САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэкенд разработка

Отчет

Домашняя работа 1

Выполнил:

Берулава Леон Алхасович

К3439

Проверил:
Добряков Д. И.

Санкт-Петербург

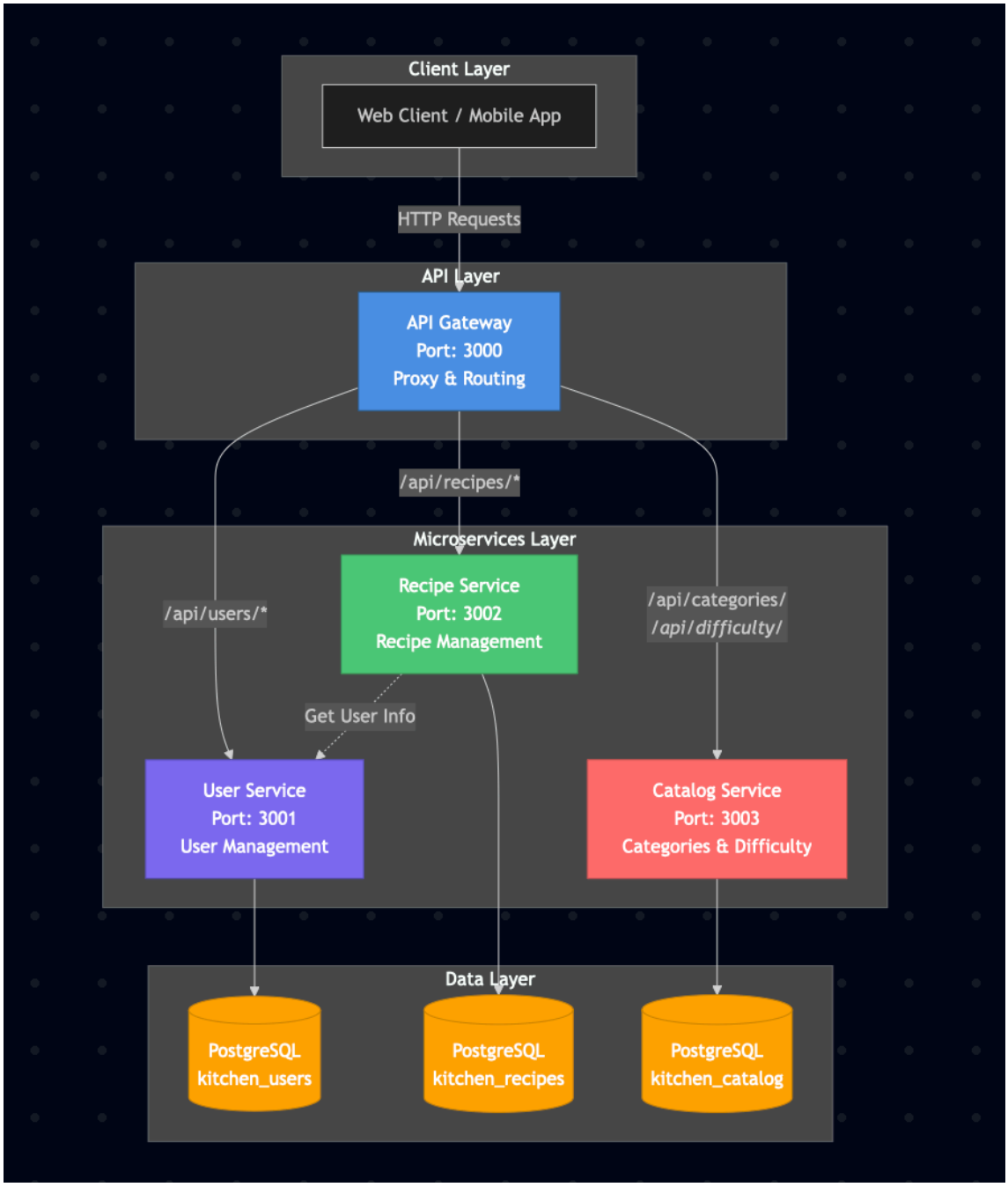
2025 г.

Задача

Необходимо спроектировать набор следующих диаграмм:

- общая архитектура решения (сервисы и их взаимосвязи, клиент-серверное взаимодействие);
- диаграмма компонентов;
- диаграммы БД по каждому сервису;
- диаграммы основных пользовательских сценариев (те сценарии, которые позволяют вашим приложением полноценно воспользоваться, пройти весь путь).

Ход работы



Эта диаграмма представляет собой высокоуровневый обзор всей системы - как клиент взаимодействует с backend, как микросервисы связаны друг с другом и с базами данных.

