

Министерство образования и науки Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий
Высшая школа программной инженерии

Курсовой проект

по дисциплине «Объектно-ориентированное программирование»

Выполнил
студент группы 3530904/80001

<подпись>

Трошев Д. М.

Проверил
преподаватель

<подпись>

Д.С. Эйзенах

Содержание

Задание	2
Описание архитектуры	4
Структура классов back-end	5
Схема базы данных	6
Описание REST API	7
Security	15
Назначение классов front-end	16
Интерфейс	17
Заключение	23

Задание

Разработать клиент-серверное приложение на заданную тему (автоматизация работы оптовой фирмы). Список обязательных к использованию технологий при выполнении работы:

Клиент:

требования отсутствуют, можно использовать любые языки

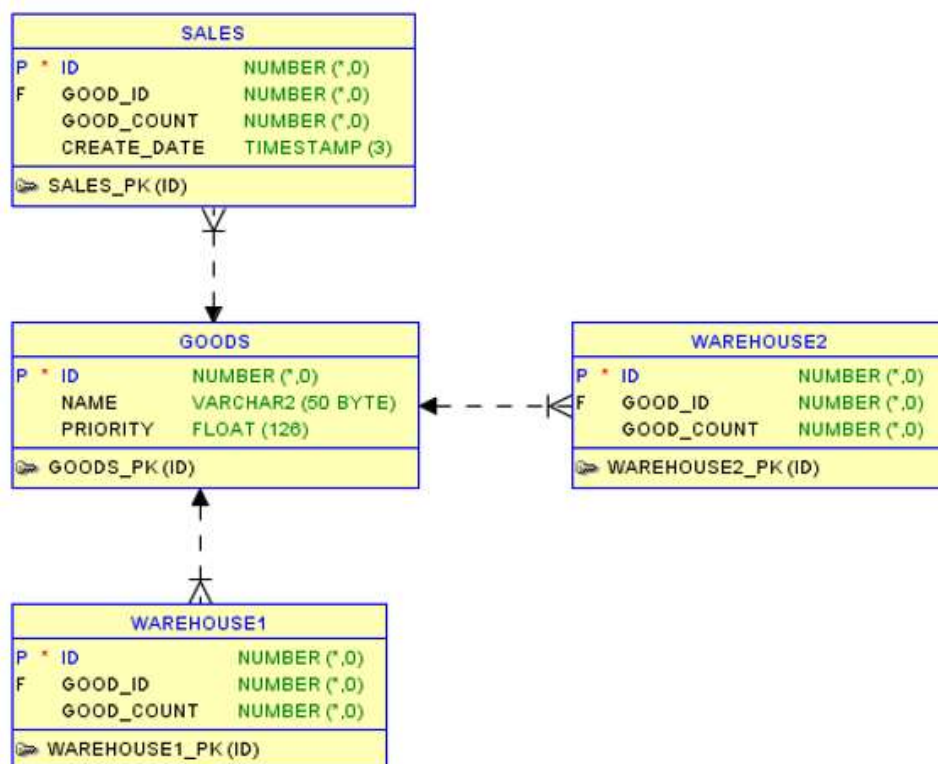
Сервер:

Java 12 и выше

База данных на выбор - SQLite/PostgreSQL/MS SQL/Oracle. Схему БД, соответствующую заданию, см. ниже.

SpringData/Hibernate для работы с БД

Spring Security



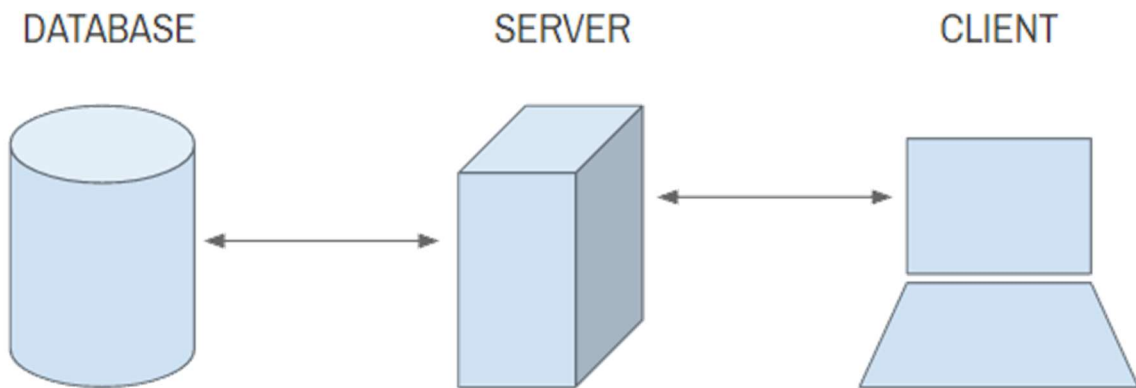
Имя таблицы	Имя колонки	Расшифровка
sales		Таблица заявок
	id	Идентификатор заявки
	good_id	Идентификатор товара
	good_count	Количество товара
	create_date	Дата заявки
goods		Таблица товаров
	id	Идентификатор товара
	name	Наименование товара
	priority	Приоритет товара
warehouse1, warehouse2		Склад1, Склад2
	id	Идентификатор записи
	good_id	Идентификатор товара
	good_count	Количество товара на складе

Взаимодействие с клиентом осуществляется посредством REST API.

