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

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

**ПРАКТИЧЕСКАЯ РАБОТА №3**

по дисциплине «Технологии виртуализации клиент-серверных приложений»

**Студент группы** ИКБО-20-19 Анваржонов Ж. Т

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**Руководитель практической работы** старший преподаватель Волков М.Ю.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Работа представлена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Допущен к работе «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Москва 2022

Запускать веб-приложение будем на виртуальной машине, поэтому первым делос устанавливаем docker на ВМ машину

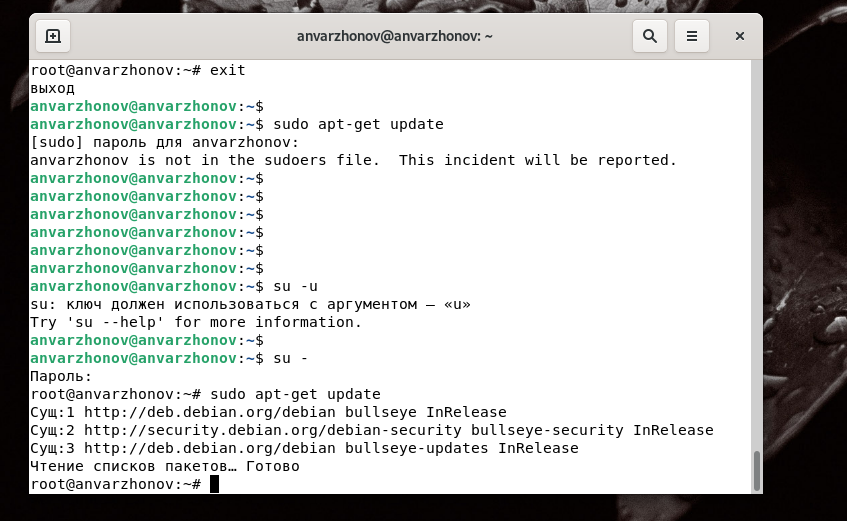


Рисунок 1 – установка docker на ВМ

Добавили grg docker ключ

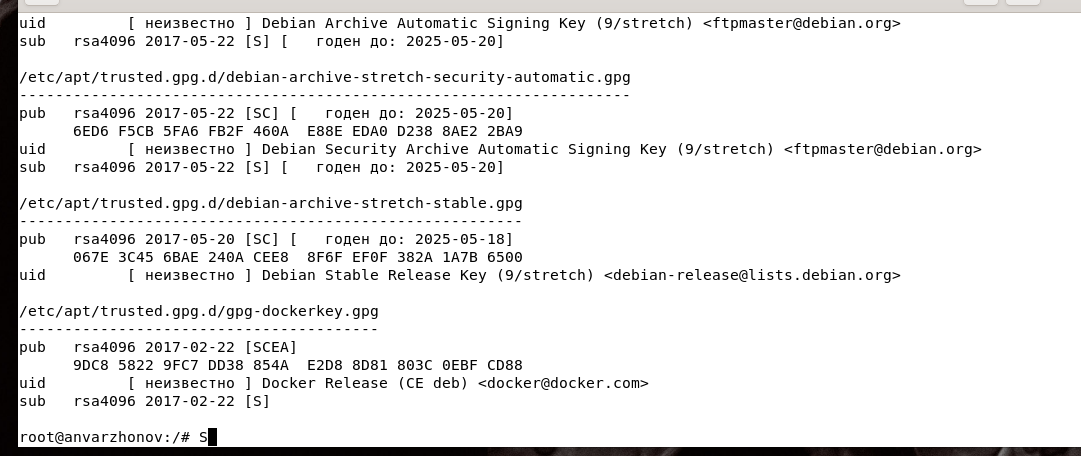


Рисунок 2 – Добавление grg-docker ключа

Установка docker и docker compose

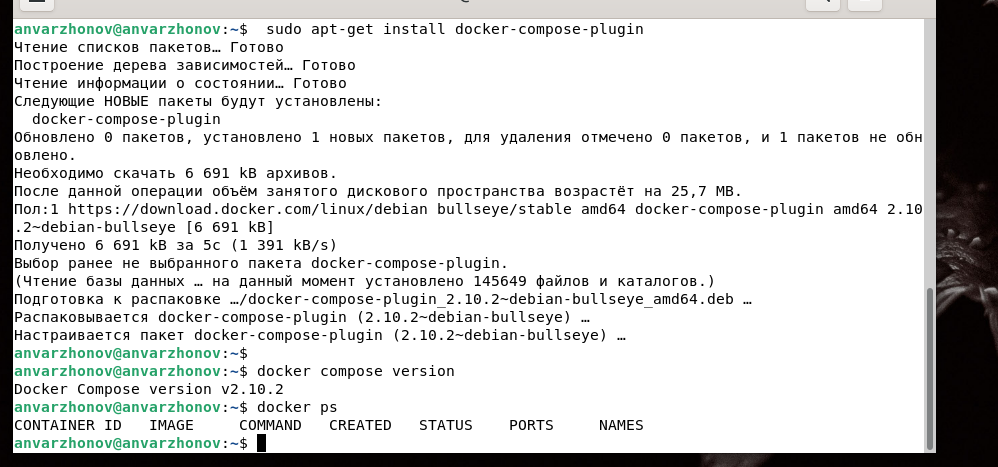


Рисунок 3 – установка docker и docker-compose на ВМ

Зайдем по ssh на ВМ



Рисунок 4 – попытка зайти на ВМ по ssh

Чтобы постоянно не вводить пароль при попытке зайти на ВМ создадим ssh-ключ и перенесем в домашнюю директорию ВМ

