PORTAL WEB DE GESTIÓN DE CENTRO DE ENTRENAMIENTO – True Fit Gym

Adrián Rodríguez Díaz
Grado de ingeniería Informática
Desarrollo Web

Gregorio Robles Martínez
Santi Caballe Llobet

17/01/2023

FICHA DEL TRABAJO FINAL

Título del trabajo:	Portal web para la gestión de centro de entrenamiento – True Fit Gym
Nombre del autor:	Adrián Rodríguez Díaz
Nombre del consultor:	Gregorio Robles Martínez
Fecha de entrega (mm/aaaa):	01/2023
Área del Trabajo Final:	Desarrollo Web
Titulación:	*Ingeniería Informática*

Resumen del Trabajo (máximo 250 palabras):

True Fit Gym ha surgido como consecuencia de las limitaciones de aforos en los gimnasios, producto de la necesidad del distanciamiento social que provocó la pandemia COVID-19. Actualmente, la mayoría de los gimnasios cuentan con una aplicación o sitio web donde los usuarios pueden y deben registrarse si desean asistir a una clase colectiva, o disfrutar de alguna de las zonas disponibles del centro.

TFG Se trata de un sitio web capaz de gestionar las actividades diarias de un gimnasio, permitiendo el registro de nuevos usuarios, dar de alta nuevas actividades colectivas controlando el aforo máximo de cada una, los monitores que serán asignados, las salas dónde se realizarán, y muchas más opciones.

Además, el sitio web está pensado para el acceso de 3 tipos de perfiles diferentes (usuarios, monitores, administradores), donde cada uno tendrá asignadas distintas funciones y permisos durante la navegación.

Ha sido desarrollado mediante Java con Spring Boot v3.0, ya que es una de las herramientas más utilizadas en los ultimos tiempos gracias a su facilidad de despliegue, su inyección de dependencias y su inversión de control. La base de datos es MySQL y se realiza la persistencia mediante Java Persistence Api. También se ha utilizado Thymeleaf en la capa frontal como motor de plantillas para HTML y Bootstrap 5 para el diseño.

El objetivo de este proyecto es crear un entorno web responsive, amigable e intuitivo, que cumpla con los requerimientos básicos de un gimnasio a la hora de gestionar sus actividades y usuarios.

Abstract (in English, 250 words or less):

True Fit Gym has emerged because of capacity limitations in gyms, a product of the need for social distancing caused by the COVID-19 pandemic. Currently, most gyms have an application or website where users must register if they wish to attend a collective class or use any of the available areas of the center.

TFG It is a website capable of managing the daily activities of a gym, allowing the registration of new users, registering new collective activities by controlling the maximum capacity of each one, the instructors that will be assigned, the rooms where they will be carried out, and many more options.

In addition, the website is designed for access by 3 different types of profiles (users, monitors, administrators), where each one will be assigned different functions and permissions while browsing.

It has been developed using Java with Spring Boot v3.0, because it is one of the most used tools in recent times thanks to its ease of deployment, its dependency injection, and its inversion of control. The database is MySQL and persistence is done using the Java Persistence Api. Thymeleaf has also been used in the front layer as a template engine for HTML and Bootstrap 5 for design.

The objective of this project is to create a responsive, friendly, and intuitive web environment that meets the basic requirements of a gym when it comes to managing its activities and users.

Palabras clave (entre 4 y 8):

Gym, Java, Spring, Maven, Thymeleaf, MySQL, Bootstrap

Índice

Lista de figuras

Lista de tablas

1. Introducción

1.1 Contexto y justificación del Trabajo

La pandemia provocada por la COVID-19 ha generado una cantidad abrumadora de consecuencias negativas, sin embargo, el mundo de las tecnologías y particularmente las orientadas al desarrollo web, han experimentado un gran crecimiento, esto se debe en gran parte a la necesidad de la gran mayoría de negocios a seguir desarrollando sus actividades, y además, cumplir con las nuevas normas establecidas por las autoridades.

Entre los negocios afectados, se encuentran los gimnasios o centros de entrenamiento, que se han visto en la obligación de respetar los aforos máximos establecidos, a respetar el distanciamiento social, el uso de nuevas herramientas de protección, etc.

Para poder concurrir a estos centros hoy en día, es casi obligatorio haber realizado previamente una reserva, ya sea de clases, de sala o incluso de vestuarios. Por los que muchos de estos centros han optado por realizar este tipo de gestión mediante sitios web o aplicaciones móviles, de fácil acceso y registro para los usuarios, para que además puedan también obtener información relevante como horarios, profesores asignados a las clases (Body Pump, Zumba, Spinning, etc.).

Todo esto ha generado una buena oportunidad para los desarrolladores de sitios web de poder trabajar y colaborar con este tipo de negocios para poder cubrir estas nuevas necesidades.

1.2 Objetivos del Trabajo

El objetivo principal será la creación de una solución web capaz de gestionar las tareas principales de un gimnasio o centro de entrenamiento.

La aplicación contará con 3 roles principales muy diferentes, uno de administrador, que será el encargado de toda la gestión del portal; otro rol para los usuarios, que será el que pueda acceder a leer la información que el administrador ha generado, a las reservas de clases o salas y a sus datos principales; y por último, un rol para los monitores, que podrán acceder a la aplicación para consultar sus horarios y sus clases asignadas.

Por lo tanto, las tareas principales detectadas inicialmente para cada uno de los roles serán:

Administrador:

- Control de usuarios
- Creación de actividades

- Alta de nuevos monitores
- Alta de nuevos administradores
- Modificación de clases
- Revisar contactos hacia el centro

Usuarios:

- Reserva de clases
- Acceso a historial de reservas
- Registrar entrenamientos
- Modificación de datos personales
- Contactar con centro

Monitores:

- Consulta de actividades asignadas
- Consultar usuarios inscriptos
- Navegación por todo el sitio web

Para la creación de este proyecto será necesaria la utilización de los conocimientos adquiridos a lo largo de la realización del grado, para poder planificar y realizar efectivamente un proyecto desde el comienzo hasta el final "*end-to-end"*. Realizando un correcto análisis de requerimientos, análisis funcional, diseño técnico, codificación y *testing*, hasta el momento de entrega.

Además, cabe destacar dentro de los objetivos personales, el valor añadido que me proporcionará el aprendizaje adquirido al trabajar con diferentes herramientas tecnológicas que se utilizan actualmente en el mercado, desde las herramientas de planificación de proyecto, las de control de versiones de codificación (Git), hasta las utilizadas para realizar el desarrollo tanto *Back-End* como *Front-End*.

1.3 Enfoque y método seguido

Este proyecto, como mencionamos anteriormente, nace de una necesidad real a la que se han tenido que enfrentar este tipo de negocios, que se han visto obligados a recurrir en muchos casos a aplicaciones genéricas, sin reflejar la identidad del centro y que terminan siendo poco intuitivas para el usuario final.

La metodología de trabajo elegida para el desarrollo de este TFG será la de *Waterfall* (cascada), que consiste en desarrollar el proyecto de forma secuencial, comenzando por las fases de análisis y diseño y terminando por con las fases de prueba y puesta en marcha en producción. Esta metodología sigue la siguiente secuencia:

- Definición de requisitos
- Análisis
- Diseño
- Implementación

- Verificación (*Testing*)
- Despliegue

He optado por este tipo de metodología, que si bien no es la más moderna (ya que están las metodologías ágiles), me parece la más efectiva para un proyecto de desarrollo que será gestionado por una única persona.

