МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГУ»)

Факультет компьютерных наук

Кафедра информационных технологий управления

*Инновационное интегрирование комплексов сервисов 1С на технологическую платформу по обработке данных в кредитных организациях*

Бакалаврская работа

09.03.02 Информационные системы и технологии

Профиль «Информационные системы и технологии в управлении предприятием»

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ д. т. н., профессор Матвеев М. Г

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. 4 курса оч. отд. Лямкин Е. С.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. преподаватель Бебнева С. В.

Воронеж2025

Содержание

[Введение 3](#_Toc196326601)

[1 Постановка задачи 4](#_Toc196326602)

[1.1 Актуальность темы 4](#_Toc196326603)

[1.2 Определение ключевых целей работы 5](#_Toc196326604)

[1.3 Методологическая база и инструменты 6](#_Toc196326605)

[2 Анализ предметной области и постановка задачи 8](#_Toc196326606)

[2.1 Современные подходы к интеграции сервисов 1С 8](#_Toc196326607)

[2.2 Особенности документооборота и анализа финансовых данных в кредитовании 9](#_Toc196326608)

[2.3 Технология продаж банковских продуктов клиентам 11](#_Toc196326609)

[2.4 Потоки данных между компаниями и банками 14](#_Toc196326610)

[3 Теоретические и технологические основы разработки веб-сервисов 15](#_Toc196326611)

[3.1 Обзор технологий интеграции 1С (HTTP API, JSON, 1С-обработки) 15](#_Toc196326612)

[3.2 Технологии веб-разработки: React, Express, PostgreSQL 15](#_Toc196326613)

[3.3 Архитектурные паттерны для проектирования распределённых систем 16](#_Toc196326614)

[3.4 Обоснование выбора инструментов и подходов 17](#_Toc196326615)

[4 Проектирование и реализация программного решения 18](#_Toc196326616)

[4.1 Архитектура веб-сервиса 18](#_Toc196326617)

[4.2 Разработка REST API на Express.js 19](#_Toc196326618)

[4.3 Структура базы данных PostgreSQL 24](#_Toc196326619)

[4.4 Пользовательский интерфейс на React 27](#_Toc196326620)

[4.5 Организация обмена данными с 1С через HTTP 31](#_Toc196326621)

[4.6 Роли пользователей: компания, банк 33](#_Toc196326622)

[4.7 Безопасность и авторизация пользователей 36](#_Toc196326623)

[5 Анализ результатов 39](#_Toc196326624)

[5.1 Анализ корректности и устойчивости работы системы 39](#_Toc196326625)

[5.2 Выводы по результатам тестирования 40](#_Toc196326626)

[Заключение 41](#_Toc196326627)

[Список использованных источников 43](#_Toc196326628)

Введение

Современный бизнес предъявляет высокие требования к скорости, точности и безопасности обработки финансовой информации. Особенно актуальной эта задача становится в банковской сфере, где принятие решений о предоставлении финансирования малому и среднему бизнесу во многом зависит от анализа актуальных и достоверных данных. Одним из источников таких данных выступают учетные системы на базе 1С, повсеместно используемые компаниями для ведения бухгалтерии и финансовой отчетности.

Традиционные методы взаимодействия между компаниями и банками зачастую требуют ручной подготовки и передачи документов, что замедляет процесс рассмотрения кредитных заявок, создаёт потенциал для неудобства обеих сторон и усложняет процесс получения финансирования на конкретных условиях необходимых клиенту. В условиях стремительно развивающейся экономики возникает необходимость в создании универсальных цифровых решений, способных автоматизировать этот процесс. Инновационное интегрирование сервисов 1С с технологической платформой по обработке данных позволяет упростить передачу финансовой отчетности в банк, сократить время на принятие решений и повысить прозрачность взаимодействия между сторонами.

Целью данной выпускной квалификационной работы является разработка программного решения — веб-сервиса, обеспечивающего интеграцию учетных систем 1С организаций с технологической платформой банка. Решение должно позволять организациям отправлять заявки на финансирование, а банкам — получать необходимые финансовые сведения в удобном для анализа виде. Особенность предлагаемого подхода заключается в использовании современных технологий веб-разработки (React, Express.js, PostgreSQL), а также стандартизированного взаимодействия с 1С посредством JSON over HTTP.

1. Постановка задачи
   1. Актуальность темы

Цифровизация финансовых процессов оказывает значительное влияние на рынок банковских услуг, в частности на процессы кредитования малого и среднего бизнеса (МСБ). Одним из приоритетных направлений становится повышение удобства взаимодействия между клиентом и финансовой организацией, особенно на этапе подачи заявки и анализа финансовых данных. Стандартный путь подачи заявки через сайт конкретного банка, как правило, предполагает строгую привязку к условиям данной кредитной организации и требует от клиента индивидуальной подготовки пакета документов, что связано с существенными временными и организационными затратами.

В рамках современного подхода к удобству продаж банковских продуктов важно учитывать не только автоматизацию документооборота, но и создание платформенных решений, позволяющих упростить и ускорить процессы с обеих сторон. Разработка цифровых сервисов, обеспечивающих одновременное взаимодействие с несколькими банками, открывает возможности для расширения охвата потенциальных клиентов за счёт отсутствия привязки к конкретному банку, применения «олимпийской системы» оценки заявок, при которой несколько кредитных организаций независимо принимают решение по одному и тому же проекту, снижения трудозатрат для клиента, который в противном случае вынужден адаптировать документы под требования каждого банка отдельно, а так же повышения прозрачности и открытости рынка финансирования — клиент видит предложения сразу от нескольких банков и может выбрать наиболее выгодное.

С другой стороны, данная модель выгодна и для банков, поскольку обеспечивает более удобный и структурированный входящий поток заявок, где уже присутствуют готовые структурированные данные, автоматически полученные из систем 1С. Это упрощает работу менеджеров, ускоряет первичную обработку заявок и позволяет выстраивать эффективную систему продаж кредитных продуктов.

Таким образом, разрабатываемое в рамках данной выпускной квалификационной работы веб-приложение направлено не только на техническую интеграцию с учетными системами клиентов (1С), но и на оптимизацию бизнес-процессов как со стороны компаний, так и со стороны банков. Такой подход позволяет построить альтернативный цифровой канал подачи заявок, обладающий высокой степенью универсальности, удобства и гибкости. Это делает тему работы особенно актуальной в современных условиях.

* 1. Определение ключевых целей работы

Основной целью данной выпускной квалификационной работы является разработка веб-сервиса, обеспечивающего интеграцию между информационными системами компаний-клиентов (на базе 1С) и технологической платформой банков, с возможностью автоматизированной подачи заявок на финансирование. Предлагаемое решение направлено не только на техническую интеграцию, но и на оптимизацию процессов взаимодействия между сторонами — компаниями и банками — с фокусом на повышение прозрачности, удобства и эффективности.

Ключевыми задачами, необходимыми для достижения цели, являются:

* анализ существующих подходов к интеграции платформы 1С с внешними сервисами, в том числе банками и финансовыми платформами;
* построение архитектуры веб-приложения, способного обеспечить авторизованный доступ клиентов и банковских сотрудников, разграничение ролей и сценариев использования;
* реализация механизма «единых заявок», позволяющего клиенту без привязки к конкретному банку подать структурированную заявку, которая может быть одновременно рассмотрена несколькими кредитными организациями;
* обеспечение автоматизированного получения финансовых данных из 1С, в том числе формирования пакета документов, соответствующего требованиям банков;
* разработка REST API и пользовательского интерфейса, ориентированного на высокую скорость работы и простоту подачи заявки;
* реализация логики обработки заявки с разных сторон: клиент может отслеживать её статус, а сотрудники банка — получать уведомления, просматривать и оценивать предложения;
* обеспечение надежности, отказоустойчивости и безопасности хранения персональных и финансовых данных, в том числе авторизации по ролям и защите передаваемой информации.

Дополнительно в рамках проекта предусматривается создание альтернативной модели кредитной оценки, при которой банки «конкурируют» за заявки клиента, формируя предложения на основе единых входных данных. Такой подход снижает барьеры для клиента, повышает гибкость и прозрачность рынка кредитования, а также способствует улучшению качества обслуживания в финансовом секторе.

* 1. Методологическая база и инструменты

В качестве методологической основы используются принципы построения клиент-серверной архитектуры, REST-подход к проектированию API, а также практика модульной и компонентной разработки программных решений.