Название	Primary Key	Foreign Key
fk sales goods	goods.id	sales.good_id
fk_warehouse1_goods	goods.id	warehouse1.good_id
fk_warehouse2_goods	goods.id	warehouse2.good_id

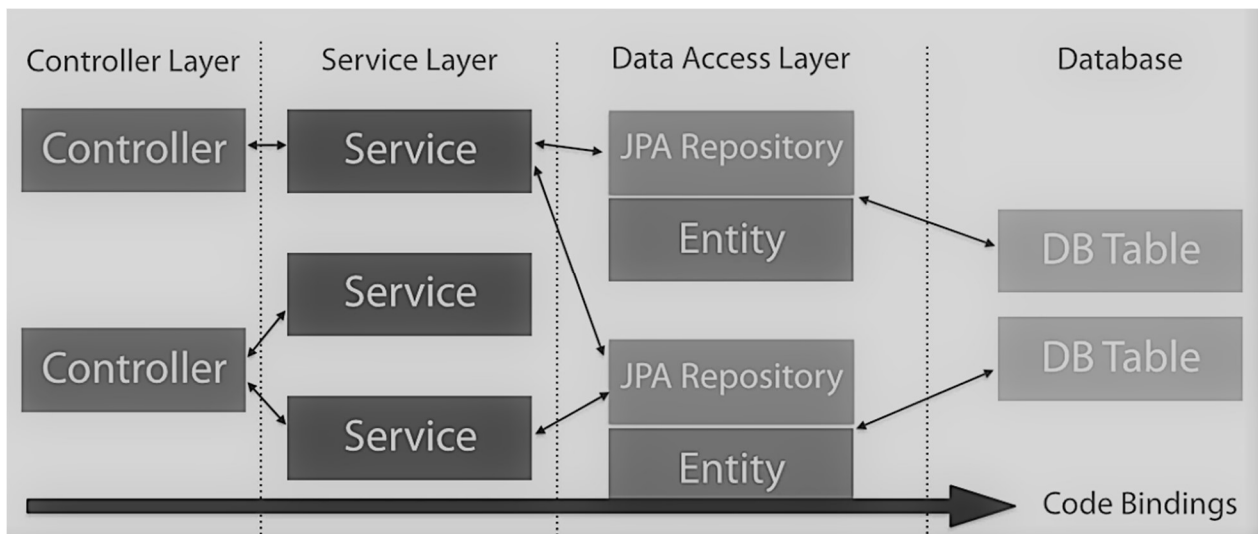
Описание архитектуры

В приложении использована классическая архитектура клиент-сервер-



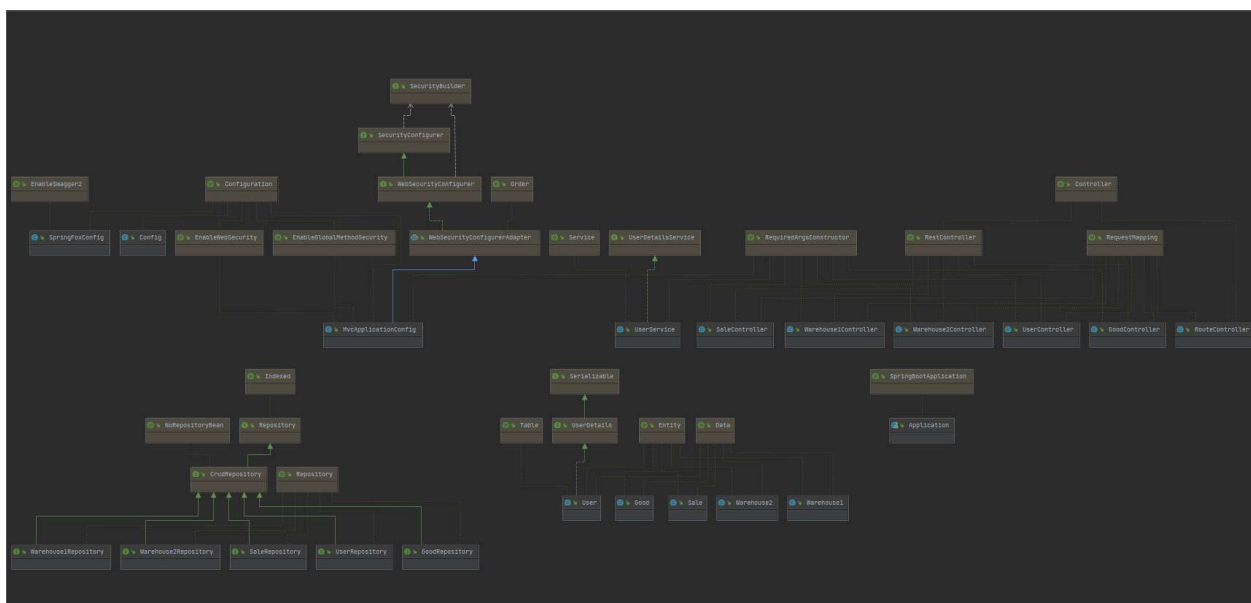
ного приложения.

