МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра Прикладная математика и информатика
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки / специальности)

Разработка программного обеспечения
(направленность (профиль) / специализация)

# ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка приложения для управления платежами и переводами между поставщиками и потребителями услуг


Обучающийся Д.В. Никонов
(Инициалы Фамилия) (личная подпись)

Руководитель Н.Н. Рогова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант доцент, М.В. Дайнеко
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)


Тольятти 2025

## Аннотация

Выпускная квалификационная посвящена разработке веб-приложения для управления платежами и переводами между потребителями и поставщиками услуг в организации «Квартплата24». Объектом исследования являются процессы приёма платежей и расчётов между сторонами внутри предприятия, предметом — методы автоматизации этих процессов с использованием современных информационных технологий.

Работа актуальна в условиях перехода к цифровым формам взаимодействия внутри организаций. Отсутствие централизованной системы ведёт к ручной обработке данных, высокой нагрузке на сотрудников и риску ошибок. Разработка веб-приложения позволяет автоматизировать обработку платежей, упростить управление справочниками, формировать переводы и отчёты, а также эмулировать интеграцию с банковской системой.

Цель работы — разработка веб-приложения, обеспечивающего автоматизацию расчётных операций между потребителями и поставщиками услуг, с возможностью расширения функционала в будущем.

Для достижения поставленной цели были проанализированы аналогичные решения, сформулированы функциональные и нефункциональные требования, разработана архитектура, реализованы основные модули и проведено тестирование.

Работа включает введение, три главы, заключение, список использованной литературы и приложения. В первой главе выполнен анализ предметной области, обзор существующих решений и формализация требований. Во второй главе изложены архитектура приложения, структура базы данных, интерфейсы и реализация. Третья глава посвящена тестированию, демонстрации интерфейса и оценке эффективности работы.

Объём работы составляет 53 страницы, включает 34 рисунка, 5 таблиц и 20 источников.

## Abstract

The title of the graduation work is “Development of an application for managing payments and transfers between consumers and service providers”.

The graduation work consists of an introduction, three chapters, 34 figures, 5 tables, a conclusion, a list of 20 references, and appendices.

The aim of this graduation work is to develop a web application that automates settlement operations between consumers and service providers with the potential for further functional extension.

The object of the graduation work is the processes of receiving payments and settlements between parties within the enterprise.

The subject of the graduation work is the methods of automating these processes using modern information technologies.

The key issue of the graduation work is the organization of centralized and secure management of financial operations with minimized errors and reduced staff workload.

The graduation work may be divided into several logically connected parts: analysis and formalization of requirements, design and implementation of the architecture, testing and performance evaluation.

The first chapter presents an analysis of the subject area, a review of existing solutions, and the formalization of requirements.

The second chapter outlines the application architecture, the database structure, interfaces, and the implementation of main modules.

The third chapter focuses on testing, interface demonstration, and performance evaluation of the system.

In conclusion, the main results are summarized, key findings are formulated, and prospects for further development of the application are noted.

The work is of interest to specialists in the field of payment management automation and information system development for the utility sector.

## Оглавление

Введение ..... 5
Глава 1 Постановка задачи на разработку веб-приложения для управления платежами и переводами ..... 7
1.1 Значение программного обеспечения в управлении платежами и переводами ..... 7
1.2 Основные бизнес-процессы и их формализация ..... 8
1.3 Обзор существующих решений и их анализ ..... 10
1.4 Определение требований к приложению ..... 13
Глава 2 Проектирование программного обеспечения ..... 18
2.1 Архитектурные и структурные решения ..... 18
2.2. Выбор технологий и инструментальных средств ..... 22
2.3. Физическая модель базы данных ..... 24
2.4. Визуализация архитектуры и интерфейса приложения ..... 26
Глава 3 Оценка эффективности разработанного программного обеспечения ..... 32
3.1 Тестирование и отладка приложения ..... 32
3.2 Функциональное тестирование ..... 32
3.3 Нагрузочное тестирование ..... 33
3.4 Демонстрация функционала ..... 34
Заключение ..... 50
Список используемой литературы и источников ..... 52

# Введение

Организации, работающие в сфере жилищно-коммунального обслуживания, ежедневно сталкиваются с необходимостью учёта большого количества платежей, управления справочниками поставщиков и потребителей, а также формирования отчётной информации. Ручная обработка таких данных часто приводит к ошибкам, задержкам и росту нагрузки на персонал. Особенно остро такая проблема проявляется в среде локальных предприятий, где использование устаревших методов и отсутствие автоматизации снижают эффективность бизнес-процессов.

Предприятие «Квартплата24», выполняющее функции приёма платежей от потребителей и перевода средств поставщикам услуг, нуждается в удобной и безопасной информационной системе, способной автоматизировать обработку финансовых данных. Такая система должна обеспечивать не только ввод и учёт платежей, но и формирование переводов, отчётов, работу со справочниками и защиту доступа к информации. Современные веб-технологии позволяют реализовать эти задачи с высокой степенью гибкости и масштабируемости.

Целью данной выпускной квалификационной работы является разработка веб-приложения для управления платежами и переводами между потребителями и поставщиками услуг, ориентированного на внутреннее использование сотрудниками организации.

Объектом исследования выступает деятельность предприятия в части приёма платежей и расчётов с поставщиками.

Предметом исследования является проектирование, реализация и внедрение веб-приложения управления платежами и переводами на основе современных технологий.

Для достижения поставленной цели в работе были решены следующие задачи:

– выполнен анализ существующих решений и формализованы требования к системе;

– спроектированы архитектура приложения, структура базы данных и интерфейс;

– реализован веб-функционал по работе с платежами, переводами, справочниками и отчётами;

– обеспечена защита доступа через регистрацию с кодом и двойную аутентификацию;

– проведено функциональное и нагрузочное тестирование;

– продемонстрирована работа всех основных сценариев приложения.

Методологической основой работы является структурный и объектно-ориентированный подход к проектированию программного обеспечения, с использованием диаграмм UML и принципов многослойной архитектуры веб-приложений.

# Глава 1 Постановка задачи на разработку веб приложения для управления платежами и переводами

## 1.1 Значение программного обеспечения в управлении платежами и переводами

Для предприятий, оказывающих услуги в сфере жилищно-коммунального хозяйства, эффективность финансового управления напрямую влияет на стабильность операционной деятельности. Одной из ключевых задач в данной области является организация своевременного и прозрачного распределения платежей между потребителями и поставщиками услуг. В условиях большого потока данных, многообразия расчетных схем и необходимости строгого контроля каждое предприятие сталкивается с рядом характерных проблем.

На примере компании «Квартплата24» выявлены следующие сложности: отсутствие централизованной информационной системы; необходимость ручного подбора платежей под сумму перевода; невозможность быстрого формирования отчетности; высокая нагрузка на сотрудников, вынужденных обрабатывать данные вручную; отсутствие ограниченного доступа и авторизации для сотрудников. Эти факторы не только увеличивают количество ошибок, но и замедляют рабочие процессы, затрудняют контроль и снижают прозрачность расчетов [5].

Выявленная проблема заключается в отсутствии специализированного программного обеспечения, способного централизовать управление платежами, автоматизировать подбор сумм при формировании переводов, обеспечить защищенный доступ к системе только для сотрудников и предоставить удобные инструменты для формирования отчетов.

Для наглядной формализации причин описанных затруднений разработана диаграмма причинно-следственных связей низкой эффективности

управления платежами и переводами между поставщиками и потребителями услуг (рисунок 1).

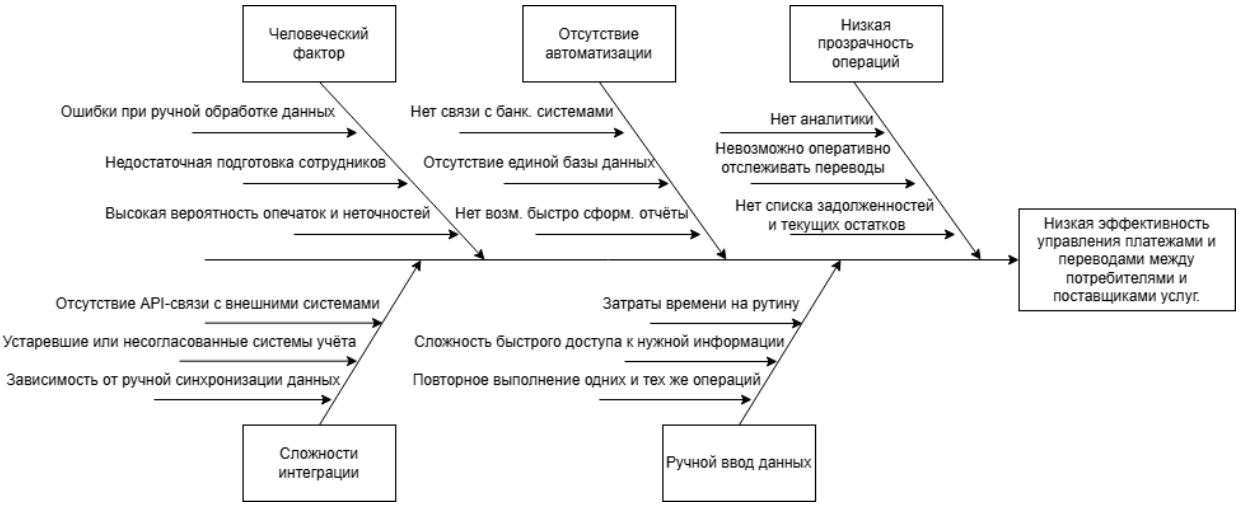


Рисунок 1 - Диаграмма причин и следствий низкой эффективности управления платежами и переводами

Диаграмма отражает основные первопричины низкой эффективности финансового управления: человеческий фактор, отсутствие автоматизации, ручной ввод данных, недостаточная прозрачность и отсутствие интеграции с внешними системами. Эти факторы в совокупности формируют предпосылки для разработки специализированного решения, адаптированного к специфике деятельности предприятия.

## 1.2 Основные бизнес-процессы и их формализация

Разработка программного обеспечения начинается с анализа и описания ключевых бизнес-процессов, которые подлежат автоматизации. В контексте деятельности предприятия «Квартплата24» основной задачей является организация управления финансовыми потоками - от ввода платежных данных до формирования отчетности и проведения переводов. Важную роль в этом

процессе играют операторские действия, внутренняя логика приложения и взаимодействие с внешними системами.

К числу основных процессов, реализуемых в рамках приложения, относятся:

- ввод и проверка данных о платеже. Оператор указывает номер лицевого счета, сумму и дату платежа. Система автоматически проверяет корректность и уникальность введенных данных;

- редактирование справочников поставщиков и потребителей. Предусмотрены функции добавления, изменения и удаления записей в справочниках, используемых при формировании переводов;

- формирование перевода. Приложение отображает общую сумму непереведенных платежей конкретному поставщику. Оператор указывает требуемую сумму перевода, после чего система автоматически подбирает комбинацию платежей, максимально близкую к указанной сумме;

- генерация отчетов. Система предоставляет оператору возможность сформировать отчеты по совершенным операциям в удобной форме;

- интеграция с внешними API. Предусмотрена имитация отправки данных о переводах во внешнюю банковскую систему;

На рисунке 2 приведена диаграмма потоков данных (Data Flow Diagram, DFD) уровня A0, отражающая потоки информации между пользователями, модулями системы и внешними источниками. Она формализует основные процессы, отображает связи между ними и иллюстрирует движение данных по системе. Данная модель позволяет обосновать архитектурные решения и обеспечить структурированное описание компонентов системы.

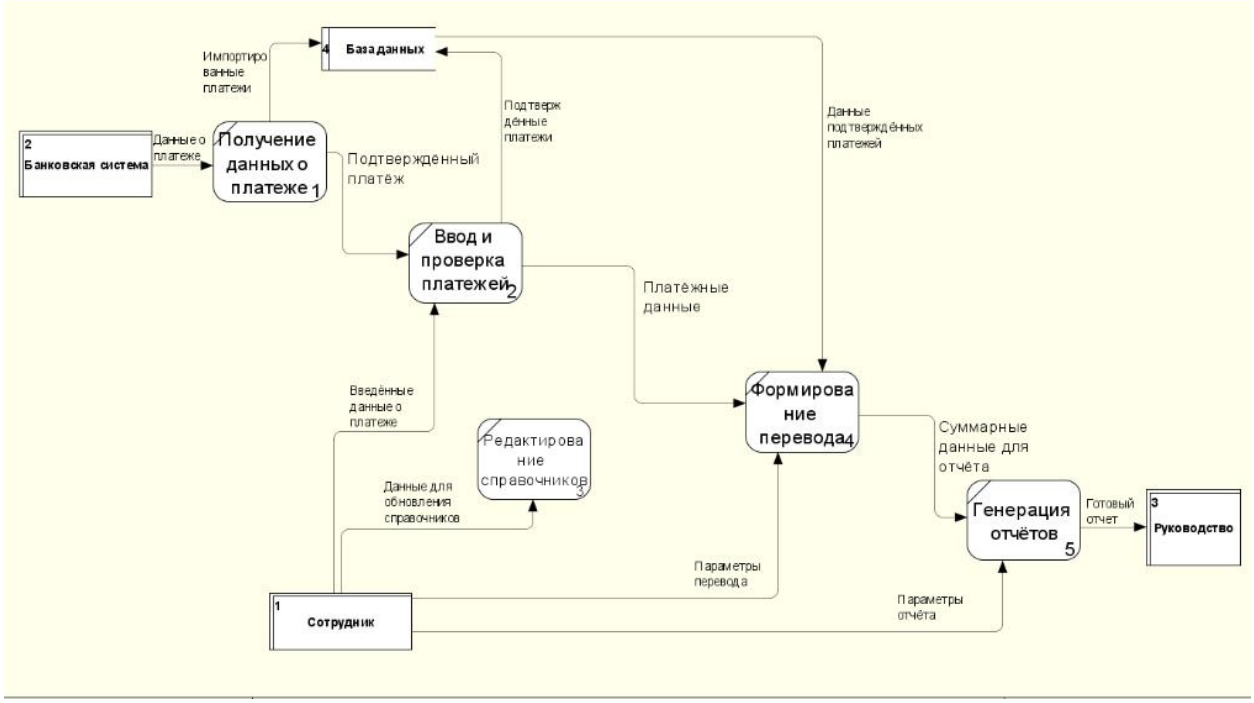


Рисунок 2 - DFD Диаграмма потока данных уровня A0 для приложения по управлению платежами и переводами

Диаграмма включает в себя внешние сущности (сотрудник, банковская система и руководство), пять базовых процессов (получение данных о платеже, ввод и проверка данных, формирование перевода, отчетность и управление справочниками), а также одно хранилища информации платежей: база данных. Такое представление позволяет формализовать структуру будущей системы и определить границы ответственности ее компонентов.

### 1.3 Обзор существующих решений и их анализ

Далее рассмотрен анализ современных программных продуктов, предназначенных для автоматизации финансовых процессов в сфере жилищно-коммунальных услуг. Обзор сосредоточен на решениях, которые используются для централизованного учета платежей, распределения средств и обеспечения прозрачности финансовых операций. Среди рассмотренных систем можно выделить следующие продукты:

«ГИС ЖКХ» - представляет собой государственную информационную систему, разработанную в соответствии с нормативными требованиями в сфере жилищно-коммунального хозяйства. Основное преимущество данного решения заключается в наличии единой базы данных, которая минимизирует риск ошибок при передаче информации. Однако, жестко заданная структура системы и сложность ее освоения затрудняют адаптацию под специфические бизнес-процессы отдельных предприятий. Кроме того, ограниченная гибкость настроек затрудняет возможность внесения оперативных изменений в функциональность системы.

«Домоплат» - является облачным решением, ориентированным на малые и средние управляющие компании. Ключевыми преимуществами данной системы являются интуитивно понятный интерфейс и возможность быстрого развертывания продукта без значительных временных затрат. Тем не менее, «Домоплат» ограничен в возможностях кастомизации: автоматический обмен данными с внешними финансовыми организациями отсутствует, что вынуждает пользователей прибегать к дополнительным ручным операциям. Это ограничивает применение системы в условиях высоких требований к автоматизации процессов.