В техническом плане для реализации программного продукта применяются следующие инструменты и технологии:

* **React** для разработки клиентской части веб-приложения;
* **Express.js (Node.js)** для построения серверной части и REST API;
* **PostgreSQL** в качестве системы управления базами данных;
* **1С:Предприятие** как источник финансовых данных со стороны компаний;
* **HTTP и JSON** как протокол и формат обмена данными;
* **UML-диаграммы** для описания архитектуры и взаимодействия компонентов;
* **Postman** и ручное тестирование для проверки корректности API и интерфейсов.

Применение указанных технологий обусловлено их широким распространением, поддержкой со стороны профессионального сообщества и возможностью быстрой интеграции в существующие ИТ-инфраструктуры банков и компаний.

1. Анализ предметной области и постановка задачи
   1. Современные подходы к интеграции сервисов 1С

Интеграция программных решений с платформой 1С:Предприятие представляет собой одну из важнейших задач при построении корпоративных и межкорпоративных информационных систем. Особенно актуальной эта задача становится в случае, когда требуется организовать автоматизированный обмен данными между компаниями и внешними организациями, в том числе кредитными учреждениями.

Платформа 1С предлагает широкий спектр технических решений для интеграции с внешними системами. К числу наиболее распространённых подходов относятся:

* **HTTP-сервисы (REST API)** — один из наиболее удобных и универсальных способов обмена данными между 1С и сторонними приложениями. На стороне 1С может быть настроен HTTP-сервер, способный принимать и обрабатывать запросы в формате JSON или XML, а также отправлять структурированные ответы;
* **Внешние обработки и обмен по расписанию** — используются в случаях, когда взаимодействие с внешними системами необходимо выполнять пакетно или асинхронно. Обработка может экспортировать или импортировать данные в стандартных форматах (CSV, XML, JSON) и запускаться вручную или по таймеру;
* **Обмен файлами через FTP, сетевые папки, email** — подход, распространённый в старых системах, но всё чаще заменяемый API-интеграцией;
* **Интеграция через брокеры сообщений (RabbitMQ, Kafka)** — используется в более сложных и распределённых системах, где необходимо обрабатывать события и данные в реальном времени или с высокой степенью масштабируемости.

Современные подходы также предполагают переход от монолитных сценариев обмена к сервисно-ориентированной архитектуре (SOA) и использованию микросервисов, что особенно важно при интеграции платформ вроде 1С с веб-сервисами или платформами финансовых организаций.

Особый интерес представляет обратная интеграция, при которой инициатором взаимодействия выступает сторонняя система (например, веб-платформа подачи заявок), запрашивая у 1С клиента определённый набор информации — оборотно-сальдовую ведомость, налоговые показатели, сведения о контрагентах и т.д. Полученные данные используются для формирования автоматических заявок на кредитование.

В контексте разработки мультибанковского сервиса, описанного в данной работе, ключевую роль играет унификация и стандартизация формата передаваемых данных, а также минимизация трудозатрат со стороны клиента. Реализация API-интерфейсов, работающих с конфигурациями 1С через HTTP-запросы и JSON-структуры, позволяет централизованно собирать финансовые данные и оперативно передавать их в банки, обеспечивая единый цифровой канал подачи заявок на кредитование.

Таким образом, современные технологии интеграции 1С не только обеспечивают техническую связность между системами, но и становятся инструментом упрощения бизнес-процессов как для клиентов, так и для банков, формируя основу для цифровой трансформации финансовых услуг. Однако с появление инновационных технологий механизмы и инструменты передачи и обработки информации требуют совершенно иного подхода и дополнительных разработок.

* 1. Особенности документооборота и анализа финансовых данных в кредитовании

Для получения предварительного решения о финансировании компании необходимо предоставить банку набор документов подтверждающих финансовую стабильность организации. В этот набор входят такие документы как:

* Бухгалтерский баланс на пять последних отчетных дат;
* Отчет о прибылях и убытках на пять последних отчетных дат;
* Оборотно-сальдовые ведомости по счетам 66 "Расчеты по краткосрочным кредитам и займам" и 67 "Расчеты по долгосрочным кредитам и займам" на последнюю отчетную дату;
* Оборотно-сальдовая ведомость 01 счета "Основные средства".

Для автоматизации передачи части этих документов уже существуют встроенные в типовые 1с конфигурации решения, однако остаток пакета документов должен передаваться отдельно по электронной почте или доставляться в банк непосредственно сотрудником компании. Данный подход создаёт спектр неудобств как для клиента, так и для банка. Процесс получения финансирования затягивается и обретает дополнительные необязательные ступени, приводящие к потере времени и, соответственно, денег.

Как правило, подготовка и передача остатка пакета документов осуществляется вручную, через загрузку файлов (PDF, Excel, сканы) на портал банка или передачу по email. Такой подход имеет ряд существенных недостатков:

* отсутствие стандартизации форматов;
* невозможность машинной обработки вложений без предварительного ручного анализа;
* повышенные требования к квалификации клиента при самостоятельной подготовке документов;
* повторение одних и тех же действий при подаче заявок в разные банки.

Для компаний малого и среднего бизнеса такие процессы представляют собой избыточную нагрузку, особенно если они не располагают собственным бухгалтером или специалистом по финансам.

Одновременно с этим, банки, получая документы в произвольной форме, сталкиваются с дополнительной нагрузкой на специалистов, занятых ручным вводом данных в кредитную систему. Это увеличивает время рассмотрения заявки, снижает эффективность и увеличивает вероятность ошибок или упущений.

* 1. Технология продаж банковских продуктов клиентам

Технология продажи продуктов, используемая в современной банковской системе, подразумевает непосредственное взаимодействие специалистов из отдела продаж с клиентами банка. На рисунке 1 представлена типичная схема взаимодействия банка с клиентами в виде диаграммы процесса.



Рисунок 1 - Диаграмма процесса продаж банковских продуктов клиентам

Продажи начинаются с доведения квартального плана продаж до сотрудников. Начальник подразделения формирует индивидуальные планы, после чего специалист отдела продаж приступает к созданию клиентской базы в CRM. Источниками являются как действующие клиенты банка, так и новые потенциальные компании, отобранные на основании соответствия целевой аудитории и банковских продуктов.

Важным моментом является регулярный пересмотр и обновление базы, включая проверку актуальности данных, выявление новых контактов, анализ клиентов, ранее отказавшихся от сотрудничества.

После подготовки базы специалист:

* осуществляет поиск дополнительной информации о клиентах (сфера деятельности, обслуживающий банк, конкурентные преимущества);
* планирует структуру беседы;
* инициирует контакт (чаще всего — телефонный звонок).

Если дозвон не состоялся, планируется повторный контакт. В случае успешного соединения специалист выясняет заинтересованность клиента в продукте:

* при отказе — предлагает альтернативные продукты;
* при частичной заинтересованности — направляет коммерческое предложение (КП) и планирует дальнейшие действия.

Если клиент готов к обсуждению условий, то проводится либо личная встреча, либо переговоры в дистанционном формате.

Во время переговоров формируются особенности индивидуального решения и принимается решение. Либо клиент соглашается с условиями и взаимодействие продолжается уже с профильным подразделением банка для заключения договора, либо не соглашается и получает предложение альтернативных продуктов.

Анализ показал ряд **недостатков существующей схемы взаимодействия с клиентами:**

* низкий уровень адаптации сценариев общения: специалисты используют **стандартизированные заученные фразы**, не адаптированные под конкретного клиента;
* часто контакт устанавливается с лицами, не принимающими решений;
* применение **“холодных звонков” с незнакомых номеров** вызывает негатив у клиентов;
* большое количество повторных звонков формирует ощущение давления и навязчивости;
* отсутствует персонализированный подход и стратегическое планирование продаж.
  1. Потоки данных между компаниями и банками

Для улучшения существующей системы передачи документов возможно использовать веб-сервис, позволяющий передать банку единый пакет документов, содержащий в себе достаточный объём данных для принятия банком предварительного решения об финансировании. Типовой процесс подачи заявки на финансирование включает следующие шаги:

* клиент (компания) инициирует запрос на получение кредита через веб-сервис;
* сервис позволяет сформировать пакет данных, включающий как необходимые документы, так и финансовые показатели, извлечённые из 1С;
* банковский сотрудник получает доступ к структурированной информации и проводит предварительный анализ;
* если параметры заявки соответствуют требованиям банка, данные передаются во внутреннюю банковскую систему для принятия решения;
* клиент получает уведомление о результате рассмотрения и возможные дальнейшие шаги.

