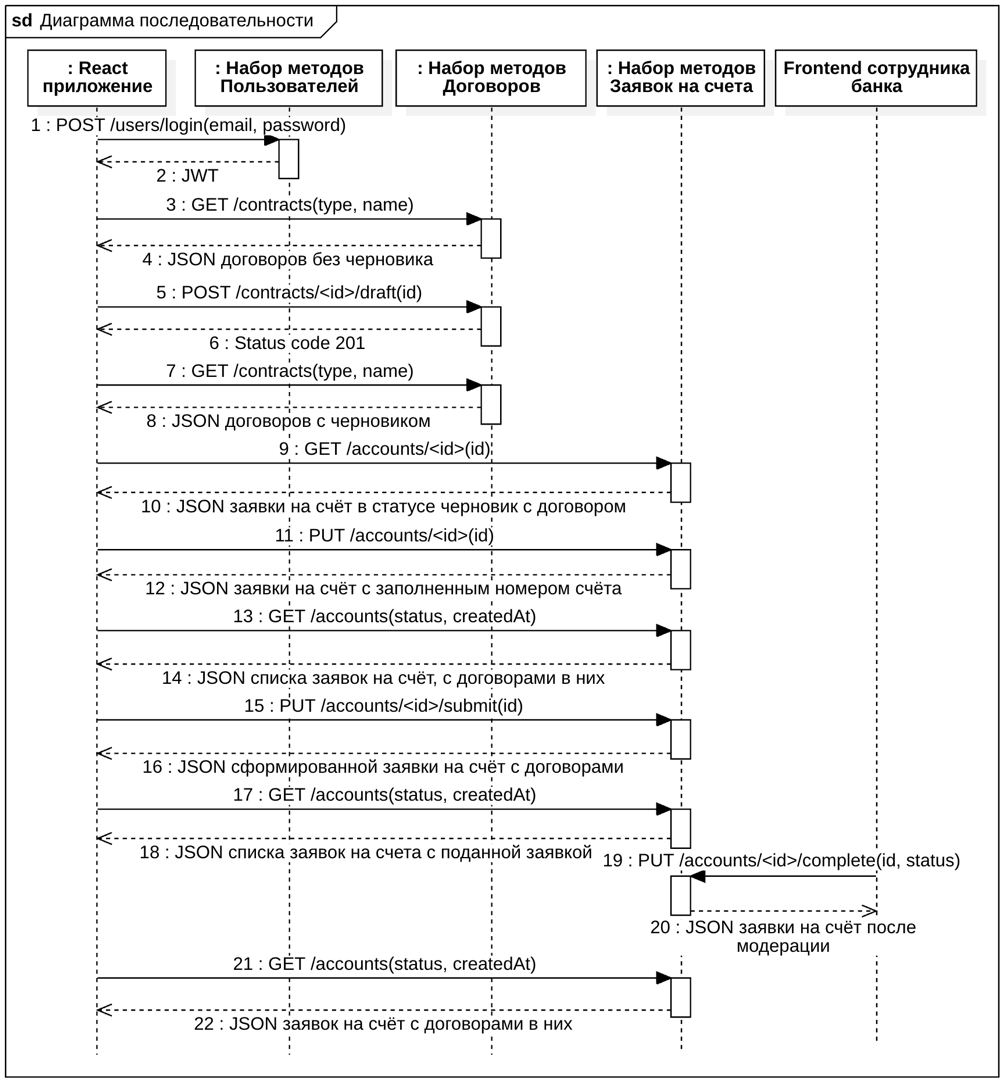


Рисунок 8 – Диаграмма последовательности

В начале бизнес-процесса происходит аутентификация пользователя. Через графический интерфейс клиент банка вводит логин и пароль для доступа к системе. Если учетная запись с указанными данными существует в базе, сервис возвращает информацию о пользователе и JWT в ответе. Если же учетной записи с введённым логином не существует или пароль неверен, клиент получает сообщение об ошибке. В таком случае клиенту предлагается пройти регистрацию или повторно ввести корректные данные. На этом этапе система также определяет роль пользователя — клиент или сотрудник банка.

После успешной аутентификации клиент через графический интерфейс запрашивает у веб-сервиса список доступных договоров, который возвращается в формате JSON. Клиент может выбрать интересующий его договор и добавить его в заявку на открытие счета. Для этого он нажимает кнопку «оформить» в графическом интерфейсе. Процесс выбора договоров и их добавления в черновую заявку может повторяться несколько раз.

Когда клиент завершает выбор договоров и указывает необходимые данные для открытия счета (номер счёта и основной договор), он нажимает кнопку «сформировать заявку». Приложение отправляет на веб-сервис запрос на создание заявки. После этого клиент может отслеживать статус своих заявок на специальной странице интерфейса, где отображаются все его созданные заявки с указанием текущего статуса.

Сотрудники банка, выполняющие роль модераторов, также работают через графический интерфейс. Они имеют доступ к списку всех заявок, созданных клиентами, и могут просматривать их, согласовывать или отклонять с помощью соответствующих кнопок. Сотрудникам доступна фильтрация заявок по логинам клиентов, статусам или дате создания. Также они могут изменять информацию о договорах, доступных для выбора клиентами, и редактировать детали уже существующих договоров. Для выполнения каждой из этих операций интерфейс отправляет соответствующий запрос на веб-сервис, обеспечивая необходимую функциональность.

# ОПИСАНИЕ ИНТЕРФЕЙСА

Главное меню приложения включает пункты, которые доступны в зависимости от роли пользователя (рисунки 9, 10, 11).