Основные элементы:

Client Layer (Клиентский уровень)

- Web Client / Mobile App - точка входа пользователя
- Отправляет HTTP запросы к системе

API Gateway (Порт 3000)

- **Единая точка входа** для всех клиентских запросов
- Выполняет роль **reverse proxy** - принимает запросы и перенаправляет их на нужные микросервисы
- Маршрутизация:
  - `/api/users/*` → User Service
  - `/api/recipes/*` → Recipe Service
  - `/api/categories/*` и `/api/difficulty/*` → Catalog Service
- Реализован с помощью **Express.js** и **http-proxy-middleware**

Microservices Layer (Уровень микросервисов)

User Service (Порт 3001)

- Управление пользователями
- Регистрация и авторизация
- Хранение профилей
- Хэширование паролей (bcrypt)

Recipe Service (Порт 3002)

- CRUD операции с рецептами
- Хранение ингредиентов, инструкций, времени приготовления
- **Важно:** Обращается к User Service для получения данных об авторах рецептов (пунктирная стрелка на диаграмме)

Catalog Service (Порт 3003)

- Управление категориями (Завтраки, Основные блюда, Супы и т.д.)
- Управление уровнями сложности (Легко, Средне, Сложно)

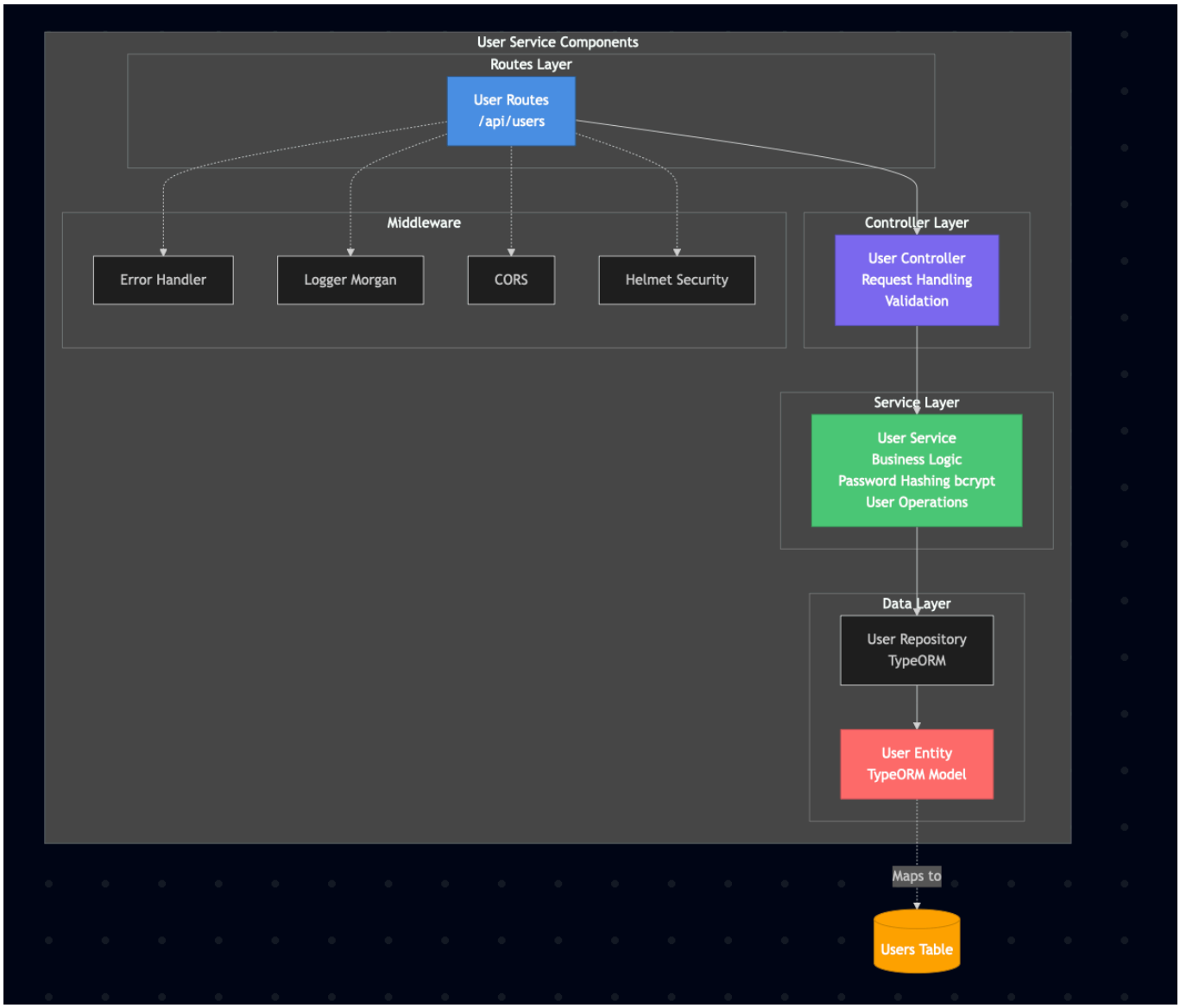
Data Layer (Уровень данных)