Las tecnologías que se van a aplicar para el desarrollo de este proyecto serán las siguientes:

- **Control de Versiones:** Para el control de versiones utilizaré *GitHub*. Se trata de una de las herramientas de control de versiones más utilizadas en el mercado actualmente.
- **Base de Datos**: Será de tipo relacional y utilizaré el motor de bases de datos MySql, ya que es de código abierto y de licencia de uso libre.
- **Back-End**: Se realizará mediante Java, con el *framework* ***Spring boot***, que está diseñado especialmente para facilitar el desarrollo de aplicaciones web y también es de código abierto.
  Además, utilizaré la librería de java ***Thymeleaf***, que implementa un motor de plantillas de XML/XHTML/HTML5 y su utilización con Spring es muy sencilla.
- **Front-End**: Para la capa frontal utilizaré ***Bootstrap 5*** , que es un conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web.

1.4 Planificación del Trabajo

La planificación del proyecto está basada en el calendario establecido por la universidad para la entrega de las diferentes tareas o "PEC's".

Para la creación del plan de proyecto he utilizado el software de licencia gratuita *GranttProject* en su versión 3.2, cuyo objetivo es la administración de proyectos utilizando el diagrama de Grantt.

He visto necesario la creación de "Hitos" previos a la fecha de entrega de cada tarea, con el fin de ir comunicando los avances a mi tutor y contar con margen de tiempo para realizar cualquier tipo de modificación.

PEC 1:

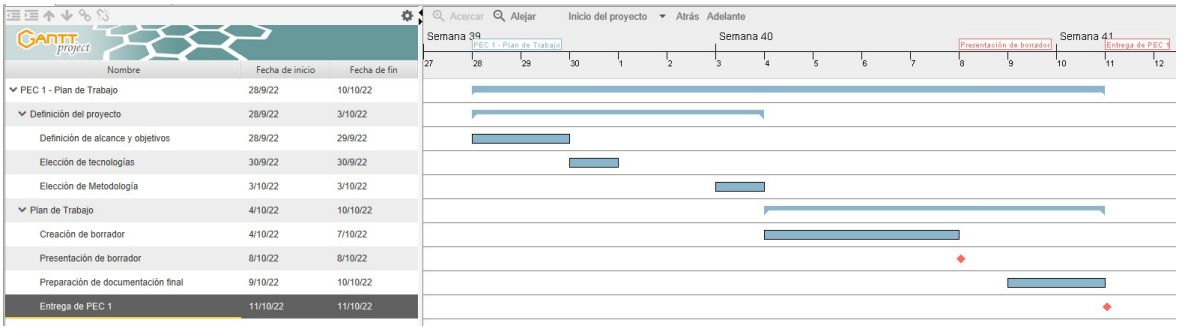


Ilustración 1 Diagrama de Grantt - Planificación PEC1

PEC 2:

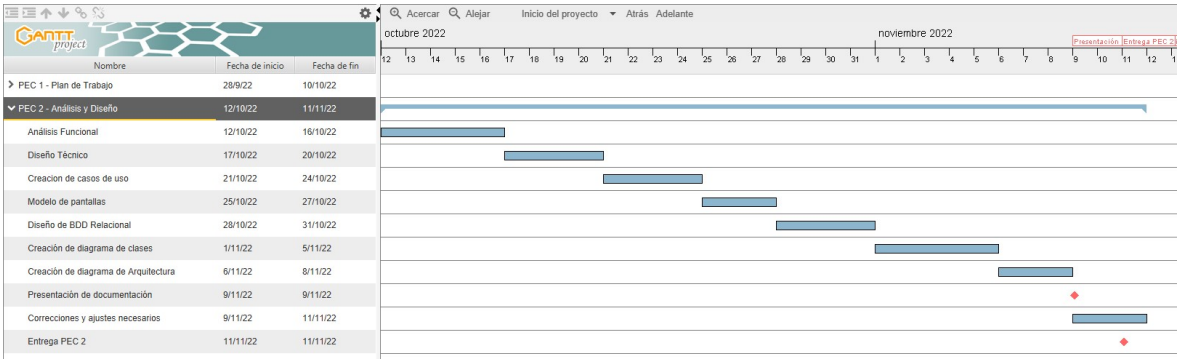


Ilustración 2 Diagrama de Grantt - Planificación PEC2

PEC 3:

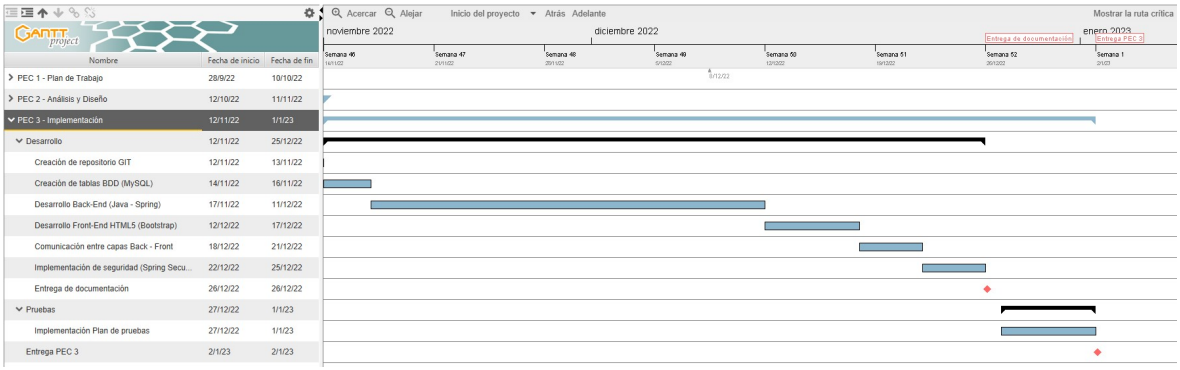


Ilustración 3 Diagrama de Grantt - Planificación PEC3

PEC 4:

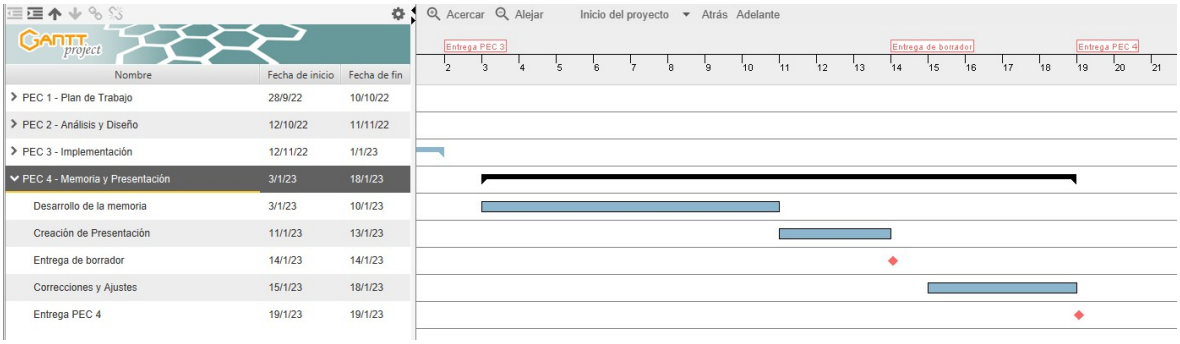


Ilustración 4 Diagrama de Grantt - Planificación PEC4

1.5 Breve sumario de productos obtenidos

Los productos obtenidos durante el desarrollo de el proyecto son los siguientes:

- Aplicación Web:
    - Código fuente (Clases Java y páginas HTML)

    - JAR desplegable
    - Base de datos MySQL
- Documentación:
    - Manual de despliegue local
    - Memoria
    - Presentación virtual

1.6 Breve descripción de los otros capítulos de la memoria

La memoria se divide en 4 capítulos principales que son “Introducción”, en el que definimos las principales motivaciones del proyecto que vamos a realizar, la planificación y el enfoque que le vamos a dar.

