

2024

PROYECTO TRANSVERSAL

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

CONTENIDO

1. EXPLICACION DEL PROYECTO	2
2. DISEÑO DEL PROYECTO	3
3. BASE DE DATOS	10
4. SPRING	12
5. VISTAS	34
6. ADAPTACIÓN A OTROS DISPOSITIVOS	41
7. DESPLIEGUE DE LA APLICACION	46
8. MODO DE TRABAJO	50

1. EXPLICACIÓN DEL PROYECTO

El proyecto final ha sido la creación de un casino donde hemos integrado nuestras prácticas que hemos realizado durante el curso, adaptándolas a las solicitudes que se pedían en este proyecto transversal como puede ser el uso de MVC o Responsive, aplicando lo aprendido en los distintos módulos que hemos cursado.

2. DISEÑO DEL PROYECTO

Para el diseño se escogió usar la temática de Arcane del universo League Of Legends (Tematica SteamPunk) tras ver la práctica de la ruleta de Jose Maria.



JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS



A partir de esta se fue adaptando todo el diseño del proyecto utilizando el estilo de la misma, por ejemplo:

-Fuente personalizada (Con licencia para uso comercial y personal)



-Colores coherentes

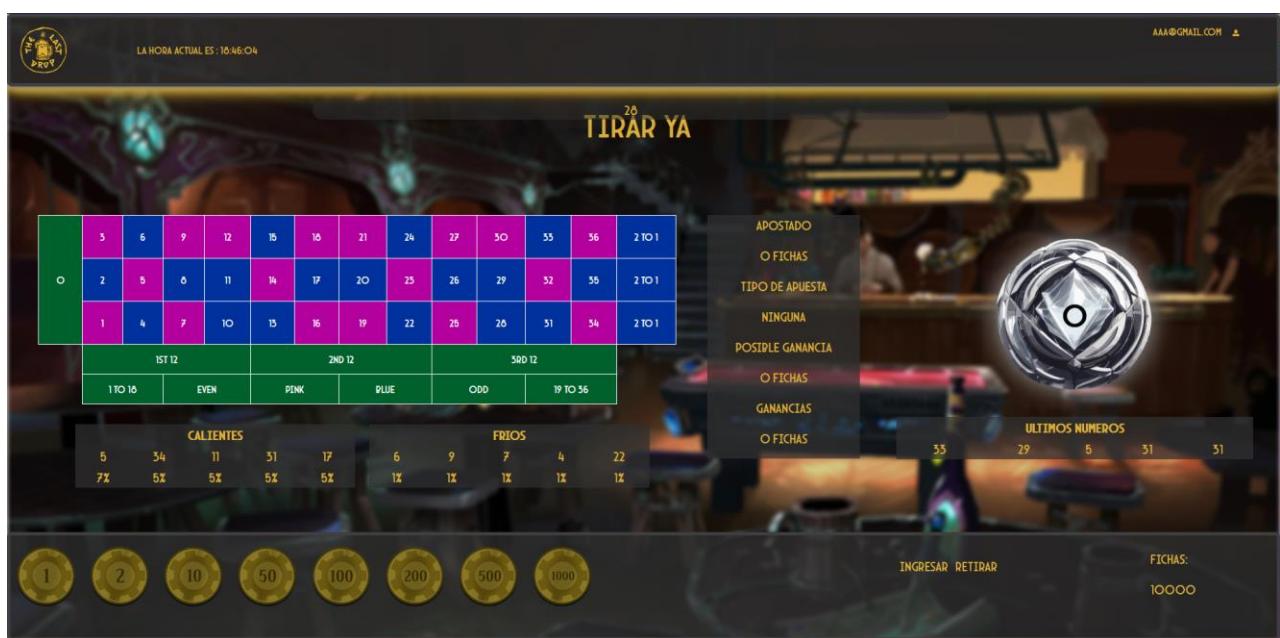
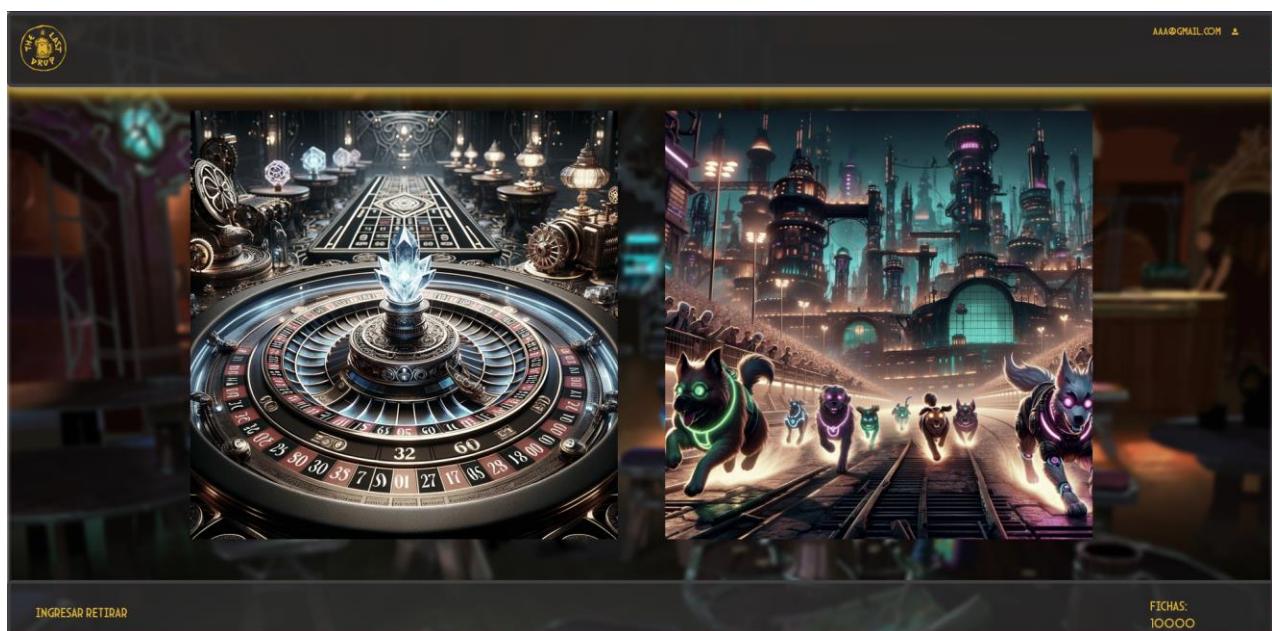
Hemos seleccionado un color para la fuente que se pudiera ver bien con fondos oscuros

● #dcb038

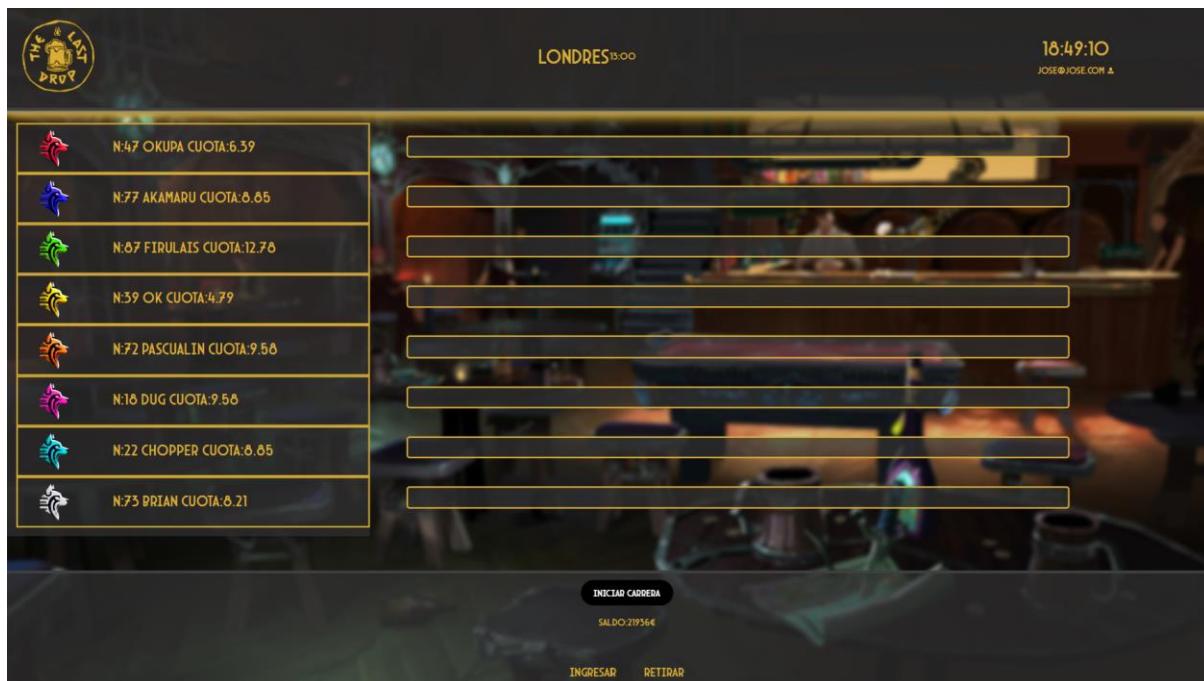
JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

Los distintos elementos de la página, como header, footer etc. presentan un fondo negro con cierta opacidad

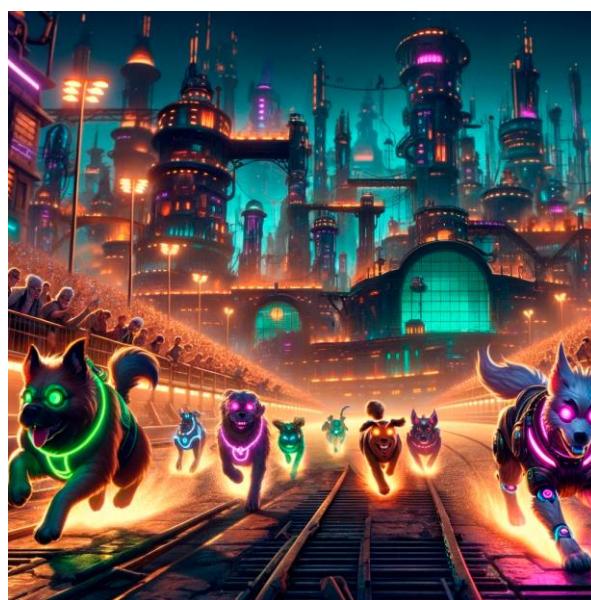
-Diseño homogéneo



JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS



-Temática de imágenes y fondos



JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS



-Iconografía

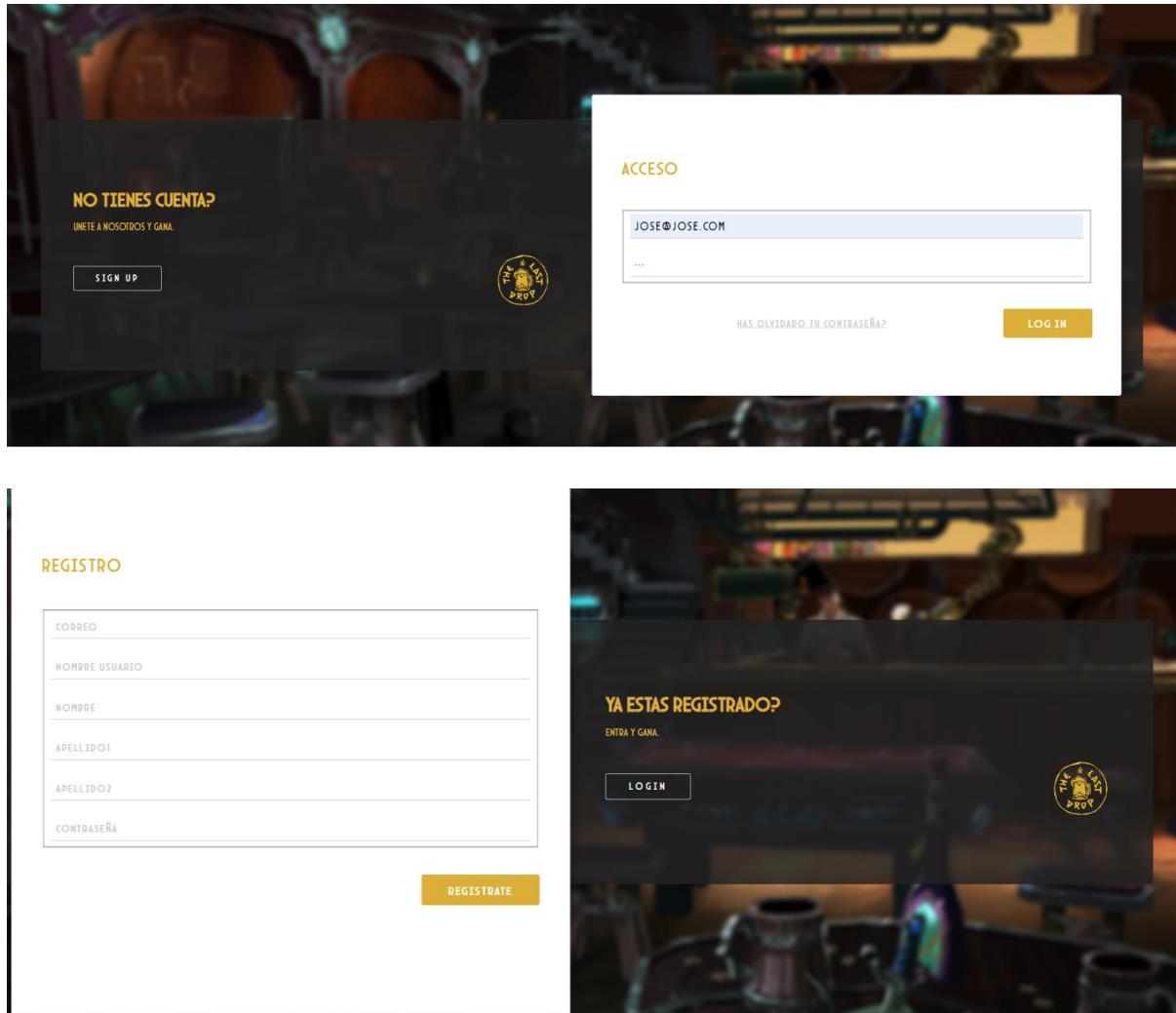


JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

La mayoría de imágenes han sido generadas con la ayuda de herramientas de inteligencia artificial. Aun así, se han utilizado herramientas de edición de imágenes tanto online (www.remove.bg) como de escritorio (Adobe Photoshop/Illustrator).

