Министерство образования и науки РФ

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа программной инженерии

Отчёт по курсовой работе

по дисциплине

«Объектно-ориентированное программирование»

Выполнил студент гр. 3530904/80002 Андреева Е.Д.

Руководитель Аристаева О.Ц.

Оглавление

[Постановка задачи 3](#_Toc41353187)

[Описание классов 5](#_Toc41353188)

[Backend: 5](#_Toc41353189)

[Frontend: 7](#_Toc41353190)

[REST API 8](#_Toc41353191)

[Заключение 9](#_Toc41353192)

Постановка задачи

Разработать клиент-серверное приложение на заданную тему (см. распределение в приложенном файле). Список обязательных к использованию технологий при выполнении работы:

1. Клиент:

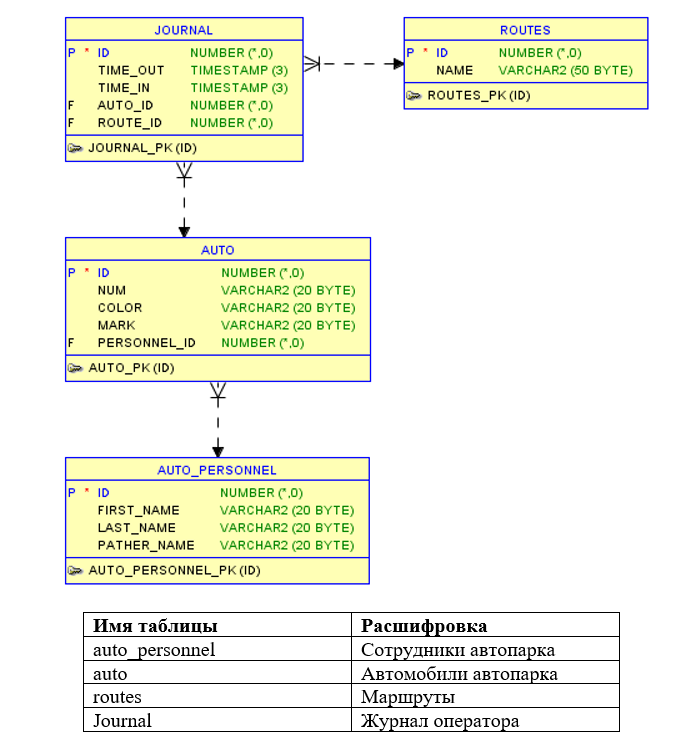
* требования отсутствуют, можно использовать любые известные языки.

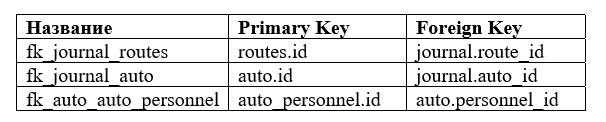
1. Сервер:

* Java 12 и выше.
* База данных на выбор - SLQLite/PostreSQL/MS SQL/Oracle. Схему БД, соответствующую заданию, см. ниже.
* SpringData/Hibernate для работы с БД.
* Spring Security.
* Взаимодействие с клиентом осуществляется посредством REST API.

Предварительно составленные функциональные требования на приложение перед реализацией необходимо согласовать с преподавателем.

Вариант 1: Автоматизация работы автопарка

Схема базы данных:

Связи между таблицами:

Описание классов

Backend:

***Config*** – предоставляет механизм построения систем идентификации, аутентификации и авторизации.

*MvcConfig* связывает *http-*запросы с одноименными *jsp* файлами, которые рисуют графику страницы.

*WebSecurityConfig* отдельный класс, в котором прописано, кому принадлежат ссылки.

***Controller*** – обрабатывает запросы пользователя, использует класс *SimpleResponse*, который создает объекты ответов на запрос, у которых есть поля сообщение об ошибке и состояние запроса: успешно или нет.

Для полей машины, журнал, регистрация, маршруты и пользователи классы соответственно: *AutoRestController*, *JournalRestController*, *RegistrationRestController, RoutesRestController, RoutesRestController*.

В них включены следующие методы:

*addOrUpdateSale* – метод, с помощью которого добавляется новая запись в таблице.

*deleteSaleById* – позволяет удалить запись по id.

*getAllSales* – возвращает все записи.

