Спецификация GULAII

Платформа для городской активности и здорового образа жизни

Документ: Техническая спецификация системы
Версия: 1.0
Дата: 10.03.2025

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

1.1. Назначение документа

Настоящий документ представляет собой техническую спецификацию микросервисной архитектуры системы Gulaii. Документ разработан в соответствии с ГОСТ 34.602-89 и ГОСТ 19.201-78 и предназначен для использования командой разработки, аналитиками и другими заинтересованными лицами в рамках проекта.

1.2. Цели и задачи системы

Система Gulaii представляет собой платформу для поддержания здорового образа жизни, сочетающую мониторинг питания с городскими исследованиями. Платформа предоставляет персонализированные маршруты для пользователей на основе их пищевых предпочтений, дневного потребления калорий и доступного времени, превращая традиционные упражнения в увлекательное исследование города.

Основные задачи системы:

- Учет потребления пищи и нутриентов
- Мониторинг физической активности
- Генерация персонализированных городских маршрутов
- Предоставление рекомендаций по здоровому образу жизни
- Обеспечение социального взаимодействия пользователей

1.3. Область применения

Система Gulaii предназначена для жителей городов, заботящихся о своем здоровье и желающих исследовать городское пространство. Приложение может быть использовано как частными лицами, так и городскими администрациями, туристическими сервисами и локальными бизнесами.

2. ОБЩЕЕ ОПИСАНИЕ СИСТЕМЫ

2.1. Концепция Gulaii

Gulaii трансформирует традиционный подход к фитнес-трекингу, объединяя мониторинг питания с городскими исследованиями. Вместо того чтобы концентрироваться на статистике и создавать чувство обязательства, Gulaii предлагает позитивный подход: “Поел вкусный обед? Открой новый уголок города и получи новые впечатления, а здоровье улучшится само собой”.

Система создает персонализированные маршруты на основе:

- Потребленных калорий
- Доступного времени пользователя
- Интересов и предыдущих маршрутов
- Погодных условий и времени суток

2.2. Обзор архитектуры

Gulaii построен на микросервисной архитектуре с использованием асинхронного взаимодействия через брокер сообщений.

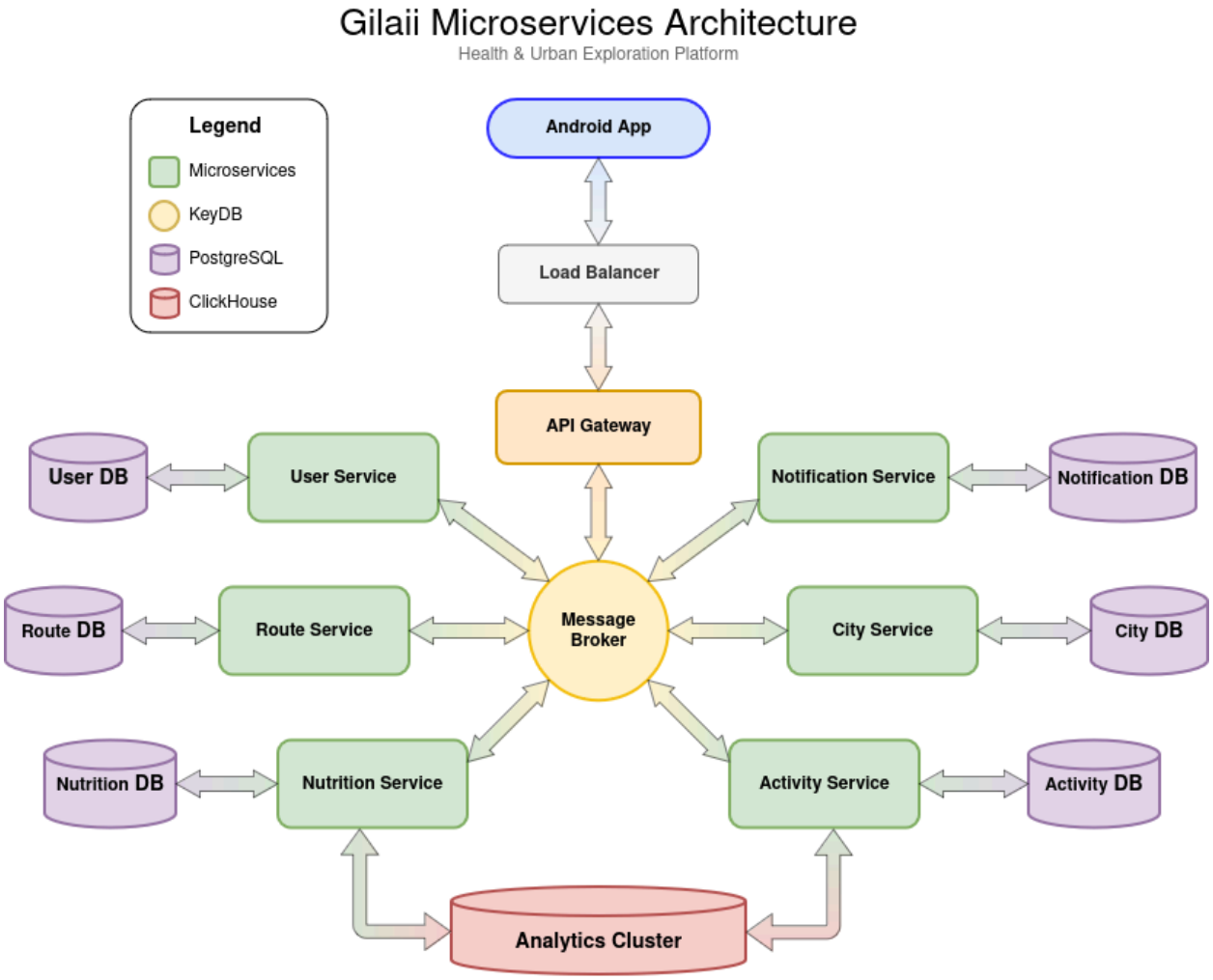


Рисунок 1. Микросервисная архитектура системы Gulaii

Система состоит из следующих основных компонентов:

1. **Android-приложение** – клиентская часть для конечных пользователей
2. **API Gateway** – входная точка для всех клиентских запросов
3. **Микросервисы** – специализированные сервисы для различных функций

4. **Брокер сообщений** – для асинхронного обмена данными между компонентами
5. **Базы данных** – индивидуальные для каждого микросервиса
6. **Аналитический кластер** – для обработки больших объемов данных