El segundo capítulo se trata de el “Análisis Funcional y Diseño Técnico”, en el cuál comentaremos los requerimientos que tendrá el proyecto (tanto funcionales como no funcionales), definimos los casos de uso de la aplicación, realizamos un primer modelado de pantallas que será como queremos que se vea nuestra aplicación. Y por último realizamos el diseño de los diagramas de clase, de la base de datos relacional y comentamos las desviaciones que hemos tenido a lo largo del diseño.

El tercer capítulo está centrado en la implementación del proyecto. Se comenta más a fondo cómo se ha realizado el proyecto, las herramientas que hemos utilizado, la configuración de estas herramientas, la comunicación entre ambas partes, la estructura de nuesro proyecto y por último explicamos cómo funciona cada uno de los elementos.

En el cuarto capitulo, explicamos cómo funciona la aplicación, mostramos todas las páginas que tiene, explicamos su funcionamiento, los permisos que tendrán cada una de ellas, y mostraremos una captura de todas las páginas.

2. Análisis Funcional y Diseño Técnico

Separaremos el análisis en 2 partes, por un lado analizaremos los *Requerimientos funcionales*, que será lo que queremos que haga nuestro portal web, cómo reaccionará ante determinadas acciones de los usuarios, cómo queremos que interactúe con los diferentes sistemas, etc. Y por otro lado los *Requerimientos no funcionales*, que serán los requisitos que no se refieren directamente a las funciones específicas del sistema, sino a las propiedades de este, como su rendimiento, seguridad, disponibilidad, etc.

2.1 Requerimientos funcionales

El portal se dividirá en varios subportales, en los que cada uno debe de poder realizar una función diferente. Por lo que será necesario realizar un análisis funcional de cada uno de estos subportales.

Página Principal (Home):

La primera página de nuestro portal web estará dividida en 2 componentes principales, por un lado tendremos un *menú* con diferentes enlaces y acciones que tiene que mantenerse activo durante toda la navegación del portal para acceder desde cualquier sitio. Y por otro lado tendremos un carrousel de imágenes del centro, con enlaces rápidos a diferente información.

- Menú Principal:
    - Registro de nuevos clientes.
    - Login usuarios registrados.
        - Actividades del día (Sólo usuarios registrados).
        - Datos usuario
    - Información de actividades.
    - Información del centro.
    - Contacto
- Carrousel Dinámico:
    - Noticias del centro e información relevante.

Registro de nuevos clientes:

Enlace a un formulario de registro, en el que se guardarán los datos principales del usuario.

Login usuarios registrados:

Enlace a formulario de ingreso, solamente nombre de usuario y contraseña. Si el usuario ya está logueado permitirá desconectarlo (Logout). Posibilidad de cambiar contraseña en caso de olvido.

El menú para los usuarios registrados sufrirá algunas variaciones, ya que permitirá consultar el calendario con las próximas actividades y permitirá que el usuario se registre a las actividades disponibles. Además, habrá un nuevo botón con la información relativa al usuario.

Calendario de actividades:

Acceso rápido a la agenda de actividades del día en que se realice la consulta. Sólo podrán acceder los usuarios que ya estén registrados, debido a que se tendrá que realizar una consulta a la base de datos con las actividades programadas para la fecha.

Menú datos cliente:

El usuario registrado podrá acceder a sus datos a través de este enlace. Aparecerá información de registro, historial de registro de clases. También podrá modificar datos personales.

Información de actividades:

Enlace a información relativa a las diferentes actividades que realiza el centro. Como fútbol, Natación, Crossfit, BodyPump, Spinning, Sala Fitness, etc.

Dentro de esta página habrá un botón con un enlace a cada una de ellas.

Información del centro:

Toda la información relacionada con el centro de entrenamiento, las salas de las que dispone, el equipamiento, el personal calificado, etc.

Conctacto:

Formulario de contacto con el centro. Se tomarán los datos principales del consultante, además de su correo electrónico para recibir la respuesta del centro.

2.2 Requerimientos no funcionales

Como hemos explicado anteriormente, estos requisitos son los que no describen información a guardar, ni funciones a realizar, sino características de funcionamiento. Dentro de estos tenemos los siguientes:

- El sitio web tiene que ser *responsive*. Por lo tanto, todos los elementos de la web se tienen que reestructurar para optimizar todo el espacio y asegurar una buena experiencia de usuario, un excelente aspecto visual y la funcionalidad. Permitiendo que se altere el diseño y apariencia para adaptarse a diferentes anchos de pantalla, resoluciones, etc.
- El sitio web tiene que ser seguro. Los datos de los usuarios, el acceso, las contraseñas, etc.
- El sitio web debe tener buena disponibilidad. Se estima que al sitio web puedan acceder cientos de usuarios al mismo tiempo, por lo que debería ser capaz de soportar dicha carga.
- Debe ser intuitivo. El usuario tiene que poder cubrir sus necesidades de forma sencilla.

2.3 Casos de Uso

Como hemos visto anteriormente, para nuestro sitio web tendremos 3 roles diferentes. Pero dentro del rol de usuarios tenemos dos actores (Usuarios autenticados y Usuarios no autenticados) con diferentes acciones. Entonces, los actores serán los siguientes:

- Los Administradores, que serán los encargados de gestionar la aplicación, dar de alta a nuevos administradores y monitores, gestionar datos de usuarios (consultar pagos, anular suscripción, etc.).
- Los Monitores, que tendrán acceso a sus horarios, a sus clases asignadas y a los usuarios registrados en sus clases.
- Los Usuarios autenticados, que podrán registrarse a clases, consultar y modificar sus datos, consultar calendario de actividades.
- Los Usuarios no autenticados, podrán acceder al registro de usuarios, a la información del centro, información de actividades, contacto.

