

Документация сервиса «Онлайн библиотека»

Лабораторная работа №7

Содержание

1	220. Архитектура системы	2
1.1	Общее описание	2
1.2	Технологический стек	2
1.3	Структура проекта	2
1.4	Компоненты и их взаимодействие	2
1.5	База данных (проектируемая)	3
2	230. Сервисы (API)	5
2.1	Общая информация	5
2.2	Описание методов API	5
2.2.1	Добавление книги	5
2.2.2	Обновление книги	6
2.2.3	Получение информации о книге	6
2.2.4	Регистрация автора	7
2.2.5	Изменение информации об авторе	7
2.2.6	Получение информации об авторе	8
2.2.7	Получение книг автора (поток)	8
3	240. Пользовательский интерфейс (UI)	9
3.1	Общая структура интерфейса	9
3.2	Описание экранов	9
3.2.1	Главная страница / Поиск	9
3.2.2	Страница книги	10
3.2.3	Страница автора	10
3.2.4	Формы добавления/редактирования	11
3.3	Сценарии использования	11
3.4	Соответствие API	12

1 220. Архитектура системы

1.1 Общее описание

Сервис «Онлайн библиотека» реализован на языке Go и предназначен для управления информацией о книгах и авторах. Взаимодействие с сервисом осуществляется через **gRPC** и **REST** (с использованием gRPC-gateway). Данные хранятся в **PostgreSQL**. Проект следует принципам **чистой архитектуры (Clean Architecture)**, что обеспечивает независимость бизнес-логики от внешних фреймворков и баз данных.

1.2 Технологический стек

Компонент	Технология
Язык программирования	Go 1.22
Протоколы API	gRPC, REST (через gRPC-gateway)
База данных	PostgreSQL 15
Миграции	goose
Логирование	zap
Валидация	protoc-gen-validate
Конфигурация	Переменные окружения (env)
Контейнеризация	Docker
Оркестрация	Kubernetes (планируется)

1.3 Структура проекта

```
api/                                # Protobuf-спецификации
  library.proto                     # Описание сервиса Library и сообщений
cmd/
  library/                          # Точка входа в приложение
    main.go
config/                             # Работа с конфигурацией
  config.go
db/
  migrations/                      # SQL-миграции для PostgreSQL
internal/                          # Внутренние пакеты (не экспортируются)
  app/
    grpc/                         # Реализация gRPC-сервера (хендлеры)
    server/                       # Запуск gRPC и HTTP-gateway серверов
  entity/                         # Бизнес-сущности (Author, Book)
  usecase/                       # Бизнес-логика (интерфейсы и реализации)
  repo/                          # Репозитории (inmemory, postgres)
pkg/
  api/                            # Сгенерированный код из proto (grpc, gateway)
go.mod, go.sum
Makefile                          # Сборка, генерация кода, тесты
```

1.4 Компоненты и их взаимодействие

Система состоит из следующих логических слоёв:

- **Транспортный слой (API)**

- **gRPC-сервер** – реализует интерфейсы, описанные в `library.proto`. Преобразует gRPC-вызовы в вызовы usecase-слоя.
- **gRPC-gateway** – принимает REST-запросы, транслирует их в gRPC и возвращает ответы в JSON.

- **Слой бизнес-логики (Usecase)**

- Содержит интерфейсы репозитория и реализацию методов: добавление книги, регистрация автора и т.д. Не зависит от деталей хранения данных.

- **Слой репозитория (Repository)**

- Реализует доступ к данным. В текущей версии используется **in-memory** хранилище, но подготовлена интеграция с **PostgreSQL** (миграции, драйвер `pgx`).

- **Слой сущностей (Entity)**

- Определяет основные структуры данных (`Author`, `Book`), которые используются во всех слоях.

Взаимодействие компонентов (на примере запроса `AddBook`):

1. Клиент отправляет REST-запрос `POST /v1/library/book` на gRPC-gateway.
2. Gateway преобразует его в gRPC-вызов `AddBook` и направляет на gRPC-сервер.
3. gRPC-сервер вызывает соответствующий метод `AddBook` в usecase-слое.
4. Usecase валидирует данные (с использованием правил из `library.proto`) и обращается к репозиторию для сохранения книги.
5. Репозиторий (in-memory или БД) выполняет операцию и возвращает результат.
6. Ответ проходит обратную цепочку к клиенту.

