Национальная научно образовательная корпорация ИТМО
Факультет программной инженерии и компьютерной техники

Проект Gulaii

по дисциплине
«Разработка мобильных приложений»

Выполнили:
Касьяненко Вера
Кремпольская Екатерина
Дворкин Борис
Барсуков Максим
Вальц Мартин
Мальков Павел
Лисейчиков Глеб
Демурчян Владимир
Преподаватель:
Ключев Аркадий Олегович

Санкт-Петербург, 2025

Введение

Актуальность

Современные реалии диктуют новый подход к вопросам здоровья и образа жизни. Малоподвижность, неправильное питание, высокий уровень стресса — всё это приводит к ухудшению физического состояния и росту хронических заболеваний. При этом наблюдается значительный рост интереса к темам здорового образа жизни (ЗОЖ), осознанного питания и спортивной активности.

Мобильные технологии становятся основным инструментом самоконтроля и мотивации. Пользователи стремятся отслеживать потребление калорий, следить за балансом БЖУ, заниматься спортом и видеть реальные результаты. Однако многие приложения либо перегружены, либо фокусируются на узких функциях.

Создание компактного и интуитивного приложения с ключевыми возможностями — такими как контроль питания, учёт физической активности и мониторинг массы тела — может эффективно закрыть основные потребности большинства пользователей. Простота интерфейса, персонализация и визуализация прогресса делают такой продукт особенно актуальным для широкой аудитории.

Актуальность разработки мобильного приложения

Мобильные устройства стали неотъемлемой частью повседневной жизни, и именно через них пользователи всё чаще решают задачи, связанные с контролем здоровья и физического состояния. Разработка мобильного приложения для ЗОЖ актуальна потому, что оно обеспечивает постоянный доступ к функциональности, позволяя удобно отслеживать питание, физическую активность и динамику массы тела. В отличие от веб-решений, мобильное приложение обеспечивает более тесную интеграцию с устройством, например, для автоматического подсчета шагов и отправки уведомлений. Кроме того, мобильный формат соответствует ожиданиям целевой аудитории, обеспечивая простоту.

Цель

Создать мобильное приложение, которое позволит пользователям следить за рационом питания, уровнем физической активности и динамикой массы тела, тем самым способствуя формированию здоровых привычек и достижению личных целей по улучшению физической формы.

Задачи

1) Изучить потребности целевой аудитории и существующие решения на рынке мобильных ЗОЖ-приложений.
2) Сформировать техническое задание, включающее описание функциональных и нефункциональных требований.

3) Разработать архитектуру приложения с использованием микросервисного подхода.

4) Реализовать функционал по учёту питания:
   - Подсчёт калорий;
   - Контроль состава продуктов по БЖУ;
   - Добавление приемов пищи.

5) Реализовать функции отслеживания физической активности:
   - Учёт шагов;
   - Ввод тренировок.

6) Реализовать мониторинг роста и веса.

7) Разработать мобильный клиент на платформе Android.

8) Реализовать серверную часть с микросервисами:
   - авторизация;
   - работа с данными питания и активности;
   - ведение логов;
   - работа с базой данных.

9) Реализовать нагрузочное тестирование, имитирующее работу до 10 000 пользователей.

Техническое задание

Введение

Приложение ориентировано на пользователей, желающих контролировать своё питание, отслеживать физическую активность и следить за изменениями массы тела и роста.

Система должна обеспечивать стабильную работу на устройствах под управлением Android, предоставляя интуитивно понятный интерфейс, точный учёт данных и визуализацию прогресса. Также приложение должно быть способно обрабатывать высокую нагрузку, соответствующую количеству активных пользователей до 10 000 человек.

В рамках данного проекта планируется реализация клиентской и серверной части, а также архитектуры, построенной на микросервисном подходе, обеспечивающем масштабируемость, отказоустойчивость и гибкость системы.

Назначение разработки

Разрабатываемое мобильное приложение предназначено для пользователей, стремящихся к улучшению физической формы и поддержанию здорового образа жизни. Приложение предоставляет интерфейс для учёта калорий и состава пищи, регистрации физической активности и отслеживания изменений роста и веса. Оно позволяет анализировать динамику показателей и достигать поставленных целей, обеспечивая при этом простоту взаимодействия.

