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

Факультет Компьютерных наук

Кафедра программирования и информационных технологий

Мобильное приложение для заказа тортов с возможностью выбора диаметра, картинки или текста «Caker»

Курсовая работа

Направление: 09.03.04. Программная инженерия

Зав. Кафедрой _____ д. ф.-м. н, доцент С.Д. Махортов

Руководитель _____ ст. преподаватель В.С. Тарасов

Руководитель практики _____ А.В. Москаленко

Обучающийся _____ Е.В. Мишненкова, 3 курс, д/о

Обучающийся _____ В.С. Бакланова, 3 курс, д/о

Обучающийся _____ В.П. Кильченко, 3 курс, д/о

Обучающийся _____ М.И. Стрельников, 3 курс, д/о

Обучающийся _____ М.М. Чаленко, 3 курс, д/о

Воронеж 2025

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Термины, используемые в данном документе описаны в таблице 1.

Таблица 1 - Термины, используемые в курсовом проекте

Термин	Значение
API	Интерфейс, предоставляемый программой для использования ее в другой программе.
Android	Операционная система для мобильных устройств с сенсорным экраном.
Back-end	Часть программного обеспечения, отвечающая за обработку данных и представляющая собой серверное приложение.
Git	Распределенная система управления версиями, которая обеспечивает контроль изменений в коде, возможность ветвления и слияния кода.
GitHub	Платформа для хостинга проектов на базе Git, которая обеспечивает возможность хранения кода, управления задачами, рецензирования кода и совместной работы над проектами.
HTTP	Протокол передачи данных в сети Интернет, который используется для передачи информации между клиентом и сервером.
HTTPS	Защищенная версия протокола HTTP, использующая шифрование для безопасной передачи данных.

Продолжение таблицы 1

Термин	Значение
Java	Строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems.
PostgreSQL	Объектно-реляционная система управления базами данных (СУБД) с открытым исходным кодом.
REST API	Архитектурный стиль взаимодействия между клиентом и сервером через HTTPS.
Авторизированный пользователь	Пользователь, который прошел процедуру аутентификации и получил доступ к определенным ресурсам, функциям или услугам в рамках системы или приложения.
Аутентификация	Процесс проверки подлинности личности или учетных данных пользователя для подтверждения его идентичности.
Незарегистрированный пользователь	Пользователь, который не прошел процедуру аутентификации или идентификации при доступе к ресурсам, функциям или услугам.
Пользователь-клиент	Физическое или юридическое лицо, которое приобретает товар (совершает заказ торта).
Пользователь-кондитер	Лицо, которое предоставляет товар (торты), получает заказы и реализует их.

Продолжение таблицы 1

Термин	Значение
Система управления базами данных (СУБД)	Совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.
Таск-менеджер	Специальное программное обеспечение или онлайн-сервис, предназначенный для управления задачами и проектами.
Токен аутентификации	Специальный текстовый код, используемый для подтверждения легитимности и подлинности пользователя при доступе к определённым ресурсам или сервисам.
Эндпоинт	Конечная точка в API, к которой можно обратиться для выполнения нужного действия или получения данных.

ВВЕДЕНИЕ

Заказ индивидуальных тортов традиционно связан с существенными временными затратами и сложностями согласования деталей между клиентом и кондитером. Существующие решения часто ограничиваются стандартными каталогами без возможности полноценной кастомизации, что не отвечает растущему спросу на персонализированные кондитерские изделия.

Традиционный процесс заказа требует многочисленных уточнений по телефону или при личной встрече, что создает неудобства для обеих сторон. Клиенты сталкиваются с трудностями визуализации конечного продукта, а кондитеры – с необходимостью ручной обработки каждого запроса. Кроме того, отсутствие прозрачных механизмов выбора исполнителя и отслеживания выполнения заказа снижает доверие к сервису.

Разработка специализированного мобильного приложения, объединяющего конструктор тортов с системой заказов, позволит оптимизировать этот процесс. Цифровое решение предоставит клиентам интуитивно понятный инструмент для самостоятельного создания дизайна с мгновенной визуализацией, а кондитерам – удобную платформу для управления заказами. Такой подход соответствует современным тенденциям цифровизации сферы услуг и персонализированного потребления.

Данная работа направлена на создание мобильного приложения с модульной архитектурой, сочетающего функционал конструктора тортов с системой заказов и коммуникации между пользователями. Решение будет основано на принципах удобства использования, прозрачности процессов и масштабируемости, что позволит упростить процесс создания и заказа индивидуальных кондитерских изделий для конечных пользователей.

1 Постановка задачи

Цель курсовой работы – разработка мобильного приложения "Caker" для заказа индивидуальных тортов с функционалом конструктора дизайна. Мобильное приложение должно быть реализовано для операционной системы Android версии 11 и выше. Серверная часть должна использовать микросервисную архитектуру для обеспечения масштабируемости и надежности.

1.1 Цели проекта

Целью курсового проекта является создание системы, которая должна обеспечивать выполнение следующих целей:

– Увеличение количества принятых заказов по мнению пользователей-кондитеров. Из опрошенных не менее 70% должны отметить, что объем заказов увеличился;

– Ускорение процесса заказа торта с индивидуальным дизайном по мнению пользователей-клиентов. Из опрошенных не менее 70% должны отметить, что время согласования деталей индивидуального дизайна сократилось.

1.2 Задачи приложения

Основными задачами разрабатываемого приложения "Caker" являются предоставление пользователям следующих возможностей:

Для клиентов:

– Создание индивидуального дизайна торта с использованием встроенного конструктора;

– Просмотр каталога кондитеров;

– Оформление заказов с указанием параметров торта, даты изготовления;

– Создание и редактирование своего профиля;

– Просмотр своих заказов и их текущих статусов.

Для кондитеров:

– Создание и редактирование профиля с портфолио работ;

– Публикация информации о доступных вариантах тортов/десертов (основы, начинки, декор);

– Прием и обработка поступающих заказов.

1.3 Функциональные требования

Пользователи разделены на следующие группы:

– Неавторизованный пользователь;

– Авторизованный пользователь-клиент;

– Авторизированный пользователь-кондитер;

– Администратор.

Каждой группе система должна позволять решить определенные задачи. Диаграммы представлены в приложениях А и Б.

Список функций, которые система должна предоставлять неавторизованному пользователю:

– Регистрация в системе

– Авторизация в системе

– Просмотр базового каталога кондитеров

– Просмотр профилей кондитеров

– Просмотр страниц товаров (тортов)

Авторизованному пользователю-клиенту должны быть доступны все функции, доступные неавторизованному пользователю, за исключением авторизации и регистрации, а также следующие функции:

– Управление профилем (просмотр, редактирование, удаление);

– Работа с корзиной (добавление/удаление товаров);

– Оформление и отслеживание заказов;

– Коммуникация с кондитерами;

– Использование конструктора тортов (выбор параметров, цветов, надписей);

– Загрузка изображений для дизайна;

– Оставление отзывов и оценок;

– Просмотр истории заказов.

Авторизированному пользователю-кондитеру должны быть доступны все функции, доступные неавторизованному пользователю, а также следующие функции:

– Управление профилем (просмотр, редактирование, удаление);

– Управление каталогом своих готовых товаров (добавление/редактирование/удаление);

– Управление заказами (прием, изменение статусов);

– Коммуникация с клиентами;

– Формирование ценовых предложений;

– Просмотр финансовой статистики.

1.4 Задачи, решаемые в процессе разработки

Разработка системы включает следующие задачи:

– Провести анализ предметной области и исследование аналогов

– Разработать техническое задание

– Спроектировать архитектуру системы

– Создать UML-диаграммы и схему БД

– Разработать дизайн мобильного приложения

– Реализовать backend-часть (API)

– Реализовать frontend-часть (мобильное приложение)
– Протестировать систему
– Проанализировать перспективы развития.

2 Анализ существующих решений

В данном разделе рассматриваются существующие на рынке решения, позволяющие осуществлять заказ тортов с индивидуальным дизайном. Анализируемые сервисы разделены на две категории:

– Специализированные платформы для заказа кастомных тортов;

– Сервисы для кондитеров, предназначенные для работы с бизнесом.

В ходе исследования были проанализированы следующие ключевые конкуренты: Sweetify, PastryPro и BakeMaster с точки зрения кондитера, Sweetify, Flowwow и Авито с точки зрения клиента.

2.1 Платформы для заказа тортов

В таблице 2 представлено краткое сравнение платформ для заказа тортов.