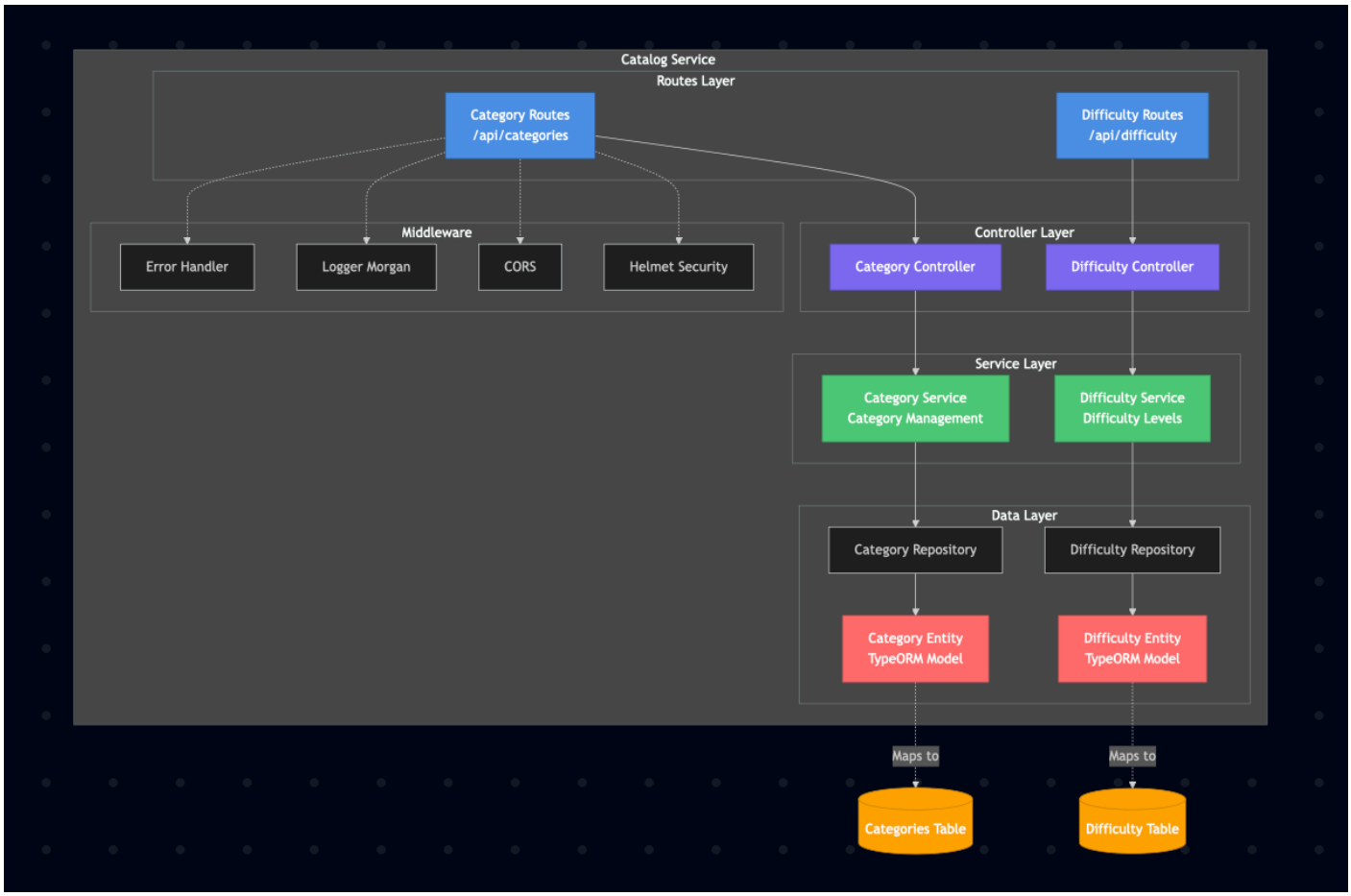
- Три **независимые** PostgreSQL базы данных