Para el login y el menú de selección de juego del lobby hemos buscado en páginas como CodePen para adaptarlas a nuestro proyecto. Esta adaptación se ha realizado tanto con la modificación de archivos CSS como JS.

En el login tenemos 2 opciones, una para acceder a la aplicación y otra para registrarse, el cual aparece cuando le das a un botón u a otro.



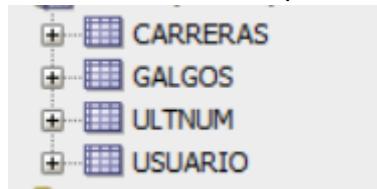
Para el apartado responsive consultar el punto 6 de este documento.

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

Para finalizar en el tintero se nos quedaron ideas como la ampliación de añadir un circuito que se actualizará al momento en el juego de los galgos y el añadido de un modo claro y oscuro.

3. BASE DE DATOS

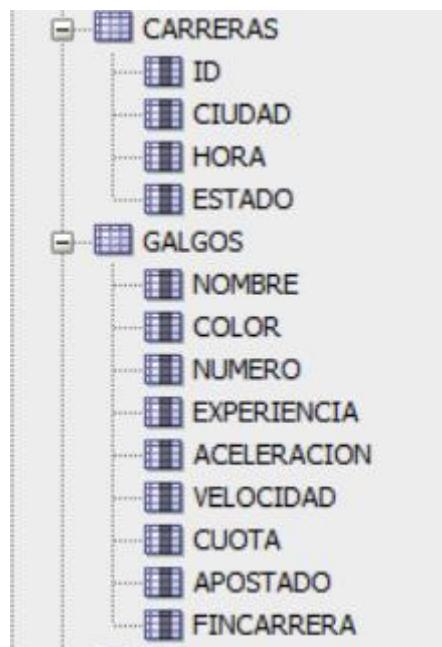
Las tablas de la base de datos creadas para nuestro proyecto son



Para la gestión del usuario tenemos la tabla usuario donde guardamos la información referente a su saldo, así como el saldo del mismo:

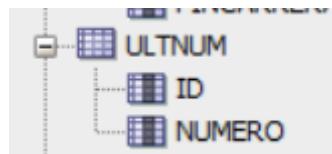


Para los galgos utilizamos las tablas galgos y carreras donde se guarda la información relevante de ambos



En la tabla UltNum guardamos la lista de las tiradas que se realizan de la ruleta. Se guarda un ID que se aumenta automáticamente en los controladores y el número que ha salido.

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS



Para guardar automáticamente el ID hemos creado esta secuencia (empezamos con 11 debido a los datos que se insertan al crear la base de datos con el documento SQL).

```
CREATE SEQUENCE SEQ_ULTNUM
START WITH 11
INCREMENT BY 1
MAXVALUE 99999999999999999999
MINVALUE 1
NOCACHE
NOCYCLE;
```

4. SPRING

Lo primero que creamos en nuestro proyecto fueron las entidades que íbamos a usar. En este caso tenemos 4.

```
package Jose_Antonio.transversal.domain.entity;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Table;

@Entity
@Table(name="CARRERAS")
public class CarreraEntity {

    @Id
    @Column(name="ID")
    private Integer id;

    @Column(name="CIUDAD")
    private String ciudad;

    @Column(name="HORA")
    private String hora;

    @Column(name="ESTADO")
    private String estado;
```

Entidad carreras

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

```
package Jose_Antonio.transversal.domain.entity;

import java.io.Serializable;

@Entity
@Table(name="GALGOS")
public class GalgoEntity implements Serializable {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    @Id
    @Column(name="NOMBRE")
    private String nombre;

    @Column(name="COLOR")
    private String color;

    @Column(name="NUMERO")
    private Integer numero;

    @Column(name="EXPERIENCIA")
    private Integer experiencia;

    @Column(name="ACELERACION")
    private Integer aceleracion;

    @Column(name="VELOCIDAD")
    private Integer velocidad;

    @Column(name="CUOTA")
    private Double cuota;

    @Column(name="APOSTADO")
    private Double apostado;

    @Column(name="FINCARRERA")
    private Boolean finCarrera;
```

Entidad Galgos

Tanto las carreras como los galgos pensamos en relacionarlos desde un principio, pero dado como está estructurada la práctica, no nos ha dado tiempo a construir una relación funcional entre ellos, por lo que, aunque se utilizan tanto las carreras como los galgos guardados en base de datos, la asignación de los galgos en las distintas carreras se hacen desde la parte del cliente

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

```
package Jose_Antonio.transversal.domain.entity;

import java.io.Serializable;

@Entity
@Table(name = "ULTNUM")
public class UltNumEntity implements Serializable{

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    @GeneratedValue(strategy = GenerationType.SEQUENCE,generator = "secuencia")
    @SequenceGenerator(name = "secuencia",allocationSize = 1,sequenceName = "SEQ_ULTNUM")
    @Id
    @Column(name="ID")
    private Integer id;

    @Column(name="NUMERO")
    private String numero;
```

Entidad Últimos Números.

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

```
1 package Jose_Antonio.transversal.domain.entity;
2
3 import java.io.Serializable;
4
5 @Entity
6 @Table(name="USUARIO")
7 public class UsuarioEntity implements Serializable {
8
9     /**
10      *
11      */
12     private static final long serialVersionUID = 1L;
13
14     @Id
15     @Column(name="CORREO")
16     private String correo;
17
18     @Column(name="NOMBRE_USUARIO")
19     private String nombre_usuario;
20
21     @Column(name="CONTRASENA")
22     private String contrasena;
23
24     @Column(name="NOMBRE_REAL")
25     private String nombre_real;
26
27     @Column(name="APELLIDO1")
28     private String apellido1;
29
30     @Column(name="APELLIDO2")
31     private String apellido2;
32
33     @Column(name="SALDO")
34     private Integer saldo;
```

Entidad Usuarios

Para los últimos números hemos usado una secuencia guardada en base de datos para la creación automática de estos sin necesidad de proporcionar un ID.

El siguiente punto fueron las interfaces de los DAO. Como se pedía 50% CRUD y 50% a mano, tenemos dos con CRUD y dos sin ellas. Las de CRUD son las siguientes.

```
1 package Jose_Antonio.transversal.dao;
2
3 import org.springframework.data.repository.CrudRepository;
4
5
6
7 public interface IDAOUsuario extends CrudRepository<UsuarioEntity, String> {
8
9 }
```

Interfaz DAO usuario

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

```
1 package Jose_Antonio.transversal.dao;
2
3+ import org.springframework.data.repository.CrudRepository;[]
4
5 public interface IDAOGalgos extends CrudRepository<GalgoEntity, String> {
6
7 }
8
9 }
```

Interfaz DAO Galgos

```
1 package Jose_Antonio.transversal.dao;
2
3+ import java.util.List;[]
4
5
6 public interface IDAOCarreras{
7
8     public void merge(CarreraEntity carrera);
9     public CarreraEntity findById(Integer id);
10    public List<CarreraEntity> findAll();
11
12 }
13
14
```

Interfaz DAO Carreras

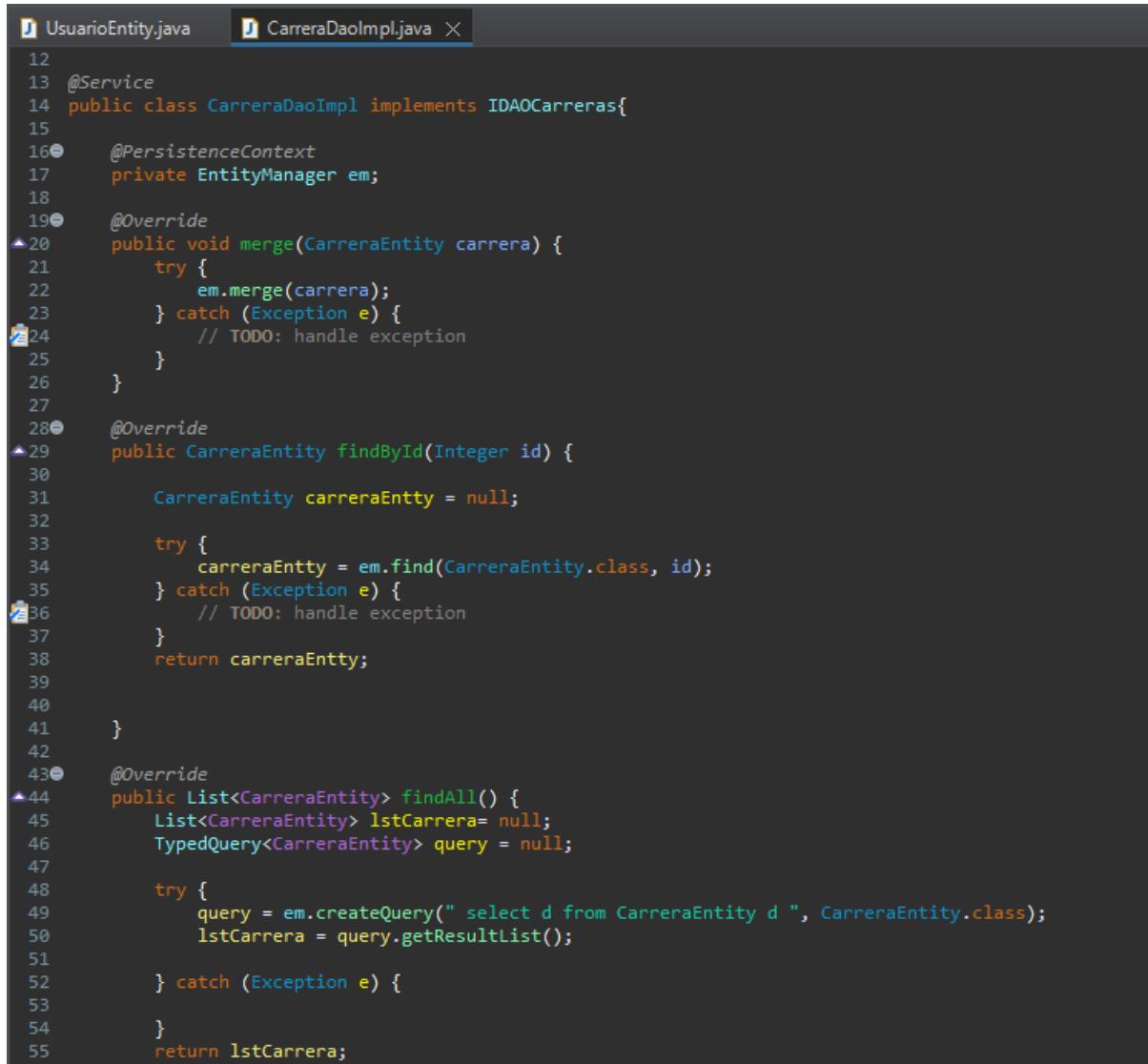
```
1 package Jose_Antonio.transversal.dao;
2
3+ import java.util.List;[]
4
5
6 public interface IDAOUltNum{
7
8     public void save(UltNumEntity numero);
9     public void remove(UltNumEntity numero);
10    public List<UltNumEntity> findAll();
11    public List<String> masRepetidos();
12    public List<String> menosRepetidos();
13
14 }
```

Interfaz DAO Últimos Números

En esta última implementación explicaremos más adelante que son los métodos más repetidos y menos repetidos.

El siguiente paso eran las implementaciones de los DAO para las dos interfaces que no tenían CRUD repository.

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS



```
1 UsuarioEntity.java
2 CarreraDaoImpl.java ×
3
4
5
6
7
8
9
10
11
12
13 @Service
14 public class CarreraDaoImpl implements IDAOcarreras{
15
16     @PersistenceContext
17     private EntityManager em;
18
19     @Override
20     public void merge(CarreraEntity carrera) {
21         try {
22             em.merge(carrera);
23         } catch (Exception e) {
24             // TODO: handle exception
25         }
26     }
27
28     @Override
29     public CarreraEntity findById(Integer id) {
30
31         CarreraEntity carreraEntty = null;
32
33         try {
34             carreraEntty = em.find(CarreraEntity.class, id);
35         } catch (Exception e) {
36             // TODO: handle exception
37         }
38         return carreraEntty;
39
40     }
41
42
43     @Override
44     public List<CarreraEntity> findAll() {
45         List<CarreraEntity> lstCarrera= null;
46         TypedQuery<CarreraEntity> query = null;
47
48         try {
49             query = em.createQuery(" select d from CarreraEntity d ", CarreraEntity.class);
50             lstCarrera = query.getResultList();
51
52         } catch (Exception e) {
53
54         }
55         return lstCarrera;
56     }
57 }
```

Implementación DAO Carreras

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

```
package Jose_Antonio.transversal.dao.impl;

import java.util.List;

@Service
public class UltNumDaoImpl implements IDAOUltNum{

    @PersistenceContext
    private EntityManager em;

    @Override
    public void save(UltNumEntity numero) {
        try {
            em.persist(numero);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    @Override
    public void remove(UltNumEntity numero) {
        // TODO Auto-generated method stub
    }

    @Override
    public List<UltNumEntity> findAll() {
        List<UltNumEntity> lstNumeros= null;
        TypedQuery<UltNumEntity> query = null;

        try {
            query = em.createQuery(" select d from UltNumEntity d ", UltNumEntity.class);
            lstNumeros = query.getResultList();
        } catch (Exception e) {

        }
        return lstNumeros;
    }
}
```

Implementación de métodos estándar de Últimos Números

Como he dicho anteriormente, tenemos dos métodos diferentes en últimos números los cuales a través de una consulta JPA nos saca los 5 números más y menos repetidos.

```
@Override
public List<String> masRepetidos() {
    List<String> lstNumeros= null;
    TypedQuery<String> query = null;

    try {
        query = em.createQuery("SELECT u.numero FROM UltNumEntity u GROUP BY u.numero ORDER BY COUNT(u) DESC", String.class);
        query.setMaxResults(5);
        lstNumeros = query.getResultList();
        System.out.println("asdf"+lstNumeros.get(0));
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    return lstNumeros;
}

@Override
public List<String> menosRepetidos() {
    List<String> lstNumeros= null;
    TypedQuery<String> query = null;

    try {
        query = em.createQuery("SELECT u.numero FROM UltNumEntity u GROUP BY u.numero ORDER BY COUNT(u) ASC", String.class);
        query.setMaxResults(5);
        lstNumeros = query.getResultList();
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    return lstNumeros;
}
```

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

Además, el tipo de objeto TypedQuery nos permite acotar los resultados a los primeros 5 con lo que nos soluciona este problema.

