КАФЕДРА №

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №8

Разработка микросервиса

по дисциплине: Технология разработки серверных информационных систем

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР.		подпись, дата	инициалы, фамилия

Санкт-Петербург
2023

Текст и вариант задания:

13. Торговля акциями на бирже.

Описание разрабатываемого продукта:

В программе создан сервис для покупки, продажи и просмотра портфеля акций.

Задание на лабораторную работу.

1. Подготовьте Ваше приложение к разворачиванию в облачном сервисе или компоненте Docker.

2. Реализуйте файл docker-compose.xml, который будет содержать все необходимые для работы Вашего приложения ресурсы

3. Внимание. В виду того, что далеко не на каждом компьютере можно запустить систему виртуализации, данная лабораторная работа сдается в электронном виде, без демонстрации преподавателю.

Текст основных фрагментов кода:

Itemobj.java

```
package com.example.Project;


import jakarta.persistence.Entity;

import jakarta.persistence.Id;


@Entity

public class ItemObj {

    @Id

    private int stockID;

    private String stock_name;

    private String purchase_date;


    public ItemObj() {}


    public ItemObj(int stockID, String stock_name, String purchase_date) {

        this.stockID = stockID;
    this.stock_name = stock_name;
    this.purchase_date = purchase_date;
}

public int getStockID() {
    return stockID;
}

public void setStockID(int stockID) {
    this.stockID = stockID;
}

public String getStock_name() {
    return stock_name;
}

public void setStock_name(String stock_name) {
    this.stock_name = stock_name;
}

public String getPurchase_date() {
    return purchase_date;
}

public void setPurchase_date(String purchase_date) {
    this.purchase_date = purchase_date;
}

@Override
```

```
    public String toString() {
        return "ItemObj{" +
                "stockID=" + stockID +
                ", stock_name='" + stock_name + '\'' +
                ", purchase_date='" + purchase_date + '\'' +
                '}';
    }
}
```

Itemobjrepository

```
package com.example.Project;

import org.springframework.data.jpa.repository.JpaRepository;

public interface ItemObjRepository extends JpaRepository<ItemObj, Integer> {
}
```

Logincontroller

```
package com.example.Project;

import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;

@Controller
public class LoginController {
    @GetMapping("/login")
    public String login() {
        return "login";
    }

    @PostMapping("/logout")
    public String logout(HttpServletRequest request) throws ServletException {
        request.logout();
        return "redirect:/login";
    }
}
```

MyKafkaConsumer

```
package com.example.Project;

import com.fasterxml.jackson.databind.ObjectMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Profile;
import org.springframework.kafka.annotation.KafkaListener;
import org.springframework.stereotype.Service;

@Service
@Profile("consumer")
public class MyKafkaConsumer {
    private final ItemObjRepository repository;
    private final ObjectMapper objectMapper = new ObjectMapper();

    @Autowired
    public MyKafkaConsumer(ItemObjRepository repository) {
        this.repository = repository;
    }

    @KafkaListener(topics = "myTopic")
    public void listen(String message) {
        System.out.println(message);
        // Преобразование сообщения обратно в ItemObj и сохранение его в базе данных
        ItemObj itemObj = convertMessageToItemObj(message);
        repository.save(itemObj);
    }

    private ItemObj convertMessageToItemObj(String message) {
        try {
            return objectMapper.readValue(message, ItemObj.class);
        } catch (Exception e) {
            throw new RuntimeException("Ошибка при преобразовании сообщения", e);
        }
    }
}
```

myKafkaProducer

```
package com.example.Project;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.stereotype.Service;
```

```
@Service
public class MyKafkaProducer {
    private final KafkaTemplate<String, String> kafkaTemplate;

    @Autowired
    public MyKafkaProducer(KafkaTemplate<String, String> kafkaTemplate) {
        this.kafkaTemplate = kafkaTemplate;
    }

    public void sendMessage(String topic, String key, String message) {
        kafkaTemplate.send(topic, key, message);
    }
}
```