«МКД Онлайн» - разработана с акцентом на управление многоквартирными домами и автоматизацию учета платежей. Система предлагает удобное мобильное приложение для контроля платежей и подачи заявок, что улучшает коммуникацию с конечными пользователями. Однако сложность внесения изменений в стандартный функционал снижает гибкость данного решения. Из-за этого адаптация МКД Онлайн к специфическим требованиям конкретного предприятия может оказаться затруднительной, особенно при необходимости интеграции с корпоративными информационными системами.

Проведенный сравнительный анализ показывает, что каждое из рассмотренных решений имеет свои достоинства и недостатки. В случае ГИС ЖКХ наблюдается высокая надежность и соответствие законодательным

требованиям, однако жесткая архитектура ограничивает возможности настройки. «Домоплат» выигрывает за счет простоты и быстроты внедрения, но уступает по функциональной гибкости и автоматизации обмена данными. «МКД Онлайн» демонстрирует удобство использования для конечных пользователей, но не всегда может удовлетворить требования предприятий к масштабируемости и адаптивности системы. Сравнение ключевых критериев оценки представлены на таблице 1.

Таблица 1 - Сравнение аналогов по критериям

| Критерии | ГИС ЖКХ | Домоплат | МКД Онлайн |
|---|---|---|---|
| Тип системы | Государственная информационная система для учета данных в сфере ЖКХ. | Облачное решение для автоматизации расчетов ЖКХ. | Российская система для управления финансовыми и коммунальными услугами. |
| Основной функционал | Учет услуг ЖКХ | Автоматизация услуг ЖКХ | Управление услугами ЖКХ |
| Доступность | Через веб-интерфейс | Через веб-интерфейс | Через веб-интерфейс и мобильный интерфейс |
| Гибкость настройки | Низкая | Средняя | Средняя |
| Удобство использования | Сложное | Удобное | Сложное |
| Интеграция | Ограниченная | Ограниченная | Средняя |

Анализ существующих решений показал, что ни одно из них не обеспечивает необходимую степень гибкости и автоматизации процессов, связанных с управлением платежами и формированием переводов. Все платформы имеют фиксированную бизнес-логику, не позволяют адаптировать распределение платежей под внутренние правила компании, а также не реализуют функции подбора суммы перевода из набора непереведенных платежей. Кроме того, отсутствует полноценная авторизация сотрудников, формирование отчетов ограничено и не поддерживается имитация взаимодействия с внешними API.

Предлагаемое решение устраняет эти ограничения. Система разрабатывается с учетом конкретных потребностей компании «Квартплата24» и позволяет: вводить и проверять данные о платежах; редактировать справочники поставщиков и потребителей; формировать переводы с автоматическим подбором подходящих платежей; ограничивать доступ с помощью секретного кода; генерировать отчеты на основе внутренних данных; имитировать обмен с банковскими API.

Сравнительный анализ подтвердил обоснованность разработки новой системы, ориентированной на специфику работы предприятия. Разработка нацелена на решение выявленных проблем и обеспечит необходимый уровень автоматизации, защищенности и управляемости процессов. Именно на основе этой оценки формируются функциональные и нефункциональные требования, описанные далее.

## 1.4 Определение требований к приложению

На основании анализа предметной области, выявленных проблем предприятия, бизнес-процессов, а также оценки аналогов, были сформулированы функциональные и нефункциональные требования к разрабатываемой информационной системе.

Для представления пользовательских сценариев взаимодействия с системой разработана диаграмма вариантов использования (рисунок 3). Она отображает основные действия, которые могут выполнять акторы, и связи между ними.

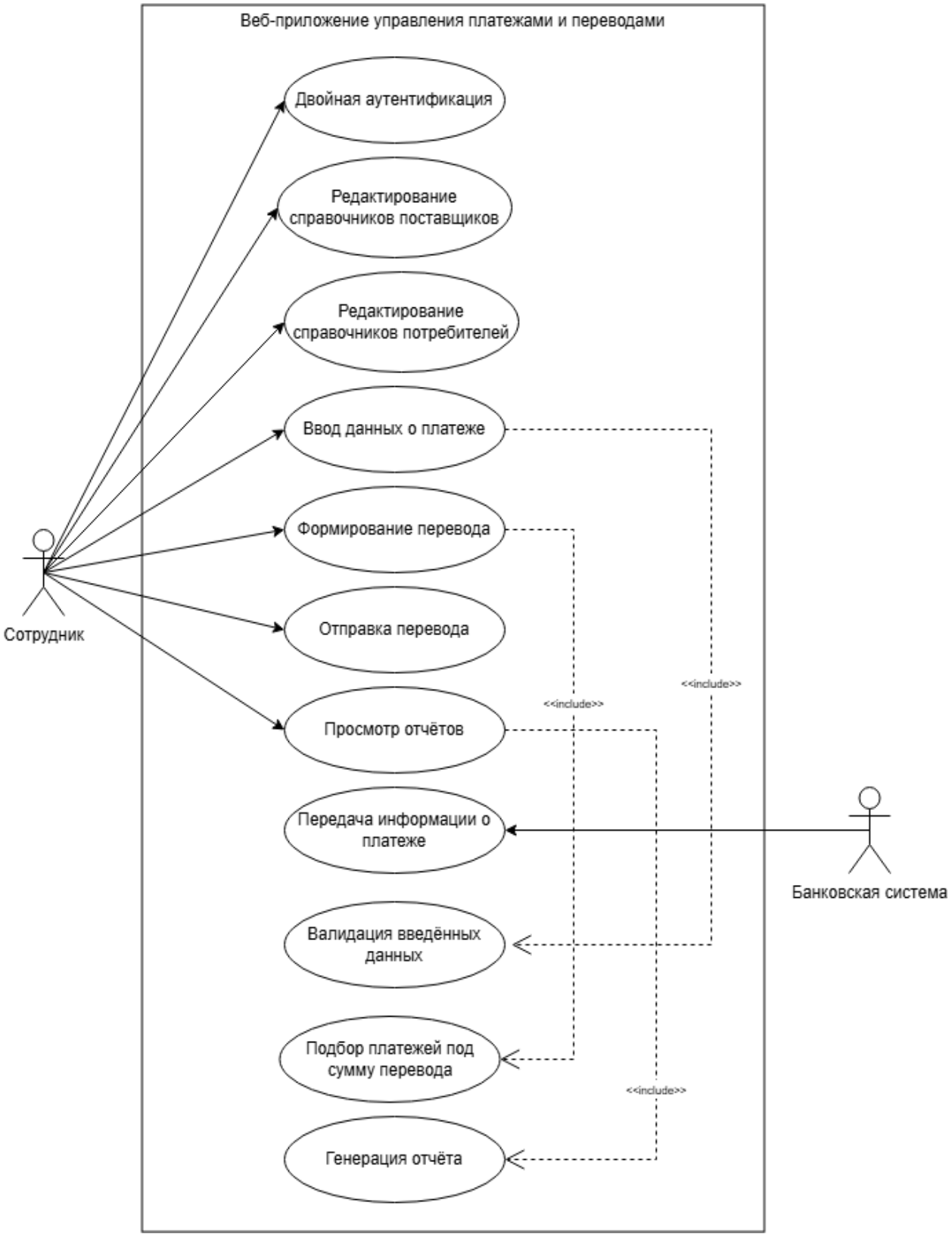


Рисунок 3 - Диаграмма вариантов использования приложения управления платежами и переводами

Диаграмма включает двух акторов - оператора (сотрудника) и банковскую систему - а также основные варианты использования: двойная аутентификация, редактирование справочников, ввод данных, формирование перевода, отправка данных во внешний API и генерация отчетов. Отношения <<include>> отражают обязательную вложенность отдельных операций (например, валидация данных включена в процесс ввода, подбор платежей включен в формирование перевода). Диаграмма отражает основной набор пользовательских сценариев, которые позже легли в основу проектирования архитектуры и взаимодействия компонентов.

Для систематизации требований использована модель FURPS+, включающая как функциональные требования, так и нефункциональные требования (удобство использования, надежность, производительность, сопровождаемость и дополнительные характеристики). Требования сформированы с учетом роли оператора (сотрудника управляющей организации), взаимодействующего с системой через веб-интерфейс, а также логики бизнес-процессов, описанных ранее. Подробная классификация приведена в таблице 2.

Таблица 2 - Классификация требований к системе в соответствии с моделью FURPS+

| Категория | Требования |
|---|---|
| Функциональность | – Ввод данных о платеже (№ л/с, сумма, дата) через интерфейс<br>– Редактирование справочников «Поставщики» и «Потребители»<br>– Формирование перевода с подбором платежей под сумму<br>– Генерация отчетов по операциям<br>– Двойная аутентификация сотрудников через секретный код |
| Удобство использования | – Интуитивно понятный интерфейс<br>– Минимальное время обучения для операторов<br>– Адаптивный дизайн для разных экранов |

Продолжение таблицы 2

| | |
|---|---|
| Надежность | – Проверка введенных данных на дублирование и ошибки<br>– Исключение некорректных сумм<br>– Стабильность работы при постоянной нагрузке |
| Производительность | – Обработка операций в течение ≤2 секунд<br>– Поддержка одновременной работы 10+ сотрудников<br>– Быстрое формирование отчетов |
| Дополнительные характеристики | – Возможность масштабирования (добавление новых модулей)<br>– Интеграция с внешними API (банк)<br>– Защита доступа: доступ только для сотрудников<br>– Логика расширения на дополнительные типы платежей |

FURPS+ охватывает ключевые параметры системы: функциональность, удобство, надежность, производительность, безопасность и возможность масштабирования. Эти параметры учитывались при проектировании архитектуры и интерфейса приложения.

Вывод по 1 главе.

В первой главе выполнен анализ текущей ситуации в сфере автоматизации финансовых процессов для предприятий жилищно-коммунального сектора. Установлено, что существующие решения («ГИС ЖКХ», «Домоплат», «МКД Онлайн») не обеспечивают достаточную гибкость настройки, не поддерживают автоматическое распределение платежей по заданным параметрам и не интегрируются полноценно с внешними банковскими сервисами.

Выявленная проблема заключается в отсутствии специализированного программного обеспечения, которое позволяло бы централизованно управлять платежами, выполнять интеллектуальное формирование переводов и обеспечивать защищенный доступ к аналитическим данным. Это обуславливает необходимость создания нового программного решения.

На основе выявленных потребностей определены функциональные и нефункциональные требования к системе. Диаграмма потоков данных

позволила формализовать структуру ключевых процессов, а диаграмма вариантов использования отразила взаимодействие пользователя с системой. Сформулированные требования классифицированы с использованием модели FURPS+, что позволит учесть как поведенческие, так и качественные характеристики при проектировании системы [5].

Глава 1 заложила обоснованную основу для разработки программного продукта, ориентированного на специфику работы предприятия и способного повысить эффективность управления финансовыми потоками. В результате проведённого анализа и формализации требований обеспечены необходимые предпосылки для последующего проектирования архитектуры системы. Определённые функциональные и нефункциональные требования создают базу для реализации всех заявленных задач, а выявленные особенности предметной области учитываются при выборе оптимальных решений на следующих этапах разработки.

# Глава 2 Проектирование программного обеспечения

## 2.1 Архитектурные и структурные решения

Разработка веб-приложения для управления платежами и переводами между потребителями и поставщиками требует тщательной проработки внутренней архитектуры. Важными задачами на этапе проектирования стали: определение логики пользовательского взаимодействия, описание состояний ключевых объектов, выбор технологий реализации, построение модели базы данных и структурирование компонентов приложения.

На следующем этапе была выполнена формализация всех элементов архитектуры, включая поведенческие сценарии, структуру базы данных, логику работы с данными, механизмы аутентификации и взаимодействие с внешней банковской системой. Отдельное внимание уделено модульности системы, обеспечивающей раздельную реализацию бизнес-логики, представления и взаимодействия с базой данных.

Архитектура ориентирована на упрощение сопровождения и масштабирования проекта. Для обеспечения полной прослеживаемости операций реализовано логирование изменений и статусных переходов, что критично при работе с финансовыми данными. Комплексное описание проектных решений представлено через набор диаграмм, отражающих поведенческие, структурные и функциональные аспекты системы.

### 2.1.1 Поведение пользователя в системе

Логика пользовательского взаимодействия с системой представлена с помощью диаграммы деятельности. Последовательность действий охватывает как обязательные шаги (вход в систему, навигация), так и прикладные операции, такие как работа со справочниками поставщиков и потребителей, формирование и сохранение платежей, создание переводов и экспорт отчётов. Также реализован механизм интеграции с внешней банковской системой для загрузки актуальных данных. Взаимодействие организовано таким образом,

чтобы минимизировать количество переходов между экранами и обеспечить предсказуемость поведения интерфейса. Диаграмма деятельности представлена на рисунке 4.

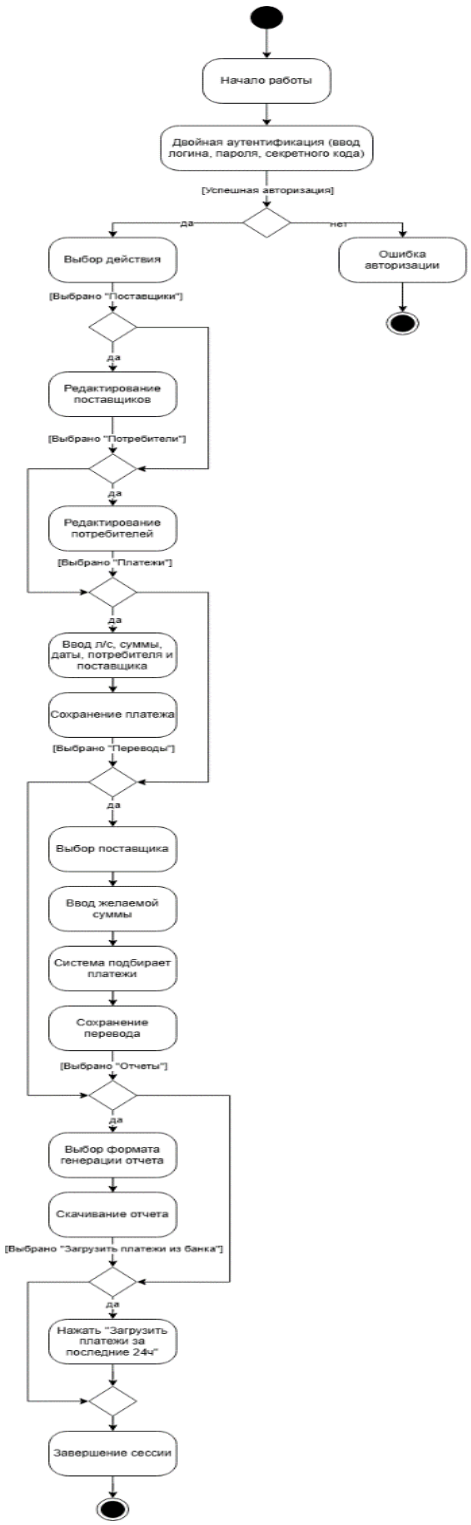


Рисунок 4 - Диаграмма деятельности пользователя

На диаграмме показаны сценарии:

- успешная и неуспешная авторизация;
- выбор раздела после входа;
- работа со справочниками, платежами, переводами, отчетами;
- загрузка платежей из банка.

Каждое действие ведет к возврату на главный выбор, что соответствует концепции модульного взаимодействия и циклической работы пользователя с системой [14].

Для отражения поведения конкретного объекта в процессе жизненного цикла, была разработана диаграмма состояний для объекта «Платеж» (см. рисунок 5). Она показывает, как платеж переходит из состояния созданного платежа в проверенное состояние, может быть отклонен, включен в перевод или удален.