El siguiente paso fue crear las DTOs.

```
package Jose_Antonio.transversal.domain.dto;

import java.util.List;

public class CarreraDTO {

    private Integer id;

    private String ciudad;

    private String hora;

    private String estado;

    public CarreraDTO() {
        super();
    }
}
```

DTO de Carrera

```
package Jose_Antonio.transversal.domain.dto;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

public class GalgoDTO {

    private String nombre;

    private String color;

    private Integer numero;

    private Integer experiencia;

    private Integer aceleracion;

    private Integer velocidad;

    private Double cuota;

    private Double apostado;

    private Boolean finCarrera;
}
```

DTO de Galgo

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

```
1 package Jose_Antonio.transversal.domain.dto;  
2  
3 import java.io.Serializable;  
4  
5 public class UltNumDTO{  
6  
7     private Integer id;  
8     private String numero;
```

DTO de Últimos Números

```
1 package Jose_Antonio.transversal.domain.dto;  
2  
3+ import java.io.Serializable;  
4  
5 public class UsuarioDTO implements Serializable {  
6  
7     /**  
8      *  
9      */  
10    private static final long serialVersionUID = 1L;  
11  
12    @NotEmpty  
13    @Email  
14    private String correo;  
15    @NotEmpty  
16    private String nombre_usuario;  
17    @NotEmpty  
18    private String contrasena;  
19  
20    @NotEmpty  
21    private String nombre_real;  
22  
23    @NotEmpty  
24    private String apellido1;  
25  
26    @NotEmpty  
27    private String apellido2;  
28  
29    private Integer saldo;
```

DTO de Usuario

El siguiente paso fue la creación del interfaz de los servicios.

```
1 package Jose_Antonio.transversal.services;  
2  
3+ import java.util.List;  
4  
5 public interface IServiceCarrera {  
6  
7     public void merge(CarreraDTO carrera);  
8     public CarreraDTO findById(int id);  
9     public List<CarreraDTO> EncontrarTodasCarreras();  
10 }  
11
```

Interfaz de servicios de Carreras

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

```
1 package Jose_Antonio.transversal.services;
2
3+ import java.util.List;
4
5
6 public interface IServiceGalgo {
7
8     public void save(GalgoDTO galgo);
9     public void merge(GalgoDTO galgo);
10    public List<GalgoDTO> TodosGalgos();
11
12 }
13
```

Interfaz de servicio de Galgos

```
package Jose_Antonio.transversal.services;
+ import java.util.List;
+
public interface IServiceUltNum {
    public void save(UltNumDTO ultNum);
    public void merge(UltNumDTO ultNum);
    public List<UltNumDTO> UltNum();
    public List<UltNumDTO> masRepetidos();
    public List<UltNumDTO> menosRepetidos();
}
```

Interfaz de servicio de Últimos Números

```
package Jose_Antonio.transversal.services;
+ import java.util.List;
+
public interface IServiceUsuario {
    public void save(UsuarioDTO usuario);
    public void merge(UsuarioDTO usuario);
    public void remove(String email);
    public UsuarioDTO findById(String email);
    public List<UsuarioDTO> findAll();
}
```

Interfaz de servicios de Usuarios

A continuación, la implementación de estos servicios.

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

```
@Service
public class ServiceCarreraImp implements IServiceCarrera{
    @Autowired
    private IDAOcarreras dao;

    @Override
    @Transactional
    public void merge(CarreraDTO carrera) {
        CarreraEntity carreraEntty = new CarreraEntity();

        try {
            carreraEntty.setId(carrera.getId());
            carreraEntty.setCiudad(carrera.getCiudad());
            carreraEntty.setEstado(carrera.getEstado());
            carreraEntty.setHora(carrera.getHora());

            dao.merge(carreraEntty);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    @Override
    public CarreraDTO findById(int id) {
        CarreraDTO carreraDTO= null;
        CarreraEntity carreraEntty=new CarreraEntity();
        try {
            carreraEntty= dao.findById(id);
            System.out.println("En service carrera"+carreraEntty.getId());
            if(carreraEntty!=null) {
                carreraDTO= new CarreraDTO();

                carreraDTO.setId(carreraEntty.getId());
                carreraDTO.setCiudad(carreraEntty.getCiudad());
                carreraDTO.setEstado(carreraEntty.getEstado());
                carreraDTO.setHora(carreraEntty.getHora());
            }
        }

        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
```

Primera parte de la implementación del servicio de carreras.

```
@Override
@Transactional(readOnly = true)
public List<CarreraDTO> EncontarTodasCarreras() {
    List<CarreraDTO> lista = new ArrayList<>();
    Iterable<CarreraEntity> listaE = null;

    try {
        listaE = dao.findAll();
        for(Iterator<CarreraEntity> iterator = listaE.iterator(); iterator.hasNext(); ) {
            CarreraEntity entidad = (CarreraEntity) iterator.next();
            CarreraDTO carrera = new CarreraDTO();
            carrera.setId(entidad.getId());
            carrera.setCiudad(entidad.getCiudad());
            carrera.setHora(entidad.getHora());
            carrera.setEstado(entidad.getEstado());
            lista.add(carrera);
        }
    }

    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    return lista;
}
```

Segunda parte de la implementación del servicio de carreras

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

```
@Service
public class ServiceGalgoImp implements IServiceGalgo {

    @Autowired
    private IDAOGalgos galgoDao;

    @Override
    public void save(GalgoDTO galgo) {
        // TODO Auto-generated method stub
    }

    @Override
    public void merge(GalgoDTO galgo) {
        // TODO Auto-generated method stub
    }

    @Override
    public List<GalgoDTO> TodosGalgos() {
        List<GalgoDTO> lstGalgos = null;
        Iterable<GalgoEntity> lstEntty = null;

        try {
            lstGalgos = new ArrayList<>();
            lstEntty = galgoDao.findAll();

            for (Iterator<GalgoEntity> iterator = lstEntty.iterator(); iterator.hasNext();) {
                GalgoEntity galgosEntity = (GalgoEntity) iterator.next();
                GalgoDTO galgo = new GalgoDTO();

                galgo.setNombre(galgosEntity.getNombre());
                galgo.setColor(galgosEntity.getColor());
                galgo.setNumero(galgosEntity.getNumero());
                galgo.setExperiencia(galgosEntity.getExperiencia());
                galgo.setAceleracion(galgosEntity.getAceleracion());
                galgo.setVelocidad(galgosEntity.getVelocidad());
                galgo.setCuota(galgosEntity.getCuota());
                galgo.setApostado(galgosEntity.getApostado());
                galgo.setFinCarrera(galgosEntity.getFinCarrera());

                lstGalgos.add(galgo);
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Implementación del servicio de galgos

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

```
@Service
public class ServiceUltNumImp implements IServiceUltNum{

    @Autowired
    private IDAOUltNum dao;

    @Override
    @Transactional
    public void save(UltNumDTO ultNum) {
        UltNumEntity entidad = new UltNumEntity();

        try {
            entidad.setNumero(ultNum.getNumero());
            dao.save(entidad);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    @Override
    @Transactional
    public void merge(UltNumDTO ultNum) {
        try {
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    @Override
    @Transactional(readOnly = true)
    public List<UltNumDTO> UltNum() {
        List<UltNumDTO> lista = new ArrayList<>();
        Iterable<UltNumEntity> listaE = null;

        try {
            listaE = dao.findAll();
            for(Iterator<UltNumEntity> iterator = listaE.iterator(); iterator.hasNext(); ) {
                UltNumEntity entidad = (UltNumEntity) iterator.next();
                UltNumDTO ult = new UltNumDTO();
                ult.setId(entidad.getId());
                ult.setNumero(entidad.getNumero());
                lista.add(ult);
            }
            System.out.println(lista.size());
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Primera parte de la implementación del servicio de últimos números

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

```
● @Override
public List<UltNumDTO> masRepetidos() {
    List<UltNumDTO> lista = new ArrayList<>();
    List<String> listaE = null;

    try {
        listaE = dao.masRepetidos();
        for(int i = 0;i<listaE.size();i++){
            UltNumDTO ult= new UltNumDTO();
            ult.setNumero(listaE.get(i));
            lista.add(ult);

        }

        System.out.println("mas repes servicio"+lista.size());
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    return lista;
}

● @Override
public List<UltNumDTO> menosRepetidos() {
    List<UltNumDTO> lista = new ArrayList<>();
    List<String> listaE = null;

    try {
        listaE = dao.menosRepetidos();
        for(int i = 0;i<listaE.size();i++){
            UltNumDTO ult= new UltNumDTO();
            ult.setNumero(listaE.get(i));
            lista.add(ult);

        }

        System.out.println("mas repes servicio"+lista.size());
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    return lista;
}
```

Segunda parte de la implementación del servicio de últimos números.

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

```
@Service
public class ServiceUsuarioImp implements IServiceUsuario {

    @Autowired
    private IDAOUsuario usuarioDao;

    @Override
    @Transactional
    public void save(UsuarioDTO usuario) {
        UsuarioEntity usuarioEntty = new UsuarioEntity();

        usuarioEntty.setCorreo(usuario.getCorreo());
        usuarioEntty.setNombre_usuario(usuario.getNombre_usuario());
        usuarioEntty.setNombre_real(usuario.getNombre_real());
        usuarioEntty.setApellido1(usuario.getApellido1());
        usuarioEntty.setApellido2(usuario.getApellido2());
        usuarioEntty.setContrasena(usuario.getContrasena());
        if(usuario.getSaldo()==null) {
            usuarioEntty.setSaldo(0);
        }else {
            usuarioEntty.setSaldo(usuario.getSaldo());
        }
        usuarioDao.save(usuarioEntty);
    }

    @Override
    @Transactional
    public void merge(UsuarioDTO usuario) {
    }

    @Override
    @Transactional
    public void remove(String email) {
    }
}
```

Primera parte de implementación del servicio de usuarios

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

```
    @Override
    @Transactional(readOnly = true)
    public UsuarioDTO findById(String email) {
        UsuarioDTO usuarioDto= null;
        UsuarioEntity usuarioEntty=new UsuarioEntity();

        usuarioEntty=usuarioDao.findById(email).orElse(null);
        try {
            if(usuarioEntty!=null) {
                usuarioDto= new UsuarioDTO();
                usuarioDto.setApellido1(usuarioEntty.getApellido1());
                usuarioDto.setApellido2(usuarioEntty.getApellido2());
                usuarioDto.setContrasena(usuarioEntty.getContrasena());
                usuarioDto.setCorreo(usuarioEntty.getCorreo());
                usuarioDto.setNombre_real(usuarioEntty.getNombre_real());
                usuarioDto.setNombre_usuario(usuarioEntty.getNombre_usuario());
                usuarioDto.setSaldo(usuarioEntty.getSaldo());
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }

        return usuarioDto;
    }

    @Override
    @Transactional(readOnly = true)
    public List<UsuarioDTO> findAll() {
        List<UsuarioDTO> listaDto = new ArrayList<>();

        return listaDto;
    }
}
```

Segunda parte de la implementación de servicios de usuario

Por ultimo y una de las partes más importantes fue hacer los controladores.

En los controladores de index, lobby, ruleta y galgos usamos una variable global que nos sirve para poder controlar que usuario tiene la sesión iniciada en ese momento.

En este caso empezamos con el index controller que contiene el inicio de sesión.

En este controller tenemos recuperar valores que nos sirve para la creación de nuevos usuarios. Como podemos observar tenemos validaciones para los formularios.