projectApplication

```
package com.example.Project;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.MessageSource;
import org.springframework.context.annotation.Bean;
import org.springframework.context.support.ReloadableResourceBundleMessageSource;
import org.springframework.web.servlet.LocaleResolver;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.i18n.LocaleChangeInterceptor;
import org.springframework.web.servlet.i18n.SessionLocaleResolver;
import java.util.Locale;
import org.apache.catalina.Context;
import org.apache.tomcat.util.descriptor.web.FilterDef;
import org.apache.tomcat.util.descriptor.web.FilterMap;
import org.springframework.boot.web.embedded.tomcat.TomcatServletWebServerFactory;
import org.springframework.boot.web.servlet.server.ServletWebServerFactory;
import org.springframework.web.filter.CharacterEncodingFilter;


@SpringBootApplication
public class ProjectApplication implements WebMvcConfigurer{

    public static void main(String[] args) {
        SpringApplication.run(ProjectApplication.class, args);
    }



    // Бин для MessageSource
    @Bean
    public MessageSource messageSource() {
        ReloadableResourceBundleMessageSource messageSource = new
ReloadableResourceBundleMessageSource();
        messageSource.setBasename("classpath:messages");
        messageSource.setDefaultEncoding("UTF-8");
        return messageSource;
    }

    // Бин для LocaleResolver
    @Bean
    public LocaleResolver localeResolver() {
        SessionLocaleResolver slr = new SessionLocaleResolver();
        slr.setDefaultLocale(Locale.ENGLISH); // Английский язык по умолчанию
        return slr;
    }

    // Бин для LocaleChangeInterceptor
    @Bean
    public LocaleChangeInterceptor localeChangeInterceptor() {
        LocaleChangeInterceptor lci = new LocaleChangeInterceptor();
        lci.setParamName("lang");
        return lci;
    }

    // Добавление интерцептора для перехвата изменений локали
    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(localeChangeInterceptor());
    }

    // Настройка кодировки UTF-8 для Tomcat
    @Bean
    public ServletWebServerFactory servletContainer() {
        TomcatServletWebServerFactory tomcat = new
TomcatServletWebServerFactory() {
            @Override
            protected void postProcessContext(Context context) {
                FilterDef filterDef = new FilterDef();
                filterDef.setFilterName("setCharacterEncodingFilter");

    filterDef.setFilterClass(CharacterEncodingFilter.class.getName());
                filterDef.addInitParameter("encoding", "UTF-8");
                filterDef.addInitParameter("forceEncoding", "true");
                context.addFilterDef(filterDef);


                FilterMap filterMap = new FilterMap();
                filterMap.setFilterName("setCharacterEncodingFilter");
                filterMap.addURLPattern("/*");
                context.addFilterMap(filterMap);
            }
        };
        return tomcat;
    }
}
```

Stockcontroller

```
package com.example.Project;


import com.fasterxml.jackson.databind.ObjectMapper;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

@Controller
@RequestMapping("/stocks")
public class StockController {

    private final ItemObjRepository itemObjRepository;
    private final MyKafkaProducer myKafkaProducer;
    private final ObjectMapper objectMapper = new ObjectMapper();



    @Autowired
    public StockController(ItemObjRepository itemObjRepository, MyKafkaProducer
myKafkaProducer) {
        this.itemObjRepository = itemObjRepository;
        this.myKafkaProducer = myKafkaProducer;

    }

    @GetMapping
    public String getAllStocks(Model model) {
        model.addAttribute("stocks", itemObjRepository.findAll());
        return "stocks";
    }

    @PostMapping
    public String addStock(@ModelAttribute ItemObj stock) {
        itemObjRepository.save(stock);
        if (myKafkaProducer != null) {
            String message = convertStockToMessage(stock);
            myKafkaProducer.sendMessage("myTopic", String.valueOf(stock.getStockID()),
message);
        }
        return "redirect:/stocks";
    }

    @GetMapping("/{id}")
    public String getStock(@PathVariable Integer id, Model model) {
        itemObjRepository.findById(id)
                .ifPresent(stock -> model.addAttribute("stock", stock));
        return itemObjRepository.findById(id).isPresent() ? "stock" : "redirect:/stocks";
    }

    @PostMapping("/delete/{id}")
    public String deleteStock(@PathVariable Integer id) {
        itemObjRepository.deleteById(id);
        return "redirect:/stocks";
    }

    private String convertStockToMessage(ItemObj patient) {
        try {
            return objectMapper.writeValueAsString(patient);
        } catch (Exception e) {
            throw new RuntimeException("Ошибка при преобразовании", e);
        }
    }
```

```
}
```