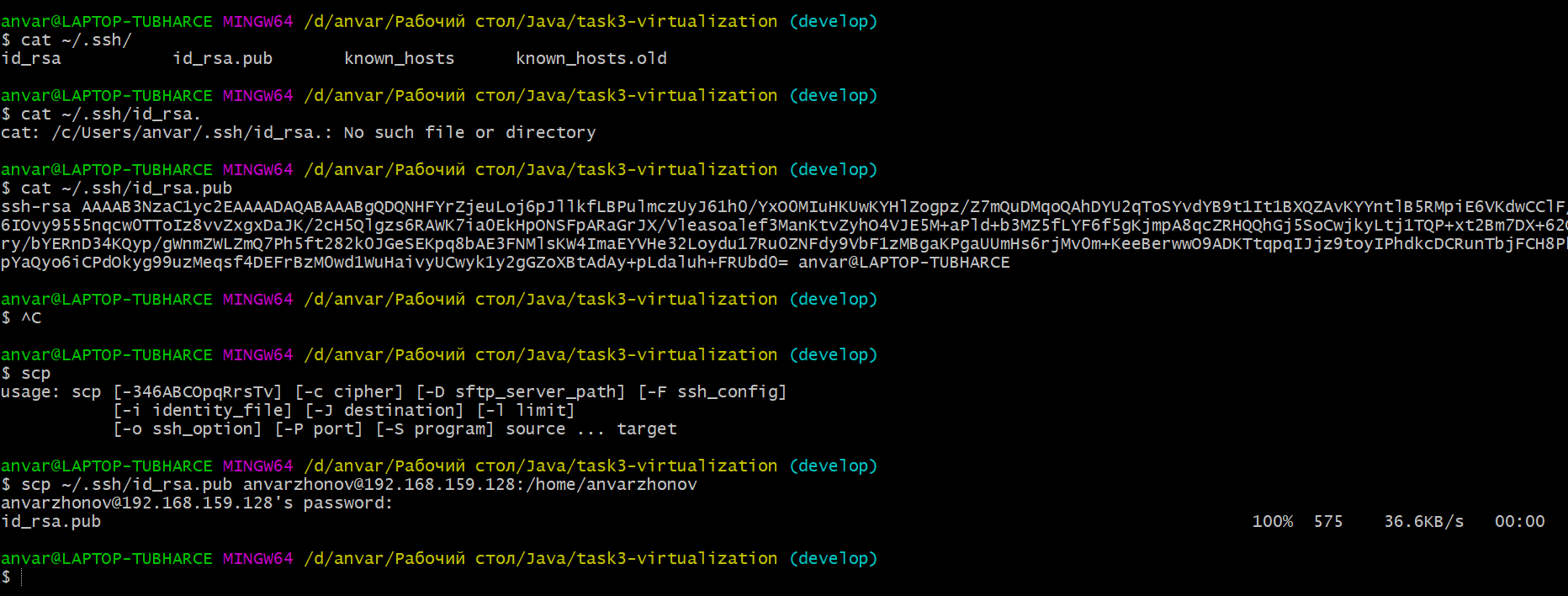


Рисунок 5– прописываем ssh ключ на ВМ

Перемещаем ключ в папку .ssh

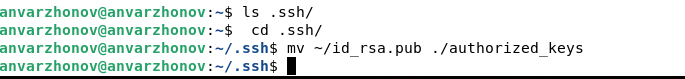
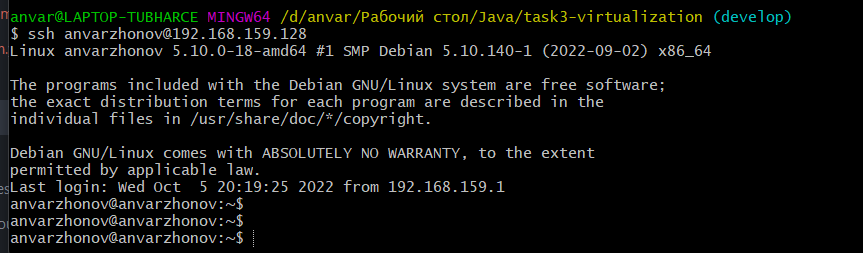


Рисунок 6 – перемещение ssh ключ

Проверяем вход на ВМ без пароля:



Перенесем структуру папок приложения Spring Boot на ВМ с помощью утилиты scp

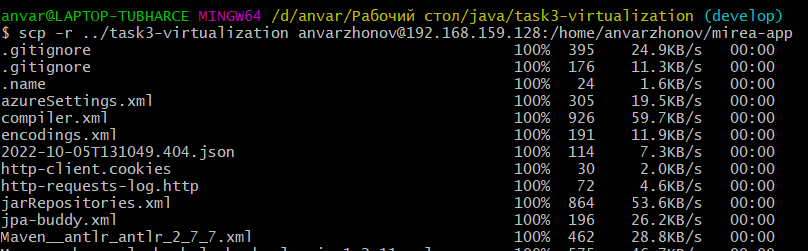


Рисунок 8 – перенос приложения, написанного на windows в debian 11

Посмотрим результат:



Рисунок 9 – перенесенные файлы spring boot приложения

Покажем Dockerfile с двумя этапами – сборка проекта и запуск



Рисунок 10 – Dockerfile с двумя этапами – build and run

Покажем код контроллера:

*/\*\*  
 \* todo Document type MainController  
 \*  
 \** ***@author*** *- Anvarzhonov Z.T. IKBO-20-19 on 04.10.2022 - 20:40  
 \*/*@Slf4j  
@Controller  
*public class* MainController {  
 *private final UserRepository* repository;  
  
 *public* MainController(*UserRepository* repository) {  
 *this*.repository = repository;  
 }  
  
 @GetMapping("/createUser")  
 *public* String newUser(*Model* model) {  
 model.addAttribute("user", *new* User());  
 *return* "user-form";  
 }  
  
 @PostMapping("/createUser")  
 *public* String createUser(@ModelAttribute User user, *Model* model) {  
 User savingUser = repository.save(user);  
 log.info("Saving new user: {}", savingUser);  
  
 *return* "user-form";  
 }  
  
 @GetMapping("/showAll")  
 *public* String showAll(*Model* model) {  
 model.addAttribute("users", repository.findAll());  
  
 *return* "users";  
 }  
  
 @GetMapping("/showGerb")  
 *public* String getGerbMirea() {  
 *return* "mirea-image";  
 }  
}

Рисунок 10 – главный контроллер

На рисунке 11 показан application.yaml, где отображаются настройки для подключения к БД postgres. Порт СУБД извлекается из переменной окружения, указанной в Dockerfile.

