МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет компьютерных наук

Кафедра информационных систем в телекоммуникациях

Система онлайн платежей «Payment-ae»

Курсовой проект

09.03.02 Информационные системы и технологии

Информационные системы в телекоммуникациях

Руководитель_____В.С. Тарасов, ст. преподаватель__.__20__

Обучающийся_____Д.А. Загреба, 3 курс, д/о

Обучающийся_____Н.А. Ягодинцев, 3 курс, д/о

Обучающийся_____К.Р. Ходжаев, 3 курс, д/о

Обучающийся_____Я. Сулиман, 3 курс, д/о

Воронеж 2023

Содержание

Введение

В настоящее время Интернет-технологии активно используются в различных сферах, в том числе и в системах онлайн платежей.

В наше время, когда электронная коммерция с каждым днём занимает всё более прочную позицию в мировой экономике, необратимо встаёт вопрос о том, как принимать платежи покупки онлайн. В этой области отличным помощником становятся платежные системы.

В данной курсовой работе рассмотрена разработка веб-приложения онлайн кассы с детальной проработкой одного из способов оплаты.

Основная цель проекта заключается в создании функциональной системы, которая позволит пользователям легко и быстро начать принимать платежи на своем сайте.

В работе будет рассмотрен процесс проектирования и разработки веб-приложения, включая выбор подходящих технологий и инструментов, создание базы данных, разработку интерфейса и реализацию основных функций приложения. Кроме того, будут рассмотрены вопросы безопасности и защиты данных пользователей.

1 Постановка задач

Целью данного проекта является создание веб-приложения онлайн кассы.

Основными задачами проекта являются реализация услуг по приёму платежей от покупателей с последующим перечислением полученных денег по указанным продавцом реквизитам.

Для достижения поставленных целей необходимо, во–первых, иметь представление о разрабатываемой системе, представленное необходимыми UML–диаграммами и разработанным дизайном веб-приложения, как в целом, так и в отдельных сценариях.

Во–вторых, необходимо реализовать базы данных, которые будут хранить информацию о магазинах клиентов и об осуществленных покупках.

Также, необходимо провести тестирование системы.

1.1 Требования к разрабатываемой системе

1.1.1 Функциональные требования

К разрабатываемому приложению выдвинуты следующие требования:

- разделение пользователей на: клиентов, мерчантов, администраторов и модераторов;
- клиент должен иметь возможность выбора способа оплаты, которым он планирует осуществлять оплату;
- мерчант должен иметь возможность создавать новые магазины, следить за операциями, проходящими через него, отслеживать суммы и статусы каждой операции;
- модератор должен иметь возможность отслеживать все операции и обрабатывать их в случае их нарушения, так же он должен иметь возможность пропускать или отменять операции в случае подозрительных действий и блокировать магазин мерчанта в случае нарушения правил площадки;

— администратор должен иметь возможность следить за всеми проходящими внутри процессами и управлять ими.

1.1.1.1 Для мерчантов

— Обеспечение возможности подключать свой магазин к платформе;
— обеспечение возможности отслеживания переводов денежных средств.

1.1.1.2 Для модераторов

— Обеспечение возможности просматривать подробную информацию о конкретной заявке;
— обеспечение возможности вывода денежных средств из системы;
— в случае если у мерчанта на сайте предусмотрен вывод средств, отслеживать запросы на выводы средств и следить за их достоверностью.

1.1.1.3 Для администраторов

— Обеспечение возможности просматривать подробную информацию (все транзакции по каждому «мерчанту»);
— обеспечение возможности просматривать успешные переводы, отслеживать статистику.

1.1.2 Технические требования

Программный продукт должен обеспечить:

— авторизацию пользователей с использованием почты и пароля;
— шифрование пароля при записи в БД;
— блокировка карт на время перевода средств одним из клиентов;
— балансировка среднего баланса на каждой карте;
— использование отдельного микросервиса для обработки REST API запросов на Go;

— панель управления на Laravel;

— интерфейс пользователя на React;

— хранение данных в БД.

1.2 Требования к интерфейсу

Интерфейс должен быть выполнен в единой для всех экранов цветовой гамме, едином стиле. Все надписи должны быть легко читаемы, все элементы управления должны быть выполнены в едином стиле, размере, должны выделяться на фоне содержимого экранов.

