Informe individual de actividades del proyecto

Datos generales

URL del Repositorio de GitHub: https://github.com/gii-is-DP1/dp1-2020-g2-07
Nombre de usuario en GitHub:FranciscoJRE14
Rama del usuario en Github: franX

Participación en el proyecto

Historias de usuario en las que he participado

He implementado completas las historias de usuario H15-Crear Circuito, H16-Eliminar Circuito, H17-Editar Circuito,H-18-Mostrar Circuitos
He desarrollado junto a mi compañero las historias de usuario H19-Añadir un horario a un empleado ,H20- Añadir sesiones a un empleado en un determinado horario,H21-Ver Horarios trabajador.

Funcionalidad implementada

Entidades creadas en solitario: Horario.java, Circuito.java.
Repositorios creados en solitario: CircuitoRepository.java.
Controladores creados en solitario: CircuitoController.java.
Servicios creados en solitario: CircuitoService.java.

He añadido métodos a los siguientes servicios y controladors:SalaService.java,SalaController.java,HorarioController.java,HorarioRepository.java.

Vistas creadas:circuitosListing,createOrUpdateCircuitosForm,horariosForm,employeeSchedule.

He añadido elementos a la vista employeeDetails.

Esto hace un total de 5 clases implementadas por mi y 4 interfaces definidas.

Pruebas implementadas

Pruebas unitarias

Test de los servicios:SalaService(RoomMockTest.java),CircuitoSerivice(CircuitMockTest.java)
Test de los controladores:CircuitoController(CircuitoControllerTest.java)
Test de los formatter:SalaFormatter(SalaFormatterTest.java)

Pruebas de Controlador

Controlador CircuitoController:

- Historia de Usuario H15-Crear Circuito
  1 caso positivo, 1 caso negativo
- Historia de Usuario H16-Editar Circuito
  1 caso positivo, 1 caso negativo

- Historia de Usuario H17-Eliminar Circuito
  1 caso positivo, 1 caso negativo
- Historia de Usuario H17-Mostrar Circuito
  1 caso positivo

Pruebas de Service

•Service SalaService (4 casos positivos, 1 negativo)
•Service CircuitoService(5 casos positvos, 1 negativo)

Ejemplos de funcionalidades implementadas

Entidades (máximo de dos ejemplos)

Circuitos(/dp1-2020-g2-07/src/main/java/org/springframework/samples/petclinic/model/Circuito.java)
Esta entidad es relevante, ya que es un objeto del modelo de dominio en donde se guarda la informacion de las salas que component un circuito y sus caracteristicas.

```java
package org.springframework.samples.petclinic.model;
import java.util.List;

@Entity
@Data
@Table(name="circuitos")
public class Circuito extends NamedEntity {

    @JoinTable(
                name = "rel_circuito_salas",
                joinColumns = @JoinColumn(name="FK_Circuito"),
                inverseJoinColumns = @JoinColumn(name="FK_Sala")
            )


    @ManyToMany
    @Size(min = 2, message = "The circuit has to be composed of two rooms minimum.")
    private List<Sala> salas;

    private Integer aforo;

    @Size(min= 10, message = "The description needs to have at least ten letters.")
    @Column(name = "descripcion", length=1024)
    private String descripcion;


    public Sala getSala(String name) {
        return getSala(name);
    }

    public void addSala(Sala s){
        salas.add(s);
    }

    public void addSalas(List<Sala> s){
        salas.addAll(s);
    }


```

Horarios(/dp1-2020-g2-07/src/main/java/org/springframework/samples/petclinic/model/Horario.java)