Таблица 2 - Сравнение платформ для заказа тортов

Параметр сравнения	Sweetify	Flowwow	Авито
Возможность редактирования торта	Да (шаблоны)	Нет	Нет
Качество дизайна	4/5	4/5	3/5
Оценка пользователей	4.0	4.5	4.3
Служба поддержки	Эл. почта, телеграм	Эл. почта, телеграм	Эл. почта, чат на платформе
Выбор содержимого торта	Выбор начинки,	Выбор из готовых	Текстовое описание пожеланий

	размера, комментарий	вариантов, комментарий	
Интеграция с изготовителями тортов	Частные пекарни, физ./юр. лица	Частные мастера, кондитерские	Частные мастера, мелкие пекарни
Система оплаты	При получении, заранее	При получении, заранее	При получении, предоплата

2.1.1 Sweetify

Sweetify — это мобильное приложение и онлайн-платформа для заказа тортов и других кондитерских изделий с возможностью кастомизации. Сервис ориентирован как на частных клиентов, так и на профессиональных кондитеров, предлагая удобный интерфейс для оформления заказов.

На рисунке 1 представлена титульная страница мобильного приложения Sweetify.

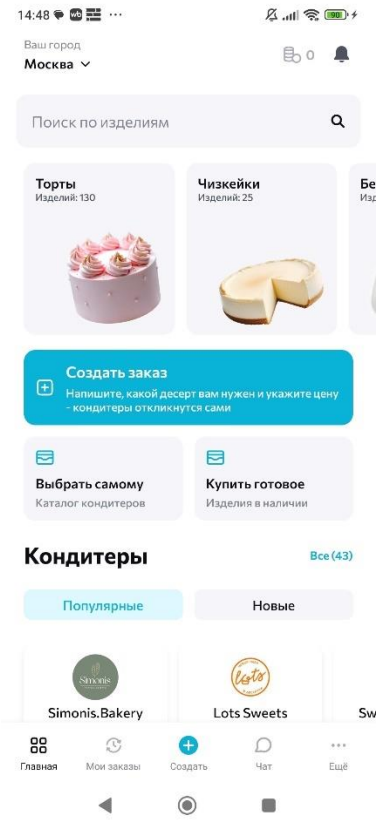

Рисунок 1 – Титульная страница мобильного приложения Sweetify

У данного приложения можно выделить следующие преимущества:

– Широкий выбор продукции;
– Гибкая система оплаты;
– Интеграция с разными типами кондитеров;
– Удобный выбор параметров;
– Интуитивный интерфейс с приятной цветовой палитрой.

В противовес можно выделить недостатки:

– Ограниченный конструктор;
– Проблемы со стабильностью;
– Платные возможности.

2.1.2 Flowwow

Flowwow — это маркетплейс по заказу авторских подарков и десертов с доставкой, включая торты, букеты и сладкие наборы. Хотя сервис не специализируется исключительно на кондитерских изделиях, его бизнес-модель и подход к персонализации заказов представляют интерес для нашего проекта.

На рисунке 2 представлена титульная страница мобильного приложения Flowwow.

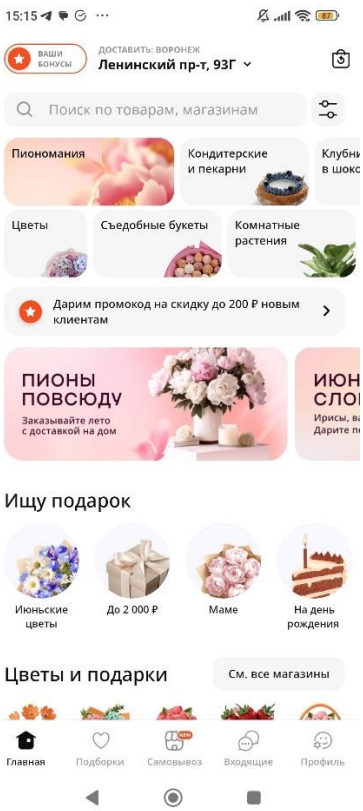

Рисунок 2 – Титульная страница мобильного приложения Flowwow

У данного решения можно выделить следующие преимущества:

– Широкий ассортимент;

– Удобный интерфейс;

– Интеграция с мастерами.

При этом можно обратить внимание на недостатки:

– Ограниченная кастомизация;

– Размытая специализация.

2.1.3 Авито

Авито – крупнейшая российская площадка объявлений, где частные кондитеры и пекарни предлагают торты и десерты на заказ. Хотя сервис не специализируется на кондитерских изделиях, его раздел по продаже сладостей представляет интерес как пример альтернативного канала сбыта.

На рисунке 3 представлена титульная страница приложения Авито.

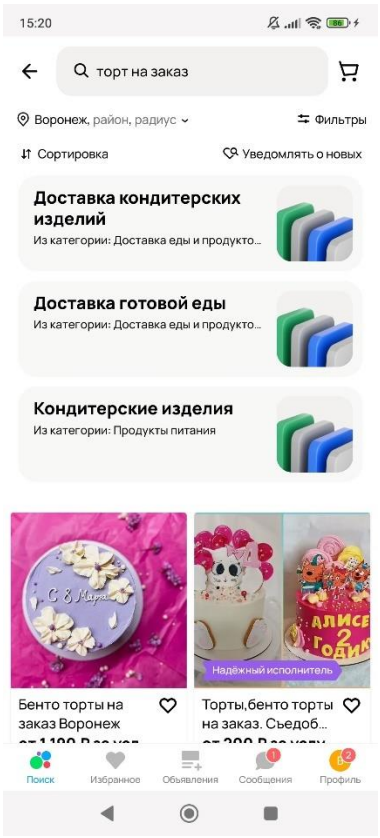

Рисунок 3 – Титульная страница сайта Авито

У данного решения можно выделить следующие преимущества:

– Огромная аудитория;
– Простота использования;
– Гибкость коммуникации;
– Разнообразие предложений.

В противовес можно выделить недостатки:

– Полное отсутствие конструктора;

– Нет системы заказов;

– Слабая модерация.

2.2 Сервисы для кондитеров

В таблице 3 представлено краткое сравнение магазинов аренды.

Таблица 3 - Сравнение сервисов для кондитеров

Параметр сравнения	Sweetify	PastryPro	BakeMaster
Тип аудитории	Массовый рынок (частные клиенты)	Профессионалы (B2B и ивенты)	Бюджетные частные заказы
Регистрация	Простая (15 мин)	Сложная (документы и верификация)	Очень простая (5 мин)
Комиссия	15-25% и платное продвижение	8-12% и фиксированная подписка	5-7% (самые низкие)
Выплаты	3-5 рабочих дней	До 7 рабочих дней	Мгновенно на карту
Инструменты	Базовый кабинет	Профессиональная аналитика	Минимальный функционал
Конструктор тортов	Шаблоны	Только текстовые заявки	Отсутствует
Поддержка	Email/чат (медленно)	Персональный менеджер	Только email

Реклама	Встроенные баннеры	Нет	Агрессивная реклама в ЛК

2.2.1 Sweetify для кондитеров

Sweetify — маркетплейс для кондитеров, объединяющий частных мастеров, пекарни и кондитерские бренды. Платформа предоставляет инструменты для продажи тортов и других десертов с базовой системой заказов.

На рисунке 4 представлена титульная страница сайта Sweetify.

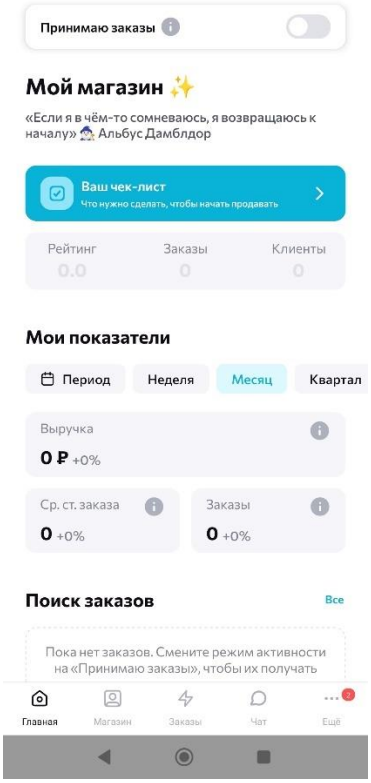

Рисунок 4 – Титульная страница приложения Sweetify

У данного решения можно выделить следующие преимущества:

– Широкая клиентская база;
– Гибкие условия работы;
– Автоматизация расчетов;

– Минимальные барьеры входа;

– Расширенный ассортимент.

В противовес можно выделить недостатки:

– Высокая комиссия;

– Конкуренция с крупными пекарнями.

2.2.2 PastryPro

PastryPro – профессиональная B2B-платформа для кондитеров и пекарен, ориентированная на систематизацию заказов и работу с постоянными клиентами. Акцент сделан на юридическую чистоту сделок и аналитику продаж.

На рисунке 5 представлена титульная страница сайта PastryPro.