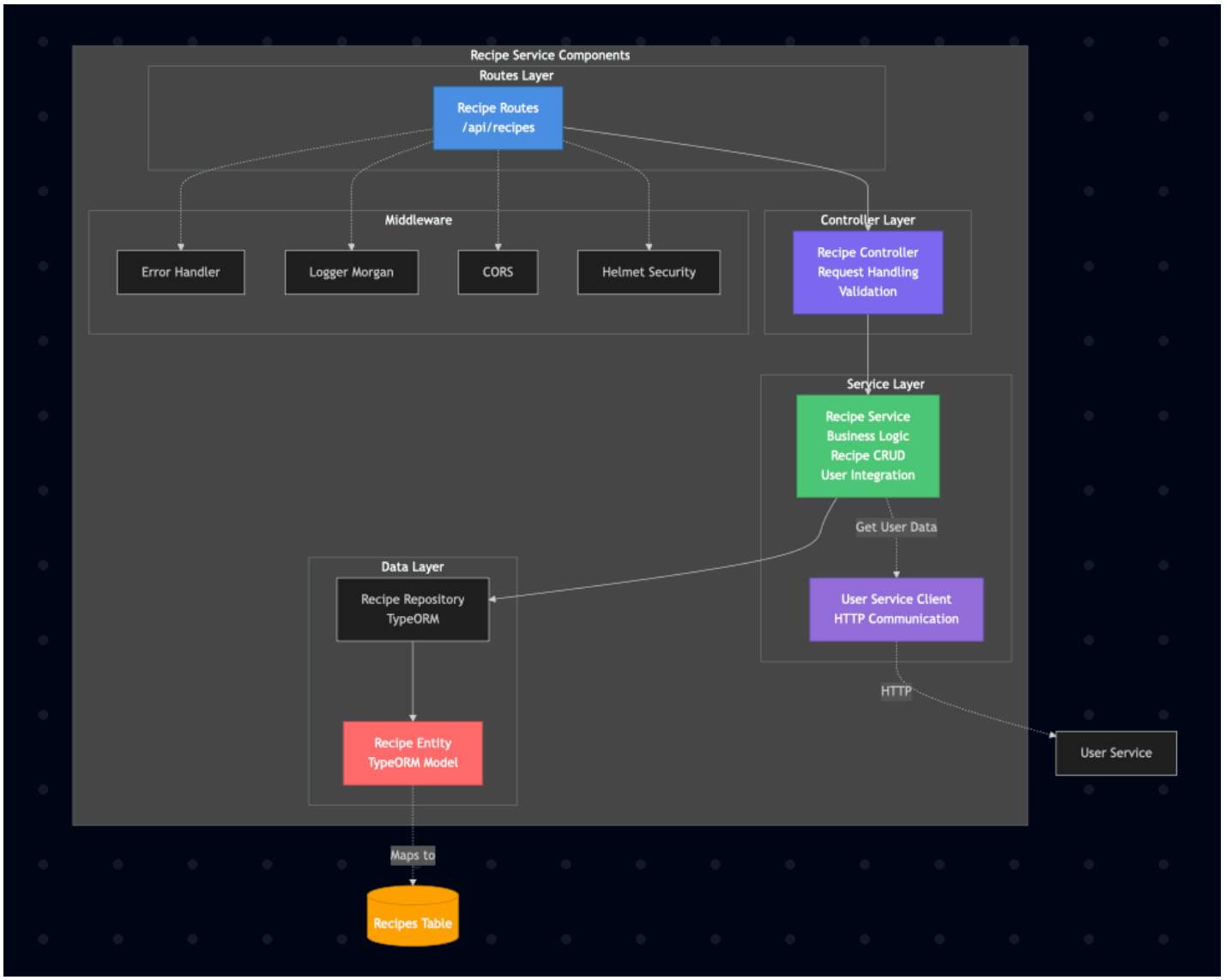
- Каждый сервис имеет свою изолированную БД (паттерн Database per Service)
- `kitchen_users`, `kitchen_recipes`, `kitchen_catalog`

Ключевые паттерны:

- **API Gateway Pattern** - централизованный вход
- **Database per Service** - изоляция данных
- **Service-to-Service Communication** - Recipe Service → User Service




Catalog Service
Routes Layer
Category Routes
/api/categories
Difficulty Routes
/api/difficulty
Middleware
Error Handler
Logger Morgan
CORS
Helmet Security
Controller Layer
Category Controller
Difficulty Controller
Service Layer
Category Service
Category Management
Difficulty Service
Difficulty Levels
Data Layer
Category Repository
Difficulty Repository
Category Entity
TypeORM Model
Difficulty Entity
TypeORM Model
Maps to
Maps to
Categories Table
Difficulty Table


Recipe Service Components
Routes Layer
Recipe Routes
/api/recipes
Middleware
Error Handler
Logger Morgan
CORS
Helmet Security
Controller Layer
Recipe Controller
Request Handling
Validation
Service Layer
Recipe Service
Business Logic
Recipe CRUD
User Integration
Get User Data
Data Layer
Recipe Repository
TypeORM
User Service Client
HTTP Communication
HTTP
Recipe Entity
TypeORM Model
User Service
Maps to
Recipes Table

2.1 User Service Components

Показывает **внутреннюю структуру** User Service - как организован код внутри сервиса.