Рисунок 9 – Меню приложения (для гостя)



Рисунок 10 – Меню приложения (для клиента)



Рисунок 11 – Меню приложения (для менеджера)

На странице с формой регистрации (рисунок 12) гость может создать новый аккаунт. После успешной регистрации открывается форма аутентификации (рисунок 13).

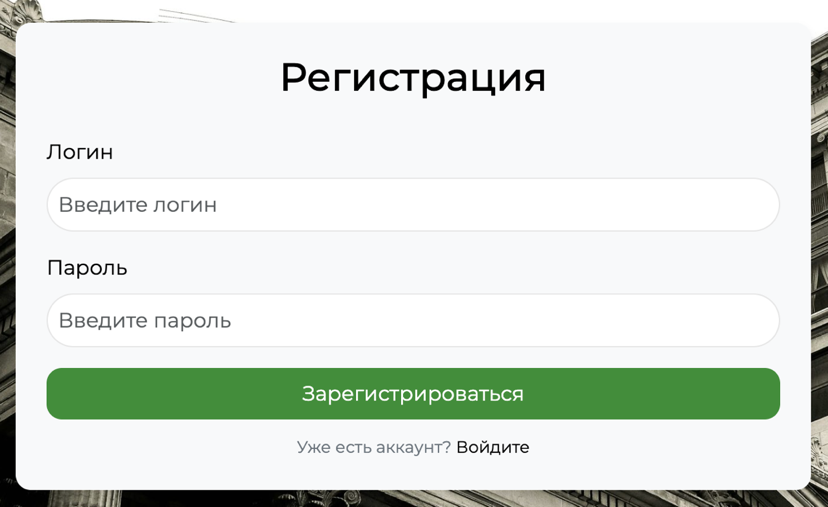


Рисунок 12 – Страница регистрации

На странице с формой входа (рисунок 13) пользователь может войти в свой аккаунт. При успешном вводе данных аккаунта на клиент приходит JWT токен, который хранится в local storage.

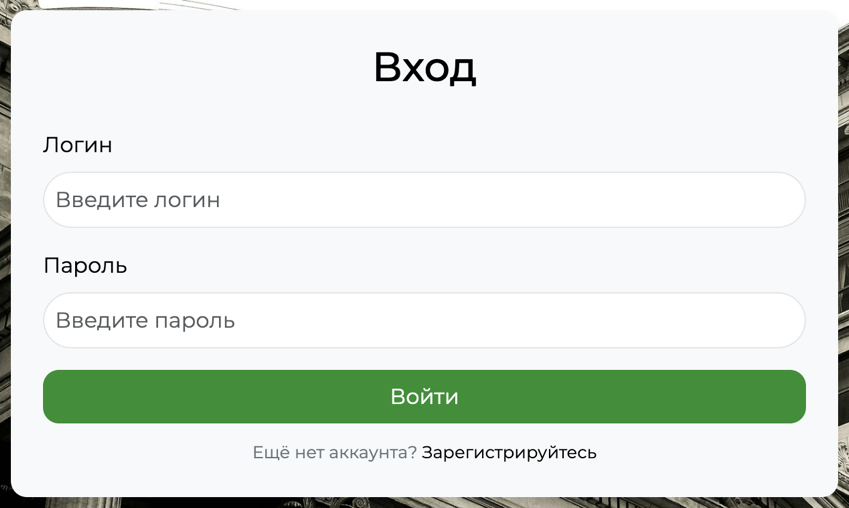


Рисунок 13 – Страница входа

После входа в аккаунт пользователь попадает на главную страницу (рисунок 14). На этой странице располагается описание разработанной системы.

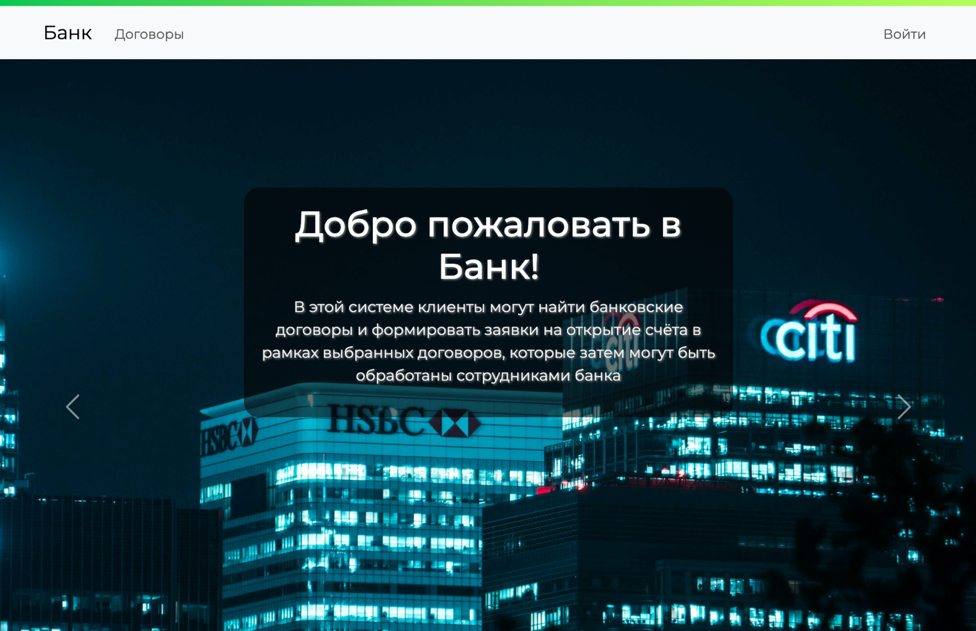


Рисунок 14 – Главная страница

Страница списка договоров (рисунок 15) содержит список доступных к оформлению договоров. При нажатии на кнопку «Подробнее» на любой карточке, открывается страница с подробным описанием. Также с помощью нажатия на кнопку «Оформить» клиент банка может добавлять договоры в черновик заявки. Вверху страницы находится строка для поиска договора, а также кнопки для фильтрации договоров по типу.