Интерфейс должен содержать только необходимую для пользователя информацию. Информация должна находиться в тех местах приложения, где она будет актуальна. Основные элементы управления должны быть заметны для пользователя.

1.3 Задачи, решаемые в процессе разработки

Перед проектом были поставлены следующие задачи:

— анализ предметной области;

— анализ аналогов;

— написание технического задания;

— описание разрабатываемой системы UML диаграммами;

— разработка БД;

— реализация ролей;

— реализация функциональных возможностей ролей;

— разработка функциональных возможностей сайта;

— создание макета дизайна и его реализация;

— реализация интерфейса;

— проведение тестирования;

— описание процесса разработки и результата.

2 Анализ предметной области

2.1 Терминология (глоссарий) предметной области

Проект, система – разрабатываемое веб-приложение.

Frontend – клиентская часть приложения. Отвечает за получение информации с программно-аппаратной части и отображение ее на устройстве пользователя.

Клиентская сторона – компьютер, использующий ресурсы сервера и предоставляющий пользователю возможность взаимодействия с системой.

Backend – программно-аппаратная часть приложения. Отвечает за функционирование внутренней части приложения.

Сервер, серверная часть – компьютер, обслуживающий другие компьютеры (клиентов) и предоставляющий им свои ресурсы для выполнения определенных задач.

GitHub – веб-сервис, как система контроля версий для совместно работы в команде разработки.

Фреймворк – программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

REST API (REST) – стиль архитектуры программного обеспечения для построения масштабируемых веб-приложений.

CSS – формальный язык описания внешнего вида веб-страницы, написанного с использованием языка разметки (HTML).

HTML – стандартизированный язык разметки для просмотра веб-страниц в браузере.

Юзабилити-тестирование – это метод оценки интерфейса со стороны удобства и эффективности его использования.

React – JavaScript-библиотека для создания пользовательских интерфейсов.

Go – компилируемый многопоточный язык программирования, разработанный внутри компании Google.

Laravel – это бесплатный PHP-фреймворк с открытым исходным кодом, специально разработанный для создания сложных веб-приложений.

UI-тестирование – это процесс тестирования элементов управления в приложении, который помогает убедиться, что интерфейс соответствует ожидаемой производительности и функциональности.

Юзабилити-тестирование – это метод оценки интерфейса со стороны удобства и эффективности его использования.

2.2 Обзор аналогов

2.2.1 Тинькофф

Банк-эквайер Тинькофф позиционирует себя как инновационный и открытый банк для любого бизнеса, в том числе и малого.

У банка собственный процессинговый центр, прозрачная комиссия, которая зависит от оборота компании, а также открытые тарифы на услуги.

Интерфейс сайта представлен в соответствии с рисунком 1, рисунком 2.

