Proyecto 1ero - Técnico Superior en Desarrollo de Aplicaciones Multiplataforma

Caso de estudio

Asociación contra el cáncer de testículo.

GitHub

VideoProyecto

Renaud Bronchart

INDICE

Me gustaría mostrar mi más sincero agradecimiento y gratitud a Laurence P. Sin su apoyo, no habría conseguido empezar a estudiar de nuevo y llegar a hacer este proyecto.

Resumen

Esta es la memoria del Proyecto 1er curso Técnico Superior en Desarrollo de Aplicaciones Multiplataforma cursada en el Instituto Formación Profesional Nebrija.

El proyecto está basado en una aplicación web diseñada y desarrollada para poder mejorar la gestión de contenido y el control de inventario en un entorno de una asociación.

La gestión será completamente editable por medio de menús y formularios los cuales han sido realizados con las tecnologías web actuales como HTML, JAVA, MySQL.

El objetivo principal de esta aplicación web es permitir la difusión de informaciones, la gestión de carreras y dar visibilidad a asociación.

1 Introducción

En nuestros días, el contenido, la gestión y la difusión de informaciones sobre una organización como una asociación es primordial. Es por ello, que cualquier organismo que no tenga una aplicación web de calidad, para facilitar la interactuación con sus usuarios le puede impactar negativamente.

En este proyecto, se pretender crear una aplicación web que permite dar visibilidad a asociación y permitir una gestión eficaz de sus contenidos con una interfaz fácil de usar y de mantener.

El portal va a permitir la creación, la modificación, la difusión y la gestión de contenidos sobre una asociación contra el cáncer de testículo.

1.1 Justificación del proyecto

En nuestra época, es muy importante de poder apoyar las diferentes asociaciones que existen. Una aplicación web como asociación contra el cáncer de testículo permite la difusión de informaciones, y dar más visibilidad a este tema tan importante.

Decidí crear una aplicación web sobre el tema del cáncer de testículo porque tuve un cáncer que me ha afecto en mi vida y desde entonces participo mas en todo en relación con el cáncer.

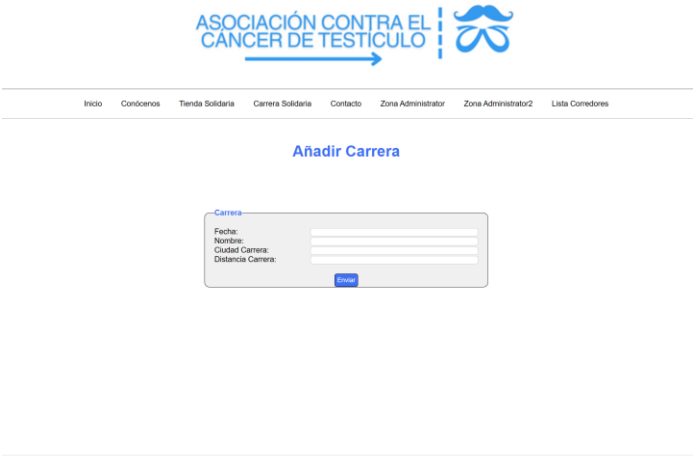
Como soy un aficionado del deporte y más del correr, quería combinar una aplicación que permite la difusión de noticias sobre asociación e informar sobre el tema. También se podría crear eventos como carrera de 10k para promover asociación y permitir dar más visibilidad sobre el cáncer de testículo.

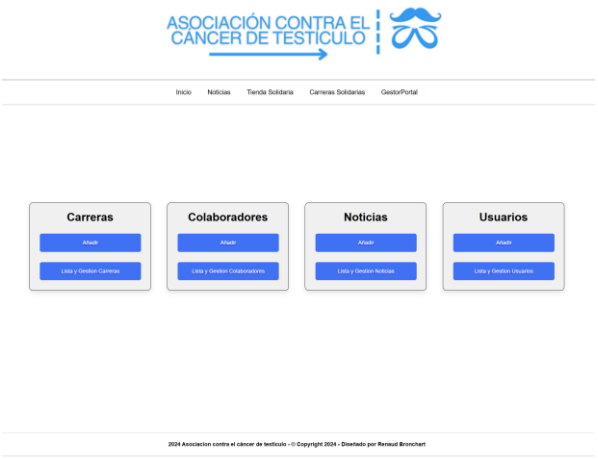
1.2 Descripción General

La estructura del portal se basará en dos partes: en una parte pública (Front-end) y en una privada (Back-end).

La aplicación web permitiría a los usuarios de acceder a las informaciones de asociación(Noticias,Colaboradores, Carreras, etc…)

El back-end es el gestor donde se puede crear, editar y borrar las carreras, las noticias y los colaboradores.





1.3 Objetivos principales

El objetivo principal del proyecto es la construcción de un portal web capaz de publicar eventos, noticias, los colaboradores y que podemos editar, borrar los contenidos que tengamos creados.

Los elementos principales son:

- Un portal web donde el usuario puede acceder a eventos creados.
- Un portal web donde el usuario puede registrase para los eventos creados.
- Un portal web donde el usuario puede ver las noticias de la asociación.
- Un portal web donde el usuario puede ver los colaboradores.
- Gestor de contenido que permite la creación, la modificación y la eliminación de eventos o noticias

2 Requisitos

2.1.1 Requisitos funcionales

- RF1 – El sistema mantendrá la disponibilidad del servicio para garantizar un acceso constante a los recursos

- RF2 - El sistema debe ser capaz de adaptarse rápidamente a los cambios de demande.

- RF 3 La plataforma deberá ser accesible desde cualquier lugar con conexión a internet

2.1.2 Requisitos específicos

Parte pública

- REP1 – Cualquier visitante de la web podrá visualizar el contenido de la parte publica de la web

- REP2 - Los usuarios anónimos podrán registrarse en el sistema utilizando el mecanismo de registro de usuarios establecido.

Parte privada

- REU1 – Los usuarios podrán inscribirse a unas carreras vía un formulario
- REU2 – Los usuarios registrados podrán acceder su perfil y realizar cambios.

Gestor

- REG 1 - Esta parte será únicamente para los usuarios guardado como administradores
- REG2 - El administrador podrá introducir nuevos colaboradores y nuevas noticias o carreras en la sesión de aplicación web
- REG3 - El administrador podrá editar o eliminar colaboradores, noticias, y carreras y en la sesión de aplicación web

3 Análisis

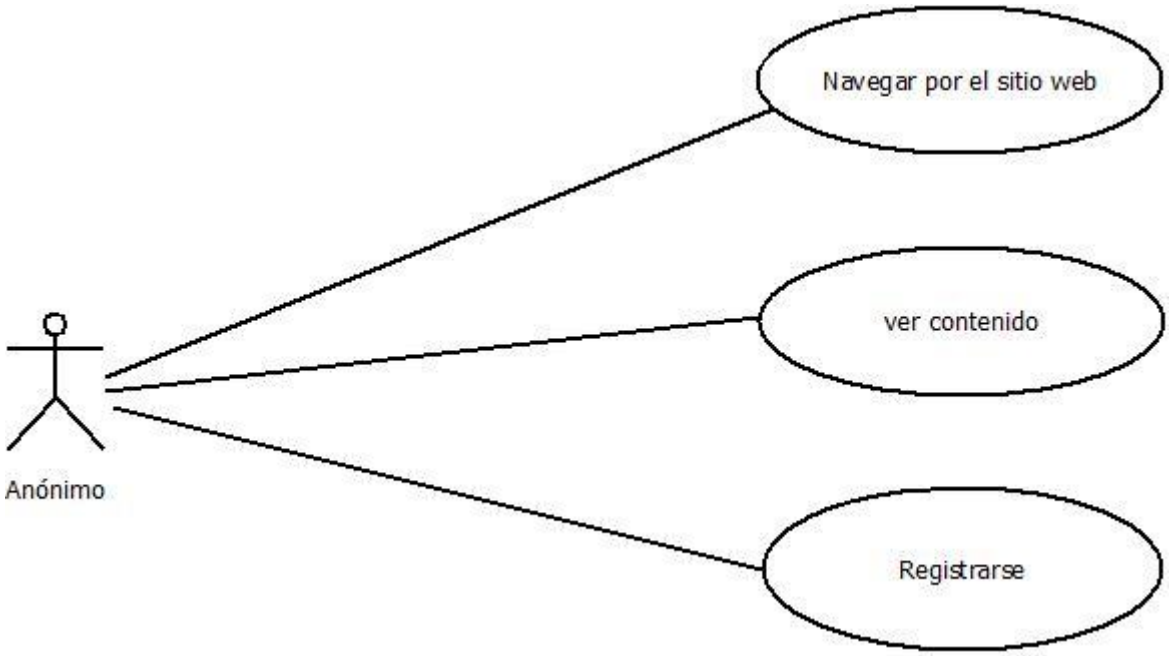
En este capítulo, se describe la fase de análisis del proyecto, su estructura y funcionalidad.

3.1 Casos de uso

Un **caso de uso** es la descripción de una acción o actividad. Un diagrama de caso de uso es una descripción de las actividades que deberá realizar alguien o algo para llevar a cabo algún proceso.[1]

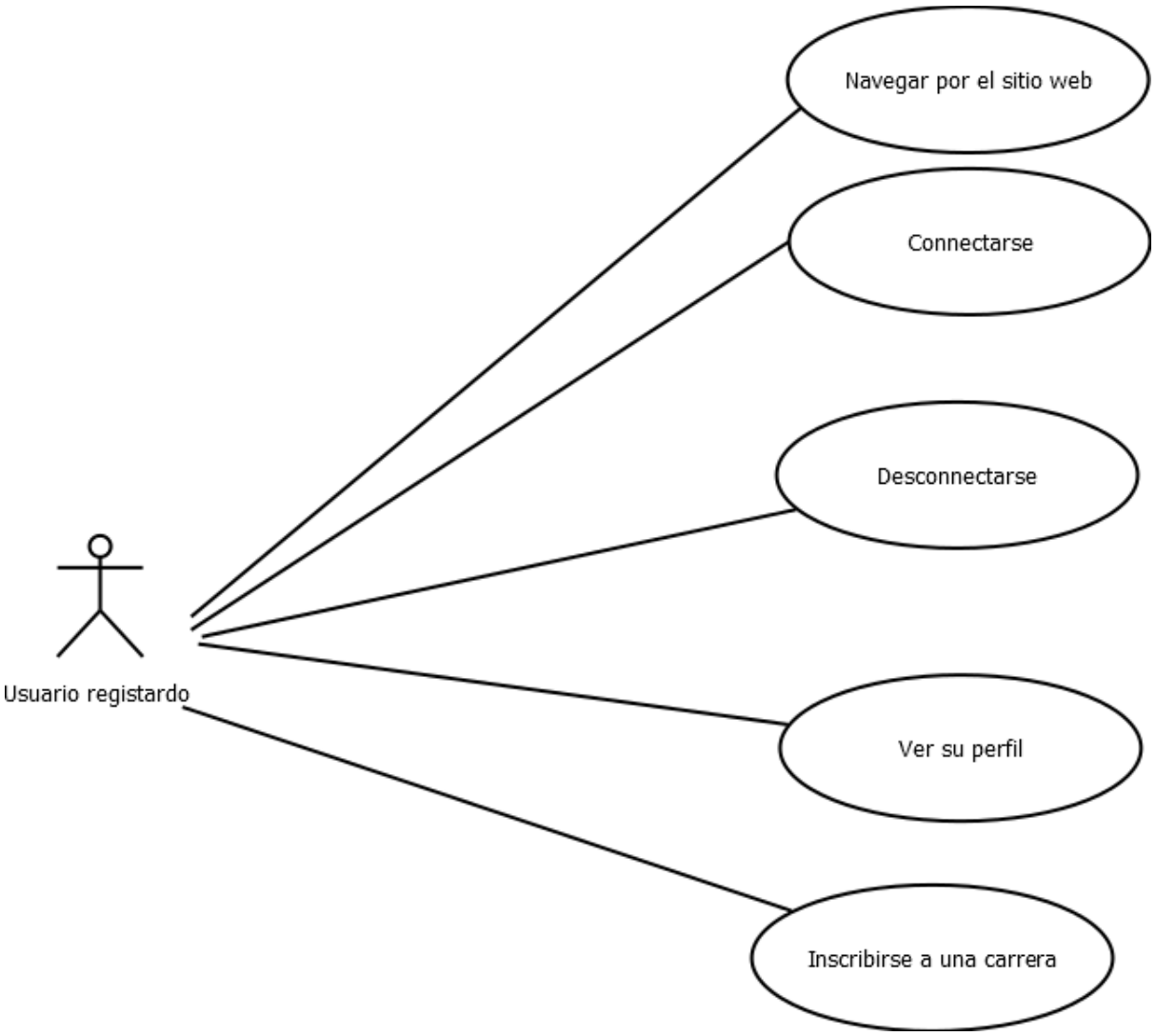
A continuación, se muestran los casos de uso de la parte del portal web, con los actores:

3.1.1 **Anónimo**: visitante estándar sin estar registrado

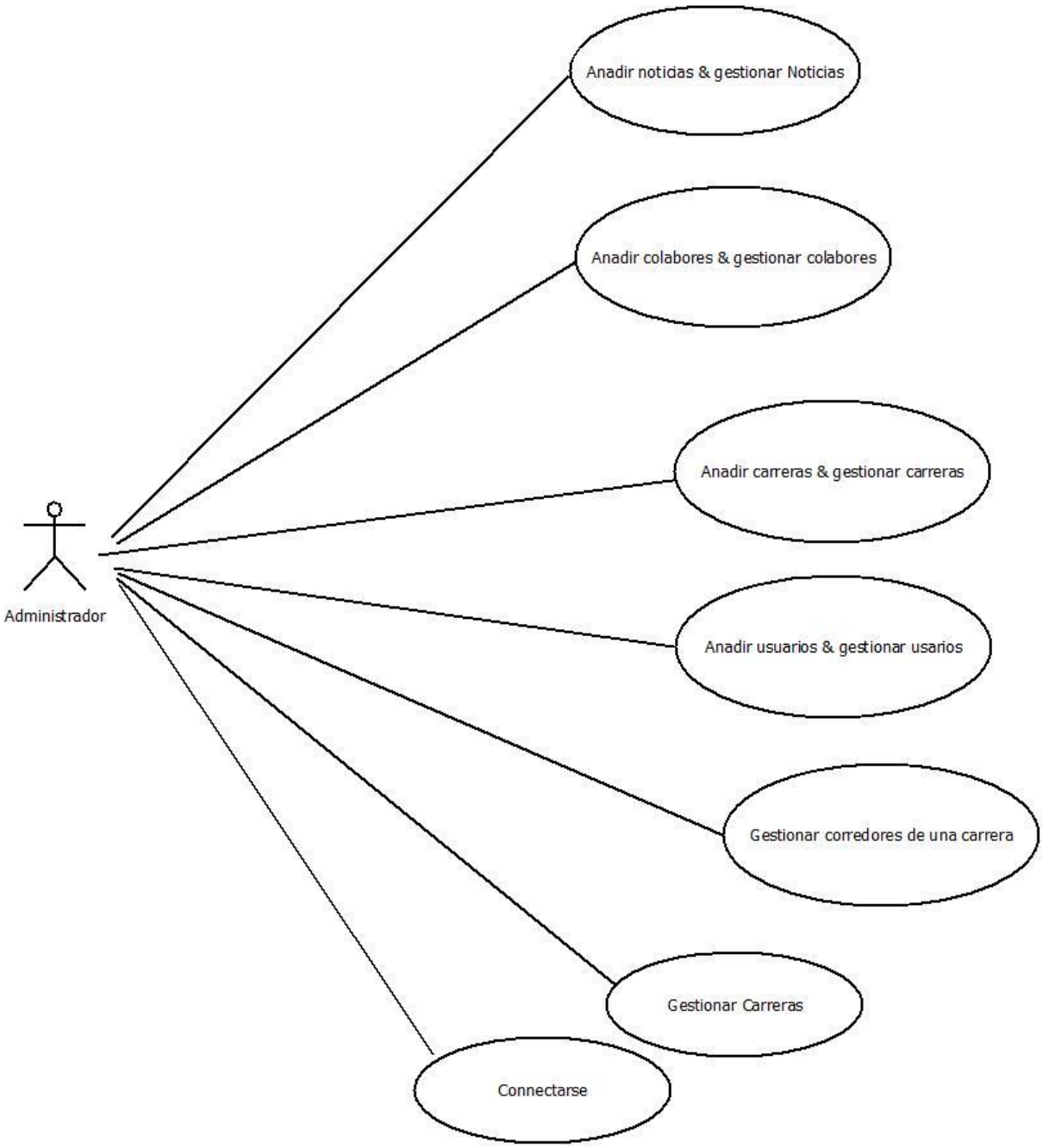


[1] Fuente : Wikipédia (ver bibliografía).

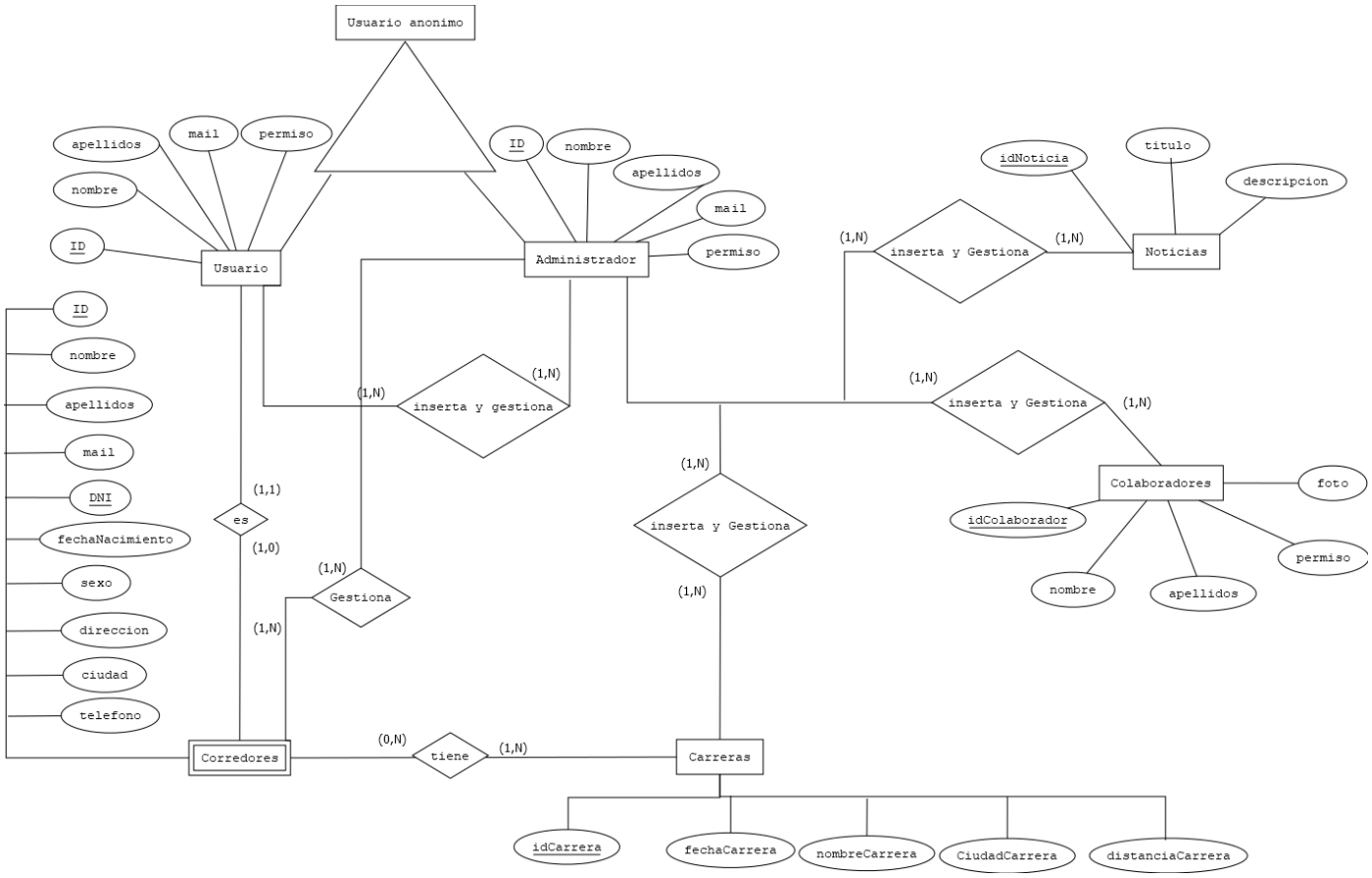
3.1.2 **Usuario**: Usuario registrado



3.1.3 **Administrador**: Administrador del portal web



3.2 Diagrama Entidad-Relación



3.2.1 Paso a Tablas - Asociación contra el cáncer

Usuarios (id, nombre, apellidos, mail, permiso, pass)

Noticias (idNoticia, titulo, descripción)

Colabores(idColaboradores, nombre, apellidos, puesto, foto)

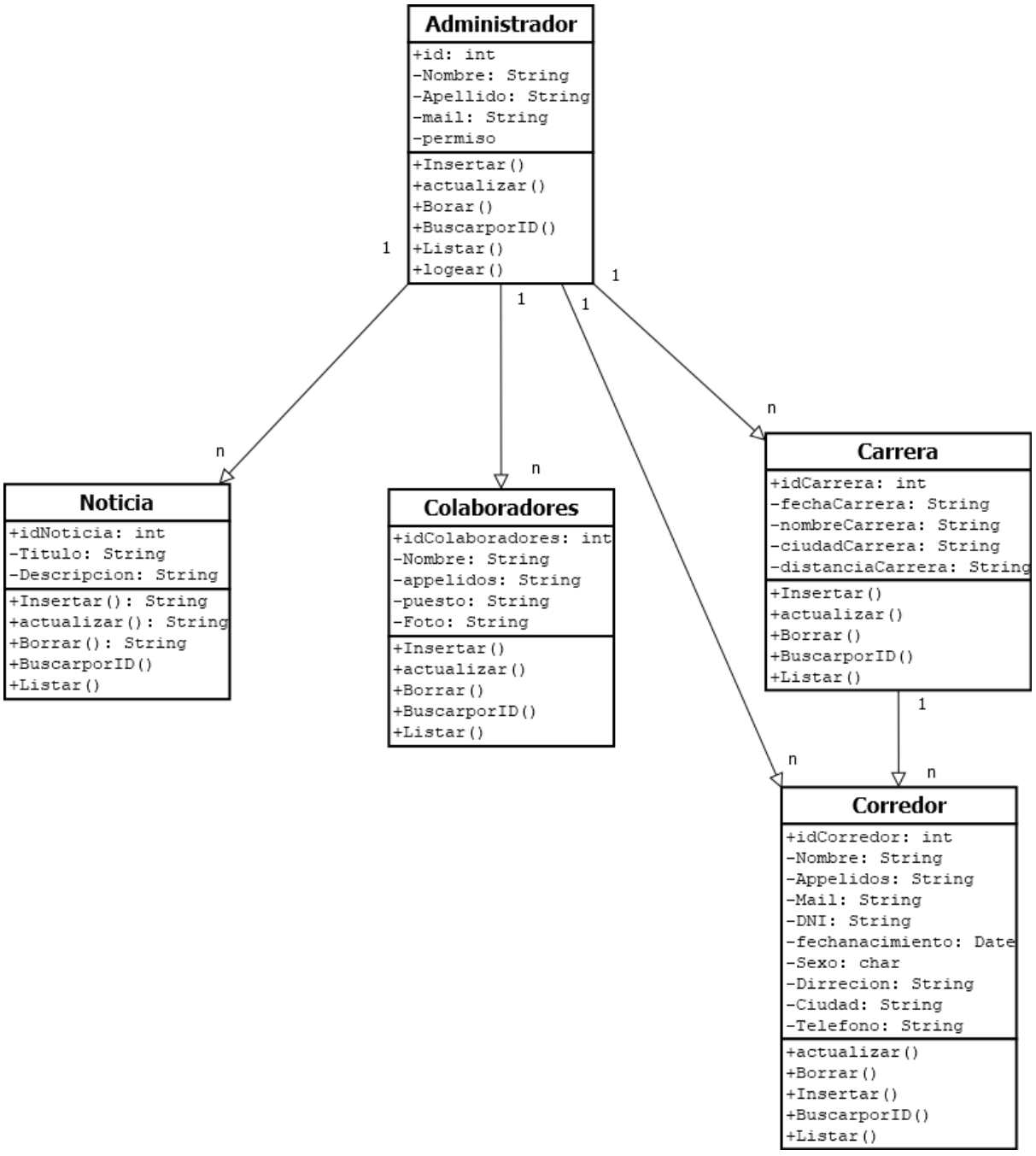
Carreras (idCarrera, fechaCarrera, nombreCarrera, ciudadCarrera, distanciaCarrera)

Corredores(id, nombre, apellidos, mail, dni, fechanacimiento, sexo, dirección, ciudad, teléfono,)

- Id(FK) > por ser PK de Usuarios

3.2.2 Diagrama de clases

Un diagrama de clases en Lenguaje Unificado de Modelado (UML) es un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos.[2]

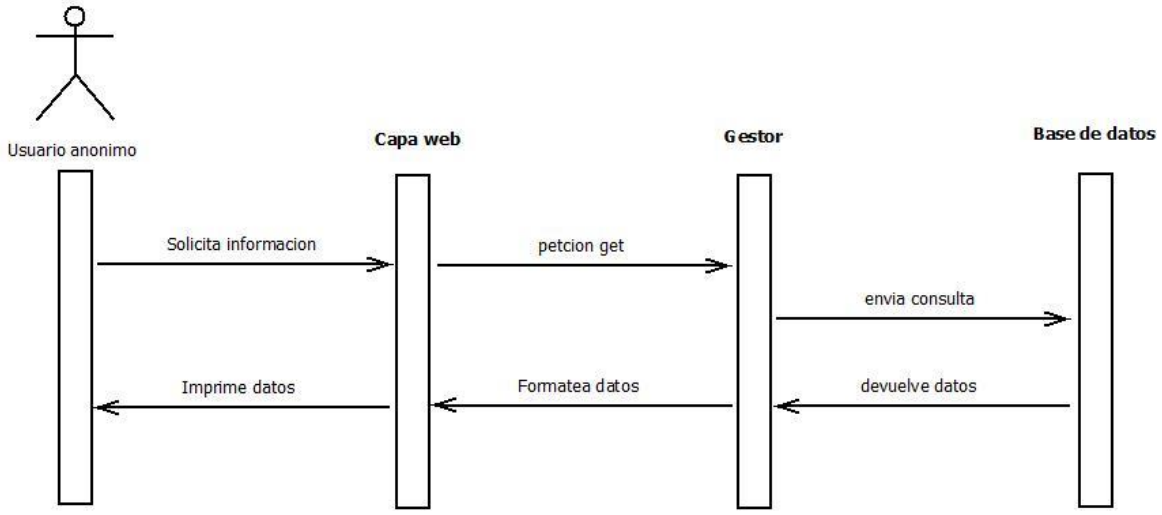


[2] Fuente : Wikipédia (ver bibliografía).