Esta entidad es importante, ya que es un objeto del modelo de dominio en donde se guarda la fecha en la que un empleado cumple su horario laboral.

```java
package org.springframework.samples.petclinic.model;

import java.time.LocalDate;

@Entity
@Table(name = "horario")
public class Horario extends BaseEntity{

    @NotNull(message = "Date can't be null")
    @Column(name = "fecha")
    @DateTimeFormat(pattern = "yyyy/MM/dd")
    private LocalDate fecha;

    @NotNull
    @ManyToOne
    @JoinColumn(name="employee_id")
    private Employee employee;

    @OneToMany(mappedBy = "horario",cascade=CascadeType.ALL)
    private List<Sesion> sesiones;

    public Horario() {
        super();
    }

    public Horario(@NotNull(message = "Date can't be null") LocalDate fecha, @NotNull Employee employee,
            List<Sesion> sesiones) {
        super();
        this.fecha = fecha;
        this.employee = employee;
        this.sesiones = sesiones;
    }

    public LocalDate getFecha() {
        return fecha;
    }

    public Employee getEmployee() {
        return employee;
    }

    public void setFecha(LocalDate fecha) {
        this.fecha = fecha;
    }

    public void setEmployee(Employee employee) {
        this.employee = employee;
    }

    public List<Sesion> getSesiones() {
        return sesiones;
    }
```

Servicio

SalaService(/dp1-2020-g2-07/src/main/java/org/springframework/samples/petclinic/service/SalaService.java)
En este servicio se encuentras los metodos que hacen eficiente la aplicacion a la horar de crear/editar salas, asi tambiente la inyeccion de dependencia de esta clase en el controlador de Circuitos, asi correcto su funcionamiento. Tambien se han añadido la excepcion DuplicatedNameException.

```java
package org.springframework.samples.petclinic.service;
import java.util.Collection;

@Service
public class SalaService {


    SalaRepository salaRepo;

    @Autowired
    public SalaService(SalaRepository salaRepo) {
        this.salaRepo = salaRepo;

    }

    public Collection<Sala> findAll(){
        return salaRepo.findAll();
    }

    public Optional<Sala> findById(int id){
        return salaRepo.findById(id);
    }

//    public List<Sala> findByIdLista(int id){
//        return salaRepo.findById(id);
//    }

    @Transactional
    public void delete(Sala sala) {
        salaRepo.deleteById(sala.getId());
    }

    @Transactional
    public void save(@Valid Sala sala) {
        salaRepo.save(sala);
    }
    public Sala getRoomwithIdDIfferent(String name,Integer id) {
        name=name.toLowerCase();
        for (Sala sala: getRooms()) {
            String compName=sala.getName();
            compName=compName.toLowerCase();
            if (compName.equals(name) && sala.getId()!=id) {
                return sala;
            }
        }
        return null;
    }
      public Collection<Sala> getRooms(){
        return salaRepo.findAll();
      }
    @Transactional(rollbackOn = DuplicatedSalaNameException.class)
    public void saveSala(Sala sala)  throws DuplicatedSalaNameException {
      Sala otherRoom=getRoomwithIdDIfferent(sala.getName(), sala.getId());
       if (StringUtils.hasLength(sala.getName()) &&  (otherRoom!= null && otherRoom.getId()!=sala.getId())) {
          throw new DuplicatedSalaNameException();
         }else
           salaRepo.save(sala);
    }

}
```

Controlador

CircuitoController(/dp1-2020-g2-07/src/main/java/org/springframework/samples/petclinic/web/CircuitoController.java)
Este controlador permite varias funcionalidades para la manipulacion de circuitos, asi como tal muestras, en el caso que se comentan, todo tipo de excepciones.