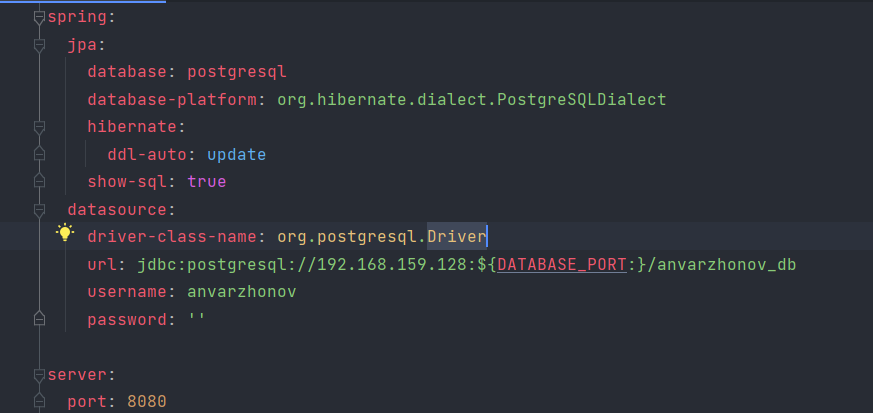


Рисунок 11 – application.yml

На рисунках 12-14 показаны шаблоны для отображения герба, форма для добавления нового пользователя и страница для показа всех пользователей