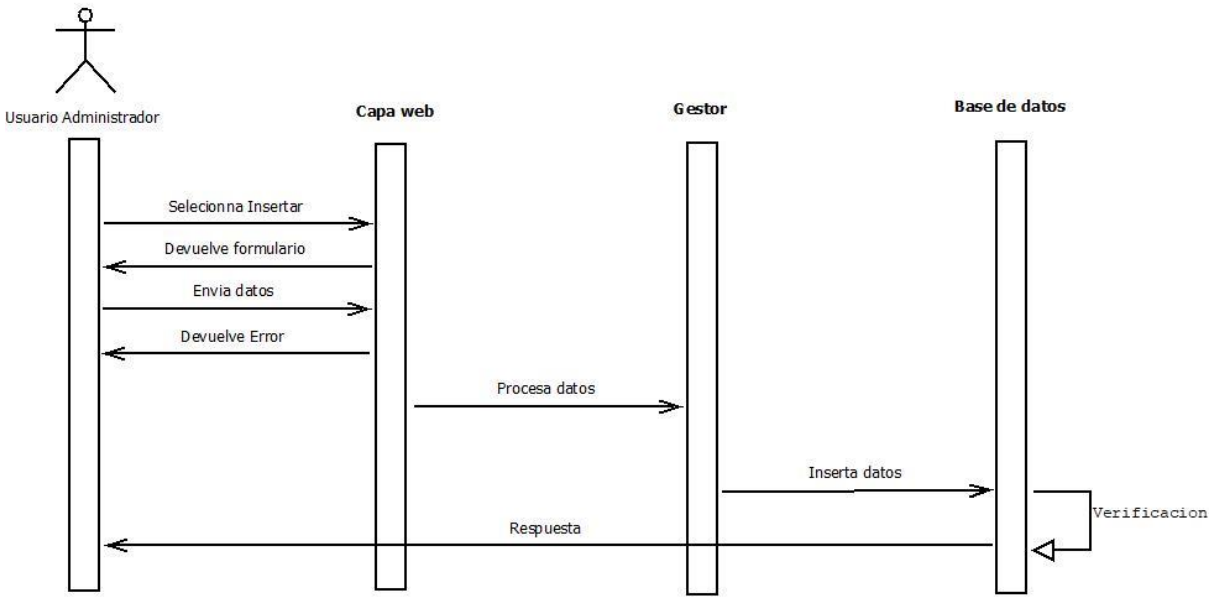
3.3 Diagrama de secuencia

Un diagrama de secuencias muestra la interacción de un conjunto de objetos de una aplicación a través del tiempo.

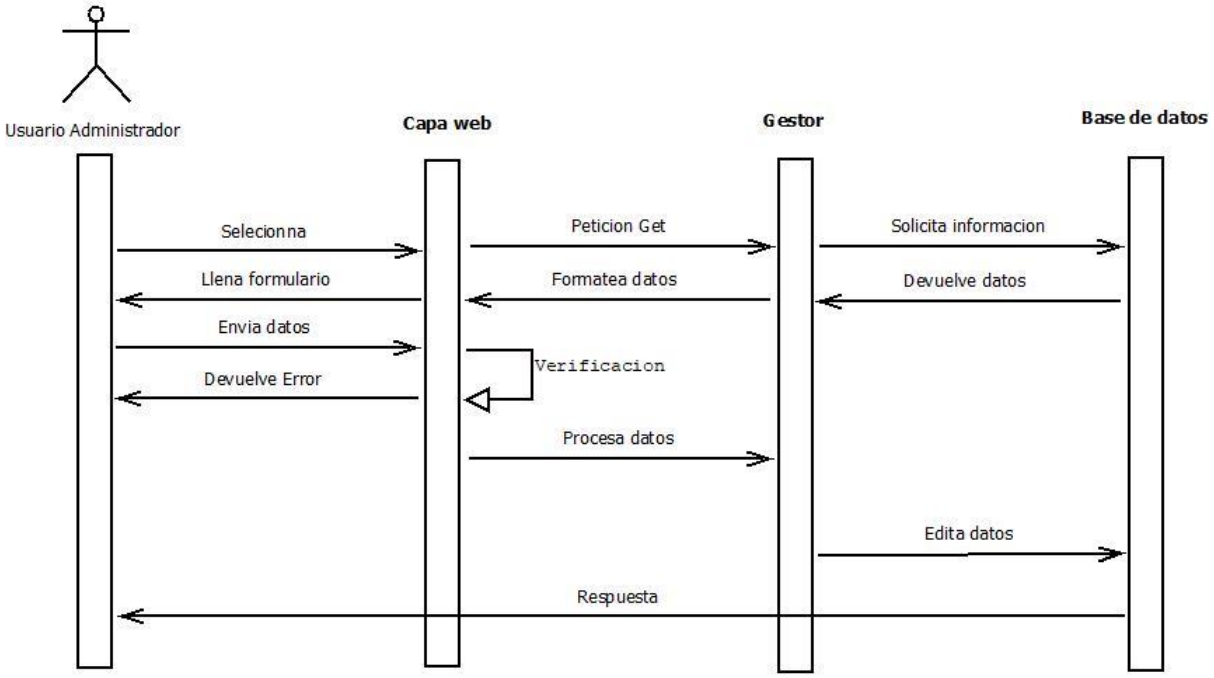
3.3.1 Visualizar contenido



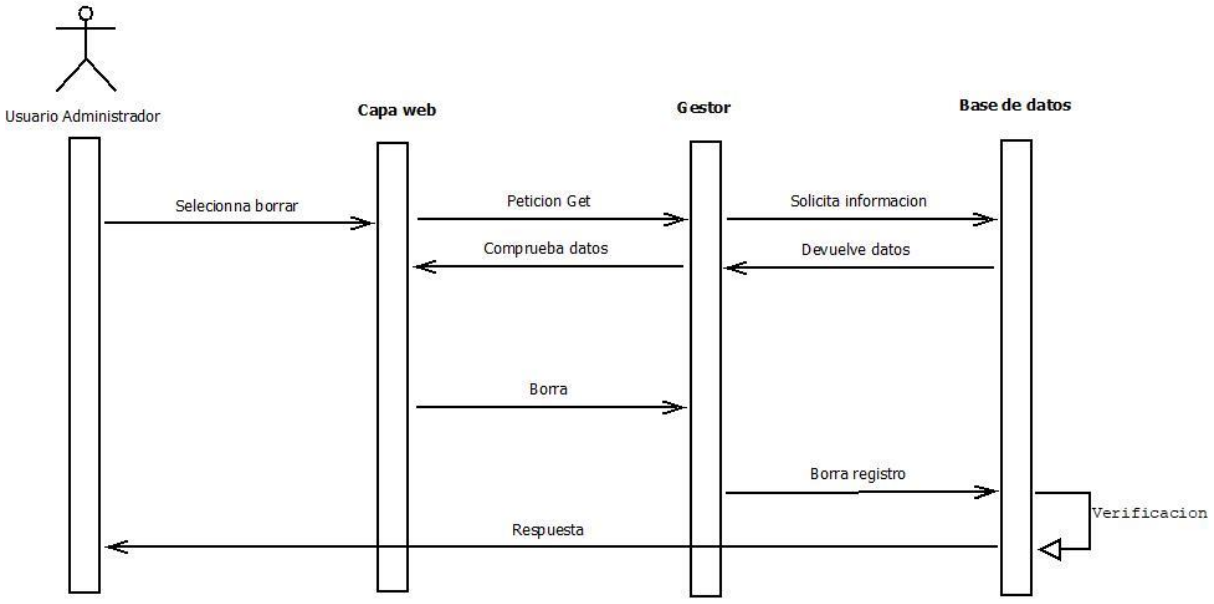
3.3.2 Insertar contenido



3.3.3 Editar contenido



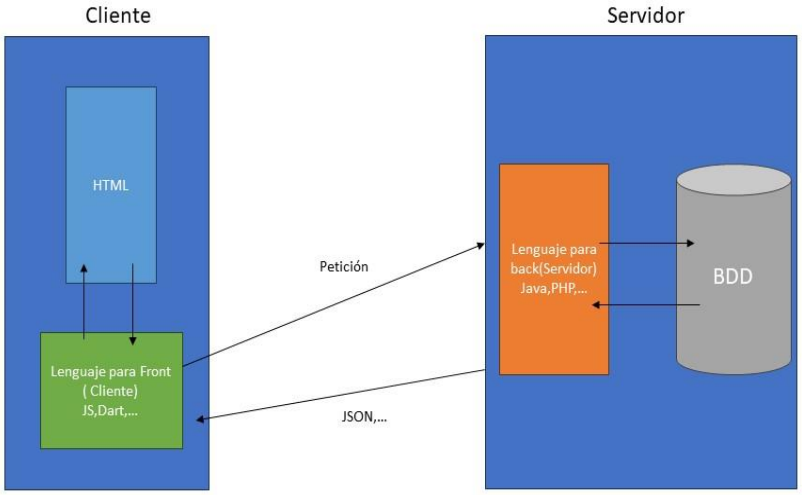
3.3.4 Borar contenido



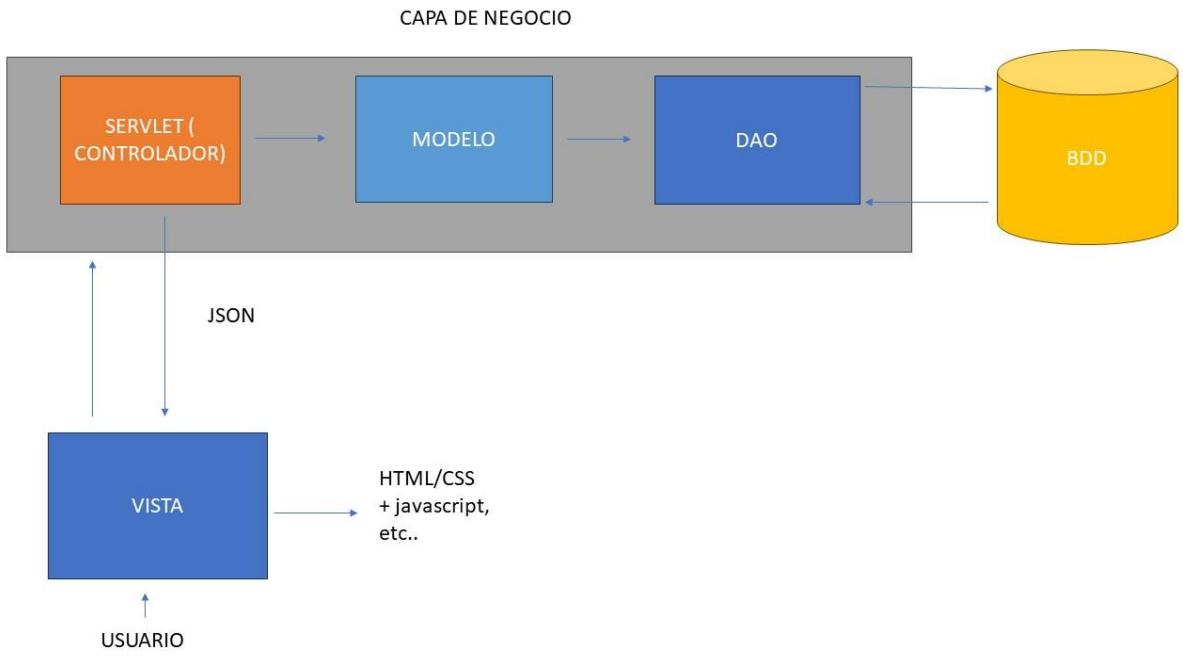
4 Diseño

4.1 Arquitectura cliente – servidor

La arquitectura cliente-servidor es un modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta.[3]



[3] [3] Fuente : Wikipédia (ver bibliografía).



-> Usuario inserte datos en el formulario creado en HTML (VISTA)

-> El servlet recibe los datos y los convierte en datos para el modelo y decide lo que va hacer con ellos

-> Las clases dao son los que van a coger los datos y se van a ocupar de gestionar esos datos y mandarlos a la BDD

-> La base de datos devolverá los datos

-> Gracias a Json, se permitirá de poner los datos a la vista para el usuario.