Websequrityconfig

```
package com.example.Project;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.provisioning.InMemoryUserDetailsManager;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
@EnableWebSecurity
public class WebSecurityConfig {

    // Определение цепочки фильтров безопасности
    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
```

```
                .authorizeRequests(authorize -> authorize
                        .requestMatchers("/", "/home", "/login", "/style.css", "/js/**", "/images/**").permitAll()
                        .requestMatchers(HttpMethod.POST, "/stocks").authenticated()
                        .anyRequest().authenticated()
                )
                .formLogin(form -> form
                        .loginPage("/login")
                        .defaultSuccessUrl("/stocks", true)
                        .permitAll()
                )
                .logout(logout -> logout.permitAll());
        return http.build();
    }

    // Настройка пользователей в памяти
    @Bean
    public InMemoryUserDetailsManager userDetailsService() {
        UserDetails user = User.withDefaultPasswordEncoder()
                .username("user")
                .password("password")
                .roles("ADMIN")
                .build();
        return new InMemoryUserDetailsManager(user);
    }

    // Настройка менеджера аутентификации
    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth
```

```
                .inMemoryAuthentication()
                .withUser("user")
                .password("{noop}password")
                .roles("ADMIN");
    }
}
```

Style.css

```
/* Основные стили */
body {
    font-family: 'Arial', sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
    color: #333;
}

.container {
    width: 90%;
    max-width: 1200px;
    margin: 20px auto;
    padding: 15px;
    background: white;
    border-radius: 10px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    overflow: hidden;
}
```

```
h1, h2 {
    color: #5cb85c; /* Зеленый цвет */
    text-align: center;
    margin-bottom: 20px;
}

/* Стили таблиц */
table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
}

th, td {
    padding: 10px;
    border: 1px solid #ddd;
    text-align: left;
}

th {
    background-color: #5cb85c; /* Зеленый цвет */
    color: white;
}

tr:nth-child(even) {
    background-color: #f2f2f2;
}

tr:hover {
    background-color: #e2e2e2;
}

/* Стили форм и элементов ввода */
form {
    margin-top: 20px;
    text-align: left;
    padding: 20px;
    background: #fff;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

label {
    margin-top: 10px;
    display: block;
    font-weight: bold;
}

input[type="text"],
input[type="number"],
input[type="date"],
input[type="password"],
select {
    width: 100%;
    padding: 8px;
    margin-top: 5px;
    border: 1px solid #ddd;
    border-radius: 4px;
    box-sizing: border-box;
}

/* Стили кнопок */
.delete-button, .details-button, button, input[type="submit"] {
    padding: 10px 15px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    margin-top: 10px;
    width: 100%;
    text-align: center;
}

.details-button { /* Зеленая кнопка для Details */
    background-color: #5cb85c;
    color: white;
}

.details-button:hover {
    background-color: #4cae4c;
}

.delete-button { /* Красная кнопка для Delete */
    background-color: #d9534f;
    color: white;
}

.delete-button:hover {
    background-color: #c9302c;
}

.submit-button { /* Зеленая кнопка для Submit */
    background-color: #5cb85c;
    color: white;
}

.submit-button:hover {
    background-color: #4cae4c;
}

/* Стили для страницы входа и информации об акции */
.login-form {
    max-width: 400px;
    margin: 50px auto;
    padding: 20px;
    background: white;
    border-radius: 10px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    text-align: center;
}

.login-form input[type="text"],
.login-form input[type="password"] {
    margin-bottom: 15px;
}

.login-form label {
    margin-bottom: 5px;
    display: block;
    text-align: left;
}
.stock-info {
    max-width: 400px;
    margin: 0 auto;
    padding: 20px;
    background: white;
    border-radius: 10px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

/* Стили ссылок */
a {
    color: #4a90e2;
    text-decoration: none;
}

a:hover {
    text-decoration: underline;
}

/* Медиа-запросы для адаптивного дизайна */
@media (max-width: 768px) {
    .container, .login-form, .stock-info {
        width: 95%;
        padding: 10px;
    }
```

```
}
```