Основные направления назначения

Мобильное приложение предназначено для комплексной поддержки пользователей, стремящихся вести здоровый образ жизни. Основные направления его назначения включают:

- Учёт питания — приложение позволяет пользователям фиксировать приёмы пищи, отслеживать потребление калорий и соотношение БЖУ, анализировать рацион и корректировать его в соответствии с личными целями.
- Контроль физической активности — пользователи могут вручную добавлять тренировки и другие показатели активности.
- Мониторинг массы тела и роста — реализована возможность фиксировать параметры тела.

Приложение служит персональным помощником для тех, кто хочет системно подходить к вопросам физического состояния и питания, при этом не перегружаясь лишним функционалом.

Состав продукта

В состав разрабатываемого программного продукта входят следующие компоненты:

1. Мобильное приложение (клиентская часть). Нативное Android-приложение, разработанное на языке Kotlin с использованием Jetpack Compose. Приложение предоставляет пользовательский интерфейс для регистрации данных о питании, активности, весе и росте.
2. Бэкенд (серверная часть). Серверное программное обеспечение, реализованное на Kotlin. Обеспечивает обработку запросов, бизнес-логику, взаимодействие с базами данных и внешними сервисами. Используется микросервисная архитектура.
3. API Gateway. Централизованный шлюз, через который проходит весь пользовательский трафик. Выполняет маршрутизацию запросов.
4. Микросервисы:
    - User Service — хранение и обработка пользовательских данных, включая рост, вес и цели.
    - Nutrition Service — учёт приёмов пищи, подсчёт калорий и расчёт баланса БЖУ.
    - Activity Service — фиксация физической активности: шаги, тренировки, потраченные калории.
    - Log Service — сбор и хранение логов событий, ошибок и пользовательских действий.
    - Load Service — генерация нагрузочного трафика для тестирования производительности системы.
5. Брокер сообщений (KeyDB). Request/Response через очереди, Publish/Subscribe (Pub/Sub) для событийного взаимодействия.
6. Базы данных: PostgreSQL и ClickHouse.

7. Сборка и доставка. Приложение распространяется в виде .apk-файла для установки на устройства с Android. Сборка осуществляется с помощью Gradle.

Этот состав обеспечивает полнофункциональную, масштабируемую и устойчивую к нагрузкам систему, ориентированную на широкую пользовательскую аудиторию.

Функциональные требования

1. Отслеживание физиологических параметров пользователя (реализуется через User Service)

1.1 Указание и фиксация антропометрических данных

1.1.1 Возможность ручного ввода текущего веса и роста

1.1.2 Просмотр истории изменений веса и роста

1.1.3 Указание целевого веса и предпочтений (снижение, поддержание, увеличение массы)

2. Учёт питания (реализуется через Nutrition Service)

2.1 Ведение пищевого дневника

2.1.1 Добавление продуктов и блюд вручную или из библиотеки

2.1.2 Расчёт потреблённых калорий, белков, жиров и углеводов

2.1.3 Просмотр записей приёмов пищи

3. Учёт физической активности (реализуется через Activity Service)

3.1 Регистрация активности

3.1.1 Ручной ввод тренировок с параметрами

3.1.3 Расчёт потраченных калорий по активности

3.1.4 Отображение истории активности

4. Регистрация и вход в систему

4.1 Процесс аутентификации

4.1.1 Регистрация нового пользователя с логином и паролем

4.1.2 Вход в систему по email и паролю

4.1.3 Выдача JWT-токена при успешной авторизации

4.1.4 Проверка токена при каждом запросе

4.1.5 Выход из системы и отзыв токена (инвалидация)

5. Обработка запросов через API Gateway (реализуется через API Gateway)

5.1 Централизованная маршрутизация запросов

5.1.1 Приём всех HTTP-запросов от клиента

5.1.2 Аутентификация запросов через JWT

5.1.3 Публикация запросов в брокер сообщений KeyDB

5.1.4 Получение и агрегация ответов от микросервисов

5.1.5 Логирование всех операций и мониторинг доступности

6. Логирование и мониторинг (реализуется через Log Service)

6.1 Работа с логами

6.1.1 Приём логов от всех микросервисов

6.1.2 Хранение логов в базе данных

7. Нагрузочное тестирование (реализуется через Load Service)