4.2 Capa de presentación

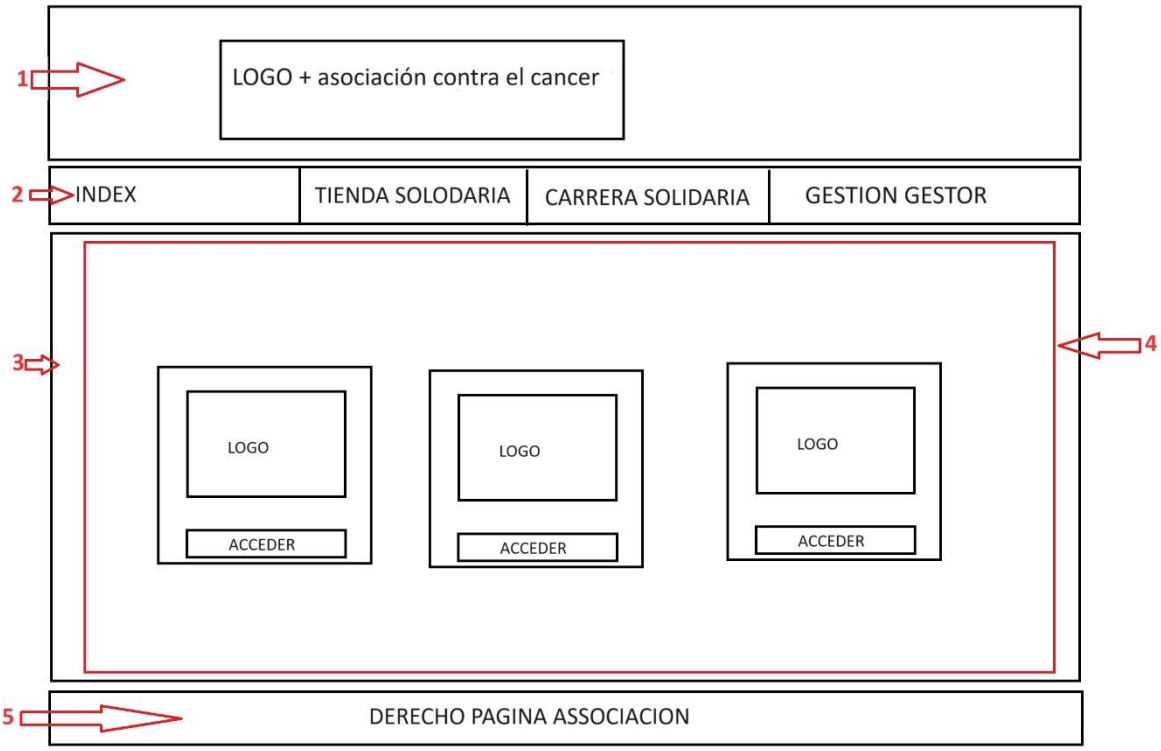
La capa de presentación, también conocida como capa de interfaz de usuario, es la interfaz gráfica de nuestro sistema, que proporciona la interacción entre los usuarios y el sistema.

4.2.1 Front-end

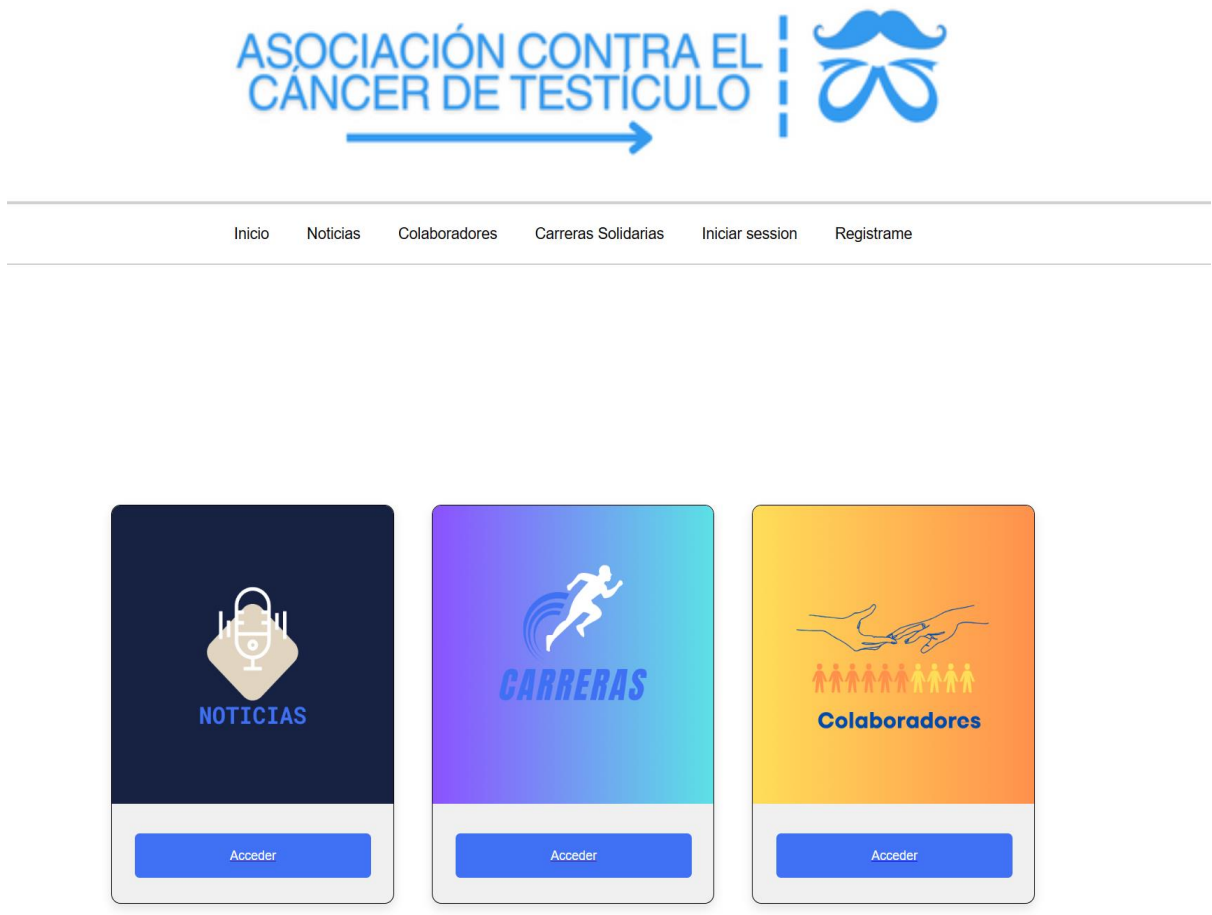
El front-end del portal es lo primero que ve el usuario al entrar en nuestra pagina web.

A continuación, mostraremos el diseño preliminar de la parte visual de nuestra aplicación con su explicación y el resultado final.

1. **CABECERA** : Logo y nombre de nuestra aplicación web
2. **Menú**: Lista de secciones disponible
3. **BODY** : parte principal donde se puede modificar el fondo de la pagina
4. **CUERPO** : parte principal de la vista donde se puede ver elementos como noticias, carreras…
5. **PIE** : datos de la pagina



Resultado final de la pagina:

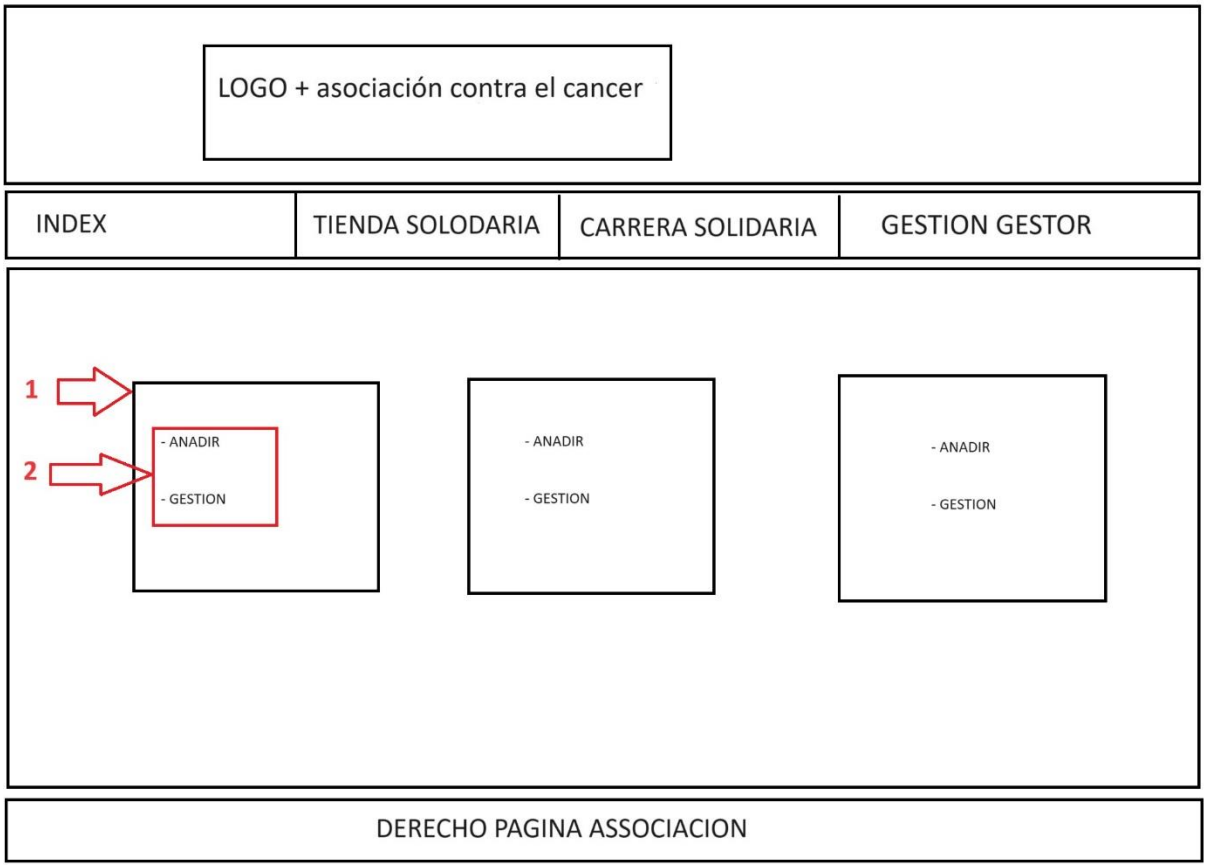


4.2.2 El Back-end

El Back-end es el gestor al que solo tendrán acceso los administradores, mediante el cual podrán mantener actualizado y configurado el portal. Es decir, podrán añadir y gestionar(editar, borrar) las carreras, las noticias, los corredores y los productos.

Mostraremos el diseño preliminar de la parte visual de nuestra aplicación con su explicación y el resultado final.

1. **Menú Gestor :** Menú del gestor donde se puede acceder a todas las secciones del portal
2. **Cuerpo Menú Gestor :** Zona de la diferentes secciones del menú gestor que permite añadir o gestionar la sección



Resultado final de la página:

Inicio Noticias Tienda Solidaria Carreras Solidarias GestorPortal

Carreras

Añadir

Lista y Gestion Carreras

Colaboradores

Añadir

Lista y Gestion Colaboradores

Noticias

Añadir

Lista y Gestion Noticias

Usuarios

Añadir

Lista y Gestion Usuarios

A continuación, se podría ver las diferentes secciones como añadir o gestionar.

Añadir

Lista y Gestión

1. **Editar :** permite cambiar datos de la carrera
2. **Borrar :** permite borrar una carrera
3. **Lista corredores :** permite ver los corredores de la carrera

Inicio Conócenos Tienda Solidaria Carrera Solidaria GestorPortal

Lista carreras

#	fecha	nombre	ciudad	distancia				
59	15/02/2025	Carrera Tetuan	Madrid	10 KM	1	Editar	Borrar	Lista Corredores
75	12/09/2024	Run para todos	Cordoba	5 KM		Editar	2 Borrar	Lista Corredores
76	8/06/2024	Carrera Profe FP Nebrija	Madri	1 km		Editar	Borrar	3 Lista Corredores

5 Implementación

5.1 Tecnologías

5.1.1 HTML

Html (HyperText Markup Language) es un lenguaje de etiquetas usado por los diseñadores y programadores para crear páginas web, este lenguaje es un estándar reconocido por en todo el mundo y cuyas normas las define un organismo sin ánimo de lucro llamado World Wide Web Consortium (http://www.w3.org/). Un documento Html está formado por su contenido y etiquetas que delimitan sus partes pudiendo formatear la página aplicando el diseño o estructura deseada

Actualmente HTML5 es la última actualización de este lenguaje de marcas, aunque realmente es un termino que sirve para agrupar las nuevas tecnologías de desarrollo de aplicaciones web: HTML5, CSS3 y nuevas capacidades de Javascript, sin las cuales ya no se concibe una pagina web.

5.1.2 CSS

Con las Hojas de estilo CSS (Cascading Style Sheets) podemos dotar a nuestras páginas HTML con diseños mas estéticos, mas hoy en día que con HTML5, la tendencia es usar el menor número de atributos, dejando así todo lo que se refiere a la maquetación, aspecto, diseño, etc. a CSS. Con las hojas de estilo no solo podremos hacer que nuestros textos y fotos se vean del modo que deseemos, además controlaremos toda la estructura de nuestra página, menús, columnas, etc.

5.1.3 JAVA

Java es un lenguaje de programación y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems.23

El lenguaje de programación Java fue desarrollado originalmente por James Gosling, de Sun Microsystems (constituida en 1983 y posteriormente adquirida el 27 de enero de 2010 por la compañía Oracle),4 y publicado en 1995 como un componente fundamental de la plataforma Java de Sun Microsystems. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son compiladas a bytecode (clase Java), que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.[4]

[4] Fuente Wikipedia Java

5.1.4 JAVASCRIPT

En contra de los lenguajes vistos hasta ahora, JavaScript (abreviado comúnmente "JS") es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas4 aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS). Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo. Con él conseguiremos mayor interacción entre las páginas y podemos usarlo junto a HTML creando HTML dinámico DHTML. Otra de las particularidades de javasript es su flexibilidad, así podemos incluirlo en el código hasta en 3 formas diferentes para crear la página

5.1.5 MySQL

MySQL es un sistema de gestión de bases de datos (DBMS, por sus siglas en inglés) de código abierto desarrollado por Oracle. Se ha ganado su lugar en el mundo digital como una base de datos relacional que permite almacenar, organizar y recuperar datos de manera eficiente. MySQL es utilizado por una amplia variedad de organizaciones y aplicaciones en todo el mundo.

Además, utiliza el lenguaje SQL (Structured Query Language) para acceder y manipular los datos.[5]

5.1.6 JSON

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript.

JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.[6]

[5] Fuente Arsys mysql
[6] Fuente Json.org json

5.2 Herramientas utilizadas

5.2.1 Dia Diagram

Creación de diagrama UM, modelo entidad relación, diagramas de casos de uso

5.2.2 Eclipse Java Web Developers

Eclipse es un entorno de desarrollo software. Eclipse fue utilizado para el desarrollo del back-end con Java

5.2.3 Apache Tomcat 10:

Permite la generación de un servidor que gestiona la aplicación java

5.2.4 Java doc

Documentación de las clases utilizado en eclipse.

5.2.5 Git

Para almacenar archivos de código y del proyecto

5.2.6 VISUAL CODE

Para el desarrollo del Front-end (HTML+ CSS + JAVASCRIPT)

5.3 Detalles implementación

En esta sección vamos a comentar algunas partes más destacadas del código de nuestro portal.

5.3.1 Interfaz Front-end (índex)

Índex será la primera página de nuestra aplicación web. A continuación, vamos a ensenar una parte del código HTML utilizado para diseñar la pagina y una parte del código CSS utilizado para permitir un diseño más estético.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Asociacion contra el cáncer de testículo</title>
<link rel="stylesheet" href="css/style.css">     <!-- para relacionar nuestro codigo css con esta pagina -->

 <!-- si queremos meter un script JAVA -->
<script>
</script>

</head>
<body>


<header><!-- Header -->
    <div class="banner">
        <img src="img/Logo.png" alt="logoheader" class="headerLogo">
    </div>
<header>


<nav><!-- Navigacion bar menu  -->
      <div class="navbar">
        <a href="index.html">Inicio</a>
        <a href="noticias.html">Noticias</a>
        <a href="TiendaSolidaria.html">Tienda Solidaria</a>
        <a href="ListaCarreraSolidaria2.html">Carreras Solidarias</a>
        <a href="Login.html">Iniciar session</a>
        <a href="InscripcionUsuario.html">Registrame</a>
      </div>
 </nav>

<main>
    <section><!-- section -->

        <div class="card-container">
                    <div class="card">
                        <img src="img/NOTICIAS.png" alt="noticias" class="noticias">
                            <div class="card-content">
                                <a href="noticias.html"><button class="cart-btn">Acceder</button></a>
                            </div>
                    </div>

                    <div class="card">
                        <img src="img/CARRERAS.png" alt="carreras" class="carreras">
                            <div class="card-content">
                                <a href="ListaCarreraSolidaria2.html"><button class="cart-btn">Acceder</button></a>
                            </div>
                    </div>

                    <div class="card">
                        <img src="img/3.png" alt="tiendaSolidaria" class="tiendaSolidaria">
                            <div class="card-content">
                                <a href="FichaProduction.html"><button class="cart-btn">Acceder</button></a>
                            </div>
                    </div>
        </div>
    </section>
</main>

<footer><!-- Footer -->
    <div class="footer">
        <h2>2024 Asociacion contra el cáncer de testículo - © Copyright
            2024 - Diseñado por Renaud Bronchart</h2>
    </div>
</footer>
</body>
</html>

```

Léxico de nuestra página web

- **Html** : Es el elemento raíz del documento. Todo el contenido del sitio web está contenido dentro de este elemento.
-
- **Head**: Contiene el titulo, y el link que permite relacionar nuestro código CSS y si queremos, los scripts de JavaScript como comentado en la página.
- **Body**: Este elemento será el contenido principal de nuestra página web.
- **Header**: Este elemento será el contenido para contener el logo del sitio.
- **Nav**: Este elemento se utiliza para contener la barra de navegación del sitio web
- **Main**: Este elemento contiene el contenido principal único a la página.
- **Section**: Se utiliza para agrupar contenido relacionado. Dentro de una sección, normalmente se encuentran elementos de encabezado y otros bloques de contenido.
- **Footer**: Este elemento contiene información acerca del documento como autor, derechos de autor y enlaces a políticas de privacidad y términos de servicio.
- **Script**: Este elemento se utiliza para incrustar o referenciar código JavaScript en el documento.

Léxico no utilizado en esta página web pero útil

- **Aside**: Se utiliza para contener contenido que está relacionado con el contenido principal, pero que puede ser considerado separado del contenido principal.
- **Label**: Este elemento se utiliza para proporcionar una descripción a los elementos de entrada de un formulario.

En la siguiente página, se podría ver una parte del código CSS comentada para entender mejor cada código para cada parte de nuestra pagina web.

```css
/* configuration de nuetro body page */
body {
  font-family: Arial;
  margin: 0;
  background-color: #ffffff;
}

/* Para header and head/
.banner {
  padding: 30px;
  text-align: center;
  background: #fff;;
  color: white;
}

.headerlogo{
  width: 50%;
  height: auto;
}

/* the nav bar (menu) */
.navbar {
  margin-bottom: 1px;
  position: relative;
  display: flex;
  justify-content: center;
  background-color: #fff;
  border-bottom: 1.5px solid #d1d0d0;
  border-top: 3px solid #d1d0d0;
}

.navbar a {
  color: rgb(0, 0, 0);
  padding: 20px 20px;
  text-decoration: none;
  text-align: center;
}

/* para cambiar el color del menu */
.navbar a:hover {
  color: #4070f4;
  font-weight: bold;
}
```

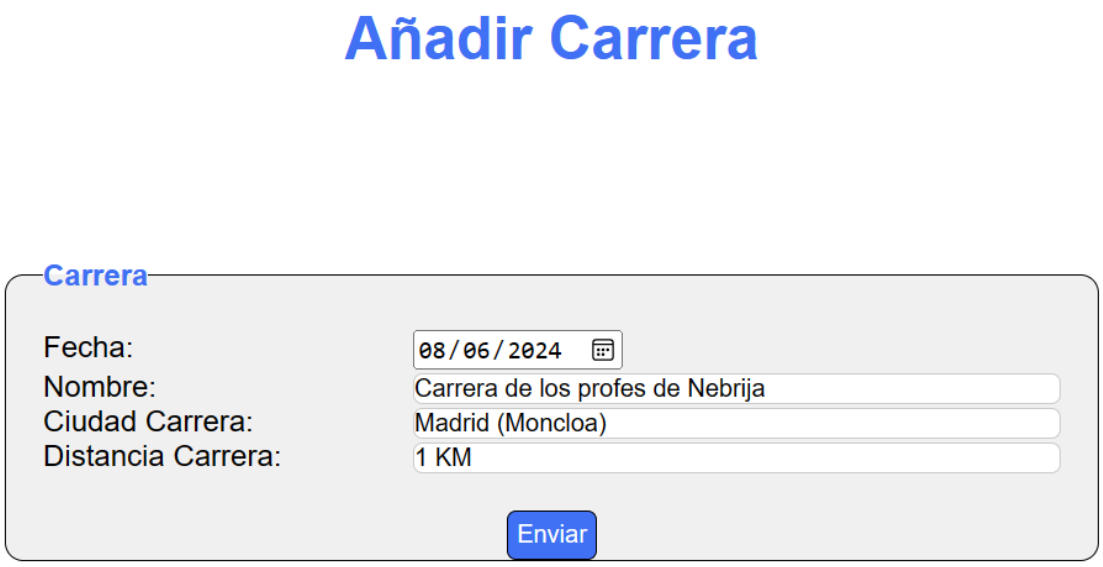
5.3.2 Añadir Carrera

Uno de los procedimientos más habituales en el portal es la inserción de contenido a través de un formulario, creado a tal efecto, un proceso de validación en JavaScript y guardado en la base de datos para que la información sea persistente.
Como ejemplo hemos cogido “Anadir una carrera”.

Un formulario fue creado para permitir al Gestor de crear una carrera.

El formulario HTML es el siguiente :

```
<h1 class="h2">Añadir Carrera</h1>
    <div class="Containerform">
        <form name="frm" method="post" action="GestionCarrera" id="ListaCarreraGestor">
            <fieldset id="form">
            <legend>Carrera</legend>
            <ol>
                <input type="hidden" id="id" name="id">
                <li><label>Fecha: </label><input type="text" id="fechaCarrera" name="fechaCarrera" size="25" /></li>
                <li><label>Nombre: </label><input type="text"  id="nombreCarrera" name="nombreCarrera" size="25" /></li>
                <li><label>Ciudad Carrera: </label><input type="text" id="ciudadCarrera" name="ciudadCarrera" size="25"  /></li>
                <li><label>Distancia Carrera: </label><input type="text" id="distanciaCarrera" name="distanciaCarrera" size="25"  /></li>
            </ul>
            <p align="center"> <input type="submit" name="submit" class="btn"  onclick="validarFormulario()" value="Enviar"></p>
            </fieldset>
            </form>
    </div>
```



El proceso de inserción de una carrera se iniciará por rellenar el formulario por el gestor.

Este envía una petición por post (method=”post”) como se puede ver en la captura del formulario a un Servlet que hemos creado y que se llama GestionCarrera.

El constructor de la clase carrera va a crear un objeto con los datos de la carrera y mediante el método de insertar se va a proceder a la inserción de los datos.

```
    public Carrera(String fechaCarrera, String nombreCarrera, String ciudadCarrera, String distanciaCarrera) {
        this.fechaCarrera = fechaCarrera;
        this.nombreCarrera = nombreCarrera;
        this.ciudadCarrera = ciudadCarrera;
        this.distanciaCarrera = distanciaCarrera;
    }
```

```
public class Carrera {

    private int id;
    private String fechaCarrera;
    private String nombreCarrera;
    private String ciudadCarrera;
    private String distanciaCarrera;
```

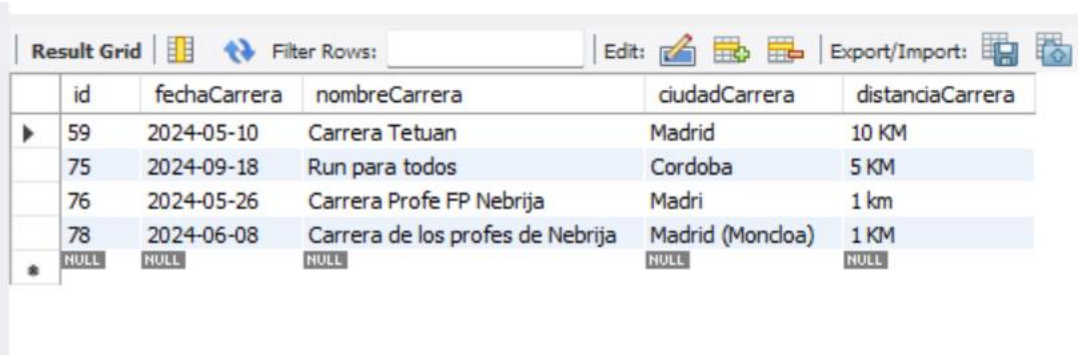
```
public void Insertar() throws SQLException {
    daoCarrera dao = new daoCarrera();
    dao.Insertar(this);
}
```

Mejor hacerlo con Singleton

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub

    String fechaCarrera = request.getParameter("fechaCarrera");
    String nombreCarrera = request.getParameter("nombreCarrera");
    String ciudadCarrera = request.getParameter("ciudadCarrera");
    String distanciaCarrera = request.getParameter("distanciaCarrera");
    String id = request.getParameter("id");

    Carrera c;

    try {

        c =  new Carrera(fechaCarrera, nombreCarrera, ciudadCarrera, distanciaCarrera);
        if(id == "") {

            daoCarrera dao = new daoCarrera();
            dao.Insertar(c);
```

```
public void Insertar(Carrera c) throws SQLException{

    String sql = "INSERT INTO carreras (fechaCarrera, nombreCarrera,ciudadCarrera,distanciaCarrera) VALUES (?,?,?,?)";
    PreparedStatement ps = con.prepareStatement(sql);
    ps.setString(1, c.getFechaCarrera());
    ps.setString(2, c.getNombreCarrera());
    ps.setString(3, c.getCiudadCarrera());
    ps.setString(4, c.getDistanciaCarrera());

    int filas = ps.executeUpdate();
    ps.close();
}
```

Esos datos van a poder estar en la BDD como se puede ver con el ID:78 que es lo que hemos añadido cuando el gesto ha completado el formulario.



Result Grid | Filter Rows: | Edit: | Export/Import:

id	fechaCarrera	nombreCarrera	ciudadCarrera	distanciaCarrera
59	2024-05-10	Carrera Tetuan	Madrid	10 KM
75	2024-09-18	Run para todos	Cordoba	5 KM
76	2024-05-26	Carrera Profe FP Nebrija	Madri	1 km
78	2024-06-08	Carrera de los profes de Nebrija	Madrid (Moncloa)	1 KM
NULL	NULL	NULL	NULL	NULL

5.3.3 Listar carreras

Un método fue creado para ir a la base de datos que permite devolver un Arraylist de la carreras creadas y insertadas en la BDD.

```
public ArrayList<Carrera> listar() throws SQLException {

    String sql = "SELECT * FROM carreras";
    PreparedStatement ps = con.prepareStatement(sql);

    ResultSet result = ps.executeQuery();

    ArrayList<Carrera> ls = null;

    while(result.next()) {
        if(ls == null) {
            ls = new ArrayList<Carrera>();
        }
        ls.add(new Carrera(result.getInt(1),result.getString(2),result.getString(3),result.getString(4),result.getString(5)));
    }
    return ls;
}
```

Como fue explicado en el punto de **3.1 Arquitectura cliente – servidor**, utilizamos una librería JSON .