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

```
@Controller
public class IndexController {

    @Autowired
    private IServiceUsuario servicioUsuario;

    private String registrado = null;
    @GetMapping(value= {"","/"})
    public String inicio(Model modelo) {
        modelo.addAttribute("usuario", new UsuarioDTO ());
        return "login";
    }

    @PostMapping(value = "/recuperarValores")
    public String recuperarValores(Model modelo,@Valid @ModelAttribute(value = "usuario") final UsuarioDTO usuario, BindingResult result) {

        String forward = "";
        Map<String, String> errors = null;
        try {
            if (result.hasErrors()) {
                modelo.addAttribute("errores", errors);
                forward="/login";
            } else {
                servicioUsuario.save(usuario);
                forward="/login";
            }
        }catch (Exception e) {
            System.out.println(e.getMessage());
        }

        return forward;
    }
}
```

Por último, tenemos el método del login que en caso de que sea correcto nos llevara al lobby y en caso incorrecto nos llevara al login de nuevo con un mensaje de error.

```
@PostMapping(value="/login")
public String login(Model modelo,final UsuarioDTO usuario) {
    String forward="login";

    try {
        UsuarioDTO usuarioRegistrado=servicioUsuario.findById(usuario.getCorreo());
        if(null!=usuarioRegistrado) {
            if(usuarioRegistrado.getContrasena().equals(usuario.getContrasena())) {
                registrado = usuarioRegistrado.getCorreo();
                modelo.addAttribute("correo",registrado);
                modelo.addAttribute("fichitas",usuarioRegistrado.getSaldo());
                forward="lobby";
            }else {
                modelo.addAttribute("error", "Usuario o contraseña incorrecto");
                System.out.println("error");
                forward="login";
            }
        }else {
            modelo.addAttribute("error", "Usuario o contraseña incorrecto");
            System.out.println("error");
            forward="login";
        }
    } catch (Exception e) {
        System.out.println(e.getMessage());
    };
}
```

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

El siguiente controller que tenemos es el de lobby que tiene tres métodos en los que tenemos redirecciones a los distintos juegos y al propio lobby para cuando dentro de un juego queramos volver al menú principal. A través de la variable global sacamos el saldo para las vistas además del propio correo.

```
package Jose_Antonio.transversal.controller;

import org.springframework.beans.factory.annotation.Autowired;

@Controller
@RequestMapping(value="/lobby")
public class LobbyController {

    @Autowired
    private IServiceUsuario servicioUsuario;
    private String registrado = "";

    @GetMapping(value="")
    public String volver(HttpServletRequest request, Model modelo) {
        System.out.println(registrado);
        UsuarioDTO usuarioRegistrado=servicioUsuario.findById(registrado);
        modelo.addAttribute("correo",registrado);
        modelo.addAttribute("fichitas",usuarioRegistrado.getSaldo());

        return "lobby";
    }

    @GetMapping(value="ruleta")
    public String ruleta(HttpServletRequest request, Model modelo) {
        registrado = request.getParameter("correo");
        System.out.println(registrado);
        UsuarioDTO usuarioRegistrado=servicioUsuario.findById(registrado);
        modelo.addAttribute("correo",registrado);
        modelo.addAttribute("fichitas",usuarioRegistrado.getSaldo());

        return "lobby/ruleta";
    }

    @GetMapping(value="galgos")
    public String galgo(HttpServletRequest request, Model modelo) {
        registrado = request.getParameter("correo");
        System.out.println(registrado);
        UsuarioDTO usuarioRegistrado=servicioUsuario.findById(registrado);
        modelo.addAttribute("correo",registrado);
        modelo.addAttribute("fichitas",usuarioRegistrado.getSaldo());

        return "lobby/galgos";
    }

}
```

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

El siguiente controller fue el del saldo. En este hacemos tanto la actualización del saldo de ingreso como el de retirada del mismo que obtenemos a través del requestbody.

```
package Jose_Antonio.transversal.controller;

import org.springframework.beans.factory.annotation.Autowired;

@Controller
@RequestMapping(value="/dinero")
public class SaldoController {

    @Autowired
    private IServiceusuario servicioUsuario;

    @PostMapping(value="/ingresar")
    @ResponseBody
    public Integer ingresar(@RequestBody UsuarioDTO cliente) {

        UsuarioDTO user =servicioUsuario.findById(cliente.getCorreo());
        int saldoNuevo=cliente.getSaldo();
        user.setSaldo(saldoNuevo);

        servicioUsuario.save(user);

        return saldoNuevo;
    }

    @PostMapping(value="/retirar")
    @ResponseBody
    public Integer retirar(@RequestBody UsuarioDTO cliente) {

        UsuarioDTO user =servicioUsuario.findById(cliente.getCorreo());
        int saldoNuevo=0;
        user.setSaldo(saldoNuevo);

        servicioUsuario.save(user);

        return saldoNuevo;
    }
}
```

El siguiente controller es el de la ruleta. Tenemos cuatro métodos. El primero de todos nos devuelve una lista con los últimos números (todos los números de las tiradas en bases de datos). El siguiente método es para guardar las tiradas que se realizan.

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

```
@Controller
@RequestMapping(value="/ruleta")
public class RuletaController {

    @Autowired
    private IServiceUltNum servicio;

    @GetMapping(value="/ultimos")
    @ResponseBody
    public List<UltNumDTO> ultimos(){
        List<UltNumDTO> lista = new ArrayList<>();
        try {
            lista = servicio.UltNum();

            // Ordena la lista de mayor a menor por el ID
            Collections.sort(lista, new Comparator<UltNumDTO>() {
                @Override
                public int compare(UltNumDTO o1, UltNumDTO o2) {
                    // Compara los IDs, de o2 a o1 para orden descendente
                    // EL RETURN NO ES DEL "METODO ULTIMOS" SI NO DEL COMPARATOR
                    return o2.getId().compareTo(o1.getId());
                }
            });
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
        return lista;
    }

    @PostMapping(value="/guardarBola")
    @ResponseBody
    public void guardarBola(@RequestBody UltNumDTO bola){
        try {
            System.out.println(bola.getNumero());
            servicio.save(bola);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Por último, tenemos dos métodos donde se traen de base de datos los 5 números que más han salido en las tiradas y los que menos han salido.

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

```
● @GetMapping(value="/hot")
@ResponseBody
public List<UltNumDTO> hot(){
    List<UltNumDTO> lista = new ArrayList<>();
    try {
        System.out.println("controller hot");
        lista = servicio.masRepetidos();
        System.out.println("controller"+lista.get(0).getNumero());
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    return lista;
}

● @GetMapping(value="/cold")
@ResponseBody
public List<UltNumDTO> cold(){
    List<UltNumDTO> lista = new ArrayList<>();
    try {
        lista = servicio.menosRepetidos();
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
    return lista;
}
```

Usamos un controlador para las carreras

```
@Controller
public class CarreraController {

    ● @Autowired
    private IServiceCarrera servicioCarrera;

    ● @GetMapping(value="/obtenerCarreras")
    @ResponseBody
    public List<CarreraDTO> obtenerCarreras() {
        List<CarreraDTO> listaCarreras= servicioCarrera.EncontarTodasCarreras();

        System.out.println(listaCarreras.size()+listaCarreras.toString());

        return listaCarreras;
    }

    ● @PostMapping(value="/finCarrera")
    @ResponseBody
    public String finCarrera(@RequestBody CarreraDTO carrera) {
        System.out.println("en fincarrera"+ carrera.getId()+carrera.getEstado());
        CarreraDTO carrer =servicioCarrera.findById(carrera.getId());
        String estado=carrera.getEstado();
        carrer.setEstado(estado);
        System.out.println(carrer.getId()+carrer.getEstado());

        servicioCarrera.merge(carrer);

        return estado;
    }
```

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

las cuales tienen un método para poder obtener desde el cliente la lista de las carreras, y otro que se utiliza para que después de una carrera se actualice el estado de la carrera

y el controlador de los galgos solo tienen un método el cual obtenemos en el cliente la lista de los galgos existentes en base de datos

```
@Controller
public class GalgoController {

    @Autowired
    private IServiceGalgo serviciogalgo;

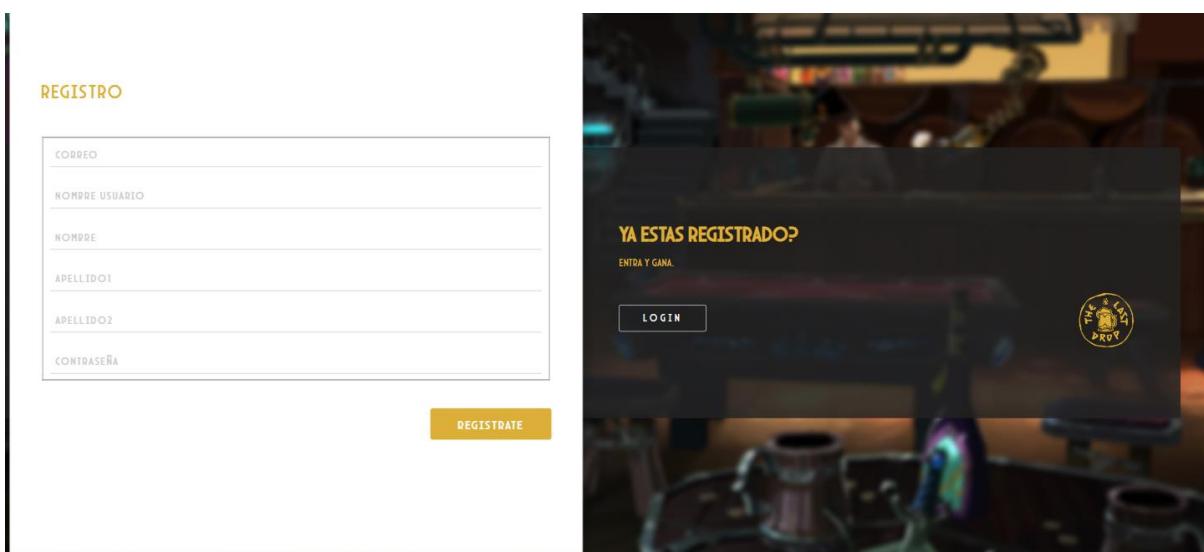
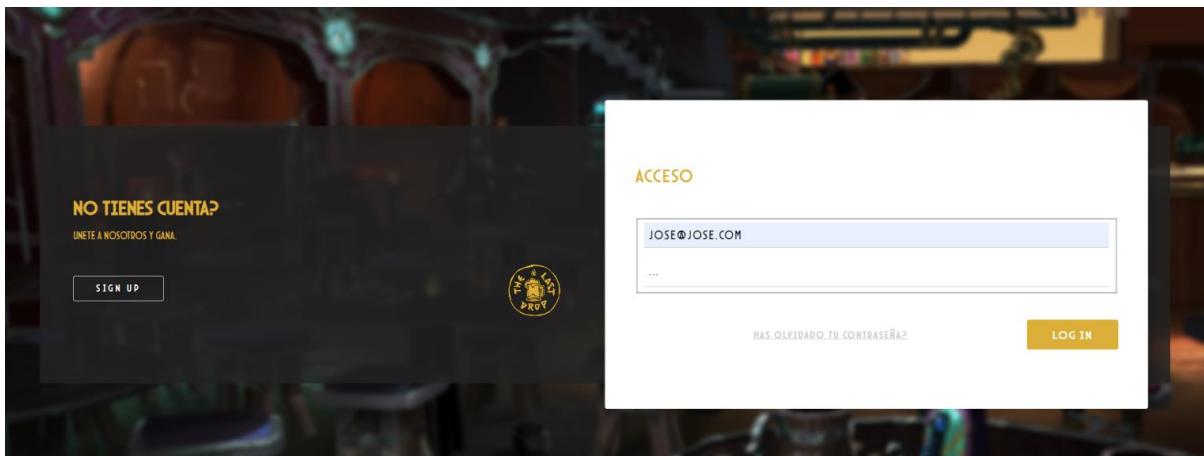
    @GetMapping(value="/obtenerGalgos")
    @ResponseBody
    public List<GalgoDTO> ruleta() {
        List<GalgoDTO> listaGalgos= serviciogalgo.TodosGalgos();