Слои доступа к базе данных (структура всего back-end) выглядит следующим



образом:

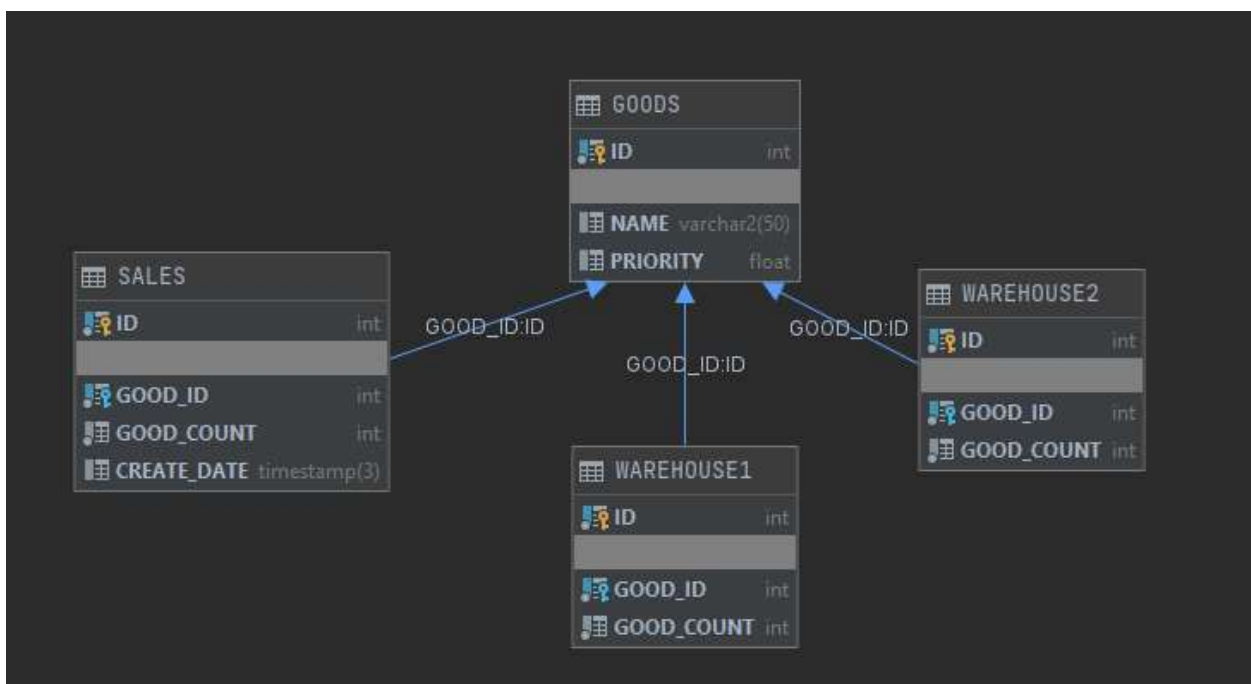
Структура классов back-end



Back-end включает в себя следующие основные пакеты:

- сущности, отражающие таблицы в базе данных
- репозитории для взаимодействия с базой данных
- сервисы для обработки данных, полученных из репозитория и поступающих в них из контроллеров
- контроллеры для взаимодействия с front-end посредством REST-запросов
- структуры «response» для некоторых контроллеров чтобы обеспечить правильную интерпретацию данных, поступивших в запросе
- файлы конфигурации, которые содержат необходимые настройки Spring Security и пути к файлам front-end.