Es decir, un método String fue creado para poder listar los datos en formato Json.

```
// creamos una libreria para poder listar con Json //
public String listarJson() throws SQLException {

    String json = "";
    Gson gson = new Gson();

    json = gson.toJson(this.listar());

    return json;
}
```

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub

    PrintWriter out = response.getWriter();


        daoCarrera Carreras;
        try {
            Carreras = new daoCarrera();
            out.print(Carreras.listarJson());

        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
```

Para poder mostrar esta lista, necesitamos pasar los datos de Json via HTML.

Por eso, vamos a utilizar un script con JavaScript que permite de mostrar los datos.

Entonces, cuando la pagina HTML se va a cargar, para poder mostrar la lista de carreras, unas funciones en JavaScript fue creado.

```
<script type="text/javascript">
    function llamada(){
        fetch('GestionCarrera?op=1')
        .then(response => response.json())
        .then(data => pintarTabla(data))
    }
    function pintarTabla(datos){

        let html = "<table>";
        html +="<thead>"
        html +="<th>Fecha</th>";
        html +="<th>Nombre</th>";
        html +="<th>Ciudad</th>";
        html +="<th>Distancia</th>";
        html +="<th></th>";
        html +="</thead>"
        for(let i=0;i<datos.length;i++){

                html +="<td>"+datos[i].fechaCarrera+"</td>";
                html +="<td>"+datos[i].nombreCarrera+"</td>";
                html +="<td>"+datos[i].ciudadCarrera+"</td>";
                html +="<td>"+datos[i].distanciaCarrera+"</td>";
                html +="<td><a href='InscripcionCarrera.html'>Inscripción</a></td>";
                html +="</tr>";
        }
        html +="</table>";
        document.getElementById("listado").innerHTML = html;
        console.log(datos);
    }
  window.onload = function() {
        llamada();
    }
    </script>
```

Gracias a este código de JavaScript, y los pasos hechos anteriormente, vamos a poder mostrar la lista de carreras que el gestor ha creado.

Lista carreras

#	fecha	nombre	ciudad	distancia			
59	2024-05-10	Carrera Tetuan	Madrid	10 KM	Editar	Borrar	Lista Corredores
75	2024-09-18	Run para todos	Cordoba	5 KM	Editar	Borrar	Lista Corredores
76	2024-05-26	Carrera Profe FP Nebrija	Madri	1 km	Editar	Borrar	Lista Corredores
78	2024-06-08	Carrera de los profes de Nebrija	Madrid (Moncloa)	1 KM	Editar	Borrar	Lista Corredores

5.3.4 Connexion a la base de datos

Hemos creado un Objeto conexión genérico que va a permitir la conexión a la BDD.

Hacemos la conexión a través de una instancia que gestiona la clase.

```
2
3  import java.sql.Connection;
7
8  public class DBConexion {
9
10     // attributo static para guardar la direccion que queremos(jdbc:mysql://) + //
11     //  Puerto de connexion(3306) + nombre base de datos(assoccontracancer) //
12     public static final String JDBC_URL = "jdbc:mysql://localhost:3306/assoccontracancer";
13
14
15     public static Connection instance = null;
16
17
18     // Metodo para devolver la instancia con la connexion a la base de datos //
19
20     public static Connection getConexion() throws SQLException {
21
22         // Establecemos la conexion y almacenarla en la variable instance //
23         if(instance == null) {
24
25             Properties props = new Properties();
26             props.put("user", "root");
27             props.put("password", "");
28             props.put("charset", "UTF-8");
29
30
31             instance = DriverManager.getConnection(JDBC_URL, props);
32         }
33         return instance;
34
35     }
36
37
38
39
40
41 }
```

5.3.5 DAO

Utilizamos el patrón DAO(Data Access object) en nuestro proyecto.
El DAO va a permitir la interactuación con la base de datos.(Consultas, inserciones, actualización, eliminación de datos).

```java
/**
 * Clase daoCarrera
 * @author: Renaud Bronchart
 * @version: 15/05/2024 v1.0
 */

public class daoCarrera {

    /**
     * Generar un atributo tipo Connection llamado con
     */

    public static Connection con = null;

    /**
     * Constructor DaoCarrera para que se hace la connexion
     */

    public daoCarrera() throws SQLException {

        this.con = DBConexion.getConexion();

    }

    /**
     * Metodo de insercion en la BDD  del objeto Carrera
     */
    public void insertar(Carrera c) throws SQLException{

        String sql = "INSERT INTO carreras (fechaCarrera, nombreCarrera,ciudadCarrera,distanciaCarrera) VALUES (?,?,?,?)";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setString(1, c.getFechaCarrera());
        ps.setString(2, c.getNombreCarrera());
        ps.setString(3, c.getCiudadCarrera());
        ps.setString(4, c.getDistanciaCarrera());

        int filas = ps.executeUpdate();
        ps.close();
    }


    /**
     * Metodo para buscar el ID
     * @return tipo int
     */

    public Carrera obtenerPorID(int idCarrera) throws SQLException {
        String sql = " SELECT * FROM carreras where idCarrera=?";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setInt(1, idCarrera);

        ResultSet result = ps.executeQuery();

        result.next();

        Carrera c = (new Carrera(result.getInt(1),result.getString(2),result.getString(3),result.getString(4),result.getString(5)));

        return c;

    }
```

5.3.6 Zona usuarios

Una zona usuario ha sido creado y que permite distinguir un usuario administrador y un usuario registrado.

Para acceder a esta parte, el usuario debería registrarse vía un formulario.

Método para hacer el login

```
public Usuario logear(Usuario u, String pass) throws SQLException {


    String sql = "SELECT * FROM usuarios WHERE mail=? AND pass=?";
    PreparedStatement ps = con.prepareStatement(sql);
    ps.setString(1, u.getMail());
    ps.setString(2,pass);

    ResultSet rs = ps.executeQuery();

    rs.next();

    Usuario aux = new Usuario(rs.getInt(1), rs.getString(2),rs.getString(3),rs.getString(4),rs.getInt(5));

    return aux;

}
```

```
public boolean logeo(String pass) throws SQLException {

    boolean ok = false;

    daoUsuario dao = new daoUsuario();
    Usuario aux = dao.logear(this, pass); // bd

    if(aux != null) {
        ok=true;
        this.setId(aux.getId());
        this.setNombre(aux.getNombre());
        this.setApellidos(aux.getApellidos());
        this.setMail(aux.getMail());
        this.setPermiso(aux.getPermiso());
    }
    return ok;
}
```

Según el usuario registrado y el permiso dado a este usuario, se dirigirá a una página especifica.

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub

    String mail = request.getParameter("mail");
    String pass = getMD5(request.getParameter("pass"));

    Usuario u = new Usuario();
    u.setMail(mail);


    //proteccion

    try {
        if(u.logeo(pass)) {

            sesion = request.getSession();


            sesion.setAttribute("id", u.getId());
            sesion.setAttribute("permiso", u.getPermiso());



            if (u.getPermiso() == 9 ) {

            response.sendRedirect("GestorPortal.html");

            } else if (u.getPermiso() == 1 ) {

                response.sendRedirect("Miperfil.html?id=2&p=2");
            }

        }else {

            response.sendRedirect("Login.html");

        }
```

Para el usuario será a la página de su perfil y para el administrador será a la página de gestión.

ASOCIACIÓN CONTRA EL CÁNCER DE TESTÍCULO

Inicio Noticias Tienda Solidaria Carreras Solidarias GestorPortal

Mi perfil

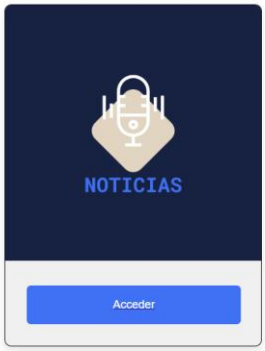
Datos

Nombre:	Soy
Apellidos:	Usuario
Mail:	test@gmail.com

Enviar

5.3.7 Diseño responsive

Para tener la capacidad de ser adaptativo a los diferentes dispositivos como una Tablet o en móvil, hemos codificado una parte del css con el siguiente código.




```
@media screen and (max-width: 600px) {
  .navbar  {
    flex-direction: column;
  }
}

  @media screen and (max-width: 400px) {
  .cardnoticia {
    width: 400px;
  }
}

@media screen and (max-width: 400px) {
  table {
    width: 400px;
  }
}

@media screen and (max-width: 600px) {
  .card-container {
    width: 600px;
  }
}
```

5.3.8 Documentación (JAVADOC)

Un documento JAVAdoc ha sido creado para para ayudar a entender el código creado para que otra persona que tenga que trabajar sobre el proyecto, puede entenderlo fácilmente.

JAVADOC es una herramienta que va a permitir documentar y anotar, de manera sencilla y rápida cada código y principalmente las clases y métodos creado y va a ser una gran utilidad para la comprensión para el desarrollador actual o los próximos.

Ejemplos creados con JAVADOC:

Package modelo

`package modelo`

Classes

Class	Description
Carrera	Clase para el modelo Carrera
Colaborador	Clase para el modelo Colaborador
Corredor	Clase para el modelo Corredor
Noticia	Clase para el modelo noticia
Usuario	Clase para el modelo usuario

Package dao

`package dao`

Classes

Class	Description
daoCarrera	Clase daoCarrera
daoColaborador	Clase dao daoColaborador
daoCorredor	Clase dao daoCorredor
daoNoticia	Clase dao daoNoticia
daoUsuario	Clase dao daoUsuario
DBConexion	Clase dao DBconnexion

Package modelo

Class Carrera

java.lang.Object
modelo.Carrera

```
public class Carrera
extends Object
```

Clase para el modelo Carrera

Version:

15/05/2024 v1.0

Author:

Renaud Bronchart

Constructor Summary

Constructors

Constructor	Description
`Carrera()`	Constructor para generar un objeto vacio
`Carrera(int idCarrera, String fechaCarrera, String nombreCarrera, String ciudadCarrera, String distanciaCarrera)`	Constructor para generar un objeto desde el formulario con INT idCarrera.
`Carrera(String fechaCarrera, String nombreCarrera, String ciudadCarrera, String distanciaCarrera)`	Constructor para generar un objeto desde el formulario sin INT idCarrera.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
void	`actualizar()`	Metodo que permite actualizar el elemento en la base de datos via DAO
void	`borrar(int idCarrera)`	Metodo que permite borrar el elemento en la base de datos via DAO
String	`dameJson()`	Metodo para devolver Json de tipo String
String	`getCiudadCarrera()`	Metodo Para obtener la ciudad de carrera
String	`getDistanciaCarrera()`	Metodo Para obtener la distancia de carrera
String	`getFechaCarrera()`	Metodo Para obtener la fecha de carrera
int	`getidCarrera()`	Metodo Para obtener el id de carrera
String	`getNombreCarrera()`	Metodo Para obtener el nombre de carrera
void	`Insertar()`	Metodo que inserta el elemento en la base de datos via DAO
void	`obtenerPorID(int idCarrera)`	Metodo que permite obtener un ID del elemento en la base de datos via DAO
void	`setCiudadCarrera(String ciudadCarrera)`	Metodo Para establecer la ciudad de carrera
void	`setDistanciaCarrera(String distanciaCarrera)`	Metodo Para establecer la distancia de carrera
void	`setFechaCarrera(String fechaCarrera)`	Metodo Para establecer la fecha de carrera
void	`setidCarrera(int idCarrera)`	Metodo Para establecer el id de carrera

Methods inherited from class java.lang.Object

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Constructor Details

Carrera

```
public Carrera()
```

Constructor para generar un objeto vacio

Carrera

```
public Carrera(int idCarrera,
               String fechaCarrera,
               String nombreCarrera,
               String ciudadCarrera,
               String distanciaCarrera)
```

Constructor para generar un objeto desde el formulario con INT idCarrera.

Parameters:

`idCarrera` - id de la carrera (PK) Atributo numero

`fechaCarrera` - fecha de la carrera Atributo tipo texto

`nombreCarrera` - nombre de la carrera Atributo tipo texto

`ciudadCarrera` - ciudad de la carrera Atributo tipo texto

`distanciaCarrera` - distancia de la carrera Atributo tipo texto

Carrera

```
public Carrera(String fechaCarrera,
               String nombreCarrera,
               String ciudadCarrera,
               String distanciaCarrera)
```

Constructor para generar un objeto desde el formulario sin INT idCarrera.

Parameters:

`fechaCarrera` - fecha de la carrera Atributo tipo texto

`nombreCarrera` - nombre de la carrera Atributo tipo texto

`ciudadCarrera` - ciudad de la carrera Atributo tipo texto

`distanciaCarrera` - distancia de la carrera Atributo tipo texto

Method Details

Method Details

getidCarrera

```
public int getidCarrera()
```

Metodo Para obtener el id de carrera

Returns:
el id en tipo entero.

setidCarrera

```
public void setidCarrera(int idCarrera)
```

Metodo Para establecer el id de carrera

Parameters:
`idCarrera` - Parametro

getFechaCarrera

```
public String getFechaCarrera()
```

Metodo Para obtener la fecha de carrera

Returns:
la fecha de la carrera

setFechaCarrera

```
public void setFechaCarrera(String fechaCarrera)
```

Metodo Para establecer la fecha de carrera

Parameters:
`fechaCarrera` - Parametro

getNombreCarrera

```
public String getNombreCarrera()
```

Metodo Para obtener el nombre de carrera

Returns:
el nombre de la carrera

setNombreCarrera

```
public void setNombreCarrera(String nombreCarrera)
```

5.3.9 Control de versiones

Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Una versión, revisión o edición de un producto, es el estado en el que se encuentra el mismo en un momento dado de su desarrollo o modificación.[7]

Para el proyecto, GitHub fue utilizado como control de versiones.

Tambien, GitKraken fue utilizado.

GitKraken es una herramienta que ayuda a manejar Git de manera sencilla.

