ГУАП КАФЕДРА № 43

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ ПРЕПОДАВАТЕЛЬ

|  |  |  |
| --- | --- | --- |
| ассистент |  | П.А.Степанов |
| должность, уч. степень, звание | подпись, дата | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №6 |
| «Разработка приложения с использованием Hibernate» |
| по дисциплине: ТЕХНОЛОГИЯ РАЗРАБОТКИ СЕРВЕРНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ |

РАБОТУ ВЫПОЛНИЛА

|  |  |  |
| --- | --- | --- |
| СТУДЕНТКА ГР. | 4232 | Е.А.Уткина |
|  | подпись, дата | инициалы, фамилия |

Санкт-Петербург

2024

1. Цель работы:

Закрепить умения и навыки разработки приложений с базами данных средствами Hibernate.

1. Задание:

- Добавьте базу данных, модель данных, JPA репозитории и сервисы.

- Реализуйте операции чтения/записи из одной из предыдущих лабораторных работ (3 или 5) с помощью JPA.

Вариант 14: Сдача недвижимости в аренду.

1. Листинг программы:

ArartmentPE.java:

/\*  
 \* this code is available under GNU GPL v3  
 \* https://www.gnu.org/licenses/gpl-3.0.en.html  
 \*/  
package info.stepanoff.trsis.samples.db.model;  
  
import jakarta.persistence.\*;  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
import java.io.Serializable;  
  
  
@Entity  
@Table(name = "APARTMENT")  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
public class ApartmentPE implements Serializable {  
  
 private static final long *serialVersionUID* = 1L;  
  
 public ApartmentPE(String city, String address, Integer price) {  
 this.city = city;  
 this.address = address;  
 this.price = price;  
 }  
  
 @Id  
 @Column(name = "APARTMENT")  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Integer id;  
  
 @Column(name = "CITY")  
 private String city;  
 @Column(name = "ADDRESS")  
 private String address;  
 @Column(name = "PRICE")  
 private Integer price;  
  
}

ApartmentDTO.java:

/\*  
 \* this code is available under GNU GPL v3  
 \* https://www.gnu.org/licenses/gpl-3.0.en.html  
 \*/  
package info.stepanoff.trsis.samples.rest.model;  
  
import io.swagger.v3.oas.annotations.media.Schema;  
import lombok.Data;  
  
@Data  
@Schema(description = "Квартира")  
public class ApartmentDTO {  
  
 @Schema(description = "Идентификатор")  
 private Integer id;  
 @Schema(description = "Город")  
 private String city;  
 @Schema(description = "Адрес")  
 private String address;  
 @Schema(description = "Цена, руб./мес")  
 private Integer price;  
}

ApartmentDataWithoutID.java:

package info.stepanoff.trsis.samples.rest.model;  
  
import lombok.Getter;  
  
@Getter  
public class ApartmentDataWithoutID {  
 private String city;  
 private String address;  
 private Integer price;  
  
 public String getCity() {  
 return city;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public Integer getPrice() {  
 return price;  
 }  
}

ApartmentService.java:

/\*  
 \* this code is available under GNU GPL v3  
 \* https://www.gnu.org/licenses/gpl-3.0.en.html  
 \*/  
  
package info.stepanoff.trsis.samples.service;  
  
import info.stepanoff.trsis.samples.rest.model.ApartmentDTO;  
  
import java.util.List;  
  
public interface ApartmentService {  
  
 List<ApartmentDTO> listAll();  
  
 void delete(Integer id);  
   
 ApartmentDTO add(String city, String address, Integer price);  
   
 ApartmentDTO findByCity(String city);  
  
}

ApartmentServiceImpl.java:

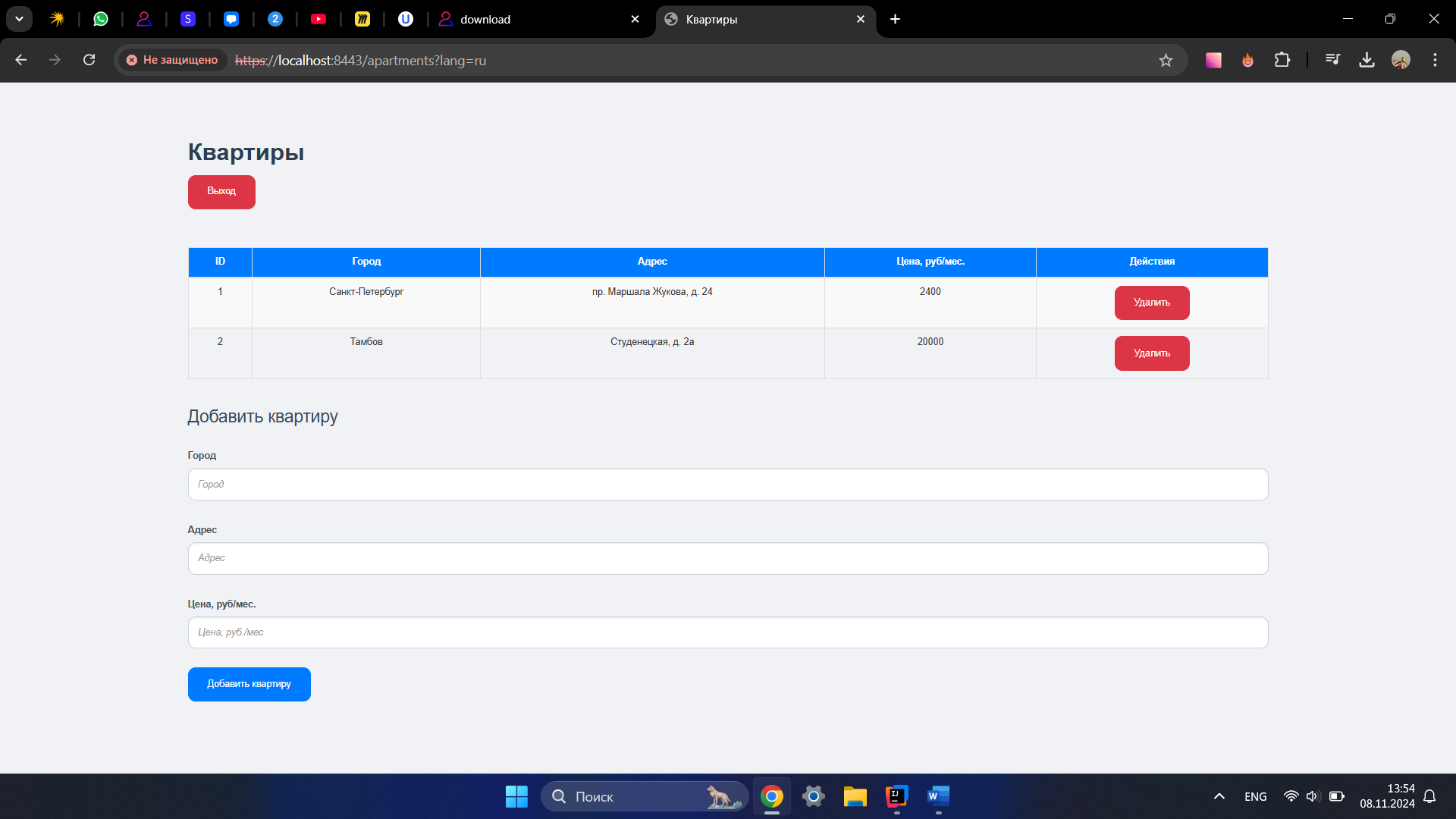
/\*  
 \* this code is available under GNU GPL v3  
 \* https://www.gnu.org/licenses/gpl-3.0.en.html  
 \*/  
package info.stepanoff.trsis.samples.service;  
  
import com.fasterxml.jackson.databind.ObjectMapper;  
import info.stepanoff.trsis.samples.db.dao.ApartmentRepository;  
import info.stepanoff.trsis.samples.db.model.ApartmentPE;  
import info.stepanoff.trsis.samples.rest.model.ApartmentDTO;  
import lombok.RequiredArgsConstructor;  
import lombok.extern.slf4j.Slf4j;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.stream.Collectors;  
  
@Slf4j  
@Service  
@RequiredArgsConstructor  
public class ApartmentServiceImpl implements ApartmentService {  
  
 private final ApartmentRepository ApartmentRepository;  
  
 private final ObjectMapper objectMapper;  
  
 @Override  
 public List<ApartmentDTO> listAll() {  
 return ApartmentRepository.findAll().stream()  
 .map(ApartmentPE -> objectMapper.convertValue(ApartmentPE, ApartmentDTO.class))  
 .collect(Collectors.toList());  
  
 }  
  
 @Override  
 public void delete(Integer id) {  
 ApartmentRepository.deleteById(id);  
 }  
  
 @Override  
 public ApartmentDTO add(String city, String address, Integer price) {  
 return objectMapper.convertValue(ApartmentRepository.save(new ApartmentPE(city, address, price)), ApartmentDTO.class);  
 }  
  
 @Override  
 public ApartmentDTO findByCity(String city) {  
 var studentPE = ApartmentRepository.findByCity(city);  
 return studentPE.map(Apartment -> objectMapper.convertValue(Apartment, ApartmentDTO.class)).orElse(null);  
 }  
}