1.5 База данных (проектируемая)

На основе миграций в `db/migrations` разработана следующая схема:

```
1  --
2  CREATE TABLE author (
3      id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
4      name TEXT NOT NULL,
5      created_at TIMESTAMP DEFAULT now() NOT NULL,
6      updated_at TIMESTAMP DEFAULT now() NOT NULL
7  );
8
9  --      updated_at
10 CREATE TRIGGER trigger_update_author_timestamp BEFORE UPDATE ON author
11 FOR EACH ROW EXECUTE FUNCTION update_author_timestamp();
12
13 --
14 CREATE TABLE book (
15     id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
```

```

16     name TEXT NOT NULL,
17     created_at TIMESTAMP DEFAULT now() NOT NULL,
18     updated_at TIMESTAMP DEFAULT now() NOT NULL
19 );
20
21 --     updated_at
22 CREATE TRIGGER trigger_update_book_timestamp BEFORE UPDATE ON book
23 FOR EACH ROW EXECUTE FUNCTION update_book_timestamp();
24
25 -- --
26 CREATE TABLE author_book (
27     author_id UUID NOT NULL REFERENCES author(id) ON DELETE CASCADE,
28     book_id UUID NOT NULL REFERENCES book(id) ON DELETE CASCADE,
29     PRIMARY KEY (author_id, book_id)
30 );
31
32 --
33 CREATE INDEX idx_author_name ON author(name);
34 CREATE INDEX idx_book_name ON book(name);
35 CREATE INDEX idx_author_book_book_id ON author_book(book_id);

```

2 230. Сервисы (API)

2.1 Общая информация

- **Базовый URL для REST:** `http://<host>:<grpc_gateway_port>/v1/library/`
- **Форматы данных:** JSON (REST), Protocol Buffers (gRPC)
- **Валидация:** осуществляется на основе правил, заданных в `library.proto` (используется `protoc-gen-validate`). Все идентификаторы должны быть в формате UUID, имена авторов и книг проходят проверку длины и допустимых символов.
- **Коды ошибок (REST):**
 - 200 OK – успех
 - 400 Bad Request – ошибка валидации
 - 404 Not Found – ресурс не найден
 - 500 Internal Server Error – внутренняя ошибка сервера

2.2 Описание методов API

2.2.1 Добавление книги

Метод: `AddBook`

HTTP: `POST /v1/library/book`

gRPC: `AddBook(AddBookRequest) returns (AddBookResponse)`

Поле	Тип	Обязательное	Валидация	Описание
AddBookRequest				
<code>name</code>	<code>string</code>	да	1–512 символов	Название книги
<code>author_ids</code>	<code>repeated string</code>	да	каждый элемент – валидный UUID	Идентификаторы авторов
AddBookResponse				
<code>book</code>	<code>Book</code>	–	–	Созданная книга

Алгоритм работы:

1. Валидация входных данных (длина `name`, корректность `author_ids`).
2. Проверка существования всех указанных авторов в репозитории.
3. Если все авторы существуют:
 - Генерация нового UUID для книги.
 - Сохранение книги в хранилище.
 - Создание связей между книгой и каждым автором.
 - Возврат полной информации о книге (с заполненными `created_at` и `updated_at`).
4. Если хотя бы один автор не найден – возврат ошибки 404 Not Found.

Пример запроса (curl):

```
curl -X POST http://localhost:8081/v1/library/book \
-H "Content-Type: application/json" \
-d '{"name": "", "author_ids": ["550e8400-e29b-41d4-a716-446655440000"]}'
```

Пример ответа:

```
{
  "book": {
    "id": "f47ac10b-58cc-4372-a567-0e02b2c3d479",
    "name": "",
    "author_id": ["550e8400-e29b-41d4-a716-446655440000"],
    "created_at": "2025-02-20T10:00:00Z",
    "updated_at": "2025-02-20T10:00:00Z"
  }
}
```

2.2.2 Обновление книги

Метод: UpdateBook

HTTP: PUT /v1/library/book

gRPC: UpdateBook(UpdateBookRequest) returns (UpdateBookResponse)