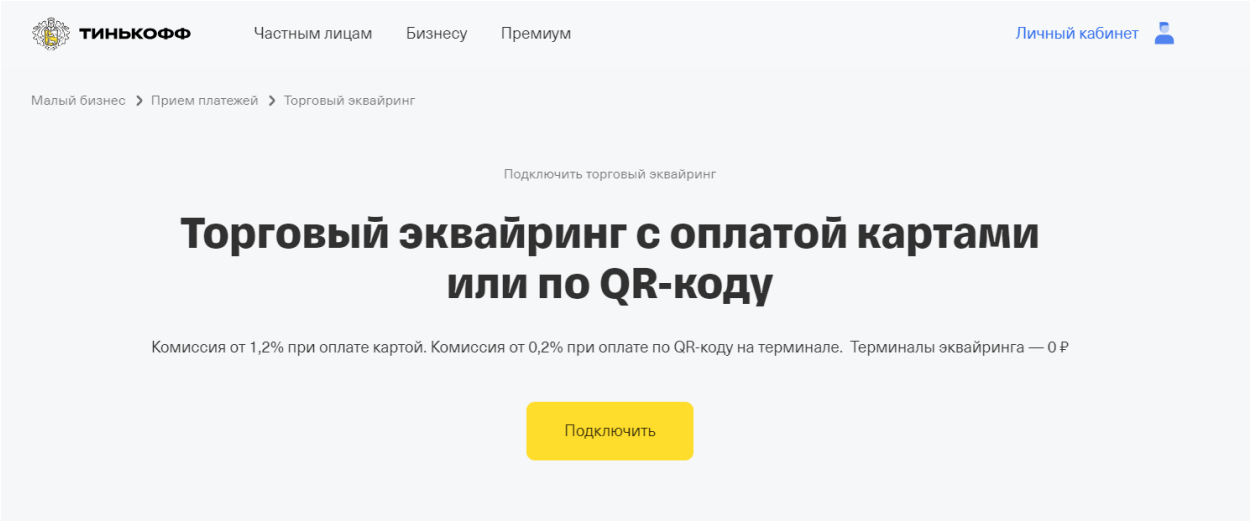

Рисунок 1 - Интерфейс сайта «Тинькофф»

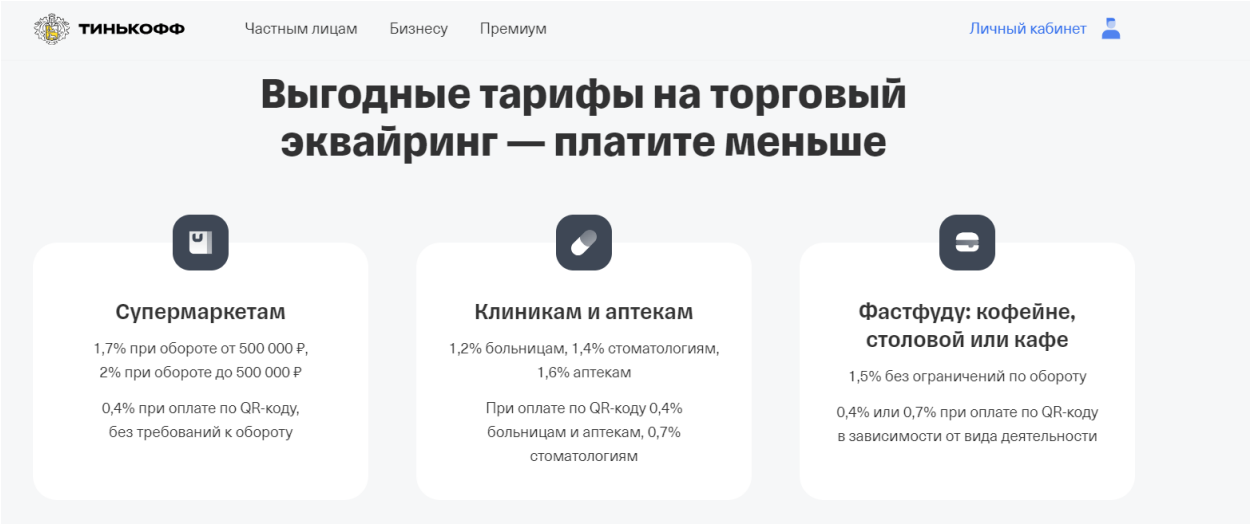

Рисунок 2 - Тарифы для клиентов

К недостаткам пользователи относят процесс валютного контроля, который зачастую может занять много времени.

2.2.2 Робокасса

Робокасса – платежный сервис, который был запущен в 2008 году и на данный момент по популярности уступает только Яндекс.Кассе.

Выделим основные плюсы Робокассы:

— широкий спектр возможностей по приему платежей: через онлайн банки, платежные терминалы, в некоторых розничных сетях, может списать средства со счета мобильного телефона или принять электронную валюту и т.д.;

— круглосуточная поддержка покупателей;

— система обслуживает как юридических, так и физических лиц;

— удобный и несложный процесс подключения.

