Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Кафедра «Информатика»
кафедра

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

Работа с базами данных в Spring MVC
тема

Преподаватель _____ А.С. Черниговский
дата, подпись инициалы, фамилия

Студент КИ18-16Б _____ К.И. Оглезнев
номер группы, зачётной книжки дата, подпись инициалы, фамилия

Красноярск 2021

1 Цель работы

Познакомиться с шаблоном MVC в Spring и тем как он используется при создании web-приложений.

2 Задачи работы

Изменить практическую работу №4 таким образом, чтобы она представляла собой web-приложение.

Web-приложение должно иметь следующие страницы:

1) Главная страница, содержит приветствие и ссылки на другие (которые дублируют по функционалу пункты меню из работы №4).
2) Страница просмотра таблицы записей.
3) Страница добавления новой записи в таблицу.
4) Страница редактирования записи.
5) Страница удаления записи из таблицы БД.
6) Страница просмотра записей согласно некоторому критерию (аналогично пункту в работе №4).

Помимо всего должны быть осуществлены проверки (не менее двух) входных данных, сопровождающиеся соответствующими сообщениями об ошибках.

3 Ход работы

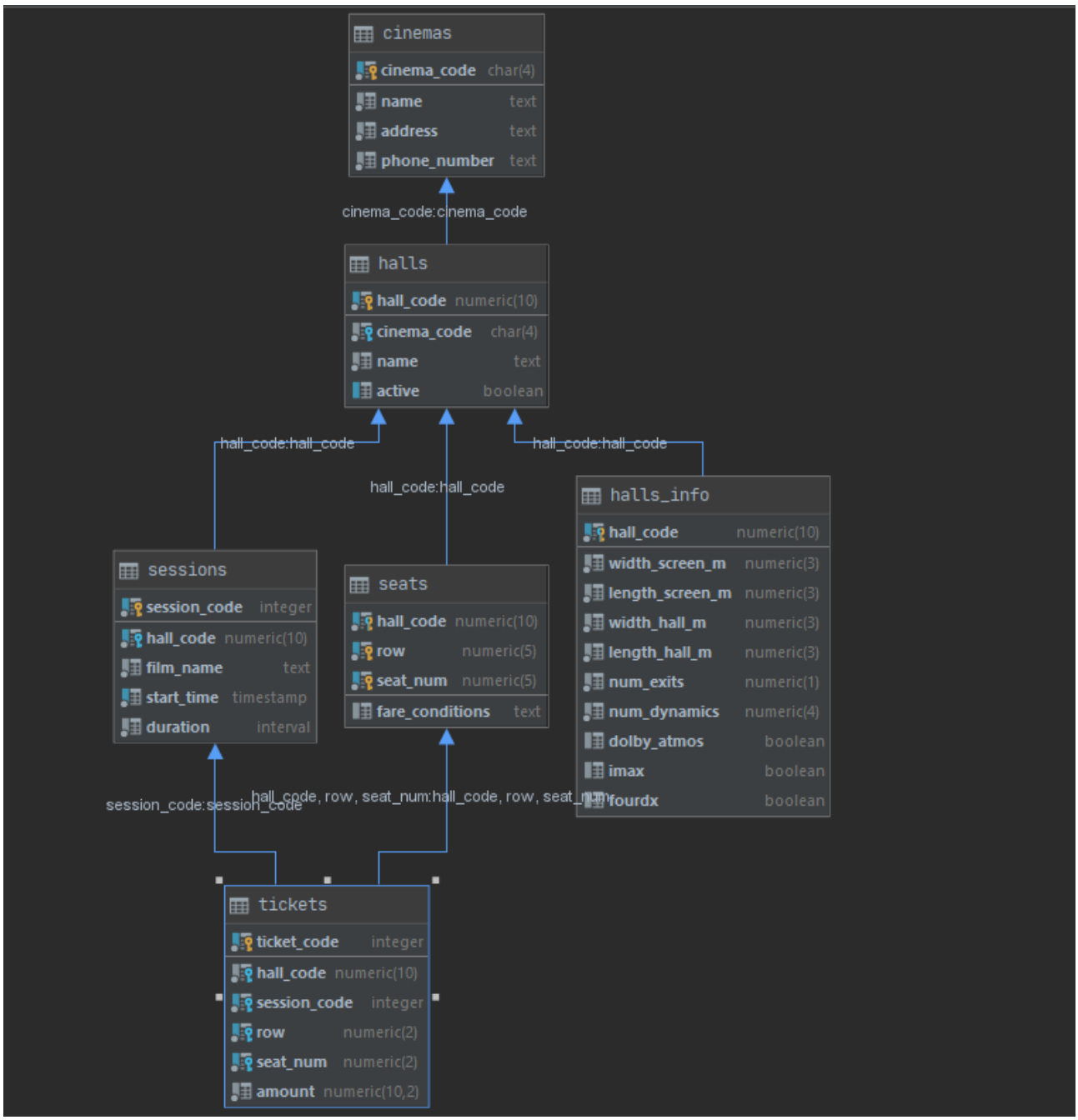
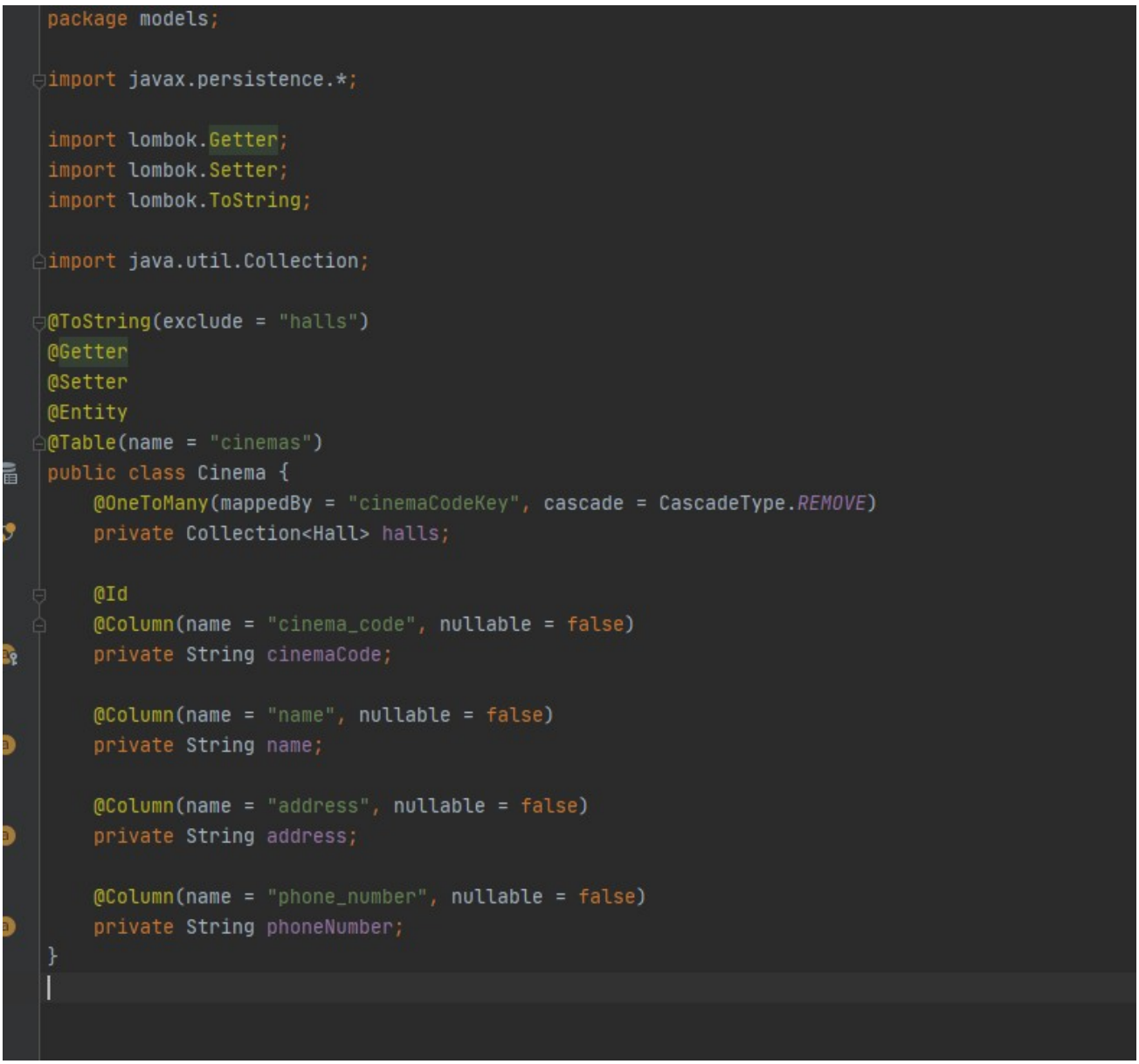