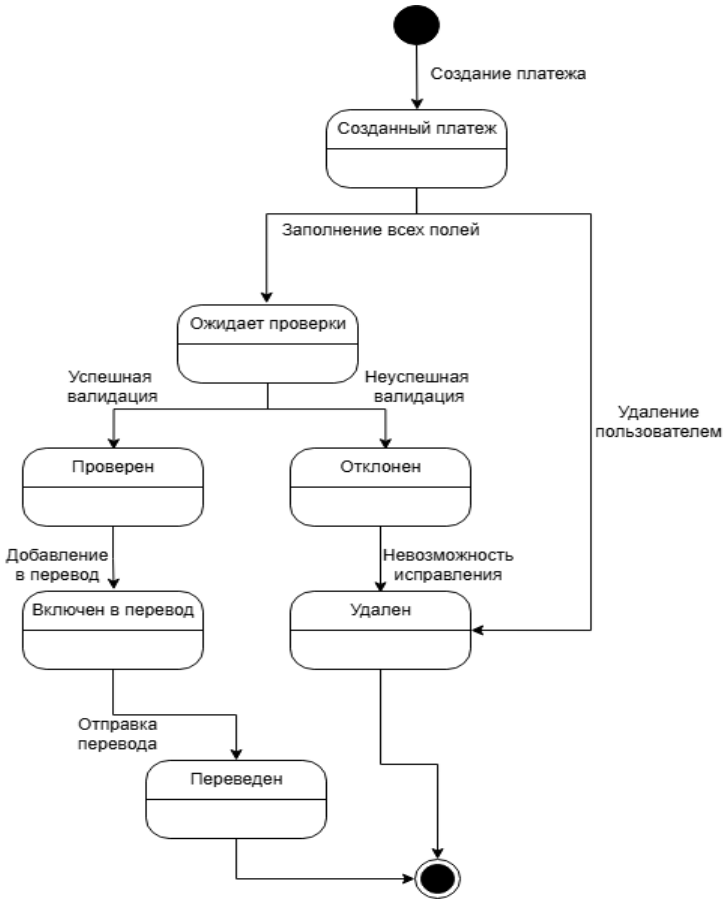


Рисунок 5 - Диаграмма состояний объекта «Платеж»

Возможные состояния:

- созданный платеж,
- ожидает проверки,
- проверен,
- отклонен,
- удален,
- включен в перевод,
- переведен.

Переходы осуществляются в зависимости от действий оператора и результатов валидации.

### 2.1.2 Структура классов и сущностей

Архитектура системы построена на объектной модели, отражающей основные элементы предметной области. На рисунке 6 представлена диаграмма классов, в которой описаны ключевые сущности, их атрибуты, методы и логические связи.

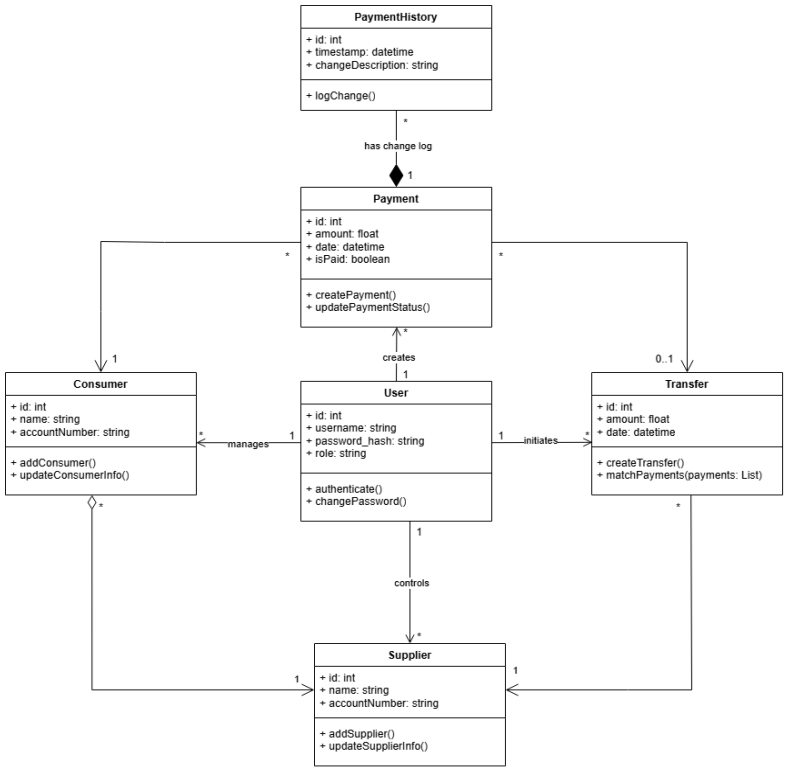


Рисунок 6 - Диаграмма классов приложения

а) сущности:

1) user - содержит логин, хэш пароля, поле role и метод changePassword() (зарезервирован для масштабирования: разграничение доступа и ролей);

2) payment - сумма, дата, потребитель, поставщик, статус и ссылка на пользователя, который создал платеж;

3) transfer - содержит сумму перевода, дату и связь с одним или несколькими платежами;

4) supplier и consumer - справочники поставщиков и потребителей;

5) paymentHistory - журнал изменений платежей.

б) связи:

1) user ассоциирован с другими сущностями: он инициирует создание платежей, переводов, редактирование справочников, отчеты;

2) payment связан с Transfer через таблицу payment_transfer (отношение M:N);

3) supplier и consumer участвуют в платежах как внешние стороны.

Модель классов полностью согласована с логикой предметной области и легла в основу структуры базы данных [1].

## 2.2 Выбор технологий и инструментальных средств

Для реализации программного обеспечения использовался стек технологий, обеспечивающий модульность архитектуры, читаемость и поддержку кода, совместимость с реляционной СУБД, а также простоту сопровождения и последующего расширения функционала. Все выбранные инструменты хорошо документированы и имеют широкое сообщество поддержки, что снижает технические риски и ускоряет разработку. Кроме того, технологии обеспечивают стабильную интеграцию между серверной и клиентской частью, а также поддержку масштабируемости. Выбор также обусловлен их применимостью к проектам с ограниченными сроками

разработки и минимальными затратами. Выбранные технологии для реализации веб-приложения приведены в таблице 3.

Таблица 3 - Выбранные технологии и инструментальные средства

| Компонент системы | Технология / Инструмент | Назначение |
|---|---|---|
| Язык программирования | Python | Основной язык серверной логики и обработки данных |
| Веб-фреймворк | Flask | Организация маршрутов, форм и логики интерфейса |
| ORM | SQLAlchemy | Связь с базой данных и управление транзакциями |
| СУБД | PostgreSQL | Хранение сущностей: пользователи, платежи, переводы |
| Шаблонизатор | Jinja2 | Генерация HTML-шаблонов на основе данных из сервера |
| Отчеты | ReportLab, модуль csv | Генерация PDF- и CSV-документов |
| IDE | Visual Studio Code | Среда разработки |
| Контроль версий | Git | Отслеживание изменений и управление кодовой базой |

Выбор языка Python обусловлен его лаконичным синтаксисом, богатой экосистемой и доступностью библиотек для веб-разработки. Он обеспечивает высокую скорость разработки и низкий порог вхождения [11, 13, 19].

В связке с Python был использован микрофреймворк Flask, который отличается гибкостью, легкостью настройки и отсутствием избыточной архитектурной нагрузки. Это особенно важно для проектов средней сложности, где нет необходимости в тяжелых структурах, как, например, в Django.

Связь между бизнес-логикой и базой данных реализована с помощью SQLAlchemy - мощной ORM-библиотеки, позволяющей описывать таблицы в виде Python-классов [17]. Это упрощает разработку, позволяет избегать прямого SQL-кода и минимизирует риск ошибок при работе с запросами.

В качестве шаблонизатора выбран Jinja2 - встроенное решение во Flask. Он позволяет генерировать HTML-страницы с наследованием шаблонов, циклами и условиями. Это упрощает поддержку и переиспользование фронтенд-части.

Хранение информации организовано в PostgreSQL - надежной СУБД с поддержкой транзакций, каскадных ограничений, индексов и работы со сложными связями между таблицами [3]. Это особенно важно при реализации отношений «многие ко многим» и отслеживании изменений данных.

Для формирования отчетов использовались модули csv (встроенный в Python) и ReportLab, позволяющий формировать отчеты в PDF с таблицами, шрифтами и заголовками [2]. Выбор этих инструментов обусловлен их кроссплатформенностью и простотой генерации документов.

Разработка велась в среде Visual Studio Code, которая обладает встроенной поддержкой Python, HTML и Jinja2, а также гибкой системой расширений и интеграцией с Git.

Система контроля версий Git применялась для управления историей изменений, сохранения промежуточных состояний и синхронизации кода при возможной командной работе.

Дополнительно для анализа структуры базы данных использовался pgAdmin4 - графический интерфейс для PostgreSQL, применявшийся исключительно на этапе разработки и визуализации схемы.

### 2.3 Физическая модель базы данных

На основании выбора технологий, представленных в пункте 2.2, в качестве СУБД используется PostgreSQL. Физическая структура базы данных

реализована через ORM SQLAlchemy и полностью соответствует логической модели. На рисунке 7 показана диаграмма физической модели базы данных.

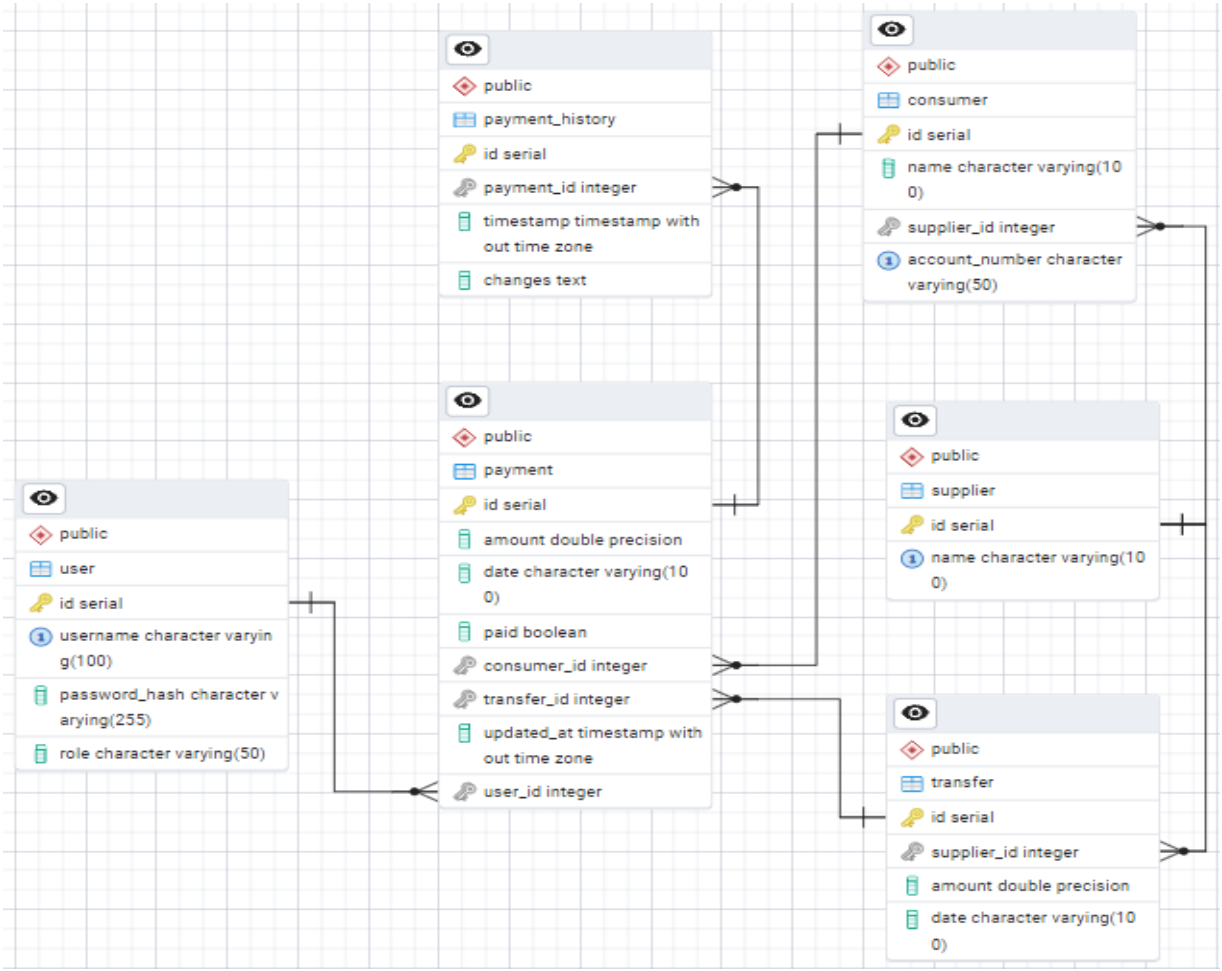


Рисунок 7 - Физическая модель базы данных приложения

В таблице user хранятся данные сотрудников: логин, хэш пароля, роль, идентификатор. Таблица связана с payment по полю user_id, что позволяет фиксировать автора каждого платежа. Эта связь заложена для обеспечения аудита и будущего разграничения доступа. Таблица payment содержит сумму, дату, ссылки на поставщика (supplier_id) и потребителя (consumer_id), статус оплаты и связь с переводами. Таблица transfer хранит информацию о переводах: сумма, дата, статус. Существует промежуточная таблица payment_transfer, реализующая отношение многие ко многим между payment

и transfer. Supplier и Consumer - справочники сторон. Каждая запись может быть связана с несколькими платежами. PaymentHistory ведет журнал изменений статуса платежей, позволяя отслеживать, кто и когда внес изменения (без привязки к user в текущей реализации).

Связи между таблицами реализованы через внешние ключи. Используются каскадные ограничения, типы serial, varchar, numeric, и индексы на основные поля. Структура соответствует третьей нормальной форме, исключает избыточность и обеспечивает логическую целостность данных. Использование одной структуры в качестве логической и физической модели упрощает поддержку проекта и делает его архитектуру прозрачной.

Структура базы данных была спроектирована с учетом принципов нормализации, что позволило исключить дублирование данных и обеспечить логическую целостность. Использование PostgreSQL обеспечивает поддержку сложных запросов, надежное хранение и масштабируемость. Реализация структуры в pgAdmin4 подтверждает соответствие логической модели физической реализации [16].

## 2.4 Визуализация архитектуры и интерфейса приложения

Логические компоненты приложения размещаются на сервере приложений, базе данных и пользовательских устройствах в соответствии с выбранной архитектурой. Модули, отвечающие за обработку платежей и управление пользователями, развёрнуты на отдельном сервере. База данных размещена на специализированном сервере для обеспечения безопасности и надёжности хранения информации. Доступ пользователей к системе реализован через клиентские устройства. Конкретная схема размещения компонентов и их взаимодействия представлена на диаграмме развертывания (рисунок 8).

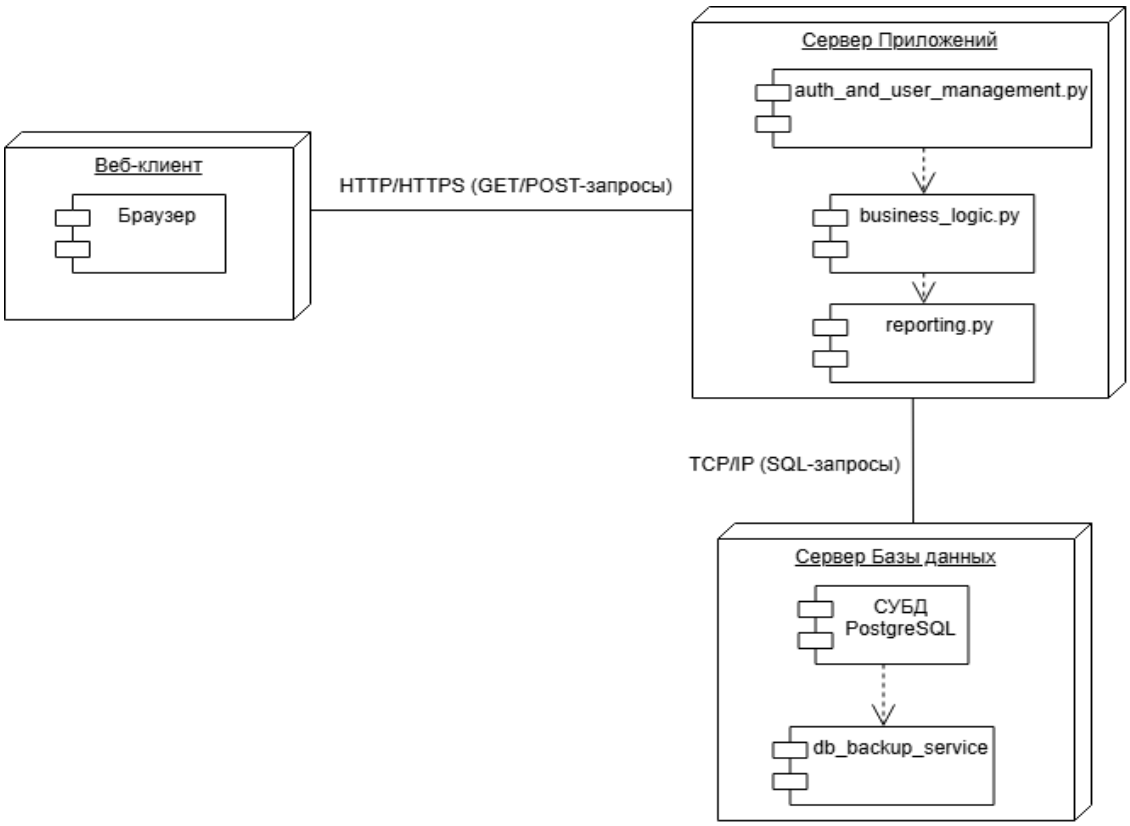


Рисунок 8 - Диаграмма развертывания компонентов

Компоненты:

- веб-клиент: работает через браузер;
- сервер приложений: реализован на Flask;
- база данных: PostgreSQL, подключение по TCP/IP;
- Mock API: имитация внешней банковской системы.