Nombre: Francisco Javier Rodriguez Elena Grupo: G2-07

```java
package org.springframework.samples.petclinic.web;
import java.util.Collection;


@Slf4j
@Controller
@RequestMapping("/circuitos")
public class CircuitoController {

    public static final String CIRCUITOS_FORM ="circuitos/createOrUpdateCircuitosForm";
    public static final String CIRCUITOS_LISTING ="circuitos/CircuitosListing";

    private final CircuitoService circuitosServices;
    private final SalaService salasServices;


    @Autowired
    public CircuitoController(CircuitoService circuitosServices, SalaService salasServices) {
        this.circuitosServices = circuitosServices;
        this.salasServices = salasServices;
    }

    @GetMapping
    public String cicuitosListing(ModelMap model) {
        Collection<Circuito> c = circuitosServices.findAll();
        for(Circuito i:c) {
            i.setAforo(circuitosServices.getAforo(i));
        }
        model.addAttribute("circuitos",c);
        return CIRCUITOS_LISTING;
    }

    @GetMapping("/{id}/edit")
    public String editCircuito(@PathVariable("id") int id,ModelMap model ) {
        Optional<Circuito> circuito = circuitosServices.findById(id);
        if(circuito.isPresent()) {
            Circuito c = circuito.get();
            c.setAforo(circuitosServices.getAforo(c));
            model.addAttribute("circuito",c);
            model.addAttribute("salas", salasServices.findAll());
            return CIRCUITOS_FORM;
        }else {
            model.addAttribute("message","We could not find the circuit you are trying to edit.");

            return CIRCUITOS_LISTING;
        }
    }
    @PostMapping("/{id}/edit")
    public String editCircuito(@PathVariable("id") int id, @Valid Circuito modifiedCircuito, BindingResult binding,ModelMap model) {
        Optional<Circuito> circuito = circuitosServices.findById(id);
        if(binding.hasErrors()) {
            model.put("circuito", modifiedCircuito);
            model.put("salas", salasServices.findAll());
            log.warn("Some mistakes here..");
            return CIRCUITOS_FORM;
        }else {

            BeanUtils.copyProperties(modifiedCircuito, circuito.get(),"id","aforo","salas","descripcion");
            try{
                modifiedCircuito.setAforo(circuitosServices.getAforo(modifiedCircuito));
                this.circuitosServices.saveCircuito(modifiedCircuito);
             }catch(DuplicatedCircuitoNameException ex){
                model.put("circuito", modifiedCircuito);
                model.put("salas", salasServices.findAll());
                binding.rejectValue("name", "duplicate", "already exists");
                log.warn("Circuit with ID "+ id +" has a ducplicate name");
                return  CIRCUITOS_FORM;
             }
            log.info("Circuit with ID "+ id + " was edit");
            model.addAttribute("message", "The circuit was updated successfully.");
            return cicuitosListing(model);
        }
    }

    @GetMapping("/{id}/delete")
    public String deleteCircuito(@PathVariable("id") int id, ModelMap model) {
        Optional<Circuito> circuito = circuitosServices.findById(id);
        if(circuito.isPresent()) {
            circuitosServices.delete(circuito.get());
            model.addAttribute("message", "The circuit was deleted successfully.");
            log.info("Circuit with ID "+ id + " was deleted");
            return cicuitosListing(model);
        }else {
            model.addAttribute("message", "We could not find the circuit you are trying to delete.");
            log.warn("Circuit with ID "+ id +" does not exit");
            return cicuitosListing(model);
        }
    }


    @GetMapping("/new")
    public String newCircuito(ModelMap model) {
        model.addAttribute("circuito",new Circuito());
        model.addAttribute("salas", salasServices.findAll());
        return CIRCUITOS_FORM;
    }
```

```java
    @PostMapping("/new")
    public String saveNewCircuito(@Valid Circuito circuito, BindingResult binding,ModelMap model) {
        if(binding.hasErrors()) {
            model.put("circuito", circuito);
            model.put("salas", salasServices.findAll());
            log.warn("Some mistakes here..");
            return CIRCUITOS_FORM;

        }else {
             try{
                circuito.setAforo(circuitosServices.getAforo(circuito));
                this.circuitosServices.saveCircuito(circuito);
             }catch(DuplicatedCircuitoNameException ex){
                model.put("circuito", circuito);
                model.put("salas", salasServices.findAll());
                binding.rejectValue("name", "duplicate", "already exists");
                log.warn("This circuit  has a duplicate name");
                return  CIRCUITOS_FORM;
             }
            model.addAttribute("message", "The circuit was created successfully.");
            log.info("You have created a new circuit!");
            return cicuitosListing(model);

        }
    }
```