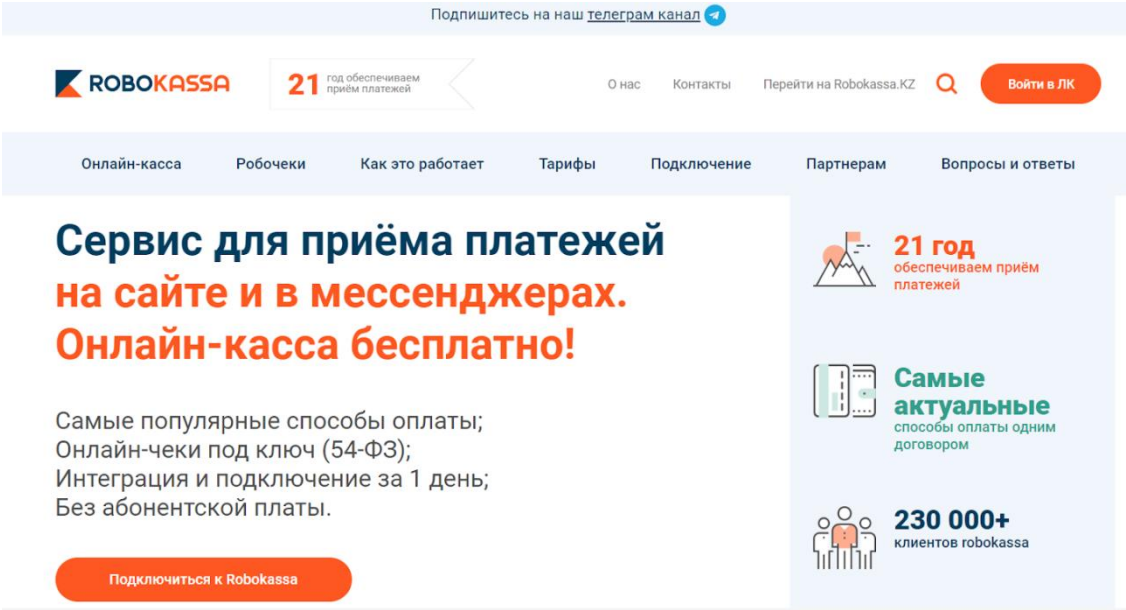

Рисунок 3 - Интерфейс сайта «Робокасса»

К недостаткам можно отнести:

— достаточно высокая комиссия – 5%;

— в истории сервиса были случаи временного прекращения приема платежей ИП и физическим лицам.