***Entity*** – интерфейс над БД, в котором прописаны ключевые поля и связи.

В каждом классе присутствуют методы *get/set*, которые работают с полями *id, number, color, mark, route, auto, role* и другими.

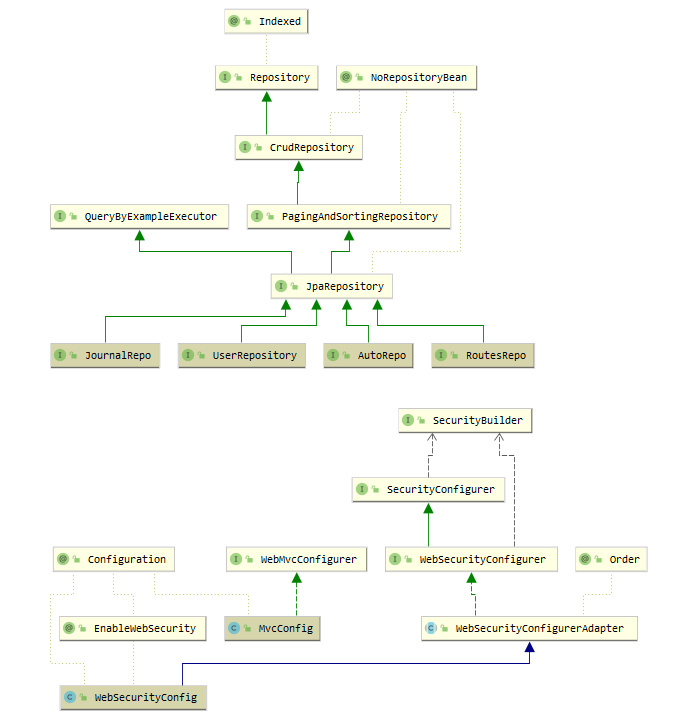
***Repository*** – основное понятие в *Spring Data*. Это несколько интерфейсов, которые используют *JPA Entity* для взаимодействия с ней.

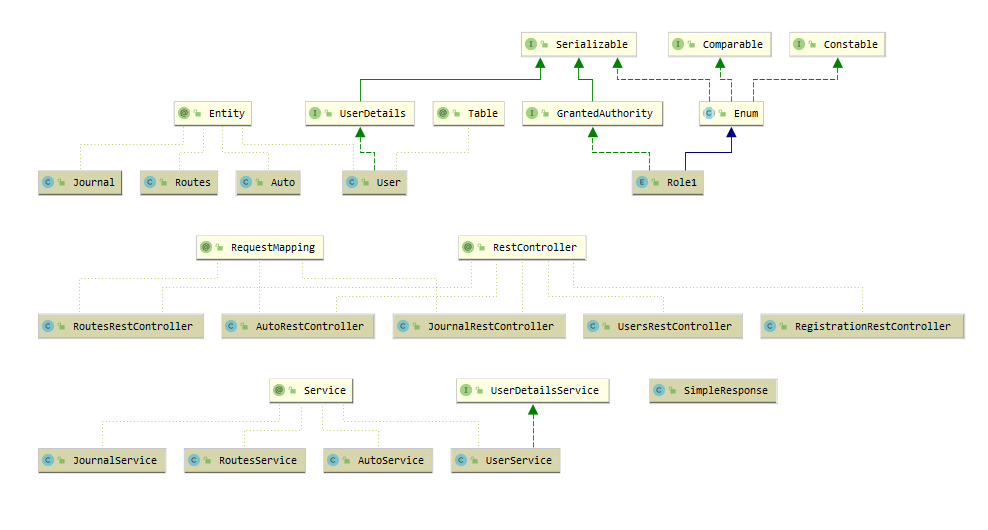
Классы также соответствуют ключевым полям. Методы определены частично, остальные определяются автоматически благодаря технологии *JPA.* Ключевые методы – *findBy/deleteBy.*

***Service*** – используется для взаимодействия с БД, уточнение репозитория.

**Схема всей работы**:

Запрос → Контроллер → Сервис → Репозиторий → Энтити.

******



Frontend:

***CSS*** – описание стилей кнопок и таблиц.

***JSP*** – каждый файл включает код на *HTML*, отвечающий за прорисовку статической части, и код на *JS*, отвечающий за динамическую часть страницы, как например сортировка *html*-таблицы. Расширение *.jsp* позволяет соединить два языка. Стили берутся из *CSS*.