Los casos de uso correspondientes a cada uno se describen a continuación:

2.3.1 Casos de uso Todos los actores:

CU_001	Consultar Información de Actividades
Actor Principal	Todos los actores
Objetivos	Obtener información de las actividades que realiza el centro
Precondiciones	Ningúna
Postcondiciones	Se vizualiza la información correspondiente
Escenario Principal	1) Se accede al menú principal 2) Se selecciona el desplegable "Actividades" y se accede a la actividad que se quiere consultar 3) Se carga la vista de la actividad seleccionada
Escenario Alternativo	-

Tabla 1 CU_001 - Consultar Información de Actividades

CU_002	Consultar Información Centro
Actor Principal	Todos los actores
Objetivos	Obtener información del centro
Precondiciones	Ningúna
Postcondiciones	Se vizualiza la información correspondiente
Escenario Principal	1) Se accede al menú principal 2) Se selecciona el desplegable "Nuestro Centro" 3) Se carga la vista de información del centro con los enlaces a las diferentes secciones (Salas, Equipamiento, Personal, etc.)
Escenario Alternativo	-

Tabla 2 CU_002 - Consultar Información Centro

CU_003	Contacto
Actor Principal	Todos los actores
Objetivos	Enviar consulta/sugerencia al centro
Precondiciones	Ningúna
Postcondiciones	Se envía el mensaje al centro
Escenario Principal	1) Se accede al menú principal 2) Se selecciona el botón "Contacto" 3) Se completan los datos solicitados por el formulario de contacto 4) Se presiona el botón Enviar.
Escenario Alternativo	3a) Los datos ingresados contienen errores (Caracteres inválido, etc.). Se le informa al usuario que debe corregirlos.

Tabla 3 CU_003 - Contacto

2.3.2 Casos de uso Usuarios No Autenticado:

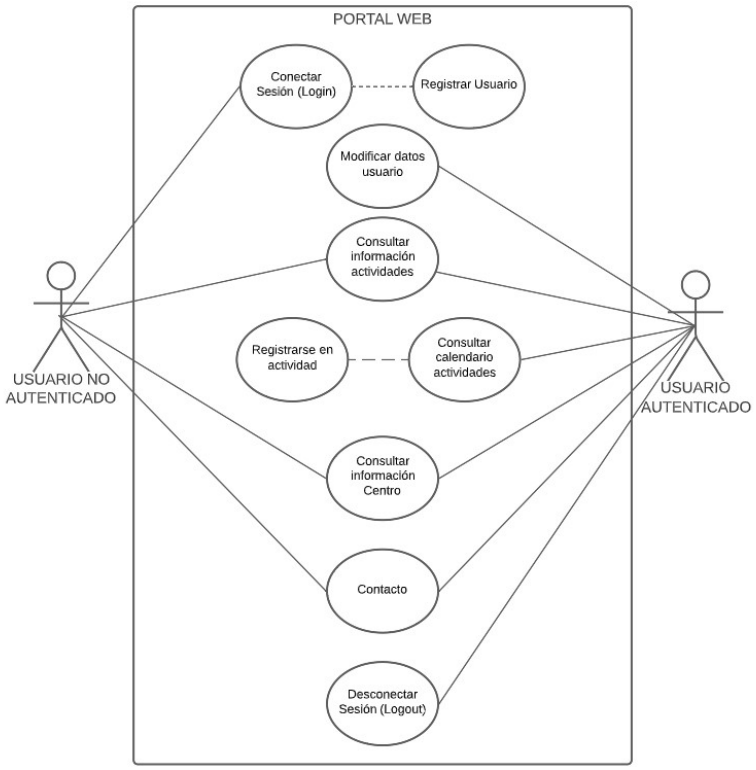


Ilustración 5 Diagrama de Casos de Uso - Usuarios Autenicados/Usuarios no Autenticados

CU_004	Conectar Sesión (Login)
Actor Principal	Usuario no Autenticado
Objetivos	Dar acceso al usuario al sitio web
Precondiciones	No estar autenticado
Postcondiciones	Usuario logueado
Escenario Principal	1) Se accede al botón de "Iniciar Sesión" ubicado en el menú principal 2) Se completa el formulario de inicio de sesión con los datos de usuario y contraseña y se presiona el botón de "Acceder" 3) Se valida el usuario y se actualiza el menú con los accesos correspondientes al rol de usuario autenticado.
Escenario Alternativo	3a) El usuario no existe o está mal escrito - Se informa del error al usuario

Tabla 4 CU_004 - Conectar Sesión (Login)

CU_005	Registrar Usuario
Actor Principal	Usuario no Autenticado
Objetivos	Dar de alta un nuevo usuario en la base de datos
Precondiciones	El usuario por registrar no existe
Postcondiciones	Se crea un nuevo usuario
Escenario Principal	1) Se hace clic sobre el botón "Regístrate" en el menú principal 2) Se completa el formulario de registro 3) Se validan los datos ingresados 4) Se presiona el botón de "Registro" y se guarda el usuario en la BBDD. 5) El usuario queda logueado y se habilita el acceso a todos los enlaces del rol "Usuario"
Escenario Alternativo	3a) Los datos ingresados contienen errores (Caracteres inválidos, contraseña no cumple requisitos mínimos, etc.) 4a) El nombre de usuario ya existe. Se informa de error y se solicita su modificación

Tabla 5 CU_005 - Registrar Usuario

2.3.3 Casos de uso Usuarios Autenticado:

CU_006	Modificar datos de usuario
Actor Principal	Usuarios Autenticados
Objetivos	Acceder a los datos del usuario y poder modificar información
Precondiciones	El usuario existe en la BBDD
Postcondiciones	El usuario se modifica
Escenario Principal	1) Se accede al icono correspondiente al usuario logueado. 2) Se hace clic sobre el botón de "Mis datos". 3) Se cambian los datos permitidos por el sitio y se da clic en Guardar.
Escenario Alternativo	3a) Los nuevos datos ingresados no cumplen con las validaciones (Caracteres inválidos, contraseñas que no cumplen los requisitos mínimos, etc.)

Tabla 6 CU_006 - Modificar datos de usuario

CU_007	Consultar Calendario de actividades
Actor Principal	Usuarios Autenticados
Objetivos	Visualizar el calendario de próximas actividades (Mes completo)
Precondiciones	El usuario está registrado
Postcondiciones	Se vizualiza el calendario de actividades
Escenario Principal	1) Se accede al botón "Calendario de Actividades" 2) Se selecciona el día del mes en el calendario 3) Se carga una nueva vista con las actividades del día.
Escenario Alternativo	3a) No existen actividades para el día seleccionado

Tabla 7 CU_007 - Consultar Calendario de actividades

CU_008	Registrarse en Actividad
Actor Principal	Usuarios Autenticado
Objetivos	Registrarse en actividad deportiva
Precondiciones	El usuario no está registrado previamente. El usuario no está suspendido por el administrador Hay cupos disponibles en la actividad
Postcondiciones	Se da de alta al usuario en la actividad
Escenario Principal	1) Se accede al botón "Próximas Actividades" 2) Se selecciona la actividad a la que se quiere registrar 3) Se da clic en "Registrarme!" 4) El usuario queda registrado en la actividad; el cupo de la actividad se decrementa en 1.
Escenario Alternativo	3a) La actividad está completa, no se permite el registro.