        return listaGalgos;
    }

}
```

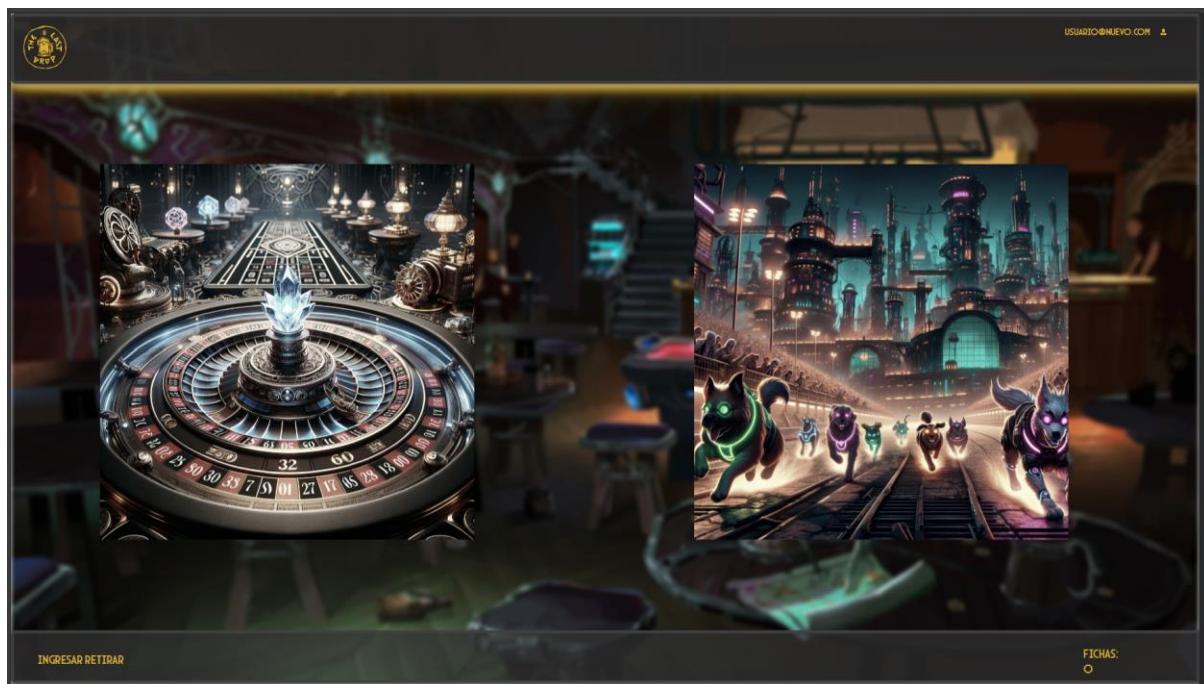
5. VISTAS

Nuestro casino consta de varias vistas, empezamos por la principal que es la vista de registro y acceso, donde el usuario puede introducir sus datos para registrarse ologuearse, esto se realiza mediante una llamada ajax, la cual manda los datos al controlador en spring, el cual realiza una llamada para guardar al usuario, en caso de registro, o una consulta en caso de acceso.

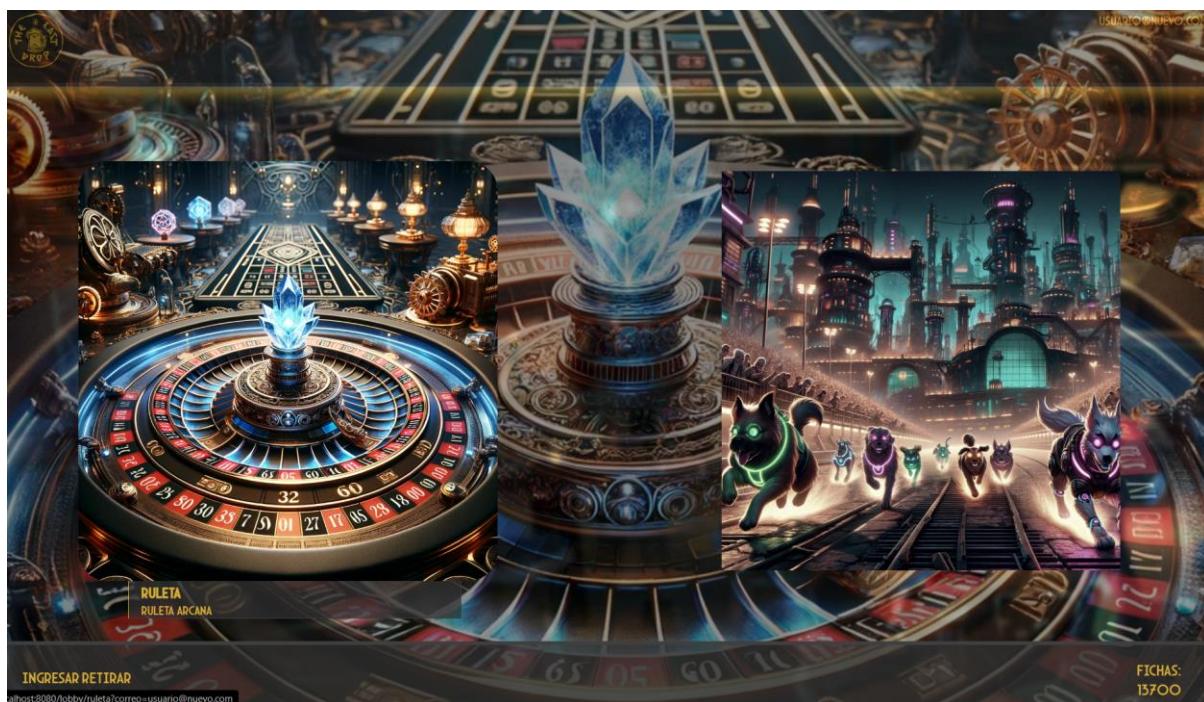


nuestra siguiente visita es el “lobby” donde el usuario puede escoger entre los distintos juegos o ingresar o retirar dinero

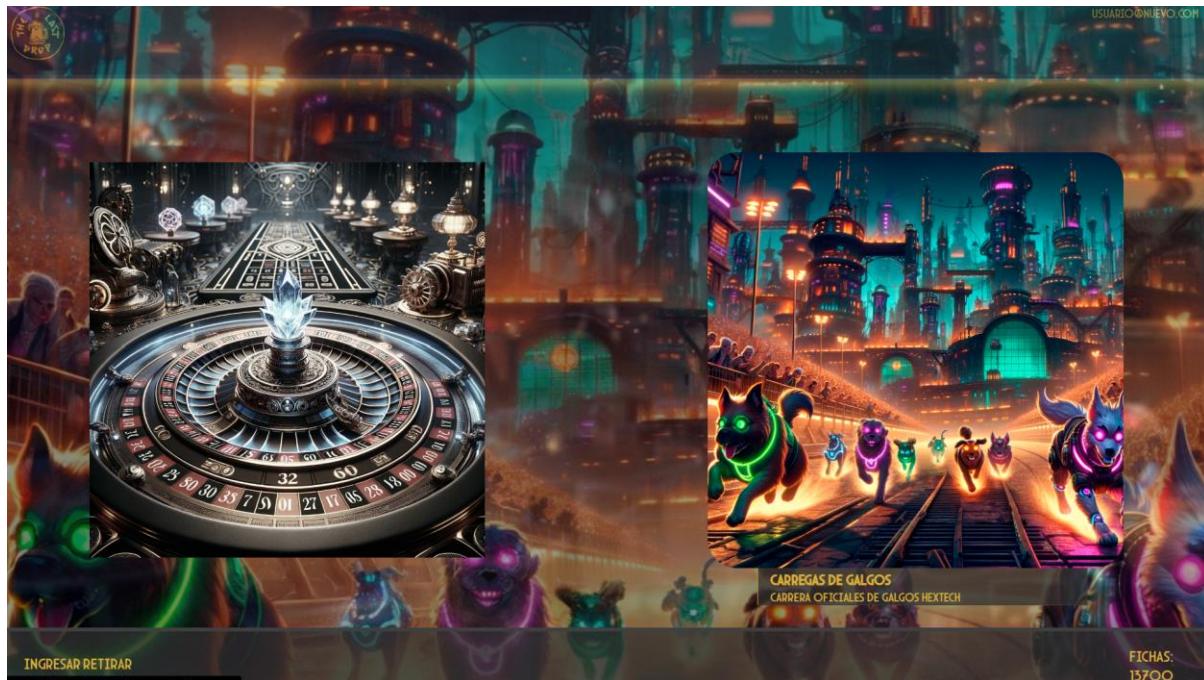
JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS



En esta vista tenemos varios elementos, los principales son los enlaces a los juegos que son las imágenes de los mismos, al pasar el cursor por encima de las imágenes el juego en cuestión se resalta, haciéndose los colores más brillantes y cambiando el fondo con la imagen seleccionada



JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS



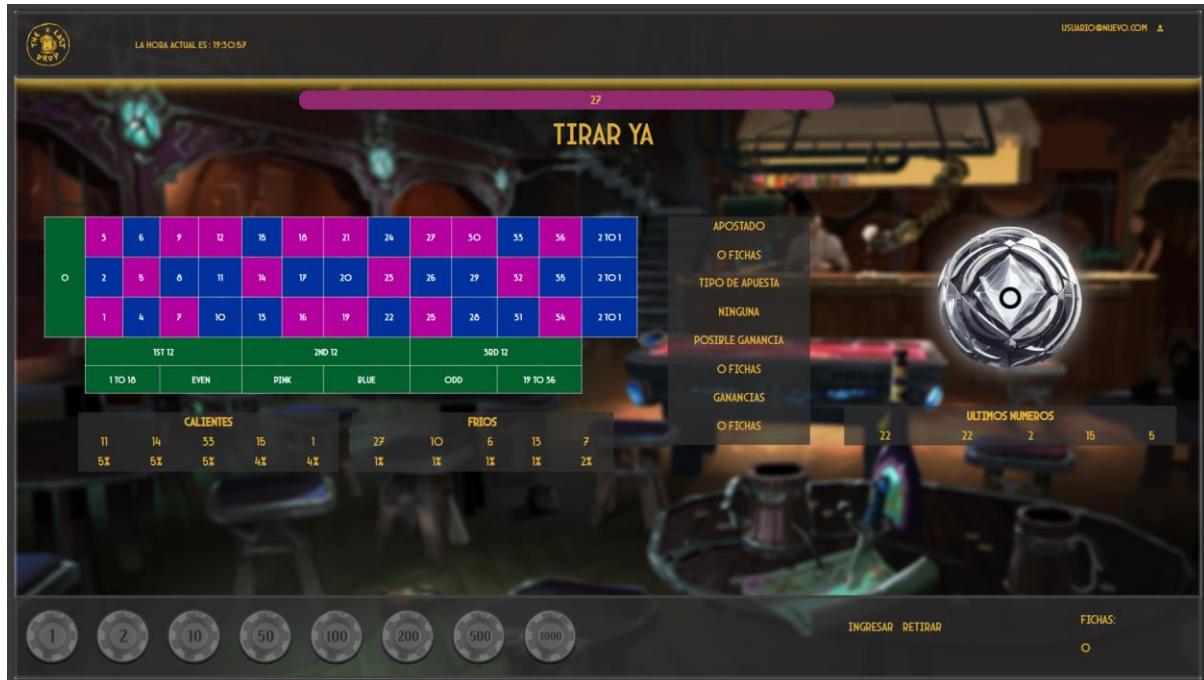
Tanto ingresar como retirar, son funcionalidades recurrentes en nuestro proyecto, ingresar abre un modal el cual podemos elegir la cantidad de fichas a ingresar, el cual al igual que al registro del usuario, se mandan los datos mediante una llamada ajax al controlador, el cual actualiza el saldo del usuario en la base de datos.



Ahora podemos elegir entre la ruleta o los galgos.

La ruleta

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS



Para la ruleta tenemos el mismo sistema para añadir y retirar dinero que tenemos en los galgos y en el lobby.

En este caso para el tema de la inserción de los números tenemos 3 llamadas Ajax diferentes.

Una que saca todos los números y dentro de esta dos que sacan los calientes y fríos y sus porcentajes. Estos se van actualizando con cada tirada.

Esta actualización se da debido a que tenemos otra llamada para insertar una tirada cuando finaliza. Es decir, cuando se da la bola ganadora, esta tirada se guarda en base de datos para poder sacar estadísticas como los Calientes y fríos.

```
function actualizar() {
    $.ajax(
    {
        type: 'GET',
        url: '/ruleta/ultimos',
        success: function(data) {
            listaNum = data;

            document.getElementById("ult1").innerHTML = data[0].numero;
            document.getElementById("ult2").innerHTML = data[1].numero;
            document.getElementById("ult3").innerHTML = data[2].numero;
            document.getElementById("ult4").innerHTML = data[3].numero;
            document.getElementById("ult5").innerHTML = data[4].numero;
        }
    })
}
```

Dentro de esta llamada después de insertar en los elementos de la página los números que pedimos, tenemos las llamadas AJAX para los calientes y fríos

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

```
$.ajax(
{
    type: 'GET',
    url: '/ruleta/hot',
    success: function(data) {

        numID = 1;
        var listCalientes = [];
        listCalientes = data;
        //window.alert(listCalientes[0].numero)
        for (var i = 0; i < listCalientes.length; i++) {
            document.getElementById("cal" + numID).innerHTML = listCalientes[i].numero;
            // document.getElementById("calP" + numID).innerHTML = listCalientes[i].porcentaje + "%";
            numID = numID + 1;
        }
        numID = 1;
        var porcentajes = calcularPorcentajes(listaNum, listCalientes);
        //alert(porcentajes.length)
        for (var i = 0; i < porcentajes.length; i++) {

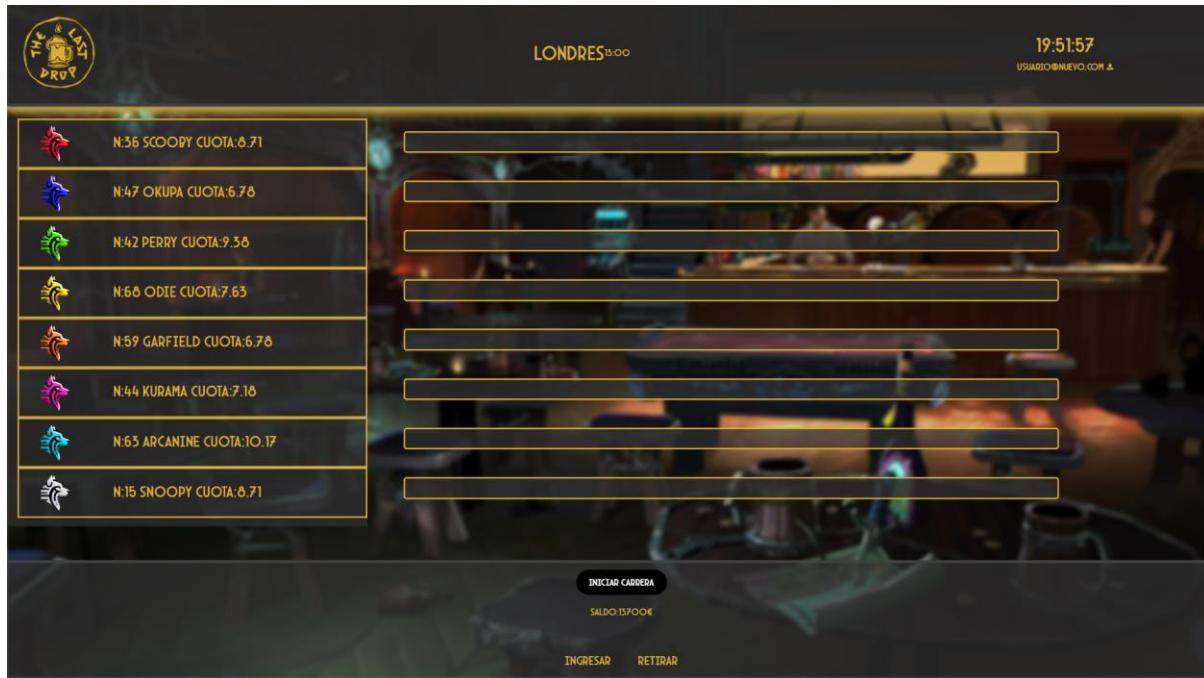
            document.getElementById("calP" + numID).innerHTML = porcentajes[i];
            // document.getElementById("calP" + numID).innerHTML = listCalientes[i].porcentaje + "%";
            numID = numID + 1;
        }
    }
});
```

```
$.ajax(
{
    type: 'GET',
    url: '/ruleta/cold',
    success: function(data) {
        numID = 1;
        var listFrios = [];
        listFrios = data;
        //window.alert(listFrios[0].numero)
        for (var i = 0; i < listFrios.length; i++) {
            document.getElementById("fri" + numID).innerHTML = listFrios[i].numero;
            // document.getElementById("friP" + numID).innerHTML = listFrios[i].porcentaje + "%";
            numID = numID + 1;
        }
        numID = 1;
        var porcentajes = calcularPorcentajes(listaNum, listFrios);
        //alert(porcentajes.length)
        for (var i = 0; i < porcentajes.length; i++) {

            document.getElementById("friP" + numID).innerHTML = porcentajes[i];
            // document.getElementById("calP" + numID).innerHTML = listCalientes[i].porcentaje + "%";
            numID = numID + 1;
        }
    }
});
```

Galgos:

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS



Lo primero que pasa al cargar la página de los galgos son distintas llamadas ajax, una para cargar los galgos y otra para cargar las carreras, una vez cargadas las carreras (id, lugar, hora, y estado), se hace dentro de la misma llamada ajax otra para cargar las carreras