[7] Wikipedia : Control_de_versiones

6 Entorno de desarrollo

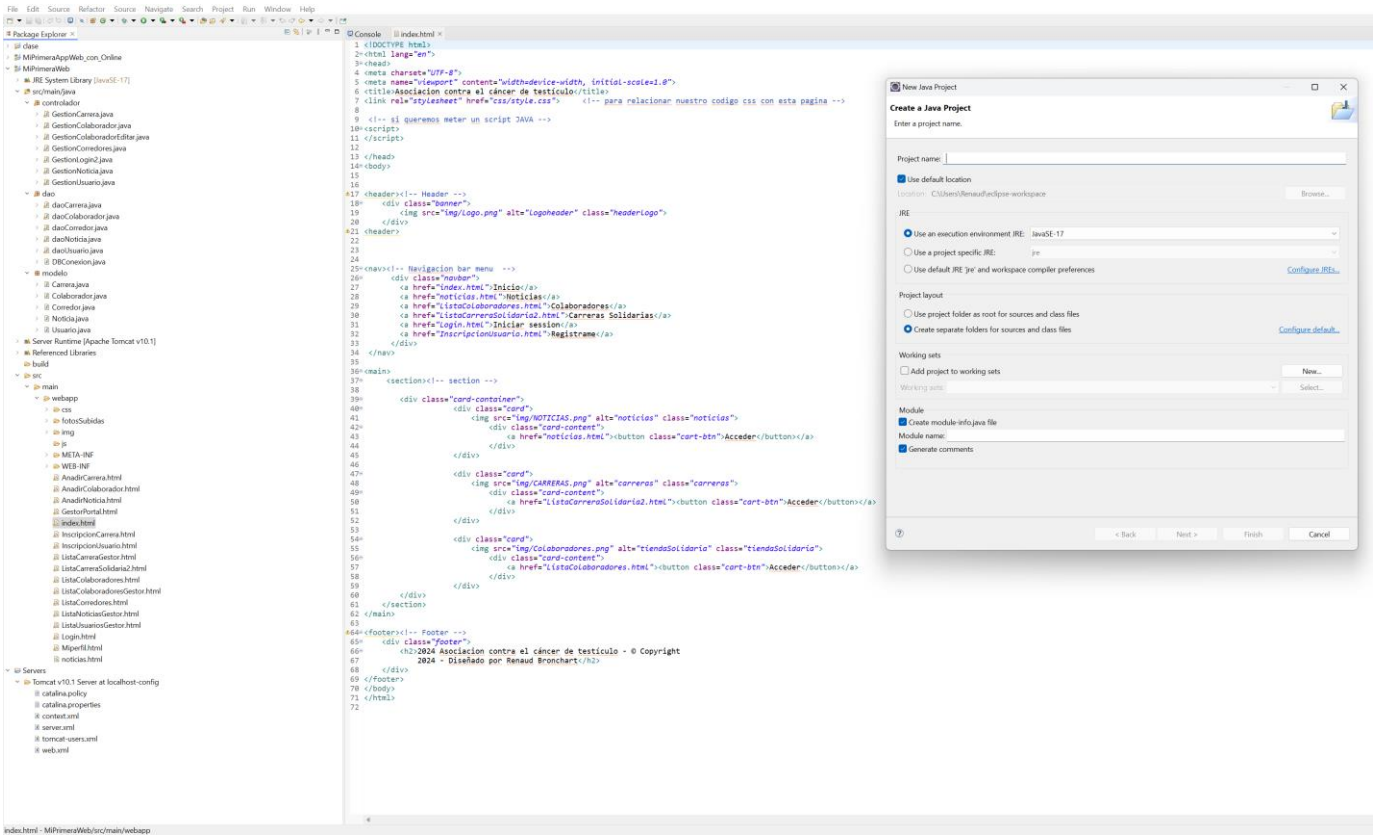
¿Qué es un entorno de desarrollo?

En nuestro caso, un entorno de desarrollo fue un conjunto de herramientas y recursos para poder escribir, probar nuestra aplicación y programa.

Para este proyecto, hemos utilizado Windows como sistema operativo y entonces no hacia falta tener una maquina Virtual y utilizar Ubuntu, pero a continuación se comentará brevemente como instalarlo con OS y con Linux.

6.1 Eclipse

Para poder desarrollar nuestro proyecto, Eclipse IDE fue utilizado.
Eclipse es un entorno de desarrollo software, que me ha permitido escribir y probar mi aplicación web.
Como se puede ver en la capture de pantalla, he podido crear diferentes carpetas que me ha permitido, guardar los códigos de html,CSS,JAVAscript,JAVA. Se puede también ver una carpeta “ Severs” con “ Tomcat”(más informaciones sobre Tomcat en el punto 6.3)



Se puede descargar el software el en siguiente link : https://eclipseide.org/.

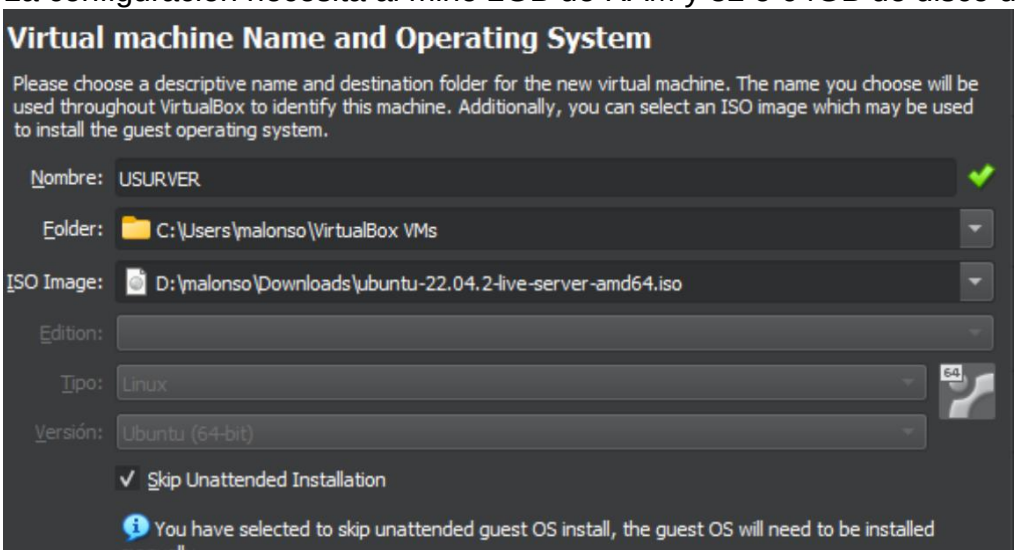
6.2 Máquina Virtual

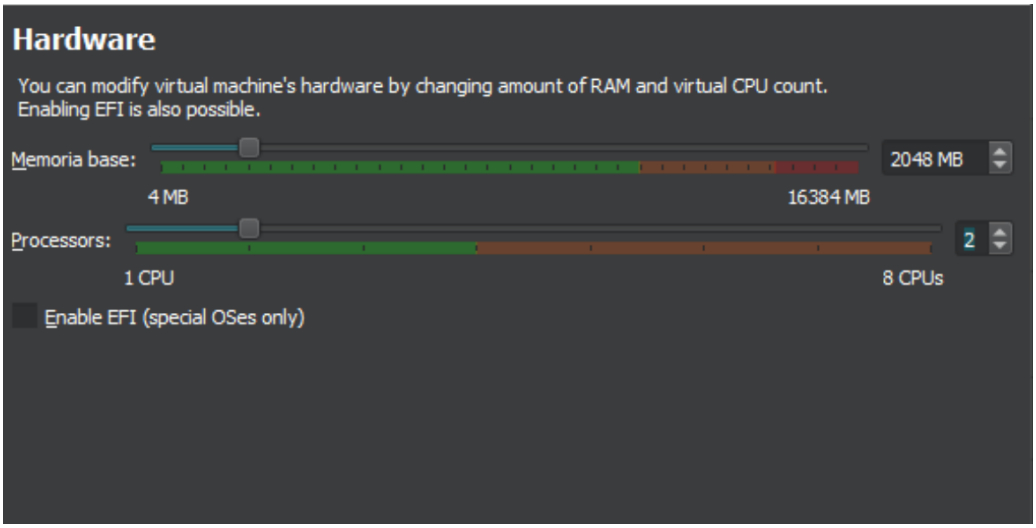
Como comentado antes, no hemos necesitado instalar una maquina virtual para poder trabajar en nuestro proyecto porque estábamos con Windows.

Una maquina virtual es un entorno de software que emula un sistema informático permitiendo de ejecutar un sistema operativo.

Para crear la máquina virtual, Se descarga Virtual box https://www.virtualbox.org/wiki/Downloads.

Luego habrá que crear la máquina virtual y completar les datos requisitos. Habrá que coger el S.O, y también vincular la imagen ISO del la página oficial de Ubuntu.
La configuración necesita al mino 2GB de RAM y 32 o 64GB de disco duro.



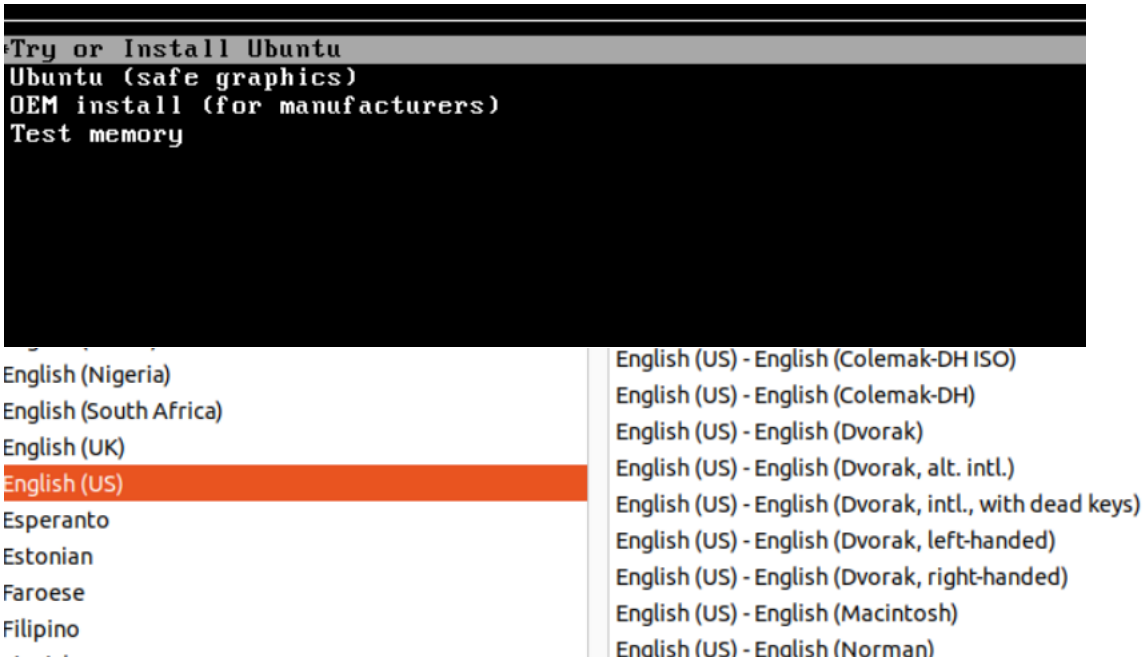


6.3 Ubuntu

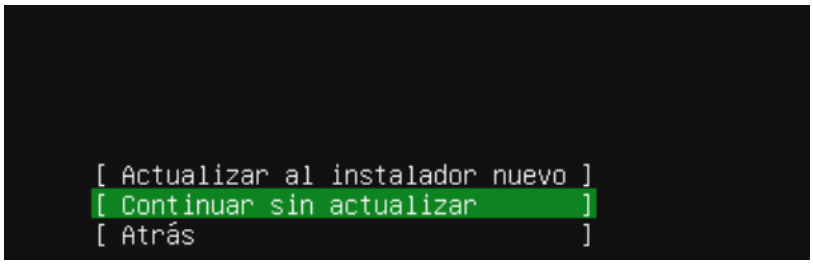
Ubuntu es una distribución GNU/Linux basada en Debian GNU/Linux, que incluye principalmente software libre y de código abierto. Puede utilizarse en ordenadores y servidores. Está orientado al usuario promedio, con un fuerte enfoque en la facilidad de uso y en mejorar la experiencia del usuario.[8]

Después de haber hecho instalación de la Maquina virtual con la imagen, vamos a instalar Ubuntu.

Primero, se va a iniciar la pantalla de instalación para instalar y seleccionar idioma.



Se podría elegir entre la instalación Normal y Mínima y también si queremos descargar la última versión y actualizarla o instalar sin actualizar.

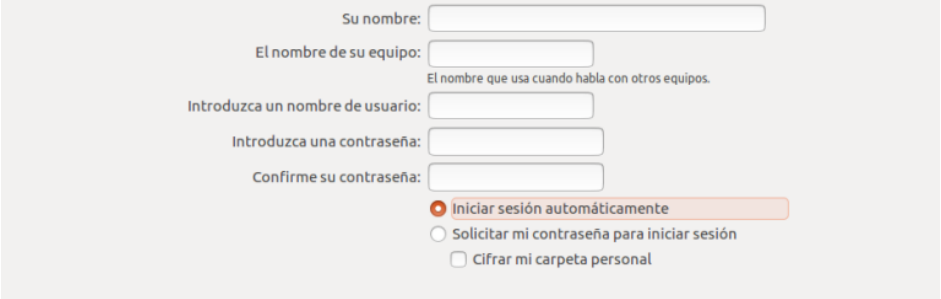


También, vamos a tener la opción de elegir la configuración de red. Se aconseja de optar por la opción por defecto.

