Федеральное государственное автономное

образовательное учреждение

Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| институт |
| Программная инженерия |
| кафедра |

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ**

|  |
| --- |
| Применение архитектурных шаблонов |
| тема |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Преподаватель | |  |  |  | Д. В. Грузенкин |
|  | |  | подпись, дата |  | инициалы, фамилия |
| Студент | КИ21-17/1Б, 032156940 |  |  |  | Н. А. Самарин |
|  | номер группы, зачётной книжки |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2024

**СОДЕРЖАНИЕ**

1 Задание............................................................................................................... 3

2 Ход работы........................................................................................................ 3

3 Вывод................................................................................................................. 7

**1 Задание**

Применить один из архитектурных шаблонов, приведённых в лекции (или  
выбрать любой другой на своё усмотрение) в коде из работы 7. Бизнес-логику  
приложения писать не нужно. Реализуем только связи между классами и  
модулями. В отчёте необходимо обосновать выбор реализованного шаблона.

**2 Ход работы**

На основе кода из работы 7 был реализован шаблон проектирования  
MVC. Использование шаблона обосновано тем, что данный компонент  
представляет собой сервер, взаимодействующий с несколькими мобильными  
приложениями через REST запросы для управления данными пользователей и  
записей. MVC позволяет разделить логику пользовательского интерфейса и  
бизнес-логику. Разработанный код представляет собой Spring RESTful сервер,  
реализующий MVC. В случае Spring RESTful сервера, контроллеры  
обрабатывают HTTP-запросы, модели предоставляют данные, а представления  
представлены в формате данных json или xml, передаваемых клиентским  
приложениям сервером с помощью HTTP.

Листинг 1 – RecordController

package org.example;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RestController;  
import java.util.List;  
@RestController  
public class RecordController {  
 RecordFactory recordFactory;  
 RecordDataRepository recordDataRepository;  
 @Autowired  
 public RecordController(RecordDataRepository recordDataRepository, RecordType   
 type) {  
 switch (type) {  
 case VACANCY -> recordFactory = new VacancyFactoryImpl();  
 case RESUME -> recordFactory = new ResumeFactoryImpl();  
 }  
 this.recordDataRepository = recordDataRepository;  
 }  
 @PostMapping  
 public void registerRecord(Object nonSpecificRecordData, Object specificRecord  
 Data) {  
 Record record = recordFactory.createRecord(nonSpecificRecordData, specificRe  
 cordData);  
 // Работа с recordDataRepository  
 }  
 @GetMapping  
 public Record getRecord(Long id) {  
 // Работа с recordDataRepository  
 return null;  
 }  
 @GetMapping  
 public List<Record> getRecords(List<Long> ids) {

Окончание листинга 1

// Работа с recordDataRepository  
 return null;  
 }  
 @PostMapping  
 public void promoteRecord(Record record){  
 // Работа с paymentSystem  
 PaymentSystem.makePayment();  
 }  
}

Листинг 2 – UserController

package org.example;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RestController;  
import java.util.List;  
@RestController  
public class UserController {  
 UserFactory userFactory;  
 UserDataRepository userDataRepository;  
 @Autowired  
 public UserController(UserFactory userFactory, UserDataRepository userDataRepo  
 sitory) {  
 this.userFactory = userFactory;  
 this.userDataRepository = userDataRepository;  
 }  
 @PostMapping  
 public void registerUser(Object userData) {  
 User user = userFactory.createUser(userData);  
 // Работа с userDataRepository  
 }  
 @GetMapping  
 public User getUser(Long id) {  
 // Работа с userDataRepository  
 return null;  
 }  
 @GetMapping  
 public List<User> getUsers(List<Long> ids) {  
 // Работа с userDataRepository  
 return null;  
 }  
 @PostMapping  
 public void sendConfirmationEmail(User user){  
 //работа с EmailSystem  
 //EmailSystem.sendEmail();  
 }  
 @PostMapping  
 public void sendPasswordRestorationEmail(User user){  
 //работа с EmailSystem  
 }  
}

Листинг 3 – AdministratorController

package org.example;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;