cinemas
cinema_code char(4)
name text
address text
phone_number text
cinema_code:cinema_code
halls
hall_code numeric(10)
cinema_code char(4)
name text
active boolean
hall_code:hall_code
hall_code:hall_code
hall_code:hall_code
halls_info
hall_code numeric(10)
width_screen_m numeric(3)
length_screen_m numeric(3)
width_hall_m numeric(3)
length_hall_m numeric(3)
num_exits numeric(1)
num_dynamics numeric(4)
dolby_atmos boolean
imax boolean
fourdx boolean
sessions
session_code integer
hall_code numeric(10)
film_name text
start_time timestamp
duration interval
seats
hall_code numeric(10)
row numeric(5)
seat_num numeric(5)
fare_conditions text
session_code:session_code
hall_code, row, seat_num:hall_code, row, seat_num
tickets
ticket_code integer
hall_code numeric(10)
session_code integer
row numeric(2)
seat_num numeric(2)
amount numeric(10,2)

Рисунок 1



```java
package models;

import javax.persistence.*;

import lombok.Getter;
import lombok.Setter;
import lombok.ToString;

import java.util.Collection;

@ToString(exclude = "halls")
@Getter
@Setter
@Entity
@Table(name = "cinemas")
public class Cinema {
    @OneToMany(mappedBy = "cinemaCodeKey", cascade = CascadeType.REMOVE)
    private Collection<Hall> halls;

    @Id
    @Column(name = "cinema_code", nullable = false)
    private String cinemaCode;

    @Column(name = "name", nullable = false)
    private String name;

    @Column(name = "address", nullable = false)
    private String address;

    @Column(name = "phone_number", nullable = false)
    private String phoneNumber;
}
```

Рисунок 2

```java
public class DAO {
    static ThreadLocal<Session> session = new ThreadLocal<>();
    static SessionFactory sessionFactory = getSessionFactory();
    static Transaction transaction = null;

    public static SessionFactory getSessionFactory() {
        if (sessionFactory == null) {

                Configuration configuration = new Configuration().configure("hibernate.cfg.xml");
                configuration.addAnnotatedClass(Cinema.class);
                configuration.addAnnotatedClass(Hall.class);
                configuration.addAnnotatedClass(HallInfo.class);
                configuration.addAnnotatedClass(Seat.class);
                configuration.addAnnotatedClass(models.Session.class);
                configuration.addAnnotatedClass(Ticket.class);
                StandardServiceRegistryBuilder builder = new StandardServiceRegistryBuilder().applySettings(configuration.getProperties());
                sessionFactory = configuration.buildSessionFactory(builder.build());


        }
        return sessionFactory;
    }

    public static Session getSession() {
        if (session.get() == null || !session.get().isOpen())
            session.set(sessionFactory.openSession());
        return session.get();
    }
```

Рисунок 3

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<!--xmlns = "http://www.hibernate.org/xsd/orm/cfg"-->
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.hbm2ddl.auto">update</property>
        <property name="hibernate.connection.driver_class">org.postgresql.Driver</property>
        <property name="hibernate.dialect">org.hibernate.dialect.PostgreSQL9Dialect</property>
        <property name="hibernate.connection.url">jdbc:postgresql://localhost:5432/Lab10</property>
        <property name="hibernate.connection.username">postgres</property>
        <property name="hibernate.connection.password">123</property>
        <property name="hibernate.show_sql">true</property>
        <property name="hibernate.temp.use_jdbc_metadata_defaults">false</property>

        <mapping class="models.Cinema"/>
        <mapping class="models.Hall"/>
        <mapping class="models.Seat"/>
        <mapping class="models.Session"/>
        <mapping class="models.Ticket"/>
        <mapping class="models.HallInfo"/>
    </session-factory>
</hibernate-configuration>
```

Рисунок 4

```java
@Controller
public class MainController {

    @GetMapping("/")
    public ModelAndView allCinemas(@RequestParam(required = false) String id) {
        ModelAndView mav = new ModelAndView( viewName: "index");
        mav.addObject( attributeName: "cinemaList", CinemaDAO.findAll());

        if (CinemaDAO.findById(id) != null)
            mav.addObject( attributeName: "cinema", CinemaDAO.findById(id));
        else
            mav.addObject( attributeName: "cinema", new Cinema());
        return mav;
    }