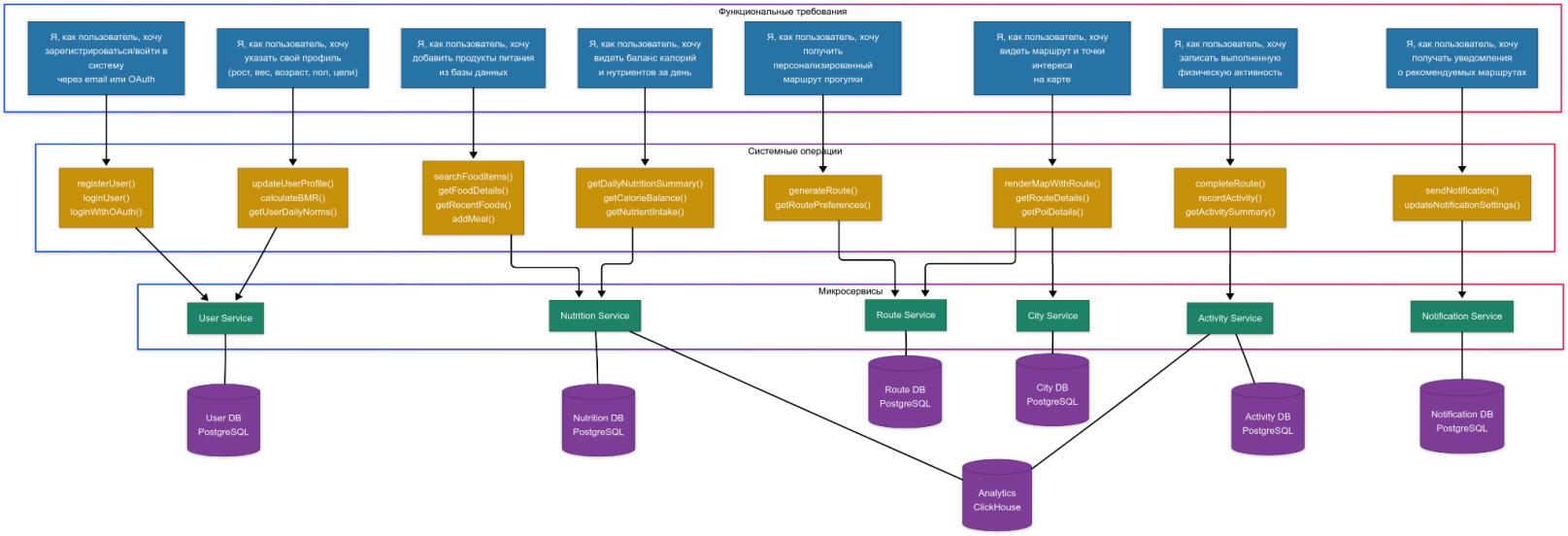


Рисунок 2. Трехшаговый процесс описания микросервисной архитектуры

2.3. Компоненты системы

Система включает следующие основные компоненты:

1. **Android-приложение** – нативное мобильное приложение, разработанное на Kotlin с использованием Jetpack-compose.

2. ***Load Balancer** – распределяет нагрузку между серверными компонентами.

3. **API Gateway** – единая точка входа для клиентских запросов, осуществляющая маршрутизацию, авторизацию и взаимодействие с брокером сообщений.

4. **Микросервисы**:

    - User Service – управление учетными записями пользователей
    - Nutrition Service – учет питания и нутриентов
    - Activity Service – отслеживание физической активности
    - Route Service – генерация и управление маршрутами
    - City Service – информация о городских локациях
    - Notification Service – управление уведомлениями
5. **Брокер сообщений (KeyDB)** – центральный компонент для асинхронного обмена сообщениями между микросервисами.

6. **Базы данных PostgreSQL** – реляционные базы данных для каждого микросервиса.

7. **Analytics Cluster (ClickHouse)** – колоночная СУБД для аналитической обработки данных.

3. АРХИТЕКТУРНОЕ РЕШЕНИЕ

3.1. Обоснование выбора микросервисной архитектуры

Для системы Gulaii выбрана микросервисная архитектура с асинхронным взаимодействием через брокер сообщений по следующим причинам:

1. **Модульность и независимая разработка** – позволяет команде разрабатывать, тестировать и развертывать компоненты независимо друг от друга.

2. **Масштабируемость** – каждый сервис может масштабироваться независимо в зависимости от нагрузки.

3. **Технологическая гибкость** – при необходимости для различных микросервисов можно использовать различные технологии.

4. **Устойчивость к отказам** – изоляция сбоев в рамках отдельных сервисов, предотвращающая каскадные отказы.

5. **Асинхронная обработка** – использование брокера сообщений обеспечивает буферизацию запросов при пиковых нагрузках и устойчивость к временной недоступности сервисов.

3.2. Паттерны проектирования

В архитектуре системы Gulaii используются следующие архитектурные паттерны:

1. **API Gateway** – централизованная точка входа для всех клиентских запросов, обеспечивающая маршрутизацию и преобразование протоколов.

2. **Message Broker** – асинхронное взаимодействие между сервисами через централизованный брокер сообщений.

3. **Database per Service** – каждый микросервис имеет свою собственную базу данных.

4. **Circuit Breaker** – предотвращает каскадные отказы при недоступности сервисов.

5. **CQRS (Command Query Responsibility Segregation)** – разделение операций чтения и записи для повышения производительности.

6. **Event Sourcing** – хранение изменений состояния в виде последовательности событий.

3.3. Структура системы

Структура системы Gulaii представлена на диаграмме архитектуры и включает следующие уровни:

1. **Клиентский уровень** – Android-приложение

2. **Уровень доступа** – Load Balancer и API Gateway

3. **Уровень микросервисов** – специализированные сервисы для различных функций системы

4. **Уровень обмена сообщениями** – брокер сообщений KeyDB

5. **Уровень данных** – PostgreSQL базы данных для каждого микросервиса и ClickHouse для аналитики

4. API GATEWAY

4.1. Назначение и функции

API Gateway в системе Gulaii является ключевым компонентом, обеспечивающим единую точку входа для всех запросов от мобильного приложения. Основные функции API Gateway:

1. **Маршрутизация запросов** – определение целевого микросервиса для обработки запроса

2. **Аутентификация и авторизация** – *проверка (и только!)* JWT-токенов и разграничение доступа

3. **Публикация сообщений в брокер** – преобразование HTTP-запросов в асинхронные сообщения

4. **Агрегирование ответов** – объединение данных от нескольких микросервисов

5. ***Логирование и мониторинг** – отслеживание всех запросов для обеспечения безопасности и производительности

6. ***Ограничение частоты запросов** – защита от перегрузок и DoS-атак

7. **Трансформация данных** – преобразование форматов данных между клиентом и микросервисами

4.2. Архитектура API Gateway

API Gateway реализован на языке Kotlin с использованием фреймворка Ktor. Внутренняя архитектура включает следующие компоненты:

1. **HTTP Server** – обработка входящих HTTP-запросов

2. **Authentication Module** – проверка JWT-токенов и извлечение информации о пользователе

3. **Routing Module** – определение целевого микросервиса на основе URL и HTTP-метода

4. **Message Producer** – формирование и публикация сообщений в брокер сообщений

5. **Response Handler** – обработка и преобразование ответов для клиента

6. **Monitoring Component** – сбор метрик о производительности и доступности

7. **Circuit Breaker** – предотвращение каскадных отказов при недоступности сервисов

4.3. Взаимодействие с брокером сообщений

Ключевой особенностью архитектуры Gulaii является **асинхронное взаимодействие через брокер сообщений**. Вместо прямых вызовов микросервисов, API Gateway публикует сообщения в KeyDB, а микросервисы подписаны на соответствующие очереди и каналы.