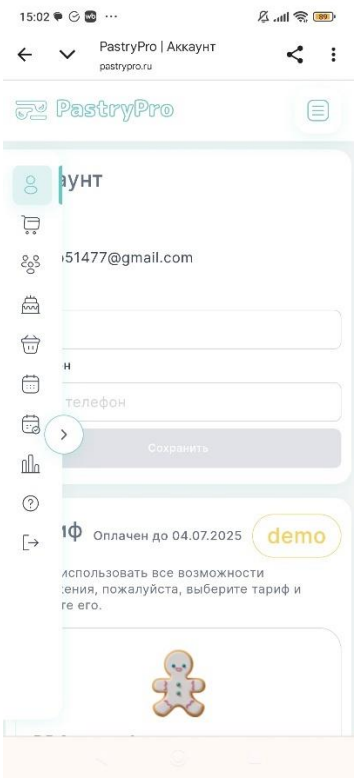

Рисунок 5 – Титульная страница сайта PastryPro

У данного решения можно выделить следующие преимущества:

– Профессиональная аудитория;
– Детальная аналитика;
– Юридическая поддержка;
– Гибкое ценообразование;
– Качественный трафик.

В противовес можно выделить недостатки:

– Высокий порог входа;
– Подписка на функционал;
– Сложный интерфейс;
– Долгий вывод средств.

2.2.3 BakeMaster

BakeMaster — упрощённая платформа для кондитеров-фрилансеров и мини-пекарен, предлагающая базовые инструменты для приёма заказов без сложных настроек. Позиционируется как "лёгкий старт" для начинающих мастеров.

На рисунке 6 представлена титульная страница сайта BakeMaster.

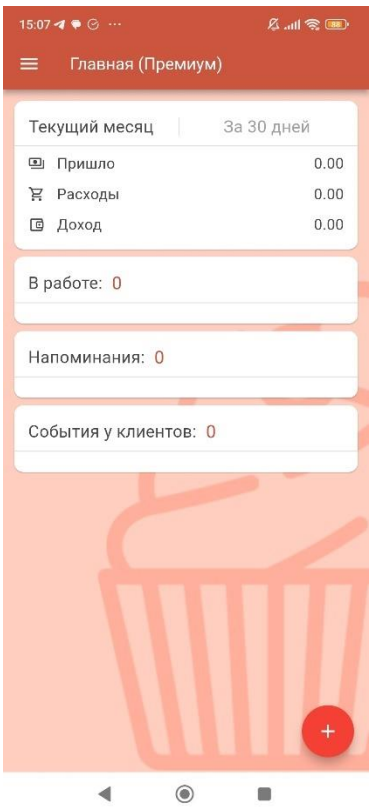

Рисунок 6 – Титульная страница сайта BakeMaster

У данного решения можно выделить следующие преимущества:

– Низкий порог входа;
– Минимальные комиссии;
– Простое управление;
– Мгновенные выплаты.

В противовес можно выделить недостатки:

– Отсутствие аналитики;
– Рекламный спам;
– Риск сорванных заказов.

2.3 Вывод по анализу существующих решений

В процессе анализа кондитерских платформ было выявлено следующее:

Каждая из рассмотренных платформ (Sweetify, PastryPro, BakeMaster) предлагает либо широкий охват аудитории за счет минимальных барьеров входа для кондитеров, либо гарантированное качество через строгую верификацию мастеров.

То есть совмещение этих подходов могло бы создать идеальную платформу, сочетающую массовость с премиальным сервисом. Однако реализуемый в рамках проекта Caker представляет собой MVP (минимально жизнеспособный продукт), который не ставит перед собой столь амбициозных задач на первом этапе. Тем не менее, данное направление развития может стать ключевым конкурентным преимуществом в будущем.

Более практическим выводом является список элементов и функций, которые присутствуют во всех или большинстве проанализированных приложений, среди которых находятся:

- Поиск и фильтрация кондитеров/тортов;
- Профиль пользователя (клиента);
- Профиль кондитера с портфолио;
- Детализированное описание товаров;
- Фотогалерея для каждого торта;
- Указание сроков выполнения заказа.

При этом предлагается избегать следующих ошибок:

- Чрезмерные требования к документам на старте (как у PastryPro);
- Неочевидная система указания дат изготовления;
- Агрессивная реклама, ухудшающая UX;
- Полное отсутствие визуального конструктора.

3 Анализ целевой аудитории

Целевой аудиторией являются частные клиенты, желающие заказать индивидуальный торт для праздника или мероприятия, а также кондитеры и пекарни, предлагающие услуги по изготовлению тортов на заказ.

Группа частных клиентов более многочисленна, но их активность носит эпизодический характер (заказы обычно приурочены к конкретным датам — дням рождения, свадьбам, корпоративам). Частные клиенты ценят удобство, наглядность (визуализацию торта) и возможность быстро оформить заказ. В свою очередь, группа кондитеров и пекарен менее многочисленна, но значительно активнее, поскольку их деятельность связана с постоянным поиском заказов и продвижением своих услуг. Кондитеры заинтересованы в простых и прозрачных инструментах для работы, низких комиссиях и эффективном продвижении своих услуг.

Стоит отдельно отметить, что среди кондитеров можно выделить две основные подгруппы:

– Частные мастера — чаще молодые специалисты, начинающие свой бизнес, которые активно используют цифровые платформы для привлечения клиентов.

– Профессиональные пекарни и кондитерские — более опытные участники рынка, имеющие стабильный поток заказов, но также заинтересованные в расширении клиентской базы.

4 Реализация

4.1 Способ ведения проекта

Система разрабатывалась с применением гибкой методологии Kanban, оптимально подходящей для команд со специализированными ролями. Данная методология подходит для команд, участники которых обладают узкоспециализированными навыками и выполняют задачи в пределах своей компетенции. Работа была структурирована в виде итеративных спринтов с четким распределением задач. Реализация проекта была организована следующим образом:

– Распределение ролей внутри команды – каждому из участников команды была отведена определённая роль;

– Обсуждение и формирование требований к системе – был обговорен и сформирован список функциональных и нефункциональных требований к системе. Все требования были разделены на 4 категории в соответствии с методом приоритизации MoSCoW;

– Создание и оформление командного таск-трекера – на платформе «Яндекс.Трекер» создан проект «ТП “Caker”», на доске которого размещаются карточки с задачами всех участников команды. В соответствие с выбранной методологией ведения проекта основная доска разделена на колонки, отражающие статус задач и этап, на котором они находятся в данный момент в текущий момент времени;

– Планирование задач – В начале каждого спринта команда совместно распределяет задачи между участниками с учетом их специализации. Задачи фиксируются в общем таск-трекере с указанием приоритетов для последовательного выполнения;

– Адаптация приоритетов – при изменении требований команда оперативно пересматривает приоритеты задач без ожидания окончания текущего спринта, обеспечивая гибкость разработки;

– Контроль прогресса – Еженедельные встречи включают стартовое обсуждение итогов прошлого спринта и планирование нового, а также промежуточную синхронизацию для выявления и устранения возникающих сложностей;

– Прозрачность процессов – По завершении каждого спринта команда формирует отчет для заказчика, обеспечивая полную информированность о ходе работ.

Процесс разработки был поделен на две крупные стадии – разработка и доработка ПМИ. В рамках первой было сделано следующее:

– Проведен анализ предметной области и собрана необходимая информация для предпроектного исследования;

– Проведен анализ рынка и выявлена целевая аудитория разрабатываемой системы

– Составлена необходимая документация и спроектированы основные диаграммы системы;

– Созданы макеты для основных экранов приложения и оформлена единая дизайн-система для обеспечения визуальной целостности проекта;

– Разработана основная часть сервиса “Profile”;

– Разработана основная часть сервиса “Product”;

– Разработан и протестирован сервис “Authorization”;

– Разработан и протестирован сервис “Gateway”;

– Разработана основная часть мобильного приложения;

– Проведено первичное тестирование приложения и обнаружены баги и неточности в идее разработки, требующие доработки;

– Разработана и протестирована База данных.

В рамках второй стадии было сделано следующее:

– Скорректирован способ решения задач: возросла частота командных встреч, а также разработчики начали проводить отдельные технические обсуждения;
– Созданы макеты для всех экранов приложения;
– Завершена разработка сервисов, созданных во период первой стадии;
– Разработан сервис “Order”;
– Разработан сервис “Payment”;
– Интегрирована нейронная сеть;
– Завершена разработка мобильного приложения;
– Проведено тестирование приложения;
– Составлена ПМИ.

4.2 Средства реализации

4.2.1 Серверная часть

Для реализации серверной части системы были выбраны следующие технологии:

– Язык программирования C# версии 12;
– Фреймворки .NET, ASP.NET Core версии 9.0.105;
– Система управления базами данных PostgreSQL;
– Виртуальный личный сервер от TimeWeb;
– DNS-регистратор Namecheap;
– Платформа контейнеризации Podman;
– HTTP веб сервер NGINX;
– Хранилище Docker образов CitHub Container Registry;
– Система автоматизации задач GitHub actions на платформе GitHub;

– Менеджер генеративных нейросетей для изображений InvokeAI с моделью FLUX Schnell (Quantized)

Преимуществами C# являются его высокая производительность, кроссплатформенность благодаря .NET, современный синтаксис с поддержкой record-типов, pattern matching и nullable reference types. C# 12 предоставляет улучшения в виде первичных конструкторов, коллекций и лямбда-выражений, что ускоряет разработку надежных и масштабируемых серверных приложений.