7.1 Проведение стресс-тестов

7.1.1 Генерация синтетических пользователей

7.1.2 Запуск типовых сценариев с высокой частотой

7.1.3 Измерение отклика микросервисов

Нефункциональные требования

1. Масштабируемость

1.1 Горизонтальное масштабирование

1.1.1 Каждый микросервис должен быть масштабируем независимо от других

2. Отказоустойчивость

2.1 Устойчивость к сбоям

2.1.1 Система должна продолжать функционировать при сбое одного или нескольких микросервисов

3. Безопасность

3.1 Аутентификация и авторизация

3.1.1 Все запросы от пользователей должны проходить проверку JWT-токена

3.1.2 Хранение паролей пользователей только в хэшированном виде (алгоритм bcrypt или аналогичный)

Архитектура системы

Система реализована на основе микросервисной архитектуры с использованием асинхронного взаимодействия через брокер сообщений. Каждый функциональный компонент системы представлен независимым микросервисом, взаимодействующим с остальными сервисами через централизованный API Gateway и брокер KeyDB. Это обеспечивает масштабируемость, отказоустойчивость и гибкость при разработке и сопровождении системы.

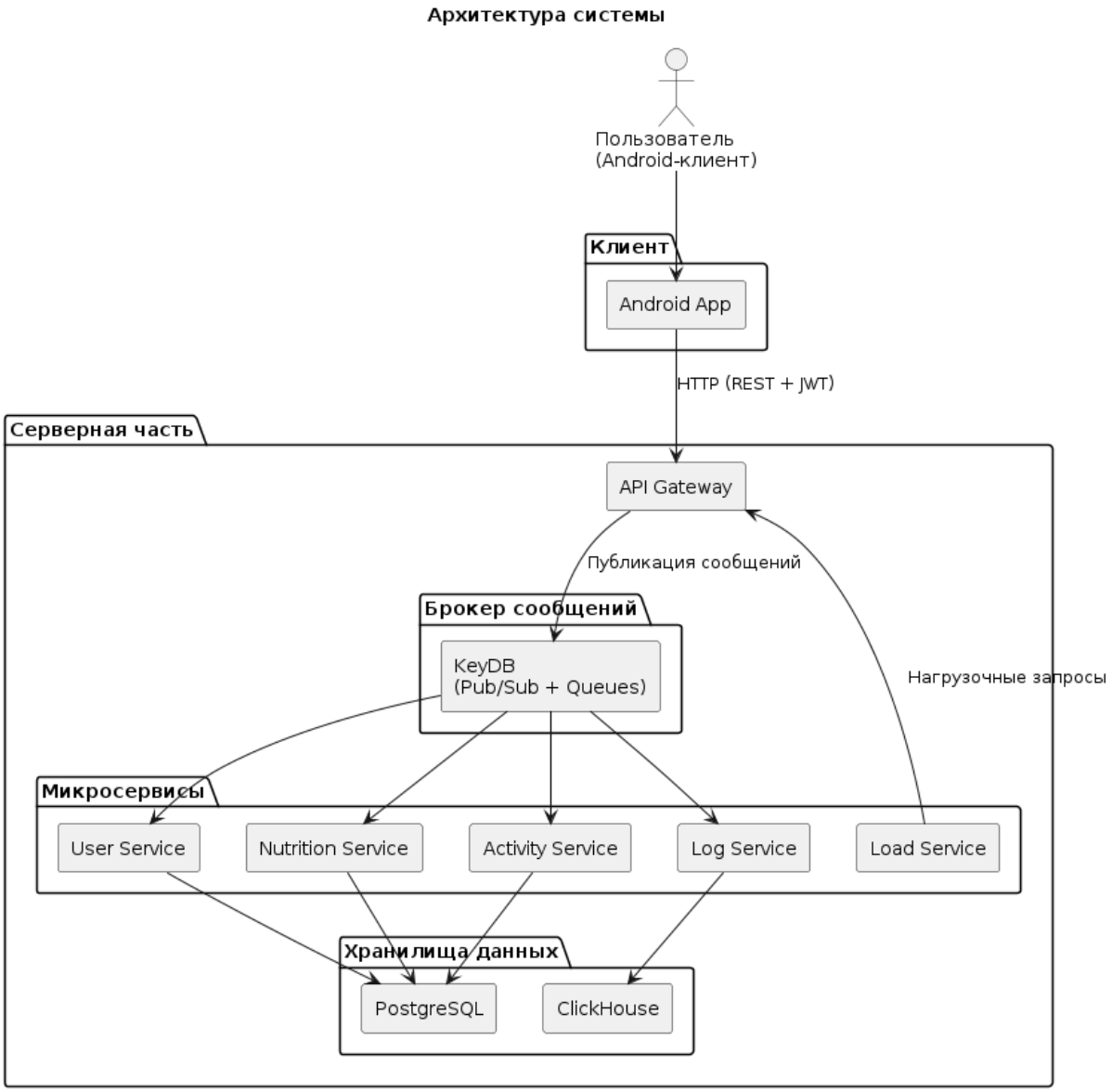

Рисунок 1

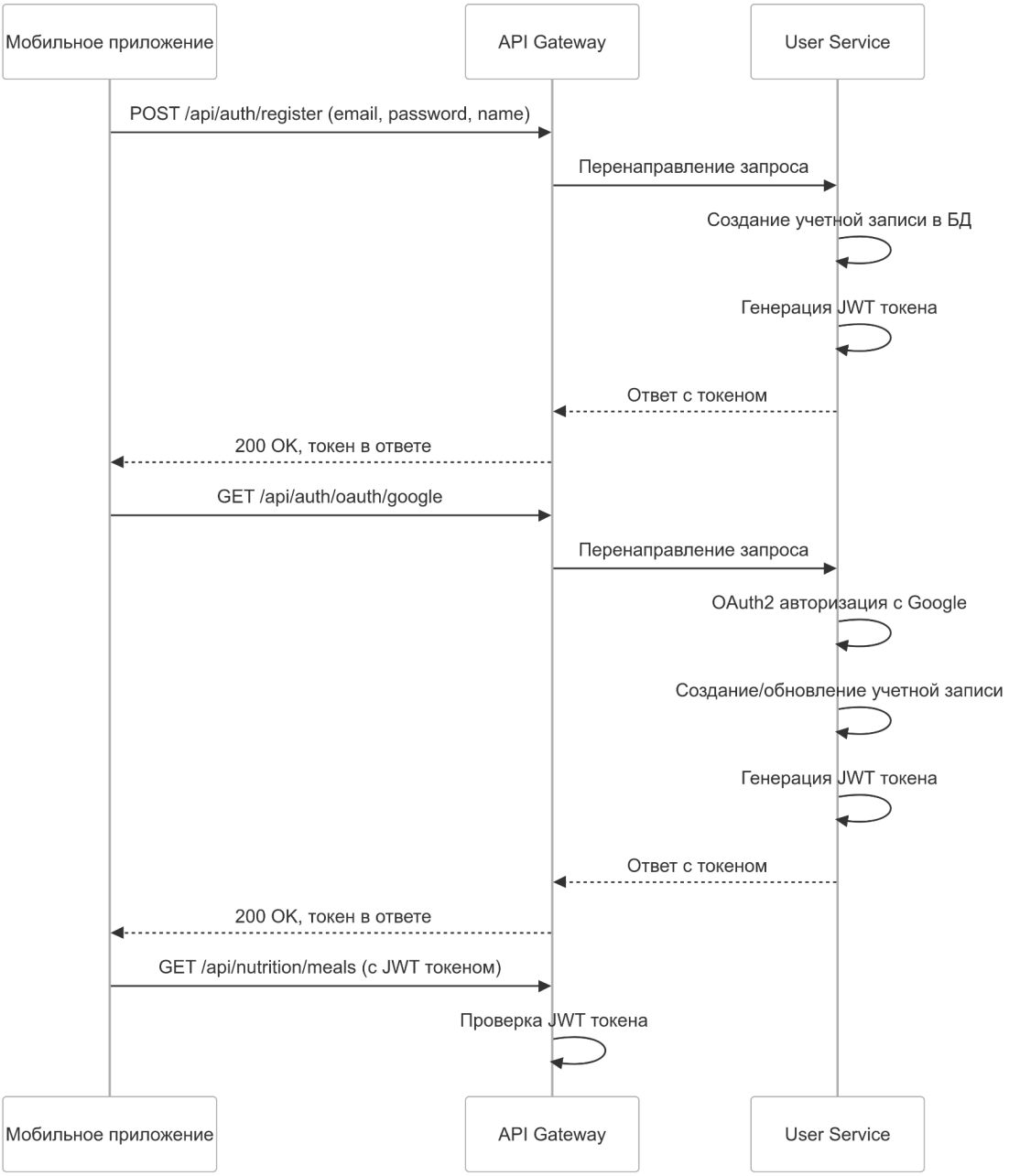

Рисунок 2

Описание реализации

1. API Gateway

Путь: src/main/kotlin/gateway

Пакеты и классы:

- config/
  - AppConfig.kt — базовая конфигурация сервиса (порты, зависимости, очереди).