Процесс обработки запроса через брокер сообщений:

1. API Gateway получает HTTP-запрос от клиента
2. Gateway аутентифицирует пользователя (проверяет JWT-токен)
3. Gateway определяет тип запроса и целевой микросервис

4. Gateway формирует сообщение, включающее:
    - Идентификатор запроса (correlation ID)
    - Данные запроса
    - Метаданные (информация о пользователе, временные метки)
5. Gateway публикует сообщение в соответствующую очередь/канал в KeyDB
6. Микросервис получает сообщение из своей очереди
7. Микросервис обрабатывает запрос и публикует ответ в канал ответов
8. Gateway получает ответ из канала ответов и отправляет его клиенту

Преимущества асинхронной обработки через брокер:

1. **Буферизация запросов** – при пиковых нагрузках запросы не отклоняются, а накапливаются в очереди
2. **Устойчивость к отказам** – временная недоступность микросервиса не приводит к ошибкам для пользователя
3. **Балансировка нагрузки** – сообщения распределяются между экземплярами микросервисов
4. **Отложенная обработка** – длительные операции не блокируют пользовательский интерфейс
5. **Независимость масштабирования** – каждый компонент масштабируется в соответствии с его нагрузкой

4.4. Авторизация и безопасность

API Gateway и User Service совместно обеспечивают безопасность системы, с четким разделением ответственности:

User Service отвечает за:

- Аутентификацию пользователей (проверка учетных данных)
- Генерацию JWT-токенов
- Хранение учетных записей и профилей

API Gateway отвечает за:

- Валидацию JWT-токенов во входящих запросах
- Маршрутизацию аутентифицированных запросов
- Добавление информации о пользователе к запросам к микросервисам

4.5. Масштабирование и отказоустойчивость

API Gateway спроектирован для обеспечения высокой отказоустойчивости и масштабируемости:

1. **Горизонтальное масштабирование**:
    - Возможность запуска нескольких экземпляров API Gateway
    - Балансировка нагрузки между экземплярами через Load Balancer
2. **Устойчивость к отказам**:
    - Паттерн Circuit Breaker для предотвращения каскадных отказов
    - Таймауты и повторные попытки при недоступности сервисов
    - Fallback-механизмы для критических операций
3. **Мониторинг здоровья**:
    - Периодическая проверка доступности микросервисов
    - Автоматическое исключение недоступных сервисов из маршрутизации
    - Возвращение сервисов в пул после восстановления

5. МИКРОСЕРВИСЫ

5.1. User Service

Назначение: управление учетными записями пользователей, аутентификация и авторизация.

Основные функции:

- Регистрация и аутентификация пользователей
- Управление профилями пользователей
- Хранение предпочтений и настроек
- Интеграция с OAuth 2.0 (Google, GitHub)
- Управление сессиями и токенами

База данных: PostgreSQL с таблицами users, roles, preferences, sessions.

5.2. Nutrition Service

Назначение: учет питания и нутриентов, рекомендации по питанию.

Основные функции:

- Отслеживание потребления пищи
- Расчет КБЖУ (калории, белки, жиры, углеводы)
- Рекомендации по питанию на основе целей пользователя
- Быстрый выбор недавних блюд
- Создание шаблонов питания

База данных: PostgreSQL с таблицами meals, food_items, nutrition_logs, meal_templates.

5.3. Activity Service

Назначение: отслеживание физической активности пользователей.

Основные функции:

- Мониторинг шагов и пройденного расстояния
- Подсчет сожженных калорий
- Отслеживание времени активности
- Интеграция с данными устройств (шагомеры, пульсометры)
- Анализ динамики активности

База данных: PostgreSQL с таблицами activity_logs, step_counts, distance_records, workout_sessions.

5.4. Route Service

Назначение: генерация и управление персонализированными маршрутами.

Основные функции:

- Создание персонализированных маршрутов на основе предпочтений и активности
- Адаптация маршрутов под доступное время
- Создание тематических маршрутов (архитектура, история, природа)
- Сохранение истории маршрутов пользователя
- Генерация квестов и заданий на маршруте

База данных: PostgreSQL с таблицами routes, route_points, user_routes, route_templates.

5.5. City Service

Назначение: управление информацией о городских локациях и достопримечательностях.

Основные функции:

- Хранение данных о городских объектах
- Категоризация мест по типам и интересам
- Предоставление информации о доступности объектов
- Интеграция с геопространственными данными
- Обновление информации о временных событиях

База данных: PostgreSQL с таблицами locations, location_types, events, operating_hours.

5.6. Notification Service

Назначение: управление уведомлениями для пользователей.

Основные функции:

- Отправка push-уведомлений
- Управление расписанием уведомлений
- Персонализация содержания уведомлений
- Отслеживание взаимодействия с уведомлениями
- Интеграция с Firebase Cloud Messaging

База данных: PostgreSQL с таблицами notifications, notification_templates, user_notification_settings.

6. БРОКЕР СООБЩЕНИЙ

6.1. Назначение KeyDB

В архитектуре Gulaii KeyDB выполняет роль центрального брокера сообщений, обеспечивающего асинхронное взаимодействие между компонентами системы. KeyDB – это высокопроизводительный форк Redis, оптимизированный для многопоточной обработки.

Основные функции KeyDB в системе:

- Обеспечение асинхронной коммуникации между сервисами
- Буферизация сообщений при пиковых нагрузках
- Поддержка различных паттернов обмена сообщениями
- Гарантия доставки сообщений
- Временное хранение данных в кэше

6.2. Модель обмена сообщениями

В системе Gulaii используется асинхронное взаимодействие через брокер сообщений KeyDB с применением двух основных стилей интеграции.