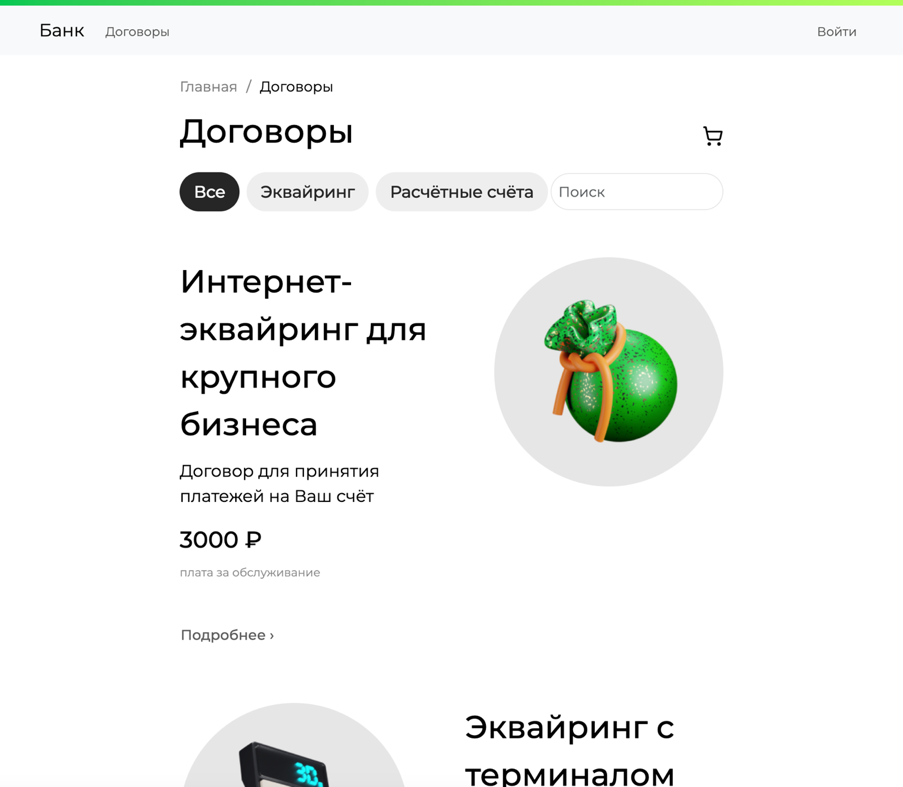


Рисунок 15 – Страница списка договоров

Страница с подробным описанием договора выглядит следующим образом (рисунок 16).

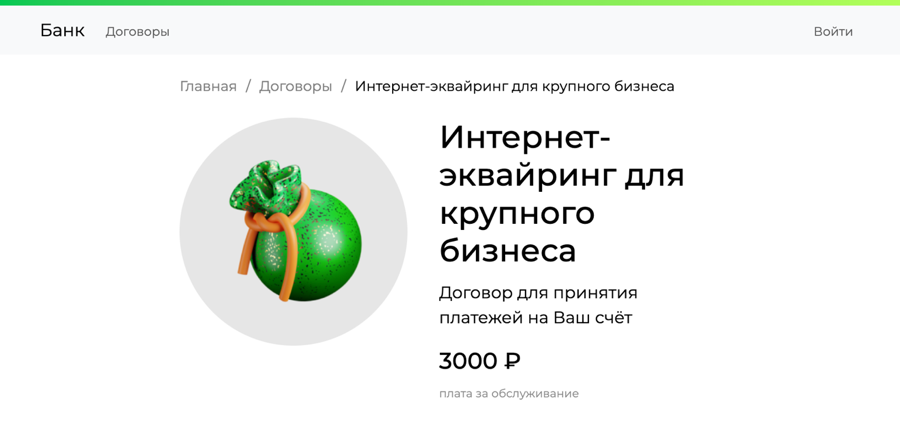


Рисунок 16 – Страница с подробным описанием договора

На странице одной заявки (рисунок 17) отображается текущая заявка на счёт пользователя, есть возможность удалить договор из заявки, выбрать главный договор для счёта, указать номер счёта, а также отправить заявку на проверку или удалить ее. Редактирование оформленных заявок запрещено.