- plugins/
  - AuthPlugin.kt — плагин валидации JWT-токена, извлекает userId.
  - ProxyPlugin.kt — логика маршрутизации входящих HTTP-запросов.

- service/
  - ProxyService.kt — преобразует HTTP-запросы в сообщения и отправляет в KeyDB.
  - AuthService.kt — проверка валидности токена.
  - LoggingService.kt — отправка логов в Log Service.
- Main.kt — точка входа, инициализация приложения.

2. User Service

Путь: src/main/kotlin/user

Пакеты и классы:

- model/
  - User.kt, Profile.kt, AuthRequest.kt — модели пользователя и запросов.
- service/
  - UserService.kt — операции с профилем (рост, вес, цели).
  - AuthManager.kt — генерация и валидация JWT-токенов.
- repository/
  - UserRepository.kt — взаимодействие с PostgreSQL (таблицы users, goals, sessions).
- routes/
  - UserRoutes.kt — обработка запросов: регистрация, авторизация, обновление профиля.
- Main.kt — конфигурация маршрутов и запуск сервиса.

3. Nutrition Service

Путь: src/main/kotlin/nutrition

Пакеты и классы:

- model/
  - FoodItem.kt, Meal.kt, NutritionLog.kt — описание приёмов пищи и нутриентов.
- service/
  - NutritionService.kt — логика учёта приёмов пищи, подсчёт КБЖУ.
- repository/
  - NutritionRepository.kt — взаимодействие с PostgreSQL (таблицы meals, food_items).
- routes/
  - NutritionRoutes.kt — обработка входящих запросов: добавить еду, получить отчёт.