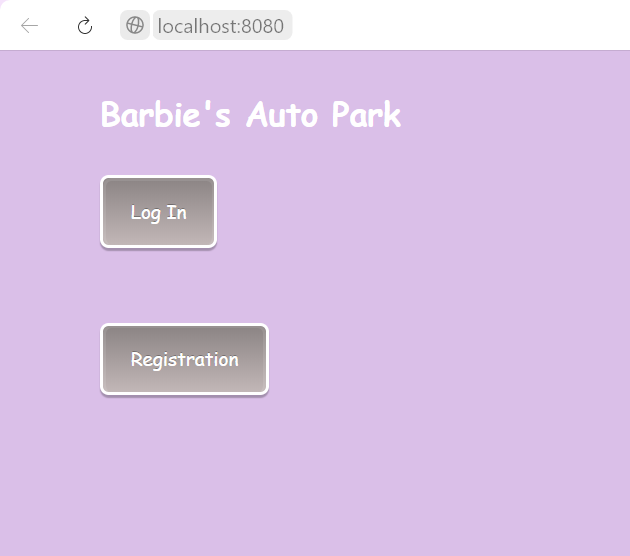
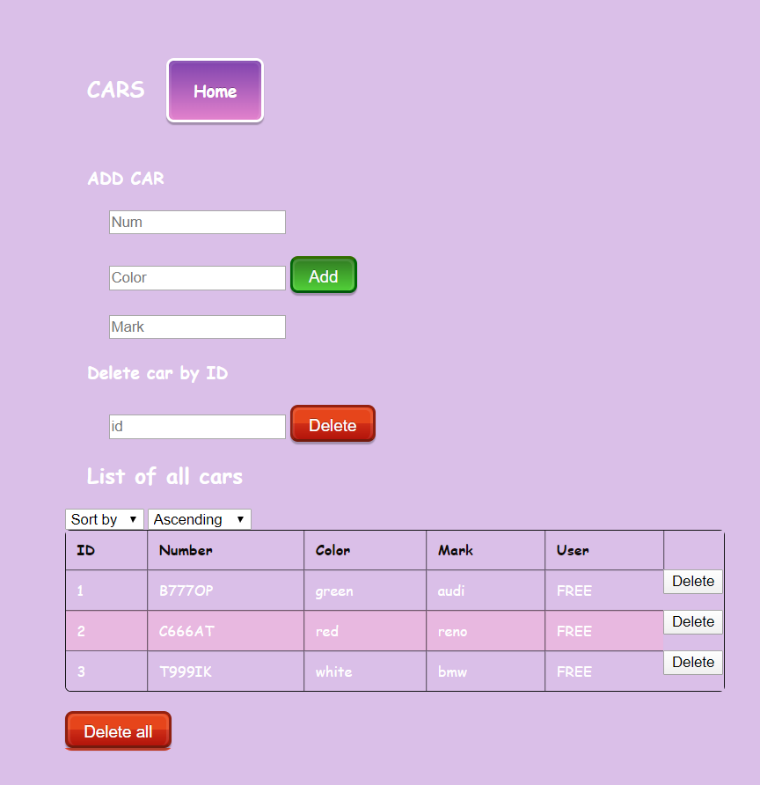


Рисунок 2 Страница "Машины"