Схема базы данных



Описание REST API

Endpoint	Method	Input	Output
/api/good	Get	None	[{ "id": 0, "name": "string", "priority": 0 }]
/api/good	Post	{ "id": 0, "name": "string", "priority": 0 }	{ "success": true, "error": "" }
/api/good/{id}	Get	None	{ "id": 0, "name": "string", "priority": 0 }
/api/good/{id}	Put	{ "id": 0, "name": "string", "priority": 0 }	{ "success": true, "error": "" }

Endpoint	Method	Input	Output
/api/good/{id}	Delete	None	<pre>{ "success": true, "error": "" }</pre>
/api/sale	Get	None	<pre>[{ "createDate": "2020-05-31T23:23:38.341Z", "good": { "id": 0, "name": "string", "priority": 0 }, "good_count": 0, "id": 0 }]</pre>

Endpoint	Method	Input	Output
/api/sale	Post	<pre>{ "createDate": "2020-05-31T23:23:55.294Z", "good": { "id": 0, "name": "string", "priority": 0 }, "good_count": 0, "id": 0 }</pre>	<pre>{ "success": true, "error": "" }</pre>
/api/sale/{id}	Get	None	<pre>{ "createDate": "2020-05-31T23:24:26.711Z", "good": { "id": 0, "name": "string", "priority": 0 }, "good_count": 0, "id": 0 }</pre>
/api/sale/{id}	Put	<pre>{ "createDate": "2020-05-31T23:24:59.130Z", "good": { "id": 0, "name": "string", "priority": 0 }, "good_count": 0, "id": 0 }</pre>	<pre>{ "success": true, "error": "" }</pre>

Endpoint	Method	Input	Output
/api/sale/{id}	Delete	None	<pre>{ "success": true, "error": "" }</pre>
/user	Get	None	<pre>{ "accountNonExpired": true, "accountNonLocked": true, "authorities": [{ "authority": "string" }], "credentialsNonExpired": true, "enabled": true, "id": 0, "password": "string", "roles": ["string"], "username": "string" }</pre>

Endpoint	Method	Input	Output
/user	Post	<pre>{ "accountNonExpired": true, "accountNonLocked": true, "authorities": [{ "authority": "string" }], "credentialsNonExpired": true, "enabled": true, "id": 0, "password": "string", "roles": ["string"], "username": "string" }</pre>	<pre>{ "success": true, "error": "", }</pre>
/user/all	Get	None	<pre>[{ "accountNonExpired": true, "accountNonLocked": true, "authorities": [{ "authority": "string" }], "credentialsNonExpired": true, "enabled": true, "id": 0, "password": "string", "roles": ["string"], "username": "string" }]</pre>

Endpoint	Method	Input	Output
/api/warehouse1	Get	None	[{ "good": { "id": 0, "name": "string", "priority": 0 }, "good_count": 0, "id": 0 }]
/api/warehouse1	Post	{ "good": { "id": 0, "name": "string", "priority": 0 }, "good_count": 0, "id": 0 }	{ "success": true, "error": "", }
/api/warehouse1/{id}	Get	None	{ "good": { "id": 0, "name": "string", "priority": 0 }, "good_count": 0, "id": 0 }

Endpoint	Method	Input	Output
/api/warehouse1/{id}	Put	{ "good": { "id": 0, "name": "string", "priority": 0 }, "good_count": 0, "id": 0 }	{ "success": true, "error": "" }
/api/warehouse1/{id}	Delete	None	{ "success": true, "error": "" }
/api/warehouse2	Get	None	[{ "good": { "id": 0, "name": "string", "priority": 0 }, "good_count": 0, "id": 0 }]
/api/warehouse2	Post	{ "good": { "id": 0, "name": "string", "priority": 0 }, "good_count": 0, "id": 0 }	{ "success": true, "error": "", }

Endpoint	Method	Input	Output
/api/warehouse2/{id}	Get	None	<pre>{ "good": { "id": 0, "name": "string", "priority": 0 }, "good_count": 0, "id": 0 }</pre>
/api/warehouse2/{id}	Put	<pre>{ "good": { "id": 0, "name": "string", "priority": 0 }, "good_count": 0, "id": 0 }</pre>	<pre>{ "success": true, "error": "" }</pre>
/api/warehouse2/{id}	Delete	None	<pre>{ "success": true, "error": "" }</pre>

Security

Шифрование

Защита осуществляется посредством Basic Authorization. Юзеры с паролями хранятся в зашифрованном виде. Использован `bCryptPasswordEncoder`.