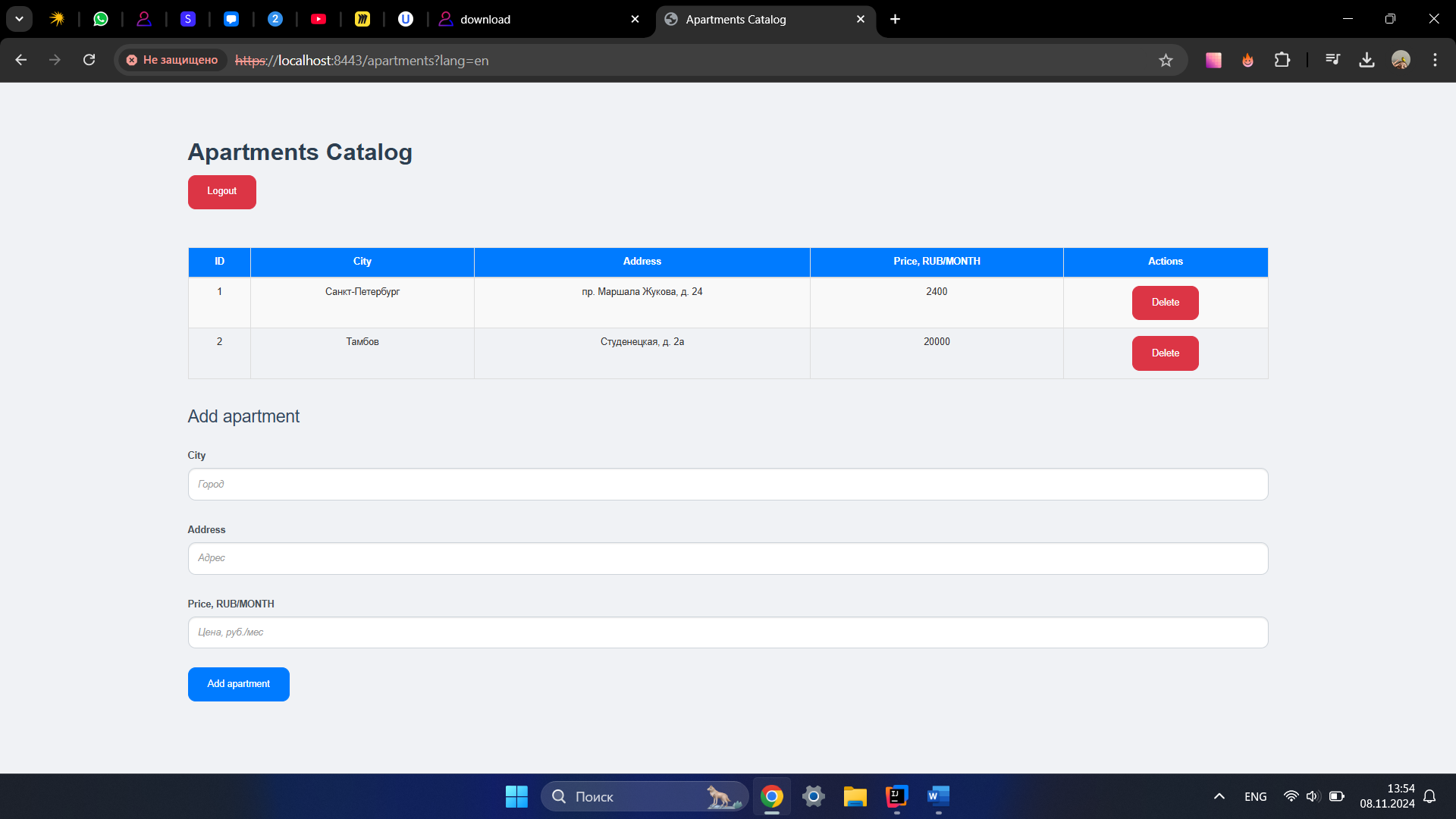
data.sql:

insert into APARTMENT (CITY, ADDRESS, PRICE) values ('Санкт-Петербург','пр. Маршала Жукова, д. 24', 2400);  
insert into APARTMENT (CITY, ADDRESS, PRICE) values ('Тамбов','Студенецкая, д. 2а', 20000);  
  
--login admin password hello  
insert into USERS (LOGIN, PASS\_HASH) values ('admin','$2a$10$6mf3CesQx9eRGB4B3sjr8e1eSr5cYO/zt87bwYVdA4O8rmjDMDdHO');

1. Пример выполнения программы:

Главная страница:





1. Вывод:

Были получены практические навыки работы с базой данных, моделями данных и JPA репозиториями и сервисами.