Login.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title th:text="#{login.title}">Вход в систему управления акциями</title>
    <link rel="stylesheet" type="text/css" th:href="@{/style.css}" />
</head>
<body>
<div class="login-form">
    <h2 th:text="#{login.header}">Вход в систему управления акциями</h2>
    <form action="/login" method="post" th:action="@{/login}">
        <input type="hidden" th:name="${_csrf.parameterName}" th:value="${_csrf.token}"/>
        <div>
            <label th:text="#{login.username}">Имя пользователя:</label>
            <input type="text" name="username" required>
        </div>
        <div>
            <label th:text="#{login.password}">Пароль:</label>
            <input type="password" name="password" required>
        </div>
        <div>
            <input type="submit" class="submit-button" th:value="#{login.submit}"
value="Войти">
        </div>
    </form>
</div>
```

```
</body>
</html>
```

Stock.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title th:text="#{stock.title}">Акция</title>
  <link rel="stylesheet" type="text/css" th:href="@{/style.css}" />
</head>
<body>
<div class="container">
  <h1 th:text="#{stock.info}">Информация об акции</h1>
  <p><strong th:text="#{stock.name}">Название акции:</strong> <span th:text="${stock.stock_name}"></span></p>
  <p><strong th:text="#{stock.id}">ID акции:</strong> <span th:text="${stock.stockID}"></span></p>
  <p><strong th:text="#{stock.date}">Дата покупки:</strong> <span th:text="${stock.purchase_date}"></span></p>
  <a th:href="@{/stocks}" th:text="#{stock.return}">Вернуться к списку акций</a>
</div>
</body>
</html>
```

Stocks.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
```

```
  <meta charset="UTF-8">
  <title th:text="#{stocks.title}">Акции</title>
  <link rel="stylesheet" type="text/css" th:href="@{/style.css}" />
</head>
<body>
<div class="container">
  <h1 th:text="#{stocks.header}">Акции</h1>
  <table>
    <tr>
      <th th:text="#{stocks.name}">Название акции</th>
      <th th:text="#{stocks.id}">ID акции</th>
      <th th:text="#{stocks.date}">Дата покупки</th>
      <th></th>
    </tr>
    <tr th:each="stock : ${stocks}">
      <td th:text="${stock.stock_name}"></td>
      <td th:text="${stock.stockID}"></td>
      <td th:text="${stock.purchase_date}"></td>
      <td>
        <a th:href="@{/stocks/{id}(id=${stock.stockID})}" class="submit-button" th:text="#{stocks.details}">Подробнее</a>
        <form th:action="@{/stocks/delete/{id}(id=${stock.stockID})}" method="post">
          <input type="submit" class="delete-button" th:value="#{stocks.delete}" value="Удалить">
        </form>
      </td>
    </tr>
  </table>
  <h2 th:text="#{stocks.add}">Добавить акцию</h2>
  <form th:action="@{/stocks}" method="post">
```

```
    <input type="text" name="stock_name" th:placeholder="#{stocks.stockName}" required>

    <input type="text" name="stockID" th:placeholder="#{stocks.stockId}" required>

    <input type="date" name="purchase_date" th:placeholder="#{stocks.purchaseDate}" required>

    <input type="submit" class="submit-button" th:value="#{stocks.submit}" value="Добавить">