- Main.kt — запуск приложения, настройка очередей и брокера.

4. Activity Service

Путь: src/main/kotlin/activity

Пакеты и классы:

- model/
  - StepLog.kt, Workout.kt, ActivityLog.kt — модели активности.
- service/
  - ActivityService.kt — логика подсчёта шагов, тренировок, калорий.
- repository/
  - ActivityRepository.kt — запросы к PostgreSQL (таблицы step_counts, workout_sessions).
- routes/
  - ActivityRoutes.kt — добавить активность, получить статистику.
- Main.kt — регистрация сервисов, очередь KeyDB, старт сервиса.

5. Log Service

Путь: src/main/kotlin/logs

Пакеты и классы:

- model/
  - LogEntry.kt — описание структуры лога (время, пользователь, уровень, сообщение).
- service/
  - LogProcessor.kt — обработка входящих логов из очереди KeyDB.
  - AnalyticsExporter.kt — сохранение логов в ClickHouse.
- repository/
  - ClickHouseLogger.kt — подключение и запись в БД.
- Main.kt — подписка на очереди, прослушка событий, логика запуска.

6. Load Service

Путь: src/main/kotlin/loadtest

Пакеты и классы:

- scenario/
  - UserScenario.kt, MealScenario.kt, StepScenario.kt — типовые действия пользователей.
- service/
  - LoadGenerator.kt — запуск параллельных корутин с запросами к API Gateway.
  - MetricsCollector.kt — сбор данных по времени ответа и количеству ошибок.
- Main.kt — параметры нагрузки, запуск сценариев, логирование результатов.