Рисунок Начальный экран

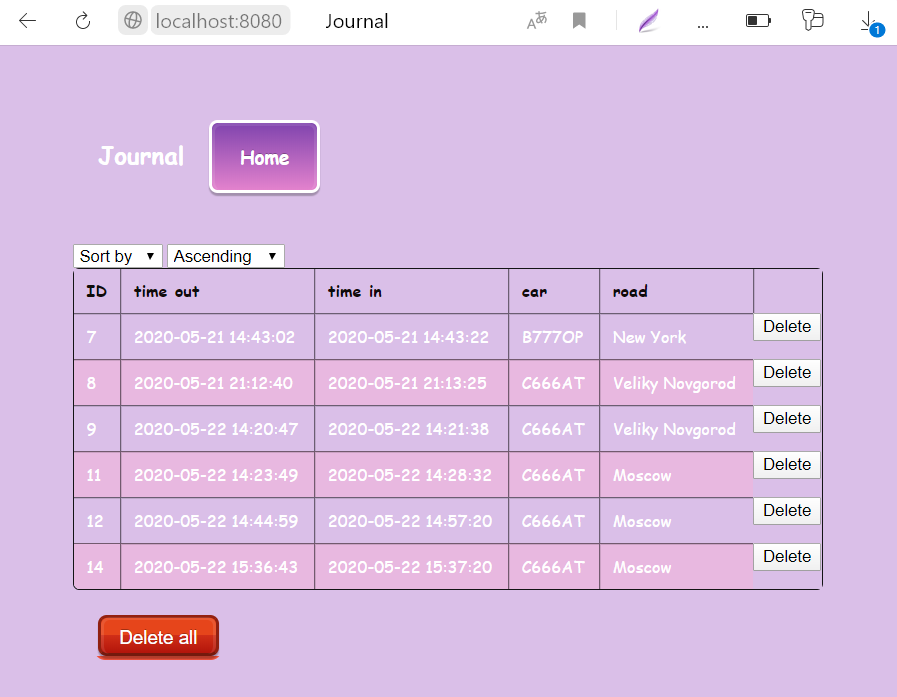
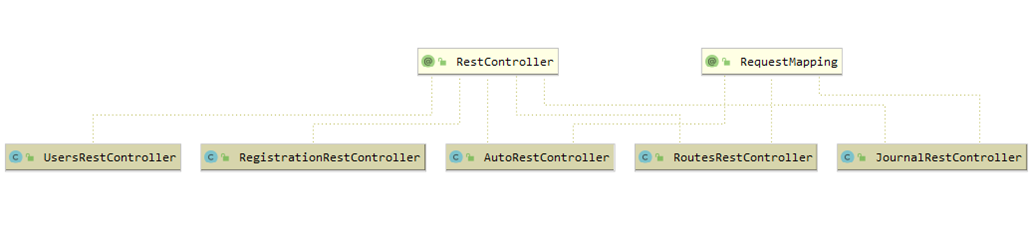


Рисунок 4 Экран обычного пользователя

Рисунок 3 Страница "Журнал"

REST API



|  |  |  |  |
| --- | --- | --- | --- |
| **Запрос** | **Входные данные** | **Выходные данные** | **Комментарий** |
| /users/get/all | - | Список пользователей | - |
| /users/delete/by/id | Json-файл с пользователем | SimpleResponse | Удаляет пользователя |
| /users/get/user | - | Текущий аунтифицированный пользователь | - |
| /cars/add | Json файл с автомобилем | SimpleResponse | Добавляет автомобиль |
| /cars/user | Json файл с автомобилем | SimpleResponse | Присваивает автомобилю текущего пользователя |
| /cars/free | Json файл с автомобилем | SimpleResponse | Освобождает автомобиль от пользователя |
| /cars/delete/by/id | Json файл с автомобилем | SimpleResponse | Удаляет автомобиль по ID |
| /cars/delete/all | - | SimpleResponse | Удаляет все автомобили |
| /cars/get/all | - | Список автомобилей | - |
| /journal/add | Json файл с записью в журнале | SimpleResponse | Добавляет запись |
| /journal/end | Json файл с записью в журнале | SimpleResponse | Присваивает полю time\_in текущее время |
| /journal/delete/all | - | SimpleResponse | Удаляет все записи |
| /journal/delete/by/id | Json файл с записью в журнале | SimpleResponse | Удаляет запись |
| /journal/get/all | - | Список записей в журнале | - |
| /register | Json файл с новым пользователем | SimpleResponse | Добавляет нового пользователя |
| /routes/add | Json файл с маршрутом | SimpleResponse | Добавляет маршрут |
| /routes/delete/by/id | Json файл с маршрутом | SimpleResponse | Удаляет маршрут по id |
| /routes/delete/all | - | SimpleResponse | Удаляет все маршруты |
| /routes/get/all | - | Список маршрутов | - |

Заключение

В результате работы было создано клиент-серверное приложение на языке программирования Java с применением уже имеющихся навыков работы с базами данных SQL. Клиент написан с применением HTML и CSS. Взаимодействие построено на REST API, а также применена технология Spring Security, фреймворк, который сфокусирован на обеспечении как аутентификации, так и авторизации в Java-приложениях.