Tabla 8 CU_008 - Registrarse en Actividad

CU_009	Desconectar Sesión
Actor Principal	Usuarios Autenticado
Objetivos	Cerrar sesión de usuario
Precondiciones	El usuario está logueado
Postcondiciones	Se cierra la sesión
Escenario Principal	1) Se da clic sobre el botón "Desconectar"
Escenario Alternativo	-

Tabla 9 CU_009 - Desconectar Sesión

2.3.4 Casos de uso Monitores:

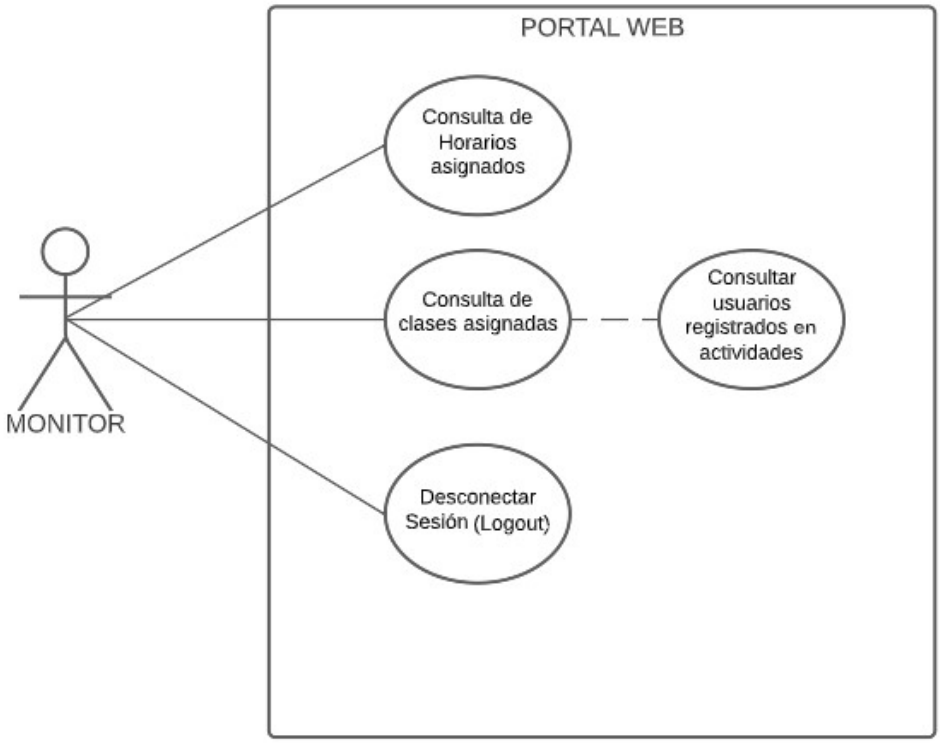


Ilustración 6 Diagrama de Casos de Uso – Monitor

CU_010	Consultar horarios asignados
Actor Principal	Monitor
Objetivos	Acceder a los horarios que tiene asignado el monitor
Precondiciones	El usuario logueado tiene el rol de monitor
Postcondiciones	Se consulta el horario
Escenario Principal	1) El monitor accede al ícono de su usuario 2) Presiona el botón "Mis horarios" 3) Se visualiza el calendario con los días de trabajo asignados y los horarios de actividades
Escenario Alternativo	2a) No tiene actividades asignadas

Tabla 10 CU_010 - Consultar horarios asignados

CU_011	Consulta de actividades asignadas
Actor Principal	Monitor
Objetivos	Acceder a las actividades que tiene asociadas el monitor
Precondiciones	El usuario logueado tiene el rol de monitor
Postcondiciones	Se consulta la actividad
Escenario Principal	1) El monitor accede al ícono de su usuario 2) Presiona el botón "Mis próximas actividades" 3) Se visualiza un listado de las actividades que tiene asignadas el usuario
Escenario Alternativo	2a) No tiene actividades asignadas

Tabla 11 CU_011 - Consulta de actividades asignadas

CU_012	Consultar usuarios registrados en actividades
Actor Principal	Monitor
Objetivos	Acceder al listado de usuarios registrados en la actividad del monitor
Precondiciones	El usuario logueado tiene el rol de monitor
Postcondiciones	Se obtiene listado de usuario registrados
Escenario Principal	1) El monitor accede al ícono de su usuario 2) Presiona el botón "Mis próximas actividades" 3) Se visualiza un listado de las actividades que tiene asignadas el usuario. 4) Selecciona una actividad en particular 5) Presionar el botón "Ver los usuarios registrados" 6) Se visualiza un listado de usuarios registrados
Escenario Alternativo	3a) No tiene actividades asignadas 6a) No existen usuarios registrados

Tabla 12 CU_012 - Consultar usuarios registrados en actividades

2.3.5 Casos de uso Administradores:

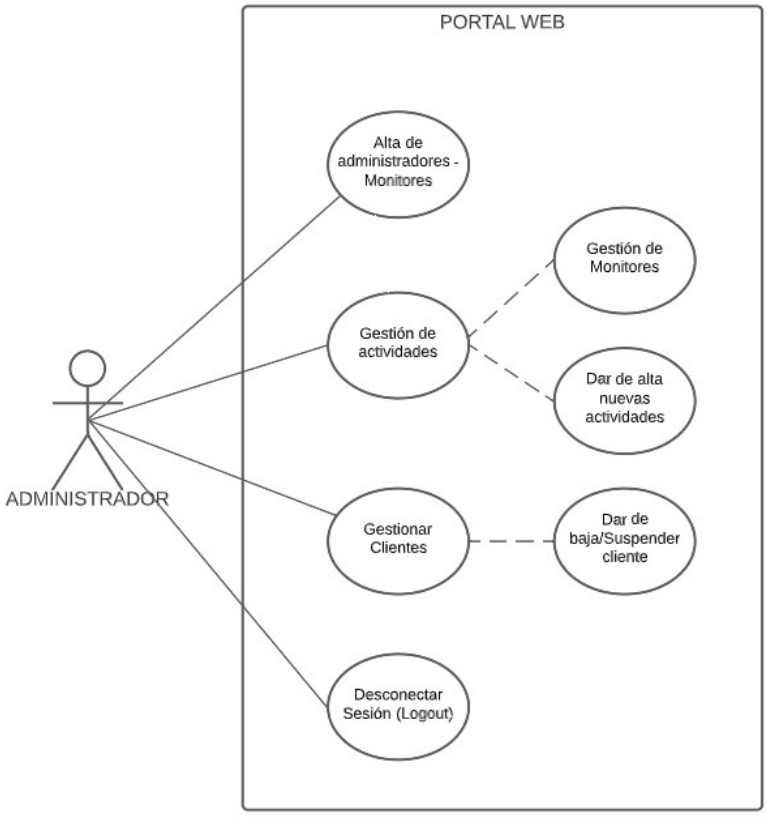


Ilustración 7 Diagrama de Casos de Uso - Administrador

CU_013	Alta de Administradores/Monitores
Actor Principal	Administradores
Objetivos	Creación de nuevos usuarios con roles de "Administrador" o "Monitor"
Precondiciones	El usuario logueado tiene el rol de Administrador
Postcondiciones	Se crea nuevo usuario
Escenario Principal	1) El Administrador accede al icono de "Gestión" (Solo disponible para los administradores. 2) Accede a la opción de "Alta de Empleados" 3) Se completa el formulario de creación de empleado. Se marca el rol correspondiente 4) Se guarda empleado en base de datos
Escenario Alternativo	3a) Los nuevos datos ingresados no cumplen con las validaciones (Caracteres inválidos, contraseñas que no cumplen los requisitos mínimos, etc.)

