

Техническое задание на разработку сервиса «Онлайн библиотека»

1. Введение

Настоящий документ определяет техническое задание на разработку сервиса «Онлайн библиотека» (далее — Сервис). Сервис предназначен для хранения и управления информацией о книгах и авторах. Реализация выполняется в рамках цикла лабораторных работ (12 этапов), результатом которых является расширение функциональности или интеграция нового компонента.

2. Назначение и цели

Сервис предоставляет API для:

- добавления, изменения и получения информации о книгах;
- регистрации, изменения и получения информации об авторах;
- получения списка книг конкретного автора.

Цель — создание масштабируемого сервиса с последующим развертыванием в контейнерной среде, настройкой мониторинга и автоматизацией тестирования.

3. Функциональные требования

Сервис должен реализовывать следующие операции:

- **Книги:** добавление (название + идентификаторы авторов), изменение, получение по ID.
- **Авторы:** регистрация (имя), изменение имени, получение по ID.
- **Связь:** получение списка книг указанного автора (потоковая передача).

4. Требования к API

Взаимодействие осуществляется через gRPC и REST (gRPC-gateway). 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 (stream).

Валидация: идентификаторы — UUID; имя автора — формат: латиница, цифры, пробелы между словами, длина 1–512; название книги — длина 1–512. Сообщения содержат поля `created_at` и `updated_at`.

5. Технологические требования

- Язык: **Go**, архитектура — `go-clean-template`.
- Протоколы: gRPC, REST (gRPC-gateway).
- Логирование: **zap**, валидация: `protoc-gen-validate`.
- База данных: **PostgreSQL**, миграции — `goose`.
- Конфигурация: переменные окружения (порты, параметры БД).
- Тестирование: unit-тесты с моками, интеграционные тесты в CI.
- Контейнеризация: Docker, Kubernetes (на следующих этапах).
- CI/CD: Gitlab CI или Jenkins.
- Мониторинг: Grafana + уведомления.
- Интеграция: Kafka, ClickHouse (финальные ЛР).

6. Требования к данным (БД)

Основные сущности:

- **author**: id (UUID, первичный ключ), name, created_at, updated_at (триггер обновления). Индекс по name.
- **book**: id (UUID), name, created_at, updated_at. Индекс по name.
- **author_book**: author_id, book_id (внешние ключи с каскадным удалением), составной первичный ключ. Дополнительный индекс по book_id.

Миграции SQL, встраиваемые через `//go:embed`, применение через `goose`.

7. Состав работ по этапам

1. **ЛР1:** ТЗ, прототип интерфейса (Figma/Balsamiq).
2. **ЛР2:** Схема БД в Oracle SQL Developer Data Modeler, скрипты создания.
3. **ЛР3:** Диаграммы BPMN и Sequence.
4. **ЛР4:** Реализация на Go (in-memory хранилище) с REST/gRPC, валидацией, graceful shutdown.
5. **ЛР5:** Интеграция с PostgreSQL, миграции, репозиторий на БД.
6. **ЛР6:** Docker, Kubernetes, CI/CD.
7. **ЛР7:** Документация по интерфейсам и алгоритмам.
8. **ЛР8:** Автотесты backend (Postman/SOAP UI).
9. **ЛР9:** Решение SQL-задач с stratascratch.com.
10. **ЛР10:** Мониторинг (Grafana, уведомления).
11. **ЛР11:** UI-автотесты (Selenium/Selenide).
12. **ЛР12:** Интеграция Kafka + ClickHouse (агрегация чисел, DLQ).

8. Дополнительные требования

- Обработка SIGINT/SIGTERM.
- Makefile с целями `build` и `generate`.
- Интеграционные тесты в CI, покрытие кода.
- README с описанием, инструкцией по сборке и запуску.