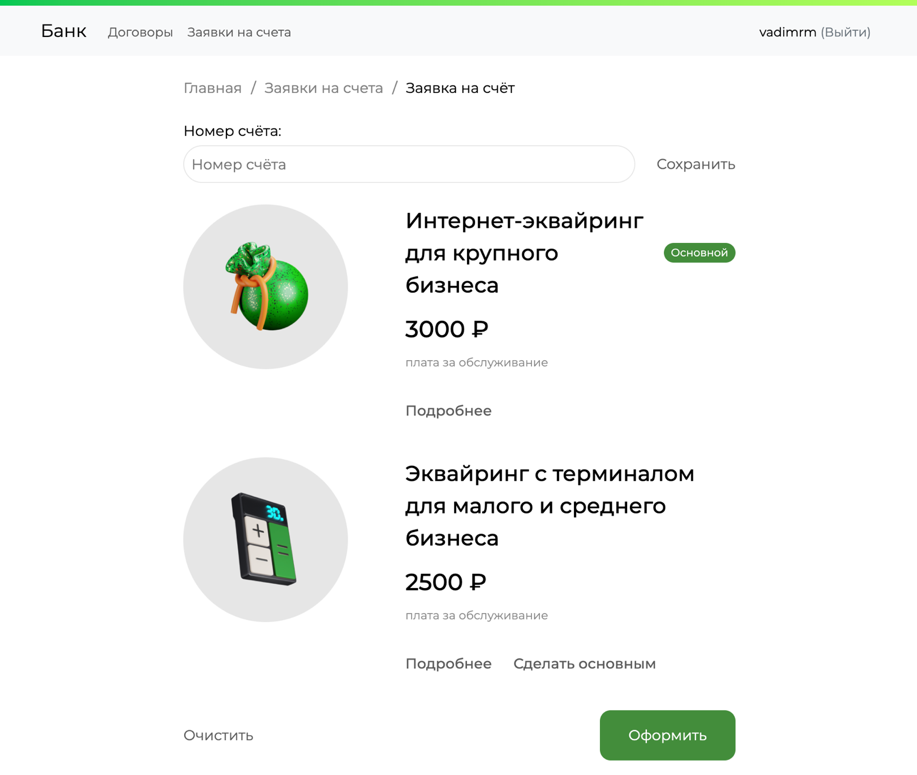


Рисунок 17 – Страница заявки на открытие счёта

На странице списка заявок (рисунок 18) клиенты могут просматривать созданные ими заявки. На этой странице можно посмотреть подробную информацию о заявке, нажав на кнопку «Открыть».

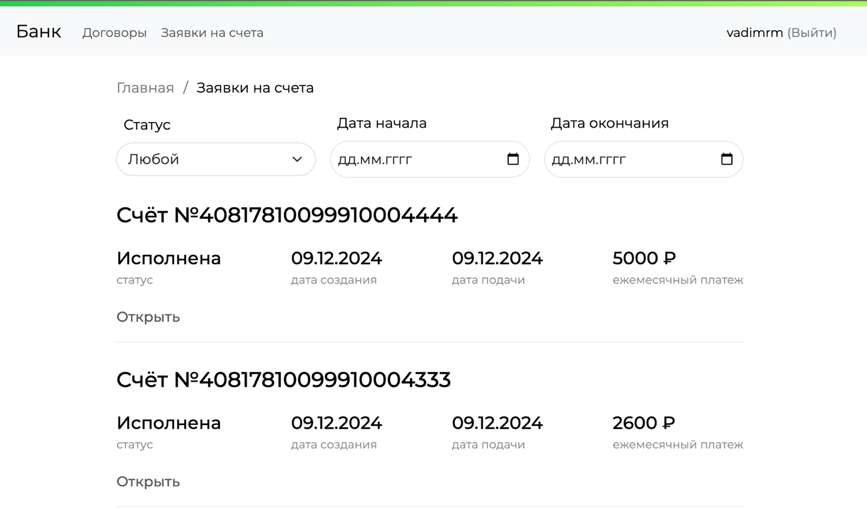


Рисунок 18 – Страница списка заявок на открытие счетов

Пользователь может изменить свой пароль на странице редактирования данных пользователя (рисунок 19). Она открывается при нажатии на логин пользователя в меню.

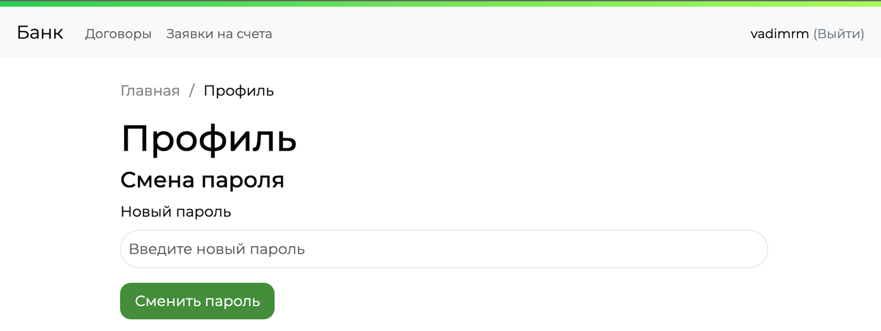


Рисунок 19 – Страница редактирования информации о пользователе

Сотрудник банка может просматривать информацию о всех договорах, находящихся в системе на странице списка договоров (рисунок 20). У сотрудника есть возможность добавить новый договор, отредактировать или удалить существующий.

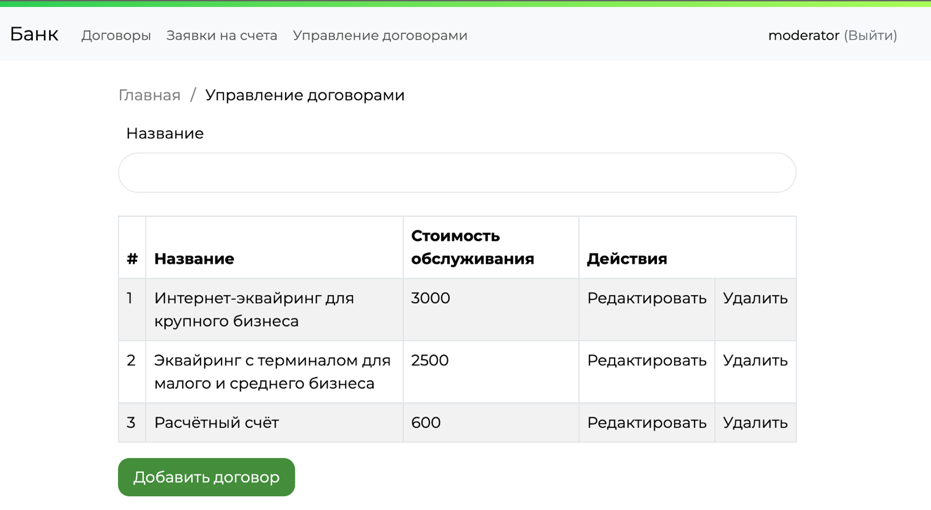


Рисунок 20 – Страница списка договоров

На странице списка всех заявок пользователей (рисунок 21), сотрудник банка может выполнить или отклонить заявки пользователей путём нажатия соответствующих кнопок в интерфейсе. Сотрудник банка может просматривать подробную информацию о заявке с помощью нажатия кнопки «Открыть». Заявки можно фильтровать по логину пользователя, статусу и датам формирования и исполнения.