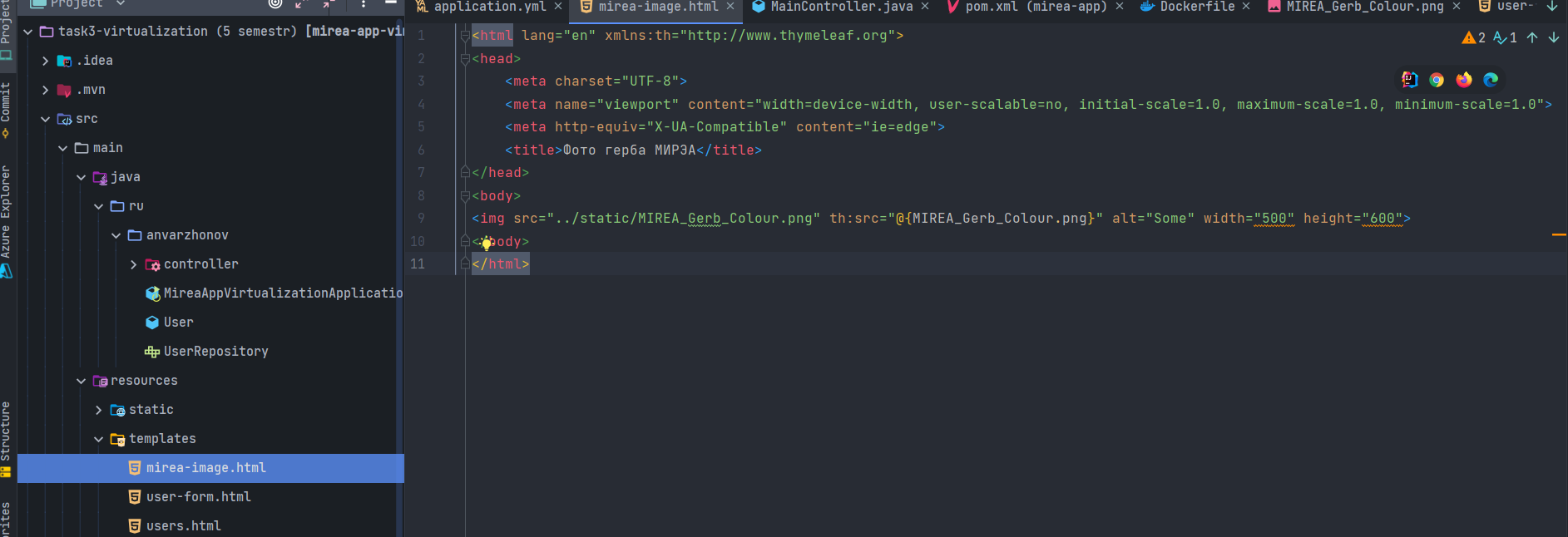


Рисунок 12 – шаблон для отображения герба

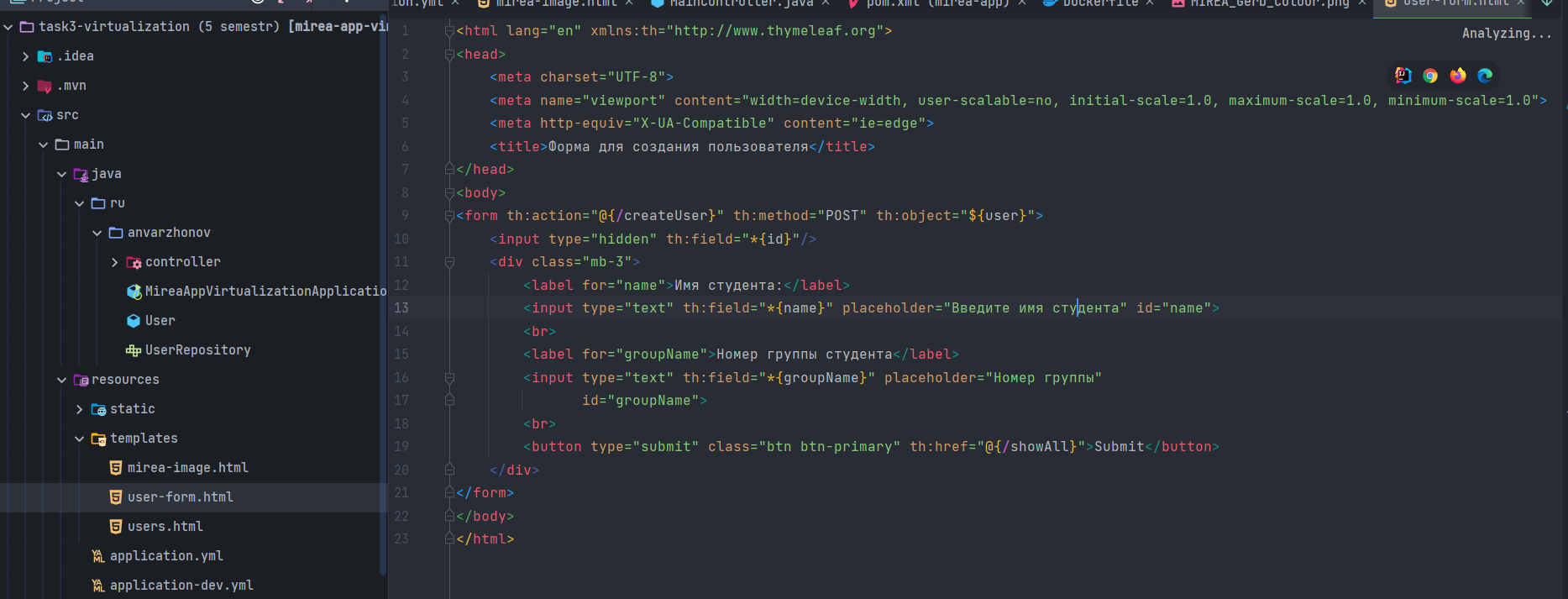


Рисунок 13 – Шаблон с формой для добавления нового пользователя