Routes Layer (Слой маршрутизации)

- Определяет API endpoints: `/api/users`
- Регистрирует HTTP методы (GET, POST, PUT, DELETE)

Controller Layer (Слой контроллеров)

- Принимает HTTP запросы
- Парсит и валидирует входные данные
- Обрабатывает ошибки
- Формирует HTTP ответы

Service Layer (Слой бизнес-логики)

- **User Service Logic** - основная бизнес-логика
- **Password Hashing** - хэширование паролей через bcrypt (соль + хэш)
- Бизнес-правила (например, проверка уникальности email)
- Операции с пользователями (создание, обновление, удаление)

Data Layer (Слой данных)

- **User Entity** - TypeORM модель (описывает структуру таблицы)
- **User Repository** - TypeORM репозиторий (методы для работы с БД)
- Маппинг JavaScript объектов на строки БД

Middleware (Промежуточное ПО)

- **Error Handler** - перехватывает и обрабатывает ошибки
- **Logger (Morgan)** - логирование HTTP запросов
- **CORS** - управление политиками Cross-Origin запросов
- **Helmet** - защита HTTP заголовков от атак

2.2 Recipe Service Components

Аналогичная структура, но с **важным дополнением**:

User Service Client (в Service Layer)

- HTTP клиент для связи с User Service

- Когда нужны данные автора рецепта, Recipe Service делает HTTP запрос к User Service
- Пример: `GET http://user-service:3001/api/users/123` для получения имени автора
- Это **межсервисное взаимодействие**