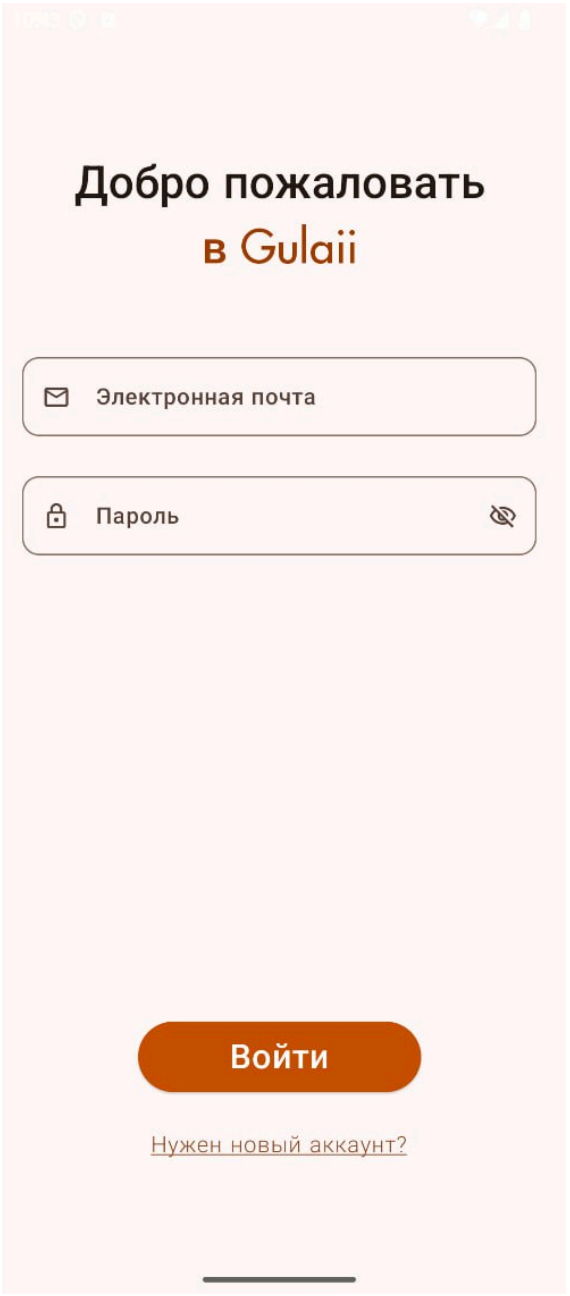
Добро пожаловать
в Gulaii
Электронная почта
Пароль
Войти
Нужен новый аккаунт?

Рисунок 3

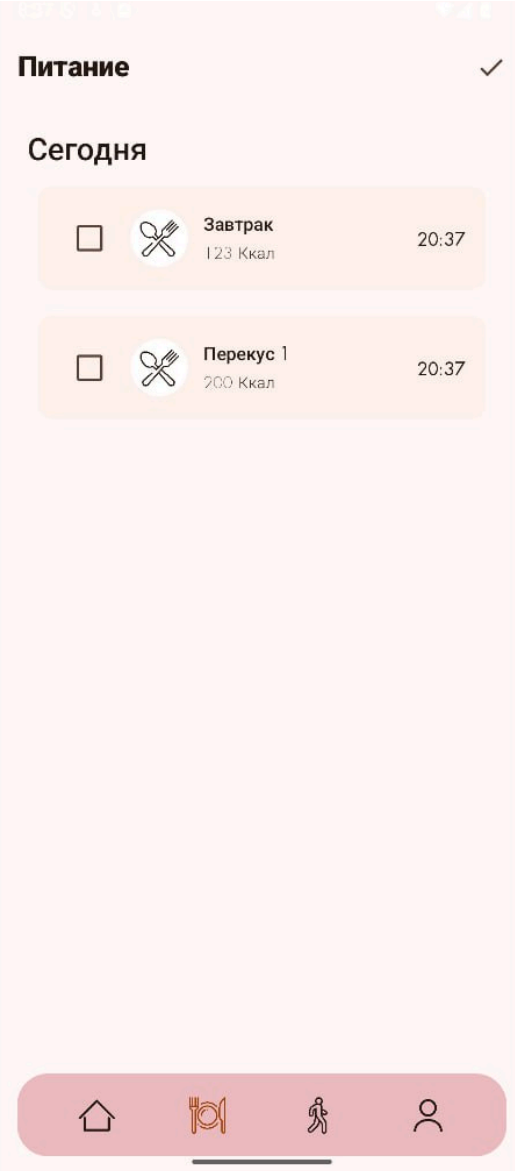
Питание
Сегодня
Завтрак
123 Ккал
20:37
Перекус 1
200 Ккал
20:37