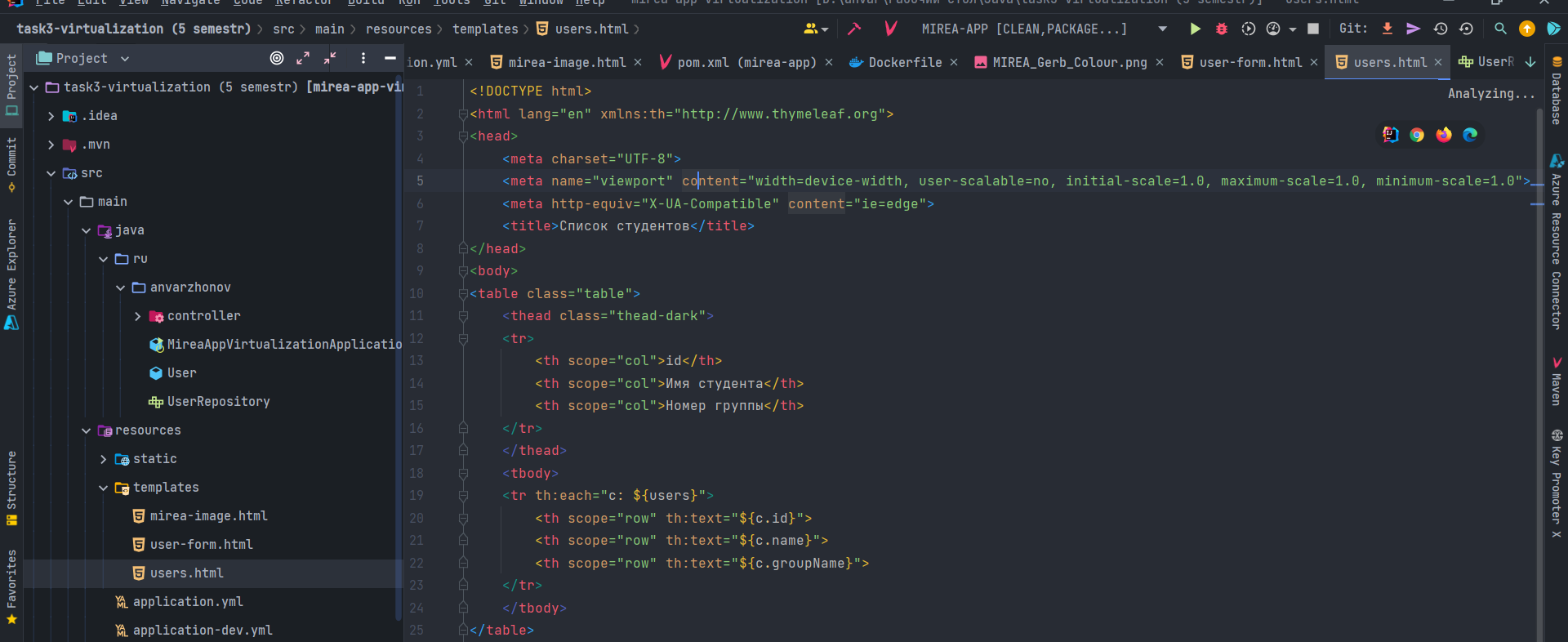


Рисунок 14 – Шаблон для отображения всех пользователей

Последним этапом загрузим образ на Docker Hub.