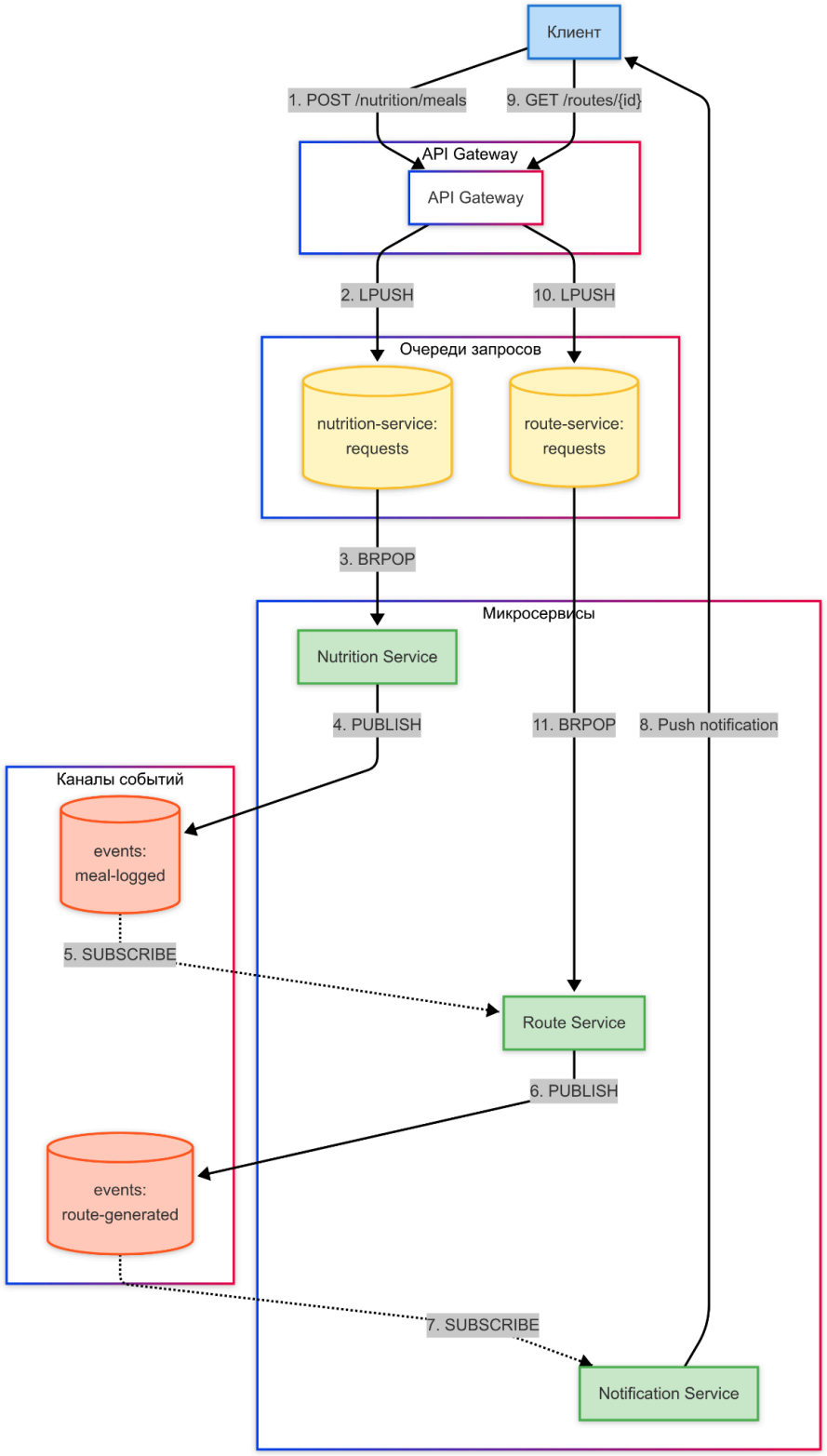
6.2.1. Хореография и оркестрация

Хореография (преобладающий подход в Gulaii) - децентрализованный стиль взаимодействия, при котором:

- Микросервисы публикуют события о значимых изменениях состояния
- Другие микросервисы подписываются на события и реагируют на них автономно
- Каждый сервис принимает независимые решения о своих действиях
- Отсутствует центральный координатор процессов

Пример: При логировании пищи Nutrition Service публикует событие “meal-logged”, на которое независимо реагируют Route Service (генерирует маршрут) и Notification Service (отправляет уведомление).

Оркестрация (применяется в ограниченных случаях) - централизованный стиль взаимодействия, где:



- Выделенный сервис (оркестратор) управляет всем процессом
- Оркестратор вызывает другие сервисы в определенной последовательности
- Оркестратор отслеживает состояние всего процесса

В Gulaii элементы оркестрации используются в основном в API Gateway для маршрутизации запросов.

6.2.2. Основные паттерны обмена сообщениями

В Gulaii используются следующие паттерны обмена сообщениями через KeyDB:

1. Request/Response через очереди

Используется для обработки запросов от API Gateway к микросервисам:

- API Gateway помещает запросы в **очереди конкретных**

микросервисов (LPUSH)

- Каждый запрос содержит correlation ID для связывания с ответом
- Микросервисы читают запросы из своих очередей (BRPOP)
- Микросервисы публикуют ответы в общий канал ответов (PUBLISH)
- API Gateway получает ответы из канала и сопоставляет их с запросами по correlation ID

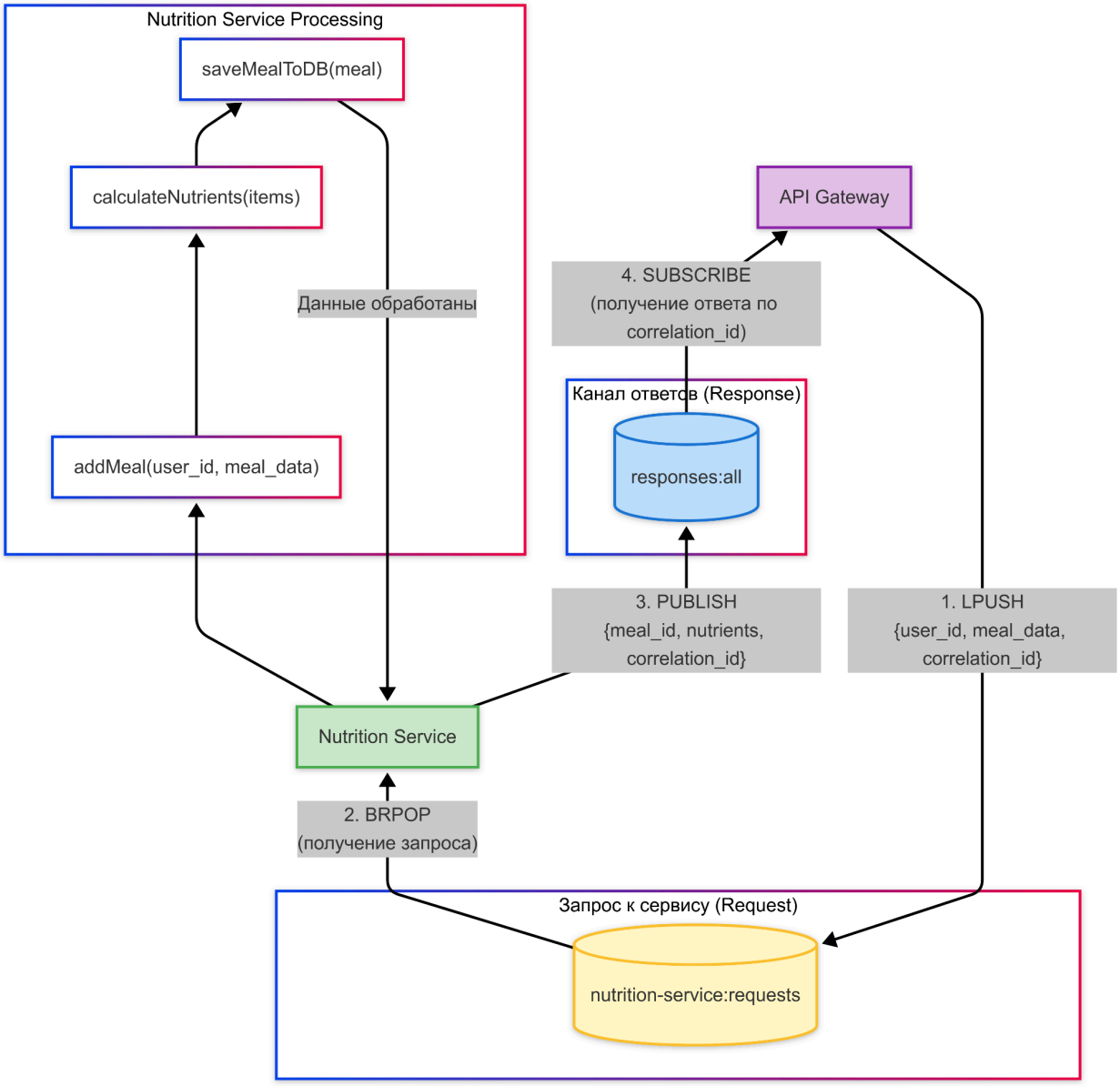


Рисунок X. Паттерн Request/Response на примере Nutrition Service

2. Publish/Subscribe (Pub/Sub)

Используется для событийного взаимодействия между микросервисами:

- Сервисы публикуют события в тематические каналы (PUBLISH)
- Заинтересованные сервисы подписываются на эти каналы (SUBSCRIBE)
- Все подписчики получают копию сообщения
- Каждый подписчик независимо решает, как реагировать на событие