Каналы:

- HTTP/HTTPS (GET/POST-запросы): между клиентом и сервером;
- TCP/IP (SQLAlchemy ORM): между сервером и БД.

Для описания модульной структуры приложения используется диаграмма компонентов. Она демонстрирует основные логические блоки системы и их взаимодействие [8]. В частности, выделены модули, отвечающие за аутентификацию, обработку платежей, генерацию отчётов и интеграцию с банковскими API. Каждый компонент взаимодействует с другими через чётко определённые интерфейсы, обеспечивая гибкость и масштабируемость

архитектуры. Такое разделение упрощает сопровождение и развитие системы на последующих этапах. Схема модульной структуры представлена на рисунке 9.

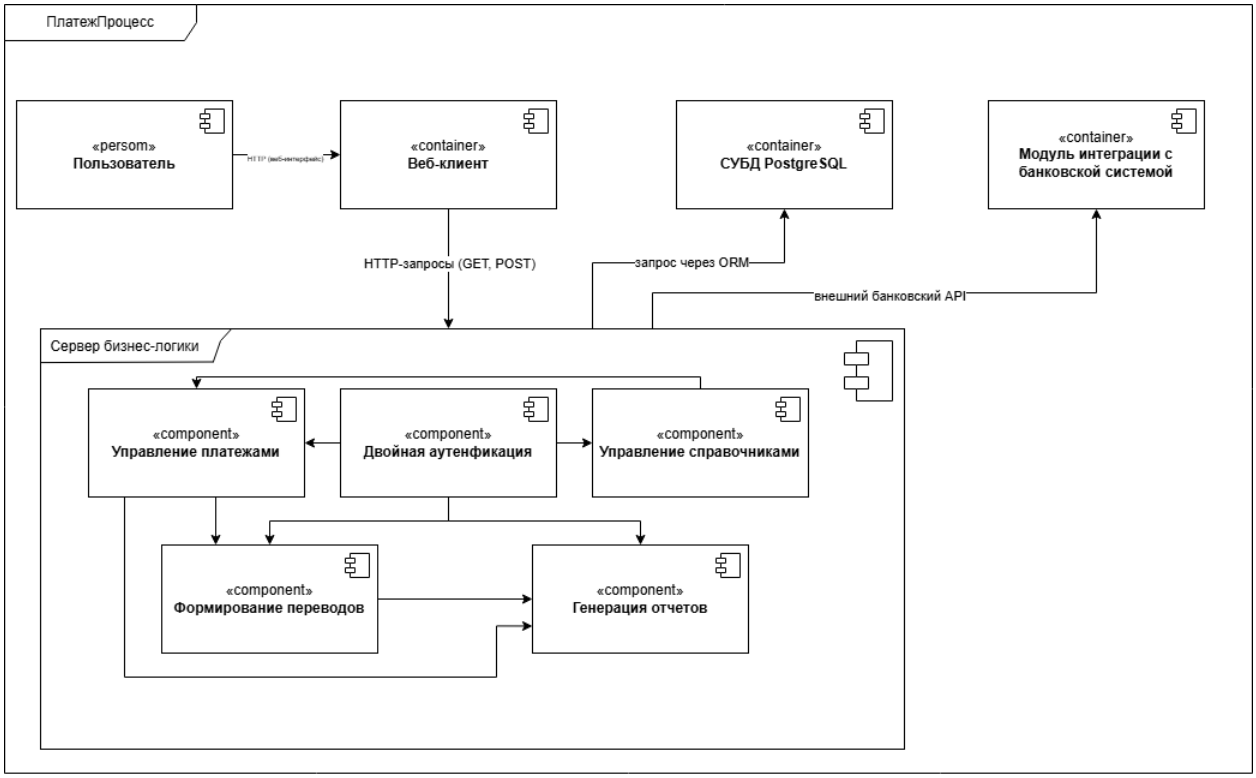


Рисунок 9 - Диаграмма компонентов приложения

Компоненты:

- аутентификация: обрабатывает вход, регистрацию, проверку кода [9];
- управление справочниками: добавление, редактирование и удаление поставщиков и потребителей;
- платежи: создание, проверка, статус;
- переводы: выбор поставщика, подбор платежей, отправка
- отчеты: генерация в PDF и CSV;
- импорт из банка: взаимодействие с эмуляцией API.

Все модули связаны с базой данных PostgreSQL через ORM SQLAlchemy [18, 12]. Взаимодействие между клиентом и сервером происходит по HTTP/HTTPS запросам (GET, POST) [6].

Функциональная структура веб-приложения отражена на диаграмме структуры приложения (рисунок 10), включающей все доступные пользователю разделы.

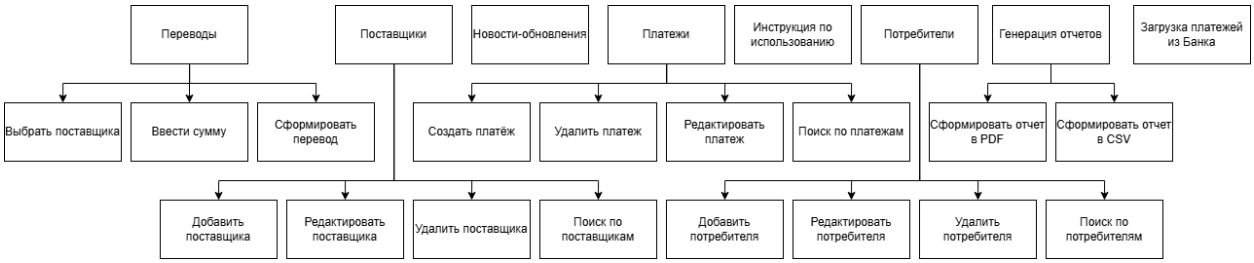


Рисунок 10 - Диаграмма структуры приложения

Разделы:

- поставщики и потребители: создание, редактирование, удаление, поиск;
- платежи: создание, редактирование, удаление, фильтрация;
- переводы: выбор поставщика, ввод суммы, формирование перевода;
- отчеты: формирование и экспорт PDF/CSV;
- загрузка из банка: получение данных из эмуляции API;
- инструкции и новости: вспомогательные, информативные модули.

Шаблоны реализованы на Jinja2, маршруты обрабатываются Flask. Пользователь взаимодействует через формы и навигационную панель [15].

Для оценки полноты реализованного функционала и визуального распределения модулей в системе представлен фрагмент главной страницы веб-приложения. Интерфейс разработан с учётом удобства навигации, акцента на ключевых функциях и соблюдения принципов адаптивности. В верхней части размещено основное меню для быстрого перехода к справочникам,

платежам, переводам, отчётам и настройкам. Дизайн выполнен в едином стиле с использованием контрастных элементов и интуитивно понятных обозначений. Представленная структура способствует быстрому обучению пользователя и снижает вероятность ошибок при взаимодействии. Визуальный пример показан на рисунке 11.

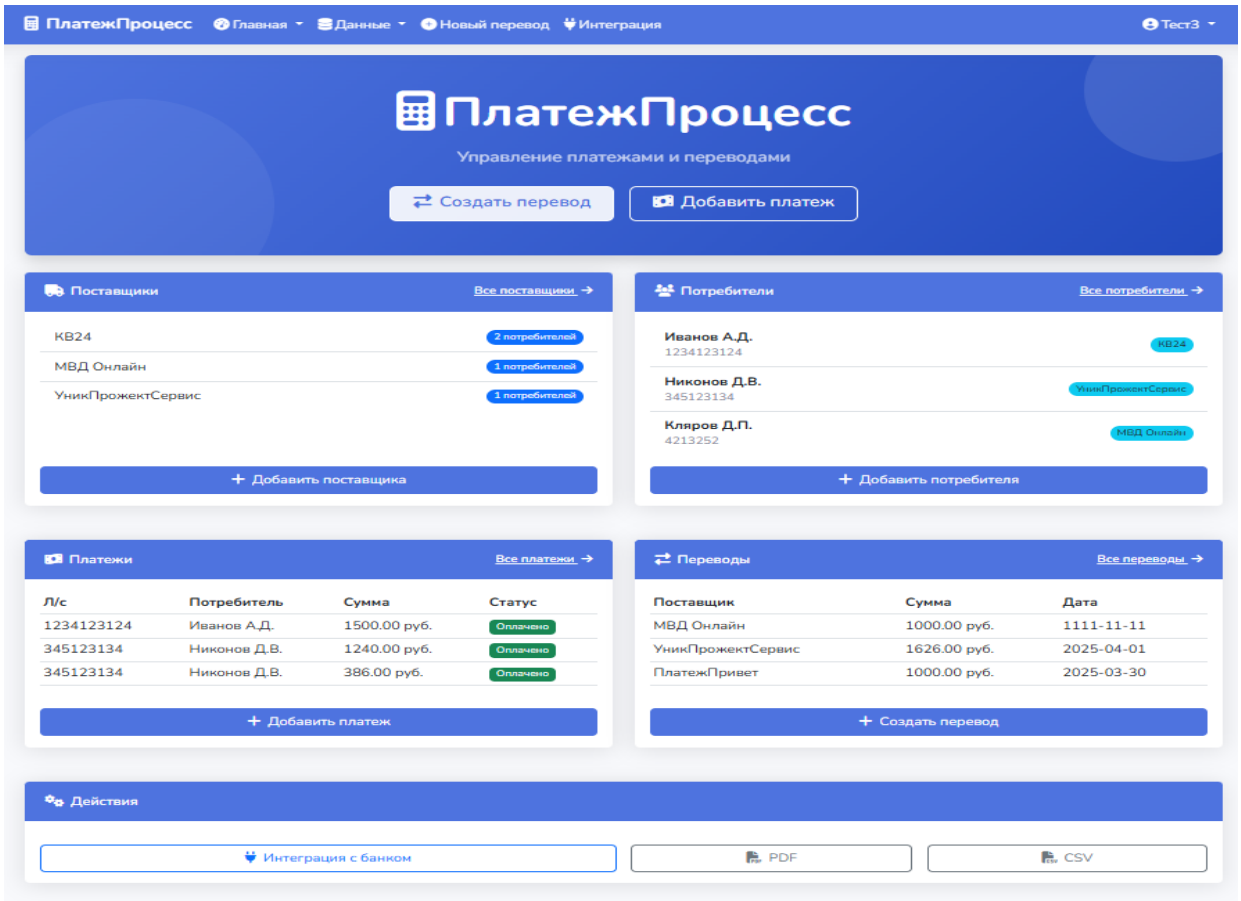


Рисунок 11 - Главная страница веб-приложения

На главной странице отображены основные разделы системы, сгруппированные в виде кнопок и ссылок, ведущих к модулям: поставщики, потребители, платежи, переводы, отчеты, инструкции и новости. Навигационные элементы обеспечивают быстрый доступ ко всем функциям, доступным авторизованному пользователю.

Интерфейс реализован на HTML с использованием Bootstrap, все шаблоны подключаются через Jinja2 [4]. Обработка маршрутов

осуществляется во Flask-приложении, где каждый раздел представлен соответствующим endpoint’ом.

Визуальное разделение интерфейса соответствует функциональной архитектуре, описанной на диаграмме структуры приложения, и подтверждает, что интерфейс охватывает весь реализованный функционал [20].

Вывод по главе 2

В процессе проектирования разработана структурная и поведенческая архитектура веб-приложения. На основе анализа предметной области была сформирована модель классов, определены основные компоненты и маршруты взаимодействия между ними. Представлены диаграммы деятельности и состояний, описывающие поведение пользователя и жизненный цикл ключевых объектов.

Физическая модель базы данных реализована в PostgreSQL и спроектирована с учетом требований нормализации, обеспечения связности и аудита. Интерфейс приложения охватывает весь функционал и построен на принципах доступности и модульности.

Выбор технологий обусловлен критериями надежности, гибкости и поддержки современных подходов к разработке веб-приложений. Полученная архитектура обеспечивает масштабируемость, логическую целостность и расширяемость при сопровождении проекта.

# Глава 3 Оценка эффективности разработанного программного обеспечения

## 3.1 Тестирование и отладка приложения

В целях проверки стабильности и корректности функционирования разработанного веб-приложения было проведено комплексное тестирование ключевых модулей системы. Проверке подвергались основные пользовательские сценарии, соответствующие требованиям, сформулированным на этапах анализа и проектирования.

Тестирование включало два этапа:

- функциональное тестирование: проверка корректности реализации логики операций;
- нагрузочное тестирование: оценка устойчивости системы при массовой обработке данных [7].

## 3.2 Функциональное тестирование

Цель данного этапа - проверка реализации основных функций системы, включая авторизацию, создание платежей, формирование переводов и генерацию отчетов. Методика испытаний заключалась в последовательном воспроизведении пользовательских сценариев с фиксацией результатов при помощи скриншотов и таблиц. Результаты фиксировались в таблице 4.

Таблица 4 - Результаты функционального тестирования

| Тестируемое действие | Ожидаемый результат | Фактический результат | Статус |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| Ввод корректного логина и пароля | Переход на главную страницу | Авторизация выполнена | Успешно |

Продолжение таблицы 4

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| Создание нового платежа | Платеж отображается в таблице | Платеж добавлен | Успешно |
| Попытка создать перевод без платежей | Отображается сообщение об отсутствии данных | Получено предупреждение | Успешно |
| Формирование отчета | Файл в формате PDF или CSV скачан | Файл успешно сгенерирован | Успешно |
| Редактирование поставщика | Изменения отображаются в таблице | Поставщик обновлен | Успешно |

Результат подтверждает корректную работу интерфейса, валидации и связи с базой данных. Все функции соответствуют ожидаемому поведению.

### 3.3 Нагрузочное тестирование

Цель данного этапа - определить устойчивость системы при массовом выполнении однотипных операций. С помощью автотестов и симуляции HTTP-запросов были выполнены множественные POST-запросы к маршрутам добавления платежей и генерации отчетов. Производительность замерялась по времени отклика и количеству успешных операций. Результаты фиксировались в таблице 5.

Таблица 5 - Результаты нагрузочного тестирования

| Сценарий | Объем операций | Среднее время отклика | Ошибки | Статус |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| Добавить 100 платежей подряд | 100 | 0.32 сек | Нет | Успешно |
| Генерация 50 отчетов | 50 | 0.45 сек | Нет | Успешно |
| Одновременные обращения к базе (10 потоков) | 10 потоков | 0.61 сек | Нет | Успешно |

На рисунке 12 представлен график, отражающий среднее время отклика при различных типах нагрузки.