Repositorio

CircuitoRepository(/dp1-2020-g2-07/src/main/java/org/springframework/samples/petclinic/repository/CircuitoRepository.java)
Tenemos bastantes metodos asi para poder encontrar circuitos o circuito especifico o incluso para obtener salas de un circuito.

```java
package org.springframework.samples.petclinic.repository;
import java.util.Collection;

public interface CircuitoRepository extends Repository<Circuito, Integer>, CrudRepository<Circuito, Integer> {

    Collection<Circuito> findAll();
    Optional<Circuito> findById(int id);
    Optional<Circuito> findById(Integer id);

    @Query("SELECT sala FROM Sala sala ORDER BY sala.name")
    public Collection<Sala> getRoomsByCircuito();

}
```

Conversor o Formatter (si aplica)

No aplica.

Validador y anotación asociada (si aplica, máximo de dos ejemplos)

No aplica.

Etiquetas personalizadas (si aplica, máximo de dos ejemplos)

No aplica.

Ejemplos de pruebas implementadas

Pruebas unitarias (máximo de dos ejemplos)

RoomMockTest(/dp1-2020-g2-07/src/test/java/org/springframework/samples/petclinic/service/RoomMockTest.java)
En el arrange creamos la sala, en el act probamos el service y comprobamos mediante assert si devuelve lo que se pide. En el ultimo metodo probamos que salte la excepcion de DuplicatedName

```java
@RunWith(MockitoJUnitRunner.class)
public class RoomMockTest {

    @Mock
    private SalaRepository salasRepo;

    @Mock
    private SalaService salaService;
    private Sala s;
    private List<Circuito> circuitos;
    private Circuito c1;
    private Circuito c2;
    private Collection<Sala> salas;
    private Optional<Sala> salaOpt;

    @Before
    public void setUp() {
        salaService= new SalaService(salasRepo);
        s= new Sala();
        c1= new Circuito();
        c2= new Circuito();
        circuitos= new ArrayList<Circuito>();
        salas= new ArrayList<Sala>();

        c1.setAforo(7);
        c1.setDescripcion("Prueba de circuito");
        c1.setId(1);
        c1.setName("Circuito1");

        c2.setAforo(7);
        c2.setDescripcion("Prueba de circuito");
        c2.setId(1);
        c2.setName("Circuito2");


        circuitos.add(c1);circuitos.add(c2);
        s.setCircuitos(circuitos);

        s.setId(1);
        s.setName("Jacuzzi");
        s.setAforo(7);

        salas.add(s);

        salaOpt=Optional.of(s);

        when(salasRepo.findAll()).thenReturn(salas);
        when(salasRepo.findById(1)).thenReturn(salaOpt);
        when(salasRepo.save(s)).thenReturn(s);

    }
```

```java
    @Test
    public void shouldFindAll() {
        Collection<Sala> salaExample=this.salaService.findAll();

        assertThat(salaExample).hasSize(1);
        assertThat(salaExample.iterator().next().getName()).isEqualTo("Jacuzzi");

    }

    @Test
    public void shouldFindById() {
        Optional<Sala> optionalSalaExample1=salaService.findById(1);
        assertTrue(optionalSalaExample1.isPresent());

        Optional<Sala> optionalSalaExample2=salaService.findById(2);
        assertFalse(optionalSalaExample2.isPresent());


    }

    @Test
    public void shouldSave() {
        salaService.save(s);

    }

    @Test
    public void shouldDeleteRoom() {
        salaService.delete(s);
    }

    @Test
    public void shouldThrowExceptionInsertingRoomsWithTheSameName() {
        s= new Sala();
        s.setId(1);
        s.setName("Jacuzzi");
        s.setAforo(7);
        try {
            salaService.saveSala(s);
        } catch (DuplicatedSalaNameException e) {
            // The room already exist
            e.printStackTrace();
        }
        Sala anotherRoomWithTheSameName= new Sala();
        anotherRoomWithTheSameName.setId(2);
        anotherRoomWithTheSameName.setName("Jacuzzi");
        anotherRoomWithTheSameName.setAforo(7);
        Assertions.assertThrows(DuplicatedSalaNameException.class, () -> {
            salaService.saveSala(anotherRoomWithTheSameName);
        });
    }
```