```
$(document).ready(function() { // Asegúrate de que el DOM esté cargado
    $.ajax({
        type: 'GET',
        url: '/obtenerGalgos',
        success: function(listaGalgos) {
            // Procesa la respuesta
            listaGalgos.forEach(function(galgoDto) {
                var galgo = new Galgo(
                    galgoDto.nombre,
                    galgoDto.color,
                    galgoDto.numero,
                    galgoDto.experiencia,
                    galgoDto.aceleracion,
                    galgoDto.velocidad,
                    galgoDto.cuota,
                    galgoDto.apostado,
                    galgoDto.finCarrera
                );
                ListaGalgos.push(galgo);
            });
            controller.obtenerCarreras();
        },
        error: function(error) {
            console.error("Error al obtener los galgos:", error);
        }
    });
});
```

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

Al realizar la primera llamada ajax hemos tenido que transformar nuestra lista de objeto galgo, a otra lista de objetos de galgo

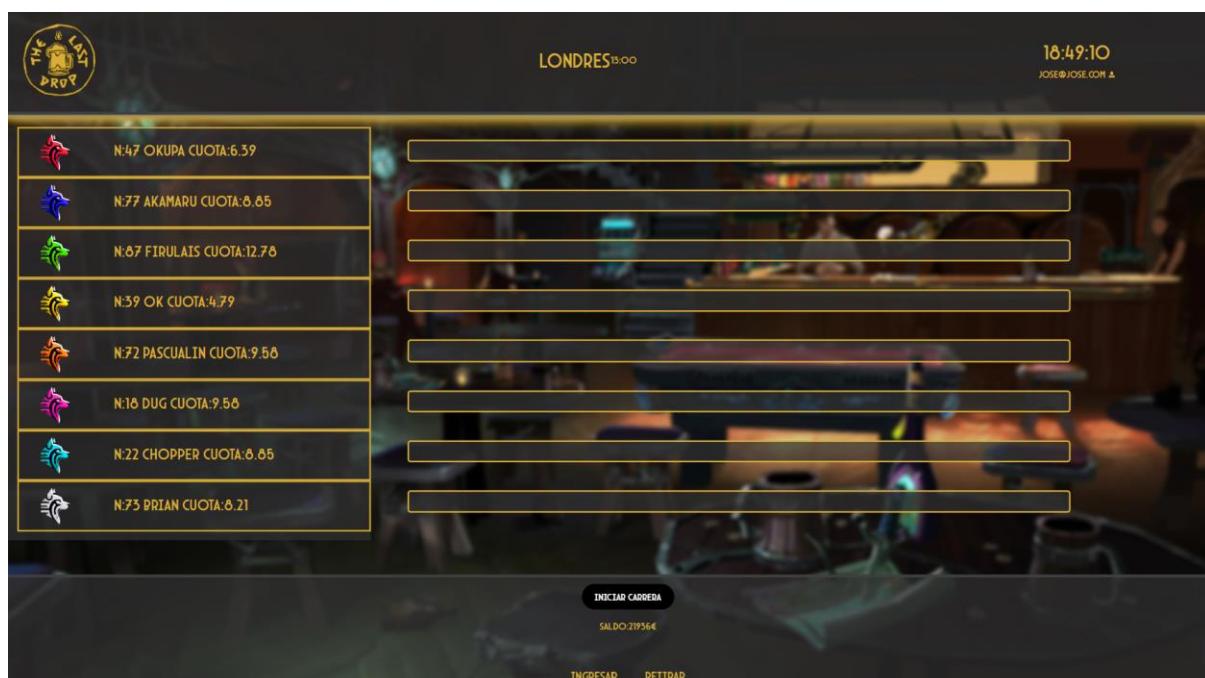
```
obtenerCarreras:function() {  
  
    $.ajax({  
        type: 'GET',  
        url: '/obtenerCarreras',  
        success: function(listaCarreras) {  
  
            listaCarreras.forEach(function(carreraDto) {  
                var carrera = new Carrera(  
                    carreraDto.id,  
                    carreraDto.ciudad,  
                    carreraDto.hora,  
                    [],  
                    [],  
                    carreraDto.estado  
  
                );  
  
                ListaCarreras.push(carrera);  
            });  
  
            controller.anteriores();  
            controller.init();  
        },  
        error: function(error) {  
            console.error("Error al obtener las carreras:", error);  
        }  
    });  
}  
};
```

En la segunda llamada ajax ahora transformamos las carreras traídas desde el controlador en un objeto carrera que utilizamos en el cliente, y una vez realizada la llamada ajax con éxito inicializamos el resto de las funcionalidades

6. ADAPTACIÓN A OTROS DISPOSITIVOS

Hemos decidido crear un punto de ruptura en todas las páginas que se adapten a los distintos tamaños de otros dispositivos. En este caso iPhone XR.

Para la página de los galgos podemos observar que el responsive funciona cuando el móvil está de forma vertical. Podemos observar como en la página los logos de los perros y las cuotas desaparecen para dejar en pantalla solo el nombre y el número del galgo. La cuota es mostrada en un modal al clicar en uno de los galgos.



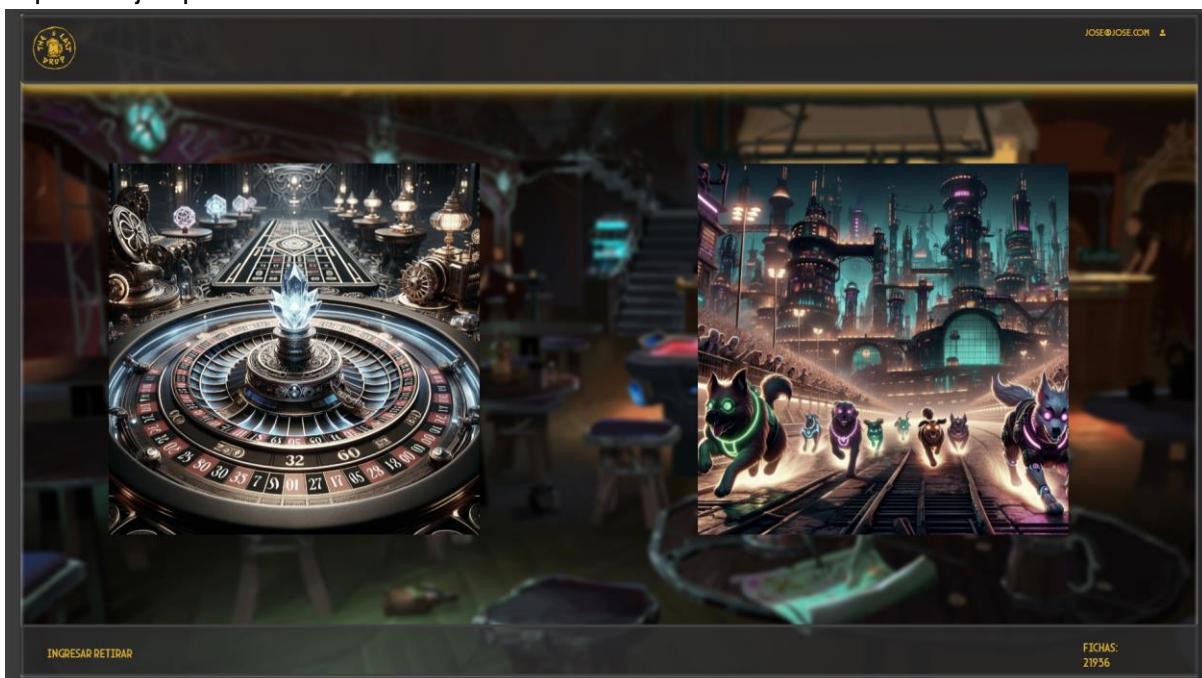
Página normal

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS



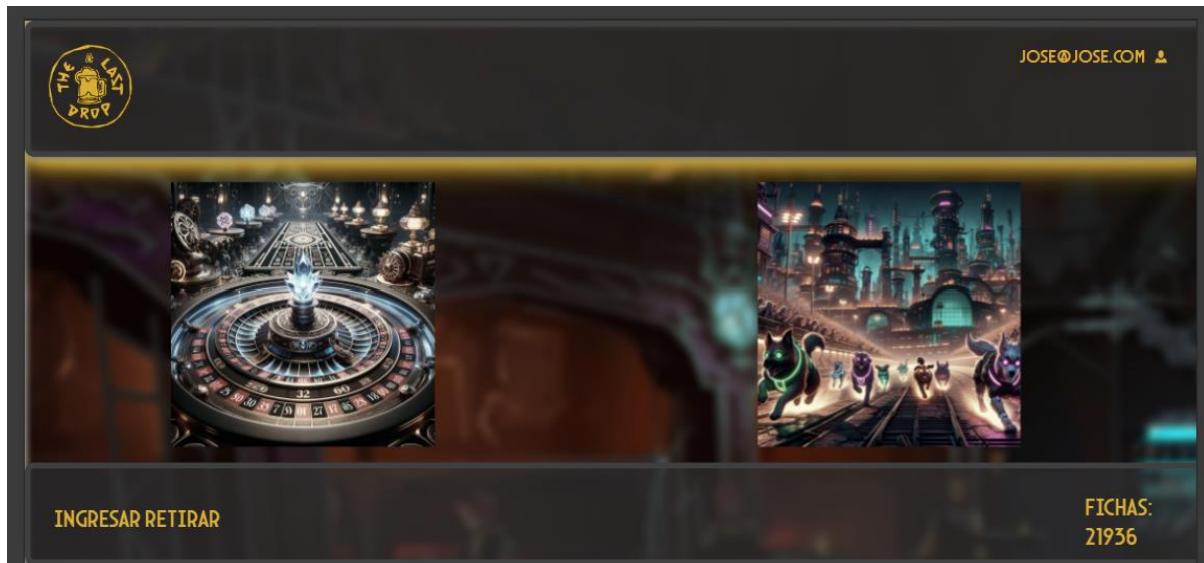
Página responsive

Mientras para el resto de páginas, el responsive es para el teléfono de forma horizontal. Aquí los ejemplos.



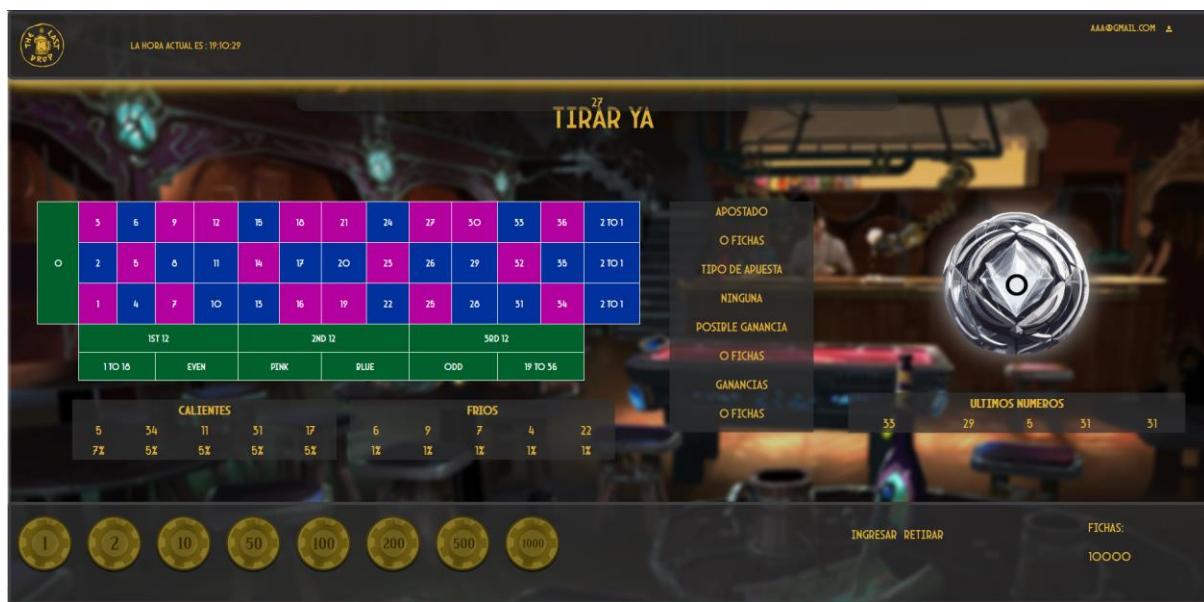
Lobby original

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS



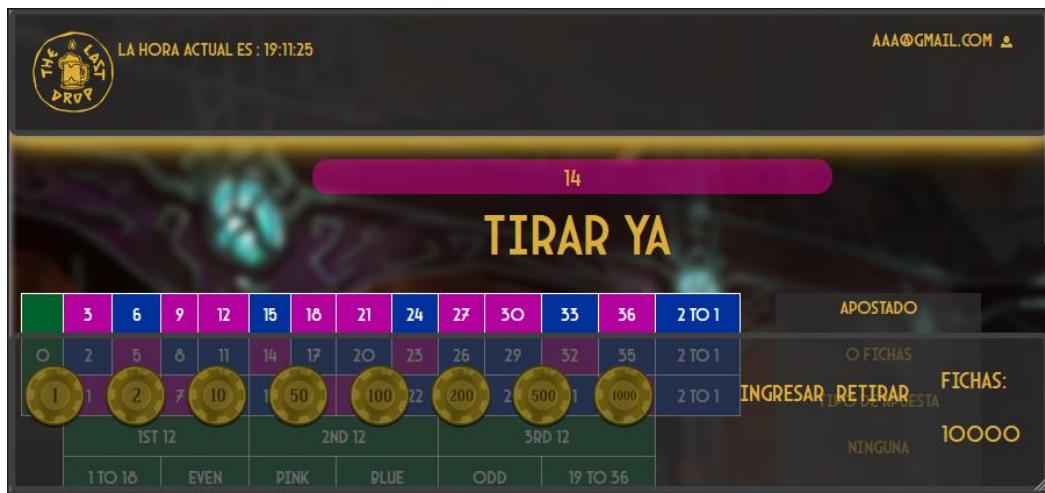
Lobby responsive

Como apunte de la ruleta, vemos como la bola (ruleta) no existe hasta que no se realiza la tirada. Además, tenemos scroll lateral para poder ver el resto de la ruleta.

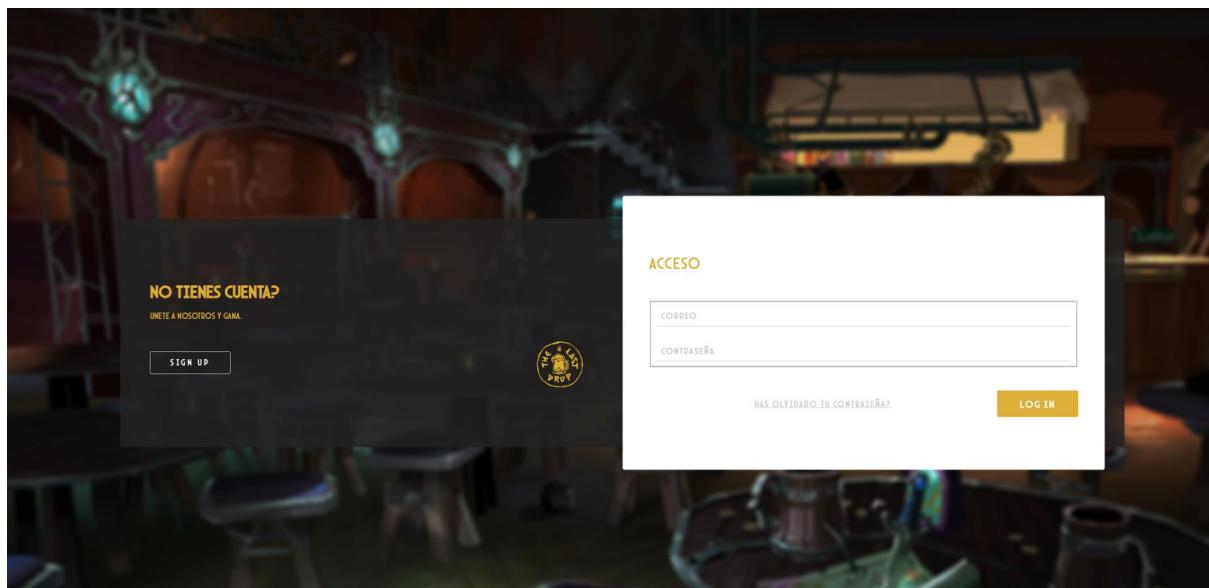


Ruleta original

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

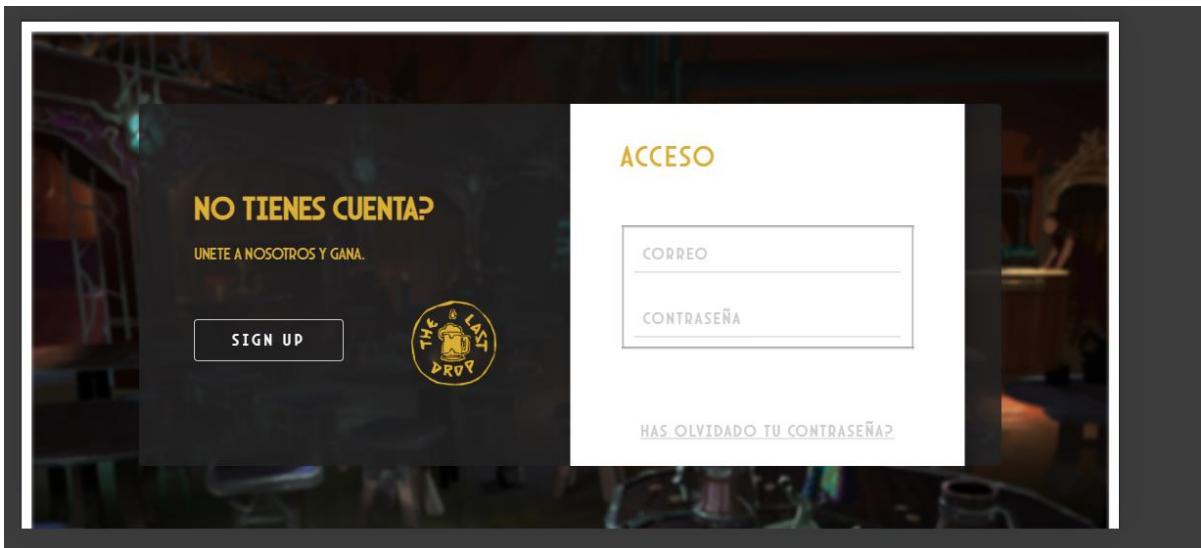


Ruleta responsive



Login original

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS



Login responsive

En el login tenemos un scroll lateral para poder ver el resto del formulario de registro.

7. DESPLIEGUE DE LA APLICACIÓN

Al ver los puntos que se pedían en el proyecto decidimos adaptar nuestro proyecto para poderlo lanzar en TomCat 9. Buscamos en la documentación oficial de spring. Y nos pusimos a ello.

Lo primero fue cambiar la clase general de nuestro proyecto para adaptarla a Spring.