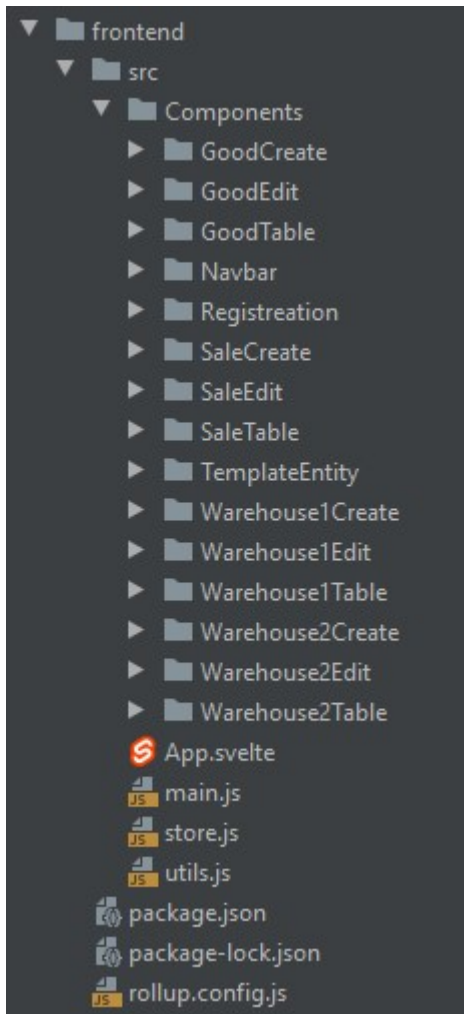
Права

В данном проекте существуют обыкновенные пользователи и администратор. Администратору доступна страница, недоступная для обыкновенных пользователей: список пользователей.

Назначение классов front-end

Структура

Front-end находится в каталоге «frontend».



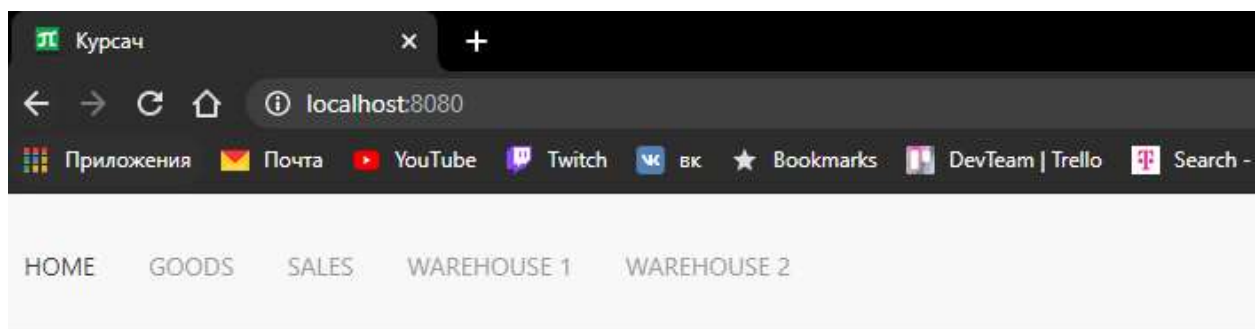
Файл css содержит стили страниц, написанных на языке CSS.

В основном для реализации Front-end был использован Фреймворк Svelte в силу своей простоты.

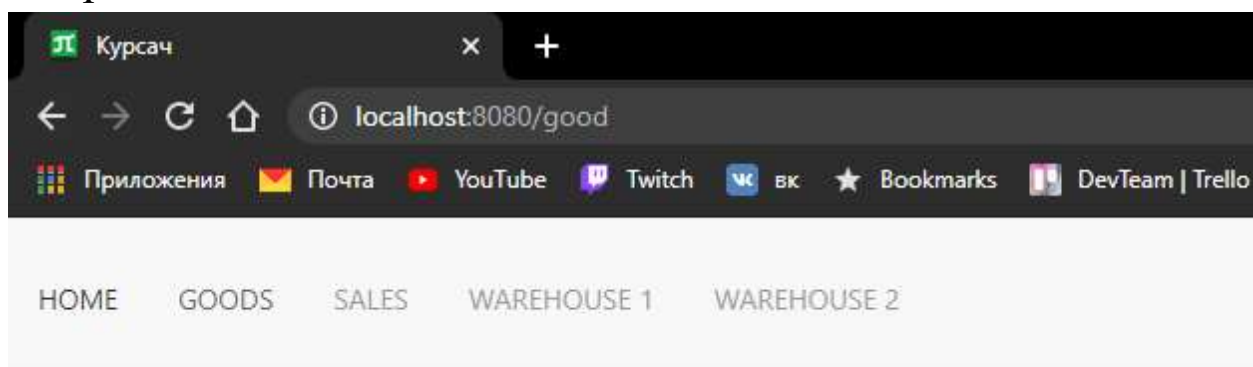
Интерфейс

В данном разделе находятся скриншоты интерфейса, реализованных в папке «frontend».