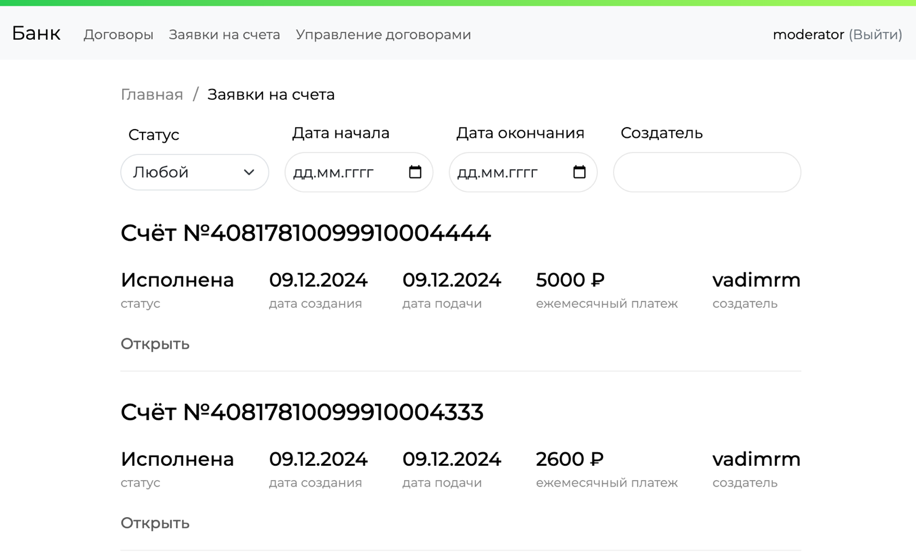


Рисунок 21 – Страница списка заявок на открытие счёта

С помощью страницы редактирования и создания договора (рисунок 22) сотрудник банка может добавлять новые договоры или изменять существующие.

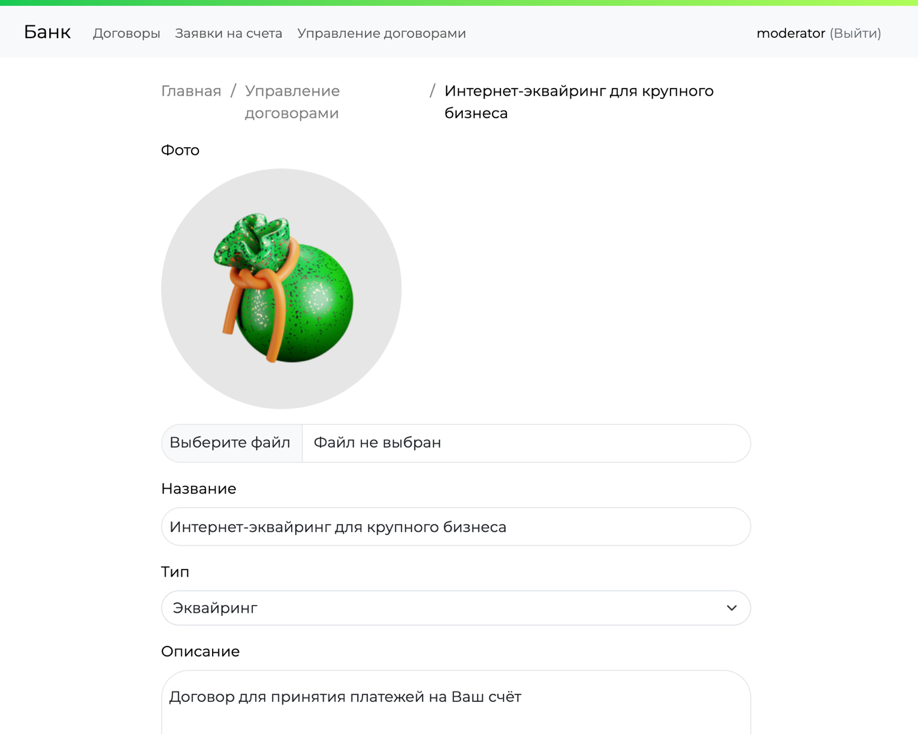


Рисунок 22 – Страница редактирования и создания договора

В случае, если пользователь запрашивает несуществующую станицу, он перенаправляется на страницу ошибки 404 (рисунок 23).

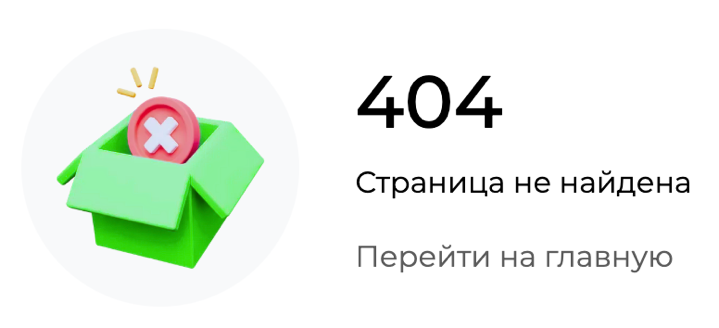


Рисунок 23 – Страница 404

В случае, если пользователь пытается получить доступ к страницам, для просмотра которых ему не хватает прав, он перенаправляется на страницу ошибки 403 (рисунок 24).

