МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий

КАФЕДРА ИНСТРУМЕНТАЛЬНОГО И ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (ИиППО)

Практическая работа №5
«Введение в многослойные клиент-серверные архитектуры»

По дисциплине: «Архитектура клиент-серверных приложений»

Выполнил студент группы ИКБО-10-19 Дараган Ф.А.

Принял преподаватель Степанов П.В.

Практическая работы выполнена «___»_____2021 г. (подпись студента)

«Зачтено» «___»_____2021 г. (подпись руководителя)

Москва 2021

Оглавление

Практическая работа № 5 Введение в многослойные клиент-серверные архитектуры

Цель работы

Ознакомить с многослойными клиент-серверными архитектурами, посмотреть разницу между ними, выявить плюсы и минусы.

Задание

Поскольку для трёхуровневой архитектуры необходимо физическое разделение подсистем, то предлагается разработать трёхслойное приложение: БД, сервер, приложение. В качестве приложения можно использовать: запросы в postman/insomnia/testmace, простой сайт, десктопное приложение, мобильное приложение. В качестве БД можно использовать: SQLite, PostgreSQL.

Выполнение практической работы

На рисунке 1 представлена иерархия файлов проекта.

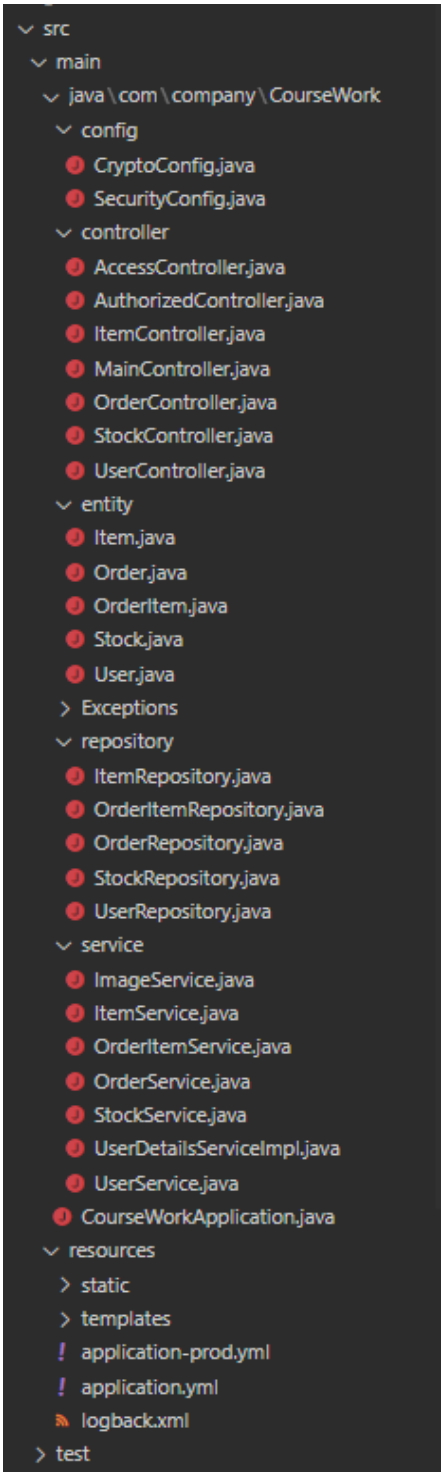

Рис. 1. Скриншот иерархии файлов проекта

Как видно из рисунка 1, проект состоит из контроллеров, сервисов, репозиториев, которые как раз и реализуют основную часть трехуровневой архитектуры.

В качестве примера рассмотрим контроллер, сервис и репозиторий для «цепочки» Item. На листинге 1 представлен класс контроллера.