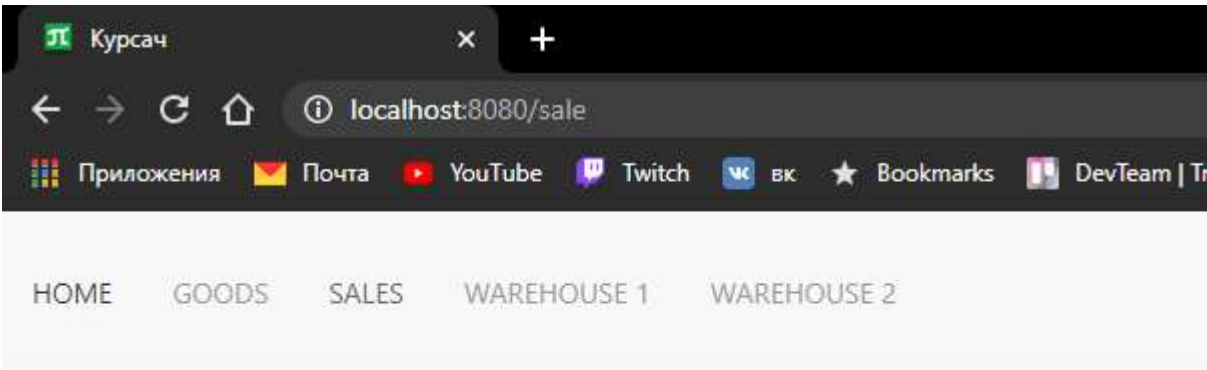
Главная страница «Home»



Открыта вкладка «Goods»



Открыта вкладка «Sales»

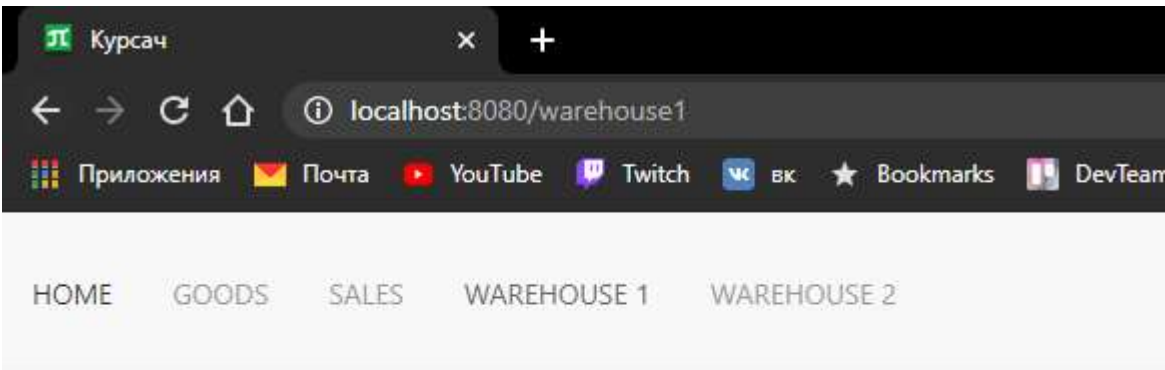


Sale

Create

GOOD GOOD COUNT CREATE DATE

Открыта вкладка «Warhouse1»