CircuitMockTest(/dp1-2020-g2-07/src/test/java/org/springframework/samples/petclinic/service/CircuitMockTest.java)
En el arrange creamos la sala, en el act probamos el service y comprobamos mediante assert si devuelve lo que se pide. En el ultimo metodo probamos que salte la excepcion de DuplicatedName y Tambien probamos en el metodo aforoCorrect si devuelve el minimo del aforo de las salas.

Nombre: Francisco Javier Rodriguez Elena Grupo: G2-07

```java
package org.springframework.samples.petclinic.service;
import static org.assertj.core.api.Assertions.assertThat;

@ExtendWith(MockitoExtension.class)
@RunWith(MockitoJUnitRunner.class)
public class CircuitMockTest {

    @Mock
    private CircuitoRepository circuitoRepo;


    @Mock
    private SalaRepository salaRepo;

    @Mock
    private CircuitoService circuitoService;
    private Circuito c;
    private Sala s;
    private Sala s2;
//  private Sala s2;
    private List<Sala> salas;
    private Collection<Circuito> circuitos;
    private Optional<Circuito> circuitoOpt;

    @Before
    public void setUp() {
        circuitoService= new CircuitoService(circuitoRepo);
        c = new Circuito();
        s = new Sala();
        s2 = new Sala();
        salas= new ArrayList<Sala>();
        circuitos=new ArrayList<Circuito>();

        s.setName("Jacuzzi");
        s.setAforo(7);
        s.setDescripcion("Prueba sobre las salas");
        s.setId(1);

        s2.setName("Relax Pool");
        s2.setAforo(7);
        s2.setDescripcion("Prueba sobre las salas");
        s2.setId(2);

        salas.add(s);salas.add(s2);
        c.setSalas(salas);

        c.setName("Circuito");
        c.setDescripcion("Circuito lleno de relajacion");
        c.setAforo(circuitoService.getAforo(c));
        c.setId(1);

        circuitos.add(c);

        circuitoOpt=Optional.of(c);

         when(circuitoRepo.findAll()).thenReturn(circuitos);
         when(circuitoRepo.findById(1)).thenReturn(circuitoOpt);
         when(circuitoRepo.save(c)).thenReturn(c);

    }
    @Test
    public void shouldFindAll() {
        Collection<Circuito> circuitoExample=this.circuitoService.findAll();

        assertThat(circuitoExample).hasSize(1);
        assertThat(circuitoExample.iterator().next().getName()).isEqualTo("Circuito");

    }

    @Test
    public void shouldFindById() {
        Optional<Circuito> optionalCircuitoExample1=circuitoService.findById(1);
        assertTrue(optionalCircuitoExample1.isPresent());

        Optional<Circuito> optionalCircuitoExample2=circuitoService.findById(2);
        assertFalse(optionalCircuitoExample2.isPresent());


    }