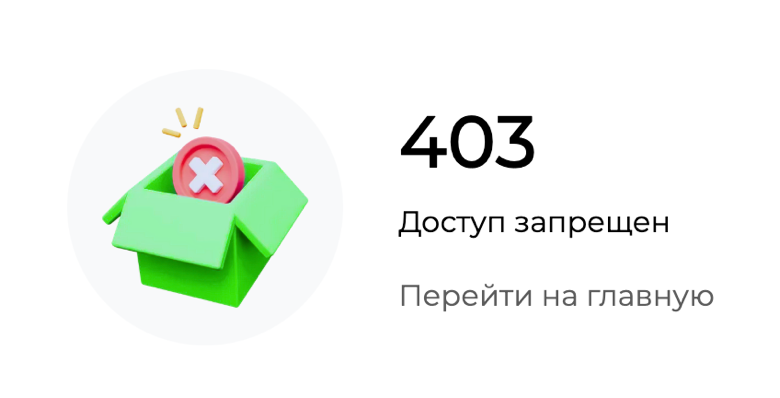


Рисунок 24 – Страница 403

# ЗАКЛЮЧЕНИЕ

В ходе работы были достигнуты следующие результаты:

* 1. Создан MVP и базового дизайн на основе sberbank.ru
  2. Создана база данных для хранения информации о договорах и заявках на открытие счетов
  3. Создан веб-сервис на Golang
  4. Реализована авторизация и хранение отозванных JWT в Redis
  5. Разработан базовый SPA на React для гостя
  6. Внедрена адаптивность, менеджер состояний Redux Toolkit, PWА, разработано Tauri приложение
  7. Завершена разработка интерфейса клиента банка в React
  8. Реализован интерфейс сотрудника банка в React
  9. Реализовано десктопное приложение Tauri
  10. Приложение развернуто при помощи GitHub Pages и доступно по ссылке: https://vadim-rm.github.io/bank-contracts-frontend/
  11. Подготовлен набор документации, включающий РПЗ, ТЗ и набор диаграмм
  12. Оформлен git-репозиторий на сервисе GitHub, содержащий исходный код проекта: https://github.com/vadim-rm/bank-contracts-frontend

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Internet Banking System Prototype [Электронный ресурс] // Arxiv. URL: https://arxiv.org/pdf/1005.4028 (дата обращения: 08.12.2024);
2. Digital financial services and open banking innovation: are banks becoming invisible? [Электронный ресурс] // Arxiv. URL: https://arxiv.org/pdf/2210.01109 (дата обращения: 08.12.2024);
3. Документация по Golang [Электронный ресурс] // Google. URL: https://www.go.dev/ (дата обращения: 20.10.2024);
4. Документация по JWT [Электронный ресурс] // JSON Web Token (JWT). URL: https://datatracker.ietf.org/doc/html/rfc7519 (дата обращения: 20.10.2024);
5. Документация по Redis [Электронный ресурс] // Netlify. URL: https://master--redis-doc.netlify.app/docs/ (дата обращения: 10.10.2024);
6. Документация Minio [Электронный ресурс] // Min. URL: https://min.io/docs/minio/kubernetes/upstream/index.html (дата обращения: 09.09.2024);
7. Документация PostgreSQL [Электронный ресурс] // Postgresql. URL: https://www.postgresql.org/docs/ (дата обращения: 15.09.2024);
8. Документация по Tauri [Электронный ресурс] // Tauri. URL: https://v2.tauri.app/develop/ (дата обращения: 20.11.2024);
9. Документация по React [Электронный ресурс] // React. URL: https://react.dev/learn (дата обращения: 01.11.2024);

# ПРИЛОЖЕНИЕ А ТЕХНИЧЕСКОЕ ЗАДАНИЕ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления»

|  |  |  |
| --- | --- | --- |
| Утверждаю  Заведующий кафедрой ИУ-5 |  | Согласовано  Научный руководитель |
| \_\_\_\_\_\_\_\_\_\_\_В.И.Терехов  "\_\_"\_\_\_\_\_\_\_\_\_\_2024 г. |  | \_\_\_\_\_\_\_\_\_\_А.И. Канев  "\_\_"\_\_\_\_\_\_\_\_\_\_2024 г. |

**Банковские счета**

Техническое задание

(вид документа)

писчая бумага

(вид носителя)

20

(количество листов)

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: |  |
| Мажитов Вадим |  |
| "\_\_"\_\_\_\_\_\_\_\_\_\_\_2024 г. |  |

Москва - 2024

## Цель

Необходимо разработать систему, включающую в себя веб-сервис, веб-приложение и десктопное приложение на Tauri, которая позволит банку отслеживать состояние по заявкам на счета, а клиентам – оставлять их.

## Назначение разработки

Необходимо разработать систему, состоящую из веб-сервиса и веб-приложения, которая будет предоставлять информацию о заявках на открытие счетов. Клиенты смогут отправлять заявки на открытие счета, указывая необходимые данные при заполнении формы. Сотрудник банка может принять заявку клиента на рассмотрение или отклонить её. Клиентам будет доступен список их заявок, включающий активные и обработанные.