В данной модели веб-сервис выступает в роли **посредника**, консолидирующего данные из клиентских систем и предоставляющего их в стандартизированном виде банкам. Это позволяет упростить процесс подачи заявок, устранить дублирование операций и минимизировать влияние человеческого фактора.

1. ****Теоретические и технологические основы разработки веб-сервисов****
   1. Обзор технологий интеграции 1С (HTTP API, JSON, 1С-обработки)

Интеграция платформы 1С с внешними системами осуществляется с использованием различных технологий:

* **REST API**: 1С предоставляет возможность взаимодействия через REST-интерфейс, позволяющий выполнять операции с данными с помощью HTTP-запросов и получать ответы в форматах JSON или XML. Это обеспечивает стандартизированный способ обмена данными между 1С и другими приложениями;
* **JSON и HTTP-запросы**: Использование JSON в сочетании с HTTP-запросами является популярным методом интеграции, позволяющим отправлять и получать данные через API, взаимодействовать с веб-сервисами и интегрировать 1С с другими приложениями;
* **1С-обработки**: Для настройки интеграции могут использоваться специализированные обработки, реализующие обмен данными с внешними системами. Эти обработки позволяют настраивать формат данных, методы передачи и обработки ошибок, обеспечивая гибкость и адаптивность интеграционных решений.
  1. Технологии веб-разработки: React, Express.js, PostgreSQL

Для организации полноценной интеграции комплексов сервисов 1С на технологическую платформу по обработке данных в кредитных организациях и для разработки современного веб-сервиса, обеспечивающего автоматизацию передачи и анализа документов целесообразно использовать следующие технологии:

* **React**: JavaScript-библиотека для создания пользовательских интерфейсов, позволяющая разрабатывать динамичные и отзывчивые веб-приложения. React обеспечивает компонентный подход к разработке, что упрощает поддержку и масштабирование кода;
* **Express.js**: Минималистичный и гибкий веб-фреймворк для Node.js, предназначенный для создания серверных приложений и API. Express.js предоставляет набор инструментов для обработки HTTP-запросов, маршрутизации и управления промежуточными обработчиками;
* **PostgreSQL**: Объектно-реляционная система управления базами данных с открытым исходным кодом, известная своей надежностью, расширяемостью и соответствием стандартам SQL. PostgreSQL поддерживает широкий спектр типов данных и предоставляет мощные инструменты для работы с транзакциями и запросами.

Комбинация этих технологий позволит создать полнофункциональное веб-приложение с разделением клиентской и серверной логики, обеспечивающее высокую производительность и удобство в разработке и сопровождении.

* 1. Архитектурные паттерны для проектирования распределённых систем

При разработке распределённых систем применяются различные архитектурные паттерны, обеспечивающие масштабируемость, отказоустойчивость и управляемость:

* **Микросервисная архитектура**: Разделение приложения на независимые сервисы, каждый из которых выполняет определённую бизнес-функцию. Это упрощает масштабирование и обновление отдельных компонентов системы;
* **Паттерн Circuit Breaker**: Механизм, предотвращающий повторные попытки обращения к сбойным сервисам, что помогает избежать перегрузки системы и ускоряет восстановление после сбоев;
* **Паттерн Saga**: Управление распределёнными транзакциями через последовательность локальных транзакций с компенсационными действиями, обеспечивая согласованность данных без использования глобальных блокировок.

Использование этих паттернов способствует созданию надёжных и устойчивых распределённых систем, способных эффективно обрабатывать большие объёмы данных и адаптироваться к изменяющимся условиям эксплуатации.

* 1. Обоснование выбора инструментов и подходов

Выбор технологий и архитектурных решений обусловлен следующими факторами:

* **Совместимость с существующими системами:** Использование REST API и JSON обеспечивает лёгкую интеграцию с платформой 1С и другими внешними системами, упрощая обмен данными и снижая затраты на разработку;
* **Масштабируемость и гибкость:** Микросервисная архитектура и использование современных веб-технологий позволяют легко масштабировать систему и адаптировать её под изменяющиеся бизнес-требования;
* **Надёжность и отказоустойчивость:** Применение паттернов Circuit Breaker и Saga обеспечивает устойчивость системы к сбоям и поддержание согласованности данных в распределённой среде.

Таким образом, выбранные инструменты и подходы соответствуют современным требованиям к разработке распределённых веб-сервисов, обеспечивая эффективность, надёжность и удобство в эксплуатации.

1. ****Проектирование и реализация программного решения****
   1. Архитектура веб-сервиса

Архитектура разрабатываемого веб-сервиса базируется на модели клиент–сервер, в которой реализована трёхуровневая логика: пользовательский интерфейс (фронтенд), серверная часть (бекенд) и слой хранения данных (база данных). Интеграция с внешними системами, в том числе конфигурациями 1С и банковскими платформами, осуществляется через HTTP API.

Для представления архитектуры взаимодействия участников системы была построена диаграмма вариантов использования, приведённая на рисунке 2.

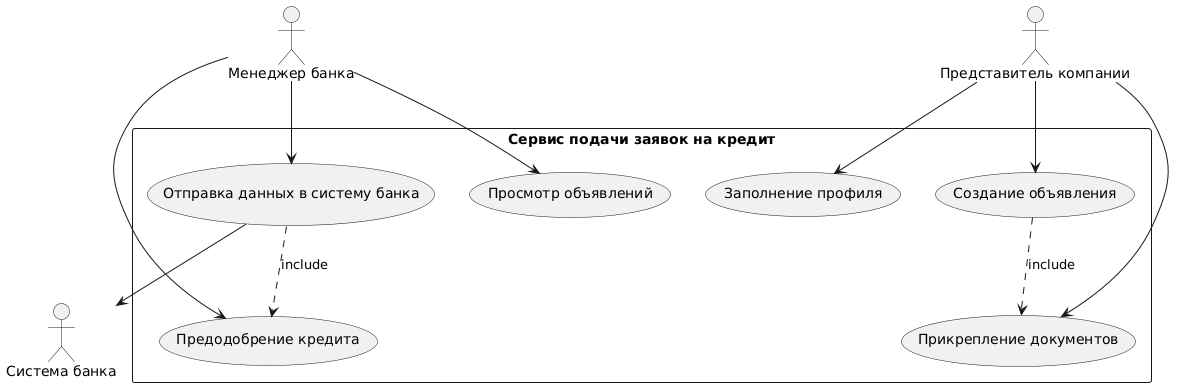


Рисунок 2 - Диаграмма вариантов использования системы подачи заявок на кредит

На диаграмме изображены способы взаимодействия с сервисом для набора пользователей.

Участники представленной системы:

* **Представитель компании** — пользователь, создающий и публикующий заявку на получение кредита. Может заполнять профиль организации и прикреплять необходимые документы;
* **Менеджер банка** — пользователь, просматривающий заявки, отправляющий их в банковскую систему и осуществляющий первичное предодобрение;
* **Система банка** — внешний актор, принимающий данные для последующей обработки заявки на стороне банка.

Основные функции:

* **Создание объявления** — ключевая функция представителя компании, в рамках которой создаётся заявка на получение кредита;
* **Прикрепление документов** — дополнительное действие, сопровождающее создание объявления и предоставляющее банку доступ к финансовым и правовым документам;
* **Заполнение профиля** — начальный этап, в котором пользователь указывает информацию о своей компании;
* **Просмотр объявлений** — доступная менеджеру банка функция для оценки поступивших заявок;
* **Отправка данных в систему банка** — инициируется менеджером банка для начала внутреннего банковского процесса рассмотрения заявки;
* **Предодобрение кредита** — вспомогательная операция, отражающая предварительное решение, основанное на данных, переданных из системы.
  1. Разработка REST API на Express.js

Проектирование архитектуры веб-приложения является ключевым этапом разработки сервиса подачи заявок на кредит, поскольку от него зависит масштабируемость, устойчивость и удобство сопровождения системы. На основании анализа предметной области и функциональных требований была выбрана компонентная архитектура, обеспечивающая модульность и разделение ответственности между различными частями системы.