```

```
}

@Test
public void aforoCorrect() {
    c = new Circuito();
    s = new Sala();
    s2 = new Sala();
    salas= new ArrayList<Sala>();
    circuitos=new ArrayList<Circuito>();

    s.setName("Jacuzzi");
    s.setAforo(7);
    s.setDescripcion("Prueba sobre las salas");
    s.setId(1);

    s2.setName("Relax Pool");
    s2.setAforo(7);
    s2.setDescripcion("Prueba sobre las salas");
    s2.setId(2);

    salas.add(s);salas.add(s2);
    c.setSalas(salas);

    c.setName("Circuito");
    c.setDescripcion("Circuito lleno de relajacion");
    c.setAforo(circuitoService.getAforo(c));
    c.setId(1);

    assertEquals(7, 7);


}

@Test
public void shouldSave() {
    circuitoService.save(c);

}

@Test
public void shouldDeleteCircuit() {
    circuitoService.delete(c);
}

@Test
public void shouldThrowExceptionInsertingCircuitsWithTheSameName() {
    c = new Circuito();
    s = new Sala();
    s2 = new Sala();
    salas= new ArrayList<Sala>();
    circuitos=new ArrayList<Circuito>();

    s.setName("Jacuzzi");
    s.setAforo(7);
    s.setDescripcion("Prueba sobre las salas");
    s.setId(1);

    s2.setName("Relax Pool");
    s2.setAforo(7);
    s2.setDescripcion("Prueba sobre las salas");
    s2.setId(2);

    salas.add(s);salas.add(s2);
    c.setSalas(salas);

    c.setId(1);
    c.setName("Circuito");
    c.setDescripcion("Circuito lleno de relajacion");
    c.setAforo(circuitoService.getAforo(c));

    try {
        circuitoService.saveCircuito(c);
    } catch (DuplicatedCircuitoNameException e) {
        // The room already exist
        e.printStackTrace();
    }
    Circuito anotherCircuitWithTheSameName= new Circuito();
    anotherCircuitWithTheSameName.setId(2);
    anotherCircuitWithTheSameName.setName("Circuito");
    anotherCircuitWithTheSameName.setAforo(7);
    Assertions.assertThrows(DuplicatedCircuitoNameException.class, () -> {
        circuitoService.saveCircuito(anotherCircuitWithTheSameName);
    });
}
```

Pruebas unitarias parametrizadas (si aplica)

No aplica

Pruebas de controlador

CircuitControllerTest(/dp1-2020-g2-07/src/test/java/org/springframework/samples/petclinic/web/CircuitControllerTest.java)
En el arrange creamos todos los objetos necesarios, en el caso del circuito, las salas son una colección. En los métodos vamos comprobando si el circuito existe, si tiene posibles fallos en los campos a rellenar en su formulario.