## Стадии и этапы разработки

* 1. Создание MVP и базового дизайна на основе sberbank.ru
  2. Создание базы данных для хранения информации о договорах и заявках на открытие счетов (Postgres, gorm)
  3. Создание веб-сервиса на Golang
  4. Реализация авторизацию и хранение отозванных JWT в Redis
  5. Разработка SPA на React для гостя
  6. Внедрение адаптивности, менеджера состояний Redux Toolkit, PWА, разработка Tauri приложения
  7. Завершение разработки интерфейса клиента банка в React
  8. Реализация интерфейса сотрудника банка в React
  9. Разработка десктопного приложения на Tauri
  10. Развертывание приложения при помощи GitHub Pages
  11. Подготовка всей документации (РПЗ, ТЗ и набор диаграмм)
  12. Оформление git-репозитория на сервисе GitHub

## Требования к функциональным характеристикам

* 1. Методы HTTP
     1. POST Регистрировать пользователя
     2. POST Аутентификация
     3. POST Деавторизация
     4. PUT Личный кабинет
     5. GET Получение списка всех договоров
     6. GET Получить один договор
     7. POST Добавление договора
     8. DELETE Удалить договор
     9. PUT Изменить договор
     10. PUT Изменение/добавление картинки договора
     11. POST Добавить договор в заявку на счёт
     12. GET Получить все заявки на счета
     13. GET Получить одну заявку на счёт
     14. PUT Изменить поля заявки на счёт
     15. DELETE Удалить заявку на счёт
     16. PUT Сформировать заявку на счёт
     17. PUT Модерировать заявку на счёт
     18. DELETE Удалить договор из заявки на счёт
     19. POST Указать договор главным в заявке на счёт
  2. Регистрация
     1. Доступно только гостям
     2. Отображает форму регистрации
        1. Поле имени
        2. Поле пароля
        3. Статус модератора
     3. Действия
        1. Регистрация клиента банка – (вызывается метод 4.1.1)
        2. Вернуться к аутентификации – перенаправляет на страницу 4.3
  3. Аутентификация
     1. Доступно только гостям
     2. Отображает форму аутентификации
        1. Поле имени
        2. Поле пароля
     3. Действия
        1. Войти – (вызывается метод 4.1.2)
        2. Регистрация – перенаправляет на страницу 4.2
  4. Личный кабинет
     1. Доступно аутентифицированному клиенту банка
     2. Действия
        1. Изменить данные клиента банка – (вызывается метод 4.1.4)
  5. Главная
     1. Доступна всем
     2. Отображается статическая информация
        1. Назначение сервиса
        2. Контакты для связи
  6. Список договоров
     1. Доступна всем
     2. Отображаются элементы карточек с договорами, вызывается метод 4.1.1
        1. Название
        2. Краткое описание
        3. Ежемесячная абонентская плата
     3. Действия
        1. Поиск по имени – перенаправляет на страницу 4.6, (используется метод 4.1.5), с фильтрующем параметром
        2. Поиск по типу – перенаправляет на страницу 4.6, (используется метод 4.1.5), с фильтрующем параметром
        3. Подробнее – перенаправляет на страницу 4.7 (используется метод 4.1.6)
        4. Оформить – добавляет договор в заявку на счёт, (вызывается метод 4.1.11), только аутентифицированные клиенты банка.
        5. Кнопка корзины – перенаправляет на страницу 4.8, только аутентифицированные клиенты банка.
  7. Один договор
     1. Доступна всем
     2. Отображается подробная информация выбранного договора, (вызывается метод 4.1.6)
  8. Одна заявка на счёт
     1. Доступно только аутентифицированным клиентам банка
     2. Отображает текущую заявку на счёт-черновик пользователя, (метод 4.1.13)
        1. Список выбранных договоров
        2. Поле номера счёта
     3. Действия, доступны только в случае, если статус «черновик»
        1. Убрать договор – удаляет договор из заявки на счёт, (вызывается метод 4.1.18)
        2. Отправить – отправляет текущую заявку на счёт-черновик, (вызывается метод 4.1.16)
        3. Очистить – удаляет заявку на счёт-черновик, (вызывается метод 4.1.15)
        4. Сохранить – вносит данные номера счёта, (вызывается метод 4.1.14)
  9. Список заявок на счета
     1. Доступно аутентифицированному сотруднику банка
     2. Отображается список заявок на счета (метод 4.1.12)
        1. Только заявки на счета, созданные данным клиентом, если он не сотрудник
        2. Все поданные заявки на счета в противном случае
     3. Действия
        1. Фильтрация – фильтрует заявки на счета по дате создания или статусу, вызывается (метод 4.1.12)
        2. Выполнить – отмечает заявку на счёт как исполненную, выполняется метод 4.1.17, доступно только сотруднику банка
        3. Отклонить – отклоняет заявку на счёт, вызывается метод 4.1.17, доступно только сотруднику банка
        4. Посмотреть подробную информацию о заявке на счёт – перенаправляет на страницу 4.8 (вызывается метод 4.1.13)
  10. Список договоров таблицей
      1. Доступно сотруднику банка
      2. Отображаются все существующие в системе договоры (вызывается метод 4.1.5)
      3. Действия
         1. Удалить – удаляет договор (вызывается метод 4.1.8)
         2. Редактирование/создание – переход на страницу 4.11
  11. Редактирование/создание договора
      1. Доступно сотруднику банка
      2. Отображается информация об изменяемом/добавляемом договоре (вызывается метод 4.1.6)
         1. Название
         2. Описание
         3. Ежемесячная плата
         4. Картинка
      3. Действия
         1. Сохранить – добавляет новый договор, вызывается метод, 4.1.2, если изменяется существующий, то вызывается метод 4.1.4
         2. Картинка – добавляет/изменяет картинку договора (вызывается метод 4.1.10)
  12. 404
      1. Доступно всем
      2. Отображается в случае отсутствия ресурса
  13. 403
      1. Доступно всем
      2. Отображается в случае запрета на использование ресурса