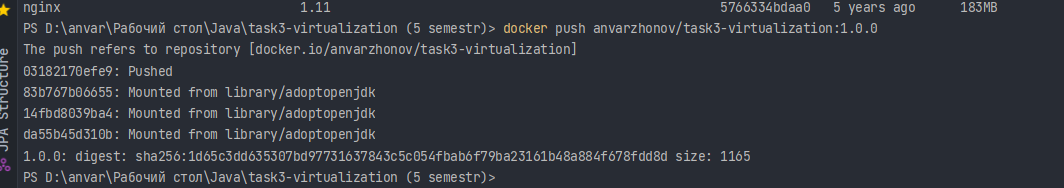


Рисунок 15 – Процесс загрузки образа в docker hub

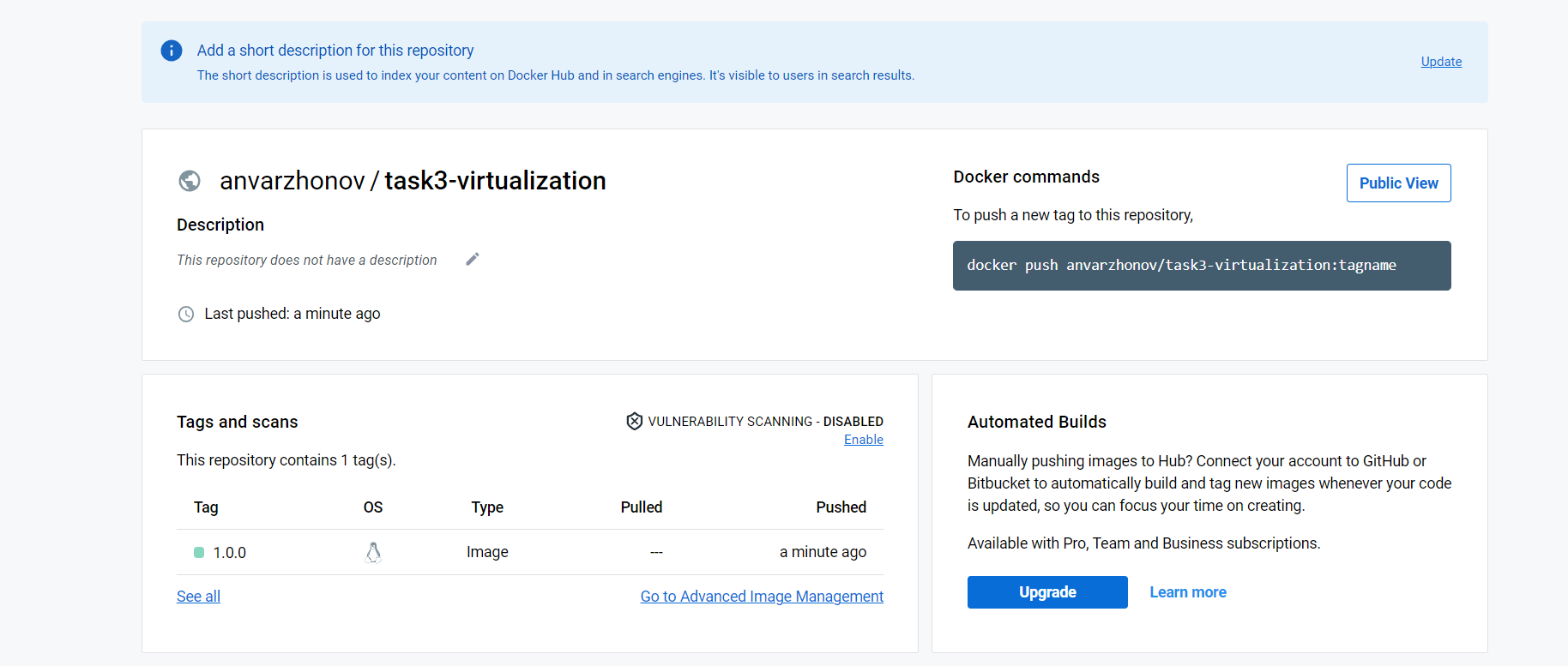


Рисунок 16 – Отображение в репозитории загруженного образа

# Вывод

В результате выполнения третьей практической работы были получены навыки работы с Docker-контейнерами. Также были изучены принципы создания Dockerfile и docker-compose.yml и запуска контейнеров. Произведена работа по контейнеризации приложения spring boot.

# Ответы на вопросы к практической работе

1. Опишите процесс запуска приложения внутри контейнера Linux, используя Docker.

Для запуска контейнера используется команда docker run с именем контейнера. Запуск приложений внутри контейнера возможен через консоль работающего контейнера или с помощью выполнения команды CMD в Dockerfile при сборке контейнера.

2. Что такое образ Docker и для чего он нужен?

Docker — это программная платформа для разработки, доставки и запуска контейнерных приложений. Он позволяет создавать контейнеры, автоматизировать их запуск и развертывание, управляет жизненным циклом. С помощью Docker можно запускать множество контейнеров на одной хост-машине.

3. Как соотносятся между собой файлы Dockerfile и Docker-Compose?

Dockerfile - это обычный текстовый файл, содержащий инструкции по созданию образов Docker. Существует стандарт Dockerfile, которому они следуют, и демон Docker в конечном итоге отвечает за выполнение Dockerfile и генерацию образа.

Docker Compose - это инструмент для определения и запуска многоконтейнерных приложений Docker. Используя файл конфигурации YAML, Docker Compose позволяет нам настраивать несколько контейнеров в одном месте. Затем мы можем запускать и останавливать все эти контейнеры одновременно, используя одну команду.