ASP.NET Core 9.0 предлагает минимальные API, улучшенную производительность, встроенную поддержку контейнеризации и инструменты для создания микросервисов. Благодаря встроенным шаблонам аутентификации, OpenAPI-документированию и эффективной маршрутизации, фреймворк упрощает создание высоконагруженных веб-приложений.

Преимуществами PostgreSQL являются его высокая производительность, надежность и поддержка сложных запросов. Благодаря широкому набору встроенных функций, поддержке расширяемости и соответствию стандартам SQL, PostgreSQL эффективно справляется с хранением и обработкой больших объемов данных, обеспечивая стабильную работу приложения в условиях высокой нагрузки.

TimeWeb предоставляет гибкие VPS-решения с возможностью выбора ОС, масштабированием ресурсов (CPU/RAM/SSD) и резервным копированием. Ключевые преимущества: доступность, низкая задержка и простота управления через веб-интерфейс.

Namecheap предлагает доменную регистрацию с DNSSEC, двухфакторной аутентификацией и защитой WHOIS Privacy. Преимущества — конкурентные цены, надежность и интеграция с SSL-сертификатами.

Podman обеспечивает запуск контейнеров без демона (daemonless), поддержку rootless-режима и совместимость с Docker-образами. Его преимущества — повышенная безопасность, интеграция с systemd и работа в Kubernetes-средах.

NGINX обрабатывает высокие нагрузки благодаря асинхронной архитектуре, предоставляет балансировку, кеширование и TLS-терминацию. Ключевые особенности: низкое потребление ресурсов, модульность и простота конфигурации.

GHCR обеспечивает безопасное хранение Docker-образов с автоматическим сканированием уязвимостей, управлением правами доступа и интеграцией с GitHub Actions. Преимущества — бесплатные тарифы для публичных репозиториев и высокая скорость развертывания.

GitHub actions — сервис для автоматизации процессов CI/CD, встроенный в платформу GitHub. Он позволяет настраивать и запускать рабочие процессы для сборки, тестирования и деплоймента приложений при изменениях в репозитории.

InvokeAI предоставляет инструменты для генерации изображений через оптимизированные модели (в т.ч. FLUX Schnell Quantized). Преимущества: поддержка аппаратного ускорения, интуитивный веб-интерфейс и расширенное управление параметрами генерации.

4.2.2 Клиентская часть

Для реализации мобильного приложения были выбраны следующие технологии:

- Язык программирования Kotlin версии 1.9.24;
- Система сборки Android Gradle Plugin версии 8.8.1;
- Библиотека для внедрения зависимостей: Hilt версии 2.51.1;
- Фреймворк для создания UI: Jetpack Compose с Compose Compiler версии 1.5.14;
- Фреймворк для создания пользовательского интерфейса: Material Design версии 1.2.0;
- Фреймворк для превью и отладки UI: Compose UI Tooling;

– Фреймворк для локального хранилища: Room 2.6.1;

– Библиотека для верстки: ConstraintLayout;

– Библиотека для загрузки изображений: Coil Compose версии 2.2.2

– Библиотека для работы с компонентом выбора цвета: ColorPicker Compose версии 1.1.2;

– Библиотека расширенных иконок: Material Icons Extended;

– HTTP-клиент: Retrofit версии 2.9.0;

– Библиотеки для работы с JSON: Moshi версии 1.15.0 и Retrofit Converter (Moshi/Gson) версии 2.9.0;

– Библиотека для работы с датой и временем: kotlinx-datetime версии 0.5.0;

– Библиотека для отображения Splash Screen: Core Splashscreen версии 1.0.1;

– Библиотека интеграции навигации с DI: Hilt Navigation Compose.

Kotlin представляет собой современный статически типизированный язык программирования, разработанный JetBrains, оторый поддерживает объектно-ориентированную парадигму и предоставляет возможности функционального программирования, обеспечивая полную совместимость с Java и мультиплатформенную поддержку.

Android Gradle Plugin — это официальный инструмент для автоматизации сборки Android-приложений, обеспечивающий настройку зависимостей, управление версиями и оптимизацию процесса разработки.

Hilt представляет собой библиотеку для внедрения зависимостей, построенную на основе Dagger, которая упрощает работу с DI в Android-приложениях за счёт автоматического создания и связывания компонентов с учётом их жизненного цикла.

Jetpack Compose — это современный декларативный фреймворк для создания пользовательских интерфейсов, позволяющий описывать UI с

помощью Kotlin вместо XML и поддерживающий реактивное программирование.

Material Design — это дизайн-система от Google, предоставляющая готовые компоненты, анимации и гайдлайны для создания интуитивно понятных и визуально согласованных интерфейсов.

Compose UI Tooling — это набор инструментов для предварительного просмотра, инспекции и отладки интерфейсов, созданных с помощью Jetpack Compose, прямо в среде разработки Android Studio.

Room представляет собой ORM-библиотеку для работы с SQLite, которая абстрагирует рутинные операции с базой данных, предоставляя удобный Kotlin-API и compile-time проверки запросов.

ConstraintLayout — это инструмент для построения сложных адаптивных макетов с минимальной вложенностью, использующий гибкие связи между элементами.

Coil Compose — это высокопроизводительная библиотека для асинхронной загрузки и отображения изображений, оптимизированная для работы с Jetpack Compose и поддерживающая кэширование и трансформации.

ColorPicker Compose — это готовое решение для выбора цветов в приложениях на Compose, поддерживающее различные форматы цветов, палитры и пользовательские настройки.

Material Icons Extended представляет собой расширенную коллекцию векторных иконок в стиле Material Design, включающую специализированные категории для различных сценариев использования.

Retrofit — это безопасный HTTP-клиент для Android и Java, который преобразует REST API в интерфейсы, упрощая выполнение сетевых запросов и обработку ответов.

Moshi представляет собой современную библиотеку для работы с JSON, использующую Kotlin-рефлексию и кодогенерацию для эффективной сериализации и десериализации данных.

kotlinx-datetime — это кроссплатформенная библиотека для работы с датой и временем, предоставляющая иммутабельные типы данных и удобные API для манипуляций с временными промежутками.

Core Splashscreen — это стандартизированное решение для отображения экрана-заставки в Android-приложениях, обеспечивающее единообразное поведение на поддерживаемых версиях ОС.

Hilt Navigation Compose — это интеграция Hilt с системой навигации Jetpack Compose, автоматизирующая передачу параметров между экранами и управление зависимостями в графе навигации.

4.3 Архитектура системы

Архитектура системы представлена на рисунке 7.

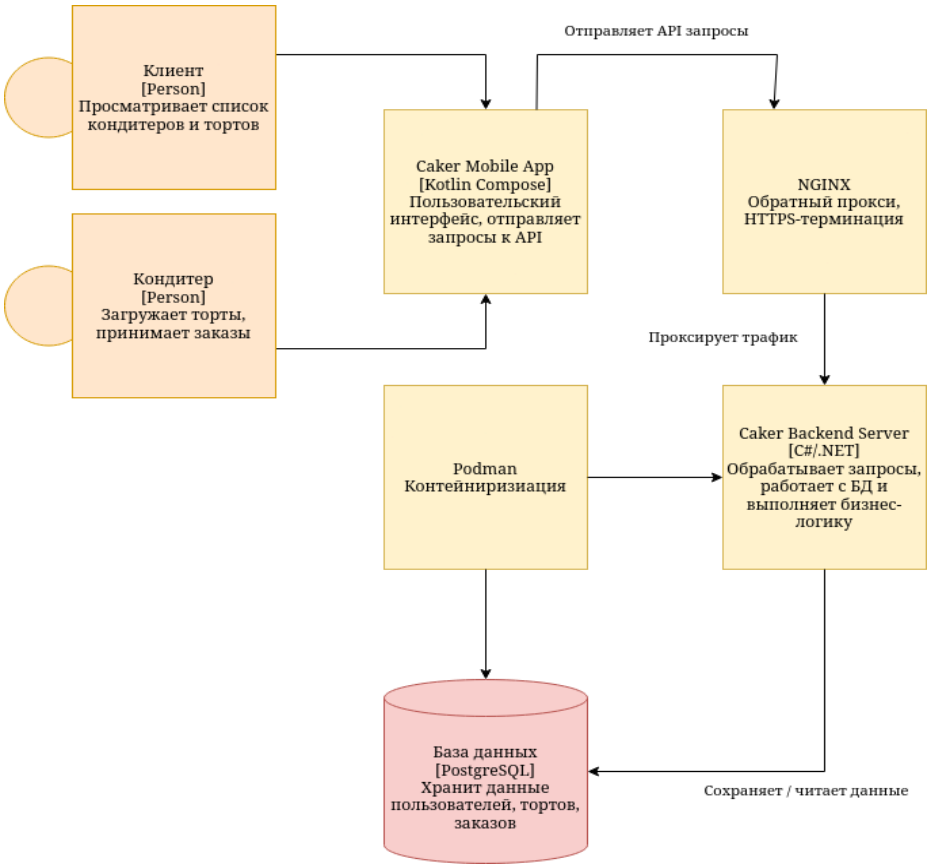

Рисунок 7 – Архитектура системы

Система реализует классическую трехзвенную архитектуру, то есть состоит из следующих частей:

– Клиентское приложение;

– Серверное приложение;

– База данных.