Окончание листинга 3

@RestController  
public class AdministratorController {  
 UserDataRepository userDataRepository;  
 RecordDataRepository recordDataRepository;  
 @Autowired  
 public AdministratorController(UserDataRepository userDataRepository, RecordDa  
 taRepository recordDataRepository) {  
 this.userDataRepository = userDataRepository;  
 this.recordDataRepository = recordDataRepository;  
 }  
 @PostMapping  
 public void sendModerationEmail(User user){  
 //работа с EmailSystem  
 //EmailSystem.sendEmail();  
 }  
 @DeleteMapping  
 public void deleteUser(Long id){}  
 @DeleteMapping("/records/{id}")  
 public void deleteRecord(@PathVariable Long id){}  
}

Листинг 4 – Record

package org.example;  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
@Entity  
public abstract class Record {  
 Object nonSpecificData;  
 private Long id;  
 public void setNonSpecificData(Object nonSpecificData) {  
 this.nonSpecificData = nonSpecificData;  
 }  
 public Object getNonSpecificData() {  
 return nonSpecificData;  
 }  
 public void setId(Long id) {  
 this.id = id;  
 }  
 @Id  
 public Long getId() {  
 return id;  
 }  
}

Листинг 5 – User

package org.example;  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
@Entity  
public abstract class User {  
 Object userData;  
 private Long id;  
 public void setUserData(Object userData) {  
 this.userData = userData;  
 }  
 public Object getUserData() {  
 return userData;

Окончание листинга 5

}  
 public void setId(Long id) {  
 this.id = id;  
 }  
 @Id  
 public Long getId() {  
 return id;  
 }  
}

Листинг 6 – RecordDataRepository

package org.example;  
import org.springframework.stereotype.Repository;  
import java.util.List;  
@Repository  
public class RecordDataRepository {  
 public void insertVacancy(IVacancy vacancy){  
 //работа с SQL  
 }  
 public void insertResume(IResume resume){  
 //работа с SQL  
 }  
 public IVacancy getVacancy(int id){  
 //работа с SQL  
 return null;  
 }  
 public IResume getResume(int id){  
 //работа с SQL  
 return null;  
 }  
 public List<IVacancy> getAllVacancies(){  
 //работа с SQL  
 return null;  
 }  
 public List<IResume> getResumes(){  
 //работа с SQL  
 return null;  
 }  
}

Листинг 7 – UserDataRepository

package org.example;  
import org.springframework.stereotype.Repository;  
import java.util.List;  
@Repository  
public class UserDataRepository {  
 public void insertUser(){  
 //работа с SQL  
 }  
 public User getUser(int id){  
 //работа с SQL  
 return null;  
 }  
 public List<User> getAllUsers(){  
 //работа с SQL  
 return null;  
 }

Окончание листинга 7

}

Листинг 8 – EmailSystem

package org.example;  
import org.springframework.stereotype.Service;  
import javax.mail.Message;  
import javax.mail.MessagingException;  
import javax.mail.Session;  
import javax.mail.Transport;  
import javax.mail.internet.InternetAddress;  
import javax.mail.internet.MimeMessage;  
import java.util.Properties;  
@Service  
public class EmailSystem {  
 public static void sendEmail(String to, String text, String subject){  
 String from = "sender@abc.com"; // sender email  
 String host = "127.0.0.1"; // mail server host  
 Properties properties = System.getProperties();  
 properties.setProperty("mail.smtp.host", host);  
 Session session = Session.getDefaultInstance(properties); // default session  
 try {  
 MimeMessage message = new MimeMessage(session); // email message  
 message.setFrom(new InternetAddress(from)); // setting header fields  
 message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));  
 message.setSubject(subject); // subject line  
 message.setText(text);  
 Transport.send(message);  
 } catch (MessagingException mex){ mex.printStackTrace(); }  
 }  
}

Листинг 9 – PaymentSystem

package org.example;  
import org.springframework.stereotype.Service;  
@Service  
public class PaymentSystem {  
 public static void makePayment(){  
 //работа с внешней системой оплаты  
 }  
}

**3 Вывод**

В ходе работы задание было выполнено.