Архитектура системы разделена на три основные области:

* **Клиентская часть** — отвечает за взаимодействие с пользователем;
* **Серверная часть** — реализует бизнес-логику и обработку данных;
* **Интеграция с 1С** — обеспечивает обмен информацией с внешними системами клиента и банка.

Клиентская часть реализована с помощью библиотеки **React**. В ней пользователь (представитель компании или менеджер банка) может выполнять действия, соответствующие своему сценарию использования: заполнять и редактировать профиль, создавать объявления, прикреплять документы, просматривать доступные предложения и отправлять заявки.

Взаимодействие с серверной частью осуществляется через **HTTP API**, реализованное с использованием **Express.js**.

Серверная часть условно разделена на несколько компонентов:

* **Express.js API** — служит точкой входа для клиентских запросов, занимается маршрутизацией и базовой валидацией данных.
* **PostgreSQL БД** — обеспечивает хранение всех сущностей приложения, включая профили пользователей, объявления, документы, статусы заявок и историю взаимодействия.
* **HTTP-модуль для 1С** — реализует интерфейс взаимодействия с внешними системами 1С, как со стороны банков, так и со стороны компаний.

Вся серверная часть построена по принципу слабо связанных компонентов, что позволяет при необходимости заменять или масштабировать отдельные части без влияния на всю систему.

Особенностью проекта является необходимость двусторонней интеграции с системами 1С:

* **1С Клиента** — отправляет на сервис необходимый набор данных;
* **1С Банка** — принимает структурированные данные о компаниях, документах и заявках для последующего анализа и принятия решения по кредитованию.

На рисунке 3 представлена компонентная диаграмма архитектуры приложения, иллюстрирующая связи между основными подсистемами и направления передачи данных.

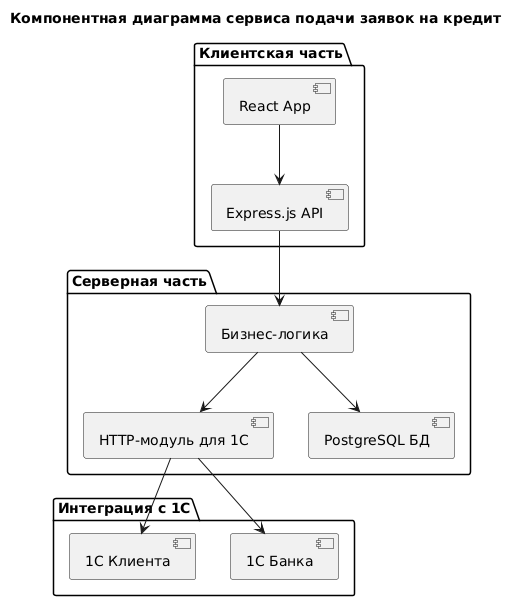


Рисунок 3 - Component диаграмма

Клиентская часть состоит из компоненты React App, которая реализует пользовательский интерфейс, обеспечивая доступ к основным функциям системы. Это одностраничное приложение (SPA), функционирующее на стороне клиента и взаимодействующее с сервером через REST API.

Ключевые задачи компонента:

* обработка пользовательского ввода (формы авторизации, регистрации, подачи заявки);
* отображение состояния данных (заявки, профили, решения банка);
* навигация между представлениями с использованием React Router;
* хранение авторизационной информации (например, токенов) в браузере.

Компонент React App отправляет HTTP-запросы к серверному API, ожидая структурированные ответы в формате JSON.

В серверной части центральным звеном архитектуры является Express.js API, реализованный с использованием платформы Node.js. Данный компонент отвечает за:

* приём запросов от клиентского приложения;
* маршрутизацию запросов по REST-подходу;
* авторизацию пользователей;
* сериализацию и валидацию данных;
* делегирование бизнес-логики.

Вложенными компонентами серверной части являются:

**Бизнес-логика (Application Logic)** является сердцем серверного приложения. Обрабатывает полученные от клиента данные, применяет правила и условия (например, валидацию профиля, проверку доступности создания заявки и т.д.), формирует ответ. В будущем может быть вынесена в отдельные модули (use case services), что особенно актуально при масштабировании.

**PostgreSQL БД** обеспечивает хранение структурированных данных системы: пользователей, компаний, заявок, документов, решений банков. Взаимодействие с базой осуществляется через ORM или вручную реализованные SQL-запросы. Особое внимание уделяется целостности данных, транзакционности и производительности.

**HTTP-модуль для 1С** является интерфейсом интеграции с внешними конфигурациями 1С. Реализован в виде специализированного клиента, выполняющего HTTP-запросы к 1С с передачей данных в формате JSON. Также отвечает за парсинг ответов от 1С и их последующую обработку в бизнес-логике. Модуль может быть расширен для поддержки разных версий API или специфических реализаций на стороне клиента.

Интеграция с внешними системами 1С обеспечивается черездва внешних компонента:

* 1С Клиента — источник исходной финансовой информации (например, оборотно-сальдовых ведомостей);
* 1С Банка — получатель структурированных данных, необходимых для анализа и принятия решений по заявкам.

Обмен данными с системами 1С осуществляется через протокол HTTPS с использованием JSON-формата. В будущем возможна реализация промежуточного буфера (message broker), если потребуется асинхронная обработка большого объёма запросов.

ВАэно отметить на компонентной диаграмме чётко прослеживаются следующие цепочки взаимодействий:

* React App → Express API

HTTP-запросы по защищённому протоколу для выполнения CRUD-операций, авторизации, загрузки данных;

* Express API → Бизнес-логика

Передача данных для интерпретации и обработки. Здесь происходит логическое принятие решений, валидация и маршрутизация запросов к базе данных или внешним API;

* Бизнес-логика → PostgreSQL БД

Взаимодействие с хранилищем данных: чтение, запись, обновление и удаление сущностей;

* Бизнес-логика → HTTP-модуль 1С → 1С Клиента / 1С Банка

Отправка структурированной информации на внешние платформы, приём ответов, контроль корректности обмена.

* 1. Структура базы данных PostgreSQL

Для реализации функционала сервиса подачи заявок на кредит была разработана логическая модель данных, отражающая ключевые сущности и связи между ними. Построение модели осуществлялось на основе требований к системе, определённых в аналитическом и проектном этапах, а также с опорой на диаграммы вариантов использования.

Логическая структура базы данных построена с использованием реляционного подхода и ориентирована на СУБД PostgreSQL. На рисунке 4 представлена ER-диаграмма, иллюстрирующая основные сущности и связи между ними.

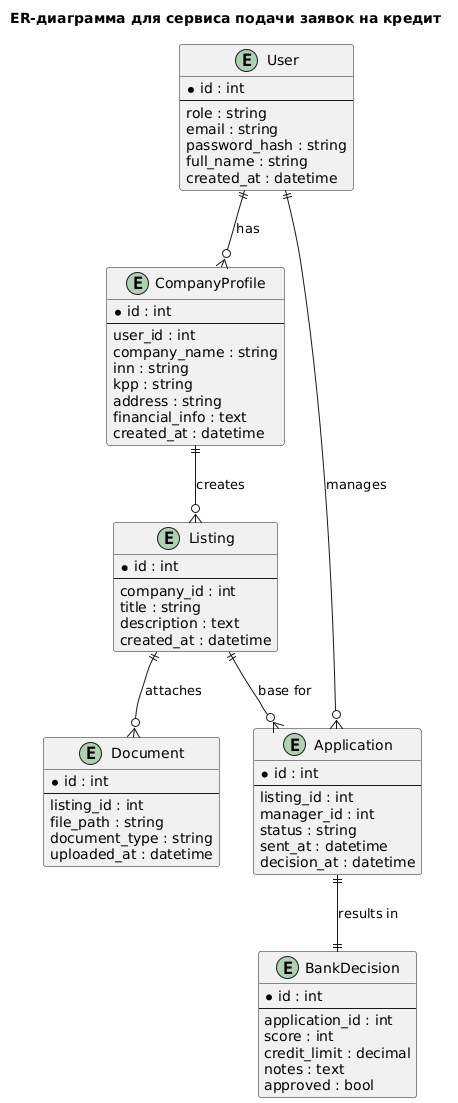


Рисунок 4 - ER-диаграмма

**Основные сущности:**

* **User** — сущность, представляющая зарегистрированных пользователей системы. Каждый пользователь имеет роль (менеджер банка или представитель компании), уникальные идентификаторы, а также базовую информацию для аутентификации и идентификации. С пользователями связаны как объявления, так и заявки на кредит.
* **CompanyProfile** — профиль компании, связанный с пользователем-экспонентом. Хранит информацию о юридическом лице, такую как ИНН, КПП, адрес и прочие данные, необходимые для подачи заявки в банк.
* **Listing** — объявление, созданное представителем компании. Включает в себя описание потребности в кредитовании и может содержать прикреплённые документы.
* **Document** — документы, приложенные к конкретному объявлению. Это могут быть финансовые отчёты, выписки и прочая документация, необходимая для рассмотрения заявки.
* **Application** — заявка, формируемая на основе объявления и отправляемая в банковскую систему. Менеджер банка рассматривает такую заявку, при необходимости возвращает её на доработку или направляет на скоринг.
* **BankDecision** — результат рассмотрения заявки банком. Содержит скоринговую оценку, возможный кредитный лимит и комментарии. Связана с конкретной заявкой.

