# Техническое задание на разработку сервиса «Онлайн библиотека»

Ласкин Павел

## 1. Цели и задачи

Основные цели разработки:

- создание высоконагруженного API для управления библиотечным каталогом;
- обеспечение целостности и консистентности данных;
- автоматизация сборки, тестирования и развёртывания (CI/CD);

## 2. Функциональные требования

Сервис должен предоставлять следующий набор операций:

- **Управление книгами:** добавление, изменение, получение по идентификатору. При добавлении книги указывается название и список идентификаторов авторов.
- **Управление авторами:** регистрация, изменение имени, получение по идентификатору.
- **Связь книг и авторов:** получение потока (stream) книг, написанных указанным автором.

## 3. Требования к API

API реализуется по протоколу gRPC с автоматической генерацией REST-эндпоинтов через gRPC-gateway. Спецификация описывается в формате Protocol Buffers (v3). Основные методы:

- `POST /v1/library/book` — добавить книгу;
- `PUT /v1/library/book` — изменить книгу;
- `GET /v1/library/book/{id}` — получить книгу;
- `POST /v1/library/author` — зарегистрировать автора;
- `PUT /v1/library/author` — изменить автора;
- `GET /v1/library/author/{id}` — получить автора;
- `GET /v1/library/author_books/{author_id}` — получить книги автора (server-side stream).

Валидация входных данных осуществляется на стороне сервера с использованием protoc-gen-validate. Все идентификаторы должны соответствовать формату UUID; имя автора должно удовлетворять регулярному выражению `^[A-Za-z0-9]+( [A-Za-z0-9]+)*$` и иметь длину от 1 до 512 символов; название книги — от 1 до 512 символов.

# 4. Технологический стек

- **Язык реализации:** Go 1.22+.
- **Архитектура:** чистая архитектура (Clean Architecture) с разделением на слои (usecase, repository, delivery).
- **Протоколы:** gRPC, REST (через gRPC-gateway).
- **База данных:** PostgreSQL 15, миграции — goose, драйвер — pgx.
- **Логирование:** zap.
- **Конфигурация:** переменные окружения (12-factor app).
- **Контейнеризация:** Docker, оркестрация — Kubernetes (на последующих этапах).
- **CI/CD:** GitHub Actions (сборка, тестирование, публикация образа).

# 5. Требования к данным

Модель данных включает три основные таблицы:

- `author`: `id` (UUID, PK), `name` (TEXT), `created_at`, `updated_at` (timestamp with time zone). Индекс по `name`.
- `book`: `id` (UUID, PK), `name` (TEXT), `created_at`, `updated_at`. Индекс по `name`.
- `author_book`: связь многие-ко-многим (`author_id`, `book_id`) с составным первичным ключом и внешними ключами с каскадным удалением. Дополнительный индекс по `book_id` для оптимизации запросов.

Миграции должны быть написаны на чистом SQL и интегрированы в код через `//go:embed`.

# 6. Этапы разработки

1. Разработка ТЗ и прототипа интерфейса.
2. Проектирование схемы БД, создание скриптов DDL.
3. Разработка BPMN- и Sequence-диаграмм ключевых сценариев.
4. Реализация ядра сервиса с in-memory репозиторием, graceful shutdown.
5. Интеграция с PostgreSQL, написание миграций.
6. Контейнеризация (Docker, Kubernetes), настройка CI/CD.
7. Документирование архитектуры и API.
8. Настройка CI/CD