# 

**КУРСОВОЙ ПРОЕКТ**

**Разработка клиент-серверного приложения для автоматизации работы библиотеки**

по дисциплине «Конструирование программного обеспечения»

Выполнили:

студенты гр. 3530904/80105,

3530904/80106 Василевский А. Н.

Луценко Д. Ю.

Цыгарова И. Н.  
Шеве Н. М.

Руководитель: Иванов А. С.

# Задание

Разработать клиент-серверное приложение для автоматизации работы библиотеки. Список обязательных к использованию технологий при выполнении работы:

**Требования**

* Java 12 и выше
* База данных на выбор - MS SQL.
* Hibernate для работы с БД
* Spring Security для обеспечения авторизации пользователю.
* Взаимодействие с клиентом осуществляется посредством REST API.

**User Story**

Как клиенту мне удобно искать книги по названиям, чтобы быстро найти те, которые мне нужны и видеть актуальную информацию о них.

Как работнику библиотеки мне удобно смотреть записи в электронном журнал, чтобы быстро найти клиента и посмотреть задолженности в возврате книг.

Как работнику библиотеки мне удобно изменять записи в электронном журнале, чтобы всегда иметь актуальную информацию о каждом клиенте и книгах, которые он взял или вернул.

Как работнику библиотеки мне удобно удалять информацию о книгах, которые уже не будут доступны в библиотеке.

Как работнику библиотеки мне удобно изменять информацию о книгах, например их доступное количество, чтобы клиенты и работники всегда имели актуальную информацию о них.

Как работнику библиотеки мне удобно изменять информацию о клиентах, чтобы всегда иметь актуальные персональные данные каждого клиента.

**Автоматизированная работа библиотеки**

**(Person)**

Клиент библиотеки с разными книгами

**Система библиотеки**

(Программная система)

Хранит всю информацию о книгах и их количества.

Получает информацию о библиотеке

**Система клиента библиотеки**

(Программная система)

Позволяет клиентам просматривать информацию о количестве книг

Просмотр и остаток книжек в библиотеке

**Автоматизированная работа библиотеки**

**(Person)**

Клиент библиотеки с разными книгами

Просмотр и остаток книжек в библиотеке

Посетить http://localhost:8080/login

**Web-Application**

**(Container: Java and Spring MVC)**

Предоставляет статический контент и многостраничное приложение для библиотеки

**Single-page Application**

**(Container: JavaScript)**

Предоставляет функционал библиотеки клиентам, через их веб-браузер

Переносит клиента в веб-браузер

Происходят вызовы по API (JSON/HTTPS)

**API Application**

**(Container: Java and Spring MVC)**

Обеспечивает функциональность библиотеки, через JSON/HTTPS запросы

Читаем и записываем информацию (JDBC)

**DATABASE**

**(Container: MySQL Database Schema)**

Хранит информацию о книгах и изменения в библиотеке, данные пользователей

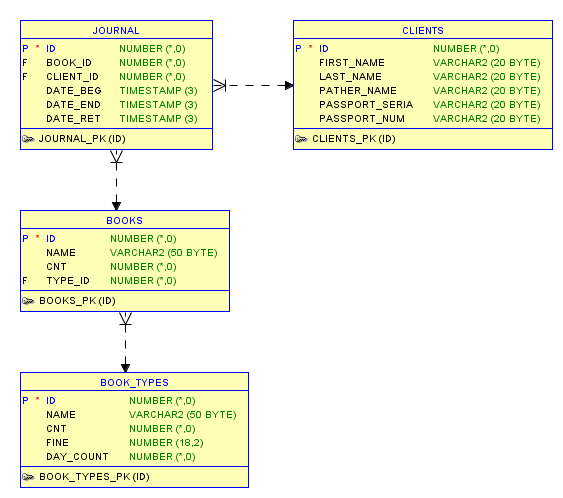
Происходят вызовы по API (XML/HTTPS)

**Mainframe Library System**

(**Software System**)

Хранит всю информацию о книгах и их количества.

Схема базы данных:



|  |  |
| --- | --- |
| **Имя таблицы** | **Расшифровка** |
| clients | Читатели (фамилия, имя, паспорт) |
| books | Книги (наименование, количество) |
| book\_types | Типы книг (название, штраф, число дней) |
| journal | Журнал библиотекаря |

Связи между таблицами:

|  |  |  |
| --- | --- | --- |
| **Название** | **Primary Key** | **Foreign Key** |
| fk\_journal\_clients | clients.id | journal.client\_id |
| fk\_journal\_books | books.id | journal.book\_id |
| fk\_books\_book\_types | book\_types.id | books.type\_id |

# Архитектура проекта (Spring Boot)

Rest

# Диаграмма классов

Изображение выглядит как монитор, внутренний, электроника, компьютер

Автоматически созданное описание

# Изображение выглядит как снимок экрана, электроника, монитор, компьютер Автоматически созданное описание Описание схемы базы данных

Изображение выглядит как электроника, снимок экрана, компьютер, монитор

Автоматически созданное описаниеИзображение выглядит как электроника, монитор, компьютер, снимок экрана

Автоматически созданное описаниеИзображение выглядит как электроника, снимок экрана, монитор, ноутбук

Автоматически созданное описание

# Описание REST API

|  |  |  |  |
| --- | --- | --- | --- |
| Запрос | Входные данные | Выходные данные | Описание |
| /addJournalRecord | JournalRecord | JournalRecord | Добавить  Запись в журнал |
| /addClient | Client | Client | Добавить клиента |
| /addBook | Book | Book | Добавить книгу |
| /addBookType | BookType | BookType | Добавить тип книги |
| /journal | - | ResponseEntity<List<JournalRecord>> | Журнал |
| /clients | - | ResponseEntity<List< Client >> | Все клиенты |
| /books | - | ResponseEntity<List< Book >> | Все книги |
| /bookTypes | - | ResponseEntity<List< BookType >> | Все типы книг |
| /journal/{id} | -, если PUT запрос, то JournalRecord | GET/PUT: ResponseEntity<List<JournalRecord>>  DELETE: void | Получить/удалить/изменить  запись в журнале |
| /client/{id} | -, если PUT запрос, то Client | GET/PUT:  ResponseEntity<List< Client >>  DELETE: void | Получить/удалить/изменить  клиента |
| /book/{id} | -, если PUT запрос, то Book | GET/PUT:  ResponseEntity<List< Book >>  DELETE: void | Получить/удалить/изменить  книгу |
| /bookType/{id} | -, если PUT запрос, то BookType | GET/PUT:  ResponseEntity<List< BookType >>  DELETE: void | Получить/удалить/изменить  тип книги |
| /client/{id}/books | - | ResponseEntity<Set<Book>> | Получить все книги конкретного клиента |

Frontend был реализован с помощью HTML, JS, CSS и фреймворка Bootstrap, также был использован шаблонизатор – Thymeleaf.