4. Что такое Dockerfile?

Dockerfile содержит инструкции для сборки образов, которые передаются в Docker. Также его можно описать как текстовый документ, содержащий все возможные команды, с помощью которых пользователь, последовательно их запуская, может собрать образ.

5. Опишите политики перезапуска контейнера.

Существуют следующие политики перезапуска контейнеров Docker:

* no: по умолчанию контейнеры не запускаются автоматически.
* always: всегда перезапускать остановленный контейнер, если контейнер не был остановлен явно
* unless-stopped: перезапустить контейнер, если контейнер не был в остановленном состоянии до остановки демона Docker (объяснено позже).
* on-failure: перезапустите контейнер, если он завершился с ненулевым кодом выхода или если демон докера перезапустился.

6. Назовите все возможные состояния контейнеров.

Контейнер — базовая единица программного обеспечения, покрывающая код и все его зависимости для обеспечения запуска приложения прозрачно, быстро и надежно независимо от окружения. Контейнер Docker может быть создан с использованием образа Docker. Это исполняемый пакет программного обеспечения, содержащий все необходимое для запуска приложения, например, системные программы, библиотеки, код, среды исполнения и настройки.

Все возможные состояния контейнера Docker:

● Created — контейнер создан, но не активен.

● Restarting — контейнер в процессе перезапуска.

● Running — контейнер работает.

● Paused — контейнер приостановлен.

● Exited — контейнер закончил свою работу.

● Dead — контейнер, который сервис попытался остановить, но не смог.