```
public class ProyectoTransversalApplication extends SpringBootServletInitializer {  
  
    @Override  
    protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {  
        return application.sources(ProyectoTransversalApplication.class);  
    }  
  
    public static void main(String[] args) {  
        SpringApplication.run(ProyectoTransversalApplication.class, args);  
    }  
}
```

A continuación, fue cambiar el pom.xml para que se adaptara como .war añadiendo en el archivo lo siguiente.

```
<packaging>war</packaging>
```

Ahora a través del terminal de Spring o desde CMD, ejecutamos en la raíz de nuestro proyecto el comando mvn clean package para generarnos un paquete limpio .war de nuestro proyecto.

```
C:\Users\yoque\Downloads\Proyecto_transversal_24-02 FIN FUNCIONALIDAD\Proyecto_transversal>mvn -v  
Apache Maven 3.9.6 (bc0240f3c744dd6b6ec2920b3cd08dcc295161ae)  
Maven home: C:\Users\yoque\Desktop\apache-maven-3.9.6  
Java version: 21.0.1, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-21  
Default locale: es_ES, platform encoding: UTF-8  
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"  
  
C:\Users\yoque\Downloads\Proyecto_transversal_24-02 FIN FUNCIONALIDAD\Proyecto_transversal>mvn clean package  
[INFO] Scanning for projects...  
[INFO]  
[INFO] -----< Jose_Antonio:Proyecto_transversal >-----  
[INFO] Building Proyecto_transversal 0.0.1-SNAPSHOT  
[INFO]   from pom.xml  
[INFO] -----[ war ]-----  
[INFO]  
[INFO] --- clean:3.3.2:clean (default-clean) @ Proyecto_transversal ---  
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/4.0.0/plexus-utils-4.0.0.pom  
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/4.0.0/plexus-utils-4.0.0.pom (8.7 kB at 28 kB/s)  
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus/13/plexus-13.pom  
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus/13/plexus-13.pom (27 kB at 856 kB/s)  
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/4.0.0/plexus-utils-4.0.0.jar  
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/4.0.0/plexus-utils-4.0.0.jar (192 kB at 4.2 MB/s)  
[INFO] Deleting C:\Users\yoque\Downloads\Proyecto_transversal_24-02 FIN FUNCIONALIDAD\Proyecto_transversal\target  
[INFO]  
[INFO] --- resources:3.3.1:resources (default-resources) @ Proyecto_transversal ---  
[INFO] Copying 1 resource from src\main\resources to target\classes  
[INFO] Copying 51 resources from src\main\resources to target\classes  
[INFO]  
[INFO] --- compiler:3.11.0:compile (default-compile) @ Proyecto_transversal ---  
[INFO] Changes detected - recompiling the module! :source  
[INFO] Compiling 29 source files with javac [debug release 17] to target\classes
```

Como observamos se ejecutaba correctamente.

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

```
downloaded from central: https://repo.maven.apache.org/maven2/org/junit/platform/junit-platform-commons/1.9.2/junit-platform-commons-1.9.2.pom (2.8 kB at 167 kB/s)

:: Spring Boot ::   (v3.2.2)