**Связи между сущностями:**

* Один пользователь может иметь один или несколько связанных профилей компании.
* Каждый профиль компании может содержать несколько объявлений.
* Каждое объявление может иметь прикреплённые документы.
* Объявления служат основанием для создания заявок.
* Заявки создаются менеджерами банка и далее сопровождаются решением со стороны банковской системы.

Такая структура обеспечивает гибкость и масштабируемость решения, позволяя эффективно хранить и обрабатывать информацию, необходимую как для компаний-заявителей, так и для сотрудников банка. Предусмотренные связи между сущностями позволяют организовать целостный жизненный цикл кредитной заявки от момента подачи до принятия решения.

Визуализация модели данных в виде ER-диаграммы представлена на рисунке 3.

* 1. Пользовательский интерфейс на React

Пользовательский интерфейс веб-сервиса реализован с использованием библиотеки **React**, которая позволяет строить масштабируемые одностраничные приложения (SPA). Архитектура фронтенда организована на основе компонентного подхода с применением маршрутизации через **React Router**. Интерфейс разделён на два основных пользовательских сценария: **для представителей компаний** и **для сотрудников банков**.

**Общая структура и маршруты**

Веб-приложение начинается с главной страницы, откуда пользователь может перейти к авторизации или регистрации. Основная логика переходов реализована с использованием защищённых маршрутов, которые разграничивают доступ к личным кабинетам в зависимости от роли пользователя. Логика переходов визуализирована на рисунке 5 средствами сервиса Miro.

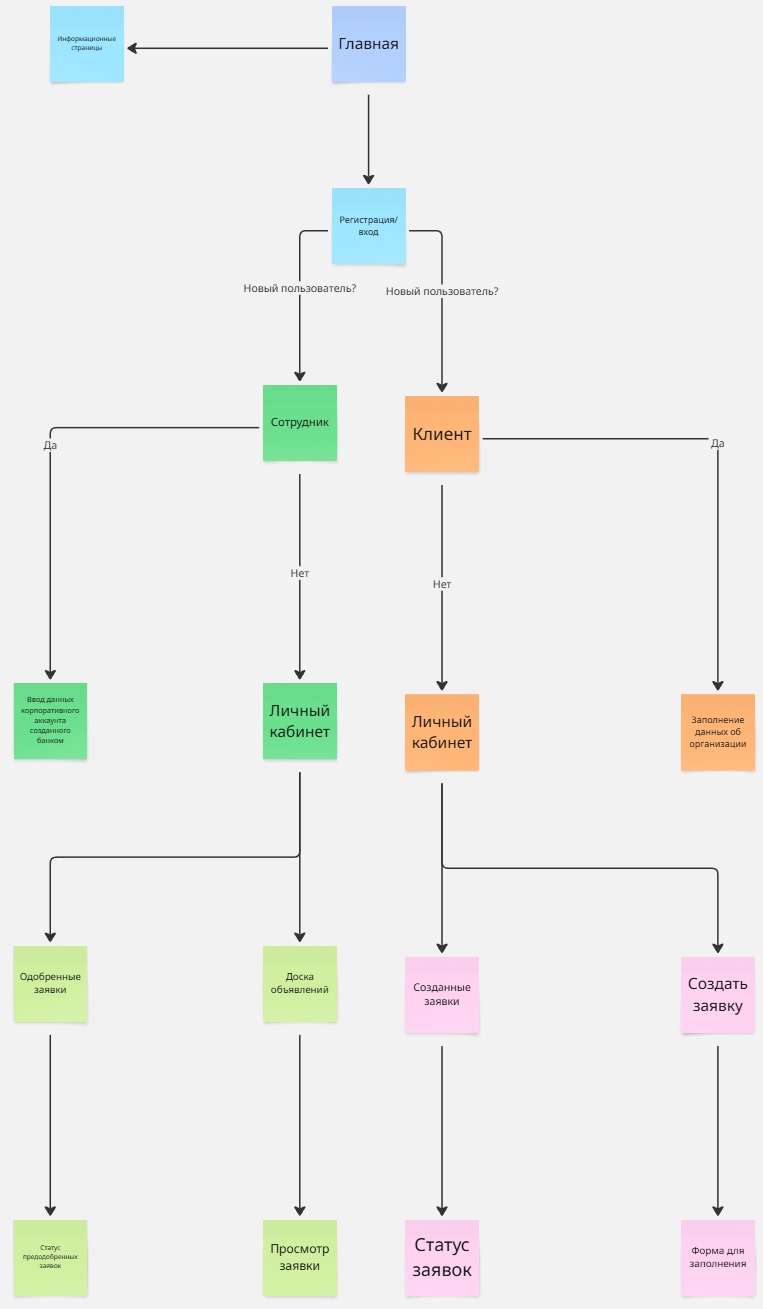


Рисунок 5 – Визуализация логики переходов средствами Miro

**Путь 1 Представитель компании:**

* **Главная** → **Авторизация**
* В блоке «Новый пользователь?» доступен переход к регистрации представителя компании.
* **Регистрация клиента** → форма заполнения информации о компании.
* После регистрации осуществляется переход в **Личный кабинет клиента**:
* **Создание заявки**
* **Прикрепление документов**
* **Просмотр статуса поданных заявок**

Этот маршрут направлен на реализацию ключевой задачи — подачу запроса на финансирование с прикреплением необходимых документов и отслеживанием результатов рассмотрения.

**Путь 2 Сотрудник банка:**

* **Главная** → **Авторизация**
* «Новый пользователь?» → регистрация сотрудника.
* **Личный кабинет сотрудника банка** включает доступ к следующим функциям:
* **Личная информация**
* **Просмотр поступивших заявок**
* **Назначение ответственного менеджера**
* **Обработка заявок**
* **Отправка решений в систему банка**
* **Просмотр статуса обработки заявок**

Каждый переход сопровождается передачей состояния и идентификаторов через URL-параметры и/или глобальное хранилище состояния.

**Компоненты и взаимодействие**

Интерфейс построен на основе следующих компонентов:

* **AuthenticationForm** — универсальный компонент для логина и регистрации.
* **Dashboard** — обёртка для личных кабинетов, различающаяся по ролям.
* **CompanyProfileForm**, **ClientApplicationForm**, **DocumentUploader** — специализированные формы для клиента.
* **ApplicationList**, **ApplicationReview**, **BankDecisionForm** — компоненты для сотрудников банка.
* **ProtectedRoute** — компонент-обёртка, проверяющий наличие токена и соответствие роли.
* **Notifications** — модуль для отображения статусов и сообщений об успехе/ошибках.
* **Header, Footer**, **Layout** — компоненты навигации и компоновки.

**Принципы UX/UI:**

Дизайн интерфейса разработан с приоритетом на:

* Простоту и предсказуемость пользовательских сценариев,
* Минимизацию количества шагов до целевого действия,
* Чёткое разграничение ролей и отображение только актуального функционала,
* Адаптивную верстку с учётом мобильных и планшетных устройств.

Пример визуализации представлен на рисунке 6.