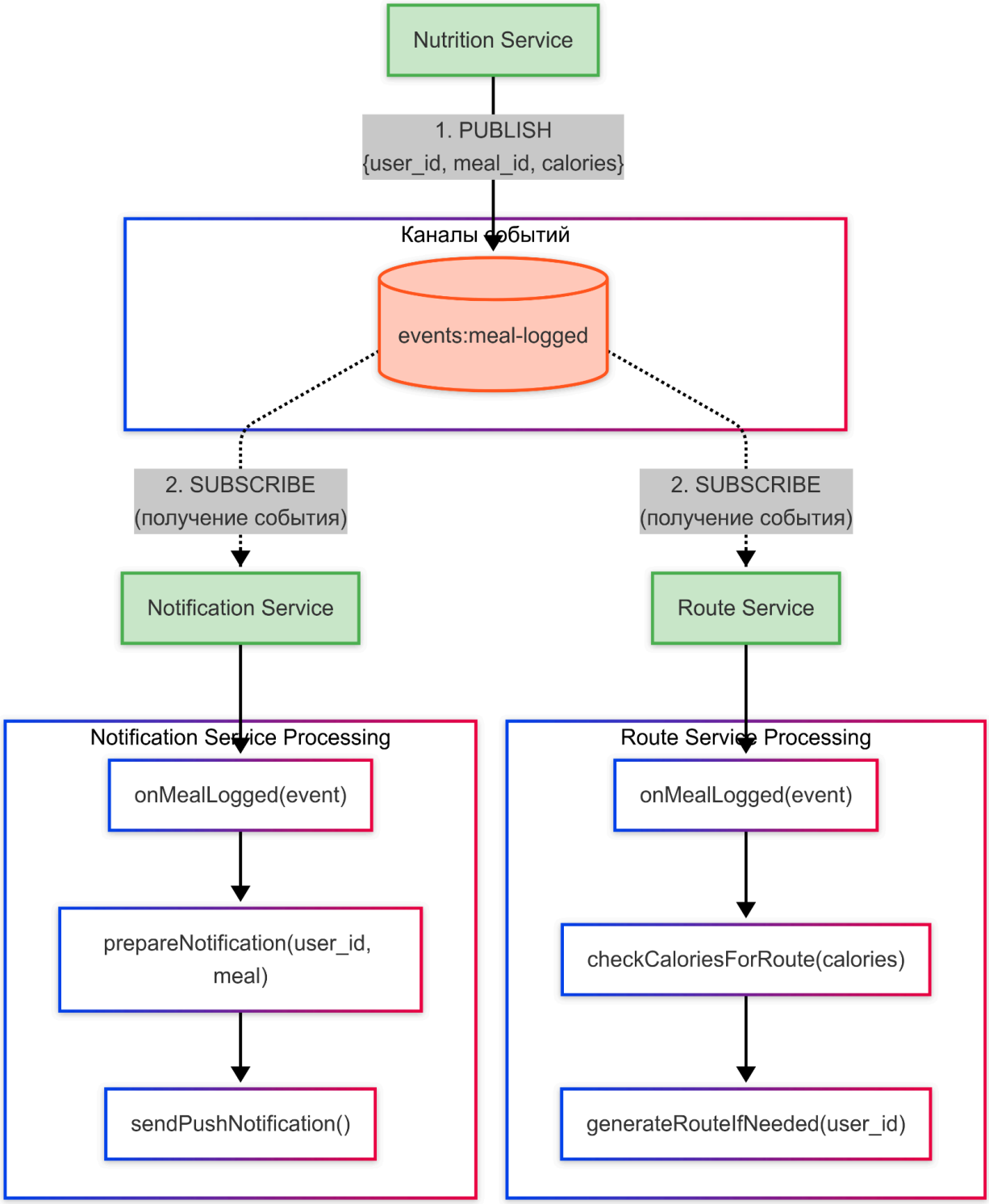


Рисунок X. Паттерн Publish/Subscribe на примере обработки логирования питания

3. Event Sourcing (для определенных компонентов)

Для критических данных и возможности аудита:

- История изменений сохраняется как последовательность событий
- Текущее состояние можно восстановить, применив все события
- Используется для важных бизнес-процессов и данных, требующих аудита

6.3. Очереди и топики

Система Gulaii использует следующие основные очереди и топики в KeyDB:

Очереди запросов (для асинхронной обработки):

- `user-service:request-queue`
- `nutrition-service:request-queue`
- `activity-service:request-queue`
- `route-service:request-queue`
- `city-service:request-queue`
- `notification-service:request-queue`

Каналы ответов (для получения результатов обработки):

- `api-gateway:response-channel`

Топики событий (для оповещения о событиях):

- `events:user-profile-changed`
- `events:meal-logged`
- `events:activity-recorded`
- `events:route-completed`
- `events:system-status`

7. СИСТЕМНЫЕ ПРОЦЕССЫ

7.1. Регистрация и авторизация

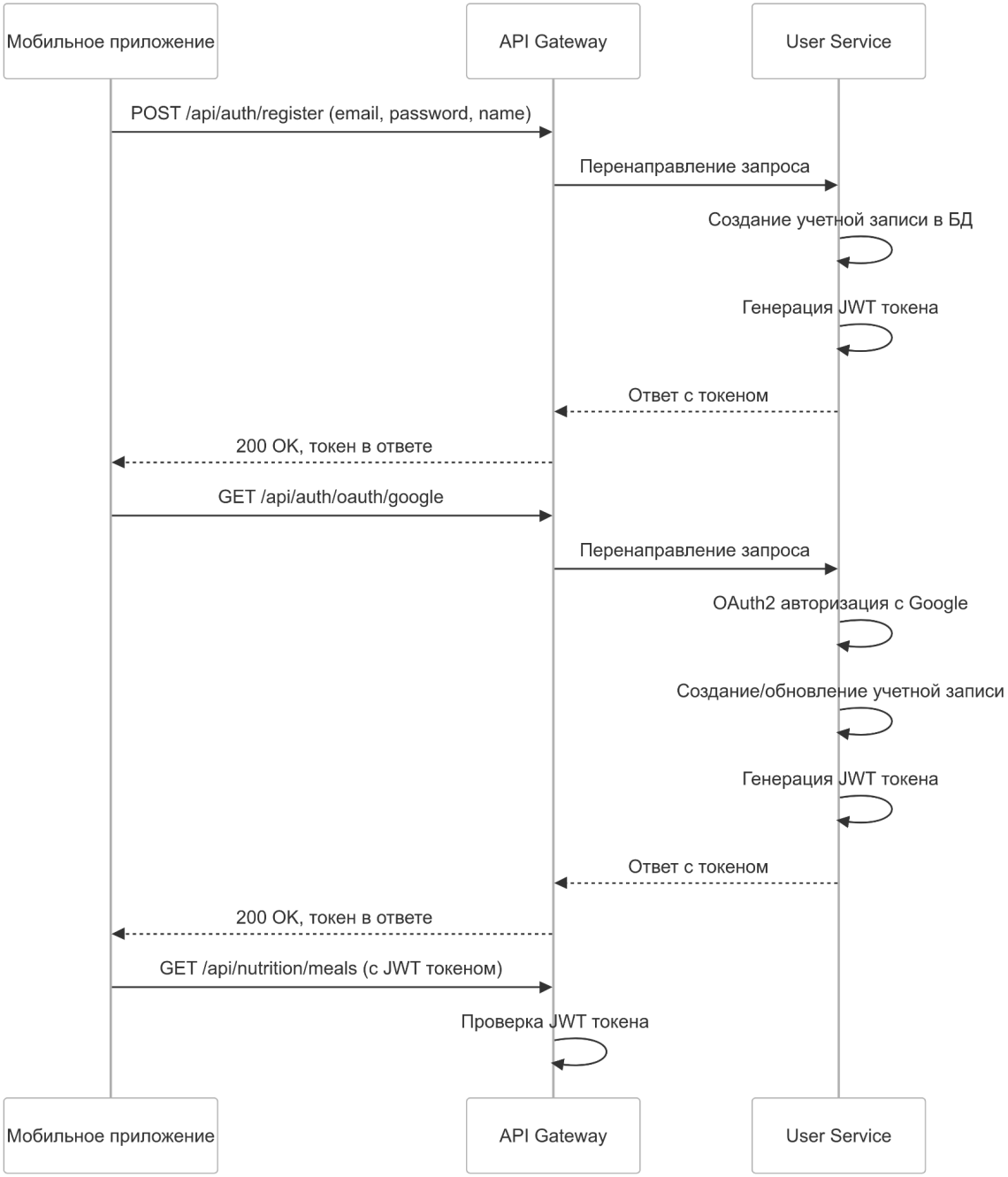


Рисунок X. Процесс аутентификации и авторизации в системе Gulaii

Регистрация пользователя:

1. Клиент отправляет запрос на регистрацию в API Gateway
2. API Gateway публикует сообщение в очередь User Service
3. User Service создает учетную запись пользователя и сохраняет данные в БД
4. User Service публикует ответ в канал ответов
5. API Gateway получает ответ и возвращает результат клиенту
6. User Service публикует событие `user-registered` в соответствующий топик

Авторизация пользователя:

1. Клиент отправляет учетные данные в API Gateway
2. API Gateway публикует запрос в очередь User Service
3. User Service проверяет учетные данные и генерирует JWT-токен
4. User Service публикует ответ с токеном в канал ответов
5. API Gateway получает ответ и возвращает токен клиенту
6. В последующих запросах API Gateway проверяет валидность токена и маршрутизирует запросы к соответствующим микросервисам

7.2. Обработка запросов пользователя

Запрос на добавление приема пищи:

1. Клиент отправляет данные о приеме пищи в API Gateway
2. API Gateway аутентифицирует пользователя по JWT-токену
3. API Gateway публикует сообщение в очередь Nutrition Service
4. Nutrition Service обрабатывает данные, рассчитывает нутриенты и сохраняет в БД
5. Nutrition Service публикует ответ в канал ответов
6. Nutrition Service публикует событие `meal-logged` в соответствующий топик
7. Activity Service и Route Service, подписанные на этот топик, получают событие
8. Route Service генерирует рекомендации по маршрутам с учетом новых данных

7.3. Генерация маршрутов

Запрос на генерацию маршрута:

1. Клиент запрашивает персонализированный маршрут через API Gateway
2. API Gateway публикует запрос в очередь Route Service
3. Route Service запрашивает данные о питании у Nutrition Service через брокер
4. Route Service запрашивает данные об активности у Activity Service через брокер
5. Route Service запрашивает данные о городских локациях у City Service через брокер
6. Route Service генерирует оптимальный маршрут на основе собранных данных
7. Route Service публикует ответ с маршрутом в канал ответов
8. API Gateway получает ответ и возвращает маршрут клиенту

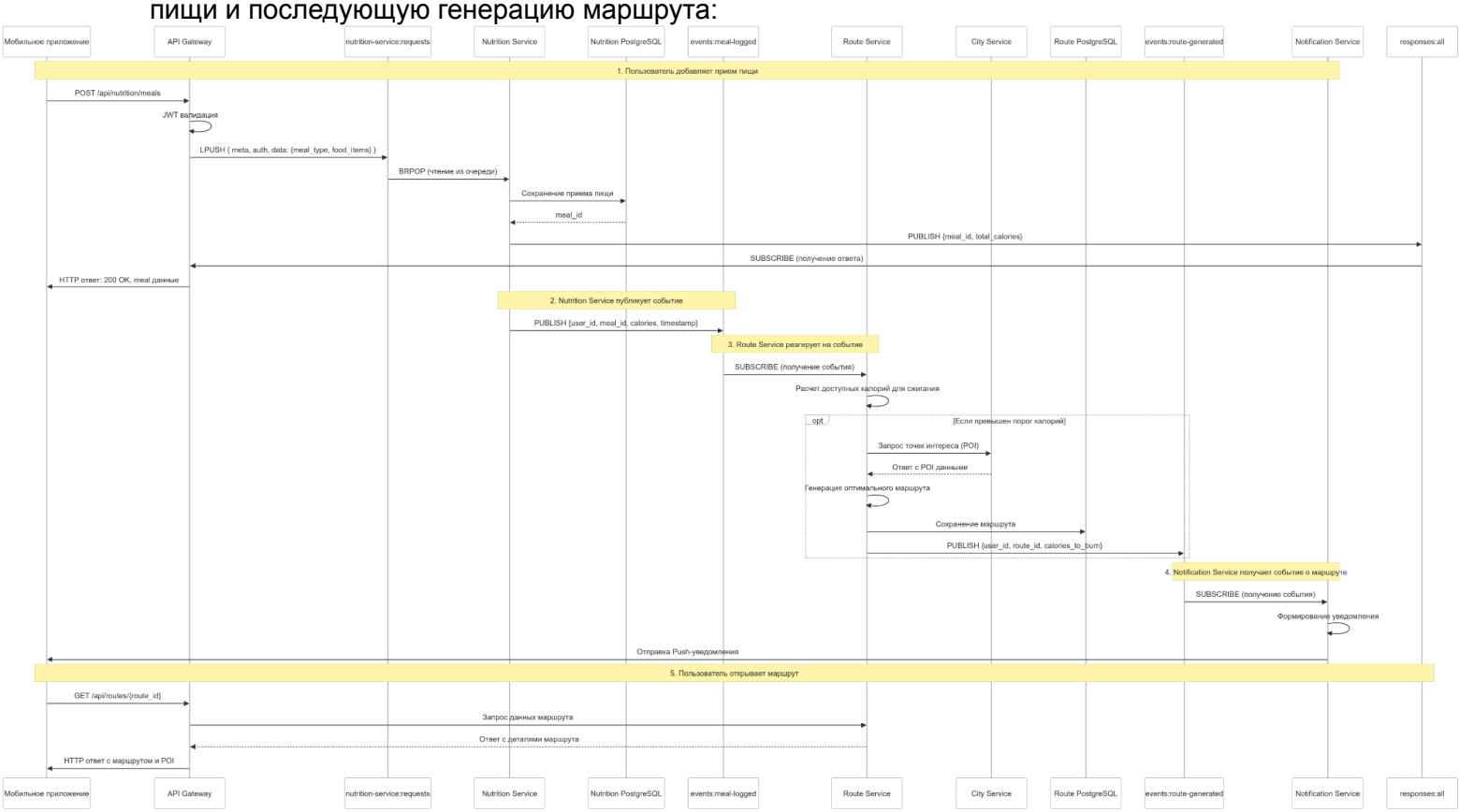
7.4. Синхронизация данных

Синхронизация данных об активности с устройств:

1. Клиент отправляет данные об активности в API Gateway
2. API Gateway публикует данные в очередь Activity Service
3. Activity Service обрабатывает и сохраняет данные в БД
4. Activity Service публикует событие `activity-updated` в соответствующий топик
5. Nutrition Service и Route Service, подписанные на этот топик, получают обновление
6. Nutrition Service обновляет расчеты калорийного баланса
7. Route Service обновляет рекомендации по маршрутам

7.5. Пример взаимодействия сервисов

Для иллюстрации взаимодействия микросервисов, рассмотрим сценарий записи приема пищи и последующую генерацию маршрута:



Данный сценарий демонстрирует асинхронную хореографию, где:

1. Nutrition Service сохраняет данные о приеме пищи и публикует событие
2. Route Service, подписанный на события питания, реагирует и генерирует маршрут
3. Notification Service, подписанный на события маршрутов, отправляет уведомление пользователю

8. ХРАНЕНИЕ ДАННЫХ

8.1. PostgreSQL базы данных

Каждый микросервис в системе Gulaii использует отдельную PostgreSQL базу данных, что обеспечивает изоляцию данных и независимость разработки.