Nombre: Francisco Javier Rodriguez Elena Grupo: G2-07

```java
package org.springframework.samples.petclinic.web;

import java.util.ArrayList;

@WebMvcTest(controllers = CircuitoController.class,excludeFilters = @ComponentScan.Filter(type = Filte
excludeAutoConfiguration = SecurityConfiguration.class)
public class CircuitoControllerTest {

    private static final int TEST_CIRCUITO_ID=1;

    @Autowired
    private MockMvc mockMvc;
    private Circuito c;
    private Optional<Circuito> circuitoOpt;
    private Sala s;
    private Sala s2;
    private Collection<Circuito> circuitos;

    @MockBean
    private CircuitoService circuitoService;

    @MockBean
    private SalaService salaService;



    @BeforeEach
    private void setUp() {

         c = new Circuito();
         s = new Sala();
         s2 = new Sala();
        List<Sala> salas= new ArrayList<Sala>();
        //circuitos=new ArrayList<Circuito>();

        s.setName("Jacuzzi");
        s.setAforo(7);
        s.setDescripcion("Prueba sobre las salas");
        s.setId(1);

        s2.setName("Relax Pool");
        s2.setAforo(7);
        s2.setDescripcion("Prueba sobre las salas");
        s2.setId(2);

        salas.add(s);
        salas.add(s2);
        c.setSalas(salas);

        c.setName("Circuito");
        c.setDescripcion("Circuito lleno de relajacion");
        c.setAforo(circuitoService.getAforo(c));
        c.setId(1);


        circuitos = new ArrayList<Circuito>();
        circuitos.add(c);
        circuitoOpt=Optional.of(c);
        given(this.circuitoService.findAll()).willReturn(circuitos);
        given(this.circuitoService.findById(TEST_CIRCUITO_ID)).willReturn(circuitoOpt);
        given(this.circuitoService.findRoomByCircuit()).willReturn(salas);

    }

    @WithMockUser(value="spring")
    @Test
    public void testCreationCircuit() throws Exception{
        mockMvc.perform(get("/circuitos/new")).andExpect(status().isOk())
        .andExpect(view().name("circuitos/createOrUpdateCircuitosForm"))
        .andExpect(model().attributeExists("circuito"))
        .andExpect(model().attributeExists("salas"));
    }

    @WithMockUser(value="spring")
    @Test
    public void testCreationCircuitSuccess() throws Exception{
        mockMvc.perform(post("/circuitos/new").with(csrf()).param("name", "Circuito")
                .param("salas","Jacuzzi","Relax Pool")
                .param("aforo", "7")
                .param("descripcion", "Circuito lleno de relajacion"))
        .andExpect(status().isOk())
        .andExpect(model().attribute("message","The circuit was created successfully."))
        .andExpect(view().name("circuitos/CircuitosListing"));
    }

    @WithMockUser(value="spring")
    @Test
    public void testCreationCircuitHasErrors() throws Exception{
        mockMvc.perform(post("/circuitos/new").with(csrf()).param("name", "Circuito")
                .param("salas","")
        .param("aforo", "siete").param("descripcion", "Circuito lleno de relajacion"))
        .andExpect(model().attributeHasErrors("circuito"))
        .andExpect(model().attributeHasFieldErrors("circuito", "aforo"))
        .andExpect(model().attributeHasFieldErrors("circuito", "salas"))
        .andExpect(status().isOk())
        .andExpect(view().name("circuitos/createOrUpdateCircuitosForm"));
    }

    @WithMockUser(value="spring")
    @Test
    public void testEditCircuit() throws Exception{
        mockMvc.perform(get("/circuitos/{id}/edit",TEST_CIRCUITO_ID))
        .andExpect(status().isOk())
        .andExpect(view().name("circuitos/createOrUpdateCircuitosForm"))
        .andExpect(model().attributeExists("circuito"))
        .andExpect(model().attributeExists("salas"));
    }

    @WithMockUser(value="spring")
    @Test
    public void testEditCircuitSuccess() throws Exception{
        mockMvc.perform(post("/circuitos/{id}/edit",TEST_CIRCUITO_ID).with(csrf()).param("name", "Circuito")
                .param("salas","Jacuzzi","Sauna")
                .param("aforo", "6").param("descripcion", "Circuito lleno de relajacion"))
        .andExpect(status().isOk())
        .andExpect(model().attribute("message","The circuit was updated successfully."))
        .andExpect(view().name("circuitos/CircuitosListing"));

    }

    @WithMockUser(value="spring")
    @Test
    public void testEditCircuitFailure() throws Exception{
        mockMvc.perform(get("/circuitos/{id}/edit",3).with(csrf()))
        .andExpect(status().isOk())
        .andExpect(model().attribute("message","We could not find the circuit you are trying to edit."))
        .andExpect(view().name("circuitos/CircuitosListing"));

    }

    @WithMockUser(value="spring")
    @Test
    public void testDeleteCircuitSuccess() throws Exception{
        mockMvc.perform(get("/circuitos/{id}/delete",TEST_CIRCUITO_ID))
        .andExpect(status().isOk())
        .andExpect(view().name("circuitos/CircuitosListing"))
        .andExpect(model().attribute("message","The circuit was deleted successfully."));
    }

    @WithMockUser(value="spring")
    @Test
    public void testDeleteCircuitFailure() throws Exception{
        mockMvc.perform(get("/circuitos/{id}/delete",2))
        .andExpect(status().isOk())
        .andExpect(view().name("circuitos/CircuitosListing"))
        .andExpect(model().attribute("message","We could not find the circuit you are trying to delete."));
    }
```

Principales problemas encontrados

El principal problema que me he encontrado al desarrollar el proyecto fue el diseño de una lógica de negocio que satisfaga la funcionalidad de circuitos y salas. El desconocimiento de la relación ManytoMany y el correcto uso de ella.

Otros comentarios

No procede.