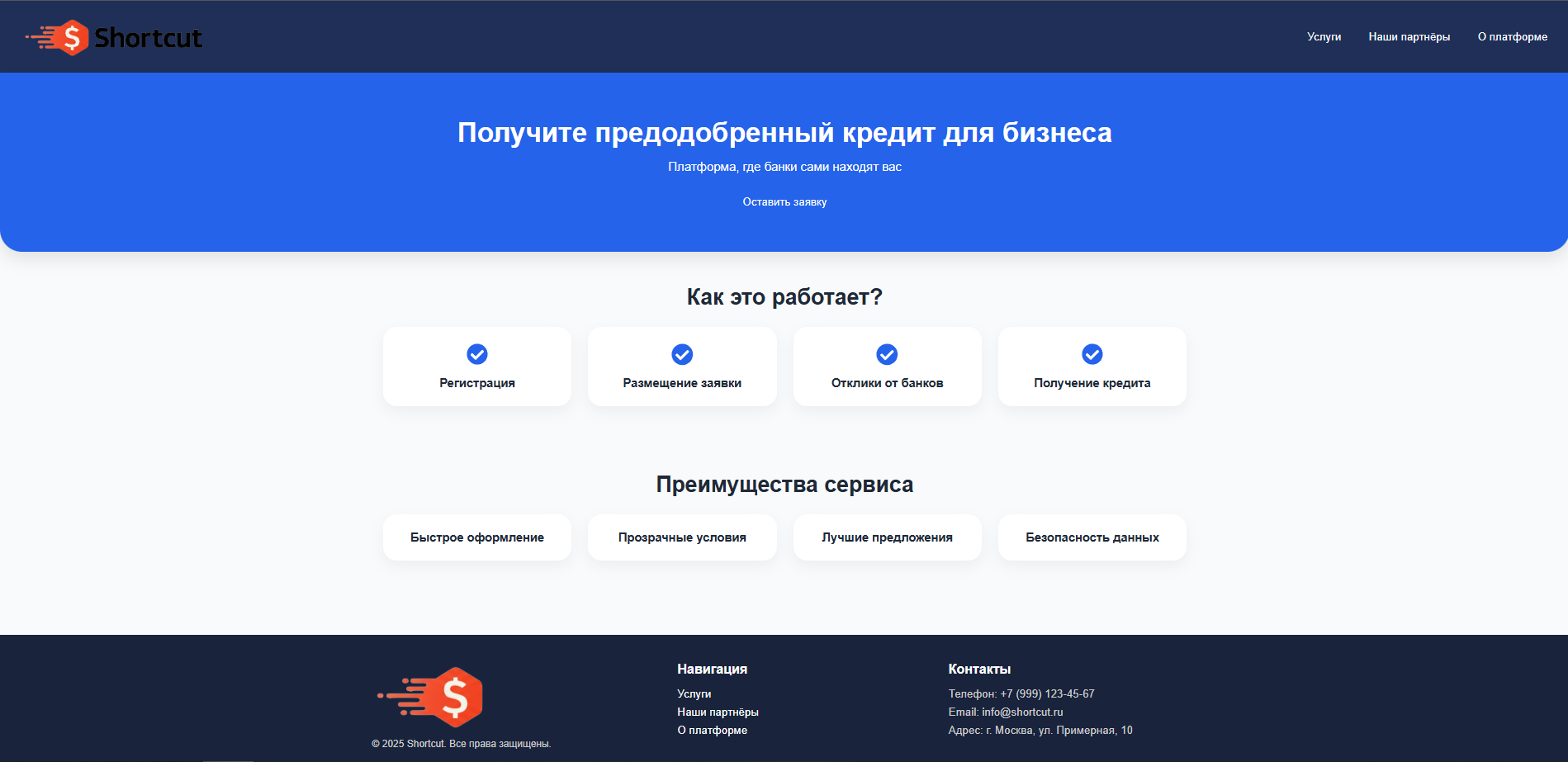


Рисунок 6 – Главная страница сервиса

При реализации интерфейса в приоритете находились минимализм и простота использования

* 1. Организация обмена данными с 1С через HTTP

Важной частью функциональности разработанного веб-сервиса является **двусторонний обмен данными с системой 1С**, используемой как со стороны компаний-клиентов, так и со стороны банков. Интеграция построена на основе **HTTP-запросов с использованием формата JSON**, что обеспечивает совместимость, масштабируемость и простоту реализации на обеих сторонах.

Обмен с системой 1С реализуется на двух ключевых этапах:

**Получение данных из 1С клиента (до подачи заявки)**

На этапе формирования заявки на получение финансирования, веб-сервис запрашивает у клиента предварительную информацию, необходимую для автоматического заполнения формы. Запрос инициируется при открытии формы создания заявки и обрабатывается следующим образом:

* **Frontend (React App)** отправляет запрос на сервер (Express.js) для получения исходных финансовых данных.
* Сервер, в свою очередь, обращается к встроенному **модулю интеграции с 1С Клиента**, который через HTTP-запрос получает необходимые данные (например, оборотно-сальдовую ведомость) из 1С-платформы клиента.
* Полученные данные преобразуются в формат, пригодный для отображения в пользовательском интерфейсе, и автоматически подставляются в соответствующие поля формы.
* Пользователь может при необходимости дополнительно отредактировать или подтвердить эти данные.

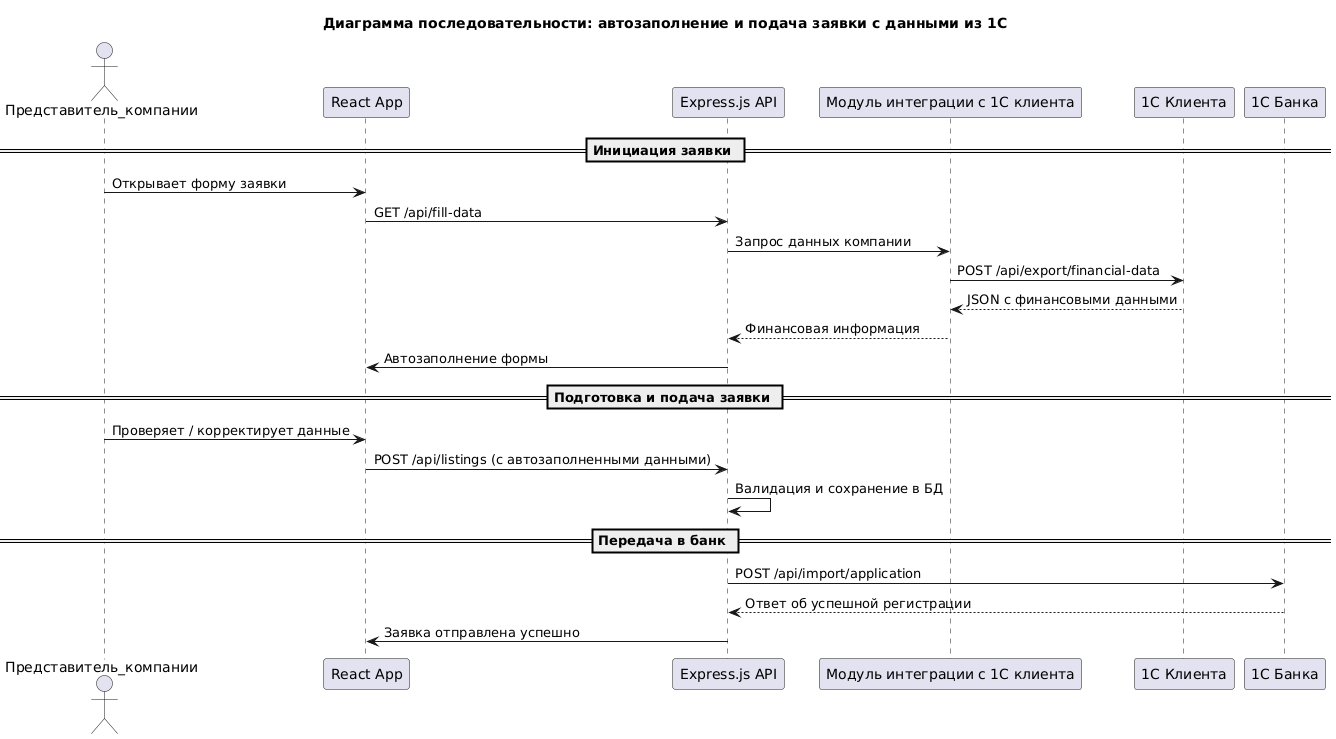
**Передача заявки и документов в 1С Банка**

После того как заявка подготовлена и подписана пользователем, она вместе с необходимыми документами отправляется в систему банка:

* Сервер формирует структурированный JSON-пакет, содержащий сведения о компании, приложенные документы, условия финансирования и прочие параметры.
* Через защищённый HTTP-интерфейс данные передаются в **1С Банка**, где происходит их регистрация и первичная обработка.
* Ответ от банковской системы об успешной передаче сохраняется в базе данных и отображается пользователю.