    @PostMapping(value = "/")
    public ModelAndView updateCinema(@ModelAttribute("cinema") Cinema cinema) {
        ModelAndView mav = new ModelAndView( viewName: "redirect:/#");
        var err :String  = CinemaDAO.insert(cinema);
        if (!err.equals("")) {
            mav.setViewName("index");
            mav.addObject( attributeName: "cinemaList", CinemaDAO.findAll());
            mav.addObject( attributeName: "cinema", cinema);

            switch (err) {
                case "Некорректный ID" -> mav.addObject( attributeName: "errID", err);
                case "Invalid name" -> mav.addObject( attributeName: "errName", err);
                case "Invalid address" -> mav.addObject( attributeName: "errAddress", err);
                case "Некорректный номер телефона" -> mav.addObject( attributeName: "errNumber", err);
            }
            return mav;
        }
        DAO.transaction();
        return mav;
    }
```

Рисунок 5

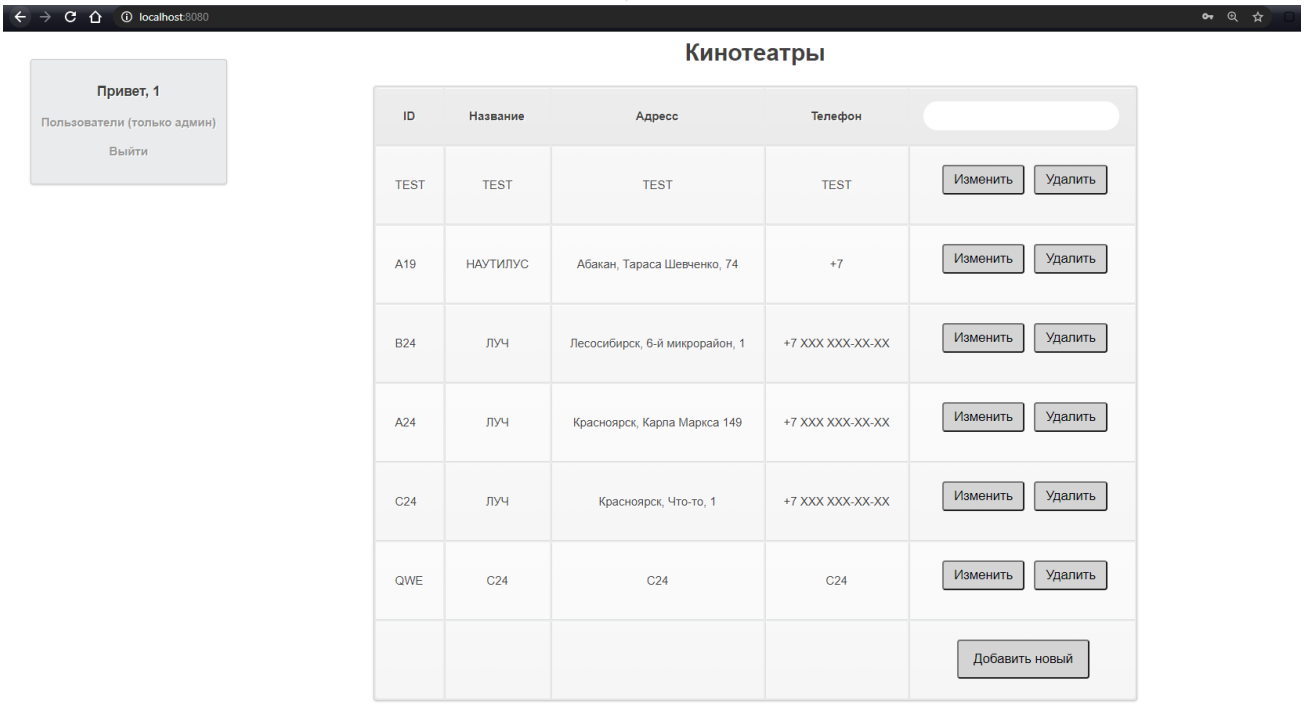


Рисунок 6

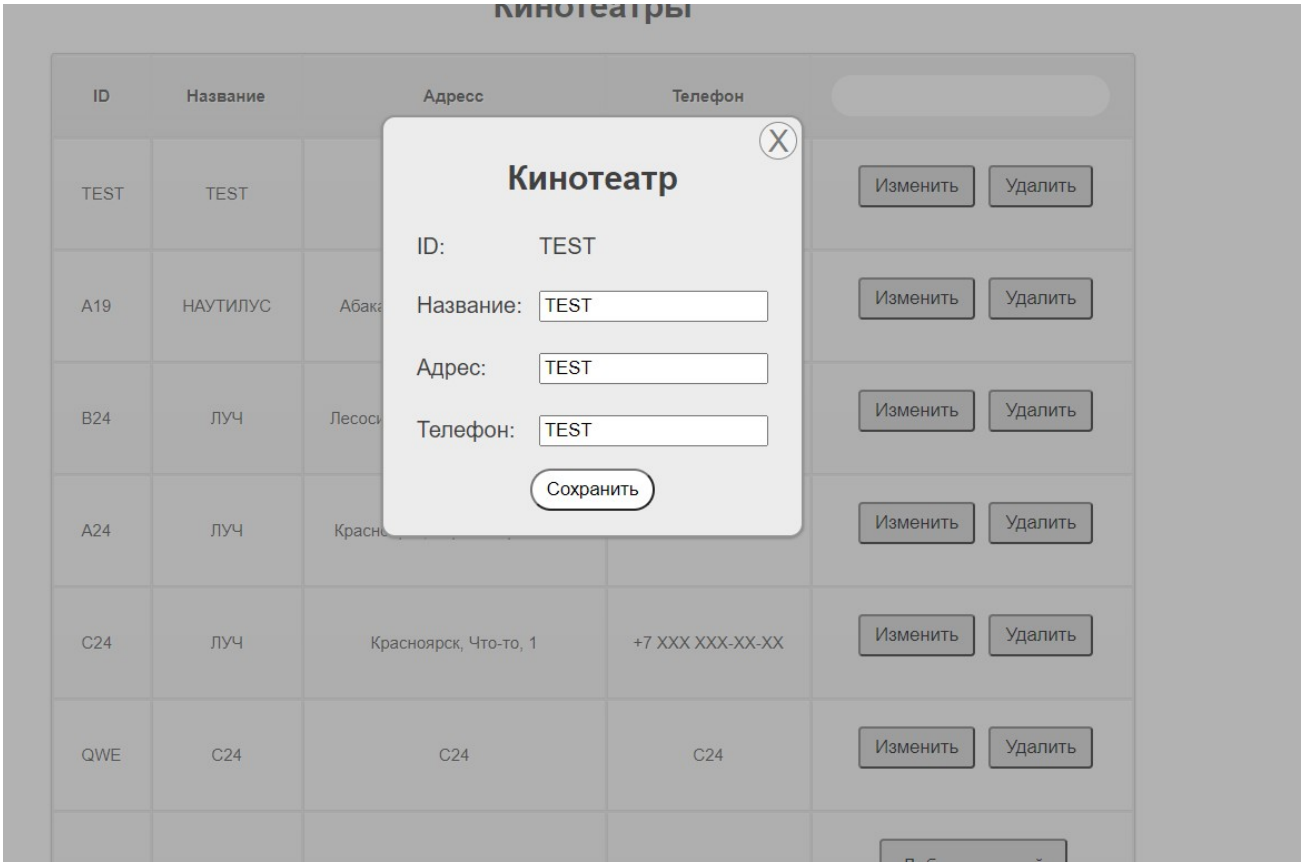


Рисунок 7

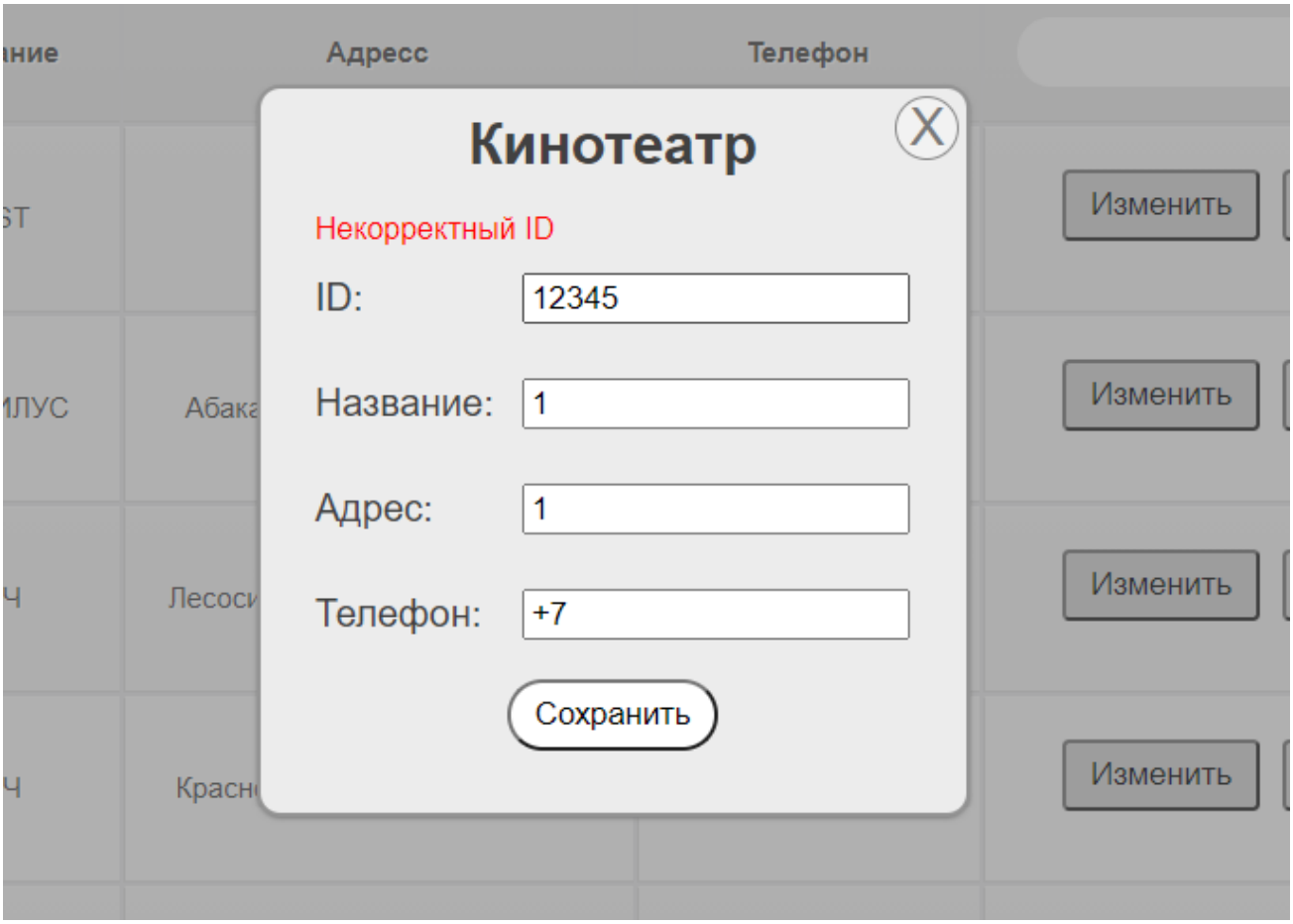
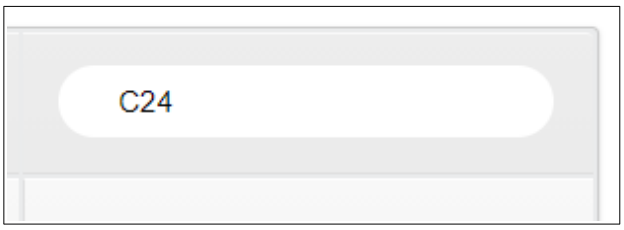


Рисунок 8

```
<sec:authorize access="hasRole('ROLE_ADMIN')">
    <form style="..." action="#edit" method="GET">
        <input type="hidden" name="id" value="${cin.cinemaCode}"/>
        <button type="submit">Изменить</button>
    </form>
    <form action="/delete" method="POST">
        <input type="hidden" name="Id" value="${cin.cinemaCode}"/>
        <input type="hidden" name="action" value="delete"/>
        <button type="submit">Удалить</button>
    </form>
</sec:authorize>
```

Рисунок 9



Результат поиска

ID	Название	Адресс	Телефон
C24	ЛУЧ	Красноярск, Что-то, 1	+7 XXX XXX-XX-XX
QWE	C24	C24	C24

Рисунок 10

4 Вывод

В ходе данной лабораторной работы был изучены основы работы с с шаблоном MVC в Spring и тем как он используется при создании web-приложений.