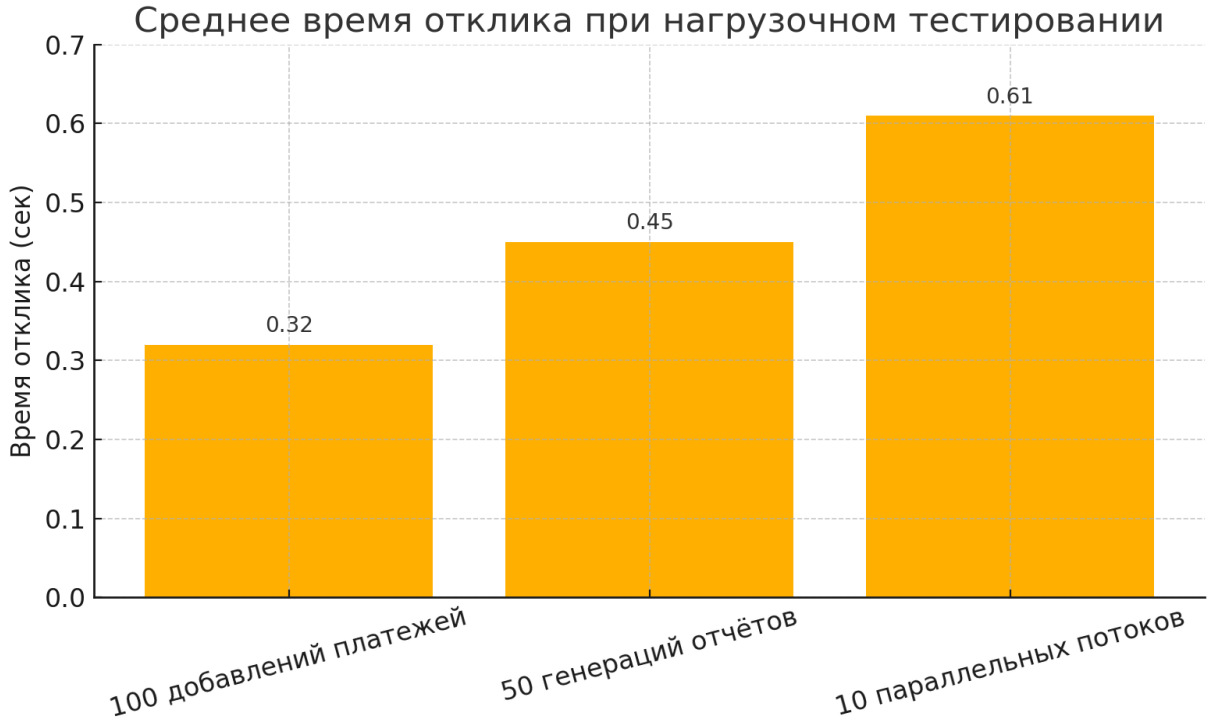


Рисунок 12 - График результатов при нагрузочном тестировании

Результаты испытаний показывают, что при параллельных запросах система сохраняет стабильность и приемлемое время отклика.

В итоге приложение сохраняет работоспособность при нагрузке, приближенной к реальному использованию. Ошибок соединения, потерь данных и сбоев обнаружено не было. Среднее время отклика не превышает 0.6 секунды даже при массовом выполнении операций, что подтверждает достаточную производительность [7].

### 3.4 Демонстрация функционала

Разработанное веб-приложение реализует весь заявленный в техническом задании и аналитических разделах функционал. На рисунках

ниже продемонстрирована работа ключевых интерфейсов, подтверждающие корректную работу модулей: авторизации, управления справочниками, обработки платежей, формирования переводов, генерации отчетов и имитации интеграции с внешними сервисами.

Одним из ключевых требований к приложению было обеспечение защищенного доступа только для сотрудников организации. Для этого реализована система регистрации с обязательным вводом секретного кода и авторизации с валидацией логина и пароля (рисунок 13).

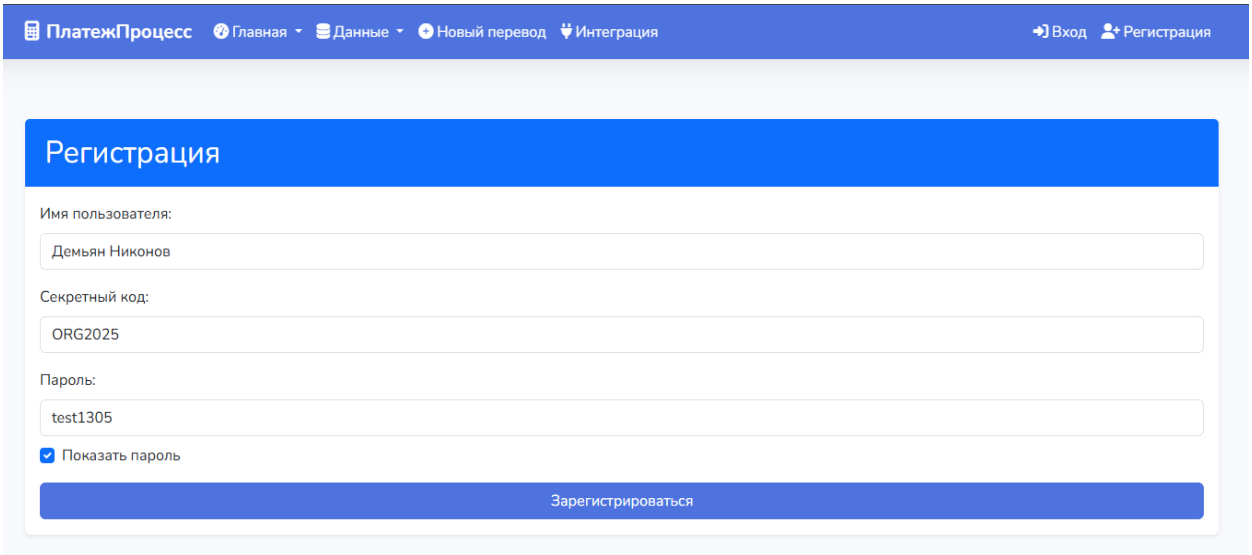


Рисунок 13 - Интерфейс регистрации нового пользователя

Форма регистрации включает поля для ввода имени пользователя, пароля и секретного кода, который известен только сотрудникам. Секретный код проверяется на серверной стороне, что исключает возможность регистрации постороннего лица. Реализована функция отображения пароля, упрощающая ввод. После успешного заполнения пользователь автоматически перенаправляется на форму входа (рисунок 14).

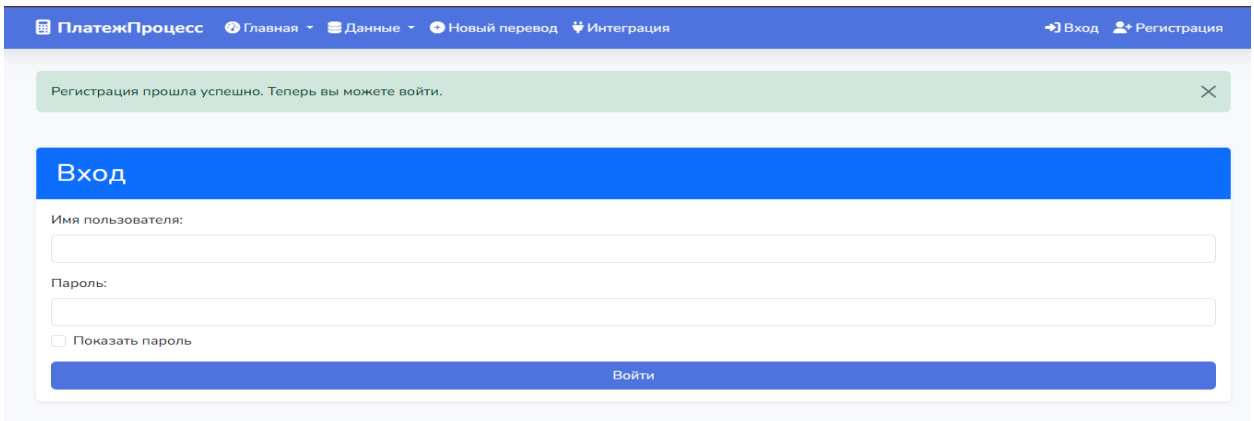


Рисунок 14 - Интерфейс регистрации нового пользователя

После успешной регистрации отображается уведомление об успешной операции. Пользователь может выполнить вход в систему, указав имя и пароль (рисунок 15). Предусмотрена возможность показа пароля. Каждое действие пользователя сопровождается уведомлениями, реализованными через flash-сообщения Flask.

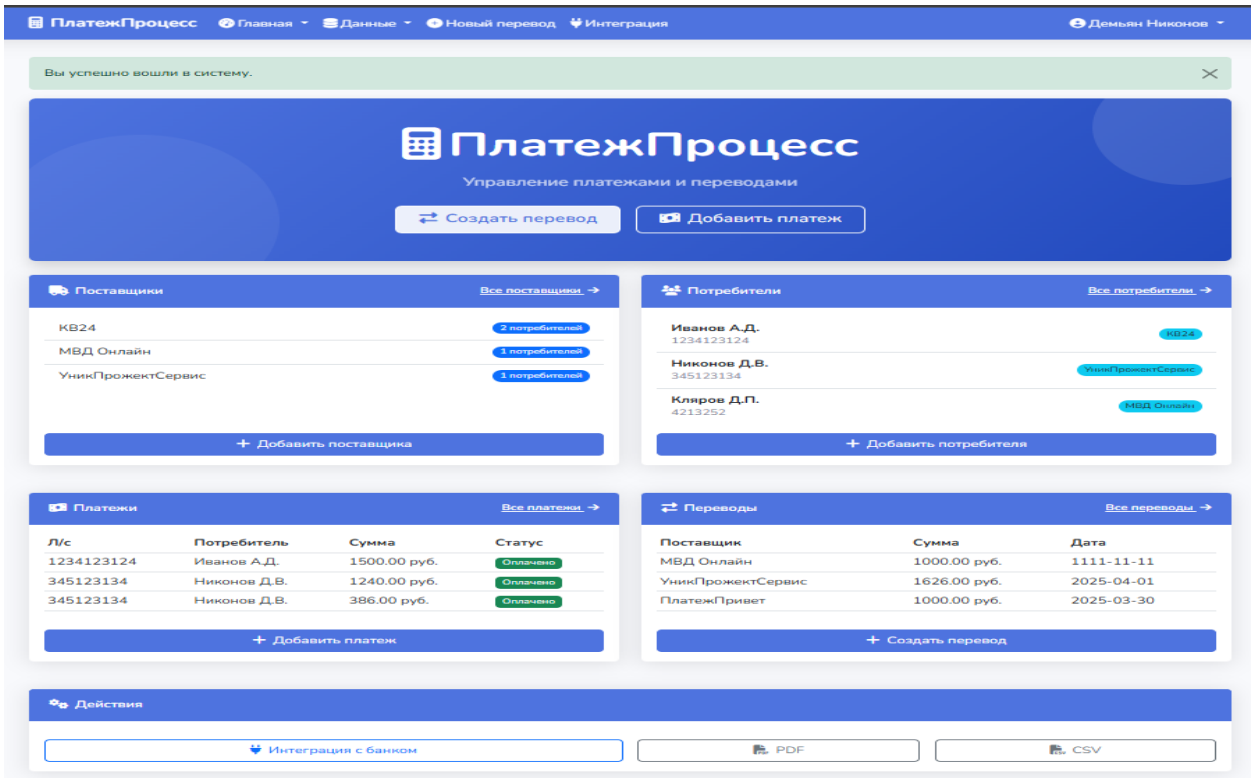


Рисунок 15 - Интерфейс регистрации нового пользователя

После входа пользователь попадает на главную страницу, где представлен краткий обзор основных сущностей: поставщики, потребители, платежи и переводы. Отображаются последние три записи и предоставлены кнопки для быстрого доступа к созданию новых объектов. Также реализованы блоки для запуска интеграции с банком и генерации отчетов в формате PDF или CSV.

Блок «Поставщики» предоставляет пользователю доступ к основным операциям над справочником поставщиков услуг. Функциональность охватывает просмотр текущих записей, добавление нового поставщика, редактирование данных и удаление (рисунок 16).

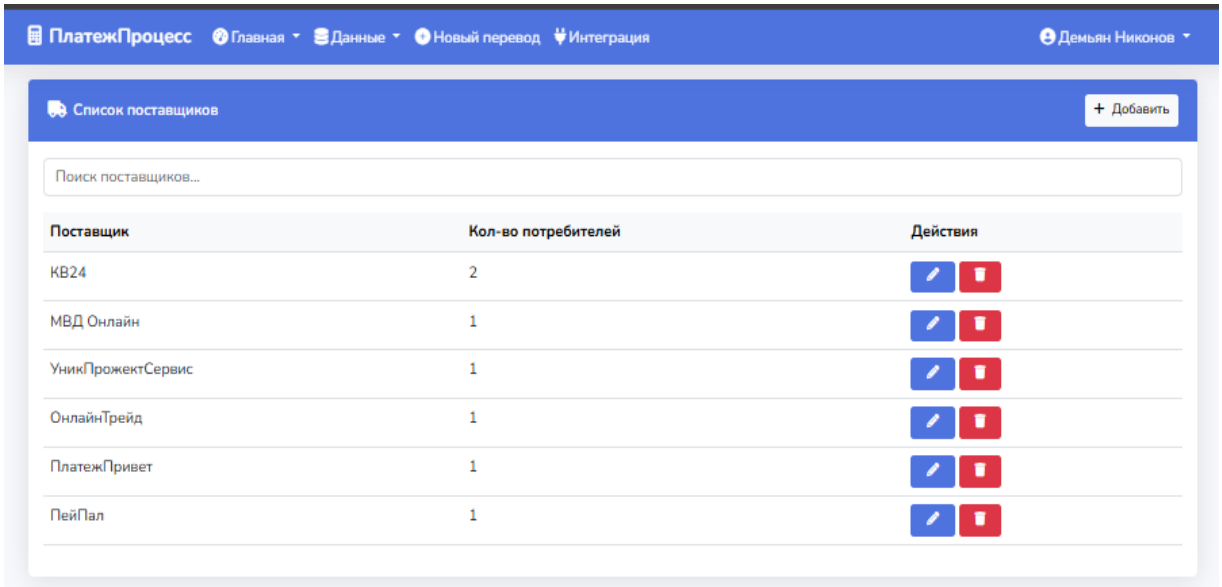


Рисунок 16 - Список поставщиков

На странице отображается таблица поставщиков с указанием количества привязанных к ним потребителей. Реализован поиск по названию, а также действия: редактирование и удаление. Кнопка «Добавить» инициирует переход к форме создания новой записи. Все действия мгновенно отражаются в таблице без необходимости перезагрузки страницы (рисунок 17).

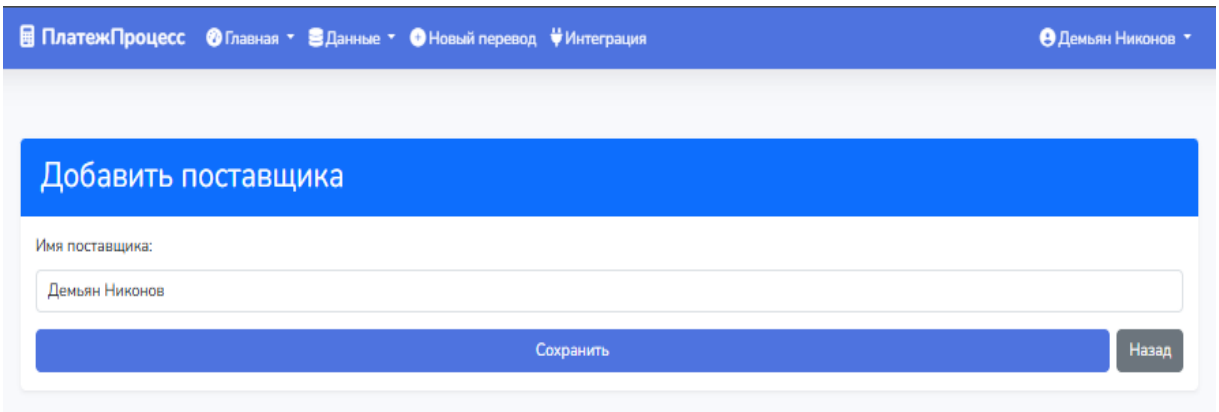


Рисунок 17 - Форма добавления поставщика

Форма добавления содержит поле ввода наименования поставщика и кнопку сохранения. При нажатии данные отправляются на сервер, где проходят валидацию и сохраняются в базу данных. После успешного добавления пользователь перенаправляется к списку поставщиков (рисунок 18).