Рисунок 4

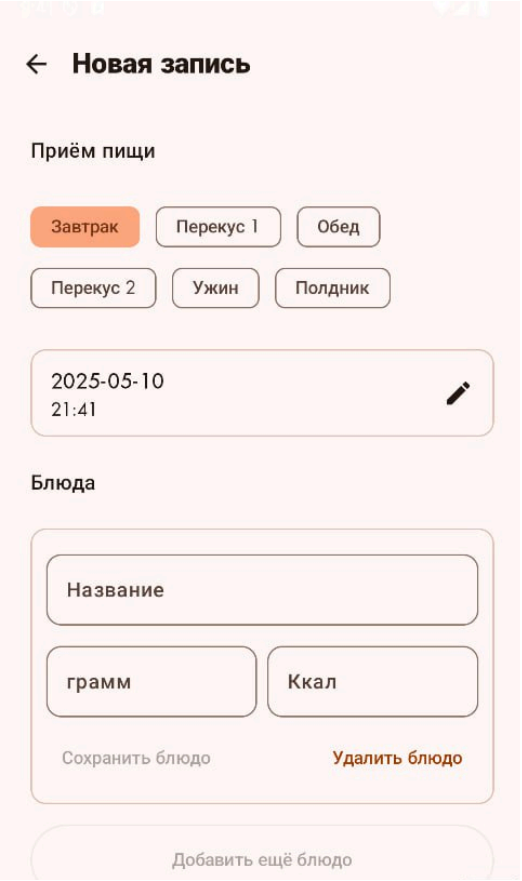

Рисунок 5

Рисунок 6

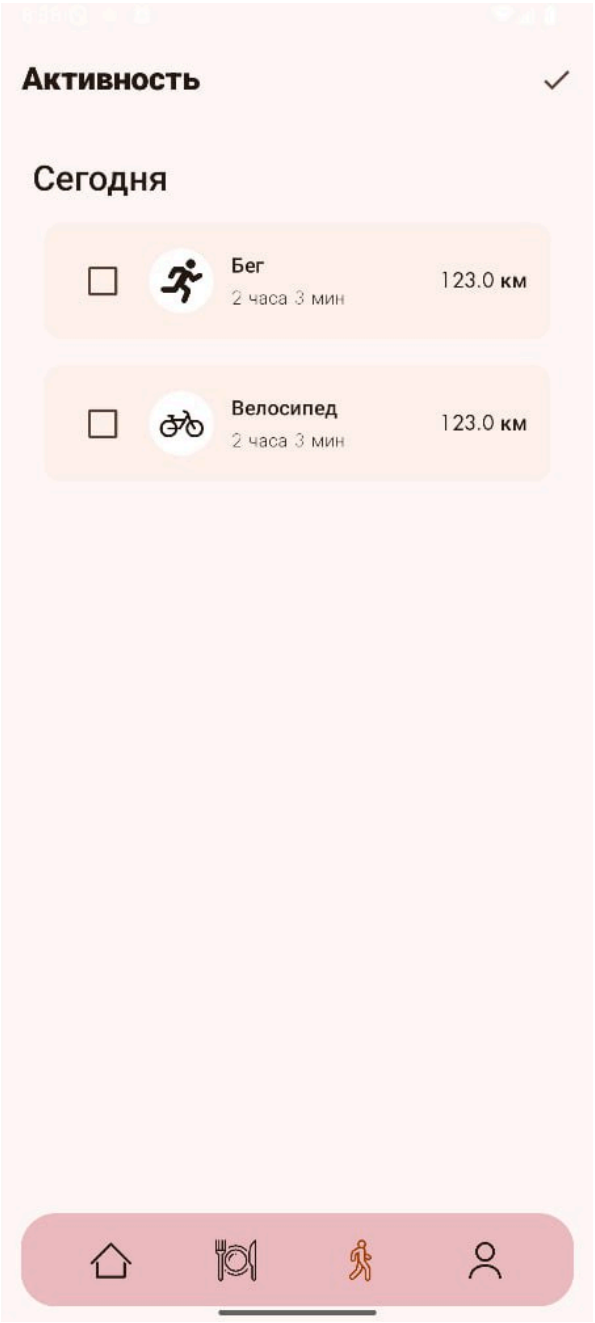
Активность
Сегодня
Бег
2 часа 3 мин
123.0 км
Велосипед
2 часа 3 мин
123.0 км