**Диаграмма последовательности**

На рисунке 7 представлена диаграмма последовательности, иллюстрирующая весь процесс интеграции с системами 1С:



**Рисунок 7 — Диаграмма последовательности: заполнение и подача заявки с данными из 1С**

Диаграмма включает следующих участников:

* **Представитель компании**, который инициирует процесс подачи заявки;
* **React App**, через который происходит взаимодействие;
* **Express.js API** — серверная логика;
* **Модуль интеграции с 1С клиента** — интерфейс обращения к локальной конфигурации 1С;
* **1С Клиента**, передающая данные;
* **1С Банка**, принимающая структурированную заявку.
  1. Роли пользователей: компания, банк

Информационная система, реализуемая в рамках данного проекта, предполагает два ключевых типа пользователей: **представителей компаний-клиентов** и **сотрудников банка**. Эти роли имеют различный набор прав и сценариев взаимодействия с системой, что отражается как в интерфейсе, так и в логике обработки данных.

**Представитель компании**

Пользователь с данной ролью регистрируется в системе, заполняет профиль своей организации и инициирует подачу заявки на кредитование. Его основная задача — сформировать корректную и полную заявку, включающую:

* основную информацию о компании;
* цель получения финансирования;
* пакет сопроводительных документов (например, финансовую отчётность).

Система позволяет автоматически подгружать данные из конфигурации 1С на стороне клиента, что упрощает процесс подачи и снижает вероятность ошибок при ручном вводе.

**Представитель компании может:**

* создать новую заявку;
* редактировать черновик до момента отправки;
* отслеживать статус заявки;
* получать обратную связь от банка;
* повторно отправлять заявку после доработки, если она была возвращена на корректировку.

**Сотрудник банка**

Пользователь, выступающий от имени кредитной организации, имеет доступ к потоку входящих заявок. Его действия зависят от стадии, на которой находится заявка:

* просмотр и первичный анализ;
* принятие решения (одобрение, отклонение, возврат на доработку);
* передача данных во внутренние банковские системы;
* назначение ответственного за ведение клиента.

Кроме того, менеджер банка может оставлять комментарии и инициировать коммуникацию с клиентом через предусмотренные каналы обратной связи (например, уведомления в интерфейсе).

**Жизненный цикл заявки**

Каждая заявка в системе проходит через ряд состояний, отражающих её жизненный цикл. На рисунке 8 представлена диаграмма состояний заявки, наглядно демонстрирующая основные переходы:

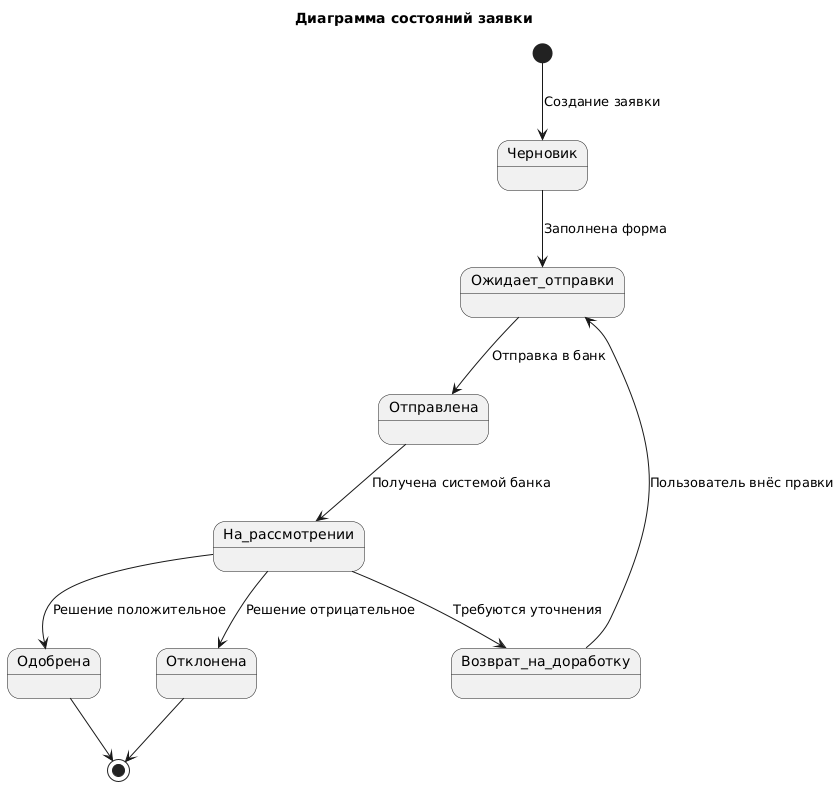


Рисунок 8 - Диаграмма состояний заявки на получение кредита

Заявка создаётся в статусе **«Черновик»**, после заполнения переходит в состояние **«Ожидает отправки»**. После нажатия кнопки «Отправить», она передаётся банку и получает статус **«Отправлена»**, а затем **«На рассмотрении»**.

В зависимости от решения банка, заявка может быть:

* **одобрена** — переходит в финальное состояние;
* **отклонена** — завершает жизненный цикл;
* **возвращена на доработку** — инициирует повторный цикл редактирования и подачи.
  1. Безопасность и авторизация пользователей

В любой системе, работающей с персональными и финансовыми данными, вопросы безопасности и контроля доступа являются критически важными. В рамках разрабатываемого веб-сервиса реализованы базовые механизмы **аутентификации, авторизации, управления ролями и безопасного хранения данных.**

**Модель пользователей и ролей**

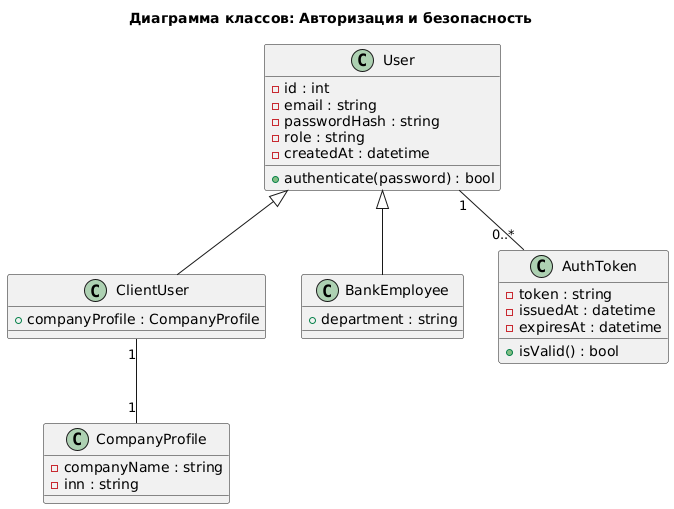
Все пользователи системы представляют собой экземпляры базового класса User, содержащего стандартные поля:

* email — используется в качестве логина,
* passwordHash — хэшированный пароль (с использованием алгоритма bcrypt или аналогичного),
* role — тип пользователя (client, bank\_employee),
* createdAt — дата регистрации.

На основе класса User реализуются две специализированные роли:

* ClientUser — представитель компании-клиента, имеющий связанный объект CompanyProfile, содержащий юридическую информацию (название, ИНН, КПП и т. д.).
* BankEmployee — сотрудник банка, связанный с внутренними процедурами обработки заявок. Может относиться к определённому подразделению.

Диаграмма классов на рисунке 4.4 наглядно иллюстрирует структуру данной модели.



**Рисунок 9 – Диаграмма классов, отражающая архитектуру безопасности и авторизации**

**Аутентификация и токены**

Для аутентификации пользователей используется **механизм JWT (JSON Web Token)**. После успешной авторизации клиент получает уникальный токен, который передаётся с каждым последующим запросом в HTTP-заголовке Authorization. Сервер проверяет подпись токена, его срок действия и принадлежность к конкретному пользователю.

Класс AuthToken включает поля:

* token — строковое представление токена;
* issuedAt — дата и время выпуска;
* expiresAt — время окончания действия.

Метод isValid() позволяет программно проверить подлинность и срок действия токена.