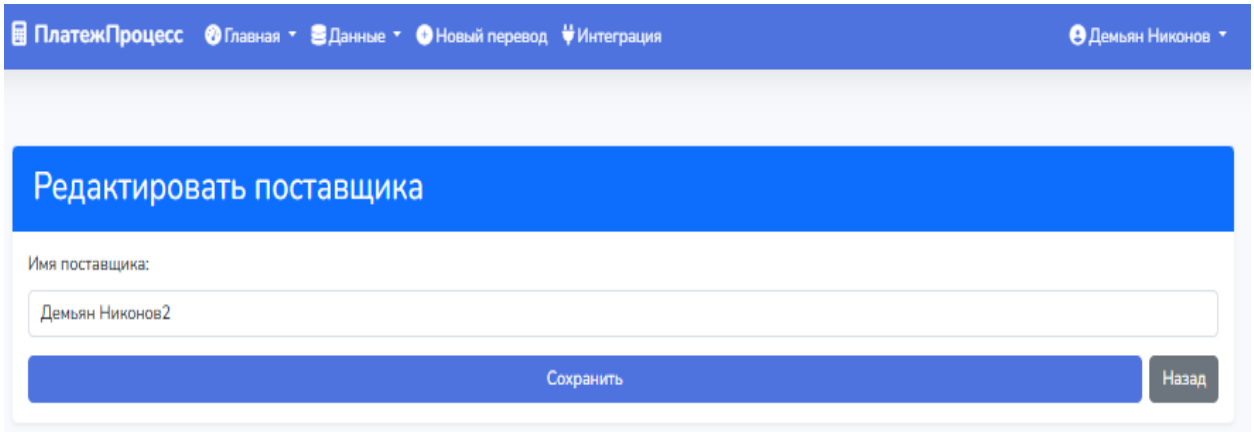


Рисунок 18 - Форма редактирования поставщика

Интерфейс редактирования аналогичен форме добавления. Пользователь может изменить имя поставщика, при этом сохраняется уникальность и привязка к потребителям. Все изменения сохраняются в базу данных с обновлением связей (рисунок 19).

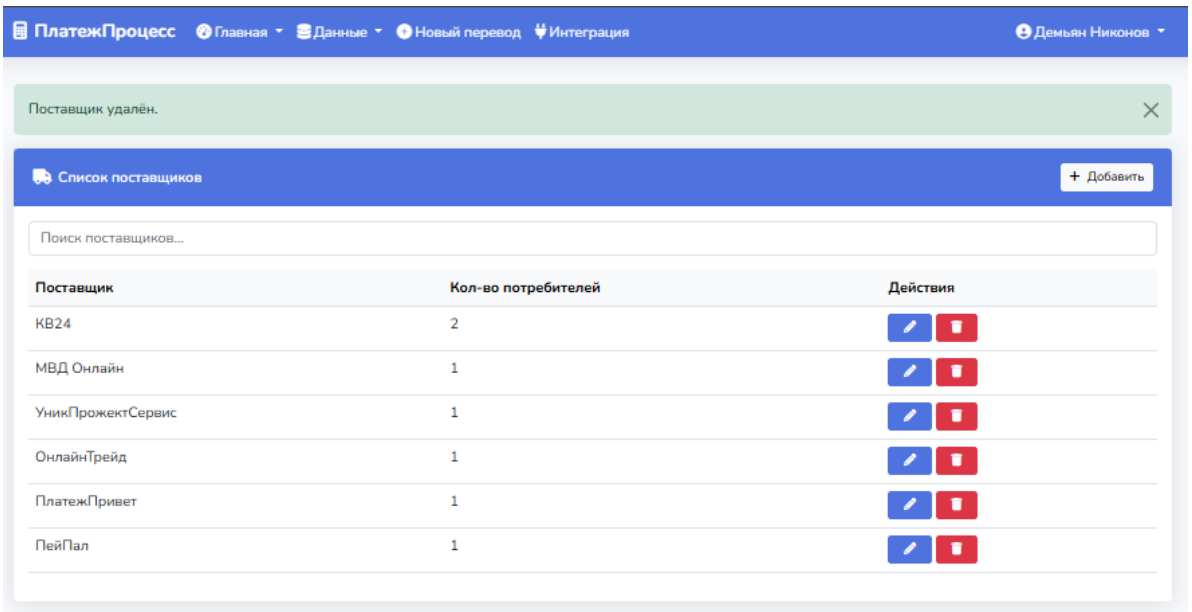


Рисунок 19 - Уведомление об удалении поставщика

После удаления поставщика отображается уведомление об успешном действии. В таблице обновленный список поставщиков. Система предотвращает удаление записей, к которым привязаны критически важные данные, благодаря валидации на уровне модели (рисунок 20).

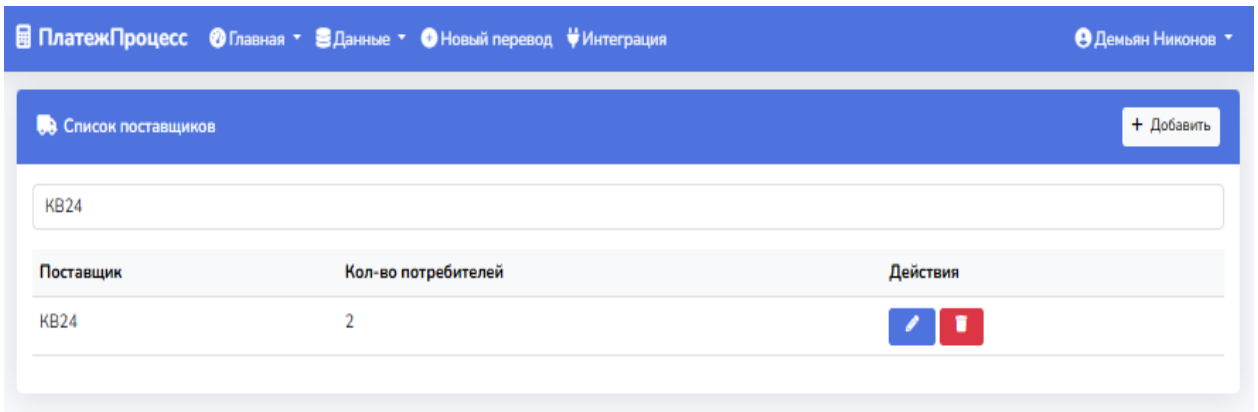


Рисунок 20 - Уведомление об удалении поставщика

Пользователь может выполнить фильтрацию по названию поставщика с помощью встроенной строки поиска. Система мгновенно отображает записи,

соответствующие введенному значению. В данном случае выполнен поиск по ключевому слову «КВ24». Это подтверждает корректную реализацию поиска в таблице и повышает удобство работы с большим объемом данных.

Блок «Потребители» позволяет регистрировать клиентов, оплачивающих услуги поставщиков. Для каждого потребителя указывается его имя, лицевой счет и поставщик, к которому он относится. Реализован полноценный CRUD-функционал с возможностью быстрого поиска (рисунок 21).

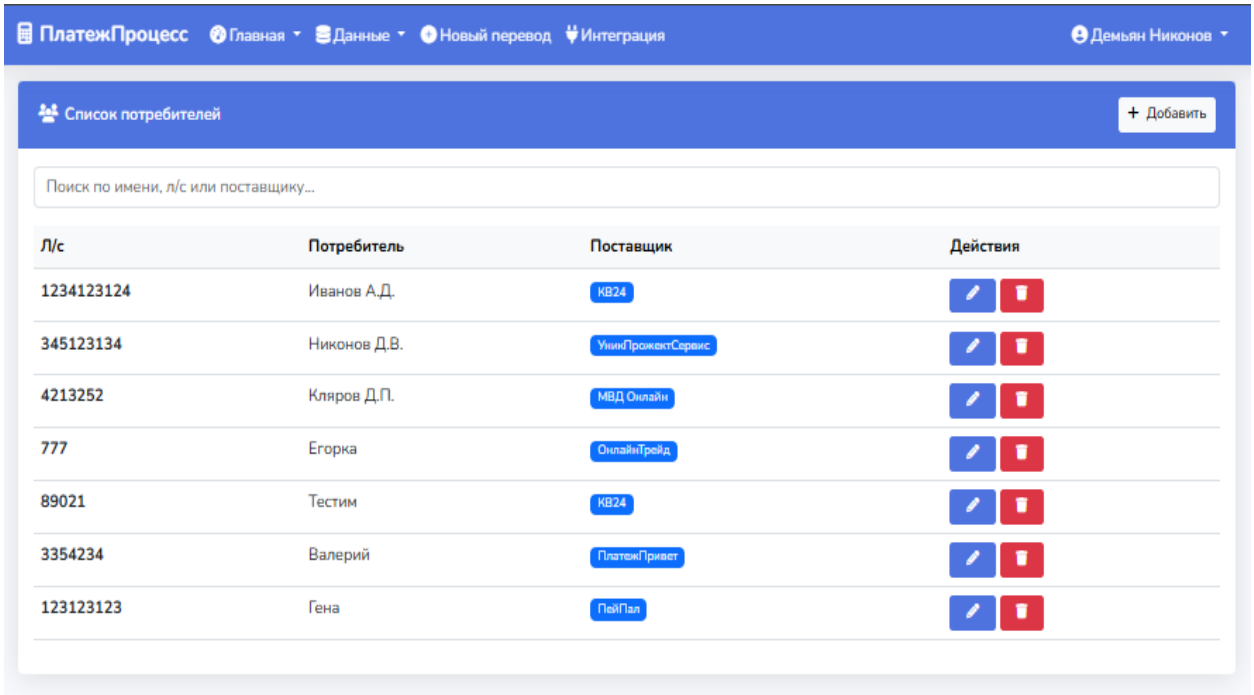


Рисунок 21 - Список потребителей с привязкой к поставщику

На экране представлен список всех потребителей, зарегистрированных в системе. В таблице отображаются данные: номер лицевого счета, ФИО, а также поставщик, к которому привязан потребитель. Доступны действия редактирования и удаления. Поисковая строка позволяет фильтровать записи по имени, л/с или названию поставщика (рисунок 22).

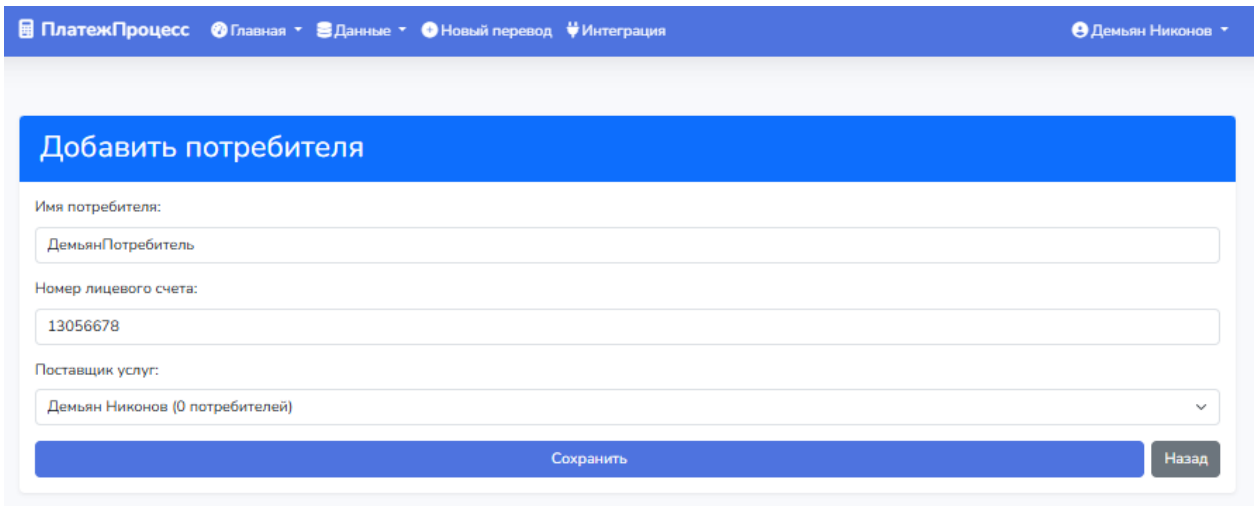


Рисунок 22 - Форма добавления потребителя

Форма добавления включает обязательные поля: имя потребителя, номер лицевого счета и выбор поставщика из выпадающего списка. Данные валидируются на стороне сервера и сохраняются в базу данных при нажатии кнопки «Сохранить» (рисунок 23).

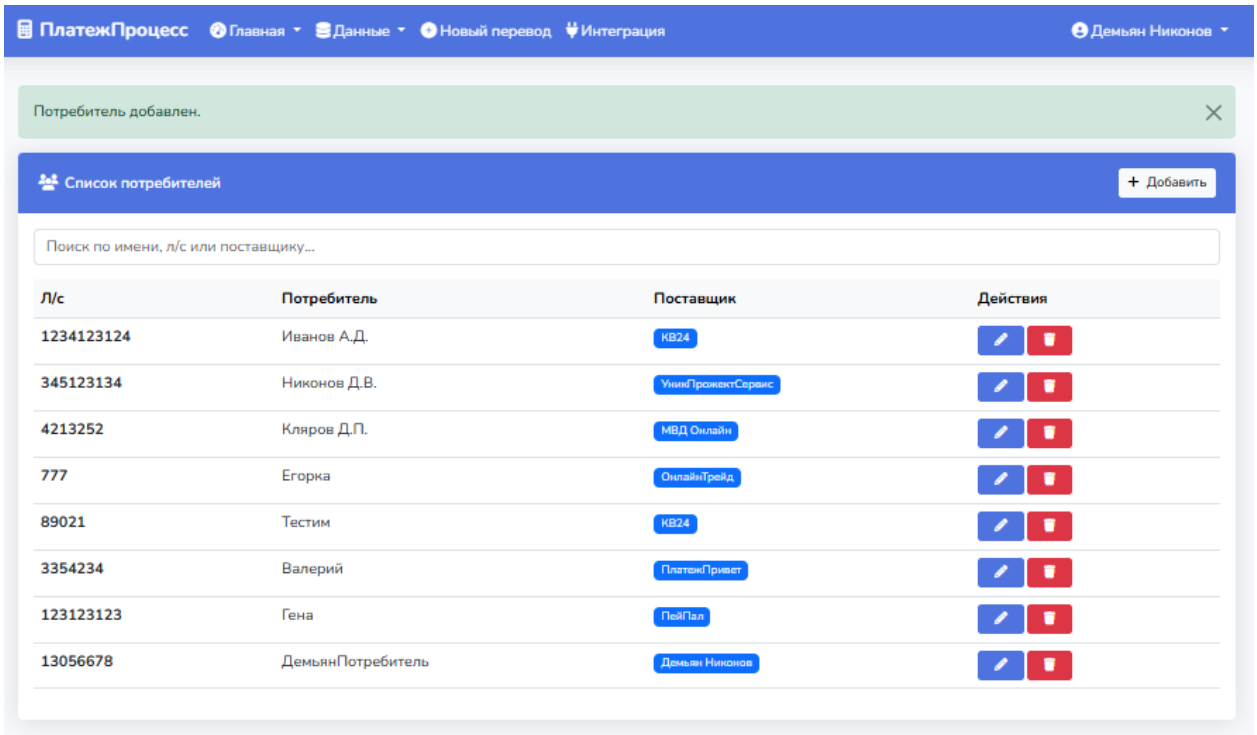


Рисунок 23 - Уведомление об успешном добавлении потребителя

После успешного добавления потребитель появляется в списке, а в верхней части интерфейса отображается подтверждение выполнения операции. Это подтверждает реализацию обратной связи пользователю и автоматическое обновление представления данных.

Блок «Платежи» предоставляет пользователю возможность регистрировать, просматривать и удалять сведения о совершенных оплатах за услуги, предоставляемые различными поставщиками. Платеж формируется на основе данных потребителя, его лицевого счета и привязки к поставщику. Система поддерживает фильтрацию по ключевым полям и визуальное отображение статуса (рисунок 24).