Листинг 1. Класс ItemController

```
@Controller
@RequestMapping(path = "/item")
public class ItemController extends AuthorizedController{

    @Autowired
    ItemService itemService;

    @Autowired
    StockService stockService;

    @GetMapping
    public String items(
            Model model
    ) {
        User user = this.getCurrentUser();

        model.addAttribute("items", itemService.getAll());
        model.addAttribute("user", user);

        return "item/index";
    }

    @GetMapping(path = "/get")
    public String getItem(
            @RequestParam long id,
            HttpServletRequest request,
            Model model
    ) {
        User user = this.getCurrentUser();

        Item item = itemService.getItemById(id);

        if (item == null)
            return getPreviousPageByRequest(request)
                    .orElse("redirect:/");

        model.addAttribute("item", item);
        model.addAttribute("user", user);
```

```
        if (user != null && user.getRole().equals("ADMIN"))
            model.addAttribute("stocks", stockService.getAll());

        return "item/item";
    }

    @PostMapping(path = "create")
    public String create(
            @RequestParam String name,
            @RequestParam double price,
            @RequestParam String description,
            @RequestParam("image") MultipartFile image,
            Model model
    ) {
        itemService.createItem(name, price, description, image);

        return "redirect:/item";
    }

    @PostMapping(path = "delete")
    public String delete(
            @RequestParam long id,
            Model model
    ) {
        itemService.removeItem(id);

        return "redirect:/item";
    }

}
```

Как можно увидеть, контроллер не осуществляет никакой бизнес логики, он вызывает сервис для этих целей. На листинге 2 показан сервис для Item, который как раз и осуществляет собой выполнение бизнес-правил.

Листинг 2. Класс ItemService

```
@Service
public class ItemService {

    @Autowired
    ItemRepository itemRepository;

    @Autowired
    ImageService imageService;

    @Transactional(readOnly = true)
    public List<Item> getAll() {
        return itemRepository.findAll();
    }
```

```
    @Transactional(readOnly = true)
    public Item getItemById(long id) {
        Optional<Item> item = itemRepository.findById(id);
        return item.orElse(null);
    }

    @Transactional
    public void createItem(String name, double price, String
description, MultipartFile file) {
        Item item = Item.builder()
                .name(name)
                .price(price)
                .description(description)
                .pathToImage(null)
                .build();

        itemRepository.save(item);

        String path = imageService.saveImage(file, item);
        item.setPathToImage(path);
    }

    @Transactional
    public void removeItem(long id) {
        Item item = getItemById(id);

        if (item != null) {
            imageService.deleteImage(item);
            itemRepository.deleteById(id);
        }
    }

}
```

Из листинга 2 видно, что вся работа с базой данных ложится на репозиторий для Item. Репозиторий содержит множество встроенных функций, позволяющих довольно просто работать с базой данных, большинство из этих функций генерируются автоматически самим Spring. Код репозитория представлен на листинге 3.

Листинг 3. Класс ItemRepository

```
@Repository
@Transactional
public interface ItemRepository extends JpaRepository<Item, Long>,
JpaSpecificationExecutor<Item> {
}
```

Рассматривать остальные части системы не имеет смысла, ибо они идентичны, остальной код проекта можно увидеть в репозитории: https://github.com/laefad/project.

На рисунке 2 можно увидеть интерфейс разработанного приложения.