4.4 Серверная часть

Сервер предоставляет REST API для использование клиентской стороной.

4.4.1 Общая структура

Серверная часть состоит из базы данных и 6 микросервисов:

– “Gateway” для управления запросами;

– “Authorization” для авторизации, регистрации и проверки токена аутентификации;

– “Profile” для обработки профилей пользователей и товаров кондитеров;

– “Product” для управления каталогом тортов и поиском;

– “Order” для обработки заказов и корзины;

– “Payment” для проведения платежей;

Далее будут описаны функции, выполняемые каждым микросервисом.

4.4.2 База данных

База данных используется для хранения информации отдельно от бизнес-логики, предоставления общего доступа разным элементам и ее безопасного сохранения данных.

Схема базы данных представлена на рисунке 8.

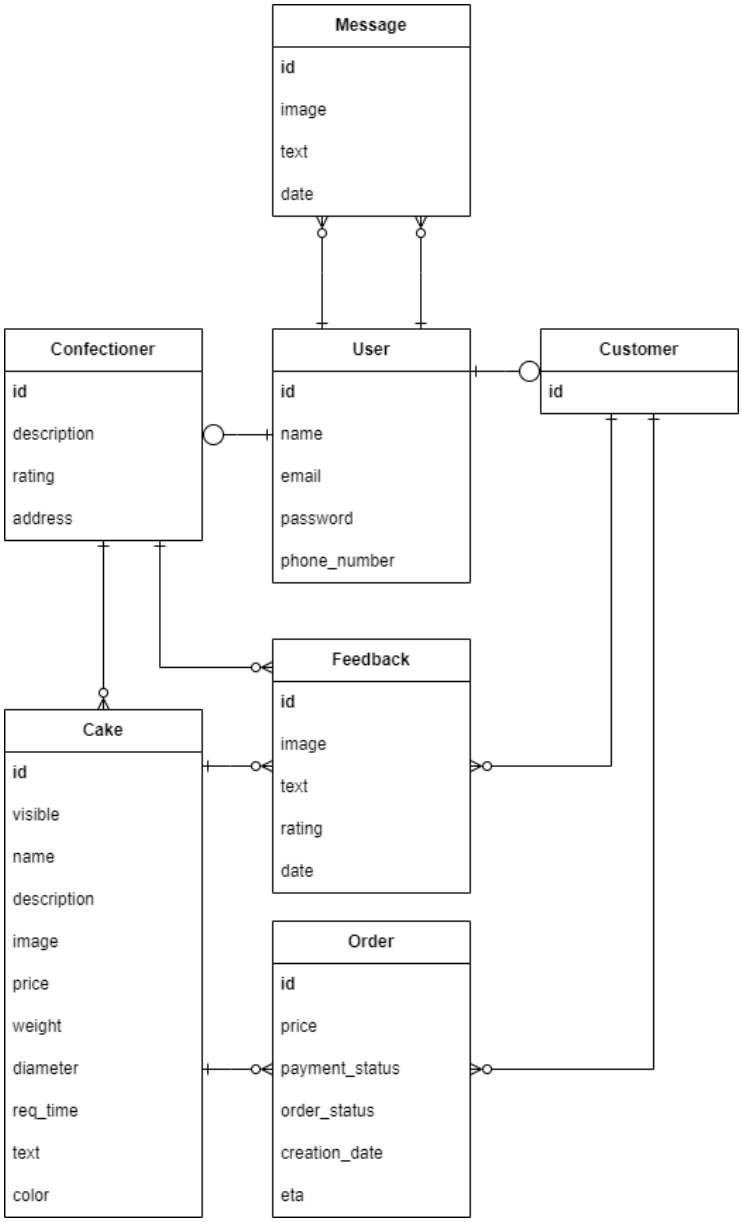

Рисунок 8 – Схема базы данных

Приведенная база данных находится в 3 нормальной форме, что позволяет избежать ошибок при вставке, удалении или обновлении данных.

4.4.3 Микросервис “Gateway”

Данный микросервис служит для управления запросами. Он представляет собой единую точку доступа, основной функцией которой является маршрутизация, то есть перенаправление входящих HTTP-запросов к соответствующим микросервисам.

4.4.4 Микросервис “Authorization”

Данный микросервис отвечает за управление процессами аутентификации и авторизации пользователей и выполняет следующие функции:

– Проверка учетных данных пользователя (адрес электронной почты и пароль) при входе в систему. Для успешной авторизации адрес электронной почты и пароль, указанные пользователем, должны совпадать с соответствующими на сервере;

– Генерирование и предоставление JWT-токена. Токен должен быть предоставлен пользователю после успешной авторизации;

– Регистрация пользователей. Для успешной регистрации адрес электронной почты, указанный пользователем, не должен быть закреплен за другим пользователем.

4.4.5 Микросервис “Profile”

Данный микросервис управляет профилями и персональными данными. Функции:

Для клиента:

– Просмотр/редактирование профиля;

– Удаление аккаунта.

Для кондитера:

– Добавление/удаление товаров;

– Редактирование информации о тортах;
– Установка цен;
– Просмотр заработка.

4.4.6 Микросервис “Product”

Данная подсистема отвечает за обработку каталога тортов и поиск. Функции:

– Просмотр каталога;
– Поиск кондитеров/тортов по параметрам;
– Управление атрибутами: цвет, диаметр, надписи, ингредиенты.

4.4.7 Микросервис “Order”

Данная подсистема отвечает за обработку каталога тортов и поиск. Функции:

Для клиента:

– Формирование корзины;
– Рассчёт стоимости;
– Указание срока выполнения;
– Просмотр истории заказов.

Для кондитера:

– Подтверждение заказов;
– Отправка ценовых предложений;
– Изменение статусов;
– Просмотр списка заказов.

4.4.8 Микросервис “Payment”

Данная подсистема отвечает за интеграцию с платёжными системами. Функции:

– Безопасная обработка транзакций;
– Отслеживание статусов оплат;
– Возвраты средств при отмене заказов.

4.4.9 Использование нейронной сети

В проекте используется нейронная сеть для генерации изображений по текстовому вводу. Обращение к ней происходит через специальный endpoint /generate в NGINX.

В данном проекте используется нейронная сеть FLUX Schnell (Quantized) и обращение к ней происходит посредством API, предоставляемого фронтендом InvokeAI, развёрнутом на личном вычислительном устройстве, оснащённом видеокартой NVIDIA RTX 3060.

4.4.10 Развертывание

Для реализации развертывания серверной части приложения были выполнены следующие шаги:

– На платформе TimeWeb была создана виртуальная машина со статическим IP-адресом, на которой запускаются все микросервисы;
– Написаны Docker файлы для создания образов микросервисов;
– Написан compose.yml файл для автоматического управления всеми элементами системы;
– Написан скрипт на GithubActions для автоматической отправки изменений на виртуальную машину при изменении в github репозитории.

4.5 Клиентская часть

4.5.1 Структура мобильного приложения

Основные компоненты проекта по директориям:

– Класс CakerApp: инициализация приложения, конфигурация Yandex Analytics;

– Класс MainActivity: настройка базовых параметров отображения, навигация по страницам;

– Директория res: изображения, цвета, строки, темы и шрифты, доступные в любом месте в приложении, иконка приложения;

– Директория core: темы, утилиты, модели, общие для всего приложения;

– Директория di: внедрение зависимостей в приложения.

А также директории каждого из экранов, которые включают:

– Уровень presentation: создание и отображение внешнего вида приложения, возможность взаимодействия с компонентами;

– Уровень domain: сценарии использования, обрабатывающие данные пользователя при взаимодействии с базой данных;

– Уровень data: взаимодействие с БД, в том числе репозитории, обработка json;

4.5.2 Архитектура мобильного приложения

Архитектура приложения построена на основе паттерна MVVM (Model–View–ViewModel) с элементами чистой архитектуры и модульным разделением по экранам. Каждый функциональный экран реализован в виде отдельного модуля, внутри которого поддерживается единая структура: слои

presentation, domain и data. Такой подход обеспечивает изоляцию бизнес-логики, визуального слоя и реализации источников данных, что способствует масштабируемости и удобству сопровождения проекта.

В визуальном слое (presentation) реализованы экраны с использованием Jetpack Compose. Они управляются через состояние, описывающее все данные, необходимые для отображения интерфейса. Обработка действий пользователя реализуется через событийную модель: все интеракции передаются во ViewModel в виде событий, например, вызовов методов, соответствующих определённому действию. ViewModel интерпретирует эти события, обновляет состояние и обеспечивает реактивный отклик UI.