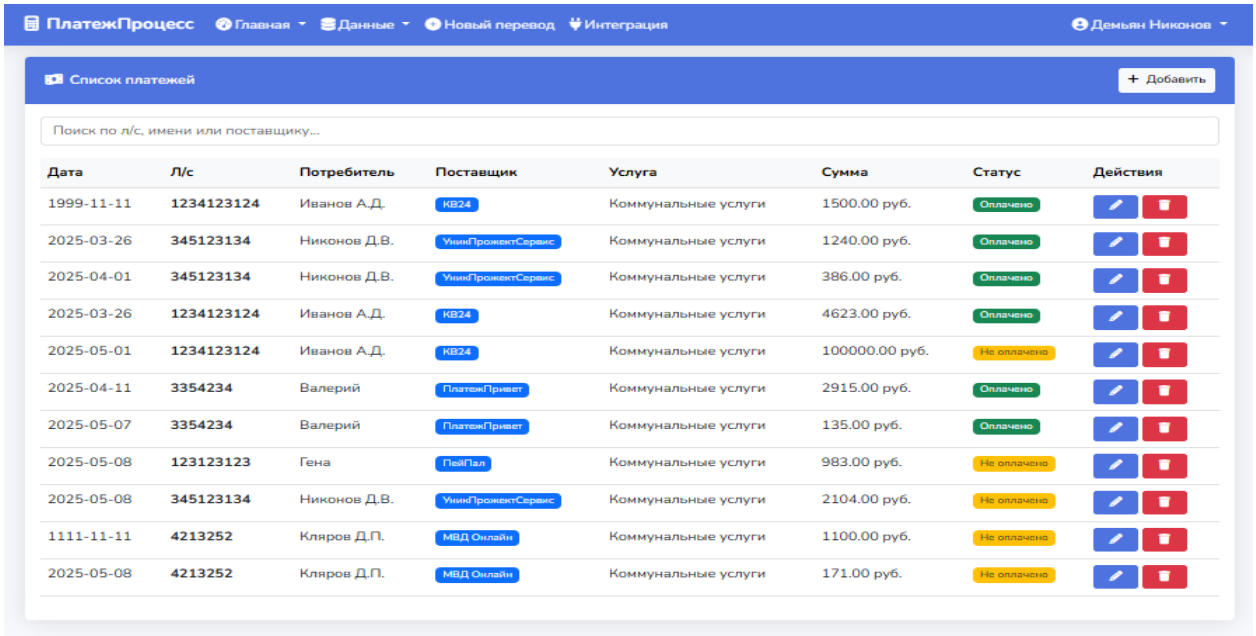


Рисунок 24 - Список зарегистрированных платежей

В таблице отображаются: дата совершения, лицевой счет, потребитель, поставщик, наименование услуги, сумма и статус оплаты. Для статуса реализованы цветовые маркеры: «Оплачено» (зеленый) и «Не оплачено» (желтый). Пользователь может отредактировать или удалить платеж, воспользовавшись соответствующими иконками. Над таблицей предусмотрен поиск по ФИО потребителя, л/с или поставщику (рисунок 25).

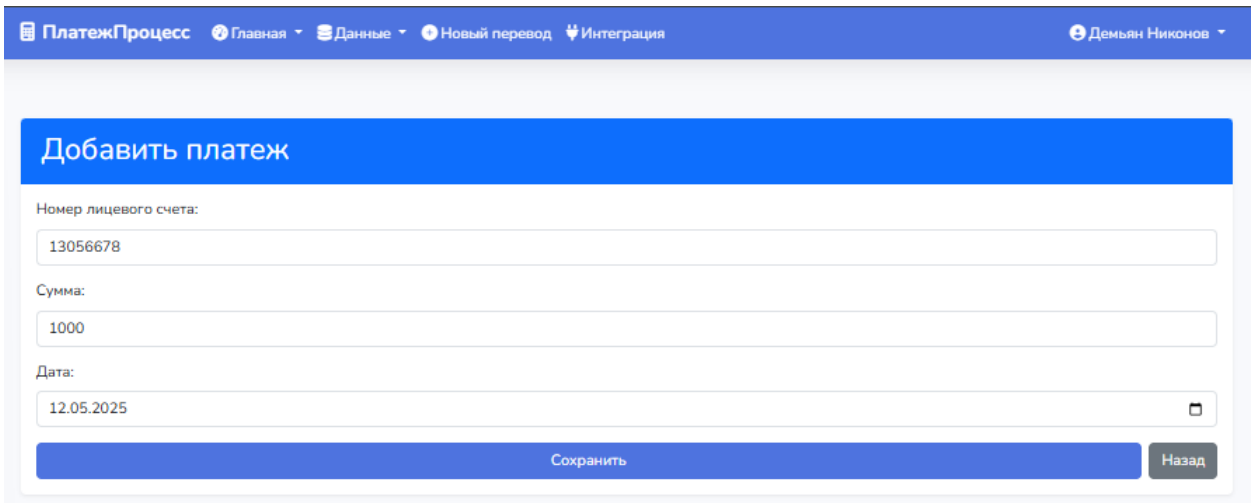


Рисунок 25 - Форма добавления нового платежа

Форма включает три обязательных поля: номер лицевого счета (с проверкой на существование потребителя), сумму платежа и дату. После нажатия кнопки «Сохранить» данные валидируются, и при успешном вводе новая запись добавляется в базу данных (рисунок 26).

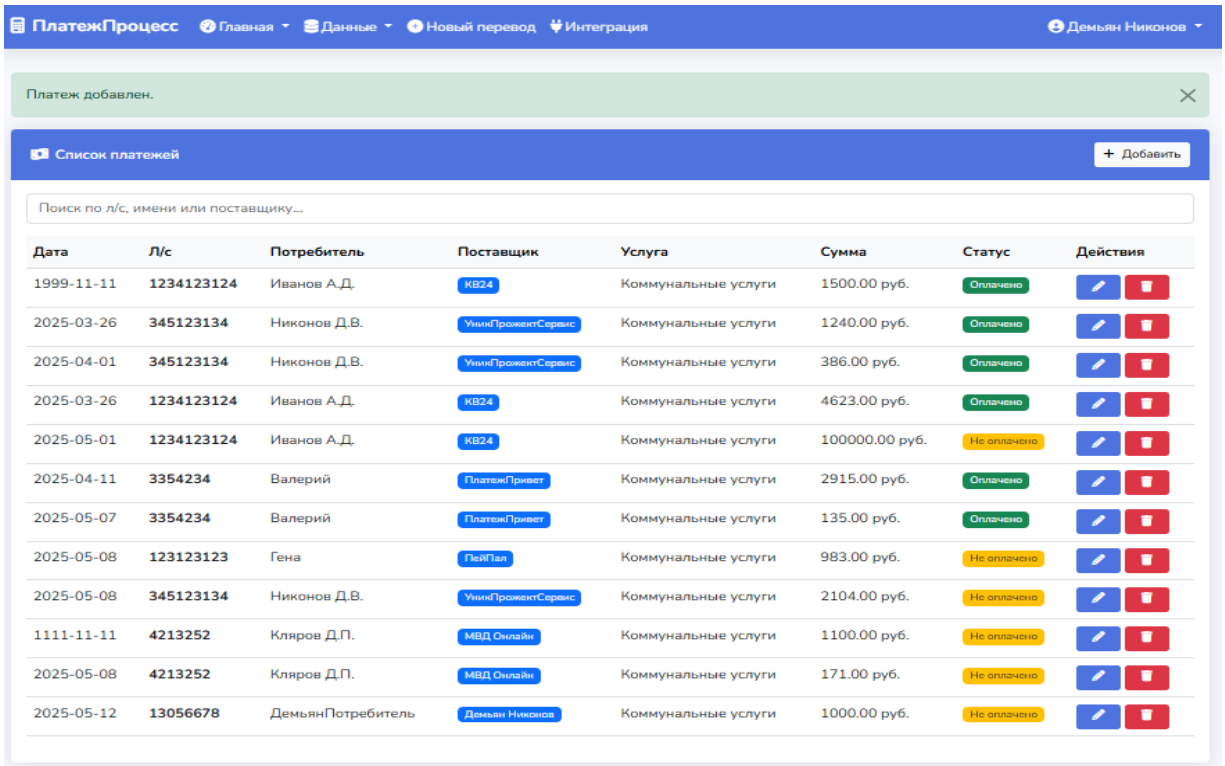


Рисунок 26 - Уведомление об успешном добавлении платежа

После сохранения пользователь возвращается к общему списку, где новый платеж появляется в верхней части таблицы. Система автоматически отображает уведомление об успешном добавлении, что подтверждает корректность выполнения операции.

Функциональный блок «Переводы» позволяет сотрудникам осуществлять распределение ранее внесённых платежей по поставщикам. Этот механизм реализует ключевую задачу системы - подбор платежей под заданную сумму перевода (рисунок 27).

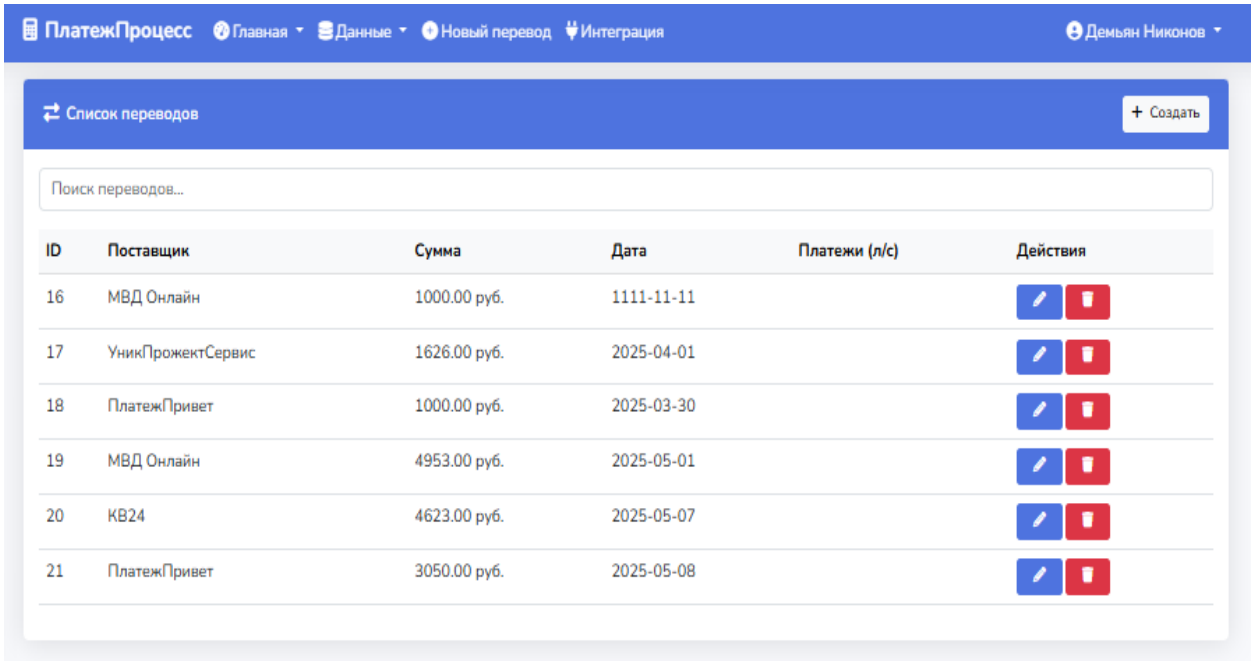


Рисунок 27 - Список переводов

На рисунке 27 показан список всех созданных переводов. Интерфейс содержит таблицу с ID перевода, поставщиком, суммой, датой и списком включенных платежей. Также реализован поиск и кнопки управления (редактирование, удаление) с возможностью добавления нового перевода (рисунок 28).

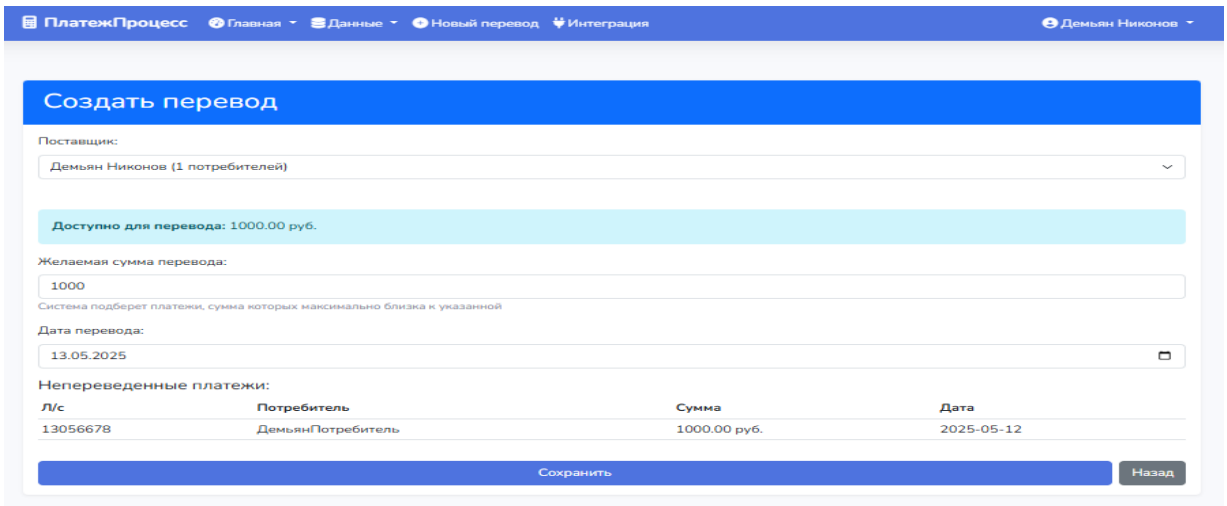


Рисунок 28 - Форма добавления перевода

При нажатии кнопки «Создать» открывается форма добавления нового перевода. Пользователь выбирает поставщика из выпадающего списка. Система автоматически рассчитывает доступную сумму по неоплаченным платежам данного поставщика. Далее вводится желаемая сумма перевода и дата. Ниже отображаются подходящие платежи, которые будут включены в перевод. После подтверждения система фиксирует новый перевод и обновляет статусы соответствующих платежей (рисунок 29).

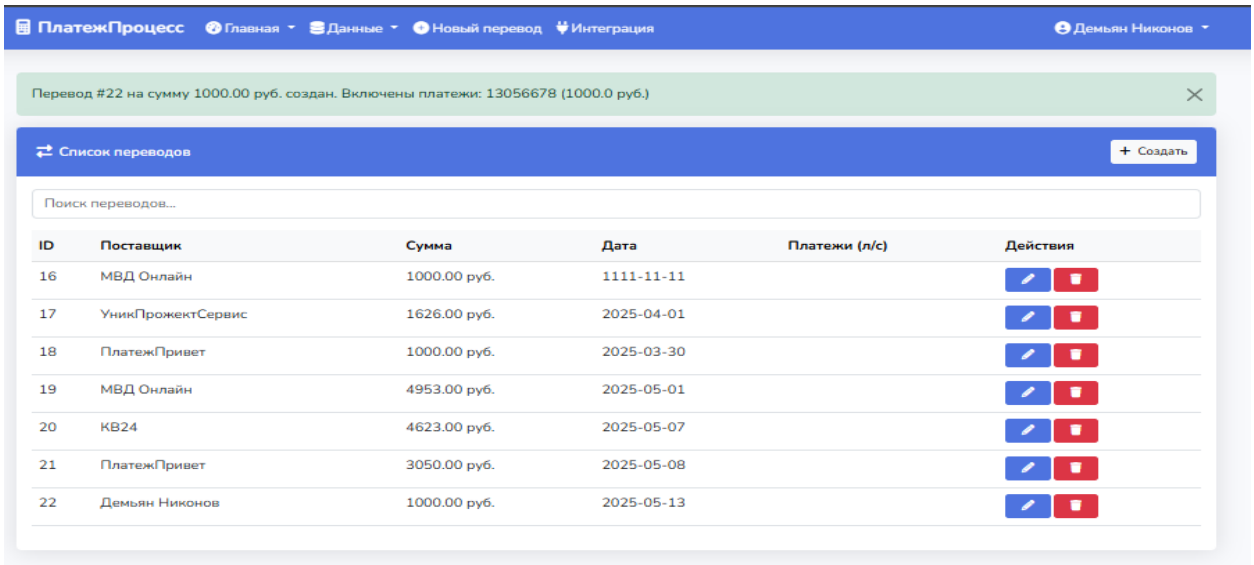


Рисунок 29 - Успешное создание нового перевода

Результатом успешного создания перевода является уведомление в верхней части страницы. В нем отображается ID перевода, сумма и список включенных платежей. Эта информация также дублируется в таблице (рисунок 30).

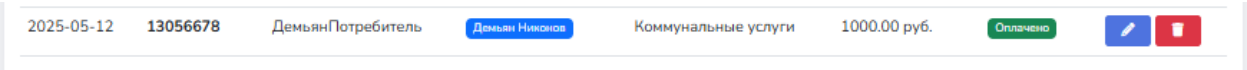


Рисунок 30 - Статус «Оплачено» для платежа, включенного в перевод

После создания перевода статусы включенных в него платежей автоматически обновляются. Представлен соответствующий фрагмент из раздела «Платежи». Платеж, участвовавший в переводе, получил статус «Оплачено», что визуально подтверждает его включение в трансакцию.