Поле	Тип	Обязательное	Валидация	Описание
UpdateBookRequest				
id	string	да	валидный UUID	Идентификатор книги
name	string	да	1–512 символов	Новое название книги
author_ids	repeated string	да	каждый элемент – валидный UUID	Новый список авторов
UpdateBookResponse			– пустой (код 200 при успехе)	

Алгоритм работы:

1. Валидация входных данных.
2. Поиск книги по id. Если не найдена – ошибка 404.
3. Проверка существования всех указанных авторов.
4. В рамках транзакции:
 - Обновление названия книги.
 - Удаление старых связей в `author_book`.
 - Вставка новых связей.
5. Возврат успеха.

2.2.3 Получение информации о книге

Метод: GetBookInfo

HTTP: GET /v1/library/book/{id}

gRPC: GetBookInfo(GetBookInfoRequest) returns (GetBookInfoResponse)

Поле	Тип	Обязательное	Валидация	Описание
GetBookInfoRequest				
id	string	да	валидный UUID	Идентификатор книги
GetBookInfoResponse				
book	Book	–	–	Найденная книга

Алгоритм:

1. Поиск книги по id.
2. Если книга найдена – возврат полной информации.
3. Если нет – ошибка 404.

2.2.4 Регистрация автора

Метод: RegisterAuthor

HTTP: POST /v1/library/author

gRPC: RegisterAuthor(RegisterAuthorRequest) returns (RegisterAuthorResponse)

Поле	Тип	Обязательное	Валидация	Описание
RegisterAuthorRequest				
name	string	да	$[A - Za - z0 - 9] + ([A - Za - z0 - 9] +) * \$$ длина 1-512	Имя автора
RegisterAuthorResponse				
id	string	–	–	Сгенерированный UUID автора

Алгоритм работы:

1. Валидация имени.
2. Генерация UUID и сохранение автора в хранилище.
3. Возврат ID.

2.2.5 Изменение информации об авторе

Метод: ChangeAuthorInfo

HTTP: PUT /v1/library/author

gRPC: ChangeAuthorInfo(ChangeAuthorInfoRequest) returns (ChangeAuthorInfoResponse)

Поле	Тип	Обязательное	Валидация	Описание
ChangeAuthorInfoRequest				
id	string	да	валидный UUID	Идентификатор автора
name	string	да	$[A - Za - z0 - 9] + ([A - Za - z0 - 9] +) * \$$ длина 1-512	Новое имя
ChangeAuthorInfoResponse				– пустой

Алгоритм работы:

1. Поиск автора по `id`. Если не найден – 404.
2. Обновление имени автора.
3. Возврат успеха.

2.2.6 Получение информации об авторе

Метод: `GetAuthorInfo`

HTTP: `GET /v1/library/author/{id}`

gRPC: `GetAuthorInfo(GetAuthorInfoRequest) returns (GetAuthorInfoResponse)`

Поле	Тип	Обязательное	Валидация	Описание
GetAuthorInfoRequest				
<code>id</code>	<code>string</code>	да	валидный UUID	Идентификатор автора
GetAuthorInfoResponse				
<code>id</code>	<code>string</code>	–	–	UUID автора
<code>name</code>	<code>string</code>	–	–	Имя автора

Алгоритм:

1. Поиск автора по `id`.
2. Если найден – возврат данных, иначе 404.

2.2.7 Получение книг автора (поток)

Метод: `GetAuthorBooks`

HTTP: `GET /v1/library/author_books/{author_id}`

gRPC: `GetAuthorBooks(GetAuthorBooksRequest) returns (stream Book)`

Поле	Тип	Обязательное	Валидация	Описание
GetAuthorBooksRequest				
<code>author_id</code>	<code>string</code>	да	валидный UUID	Идентификатор автора

Ответ: поток объектов `Book` (по одному сообщению на книгу).

Алгоритм:

1. Проверка существования автора (если нет – можно вернуть пустой поток или ошибку, реализация различается).
2. Запрос к хранилищу для получения всех книг, связанных с данным автором.
3. Отправка каждой книги по мере чтения из курсора (`stream`).