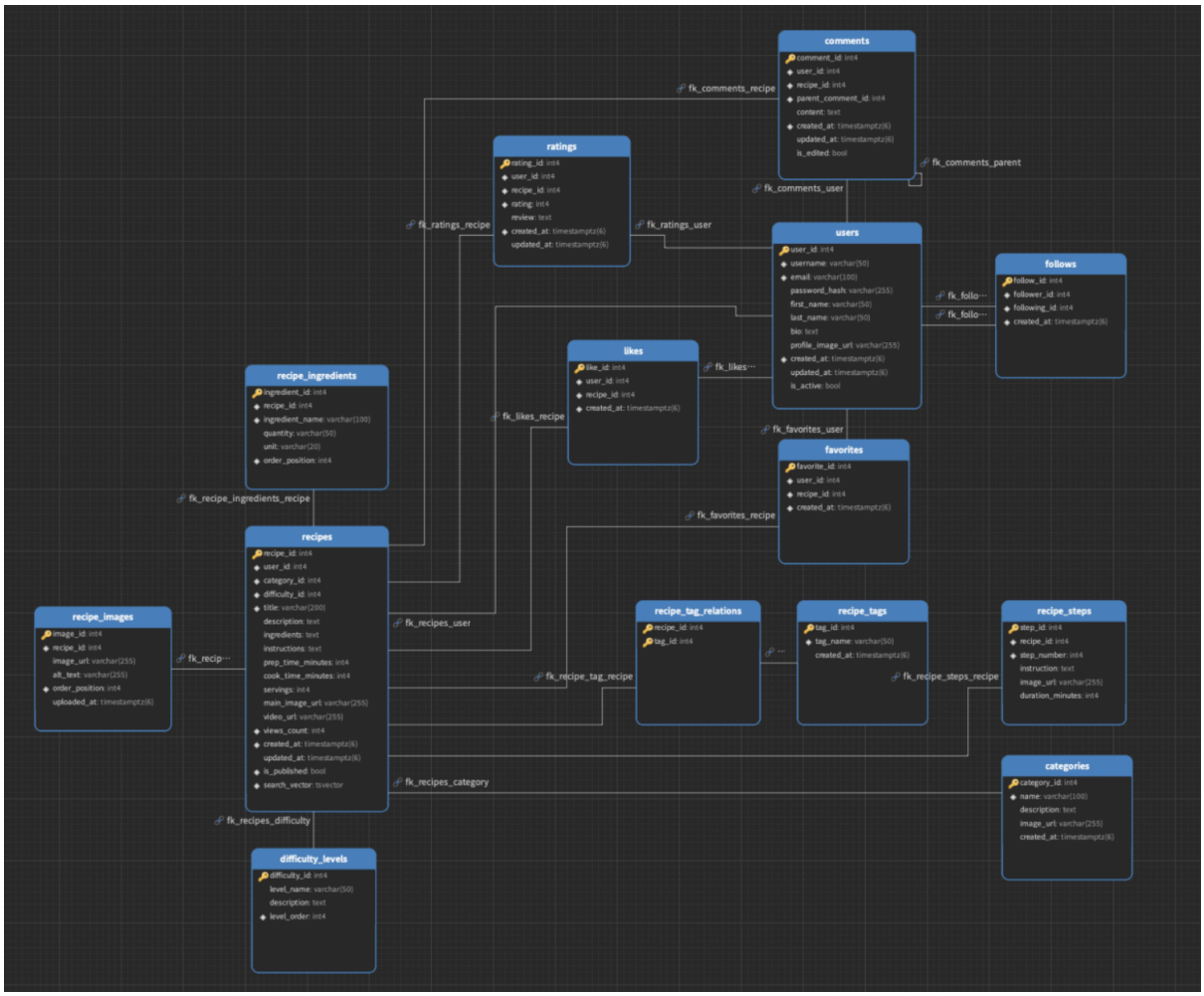
2.3 Catalog Service Components

Особенность: управляет **двумя типами сущностей**

Две параллельные структуры:

1. **Category Routes/Controller/Service/Entity** - для категорий
2. **Difficulty Routes/Controller/Service/Entity** - для уровней сложности

Каждая сущность имеет свой полный набор слоев, но они работают в одном сервисе.



USERS			
uuid	id	PK	Primary Key
varchar	username	UK	Unique, Not Null
varchar	email	UK	Unique, Not Null
varchar	password		Hashed with bcrypt, Not Null
varchar	firstName		Nullable
varchar	lastName		Nullable
text	bio		User biography, Nullable
varchar	avatar		Avatar URL, Nullable
timestamp	createdAt		Auto-generated
timestamp	updatedAt		Auto-updated

RECIPES			
uuid	id	PK	Primary Key
varchar	title		Recipe title, Not Null
text	description		Recipe description, Nullable
text	ingredients		Ingredients list, Not Null
text	instructions		Cooking instructions, Not Null
int	prepTime		Preparation time in minutes, Not Null
int	cookTime		Cooking time in minutes, Not Null
int	servings		Number of servings, Not Null
varchar	difficulty		Difficulty level: Easy/Medium/Hard, Not Null
varchar	category		Category: Breakfast/Main/Soup/Salad/Dessert, Not Null
uuid	authorId	FK	Foreign Key to User Service, Not Null
varchar	image		Recipe image URL, Nullable
timestamp	createdAt		Auto-generated
timestamp	updatedAt		Auto-updated

CATEGORIES			
uuid	id	PK	Primary Key
varchar	name	UK	Category name, Unique, Not Null
text	description		Category description, Nullable
timestamp	createdAt		Auto-generated
timestamp	updatedAt		Auto-updated

DIFFICULTY_LEVELS			
uuid	id	PK	Primary Key
varchar	level	UK	Difficulty level, Unique, Not Null
text	description		Level description, Nullable
timestamp	createdAt		Auto-generated
timestamp	updatedAt		Auto-updated

Пользовательские сценарии

Показывает **5 основных user flow** - как пользователь взаимодействует с системой от начала до конца.

Сценарий 1: Регистрация и вход

Регистрация:

1. Пользователь → `POST /api/users/register` (username, email, password)
2. Система хэширует пароль через bcrypt
3. Сохранение в таблицу Users
4. Возврат профиля пользователя

Вход:

1. Пользователь → `POST /api/users/login` (email, password)
2. Система проверяет пароль через `bcrypt.compare()`
3. Возврат токена аутентификации

Сценарий 2: Просмотр категорий и рецептов

1. Пользователь → `GET /api/categories`
2. Система отображает: Завтраки, Основные блюда, Супы, Салаты, Десерты
3. Пользователь выбирает "Десерты"
4. → `GET /api/recipes?category=Dessert`
5. **Recipe Service делает запрос к User Service** для получения имен авторов
6. Отображение рецептов с полной информацией об авторах

Сценарий 3: Создание нового рецепта

1. Аутентифицированный пользователь → `GET /api/difficulty`
2. Система показывает форму с категориями и уровнями сложности
3. Пользователь заполняет:
    - Название: "Шоколадный торт"
    - Ингредиенты: "Мука, яйца, шоколад..."
    - Инструкции: "Шаг 1: Разогреть духовку..."
    - Время подготовки: 30 минут
    - Время готовки: 45 минут
    - Порции: 8
    - Сложность: Medium
    - Категория: Dessert
4. → `POST /api/recipes` с authorId текущего пользователя
5. Валидация данных (проверка обязательных полей, корректность чисел)
6. Сохранение в БД recipes
7. Возврат созданного рецепта с ID