Указанные модули обеспечивают возможность имитации интеграции с внешним банком. Это позволяет пользователю загружать в систему новые платежи за последние 24 часа, автоматически сопоставляя их с существующими лицевыми счетами. Подобная возможность моделирует подключение к банковскому API и демонстрирует, как может функционировать импорт данных из внешних систем в дальнейшем. На рисунке 31 представлено окно «Интеграция с банком».

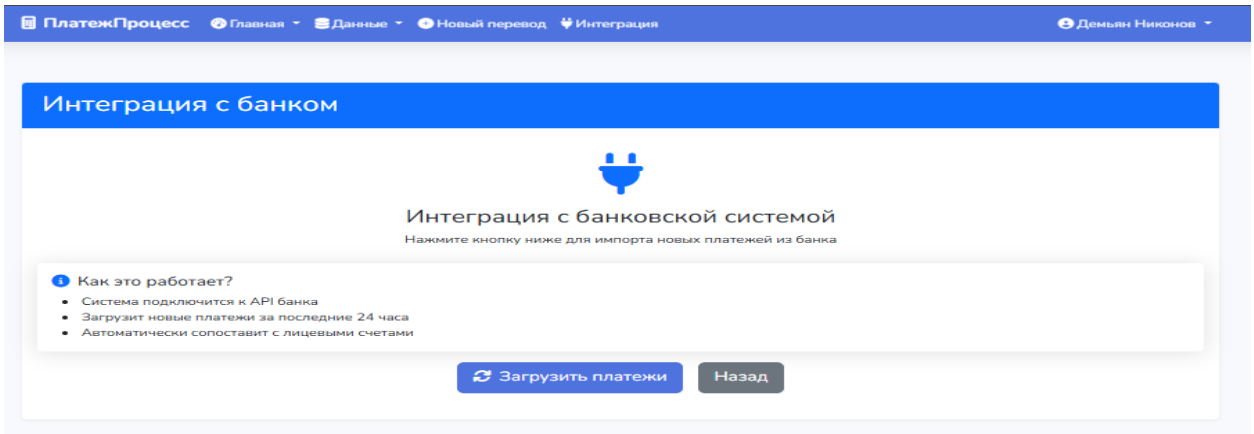


Рисунок 31 - Интерфейс модуля интеграции с банковской системой

Интерфейс содержит краткое описание механизма интеграции и кнопку «Загрузить платежи». При ее нажатии система имитирует подключение к банковскому API, загружает платежи, совершенные за последние сутки, и перенаправляет пользователя в раздел «Список платежей». Результат выполнения операции показан на рисунке 32.

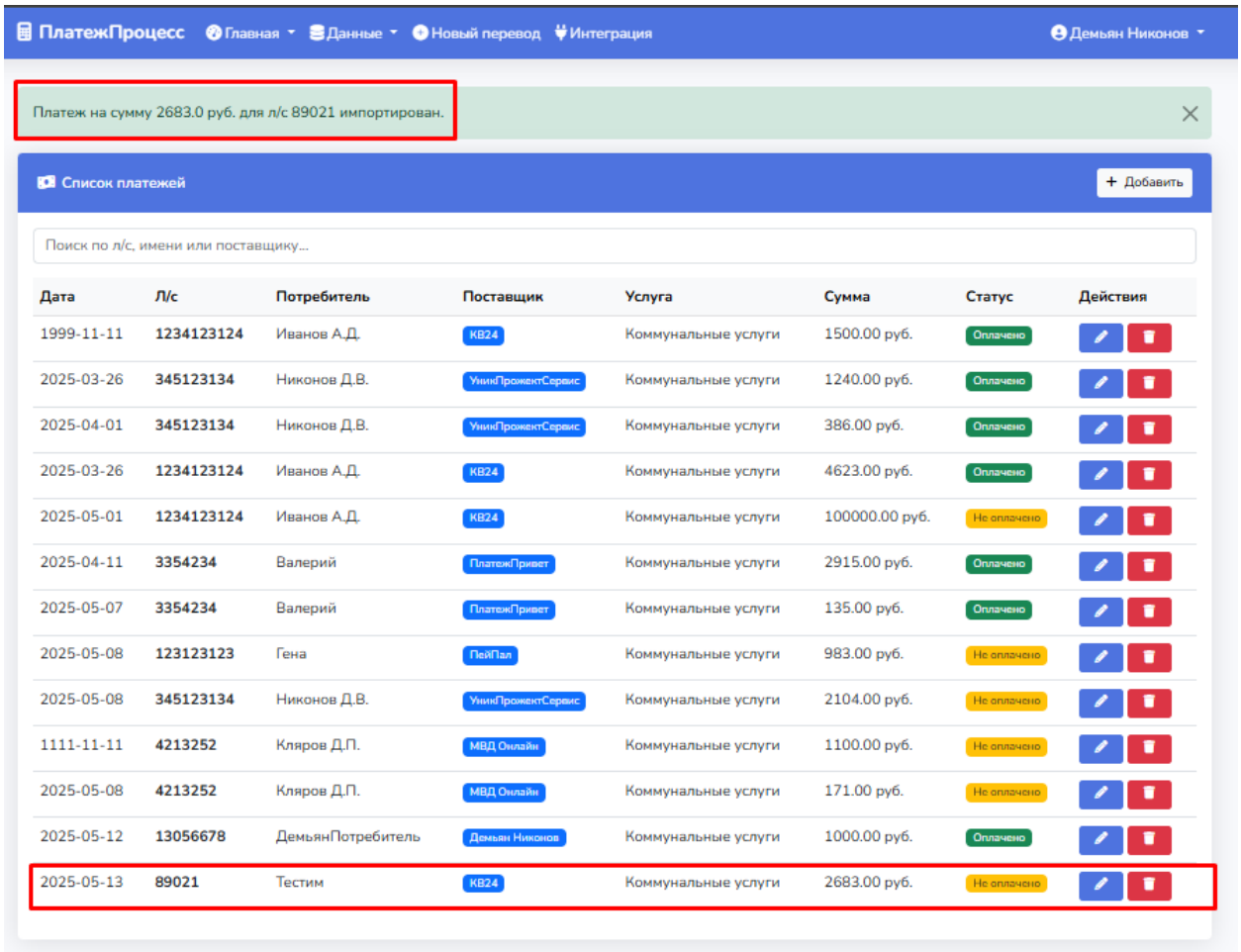


Рисунок 32 - Успешный импорт платежа из внешней системы

В верхней части экрана отображается уведомление, подтверждающее успешный импорт платежа. В таблице появляется новая запись с лицевым счетом, поставщиком, суммой и статусом. Реализованная функция позволяет имитировать процесс загрузки и автоматической обработки входящих данных.

Следующий блок - формирование отчета. Он предназначен для выгрузки собранных данных по всем зарегистрированным в системе платежам и

совершенным переводам. На рисунке 33 представлена панель действий на главной странице, где пользователь выбирает формат выгружаемого отчета.

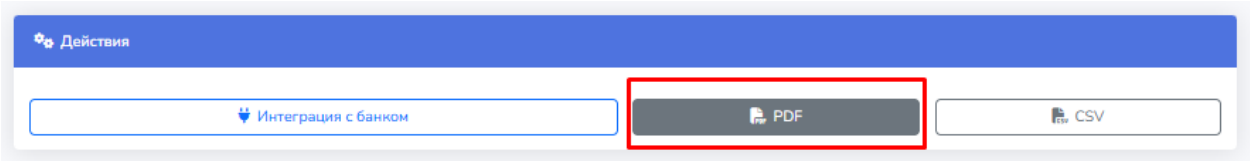


Рисунок 33 - Панель действий

Приложение предоставляет два варианта - PDF и CSV. В примере, представленном на рисунке 34, выбран формат PDF, позволяющий удобно визуализировать информацию о финансовых операциях.

**Отчет по платежам и переводам**

Дата и время формирования отчета: 2025-05-13 21:55:59

**Платежи**

| Л/с | Потребитель | Сумма | Дата | Услуга |
|---|---|---|---|---|
| 1234123124 | Иванов А.Д. | 1500.0 | 1999-11-11 | Коммунальные услуги |
| 345123134 | Никонов Д.В. | 1240.0 | 2025-03-26 | Коммунальные услуги |
| 345123134 | Никонов Д.В. | 386.0 | 2025-04-01 | Коммунальные услуги |
| 1234123124 | Иванов А.Д. | 4623.0 | 2025-03-26 | Коммунальные услуги |
| 1234123124 | Иванов А.Д. | 100000.0 | 2025-05-01 | Коммунальные услуги |
| 3354234 | Валерий | 2915.0 | 2025-04-11 | Коммунальные услуги |
| 3354234 | Валерий | 135.0 | 2025-05-07 | Коммунальные услуги |
| 123123123 | Гена | 983.0 | 2025-05-08 | Коммунальные услуги |
| 345123134 | Никонов Д.В. | 2104.0 | 2025-05-08 | Коммунальные услуги |
| 4213252 | Кляров Д.П. | 1100.0 | 1111-11-11 | Коммунальные услуги |
| 4213252 | Кляров Д.П. | 171.0 | 2025-05-08 | Коммунальные услуги |
| 13056678 | ДемьянПотребитель | 1000.0 | 2025-05-12 | Коммунальные услуги |
| 89021 | Тестим | 2683.0 | 2025-05-13 | Коммунальные услуги |

**Переводы**

| Поставщик | Сумма | Дата |
|---|---|---|
| МВД Онлайн | 1000.0 | 1111-11-11 |
| УникПрожектСервис | 1626.0 | 2025-04-01 |
| ПлатежПривет | 1000.0 | 2025-03-30 |
| МВД Онлайн | 4953.0 | 2025-05-01 |
| КВ24 | 4623.0 | 2025-05-07 |
| ПлатежПривет | 3050.0 | 2025-05-08 |
| Демьян Никонов | 1000.0 | 2025-05-13 |

Рисунок 34 - Сформированный отчет в формате PDF

После выбора формата файл автоматически генерируется и загружается на устройство. На рисунке представлен сформированный отчет по платежам и переводам в формате PDF. В верхней части отчета указывается дата и время формирования документа, а ниже - детализированная таблица с платежами (указаны ЛС, потребитель, сумма, дата, наименование услуги) и таблица переводов (поставщик, сумма, дата перевода). Формируемый файл обеспечивает полный охват финансовой активности пользователей и позволяет экспортировать данные в отчетную документацию.

Вывод по главе 3.

В результате проведенного тестирования и демонстрации работы системы установлено, что разработанное веб-приложение стабильно функционирует в соответствии с заявленными требованиями. Выполненные функциональные и нагрузочные тесты подтвердили корректность бизнес-логики, устойчивость при массовом вводе данных и отсутствие критических ошибок.

Интерфейс реализован с учетом удобства и логичности: каждая операция сопровождается уведомлением о результате, пользователь получает понятную обратную связь. Все модули - от регистрации до генерации отчетов - успешно прошли проверку и задействованы в пользовательском сценарии.

Приложение готово к эксплуатации в условиях реальной организации и полностью реализует поставленные цели автоматизации учета платежей и переводов.

## Заключение

В процессе выполнения выпускной квалификационной работы было спроектировано, реализовано и протестировано веб-приложение для управления платежами и переводами между потребителями и поставщиками услуг.

В ходе работы достигнуты следующие результаты:

– проанализирована предметная область и выявлены проблемы предприятия «Квартплата24» в части ручной обработки платежей и отсутствия централизованной системы учёта;

– изучены и сопоставлены аналогичные программные решения, выявлены их ограничения в части гибкости, настройки и функциональности;

– сформулированы функциональные и нефункциональные требования к системе, включая модель FURPS+;

– спроектирована архитектура приложения: определена структура классов, компонентов, поведение объектов и структура базы данных;

– реализовано веб-приложение с функциями: авторизация сотрудников, ввод и проверка платежей, редактирование справочников, формирование переводов, генерация отчетов и эмуляция банковского API;

– проведено функциональное и нагрузочное тестирование, подтверждающее стабильную и корректную работу приложения.

Полученные результаты:

– разработано надёжное веб-приложение с защищённой авторизацией, соответствующее задачам внутреннего документооборота предприятия;

– приложение позволяет вводить, хранить и обрабатывать платежные данные, формировать переводы и отчёты, исключая ошибки и упрощая работу сотрудников;

– обеспечено логически прозрачное поведение объектов, контроль связей между сущностями и стабильность при высоких нагрузках;

– интерфейс спроектирован с акцентом на удобство, минимизацию ручного ввода и быструю навигацию по модулям.

Практическая значимость:

– приложение может быть внедрено в рамках предприятия «Квартплата24» для использования сотрудниками финансового отдела;

– архитектура проекта допускает масштабирование: добавление новых типов операций, ролей пользователей, интеграцию с реальными банковскими API;

– использование современных технологий (Python, Flask, PostgreSQL, SQLAlchemy) обеспечивает гибкость, расширяемость и лёгкость поддержки.

Результатом работы стало полноценное веб-приложение, соответствующее требованиям надёжности, масштабируемости и удобства использования. Архитектура системы предполагает возможность дальнейшего расширения: реализацию ролей пользователей, подключение внешних API и расширение отчётности.

Система готова к применению в среде малого предприятия и может быть адаптирована для более широкого круга организаций, осуществляющих расчётные операции между потребителями и поставщиками услуг [10].

## Список используемой литературы и источников


1. Афанасьев А. В. Современный Python. Практика программирования / А. В. Афанасьев. — СПб. : Питер, 2022. — 368 с.
2. Беляев С. А. Flask и REST API: создание серверной логики / С. А. Беляев. — М. : Наука, 2022. — 320 с.
3. Богачев И. В. Практика работы с PostgreSQL / И. В. Богачев. — СПб. : БХВ-Петербург, 2020. — 288 с.
4. Гордон А. С. HTML и CSS: разработка и дизайн веб-сайтов / А. С. Гордон. — М. : Диалектика, 2019. — 360 с.
5. Мельников В. Ю. Проектирование информационных систем / В. Ю. Мельников. — М. : Юрайт, 2023. — 416 с.
6. Кузнецов Д. А. Безопасность веб-приложений / Д. А. Кузнецов. — М. : Бином, 2022. — 272 с.
7. Рябинин А. И. Тестирование программного обеспечения / А. И. Рябинин. — СПб. : Питер, 2021. — 384 с.
8. Соловьёв В. Г. Разработка клиент-серверных приложений / В. Г. Соловьёв. — М. : Академия, 2020. — 432 с.
9. Федеральный закон РФ от 27.07.2006 № 152-ФЗ «О персональных данных» (в ред. от 29.12.2024).
10. ГОСТ Р 7.0.5–2008. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая ссылка. Общие требования и правила составления.
11. Grinberg M. Flask Web Development: Developing Web Applications with Python / M. Grinberg. — Sebastopol : O’Reilly Media, 2018. — 258 p.
12. Jones K. Building RESTful Python Web Services / K. Jones. — Birmingham : Packt Publishing, 2019. — 320 p.
13. Ceder R. The Python Workshop / R. Ceder. — Birmingham : Packt Publishing, 2020. — 608 p.

14. Freeman E., Robson E. Head First Design Patterns / E. Freeman, E. Robson. — Sebastopol : O'Reilly Media, 2020. — 694 p.

15. Lutz M. Learning Python / M. Lutz. — 5th ed. — Sebastopol : O'Reilly Media, 2013. — 1648 p.

16. Kalin M. JavaScript and REST APIs: The Essential Guide / M. Kalin. — New York : Apress, 2021. — 280 p.

17. PostgreSQL Global Development Group. PostgreSQL Documentation [Electronic resource]. URL: https://www.postgresql.org/docs/ (accessed: 20.05.2025)

18. Lippert T. Pro SQLAlchemy / T. Lippert. — Berkeley : Apress, 2022. — 330 p

19. Matthes E. Python Crash Course / E. Matthes. — 3rd ed. — San Francisco : No Starch Press, 2023. — 544 p.

20. Williams B. Responsive Web Design with HTML5 and CSS / B. Williams. — 3rd ed. — Birmingham : Packt Publishing, 2021. — 630 p.