ViewModel является промежуточным звеном между UI и бизнес-логикой. Она получает события от интерфейса, вызывает соответствующие сценарии из слоя domain и обновляет состояние UI, реагируя на результат выполнения. Здесь же хранится состояние экрана, включая такие элементы, как выбранный торт, активный способ доставки или валидность введённых данных.

Бизнес-логика реализована в виде use case’ов, каждый из которых инкапсулирует один сценарий использования: например, загрузку списка доступных тортов, размещение заказа, получение доступных способов доставки и оплаты. Эти сценарии обращаются к абстрактным репозиториям, определённым в том же слое.

Интерфейсы репозиториев находятся в domain, а их реализации — в data. Реализации инкапсулируют всю работу с внешними и внутренними источниками данных. В качестве внешних источников используется REST API, доступ к которому осуществляется через Retrofit-интерфейсы, сгруппированные в папке api. Над ними строятся RemoteDataSource-классы, обрабатывающие сетевые вызовы и их ошибки. В отдельных случаях используется также LocalDataSource для временного хранения локальных данных, например, содержимого корзины или способа оплаты.

Навигация реализована на базе Jetpack Navigation. Основной NavHost управляет переходами между экранами, а параметры передаются через аргументы или общее состояние, доступное во ViewModel. Для главных разделов приложения используется нижняя панель навигации (BottomNavBar), отображающаяся только в тех экранах, которые являются частью основного пользовательского пути. Каждый модуль сам по себе не содержит логики навигации за пределы своего экрана, что обеспечивает слабую связанность между компонентами.

Отдельное внимание было уделено UX — например, при вводе банковских данных реализован автоматический переход между полями, контроль длины вводимых значений и скрытие placeholder’ов при фокусе. Все данные обрабатываются строго через слои архитектуры: UI не имеет прямого доступа к use-case’ам или репозиториям, все зависимости внедряются через Hilt.

Таким образом, архитектура приложения сочетает в себе принципы модульности, чистоты слоёв и событийно-ориентированного управления, что особенно удобно при работе над проектом, ориентированном на рост, переиспользование компонентов и добавление новых сценариев.

4.5.3 Графический интерфейс

Графический интерфейс приложения построен на логике пользовательских сценариев и back-end части.

При запуске пользователь сразу попадет на страницу входа (рисунок 9).

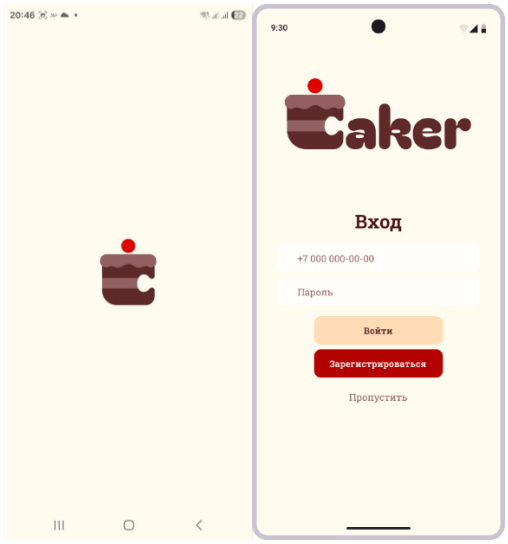

Рисунок 9 – Начальные экраны в приложении

Если пользователь выберет «Пропустить» на экране входа, то он останется незарегистрированным. У такого пользователя есть возможность просматривать основную ленту кондитеров и товаров, но нет возможности совершить заказ. Это отражено на рисунке 10.

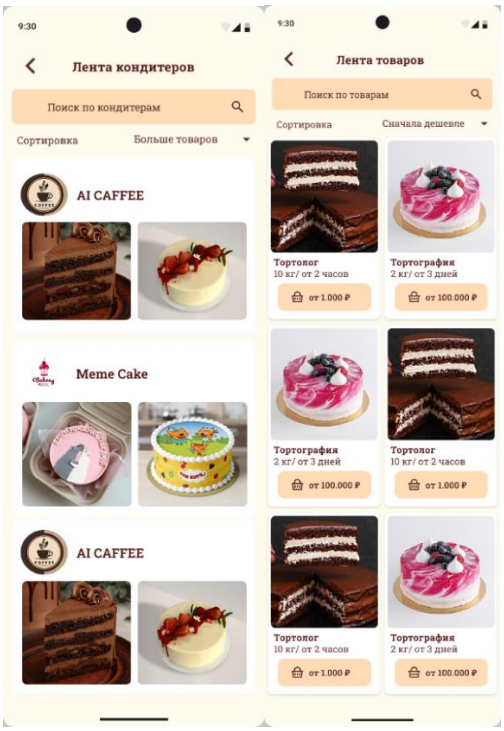

Рисунок 10 – Примеры функционала для неавторизованного пользователя

Авторизация пользователя может произойти только после регистрации, экраны авторизации показаны на рисунке 11. После этого пользователь имеет собственный профиль, возможность создавать объявления и страницу, где можно отслеживать их статус, в том случае, если пользователь выбрал быть кондитером.

Также, пользователь может видеть заказы в различных вариациях в зависимости от того, кем является пользователь, исходящие для пользователя-клиента и входящие для пользователя-кондитера, с разным статусами и отслеживанием времени выполнения заказа. Визуальная часть заявок во всех вариациях почти одинаковая, за исключением пометок о статусе.

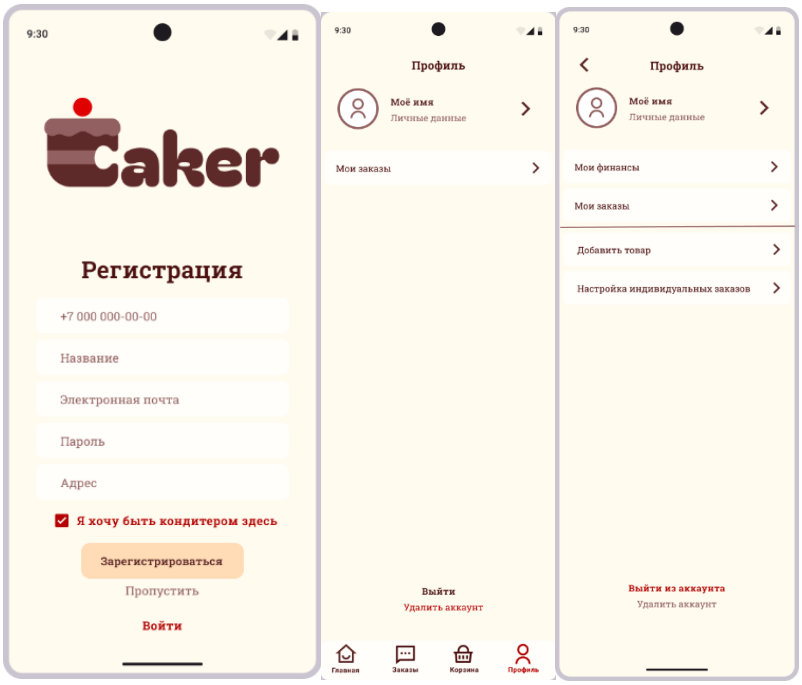

Рисунок 11 – Регистрация пользователя и его профиль

После этого пользователь-клиент может провести индивидуальный заказ с помощью конструктора торта, либо добавить товар себе в корзину. Экраны отражены на рисунке 12.

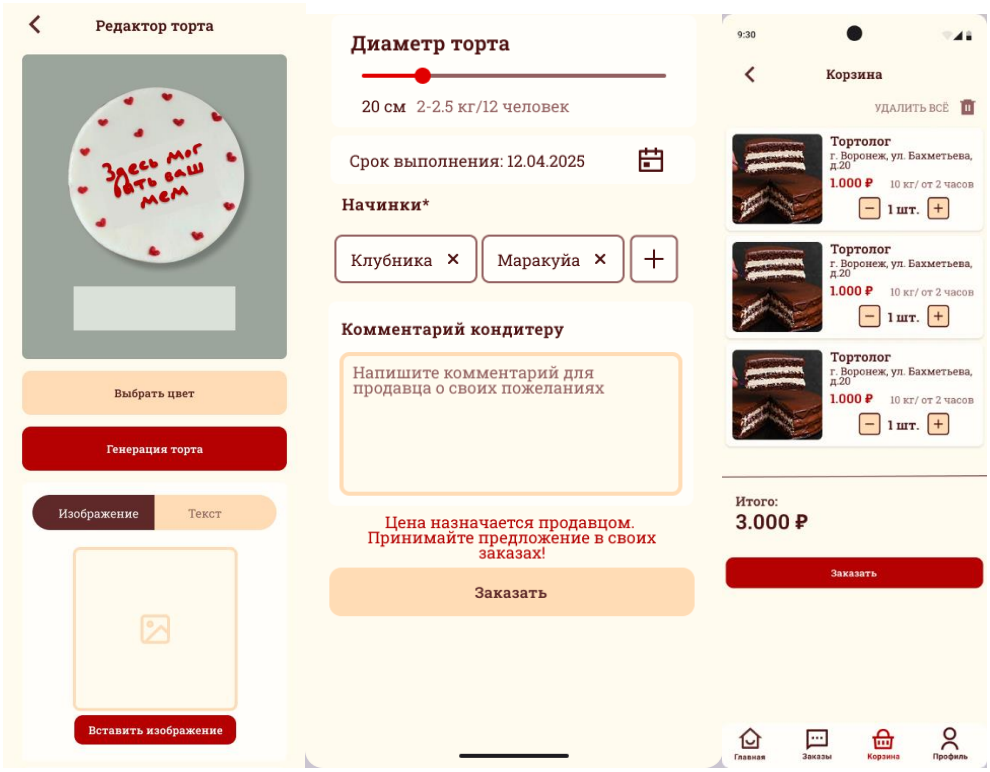

Рисунок 12 – Процесс заказа торта у клиента

Пользователь-кондитер же теперь может задавать ограничения на свои заказы, создавать новые объявления и отслеживать свои финансы, пример представлен на рисунке 13.