## Требования к составу и параметрам технических средств

* 1. Сервер
     1. Процессор Apple M1
     2. Оперативная память 16 Гб
     3. Свободное пространство на диске 10 Гб
  2. Клиент
     1. Процессор Apple M1
     2. Оперативная память 16 Гб
     3. Свободное пространство на диске 10 Гб

## Требования к информационной и программной совместимости

* 1. Серверная часть
     1. ОС Linux (6.4.12 и выше)
     2. Minio (RELEASE.2023-12-14T18-51-57Z)
     3. Redis (7.2)
     4. Golang (1.23)
     5. СУБД PostgreSQL (16)
     6. Доступ к интернету
  2. Клиентская часть
     1. Браузер (Safari 16.5.2, Firefox 121.0, Chrome 119.0.6045, Yandex 24.6.3.729, Opera 105.0.4970.16)

# ПРИЛОЖЕНИЕ Б СПИСОК HTTP МЕТОДОВ

Таблица 1 – HTTP методы разрабатываемого веб-сервиса

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Тип** | **URL** | **Описание** | **Входные данные** | **Выходные данные** |
| 4.1.1 | POST | /users | Регистрация. Доступно всем | {  name: string  email: string  password: string  } | - |
| 4.1.2 | POST | /users/login | Вход в аккаунт. Доступно всем | {  login: string  password: string  } | {  accessToken: string  expiresAt: datetime  isModerator: boolean  login: string  } |
| 4.1.3 | POST | /users/logout | Выход из аккаунта.  Доступно только авторизованным клиентам банка | jwt | - |
| 4.1.4 | PUT | /users | Обновить данные клиента. Доступно только авторизованным клиентам банка | jwt  {  password: string  } | - |
| 4.1.5 | GET | /contracts | Возвращает отфильтрованный список договоров и черновик заявки на счёт текущего пользователя. Доступно всем | contractName: string  contractType: string | {  account: {  count: integer  id: integer  },  contracts: [  {  id: int  name: string  type: string  description: string  imageUrl: string  fee: integer  }  ]  } |
| 4.1.6 | GET | /contracts/{id} | Возвращает информацию о договоре. Доступно всем | id: int | {  id: integer  name: string  type: string  description: string  imageUrl: string  fee: integer  } |
| 4.1.7 | POST | /contracts | Создает договор.  Доступно только авторизованным сотрудникам банка | jwt  {  name: string  type: string  description: string  fee: integer  } | {  id: int  } |
| 4.1.8 | DELETE | /contracts/{id} | Удаляет договор.  Доступно только авторизованным сотрудникам банка | jwt  id | - |
| 4.1.9 | PUT | /contracts/{id} | Обновляет информацию о договоре.  Доступно только авторизованным сотрудникам банка | jwt  id  {  description: string  fee: integer  name: string  type: string  } | {  id: integer  description: string  fee: integer  name: string  type: string  imageUrl: string  } |
| 4.1.10 | PUT | /contracts/{id}/image | Добавляет изображение к договору.  Доступно только авторизованным сотрудникам банка | jwt  id  multipart/form-data:  image | {  id: integer  description: string  fee: integer  name: string  type: string  imageUrl: string  } |
| 4.1.11 | POST | /contracts/{id}/draft | Добавляет договор в заявку на счёт.  Доступно только авторизованным клиентам банка | jwt  id | - |
| 4.1.12 | GET | /accounts | Возвращает отфильтрованный по датам создания и статусу список заявок на счета.  Доступно только авторизованным клиентам и сотрудникам банка | jwt  query params  {  status: string,  from: date  } | {  accounts: [  {  createdAt: string,  creator: integer  finishedAt: string  id: integer  moderator: integer  number: string  requestedAt: string  status: string  totalFee: integer  }  ]  } |
| 4.1.13 | GET | /accounts/{id} | Возвращает подробную информацию об одной заявке на счёт.  Доступно только авторизованным клиентам банка | jwt  id | {  createdAt: string  creator: integer  finishedAt: string  id: integer  moderator: integer  number: string  requestedAt: string  status: string  totalFee: integer  contracts: [  {  description: string  fee: integer  id: integer  imageUrl: string  isMain: boolean  name: string  type: string  }  ]  } |
| 4.1.14 | PUT | /accounts/{id} | Обновляет информацию о заявке на счёт.  Доступно только авторизированным клиентам банка | id  {  number: string,  }, | {  createdAt: string  creator: string  finishedAt: string  id: integer  moderator: string  number: string  requestedAt: string  status: string  totalFee: integer  contracts: [  {  description: string  fee: integer  id: integer  imageUrl: string  isMain: boolean  name: string  type: string  }  ]  } |
| 4.1.15 | DELETE | /accounts/{id} | Удаляет заявку на счёт.  Доступно только авторизированным клиентам банка | jwt  id | - |
| 4.1.16 | PUT | /accounts/{id}/submit | Формирует заявку на счёт.  Доступно только авторизованным клиентам банка | jwt  id | {  createdAt: string  creator: string  finishedAt: string  id: integer  moderator: string  number: string  requestedAt: string  status: string  totalFee: integer  contracts: [  {  description: string  fee: integer  id: integer  imageUrl: string  isMain: boolean  name: string  type: string  }  ]  } |
| 4.1.17 | PUT | /accounts/{id}/complete | Завершает заявку на счёт.  Доступно только авторизованным сотрудникам банка | {  status: string  } | - |
| 4.1.18 | DELETE | /accounts/{accountId}/contract/{contractId} | Удаляет договор из заявки на счёт.  Доступно только авторизованным клиентам банка | jwt  accountId  contractId | - |
| 4.1.19 | POST | /accounts/{accountId}/contract/{contractId}/main | Указывает договор главным в заявке на счёт.  Доступно только авторизованным клиентам банка | jwt  accountId  contractId | - |