Преимущества использования отдельных баз данных:

- Независимость схем данных
- Изоляция нагрузки
- Улучшенная безопасность
- Возможность независимого масштабирования

Стратегии обеспечения согласованности данных:

- Обмен событиями через брокер сообщений
- Хранение только необходимых ссылок на данные других сервисов
- Периодическая синхронизация данных

8.2. ClickHouse для аналитики

Для аналитической обработки данных в системе Gulaii используется колоночная СУБД ClickHouse, которая обеспечивает высокую производительность при работе с большими объемами данных.

Основные функции ClickHouse в системе:

- Агрегация исторических данных о питании и активности
- Генерация аналитических отчетов
- Выявление паттернов поведения пользователей
- Оптимизация рекомендаций на основе статистики

Процесс передачи данных в ClickHouse:

1. Микросервисы (Nutrition Service, Activity Service, Route Service) периодически экспортируют агрегированные данные
2. Данные преобразуются в подходящий для аналитики формат
3. ClickHouse импортирует и индексирует данные
4. Аналитические запросы выполняются отдельно от основных операций системы

8.3. Стратегии резервного копирования

Для обеспечения целостности и доступности данных в системе Gulaii реализованы следующие стратегии резервного копирования:

1. **Полное резервное копирование PostgreSQL**:
    - Еженедельное полное резервное копирование всех баз данных
    - Хранение копий в зашифрованном виде в облачном хранилище
2. **Инкрементальные резервные копии**:
    - Ежедневное инкрементальное резервное копирование
    - Сохранение журналов транзакций для восстановления point-in-time
3. **Резервное копирование KeyDB**:
    - Периодические снимки состояния брокера сообщений
    - Резервирование журналов событий
4. **Резервное копирование ClickHouse**:
    - Регулярное копирование аналитических данных
    - Возможность повторного импорта из первичных источников

8.4. Структуры данных микросервисов

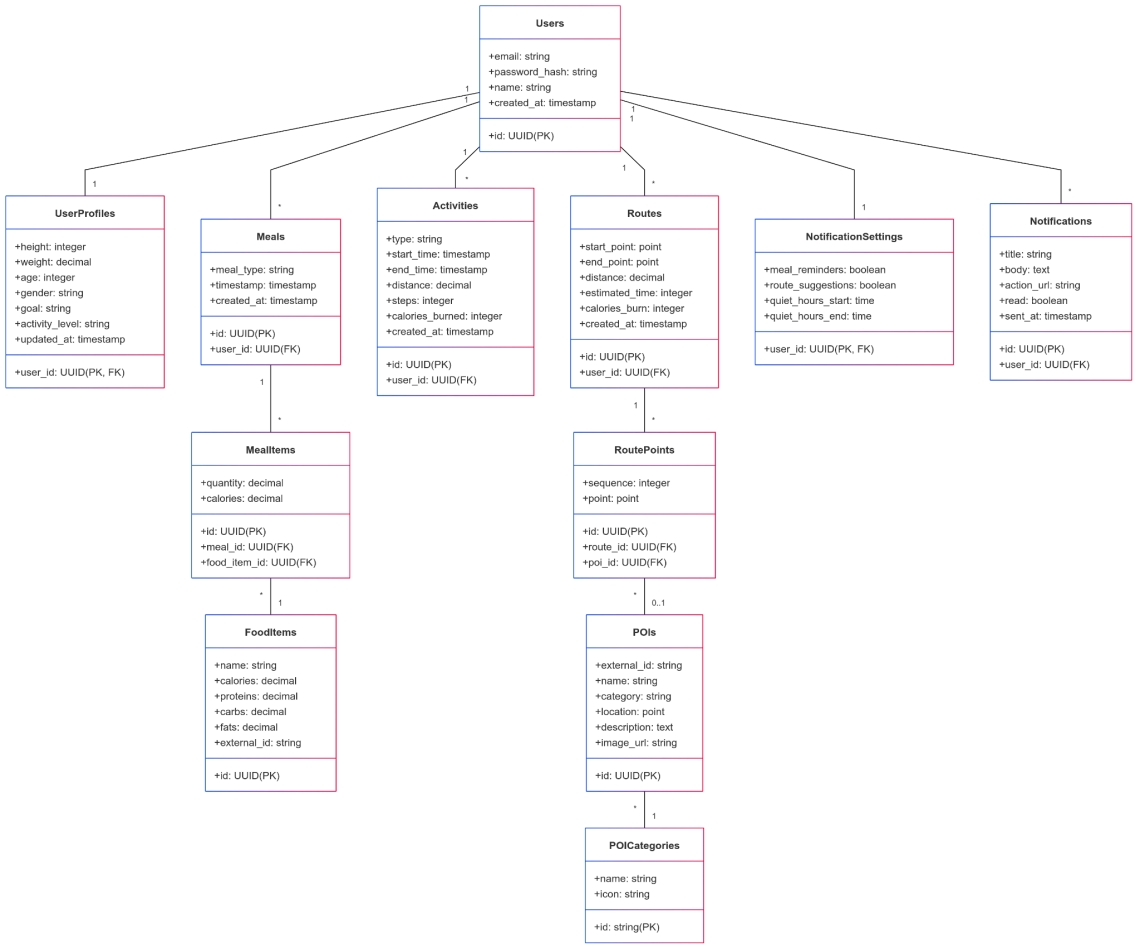


Рисунок X. Модели данных микросервисов Gulaii

9. НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

9.1. Производительность

1. **Требования к отзывчивости**:
    - Время отклика API Gateway не более 100 мс
    - Время обработки запросов микросервисами не более 500 мс
    - Генерация маршрута не более 2 секунд
2. **Требования к пропускной способности**:
    - Поддержка не менее 100 одновременных пользователей
    - Обработка не менее 50 запросов в секунду
    - Масштабирование до 10 000 пользователей в рамках текущей архитектуры

9.2. Масштабируемость

1. **Горизонтальное масштабирование**:
    - Возможность запуска дополнительных экземпляров микросервисов при увеличении нагрузки
    - Автоматическое масштабирование на основе метрик утилизации
2. **Вертикальное масштабирование**:
    - Возможность увеличения ресурсов для отдельных компонентов системы
    - Оптимизация использования ресурсов
3. **Масштабирование баз данных**:
    - Поддержка шардирования данных по мере роста объема
    - Репликация для распределения нагрузки на чтение

9.3. Отказоустойчивость

1. **Устойчивость к отказам компонентов**:
    - Способность системы функционировать при недоступности отдельных микросервисов
    - Изоляция сбоев в рамках отдельных компонентов
2. **Механизмы восстановления**:
    - Автоматический перезапуск неисправных компонентов
    - Резервное копирование и восстановление данных
    - Журналирование событий для диагностики проблем

3. **Мониторинг состояния**:
    - Отслеживание доступности и производительности всех компонентов
    - Система оповещения о критических событиях

9.4. Безопасность

1. **Защита данных**:
    - Шифрование данных в покое и при передаче
    - Безопасное хранение учетных данных пользователей
    - Регулярное тестирование на уязвимости
2. **Контроль доступа**:
    - Многоуровневая аутентификация
    - Разграничение прав доступа на основе ролей
    - Аудит действий пользователей
3. **Защита от атак**:
    - Защита от CSRF, XSS, SQL-инъекций
    - Ограничение частоты запросов (rate limiting)
    - Мониторинг подозрительной активности