Рисунок 7

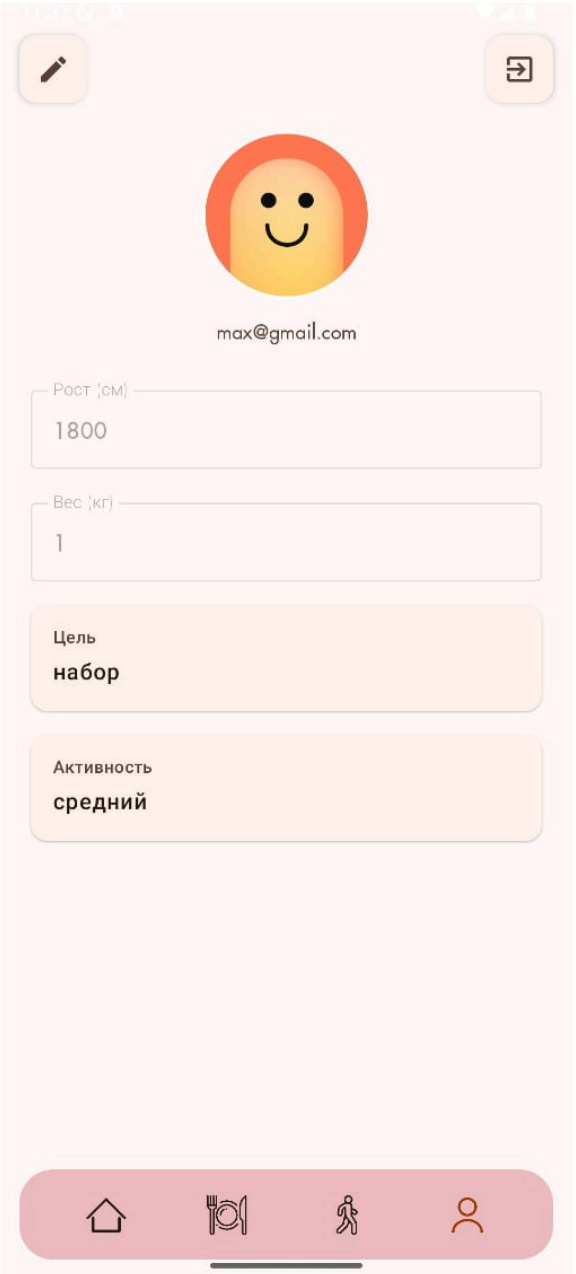

Рисунок 9

Тестирование

В рамках контроля качества системы были проведены следующие виды тестирования:

1.Нагрузочное тестирование. Эмуляция параллельной работы 10000 пользователей.

2. Модульное тестирование. Тестирование модулей приложений.

3. Ручное тестирование. После выполнения задач проводилось ручное тестирование с целью обнаружения необработанных сценариев использования.

Заключение

Техническая спецификация описывает архитектуру, функциональные и нефункциональные требования для мобильного приложения — платформы для поддержки здорового образа жизни с акцентом на учёт питания, физической активности, роста и веса. В основе реализации лежит микросервисная архитектура с использованием асинхронного обмена сообщениями через брокер, что обеспечивает масштабируемость, модульность и отказоустойчивость системы.

На текущем этапе реализованы ключевые микросервисы: User Service, Nutrition Service, Activity Service, Log Service и Load Service, а также централизованный API Gateway.

Выбранные технологии (Kotlin, PostgreSQL, ClickHouse) позволяют обеспечить высокую производительность, надёжность и удобство сопровождения. Система спроектирована с учётом расширяемости — возможна интеграция новых модулей (например, уведомлений, маршрутов, социальной активности) без изменений в существующих компонентах.

Реализация проекта закладывает основу для масштабной экосистемы цифрового здоровья, адаптированной под нужды городских пользователей, и способной развиваться в сторону персонализированных рекомендаций, взаимодействия с устройствами и социальной вовлечённости.