2.3 Моделирование системы

2.3.1 Диаграмма в стиле методологии IDEF0

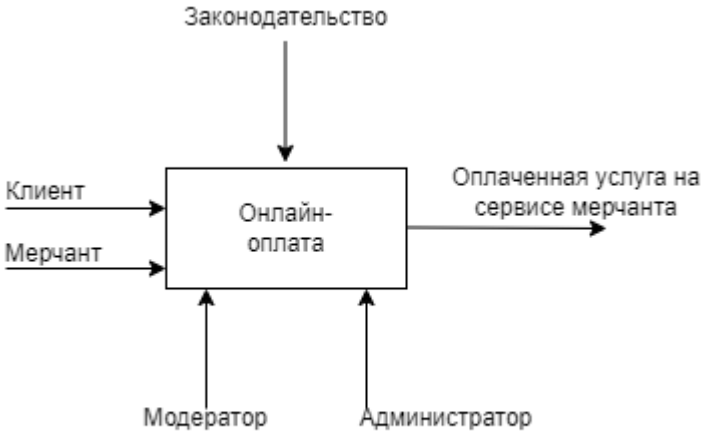

Рисунок 4 - Диаграмма IDEF0

2.3.2 Диаграмма последовательности

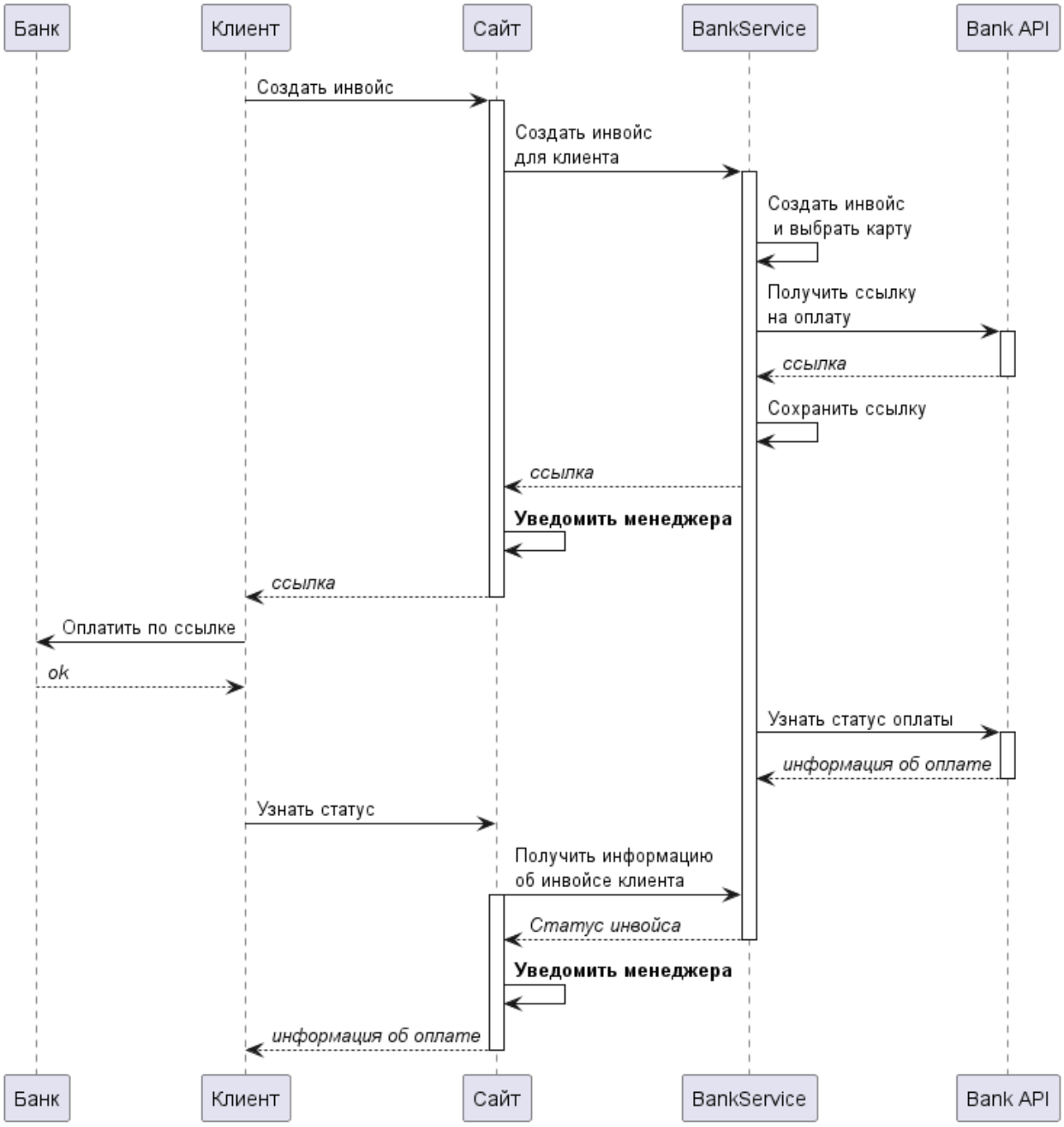

Рисунок 5 - Диаграмма последовательности

2.3.3 Диаграмма use-case

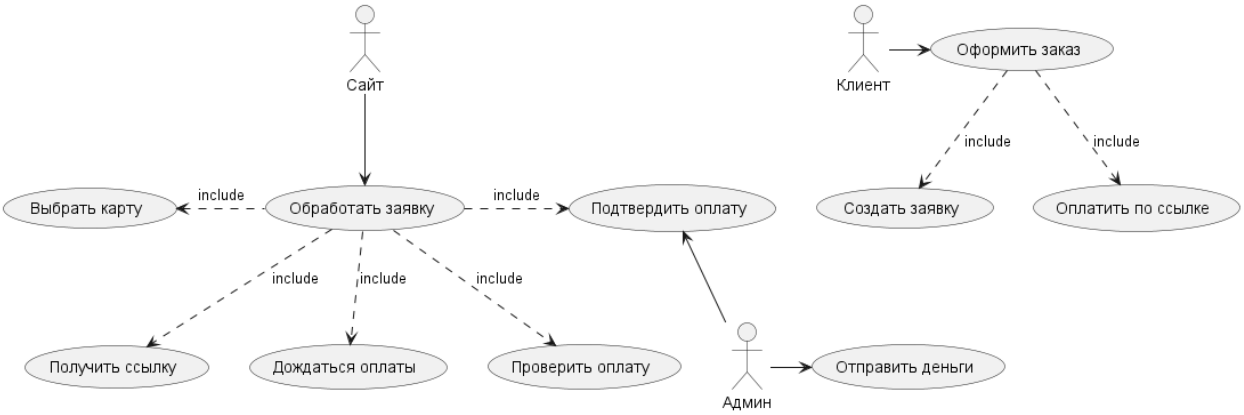

Рисунок 6 - Диаграмма use-case

2.3.4 Диаграмма сущностей для микросервиса личного кабинета

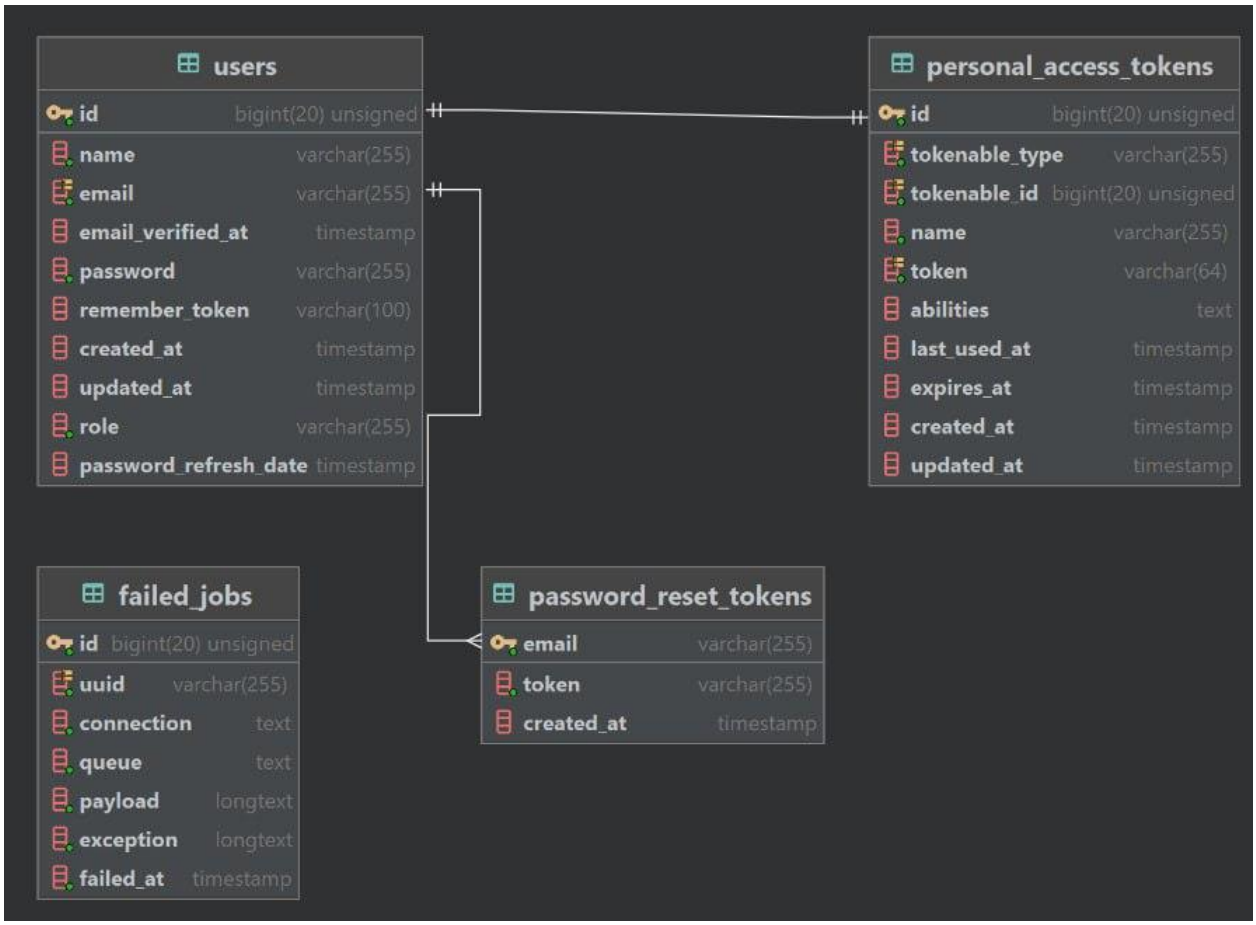

Рисунок 7 - Диаграмма сущностей для микросервиса личного кабинета

2.3.5 Диаграмма сущностей для микросервис Go

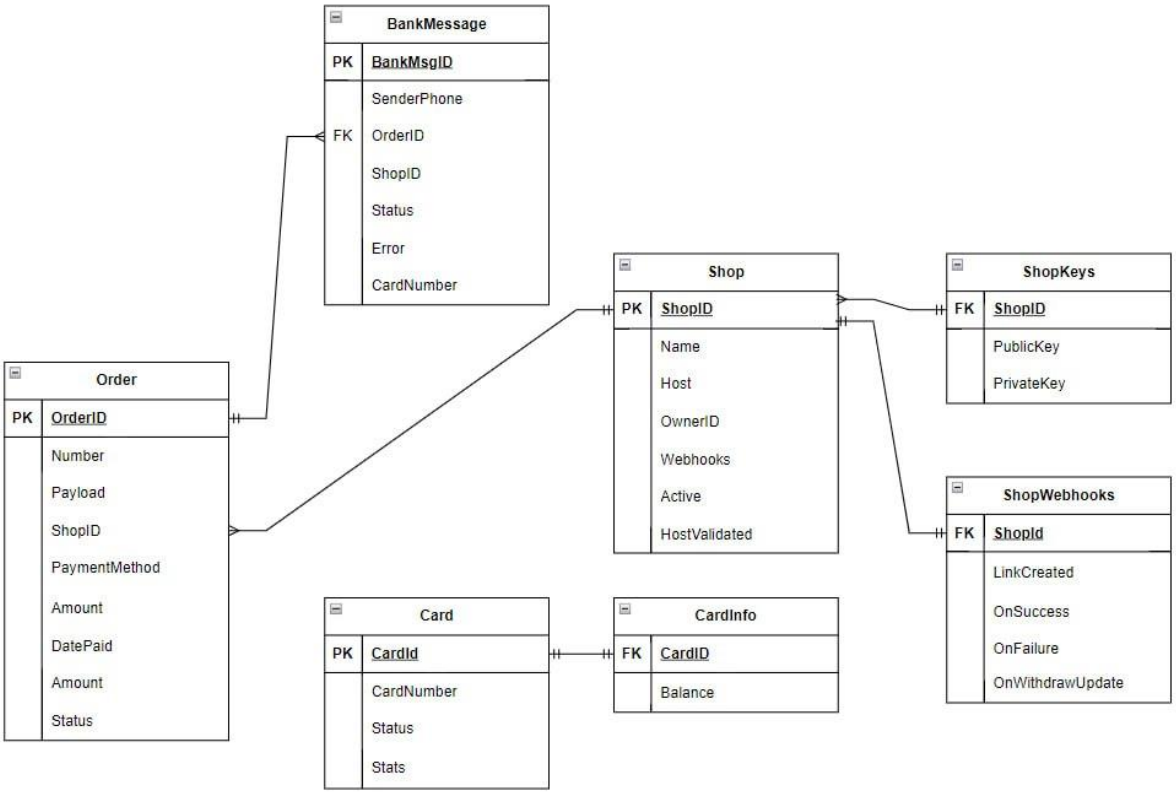

Рисунок 8 - Диаграмма сущностей для микросервиса Go

3 Реализация

3.1 Средства реализации

Веб-приложение имеет архитектуру, соответствующую шаблону клиент-серверного приложения и разделяется на backend и frontend посредством REST API. При этом сделана микросервисная архитектура.

Данная архитектура веб-приложения соотносится с основными требованиями к проекту, а именно:

- разделение пользователей на: клиентов, мерчантов, администраторов и модераторов;
- авторизация пользователя.

Для реализации программно-аппаратной части были выбраны следующие технологии:

- Laravel для панели управления ролями и интерфейсом для работы с заказами и клиентами;
- Go для обработки запросов на API системы;
- MySQL для работы с базой данных.

Для реализации клиентской части были выбраны следующие технологии:

- React для взаимодействия с пользователем и улучшения пользовательского опыта.

3.2 Реализация серверной части веб-приложения

Для осуществления основных сценариев веб-приложения необходима авторизация.

Помимо авторизации и личного кабинета клиента реализованного на Laravel, нужно учитывать что сервис должен выдерживать большое количество запросов к его REST API, соответственно было принято решение поднять отдельный сервис на GO с реализацией очереди запросов. В целом связка Laravel (php) и GO очень популярна и используется повсеместно

благодаря своей гибкости и скорости разработки очень широкого спектра сервисов.