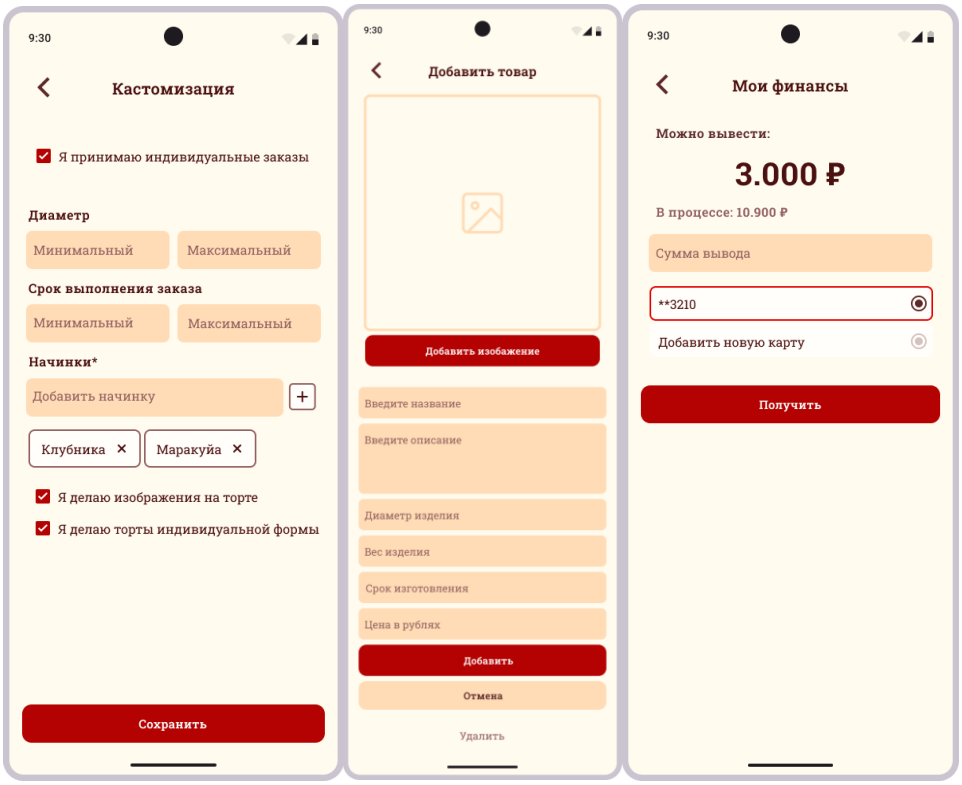

Рисунок 13 – Основные функции у кондитера

4.6 Взаимодействие компонентов при использовании

В данном параграфе будет описан основной сценарий заказа торта индивидуального дизайна приложении. Он будет описывать взаимодействие компонентов системы от входа пользователя в приложение до подтверждения от кондитера о начале работы над заказом. Для наглядной демонстрации сценарий разбит на 2 сценария:

– От лица клиента;

– От лица кондитера.

Сценарий Клиента:

Пользователь открывает приложение и выбирает понравившегося кондитера из каталога. На странице мастера он нажимает кнопку "Индивидуальный заказ", после чего переходит в конструктор тортов. В конструкторе клиент последовательно задает параметры: с помощью ползунка устанавливает диаметр (например, 25 см), выбирает вариант надписи из

предложенных шаблонов или вводит свой текст, указывает желаемую начинку из списка доступных вариантов. В специальном поле он добавляет комментарий с уточнениями по оформлению. После проверки всех параметров пользователь нажимает "Отправить заказ" и ожидает ответа кондитера. Когда приходит уведомление с предложенной ценой, клиент подтверждает заказ через интерфейс приложения.

Сценарий Кондитера:

Пользователь заранее в настройках профиля активировал функцию "Принимать индивидуальные заказы". Когда клиент отправляет запрос, кондитер получает push-уведомление о новом заказе. Открыв детали заказа в своем профиле, он видит все указанные параметры: размер, выбранную начинку, текст для надписи и комментарии клиента. Кондитер анализирует сложность работы и через интерфейс приложения рассчитывает стоимость, вводя сумму в специальное поле. После отправки цены клиенту, мастер получает подтверждение о согласии с условиями. Теперь в разделе "Мои заказы" статус меняется на "В работе", что означает начало приготовления торта по индивидуальному дизайну.

4.7 Тестирование

В ходе тестирования системы были проведены различные виды проверок:

- Функциональное тестирование;
- Тестирование удобства использования (UX/UI);
- API сервера.

Функциональное тестирование проводилось на устройстве Samsung Galaxy S23 Ultra и Samsung Galaxy A31 с установленной операционной системой Android 15 и Android 12 соответственно и осуществлялось без доступа к исходному коду, что соответствует подходу «черного ящика», при

котором тестировщик взаимодействует с мобильным приложением как обычный пользователь. В рамках проверки были протестированы ключевые функции, включая авторизацию, регистрацию пользователя, создание и оплата заказа, создание товара кондитером и установка параметров индивидуального заказа кондитером.

UX/UI тестирование является важным этапом проверка качества интерфейса и общего пользовательского взаимодействия с мобильным приложением. UI тестирование было направлено на проверку корректности отображения визуальным элементов — таких как кнопки, поля ввода, шрифты, цвета. Все основные экраны отобразились корректно и элементы интерфейса функционировали согласно требованиям. UX тестирование — было сосредоточено на оценке удобства использования и логики взаимодействия с приложением. Было устранено несколько багов, связанных с отображением интерфейса. Например, был устранен баг, при котором системные кнопки устройства перекрывали кнопки интерфейса. Сейчас элементы интерфейса доведены до рабочего состояния, отображаются корректно и функционируют согласно требованиям.

Тестирование API проводилось с использованием ручного тестирования через Postman. Проверены ключевые эндпоинты: аутентификация, регистрация, создание заказа и изменение параметров пользователя. API неоднократно переписывалось, поэтому тестировалось на поздних этапах разработки. В итоге API удалось довести до рабочего состояния.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы был успешно разработан сервис для заказа индивидуальных тортов «Caker», полностью соответствующий поставленным целям и задачам. Созданная система представляет собой комплексное решение, включающее мобильное приложение для Android и серверную часть на основе микросервисной архитектуры, которое предоставляет клиентам возможность создавать уникальные дизайны тортов через интерактивный конструктор, а кондитерам - эффективно управлять своими предложениями и заказами. В процессе разработки была организована слаженная командная работа с использованием современных инструментов управления проектами, проведен глубокий анализ рынка и пользовательских потребностей, что позволило создать оптимальную архитектуру системы и удобные интерфейсы.

Реализованный функционал охватывает все этапы взаимодействия пользователей с платформой, начиная от регистрации и заканчивая получением готового изделия и оставлением отзывов. Тестирование системы подтвердило ее высокую эффективность - пользователи отметили интуитивную понятность интерфейса, значительное сокращение времени оформления заказа по сравнению с традиционными методами, а также достижение всех запланированных показателей по количеству и качеству выполняемых заказов. Применение микросервисной архитектуры обеспечило проекту необходимую гибкость, масштабируемость и потенциал для дальнейшего развития.

Таким образом, «Caker» представляет собой готовое к коммерческому использованию решение, сочетающее в себе инновационные технологии с глубоким пониманием потребностей рынка индивидуальных кондитерских изделий. Проект демонстрирует значительный потенциал для расширения функциональности и завоевания устойчивой позиции в своей нише, предлагая

пользователям принципиально новый уровень удобства при заказе кастомных тортов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация YandexCloud [Электронный ресурс]. – Режим доступа: URL:https://yandex.cloud/ru/docs/tutorials - Заглавление с экрана. – (Дата обращения 21.05.2025);

2. Ensemble UI Documentation: Invoke API Action [Электронный ресурс]. – Режим доступа: URL: https://docs.ensembleui.com/actions/invoke-API – Заглавление с экрана. – (Дата обращения: 24.05.2025).