  </form>

</div>

</body>

</html>
```

messages_en.properties

```
login.title=Stock Management System Login

login.header=Login to Stock Management System

login.username=Username

login.password=Password

login.submit=Login

stock.title=Stock

stock.info=Stock Information

stocks.header=Stocks

stocks.name=Name

stocks.id=ID

stocks.date=Purchase Date

stocks.details=Details

stocks.delete=Delete

stocks.add=Add New Stock

stocks.stockName=Stock Name

stocks.stockId=Stock ID
```

stocks.purchaseDate=Purchase Date

stocks.submit=Submit

stock.name=Name

stock.id=ID

stock.date=Purchase date

stock.return=Return to stock list

stocks.title=Stock

messages_ru.properties

login.title=Ð’ Ñ…Ð¾Ð´ Ð² Ñ • Ð¸Ñ • Ñ‚ ÐµÐ¼Ñƒ Ñƒ Ð¿Ñ€ Ð°Ð²Ð»ÐµÐ½Ð¸Ñ • Ð°ÐºÑ† Ð¸Ñ • Ð¼Ð¸

login.header=Ð’ Ñ…Ð¾Ð´ Ð² Ñ • Ð¸Ñ • Ñ‚ ÐµÐ¼Ñƒ Ñƒ Ð¿Ñ€ Ð°Ð²Ð»ÐµÐ½Ð¸Ñ • Ð°ÐºÑ† Ð¸Ñ • Ð¼Ð¸

login.username=Ð˜ Ð¼Ñ • Ð¿Ð¾Ð»ÑŒÐ·Ð¾Ð²Ð°Ñ‚ ÐµÐ»Ñ •

login.password=ÐŸÐ°Ñ€ Ð¾Ð»ÑŒ

login.submit=Ð’ Ð¾Ð¹Ñ‚ Ð¸

stock.title=Ð • ÐºÑ† Ð¸Ñ •

stock.info=Ð˜ Ð½Ñ„ Ð¾Ñ€ Ð¼Ð°Ñ† Ð¸Ñ • Ð¾Ð± Ð°ÐºÑ† Ð¸Ð¸

stocks.header=Ð • ÐºÑ† Ð¸Ð¸

stocks.name=Ð • Ð°Ð·Ð²Ð°Ð½Ð¸Ðµ

stocks.id=Ð˜ Ð´ÐµÐ½Ñ‚ Ð¸Ñ„ Ð¸ÐºÐ°Ñ‚ Ð¾Ñ€

stocks.date=Ð” Ð°Ñ‚ Ð° Ð¿Ð¾ÐºÑƒ Ð¿ÐºÐ¸

stocks.details=ÐŸÐ¾Ð´Ñ€ Ð¾Ð±Ð½ÐµÐµ

stocks.delete=Ð£Ð´Ð°Ð»Ð¸Ñ‚ ÑŒ

stocks.add=Ð” Ð¾Ð±Ð°Ð²Ð¸Ñ‚ ÑŒ Ð°ÐºÑ† Ð¸ÑŽ

stocks.stockName=Ð • Ð°Ð·Ð²Ð°Ð½Ð¸Ðµ Ð°ÐºÑ† Ð¸Ð¸

stocks.stockId=ID Ð°ÐºÑ† Ð¸Ð¸

stocks.purchaseDate=Ð” Ð°Ñ‚ Ð° Ð¿Ð¾ÐºÑƒ Ð¿ÐºÐ¸

stocks.submit=Ðž Ñ‚ Ð¿Ñ€ Ð°Ð²Ð¸Ñ‚ ÑŒ

stock.name=Ð • Ð°Ð·Ð²Ð°Ð½Ð¸Ðµ Ð°ÐºÑ† Ð¸Ð¸

```
stock.id=ID Ð°ÐºÑ† Ð¸Ð¸
stock.date=Ð” Ð°Ñ‚ Ð° Ð¿Ð¾ÐºÑƒ Ð¿ÐºÐ¸
stock.return=Ð’ ÐµÑ€ Ð½Ñƒ Ñ‚ ÑŒÑ • Ñ •  Ðº Ñ • Ð¿Ð¸Ñ • ÐºÑƒ  Ð°ÐºÑ† Ð¸Ð¹
stocks.title=Ð • ÐºÑ† Ð¸Ð¸
```

application.properties