3 240. Пользовательский интерфейс (UI)

В папке `2_user_interface` представлены прототипы пользовательского интерфейса, разработанные в рамках лабораторной работы №2. Прототипы выполнены в стиле wireframe и отражают базовую функциональность сервиса. Ниже приведено описание основных экранов и навигации.

3.1 Общая структура интерфейса

Интерфейс состоит из четырёх основных экранов:

- Главная страница (поиск)
- Страница книги
- Страница автора
- Формы добавления/редактирования

Навигация между экранами осуществляется через ссылки на связанные объекты и кнопки действий.

3.2 Описание экранов

3.2.1 Главная страница / Поиск

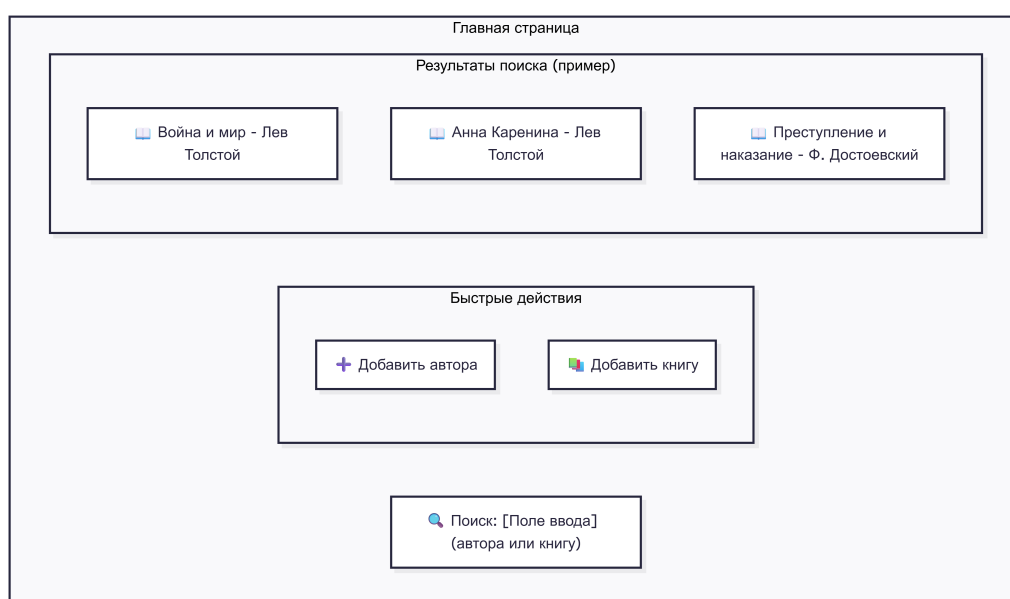


Рис. 1: Главная страница (прототип)

Элементы:

- **Шапка:** логотип «Онлайн библиотека».
- **Строка поиска:** поле ввода и кнопка «Найти». Поиск осуществляется по названиям книг и именам авторов.
- **Быстрые действия:** кнопки «Добавить автора» и «Добавить книгу» (ведут на соответствующие формы).

- **Результаты поиска:** отображаются в виде списка карточек, каждая содержит название книги и имя автора. Клик по карточке ведёт на страницу книги.