9.5. Требования к клиентской части

1. **Поддержка устройств**:
    - Минимальная версия Android: 8.0 (API level 26)
    - Оптимизация для различных размеров экранов
    - Адаптация для устройств с ограниченными ресурсами
2. **Автономная работа**:
    - Возможность использования основных функций без доступа к сети
    - Синхронизация данных при восстановлении соединения
    - Локальное кэширование данных
3. **Ограничения**:
    - Максимальный размер .apk файла: 50 МБ
    - Минимальные требования к устройству: 2 ГБ RAM, 1.5 ГГц CPU
    - Оптимизация использования батареи при фоновом отслеживании активности

10. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

10.1. Пользовательский интерфейс

1. **Общие требования к UI**:
    - Соответствие дизайн-системе Material Design 3
    - Поддержка светлой и темной темы
    - Адаптивный дизайн для различных размеров экранов
    - Burger-меню для навигации по приложению
2. **Локализация**:
    - Поддержка русского (ru_RU) и английского (en_EN) языков
    - Возможность расширения списка поддерживаемых языков
3. **Навигация**:
    - Интуитивно понятная навигация между разделами
    - Панель быстрого доступа к основным функциям
    - Страница настроек приложения
4. **Доступность**:
    - Поддержка TalkBack для пользователей с нарушениями зрения
    - Настраиваемый размер шрифта
    - Высокий контраст для элементов интерфейса

10.2. Управление профилем

1. **Регистрация и авторизация**:
    - Создание аккаунта с помощью email/пароль
    - Авторизация через OAuth 2.0 (Google, GitHub)
    - Возможность восстановления пароля
    - Возможность удаления аккаунта
2. **Профиль пользователя**:
    - Редактирование личных данных (имя, возраст, пол)
    - Указание физических параметров (рост, вес)
    - Установка целей (похудение, набор массы, поддержание веса)
    - Настройка уровня физической активности
3. **Настройки приложения**:
    - Выбор языка интерфейса
    - Переключение между светлой и темной темой
    - Настройка уведомлений
    - Управление синхронизацией данных

10.3. Питание и активность

1. **Учет питания**:
    - Добавление съеденных продуктов
    - Расчет КБЖУ (калории, белки, жиры, углеводы)
    - Быстрый выбор из недавних или избранных блюд
    - Создание шаблонов приемов пищи
2. **Отслеживание активности**:
    - Автоматическое отслеживание шагов
    - Ручное добавление активностей
    - Расчет сожженных калорий
    - Синхронизация с фитнес-устройствами
3. **Аналитика и статистика**:
    - Отображение баланса потребленных и сожженных калорий
    - Графики динамики веса и других показателей
    - Еженедельные и месячные отчеты
    - Персонализированные рекомендации
4. **Рекомендации по питанию**:
    - * Генерация рекомендаций на основе AI
    - Учет предпочтений и ограничений пользователя
    - Адаптация рациона к физической активности

10.4. Маршруты и городские исследования

1. **Генерация маршрутов**:
    - Создание персонализированных маршрутов на основе потребленных калорий
    - Адаптация маршрутов под доступное время пользователя
    - Учет погодных условий и времени суток
    - Тематические маршруты (архитектура, история, природа)
2. **Навигация по маршруту**:
    - Пошаговая навигация с использованием GPS
    - Отображение интересных мест на маршруте
    - Информация о достопримечательностях
    - Отслеживание прогресса прохождения маршрута
3. **Городские исследования**:
    - Коллекционирование посещенных мест

- Квесты и задания на маршрутах
- Социальное взаимодействие с другими пользователями
- Возможность создания и публикации собственных маршрутов

11. ТЕХНОЛОГИЧЕСКИЙ СТЕК

11.1. Обоснование выбора технологий

1. **Kotlin**:
    - Современный, лаконичный и безопасный язык программирования
    - Официальный язык для разработки под Android
    - Поддержка корутинов для асинхронного программирования
    - Совместимость с Java-библиотеками
2. **Ktor**:
    - Легковесный и производительный фреймворк для создания веб-приложений
    - Разработан специально для Kotlin с поддержкой корутинов
    - Модульная архитектура с расширяемой функциональностью
    - Удобный для создания REST API и веб-сервисов
3. **KeyDB**:
    - Высокопроизводительный форк Redis с улучшенной многопоточностью
    - Поддержка различных структур данных для разных сценариев использования
    - Эффективный механизм публикации/подписки для обмена сообщениями
    - Встроенное кэширование для повышения производительности
4. **PostgreSQL**:
    - Надежная и проверенная СУБД с поддержкой ACID-транзакций
    - Богатые возможности для работы с реляционными данными
    - Поддержка JSON-типов для гибкого хранения данных
    - Расширения для работы с геопространственными данными
5. **ClickHouse**:
    - Колоночная СУБД, оптимизированная для аналитических запросов
    - Высокая производительность при работе с большими объемами данных
    - Эффективное сжатие данных
    - Интеграция с различными источниками данных

11.2. Инструменты разработки

1. **IDE и редакторы**:
    - IntelliJ IDEA для разработки бэкенда
    - Android Studio для мобильного приложения
    - PyCharm/JupyterLab для аналитики
2. **Система контроля версий**:
    - git
    - GitHub для хостинга репозиториев
    - Git Flow для организации процесса разработки
3. **Инструменты документирования**:
    - Dokka для документации исходного кода
    - Markdown (Obsidian) для проектной документации
    - PlantUML / mermaid / draw.io для создания диаграмм
4. **Инструменты тестирования**:
    - unit/интеграционное **надо пилить САМИМ**
    - Postman для тестирования API

12. ЗАКЛЮЧЕНИЕ

Документ представляет техническую спецификацию архитектуры микросервисной системы Gulaii, объединяющей функции мониторинга питания с городскими исследованиями. Выбранная архитектура обеспечивает модульность, масштабируемость и устойчивость системы к отказам.

Ключевыми особенностями архитектуры являются:

1. Использование API Gateway как единой точки входа для клиентских запросов
2. Асинхронное взаимодействие через брокер сообщений KeyDB
3. Независимые микросервисы с собственными базами данных
4. Аналитический кластер ClickHouse для обработки больших объемов данных

Система спроектирована с учетом возможности масштабирования до 10,000 пользователей и обеспечивает безопасность, производительность и удобство использования.

13. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы»
2. ГОСТ 19.201-78 «Техническое задание. Требования к содержанию и оформлению»
3. Роберт Мартин. Чистая архитектура. – СПБ.: Питер, 2018. – 352 с.

4. Newman, S. Building Microservices / S. Newman. — Sebastopol: O'Reilly Media, Inc, 2021. — 616 с.
5. Клепманн, М. Проектирование Data-Intensive приложений. Надежные системы на ненадежной инфраструктуре / М. Клепманн. — СПб.: Питер, 2021. — 600 с.
6. Ktor - Документация. [Электронный ресурс]. - Режим доступа: https://ktor.io/docs/ (дата обращения: 15.03.2025)
7. KeyDB Documentation [Электронный ресурс]. - Режим доступа: https://docs.keydb.dev/ (дата обращения: 15.03.2025)
8. PostgreSQL: Documentation [Электронный ресурс]. - Режим доступа: https://www.postgresql.org/docs/ (дата обращения: 15.03.2025)
9. ClickHouse Documentation [Электронный ресурс]. - Режим доступа: https://clickhouse.com/docs/ (дата обращения: 15.03.2025)