Оба сервиса (личный кабинет и Rest API) реализованы с использованием контейнеризации Docker и взаимодействуют между друг другом с помощью Rest API внутри докера.

Принятие запросов на API реализовано с помощью Traefik, в силу чего на Go микросервис попадают только запросы с роутом /API/* Все роуты в этой маске – публичные. Извне Go недосягаем, только по роутам соответствующей маски.

3.3 Реализация клиентской части веб-приложения

Для реализации основных сценариев веб-приложения, клиентская часть разработки делится постранично. Каждая страница описывается языком программирования JavaScript, языком разметки HTML и благодаря использованию фреймворка React. За реализацию заранее осуществленного и утвержденного командой разработчиков дизайна используется язык стилей CSS.

Архитектура разработки была организована согласно бизнес-логике проекта на основании модульного подхода, по которому все компоненты и логика находятся рядом друг с другом.

Все страницы веб-приложения были реализованы и представлены командой разработчиков в соответствии с заявленным дизайном и соответствующими правками, внесенными в процессе разработки системы.

Для упрощения архитектуры всего приложения, было принято решение собрать React и связать его с приложением прямо внутри Laravel. В дальнейшем планируется отделить frontend принятия оплаты на отдельный сервис, все инструменты для этого присутствуют.

4 Тестирование

4.1 Ручное UI–тестирование

Ручное UI-тестирование – это тестирование, в процессе которого удается проверить качество пользовательского интерфейса, его соответствие требованием и проверка работоспособности каждой кнопки.

Мы провели UI-тестирование, в результате нам пришлось несколько раз переделать макет для упрощения процесса оформления оплаты пользователями.

4.2 Юнит тестирование

Юнит тестирование – это процесс программирования, в результате которого удается проверить корректность работы отдельных модулей исходного кода программы.

Нами были написаны юнит тесты для сервиса Go, с целью снижения рисков ошибки у публичного API. В результате удалось найти несколько ошибок и исправить их. В итоге юнит тестирование было пройдено успешно.

4.3 Юзабилити–тестирование

Юзабилити-тестирование является исследованием, выполняемым с целью определения, удобен ли некоторый искусственный объект для его предполагаемого применения.

Заключение

В ходе выполнения курсового проекта командой было разработано веб-приложение системы онлайн платежей, соответствующее поставленным перед проектом задачам.

В начале разработки был проведен анализ предметной области, определены основные требования к разрабатываемой системе, определены основные сценарии веб-приложения и пользовательские истории.

По результатам разработки проводился ряд тестов с целью проверки работоспособности системы.

В процессе работы были реализованы следующие задачи проекта:

- разделение пользователей на: клиентов, администраторов, мерчантов, модераторов;
- обеспечение возможности оплачивать услуги на сайте мерчантов;
- обеспечение возможности отслеживания информации по каждому магазину;
- обеспечение возможности подключать магазины к онлайн кассе.

Список используемых источников

1. Документация JavaScript-библиотеки React.js [Электронный ресурс]. Режим доступа: https://ru.react.js.org/docs/getting-started.html

2. Документация Sass - метаязыка на основе CSS [Электронный ресурс]. Режим доступа: https://sass-lang.com/documentation/

3. Основы HTML [Электронный ресурс] Режим доступа: https://html5book.ru/osnovy-html/

4. Ручное UI-тестирование [Электронный ресурс]. Режим доступа: https://ux-journal.ru/kak-provodit-ui-testirovanie-sravnenie-instrumentov.html

5. Юзабилити-тестирование [Электронный ресурс]. Режим доступа: https://lpgenerator.ru/blog/2016/07/28/yuzabiliti-testirovanie-poshagovaya-instrukciya-na-primere-yelp/

6. Федеральный закон от 22.05.2003 № 54-ФЗ (ред. от 23.11.2020) «О применении контрольно-кассовой техники при осуществлении расчетов в Российской Федерации» // СЗ РФ, 26.05.2003, № 21, ст. 1957.

7. ГОСТ 7.32-2001 [Электронный ресурс]. Режим доступа:https://kpfu.ru/portal/docs/F1867381138/gost7_32_2001.pdf