[8] https://es.wikipedia.org/wiki/Ubuntu

Luego, se podría ver la configuración de las particiones de instalación, el tipo de instalación y la ubicación de donde el usuario se encuentra.

Finalmente, llegamos a la página de identificación:

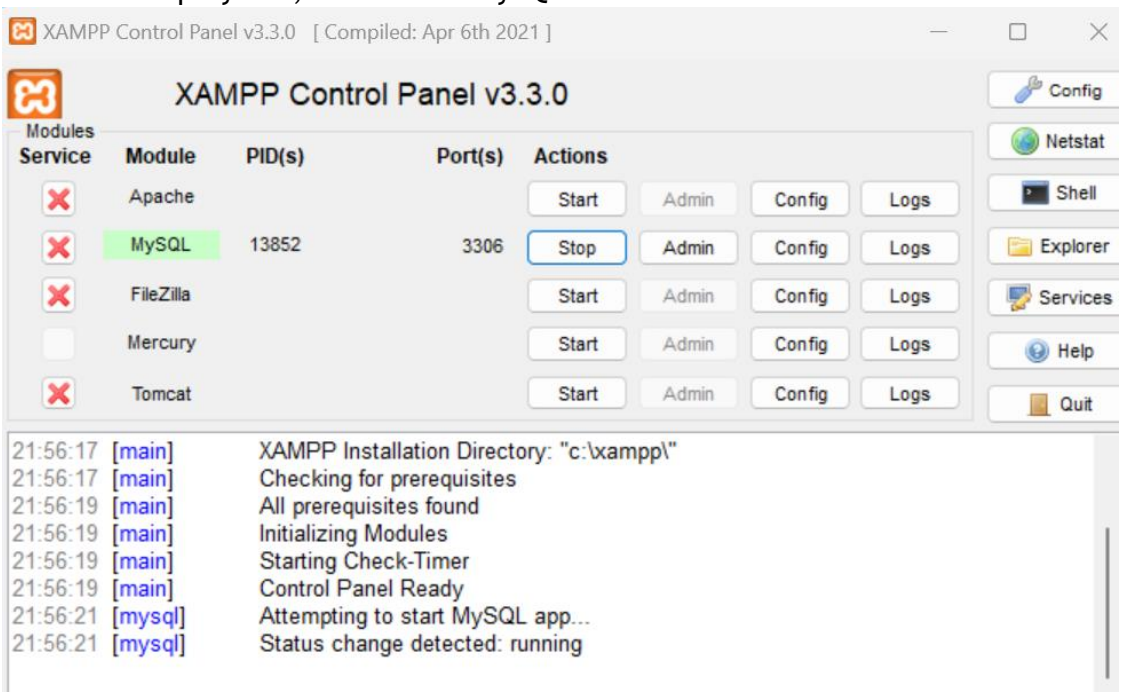


Se rellenan los datos y de hace la instalación.

6.4 XAMP

Para nuestro proyecto XAMP fue utilizado. Xamp es una distribución de software que proporciona un entono de desarrollo de servidores como MySQL,Apache,etc..

Para nuestro proyecto, únicamente MySQL fue utilizado.



Se puede descargar XAMP vía https://www.apachefriends.org/es/index.html .

Con Windows, Instalación es bastante fácil, se puede descargar directamente la versión según el S.O vía la pagina web y seguir instalación del programa.

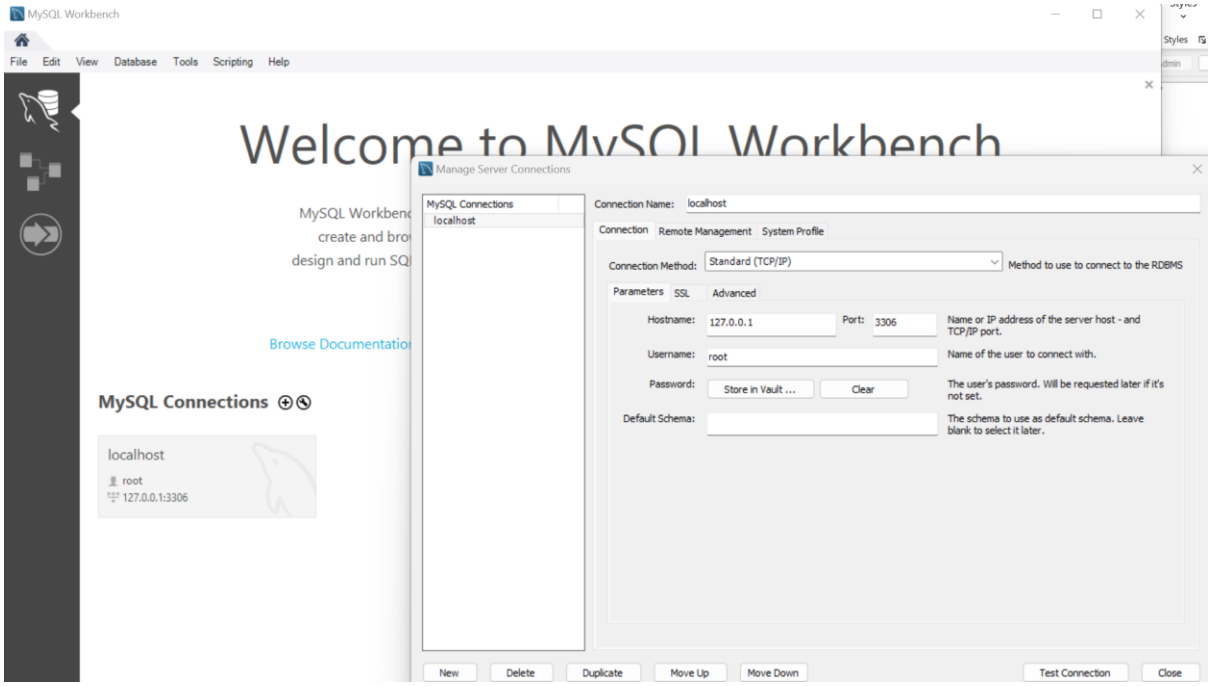
Para Ubuntu, los pasos para configurar e instalar XAMPP son los siguientes:

- Descargar https://www.apachefriends.org/xampp-files/7.2.9/xampp-linux-x64-7.2.9-0-installer.run
- Ejecutar sudo chmod 777 xampp-linux-x64-7.2.9-0-installer.run y sudo ./xampp-linux-x64-7.2.93-0-installer.run
- Luego para lanzar LAMPP, ejecutar sudo /opt/lampp/lampp start y seguir los pasos de instalación

6.5 MYSQL

MySQL es nuestro gestor de base de datos.

Después de instalar el programa, hemos creado a un local host con una dirección IP 127.0.0.1 y un Port 3306, que va a permitir la conexión nuestro programa y hacer la interacción. No hemos puesto ninguna contraseña y entonces no haría faltar configurarla como explicado en el punto 5.3.4



Para Linux, habrá que lanzar el comando **sudo apt install mysql-server** para hacer instalación.

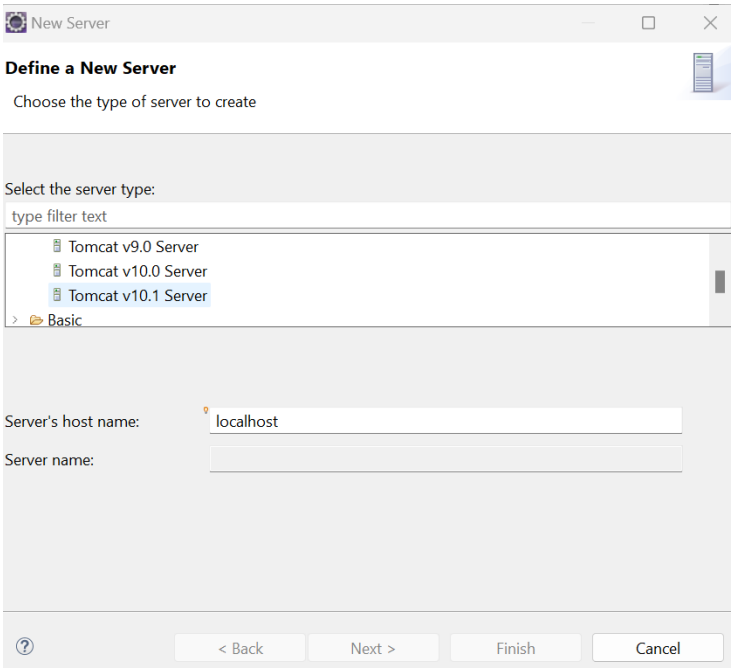
6.6 Tomcat

Apache Tomcat es un contenedor de servlets de código. Tomcat sirve como un servidor web y este diseñado para implementar aplicaciones basas en tecnologías Java.

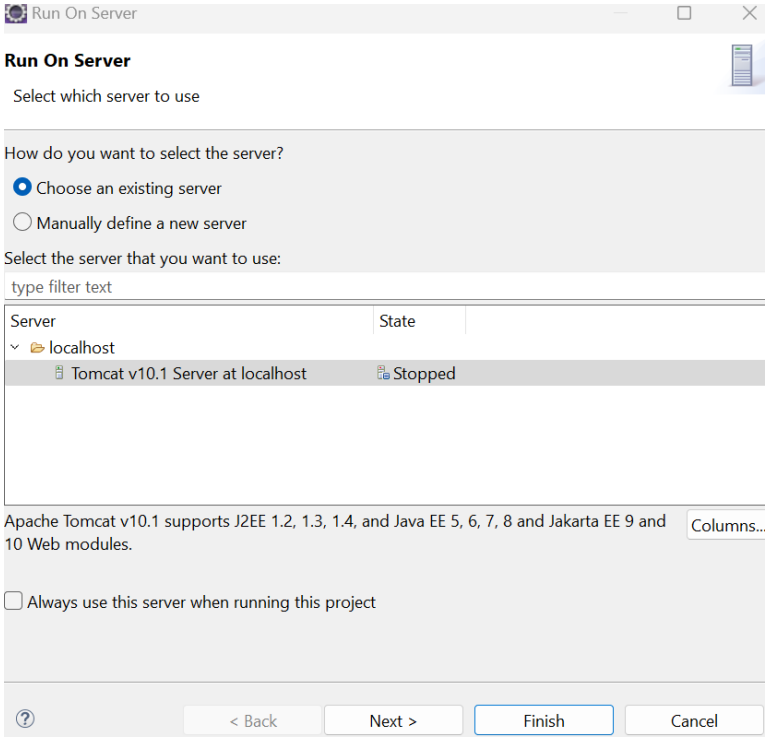
Hemos instalado un server tomcat de manera local para ejecutar el proyecto.

Se puede descargar la versión vía la pagina https://tomcat.apache.org/ (la ultima versión).

Habrá que extraer el contenido en una carpeta.
En esta carpeta, se podría añadir a la librería las librerías de conector y de GSON.
Luego, con Eclipse, se podría crear un server con la ultima versión de tomcat descargado.



Al lanzar nuestro programa y la nuestra aplicación web, se seleccionar el server configurado para poder visualizar nuestra pagina web.



Para Linux :

- utilizar **sudo apt install tomcat9 tomcat9-admin**
- Despues de haber instalado Tomcat se puede comprobar el estado con “ **sudo systemctl status tomcat9 “**

7 Versión V2 de aplicación web

Como muchas aplicaciones web, siempre se puede modificar unos elementos y mejorar las cosas.

Es por eso que, aunque el proyecto ha sido terminado, quiero dejar unas ideas para mejorar utilización de aplicación web y también para poder dar mas opciones a los utilizadoras.

Lista de los elementos a modificar, mejorar o añadir para la versión 2

- Añadir una tienda de productos y permitir las compras y donaciones
- Añadir los comentarios en las noticias
- Modificación del diseño (CSS)
- Solicitar información vía un formulario
- Permitir a usuario de ver su ultimas y futuras carreras en su perfil

8 Conclusiones

Participar y terminar este proyecto es un reto importante en muchos sentidos y una satisfacción porque significa que la mayoría de los requisitos fueron aplicados y que he cumplido las diferentes etapas del proyecto para generar un producto final.

La realización del proyecto fue una terea difícil debito a los diferentes obstáculos encontrados como la gestión de tiempos, la metodología de trabajo, poner en práctica los conocimientos aprendidos.

Antes de empezar el proyecto, muchos aspectos de html,CSS,JAVA,MYSQL, etc, eran complejos.

Durante el proceso he podido aprender, comprender y perfeccionar más los conocimientos adquiridos durante el curso. También fue una oportunidad para aplicar los conocimientos y habilidades.

Además de utilizar los diferentes documentos que tenemos a disposición gracias al curso, una propia investigación personal fue necesaria, y uno de los recursos utilizados fue internet.

En conclusión, la realización de este proyecto ha permitido meter en práctica los conocimientos obtenidos durante todo el año y aprender como manejar un proyecto con sus problemas y acercase el más posible a los proyectos que se puede tener en una empresa.

9 Bibliographia

Para la realización de esta memoria se han consultado las siguientes fuentes de contenido.

- Caso de uso - Wikipedia, la enciclopedia libre
- https://es.wikipedia.org/wiki/Diagrama_de_clases
- UML: Diagrama de Secuencia – INGENIERÍA DEL SOFTWARE (wordpress.com)
- Cliente-servidor - Wikipedia, la enciclopedia libre
- ¿Qué es MySQL? Explicación y características | Arsys
- JSON
- Ubuntu - Wikipedia, la enciclopedia libre
- Control de versiones - Wikipedia, la enciclopedia libre

GitHub

VideoProyecto