3.2.2 Страница книги

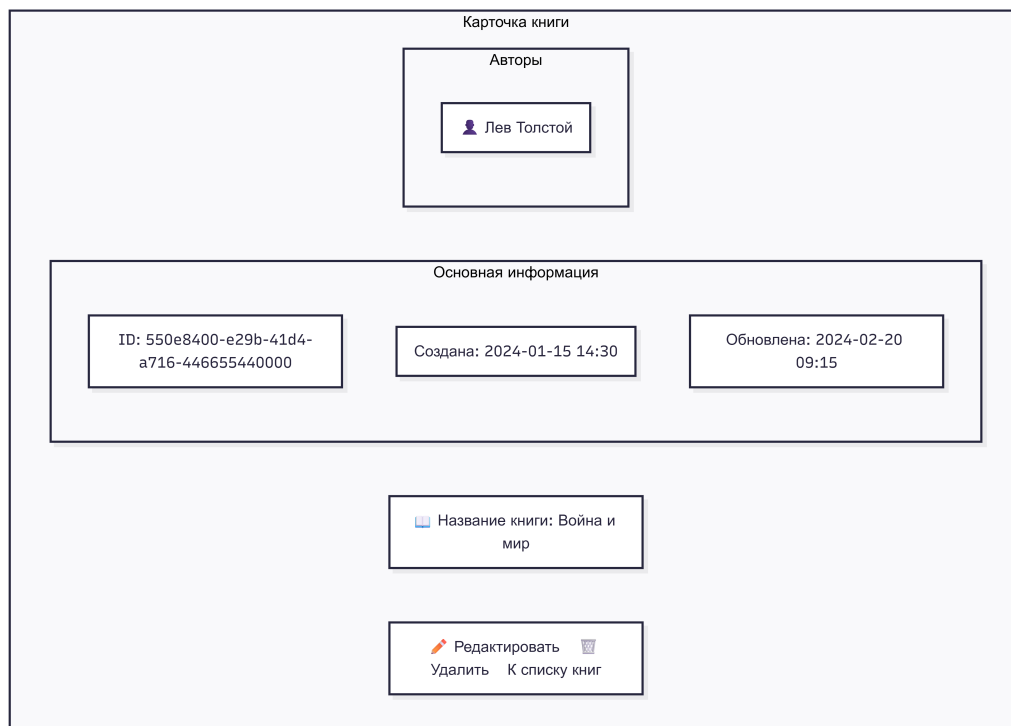


Рис. 2: Страница книги (прототип)

Элементы:

- **Заголовок:** название книги.
- **Блок информации:** ID книги, дата создания, дата последнего обновления.
- **Список авторов:** имена авторов (каждое имя является ссылкой на страницу соответствующего автора).
- **Кнопки:** «Редактировать» (ведёт на форму редактирования), «Удалить» (с подтверждением), «Назад» (возврат к результатам поиска).

3.2.3 Страница автора

Элементы:

- **Заголовок:** имя автора.
- **Блок информации:** ID автора, дата регистрации, дата последнего обновления.
- **Список книг автора:** перечень книг, написанных автором. Каждое название является ссылкой на страницу книги.
- **Кнопки:** «Редактировать» (ведёт на форму редактирования автора), «Удалить» (с подтверждением), «Назад».

Карточка автора

Книги автора

- Война и мир (1869)
- Анна Каренина (1877)
- Детство (1852)

Информация

- ID: 123e4567-e89b-12d3-a456-426614174000
- Зарегистрирован: 2026-10-01
- Обновлен: 2026-11-01

Имя: Лев Толстой

Редактировать Удалить Назад

Рис. 3: Страница автора (прототип)

3.2.4 Формы добавления/редактирования

Форма добавления/редактирования автора:

- Поле «Имя автора» с подсказкой о допустимых символах (латиница, цифры, пробелы).
- Кнопки «Сохранить» и «Отмена».

Форма добавления/редактирования книги:

- Поле «Название».
- Блок «Авторы»:
 - Выпадающие списки для выбора существующих авторов (можно добавить несколько).
 - Кнопка «Добавить ещё автора» для увеличения количества полей.
- Примечание: можно выбрать несколько авторов.
- Кнопки «Сохранить» и «Отмена».

3.3 Сценарии использования

Сценарий 1: Добавление новой книги

1. Пользователь нажимает кнопку «Добавить книгу» на главной.
2. Заполняет форму: вводит название, выбирает авторов из выпадающих списков.
3. Нажимает «Сохранить».

4. Система создаёт книгу и перенаправляет пользователя на её страницу.

Сценарий 2: Просмотр всех книг автора

1. Пользователь находит автора через поиск или переходит по ссылке со страницы книги.
2. На странице автора отображается список его книг.
3. Пользователь может кликнуть на любую книгу для просмотра деталей.

Сценарий 3: Редактирование информации об авторе

1. На странице автора нажимает «Редактировать».
2. Вносит изменения в поле имени.
3. Нажимает «Сохранить» – данные обновляются.

3.4 Соответствие API

Элемент интерфейса	Метод API
Форма добавления книги	AddBook
Форма редактирования книги	UpdateBook
Страница книги	GetBookInfo
Форма добавления автора	RegisterAuthor
Форма редактирования автора	ChangeAuthorInfo
Страница автора	GetAuthorInfo
Список книг на странице автора	GetAuthorBooks