1. Общая архитектура системы

Система «Домашняя библиотека» реализована в виде **клиент-серверного веб-приложения**, построенного на основе **REST-архитектуры**.
Клиентская часть взаимодействует с сервером посредством HTTP-запросов, передавая и получая данные в формате JSON.

Серверная часть реализует бизнес-логику, управление доступом, работу с библиотеками, книгами, закладками и цитатами, а также взаимодействие с базой данных и файловым хранилищем.

Архитектура системы является **модульной**, что обеспечивает:

- логическое разделение ответственности;
- упрощение сопровождения;
- возможность расширения функциональности.

---

2. Перечень программных компонентов системы

Система состоит из следующих логических модулей:

- Модуль пользователя
- Модуль управления библиотекой
- Модуль чтения
- Модуль закладок и цитат
- Модуль поиска
- Модуль хранения данных

Каждый модуль включает контроллеры и сервисы, реализующие соответствующую функциональность.

---

3. Описание модулей и их ответственностей

3.1. Модуль пользователя

Назначение

Обеспечивает аутентификацию пользователей и проверку прав доступа к ресурсам системы.

Состав

- Контроллер пользователя
- Сервис авторизации
- Сервис прав доступа

Ответственности

- обработка запросов на вход пользователя в систему;
- проверка подлинности пользователя;
- определение прав пользователя на доступ к библиотекам и книгам.

Предоставляемые интерфейсы

- REST API для операций аутентификации;
- интерфейс проверки прав доступа для других модулей.

3.2. Модуль управления библиотекой

Назначение

Управляет библиотеками, их созданием и доступом пользователей.

Состав

- Контроллер библиотек
- Сервис библиотек
- Сервис приглашений

Ответственности

- создание и удаление библиотек;
- управление списком библиотек пользователя;
- отправка приглашений другим пользователям;
- обработка принятия приглашений.

Особенности

При принятии приглашения:

- создаётся запись о доступе пользователя к библиотеке;
- статус приглашения изменяется.

3.3. Модуль чтения

Назначение

Обеспечивает работу с книгами и прогрессом чтения.

Состав

- Контроллер чтения
- Сервис книг
- Сервис файлов

Ответственности

- добавление и удаление книг из библиотек;
- получение списка книг;
- сохранение прогресса чтения пользователя;
- работа с файлами книг и обложек.

3.4. Модуль закладок и цитат

Назначение

Обеспечивает создание и управление пользовательскими аннотациями.

Состав

- Контроллер аннотаций
- Сервис закладок
- Сервис цитат

Ответственности

- создание и удаление закладок;
- добавление и удаление цитат;
- привязка цитат к конкретным закладкам и страницам.

3.5. Модуль поиска

Назначение

Обеспечивает поиск и фильтрацию книг.

Состав

- Сервис поиска

Ответственности

- поиск книг по названию, автору и жанру;
- фильтрация книг внутри библиотеки.

3.6. Модуль хранения данных

Назначение

Обеспечивает доступ к базе данных.

Состав

- Репозиторий данных

Ответственности

- выполнение SQL-запросов;
- изоляция бизнес-логики от конкретной СУБД;
- централизованный доступ к данным.

---

4. Описание интерфейсов взаимодействия

4.1. REST API

REST API является **основным интерфейсом взаимодействия** между клиентом и сервером.

Характеристики:

- протокол: HTTP/HTTPS;
- формат данных: JSON;
- методы: GET, POST, PUT, DELETE;
- статусы ответов: 200, 201, 400, 401, 403, 404.

REST API предоставляет доступ ко всем функциям системы:

- управлению библиотеками;
- работе с книгами;
- приглашениям;
- закладкам и цитатам.

4.2. Внутренние интерфейсы сервисов

Внутренние интерфейсы используются для взаимодействия между сервисами и обеспечивают:

- слабую связанность компонентов;
- возможность замены реализации без изменения внешнего API.

---

5. Структуры передаваемых данных

Взаимодействие по REST API осуществляется с использованием JSON-структур.

Примеры структур данных:

- данные библиотеки;
- данные книги;
- информация о приглашении;

- данные закладки и цитаты;
- информация о прогрессе чтения.

Передаваемые структуры не зависят от физической структуры базы данных и используются как DTO-объекты.

---

6. Основные сценарии взаимодействия

6.1. Создание библиотеки

1. Пользователь отправляет запрос на создание библиотеки.
2. Контроллер проверяет авторизацию.
3. Сервис библиотек создаёт библиотеку.
4. Пользователь автоматически получает права владельца.

6.2. Отправка приглашения

1. Владелец библиотеки отправляет приглашение.
2. Сервис приглашений сохраняет приглашение в базе данных.
3. Получатель может принять или отклонить приглашение.

6.3. Принятие приглашения

1. Пользователь принимает приглашение.
2. Создаётся запись о доступе к библиотеке.
3. Статус приглашения изменяется.

6.4. Добавление закладки и цитаты

1. Пользователь обновляет прогресс чтения.
2. Создаётся закладка.
3. К закладке добавляются цитаты.

---

7. Архитектура развёртывания

Система развёртывается по классической клиент-серверной модели.

- Клиент: веб-браузер пользователя.
- Сервер приложений: Python Flask + Gunicorn.
- Обратный прокси: Nginx с поддержкой TLS.
- База данных: PostgreSQL.
- Хранилище файлов: локальная файловая система.

Такое развёртывание обеспечивает:

- масштабируемость;
- безопасность;
- удобство сопровождения.

---

8. Использование Swagger (OpenAPI)

Swagger применяется для формального описания REST-интерфейса системы.

Он позволяет:

- задокументировать все эндпоинты;
- зафиксировать формат запросов и ответов;
- упростить тестирование API;
- обеспечить единый контракт между клиентом и сервером.

Swagger-описание не заменяет архитектурные диаграммы, а дополняет их, описывая внешний интерфейс системы.