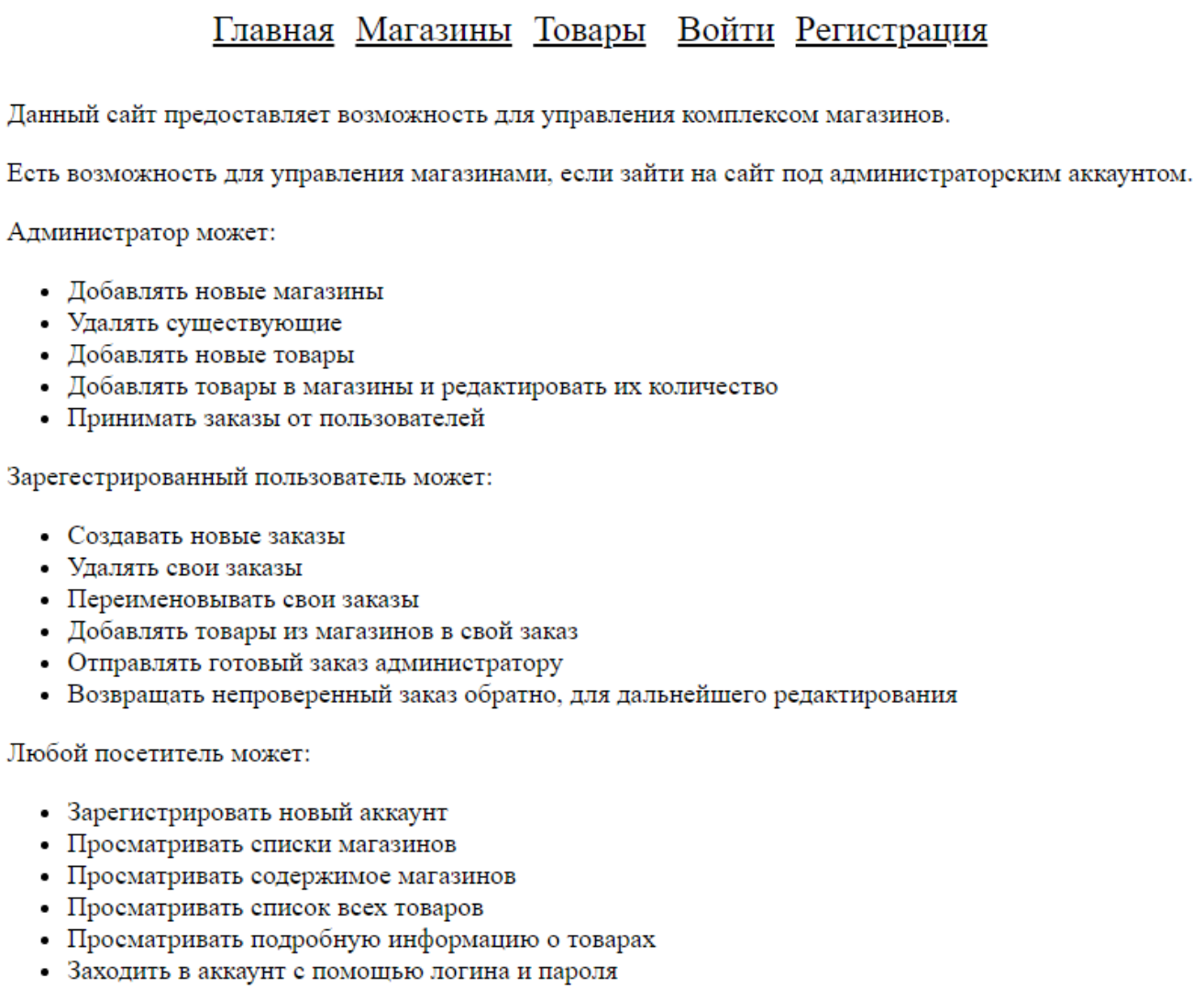

Рис. 2. Главная страница веб-приложения

Выводы по работе

В результате выполнения работы были получены навыки по разработке приложений с многослойной архитектурой.

Используемая литература

1. Вязовик, Н. А. Программирование на Java : учебное пособие / Н. А. Вязовик. — 2-е изд. — Москва : ИНТУИТ, 2016. — 603 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/100405 (дата обращения: 13.09.2021). — Режим доступа: для авториз. пользователей.
2. Наир, В. Предметно-ориентированное проектирование в Enterprise Java : руководство / В. Наир ; перевод с английского А. В. Снастина. — Москва : ДМК Пресс, 2020. — 306 с. — ISBN 978-5-97060-872-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/179503 (дата обращения: 13.09.2021). — Режим доступа: для авториз. пользователей.
3. Васильев, А. Н. Самоучитель Java с примерами и программами : учебное пособие / А. Н. Васильев. — 4-е, изд. — Санкт-Петербург : Наука и Техника, 2017. — 368 с. — ISBN 978-5-94387-745-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/101548 (дата обращения: 13.09.2021). — Режим доступа: для авториз. пользователей.