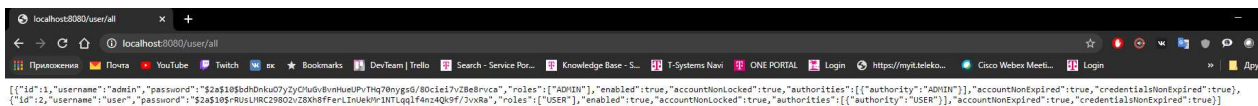


Warehouse1

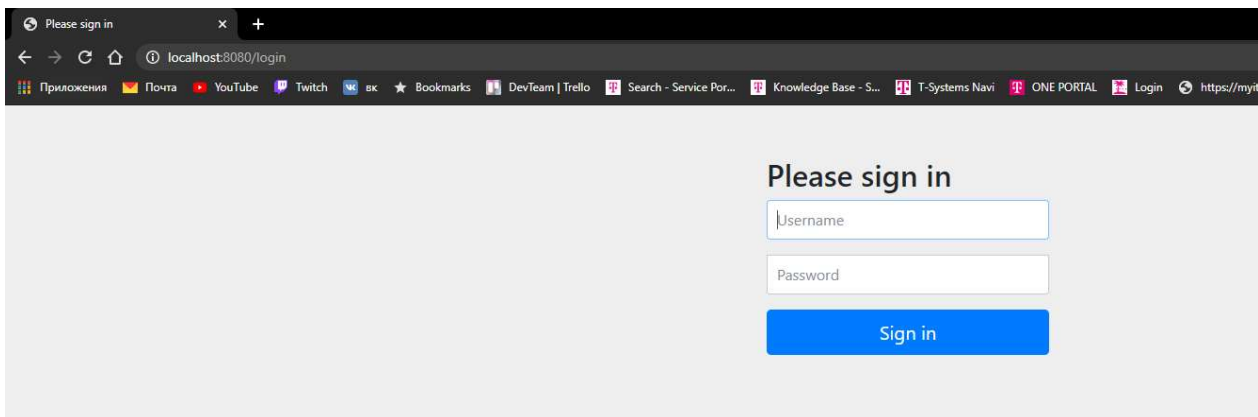
Create

GOOD GOOD COUNT

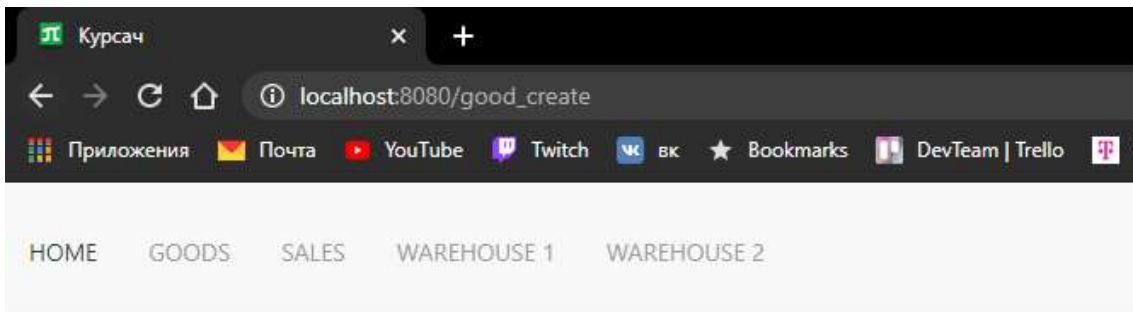
Окно, доступное только для admin(Список пользователей)



Регистрация



Добавление «Goods»