Tabla 13 CU_013 -Alta de Administradores/Monitores

CU_014	Gestión de Actividades
Actor Principal	Administradores
Objetivos	Gestionar las actividades del centro (Alta, Baja, Modificación, Asignación de monitores)
Precondiciones	El usuario logueado tiene el rol de Administrador
Postcondiciones	Se Crea/Modifica la actividad
Escenario Principal	1) El Administrador accede al icono de "Gestión" (Solo disponible para los administradores. 2) Accede a la opción de "Gestión de Actividades" 3) Aparecerá un listado de opciones a realizar (Nueva actividad, Borrado de actividad, Modificación de Actividad, Asignación de monitores)
Escenario Alternativo	-

Tabla 14 CU_014 - Gestión de Actividades

CU_015	Gestión de monitores
Actor Principal	Administradores
Objetivos	Se modifica su horario, sus especialidades, etc.
Precondiciones	El usuario logueado tiene el rol de Administrador
Postcondiciones	Se modifica el monitor
Escenario Principal	1) El Administrador accede al icono de "Gestión" (Solo disponible para los administradores. 2) Accede a la opción de "Gestión de Monitores" 3) Aparecerá un buscador con filtros para encontrar al monitor que queremos modificar. 4) Se selecciona el monitor y se selecciona la opción correspondiente (Modificar horario, Modificar especialidad, etc.)
Escenario Alternativo	-

Tabla 15 CU_015 - Gestión de monitores

CU_016	Alta de nuevas actividades
Actor Principal	Administradores
Objetivos	Crear una nueva actividad
Precondiciones	El usuario logueado tiene el rol de Administrador El usuario está situado en "Gestión de Actividades"
Postcondiciones	Se da de alta una actividad
Escenario Principal	1) Dentro de las opciones sobre las actividades se selecciona "Nueva Actividad" 2) Aparecerá un listado con el tipo de actividad que queremos crear 3) Seleccionamos la actividad y completamos los datos de la nueva actividad
Escenario Alternativo	-

Tabla 16 CU_016 - Alta de nuevas actividades

CU_017	Gestionar clientes
Actor Principal	Administradores
Objetivos	Consultar datos de clientes
Precondiciones	El usuario logueado tiene el rol de Administrador
Postcondiciones	Se obtienen los datos buscados
Escenario Principal	1) El Administrador accede al icono de "Gestión" (Solo disponible para los administradores. 2) Accede a la opción de "Gestión de Clientes" 3) Aparecerá un buscador con filtros para encontrar al cliente que queremos consultar. 4) Se selecciona el cliente y se obtienen los datos del mismo.
Escenario Alternativo	3a) El cliente no se encuentra en la base de datos

Tabla 17 CU_017 - Gestionar clientes

CU_018	Actualizar situación de clientes
Actor Principal	Administradores
Objetivos	Dar de baja a un cliente
Precondiciones	El usuario logueado tiene el rol de Administrador El cliente tiene que estar activo en la BDD El administrador se encuentra en la gestión de cliente
Postcondiciones	El usuario dado de baja no puede acceder a la web
Escenario Principal	1) Se selecciona la opción "Dar de baja al cliente", "Suspender", "Activar". 2) Se selecciona el motivo por el que se toma la decisión 3) Se actualiza la base de datos y se modifica el usuario. Si es baja se borrará el cliente y sus actividades relacionadas. Si la opción es "Suspender" se actualiza la base de datos y no les permite registrarse a nuevas actividades. Si ya estaba suspendido se permite "Activar" para que el usuario pueda volver a registrarse a las actividades.
Escenario Alternativo	

Tabla 18 CU_018 - Actualizar situación de clientes

1.4 Modelado de pantalla

Se adjunta el modelo de pantallas seleccionado para nuestro sitio web. Se trata de un esquema provisorio para utilizar como referencia, sufrirá variaciones durante la implementación de la capa frontal.

Home:

La pantalla principal tendrá una barra superior con navegación entre las diferentes páginas. La misma contendrá la información mencionada anteriormente y dependerá del usuario con el que se acceda. Si aún no hay usuario conectado se mostrará las opciones de Registro/Login. De lo contrario se mostrarán las opciones correspondientes según su rol.

Ilustración 8 Modelo de pantalla - Home (Versión Desktop)

Como hemos mencionado anteriormente, el sitio web mantendrá un diseño *responsive*, por lo que la misma página vista a través de un dispositivo móvil se vería de la siguiente manera:

Ilustración 9 Modelo de pantalla - Home (Versión Móvil)

Formulario de Registro:

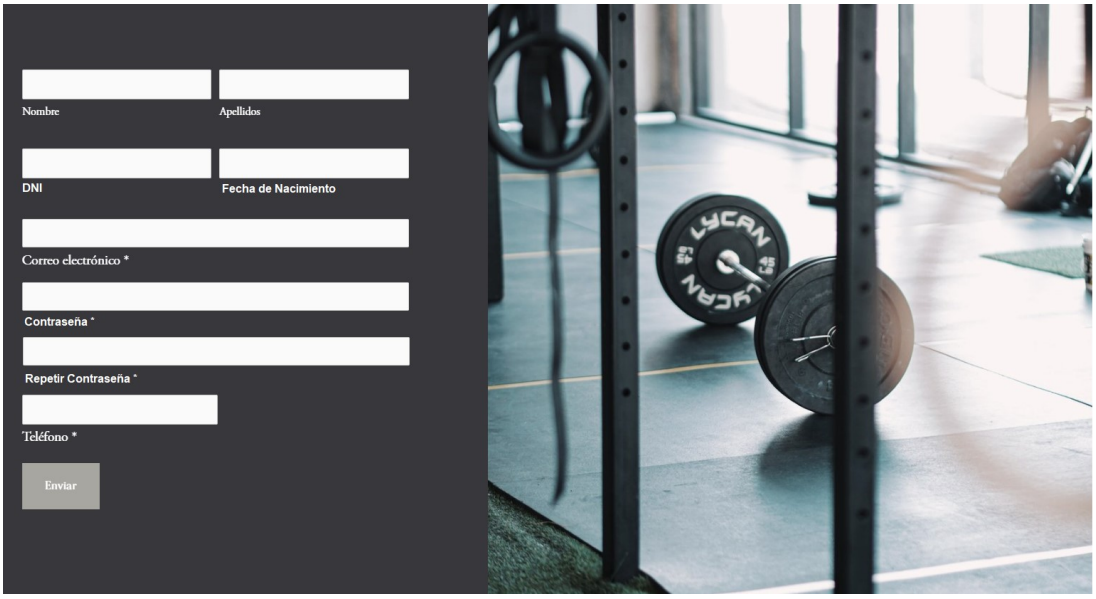


Ilustración 10 Modelo de pantalla - Registro

Formulario de Login:

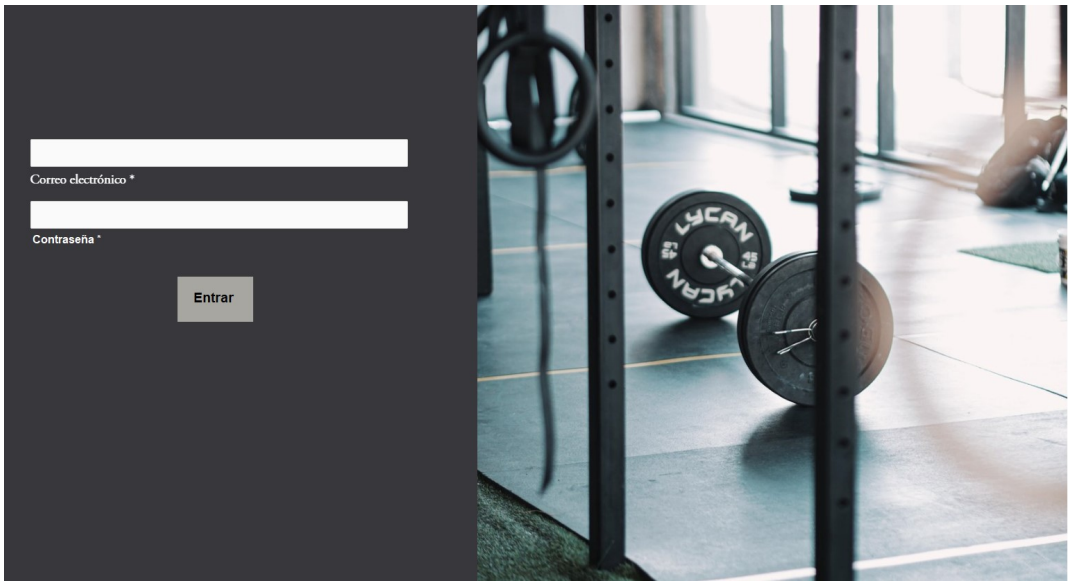


Ilustración 11 Modelo de pantalla - Login

Formulario de contacto:

Se completarán los datos necesarios para establecer el contacto con el centro.

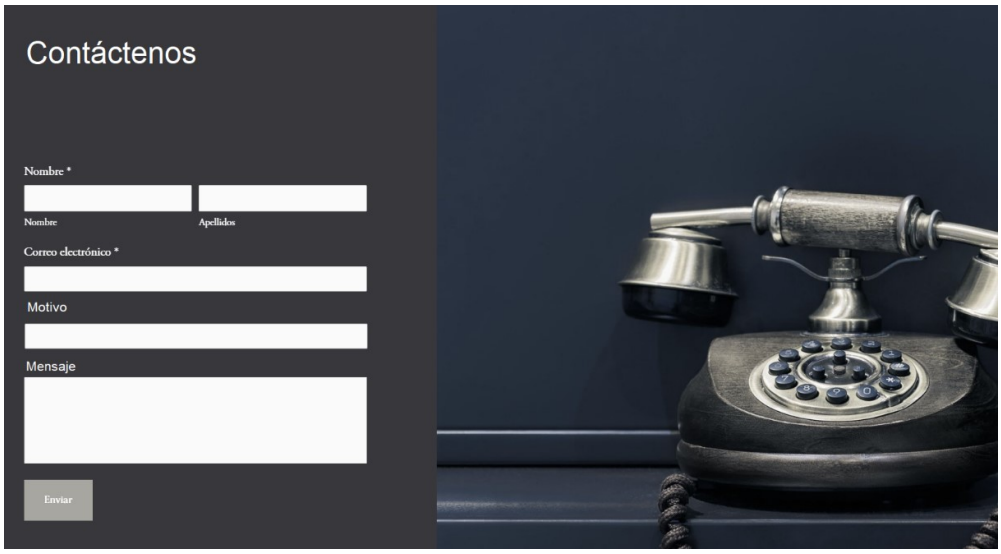


Ilustración 12 Formulario de contacto

Calendario de actividades:

Ilustración 13 Agenda de Actividades

Ilustración 14 Selección de Actividad

1.5 Diagrama de Clases UML

Se han detectado de momento 7 clases. La clase principal “User”, de la que extienden 3 clases diferentes:

- Admin: Corresponde al rol de Administrador “ADMIN”
- Monitor: Correspondiente al rol de Monitor “MONITOR”
- Client: Correspondiente al rol de Client “CLIENT”.

Una clase “Activity” que guardará la información correspondiente a las actividades, su ID, el tipo de actividad, el monitor que tendrá asignado, la sala donde se realizará, la capacidad de la actividad, y el listado de clientes apuntados.

La clase “Contact” correspondiente a los mensajes que se enviarán al centro. Se relaciona con los administradores ya que son ellos quienes podrán consultar estos mensajes.

La clase “Rooms” que se corresponde a las salas que tendrá disponible el centro deportivo.

El diagrama de clases asociado al sitio web es el siguiente

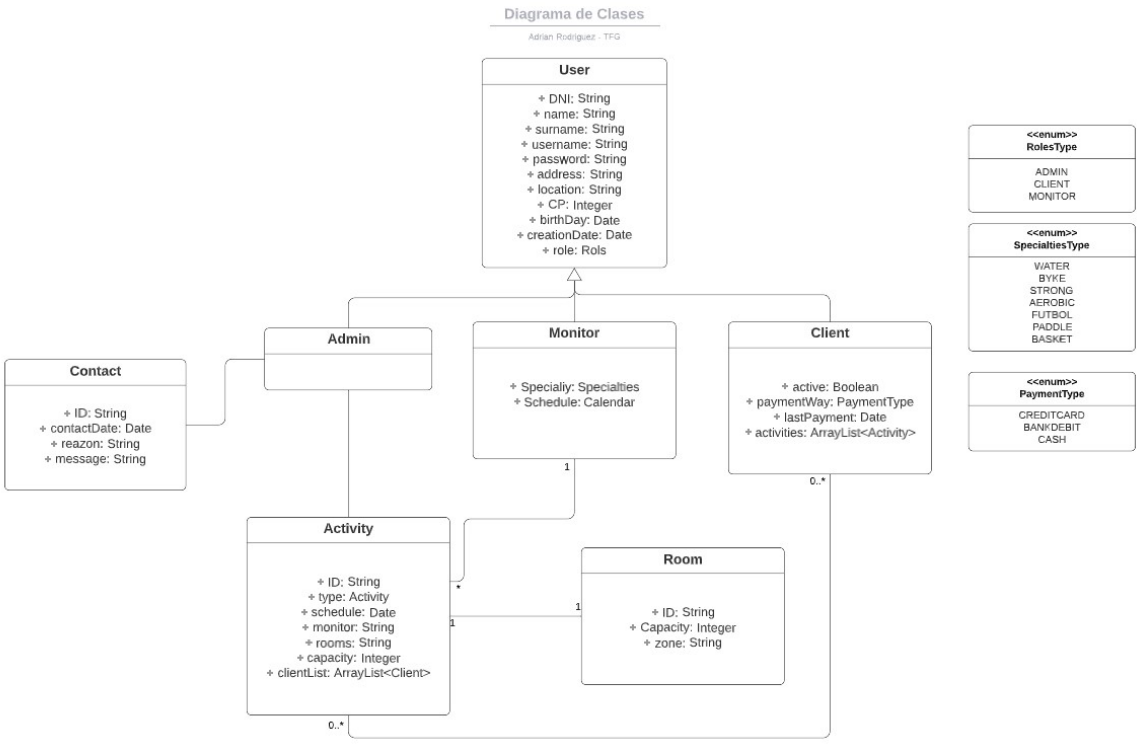


Ilustración 15 Diagrama de Clases

1.6 Diseño de base de datos relacional

A continuación se muestra el modelo Entidad-Relación identificado para nuestro sitio web. En el se destacan las siguiente entidades:

- User: Correspondiente a todos los tipos de usuario registrados.
- Role: Corresponde al rol de usuario, que pueden ser (Administrador, Monitor, Cliente).
- Monitor: Este tipo de usuario tiene una clase especial para definir su especialidad.
- Especiality: Es el tipo de especialidad, aplica tanpo para los monitores, como para las actividades y las salas (Ej: Fútbol, Bicicleta, Fuerza, Agua, etc.).
- Actividad: Es una tabla especifica para las diferentes actividades.
- Calendar_Activity: Es una tabla de relación entre los clientes, los monitores, las salas donde se realizan y las actividades. Serían las actividades que se van a realizar en determinada fecha, en determinada sala, con determinada capacidad, monitor y diferentes usuarios.
- User_Activity: Tabla para guardar las actividades a las que se ha registrado un cliente.
- Room: Corresponde a las salas del centro.
- Contact: En esta tabla se guardan los contactos realizados en la página. Están relacionadas únicamente con el administrador que trate el caso.