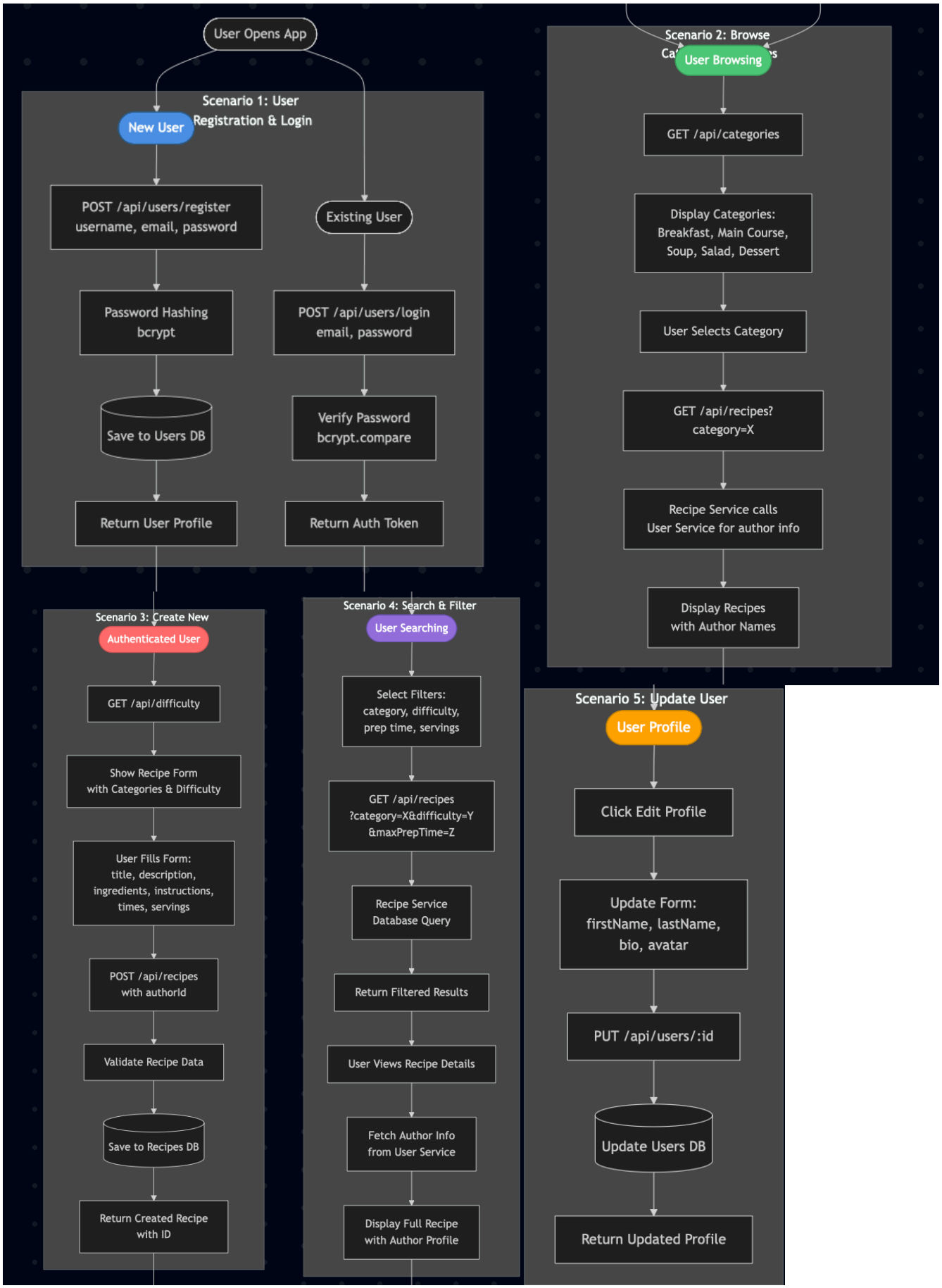
Сценарий 4: Поиск и фильтрация

1. Пользователь выбирает фильтры:
    - Категория: Main Course
    - Сложность: Easy
    - Максимальное время подготовки: 30 минут
    - Порции: 4
2. → `GET /api/recipes?category=Main&difficulty=Easy&maxPrepTime=30&servings=4`
3. Recipe Service выполняет SQL запрос с WHERE условиями
4. Возврат отфильтрованных результатов
5. Пользователь кликает на рецепт
6. → Загрузка деталей + запрос к User Service за данными автора
7. Отображение полного рецепта с профилем автора