```
server.port=8088
spring.datasource.url=jdbc:mysql://localhost:3306/stocks
spring.datasource.username=root
spring.datasource.password=12qwaszx
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
#spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.hibernate.ddl-auto=update
spring.kafka.bootstrap-servers=localhost:9092
spring.profiles.active=producer

# Producer configuration
spring.kafka.producer.bootstrap-servers=localhost:9092
```

consumer1.properties

```
server.port=9090
spring.datasource.url=jdbc:mysql://localhost:3306/stocks
spring.datasource.username=root
spring.datasource.password=12qwaszx
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
#spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.hibernate.ddl-auto=update
```

```
# Consumer configuration for consumer 1
spring.kafka.consumer.bootstrap-servers=localhost:9092
spring.kafka.consumer.group-id=myGroup
```

consumer2.properties

```
server.port=9091
spring.datasource.url=jdbc:mysql://localhost:3306/stocks
spring.datasource.username=root
spring.datasource.password=12qwaszx
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
#spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.hibernate.ddl-auto=update
# Consumer configuration for consumer 2
spring.kafka.consumer.bootstrap-servers=localhost:9092
spring.kafka.consumer.group-id=myGroup2
```

docker-compose.yml

```
version: "3.9"
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:latest
    ports:
      - "22181:2181"
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000
  kafka:
    image: confluentinc/cp-kafka:latest
    depends_on:
      - zookeeper
    ports:
      - "29092:9092"
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:9092,PLAINTEXT_HOST://localhost:29092
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1

  mysql:
    image: mysql:5.7
    environment:
      MYSQL_DATABASE: "stocks"
      MYSQL_ROOT_PASSWORD: "12qwaszx"
    ports:
      - "3306:3306"

  mykafkaproducer:
    build:
      context: /Users/andrey/Documents/IntelliJ/ProjectKafka111
    ports:
      - 8088:8088
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://mysql:3306/stocks
      SPRING_DATASOURCE_USERNAME: root
      SPRING_DATASOURCE_PASSWORD: 12qwaszx
      SPRING_DATASOURCE_DRIVERCLASSNAME: com.mysql.cj.jdbc.Driver
      SPRING_KAFKA_BOOTSTRAP_SERVERS: kafka:9092
    depends_on:
      - mysql
      - kafka

  consumer1:
    build:
      context: /Users/andrey/Documents/IntelliJ/ProjectKafka111
    ports:
      - 9090:9090
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://mysql:3306/stocks
      SPRING_DATASOURCE_USERNAME: root
      SPRING_DATASOURCE_PASSWORD: 12qwaszx
      SPRING_DATASOURCE_DRIVERCLASSNAME: com.mysql.cj.jdbc.Driver
      SPRING_KAFKA_BOOTSTRAP_SERVERS: kafka:9092
      SPRING_KAFKA_CONSUMER_GROUP_ID: myGroup
    depends_on:
      - mysql
      - kafka

  consumer2:
    build:
      context: /Users/andrey/Documents/IntelliJ/ProjectKafka111
    ports:
      - 9091:9091
    environment:
      SPRING_DATASOURCE_URL: jdbc:mysql://mysql:3306/stocks
      SPRING_DATASOURCE_USERNAME: root
      SPRING_DATASOURCE_PASSWORD: 12qwaszx
      SPRING_DATASOURCE_DRIVERCLASSNAME: com.mysql.cj.jdbc.Driver
      SPRING_KAFKA_BOOTSTRAP_SERVERS: kafka:9092
      SPRING_KAFKA_CONSUMER_GROUP_ID: myGroup2
    depends_on:
      - mysql
      - kafka
```

Dockerfile:

```
FROM eclipse-temurin:17.0.9_9-jdk
RUN mkdir -p /usr/src/myapp
COPY target/ProjectKafka111.jar /usr/src/myapp
RUN mkdir -p /usr/src/myapp/target
COPY target/keystore.p12 /usr/src/myapp/target
WORKDIR /usr/src/myapp
ENTRYPOINT ["java","-jar","./ProjectKafka111.jar"]
```