**Защита маршрутов и разграничение прав доступа**

Каждый защищённый маршрут в серверной части обрабатывается промежуточным middleware, который:

* Проверяет наличие и корректность токена;
* Определяет роль пользователя на основании payload токена;
* Сравнивает полученную роль с разрешёнными ролями маршрута.

Таким образом, например:

* Представители компаний имеют доступ только к маршрутам создания и просмотра собственных заявок;
* Сотрудники банка могут обрабатывать и просматривать входящие заявки, но не имеют доступа к созданию заявок.

**Хранение данных и безопасность на уровне сервера**

* Все пароли хранятся в базе данных только в хэшированном виде.
* Ключи подписи JWT хранятся в конфигурации сервера и не передаются на клиент.
* Доступ к административным функциям (например, назначение ответственных за клиента) ограничен на уровне ролевой модели.

**Потенциальные векторы развития**

В будущем возможно:

* Подключение **двухфакторной аутентификации (2FA),**
* Использование **ролевой модели на базе RBAC или ACL,**
* Интеграция с **внутренними системами авторизации банка** (например, LDAP),
* Отслеживание действий пользователей через систему **журналирования (audit trail).**

1. ****Анализ результатов****
   1. Анализ корректности и устойчивости работы системы

После завершения разработки веб-сервиса и интеграции с внешними системами была проведена серия тестов, направленных на проверку корректности функционирования всех компонентов, а также устойчивости системы к ошибкам, некорректным данным и сетевым сбоям.

**Тестирование проводилось по следующим направлениям:**

**Функциональная проверка компонентов интерфейса.** Были протестированы формы регистрации, авторизации, создания заявок и загрузки документов. Все поля форм прошли проверку на корректность валидации, автозаполнение данными из 1С происходило корректно. Ошибки при некорректных данных отображались предсказуемо.

**Тестирование API-запросов**

Сценарии включали корректные и ошибочные запросы к REST API. Система успешно обрабатывала:

* запросы от авторизованных пользователей;
* попытки несанкционированного доступа (возвращалась ошибка 403);
* ошибки в теле запроса (400 Bad Request);
* отсутствие токена (401 Unauthorized).

**Обработка исключений и нестандартных ситуаций**

Протестированы случаи:

* загрузки невалидных файлов;
* повторной отправки одной и той же заявки;
* удаления данных в момент отправки (конфликт доступа).

**Нагрузочное тестирование**

Сервис демонстрировал стабильную работу при последовательной подаче до 100 заявок в течение короткого промежутка времени. Были зафиксированы стандартные пики нагрузки на базу данных, но ошибок или потери данных не наблюдалось.

**Вывод по устойчивости:**

Система демонстрирует устойчивое поведение при типичных сценариях работы и нештатных ситуациях. Исключения обрабатываются корректно, основные бизнес-процессы завершались успешно. Сервис способен работать в условиях временной недоступности внешних компонентов, таких как 1С клиента или банка.

* 1. Выводы по результатам тестирования

Проведённое тестирование показало, что реализованное решение соответствует заданным требованиям и готово к эксплуатации в пилотном режиме. Были достигнуты следующие ключевые результаты:

* Обеспечена корректная работа пользовательского интерфейса с логическим разграничением по ролям;
* Все REST API-методы отрабатывали стабильно и предсказуемо при различных типах запросов;
* Автоматическое получение данных из 1С клиента успешно функционирует и ускоряет процесс подачи заявки;
* Система успешно интегрируется с внешним API 1С Банка и получает подтверждение получения заявок;
* Предусмотрены механизмы обработки ошибок и недоступности сервисов без потери данных.

В целом, созданная система продемонстрировала готовность к использованию в реальной среде и обладает потенциалом для масштабирования и внедрения дополнительных функций.

****Заключение****

В ходе выполнения выпускной квалификационной работы была разработана система, обеспечивающая эффективную интеграцию между платформой 1С и банковскими сервисами. Использование современных технологий, таких как REST API, JSON, а также библиотеки React и фреймворка Express.js, позволило создать надёжное и масштабируемое решение. Система обеспечивает автоматизированный обмен данными, снижая необходимость в ручной обработке документов и повышая точность финансовой информации.

Поставленные цели и задачи были успешно достигнуты:

* Проведён анализ современных методов интеграции 1С с внешними системами, включая использование REST API и формата JSON;
* Разработана архитектура веб-сервиса, обеспечивающая надёжный обмен данными между компаниями и банками;
* Реализован пользовательский интерфейс с использованием React, обеспечивающий удобство взаимодействия для конечных пользователей;
* Обеспечена безопасность и авторизация пользователей, соответствующие современным требованиям к защите данных.

Таким образом, все задачи, поставленные в начале работы, были выполнены, а цель — создание эффективной системы интеграции — достигнута.

Разработанное решение обладает высокой практической значимостью для организаций, стремящихся автоматизировать процесс обмена финансовыми данными с банками. Система может быть адаптирована под различные бизнес-процессы и интегрирована с другими информационными системами.

Возможные направления дальнейшего развития включают:

* Расширение функциональности системы для поддержки дополнительных форматов данных и протоколов обмена;
* Интеграция с другими банковскими сервисами и финансовыми платформами для расширения возможностей пользователей;
* Внедрение аналитических инструментов для более глубокой обработки и анализа финансовых данных;

Таким образом, разработанная система не только решает текущие задачи интеграции, но и обладает потенциалом для дальнейшего развития и адаптации к меняющимся требованиям бизнеса.

Список использованных источников

1. Платформа 1С:Предприятие. HTTP-сервисы. – URL: https://its.1c.ru/db/metod8dev/content/6211/hdoc (дата обращения: 01.04.2025). – Текст : электронный.
2. Платформа 1С:Предприятие. Внешние обработки. – URL: https://its.1c.ru/db/metod8dev/content/5267/hdoc (дата обращения: 01.04.2025). – Текст : электронный.
3. Обмен файлами в корпоративных системах. – Текст : электронный // Хабр. – 2021. – URL: https://habr.com/ru/articles/531888/ (дата обращения: 01.04.2025).
4. Интеграция 1С с брокерами сообщений RabbitMQ и Kafka. – Текст : электронный // Хабр. – 2022. – URL: https://habr.com/ru/articles/671812/ (дата обращения: 01.04.2025).
5. Fowler, M. Microservices – Text : electronic // martinfowler.com. – 2014. – URL: https://martinfowler.com/articles/microservices.html (дата обращения: 01.04.2025).
6. Microsoft. Circuit Breaker Pattern. – URL: <https://learn.microsoft.com/en-us/azure/architecture/patterns/circuit-breaker> (дата обращения: 01.04.2025). – Текст : электронный.
7. Richardson, C. Microservices Patterns: With examples in Java / C. Richardson. – Manning, 2018. – 520 p. – ISBN 978-1-61729-454-9.
8. React. Основные концепции // reactjs.org. – URL: https://reactjs.org/docs/getting-started.html (дата обращения: 01.04.2025). – Текст : электронный.
9. Express.js. Express API Reference. – URL: https://expressjs.com/en/api.html (дата обращения: 01.04.2025). – Текст : электронный.
10. PostgreSQL. Documentation. – URL: <https://www.postgresql.org/docs/> (дата обращения: 01.04.2025). – Текст : электронный.
11. Object Management Group (OMG). UML Superstructure Specification, Version 2.5. – 2015. – URL: https://www.omg.org/spec/UML/2.5 (дата обращения: 01.04.2025). – Текст : электронный.
12. Chen, P. P. The Entity-Relationship Model—Toward a Unified View of Data / P. P. Chen // ACM Transactions on Database Systems. – 1976. – Vol. 1, No. 1. – P. 9–36.
13. React Router. Руководство по маршрутизации. – URL: https://reactrouter.com/en/main (дата обращения: 01.04.2025). – Текст : электронный.
14. IETF. RFC 7519: JSON Web Token (JWT). – 2015. – URL: https://datatracker.ietf.org/doc/html/rfc7519 (дата обращения: 01.04.2025). – Текст : электронный.
15. Provos, N., Mazières, D. A Future-Adaptable Password Scheme (bcrypt) / N. Provos, D. Mazières // Proceedings of the USENIX Annual Technical Conference. – 1999. – URL: https://www.usenix.org/legacy/events/usenix99/provos.html (дата обращения: 01.04.2025). – Текст : электронный.