2024-02-24T17:27:28.724+00:00 INFO 19636 --- [           main] J.t.ProyectoTransversalApplicationTests : Starting ProyectoTransversalApplicationTests using Java 21.0.1 with PID 19636 (started by you in C:\Users\yoque\Downloads\Proyecto_transversal 24-02 FIN FUNCIONALIDAD\Proyecto_transversal)
2024-02-24T17:27:28.725+00:00 INFO 19636 --- [           main] J.t.ProyectoTransversalApplicationTests : No active profile set, falling back to 1 default profile: "default"
2024-02-24T17:27:29.416+00:00 INFO 19636 --- [           main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2024-02-24T17:27:29.541+00:00 INFO 19636 --- [           main] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 63 ms. Found 2 JPA repository interfaces.
2024-02-24T17:27:30.054+00:00 INFO 19636 --- [           main] com.zaxxer.hikari.HikariDataSource  : HikariPool-1 - Starting...
2024-02-24T17:27:30.541+00:00 INFO 19636 --- [           main] com.zaxxer.hikari.HikariDataSource  : HikariPool-1 - Added connection oracle.jdbc.driver.T4CConnection@8e25d3f
2024-02-24T17:27:30.541+00:00 INFO 19636 --- [           main] o.hibernate.jpa.internal.util.LogHelper : HHH000260: Processing PersistenceUnitInfo [name: default]
2024-02-24T17:27:30.632+00:00 INFO 19636 --- [           main] o.hibernate.jpa.internal.util.LogHelper : HHH000261: Hibernate core version 6.4.1.Final
2024-02-24T17:27:30.632+00:00 INFO 19636 --- [           main] org.hibernate.dialect.Dialect      : HHH0000511: The 18.0.0 version for [org.hibernate.dialect.OracleDialect] is no longer supported, hence certain features may not work properly. The minimum supported version is 19.0.0. Check the community dialects project for available legacy versions.
2024-02-24T17:27:31.107+00:00 WARN 19636 --- [           main] org.hibernate.orm.deprecation       : HHH90000025: OracleDialect does not need to be specified explicitly using 'hibernate.dialect' (remove the property setting and it will be selected by default)
2024-02-24T17:27:31.892+00:00 INFO 19636 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator    : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)
2024-02-24T17:27:31.895+00:00 INFO 19636 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2024-02-24T17:27:32.328+00:00 WARN 19636 --- [           main] JpaBaseConfigurations$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2024-02-24T17:27:32.770+00:00 INFO 19636 --- [           main] J.t.ProyectoTransversalApplicationTests : Started ProyectoTransversalApplicationTests in 4.272 seconds (process running for 5.091)
WARNING: Java agent has been loaded dynamically (C:\Users\yoque\Downloads\netbytewebapp\byte-buddy\byte-buddy-agent\1.44.11\byte-buddy-agent-1.44.11.jar)
WARNING: If a serviceability tool is in use, please run with -XX:+EnableAgentLoadingForTracing to hide this warning
WARNING: If a serviceability tool is not in use, please run with -Djdk.intrinsics.traceUsage for more information
WARNING: Dynamic loading of agents will be disabled by default in a future release
java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 5.081 s -- in Jose_Antonio.transversal.ProyectoTransversalApplicationTests
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] --- war:3.4.0:war (@Proyecto_transversal ---)
[INFO] Packaging war
[INFO] Assembling webapp [Proyecto_transversal] in [C:\Users\yoque\Downloads\Proyecto_transversal 24-02 FIN FUNCIONALIDAD\Proyecto_transversal\target\Proyecto_transversal-0.0.1-SNAPSHOT]
[INFO] Processing war project
[INFO] Copying webapp resources [[:\Users\yoque\Downloads\Proyecto_transversal 24-02 FIN FUNCIONALIDAD\Proyecto_transversal\src\main\webapp]]
[INFO] Building war: C:\Users\yoque\Downloads\Proyecto_transversal 24-02 FIN FUNCIONALIDAD\Proyecto_transversal\target\Proyecto_transversal-0.0.1-SNAPSHOT.war
```

Finalmente fuimos a desplegarlo en nuestro servidor de Tomcat9. Por desgracia, no fuimos capaces de lanzarlo por el tema de las rutas.

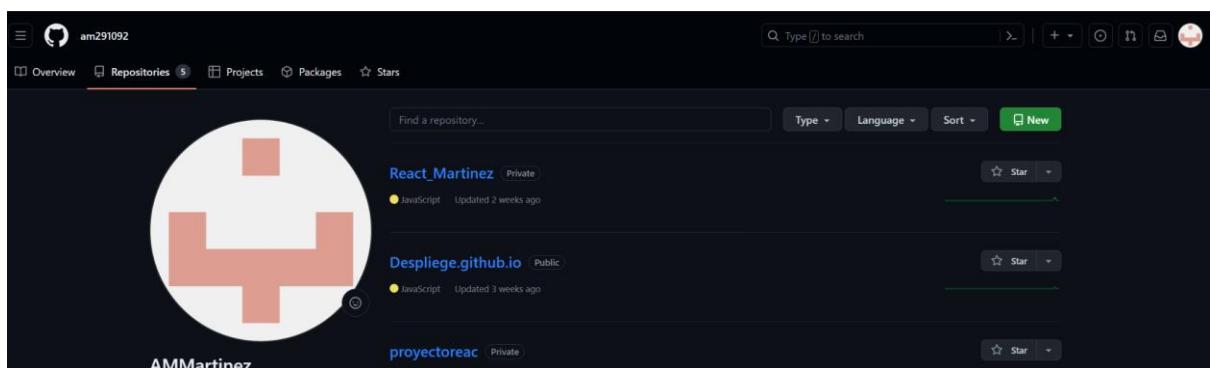
Para intentar solucionar el error miramos en varias páginas incluidas manuales de Spring, tomcat etc. pero no fuimos capaces de dar con la tecla. Aun así, ya sabemos cómo crear .war de nuestros proyectos.

Spring Boot trae un servidor TomCat 7 embebido donde lanzamos el propio servidor.

Para poder hacer el control de versiones hemos usado un repositorio de GitHub el cual también nos ha generado problemas por la cantidad de archivos que tenía nuestro proyecto y por el tamaño de los mismos ya que para subirlo completo deberíamos de pagar.

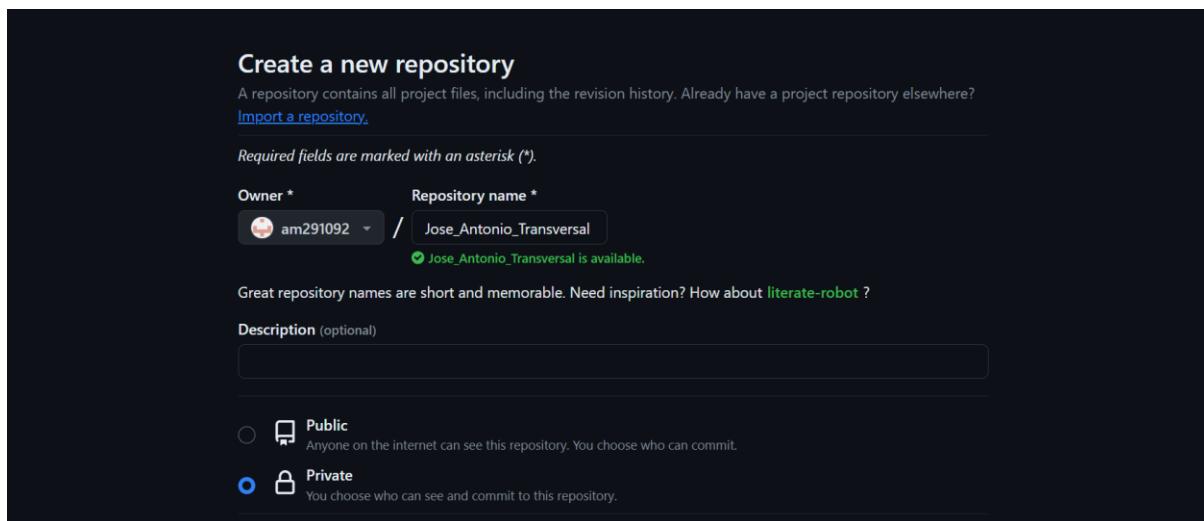
Manual de creación de repositorio mediante GitHub 

Para crear nuestro repositorio, primero debemos tener una cuenta válida, y estar logueados en la página de GitHub, una vez logueados, debemos irnos a nuestros repositorios y darle en crear uno nuevo



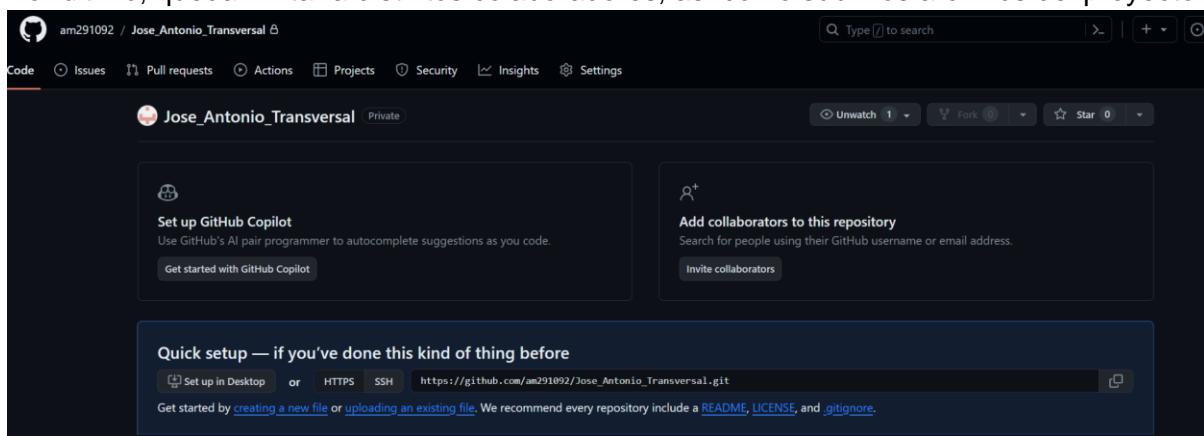
JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

una vez le hemos dado nos saldrá un menú para la creación del mismo donde debemos elegir un nombre y si lo queremos público o privado:

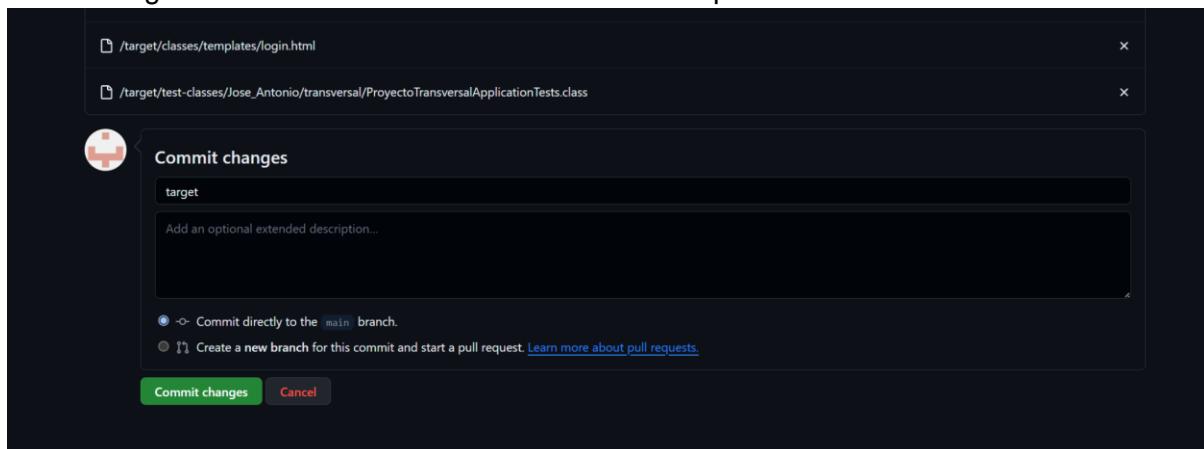


En nuestro caso lo vamos a crear privado ya que solo nos interesa tener acceso nosotros y aquellos que vayan a evaluar el mismo.

Por último, queda invitar a distintos colaboradores, así como subir los archivos del proyecto:



al ser una gran cantidad de archivos lo hemos tenido que hacer en 2 tandas



JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS

The screenshot shows a GitHub repository page for 'Jose_Antonio_Transversal'. The repository is private. It contains 1 branch and 0 tags. The commit history shows a single commit by 'am291092 target' from 'e011ec9 · now'. The commit message is 'primera tanda'. The commit was made 2 minutes ago. The repository structure includes 'src', 'target', 'HELP.md', 'mvnw', 'mvnw.cmd', and 'pom.xml', all of which were updated 2 minutes ago. The 'About' section indicates no description, website, or topics are provided. The 'Activity' section shows 0 stars, 1 watching, and 0 forks. There are sections for 'Releases' (no releases published) and 'Packages' (no packages published).

y ya tendríamos los archivos subidos.
para añadir a los colaboradores:

nos vamos a setting, add collaborator y ya buscamos a quien queramos añadir

The screenshot shows the 'Manage access' settings page for the repository. It lists five pending invite recipients: 'daniprofesor', 'dgonzalez@efamoratalaz.com', 'Moowette', 'rsanchez@efamoratalaz.com', and 'Sergiolinfantes'. Each entry includes a checkbox to select all, a user icon, the recipient's name and email, their status ('Awaiting response' or 'Pending Invite'), and a 'Remove' button. A green 'Add people' button is located at the top right. At the bottom, there is a callout for team access controls and a link to 'Create an organization'.

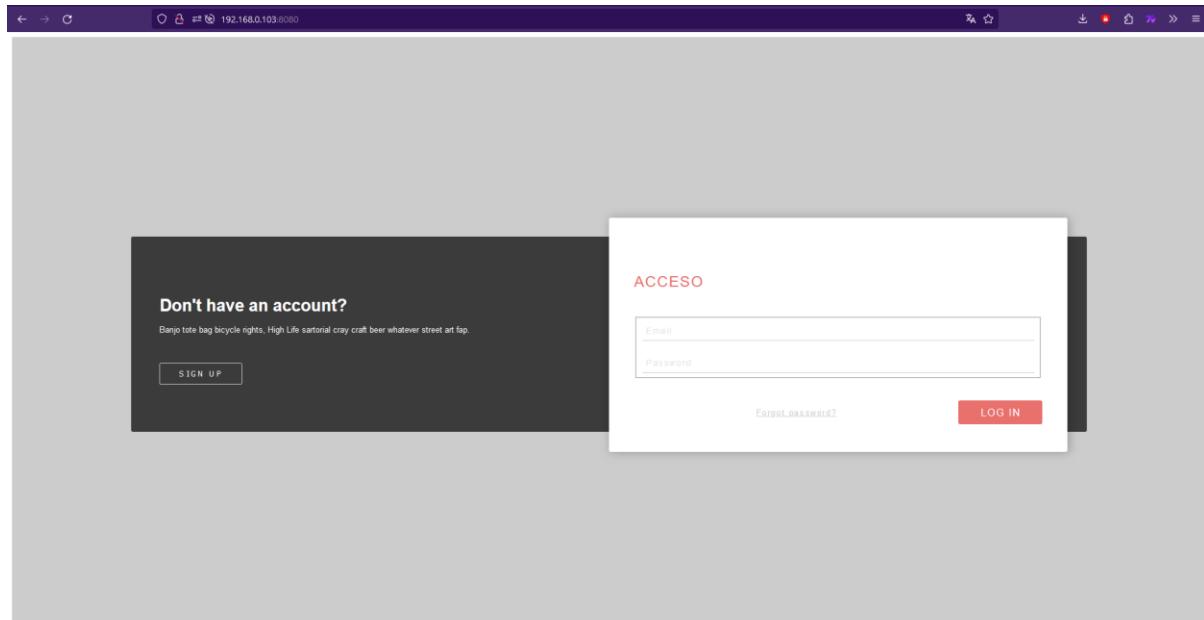
y ya estarían todos los colaboradores invitados

8. MODO DE TRABAJO

Para organizarnos de una manera más eficaz hemos usado la herramienta de GitHub lo máximo que hemos podido. Cuando esto no ha funcionado hemos usado otros medios para compartir y mergear nuestros archivos.

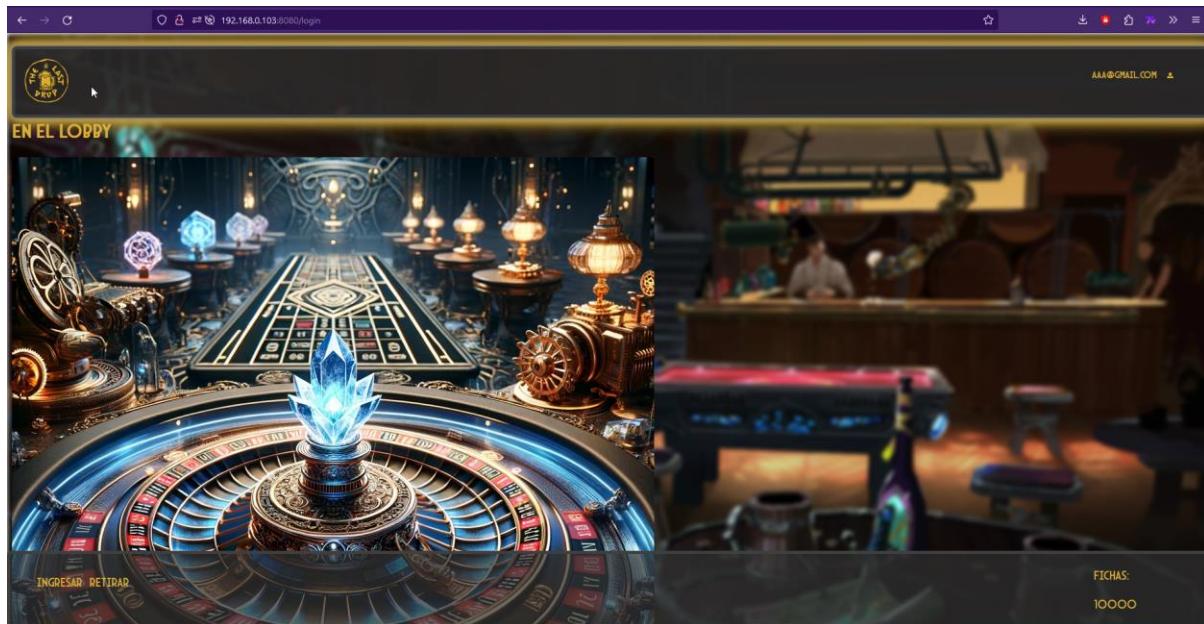
La forma de trabajar era que cada uno tocara elementos diferentes para no tener problemas con los códigos y no tocar los mismos elementos. Cuando uno terminaba algo o se atascaba nos ayudábamos mutuamente a solucionar los problemas.

Hemos priorizado funcionalidad y cuando ya estaba acabada nos hemos puesto a trabajar en el aspecto de diseño (cuadrar todo al diseño de Arcane).



Login antes de adaptarlo

JOSE MARIA DELGADO FERNANDEZ-ANTONIO MARIA MARTINEZ CEBALLOS



Lobby antes de adaptarse



Galgos con iconografía antigua y sin fuente