Сценарий 5: Обновление профиля

1. Пользователь → "Редактировать профиль"
2. Заполнение формы:
    - Имя: "Иван"
    - Фамилия: "Петров"
    - О себе: "Люблю готовить десерты"
    - Аватар: загрузка изображения
3. → `PUT /api/users/:id`

4. Обновление в таблице Users
5. Возврат обновленного профиля

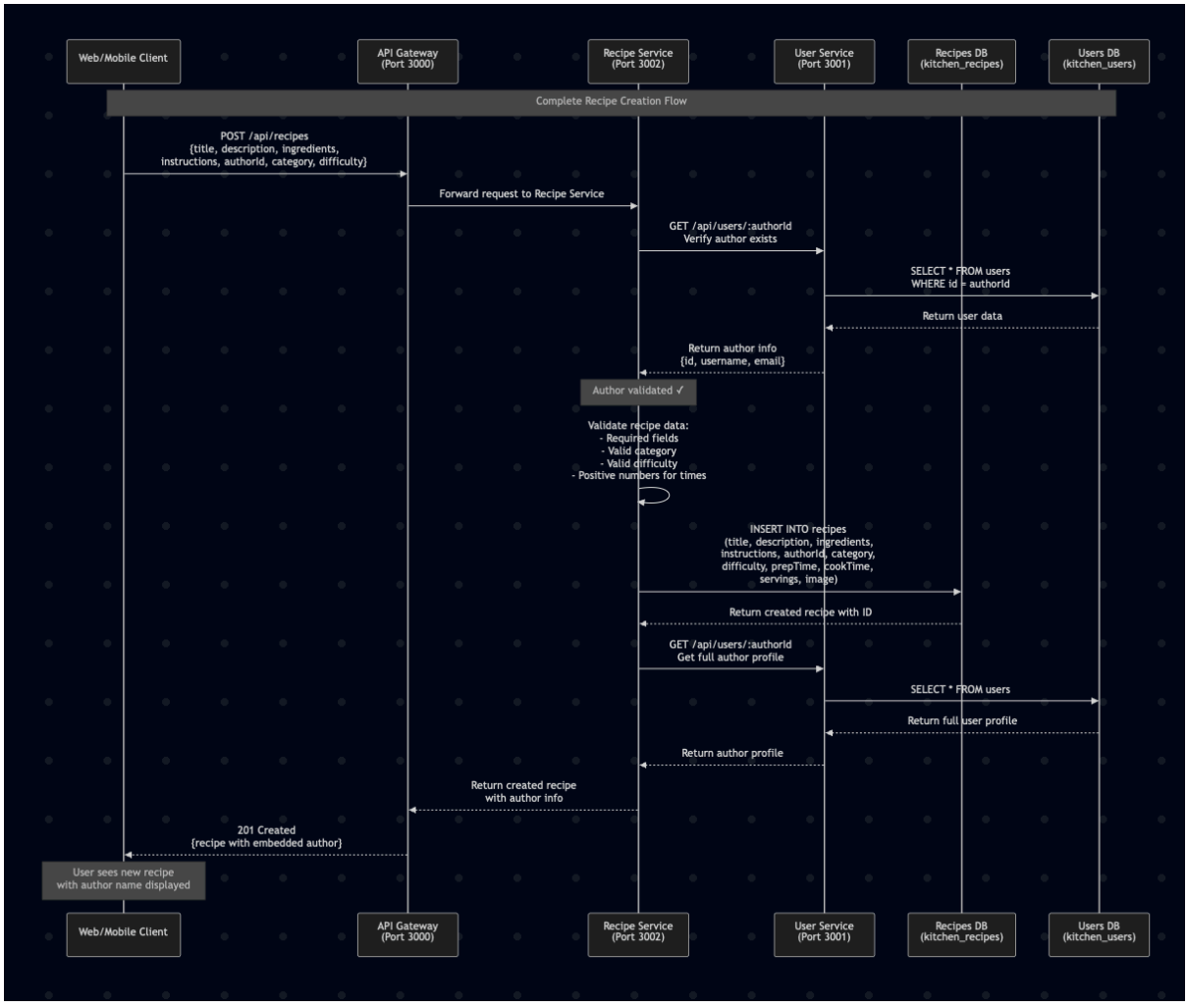


Sequence диаграмма: Создание рецепта

Показывает **детальную последовательность взаимодействий** между компонентами при создании рецепта.

Участники:

- Client (Web/Mobile)
- API Gateway
- Recipe Service
- User Service
- Recipe DB
- User DB



Вывод: В рамках данной домашней работы была спроектирована и задокументирована микросервисная архитектура для веб-приложения "Кулинарный блог". Проект демонстрирует практическое применение современных подходов к разработке распределённых систем.