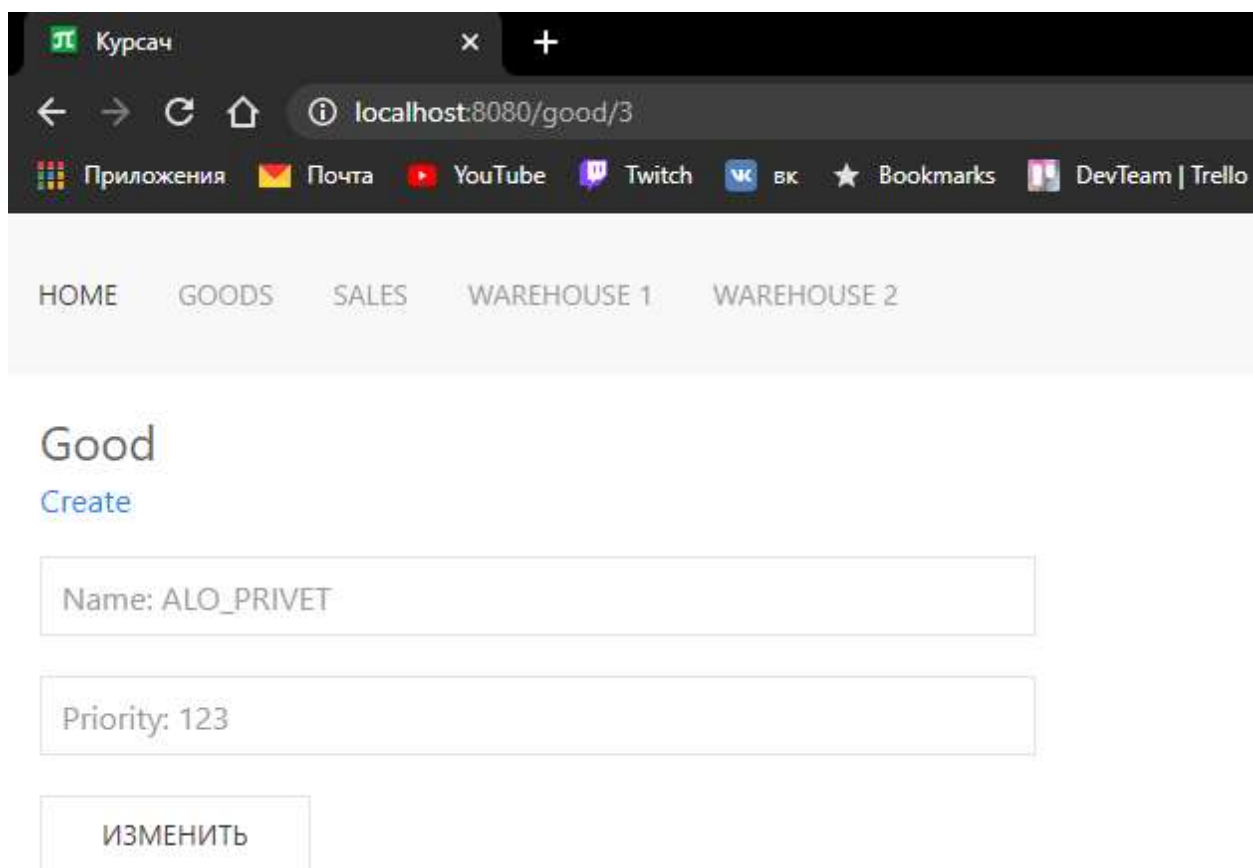
Good

Name

Priority

СОЗДАТЬ

Редактирование «Goods»



Курсач

localhost:8080/good/3

Приложения Почта YouTube Twitch вк Bookmarks DevTeam | Trello

HOME GOODS SALES WAREHOUSE 1 WAREHOUSE 2

Good

[Create](#)

Name: ALO_PRIVET

Priority: 123

ИЗМЕНИТЬ

Добавление «Sales»

Курсач

localhost:8080/sale_create

Приложения Почта YouTube Twitch вк Bookmarks DevTeam | Trello

HOME GOODS SALES WAREHOUSE 1 WAREHOUSE 2

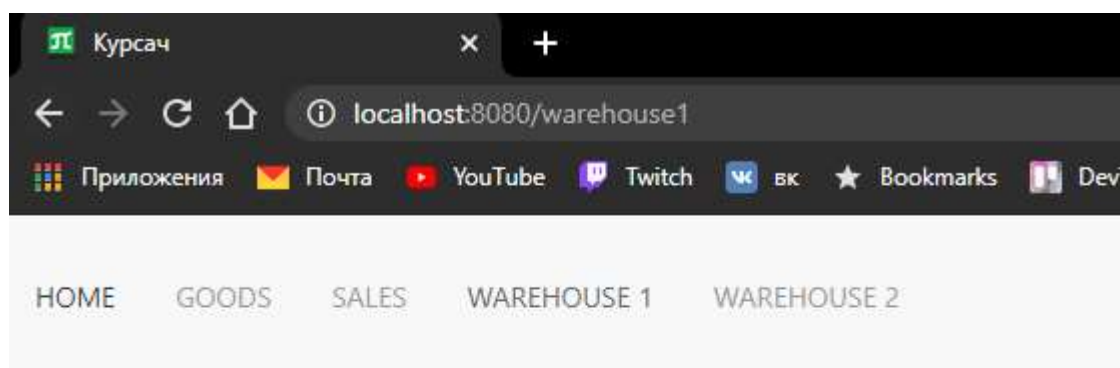
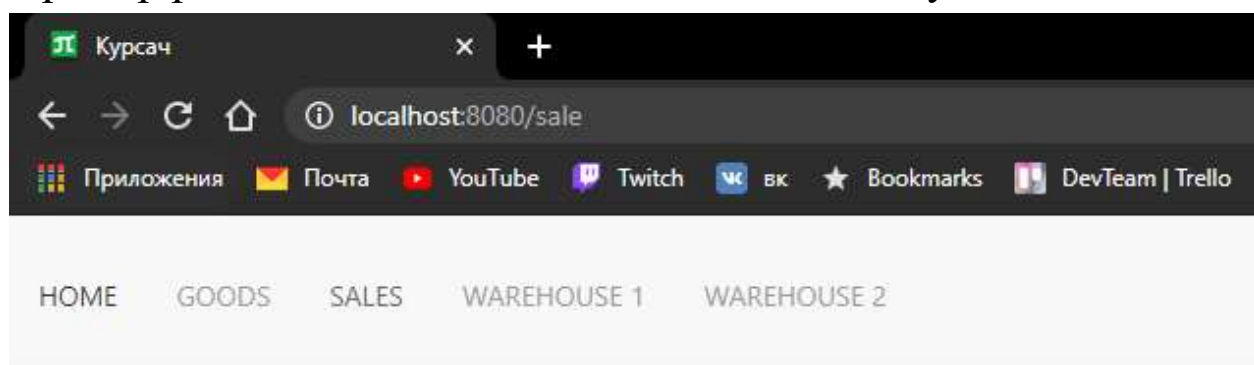
Sale

Good

Good count

Create Date

Пример работы после добавления данных в базу.



Заключение

В ходе работы было выполнено поставленное задание с использованием всех требуемых технологий.

Приложение

Исходный код доступен в репозитории:

<https://github.com/TroshevDmitry/Java2020>