3. Kotlin Programming Language Documentation [Электронный ресурс]. – Режим доступа: URL: https://kotlinlang.org/docs/home.html – Заглавление с экрана. – (Дата обращения: 24.05.2025).

4. Retrofit: A type-safe HTTP client for Android and Java [Электронный ресурс]. – Режим доступа: URL: https://square.github.io/retrofit/ – Заглавление с экрана. – (Дата обращения: 24.05.2025).

5. Android Developers: Jetpack Compose Documentation [Электронный ресурс]. – Режим доступа: URL:https://developer.android.com/develop/ui/compose/documentation?hl=ru – Заглавление с экрана. – (Дата обращения: 26.05.2025).

ПРИЛОЖЕНИЕ А

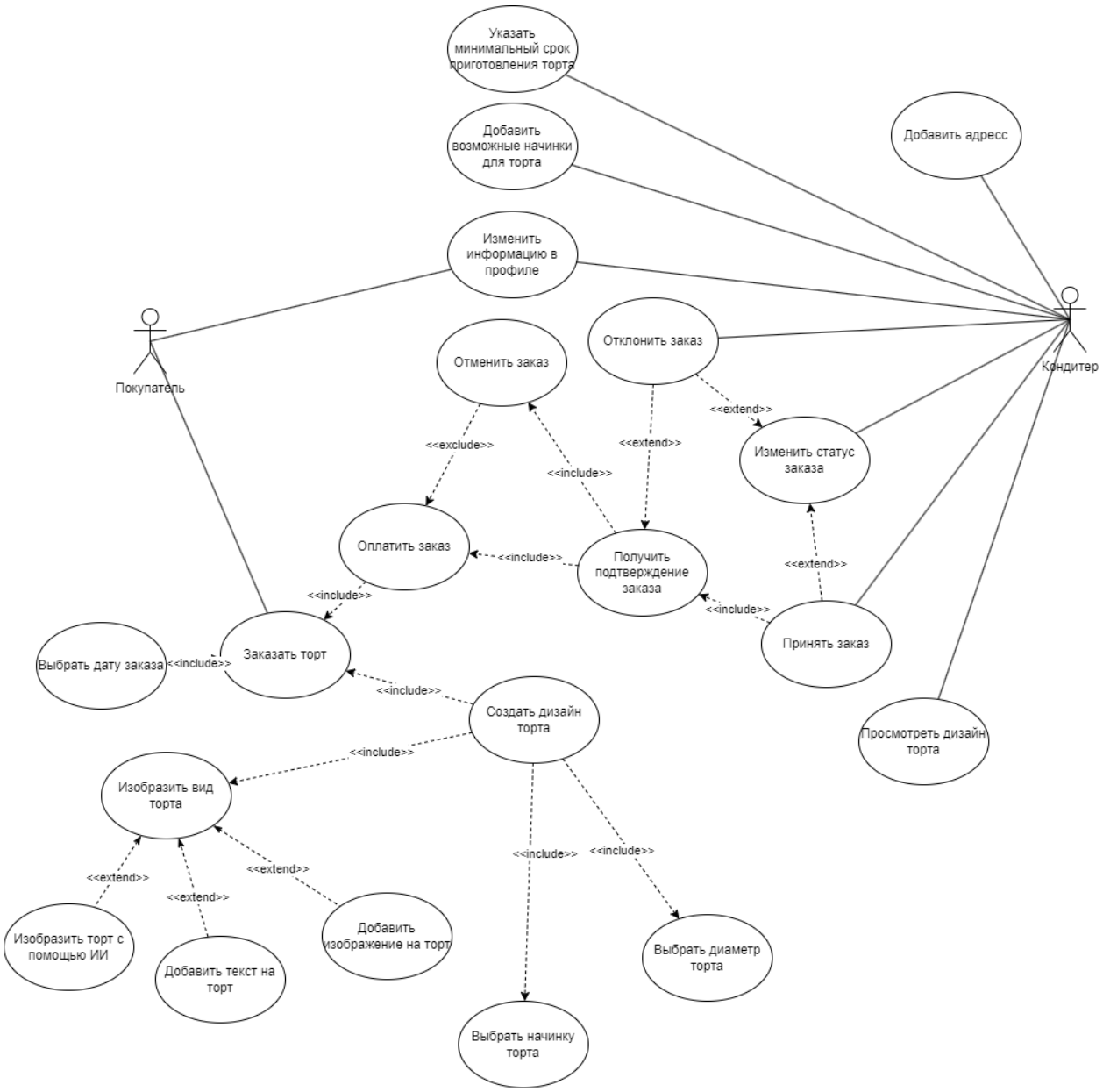

Рисунок 14 – Диаграмма использования приложения

ПРИЛОЖЕНИЕ Б

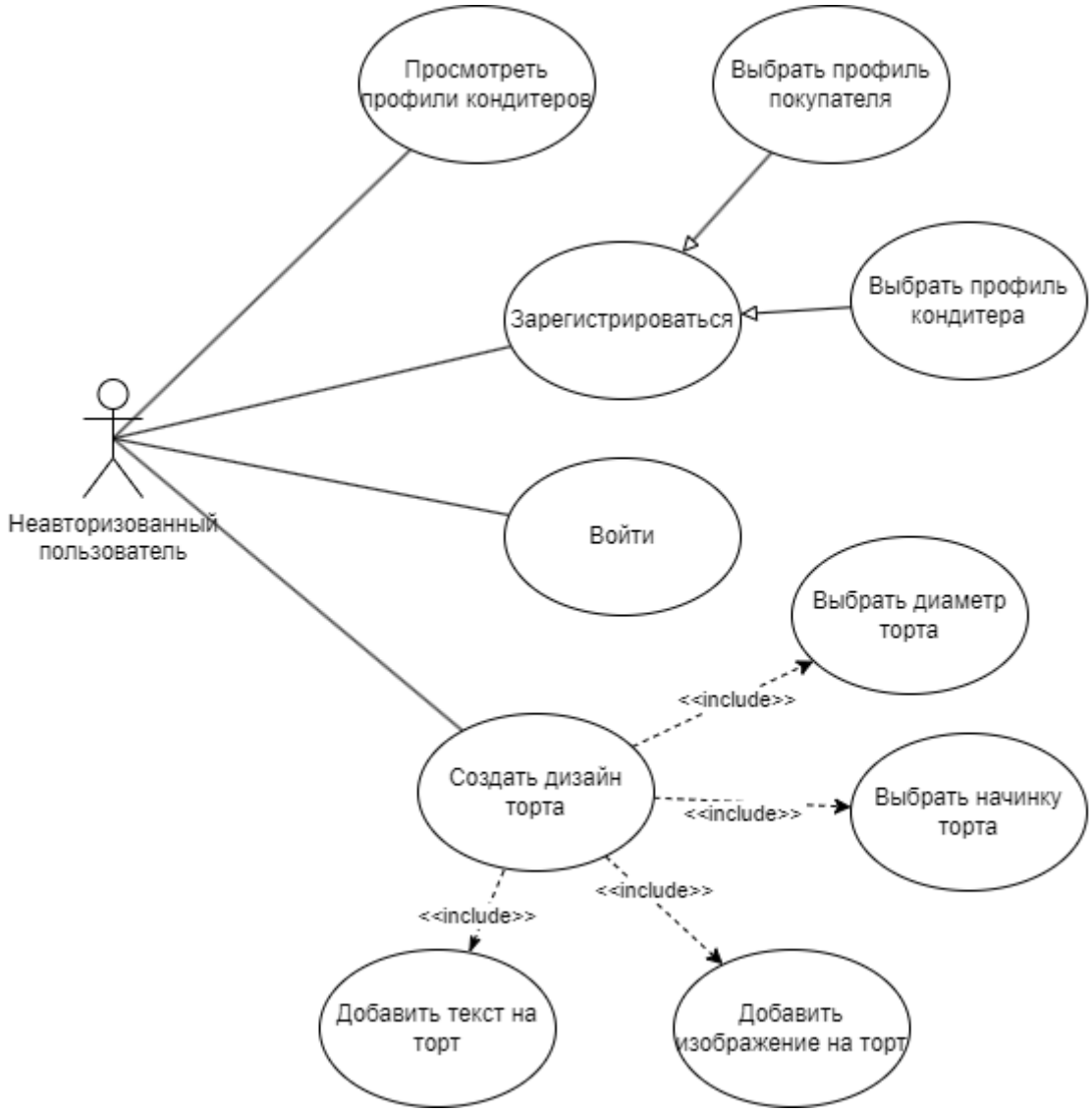

Рисунок 15 – Диаграмма для неавторизованного пользователя