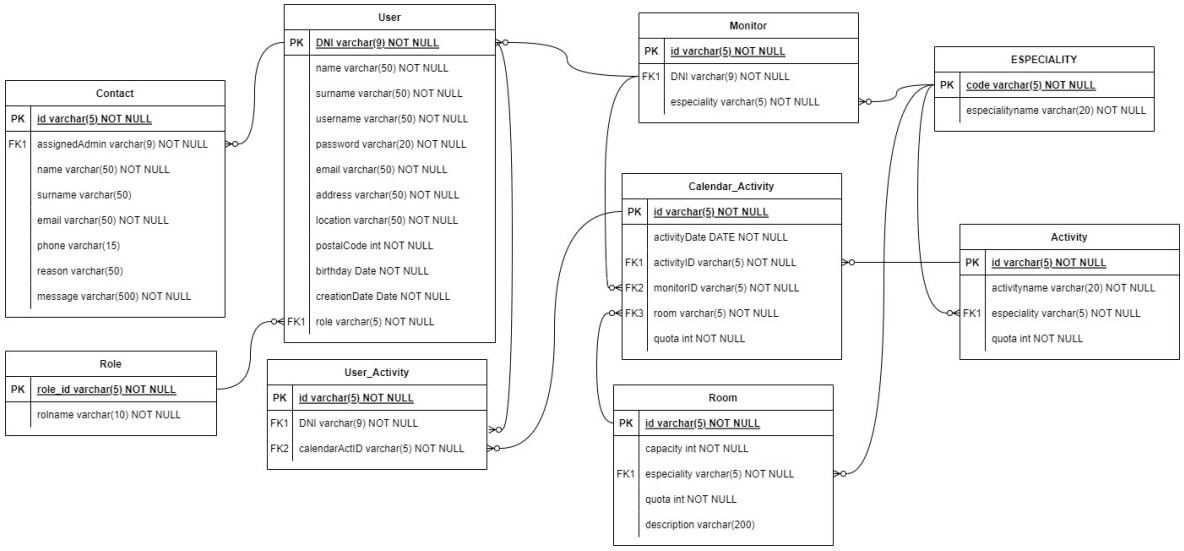


Ilustración 16 Diagrama ERD

1.7 Estado del proyecto - Desviaciones

Al finalizar la segunda entrega, hemos logrado cumplir con los hitos establecidos en el plan de trabajo. Sin embargo, se han tenido que realizar algunas correcciones tras la primera entrega de “Presentación de Documentación” y el tiempo indicado para realizar las correcciones ha sido demasiado ajustado.

Este problema no surgirá durante las dos siguientes entregas debido a que el tiempo establecido entre el hito de entrega de documentación y la entrega final es de 5 días para ambos casos.

Por otro lado, durante la fase de análisis, al realizar los diagramas de casos de uso, se ha detectado la necesidad de incluir un actor más (Monitor), por lo que he necesitado realizar algunas correcciones sobre la primera entrega. Lo que ha retrasado un día la realización del análisis funcional.

De todas maneras, se ha podido ajustar el tiempo de realización del resto de items para no sufrir retrasos sobre la fecha de entrega indicada.

3. Implementación

3.1 Creación de repositorio GitHub

Comenzaremos creando un repositorio en GitHub para realizar un control de versiones de nuestro proyecto.

Para ello simplemente accedemos a https://github.com/. Para este proyecto he creado una nueva cuenta asociada al usuario de la universidad "arodriguezdiaz1@uoc.edu". Una vez creada la cuenta procedo a crear un nuevo repositorio "TFG_ADRO":

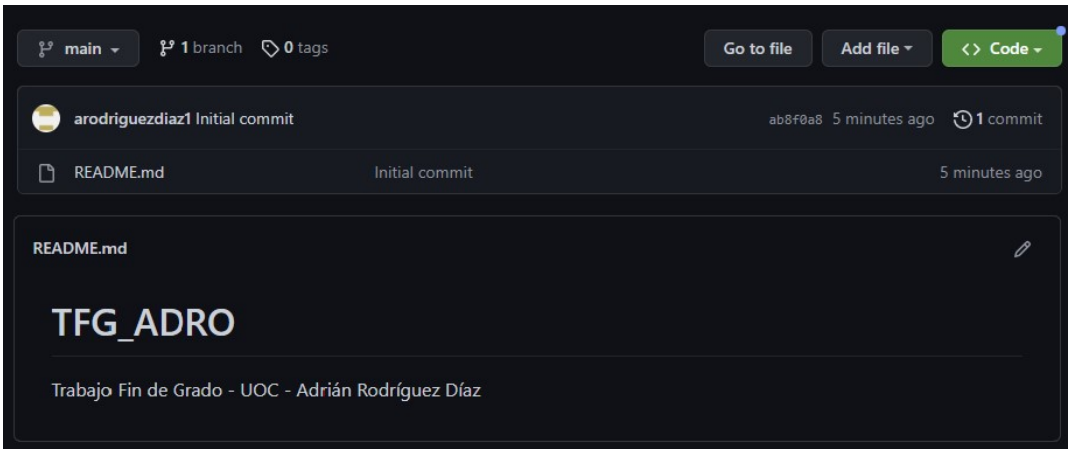


Ilustración 17 Repositorio GIT

3.2 Configuración base de datos MySQL

El siguiente paso será configurar nuestra base de datos. Para ello, como hemos mencionado anteriormente trabajaremos sobre una base de datos MySQL, para poder trabajar sobre ella sobre el sistema operativo Windows, utilizaremos la herramienta XAMPP, que es un paquete de software libre que contiene un servidor web Apache con el que podremos gestionar nuestra base de datos.

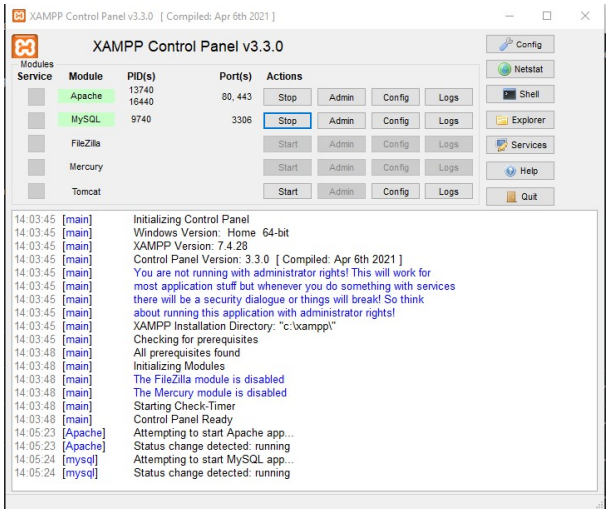


Ilustración 18 XAMPP Configuracion

Una vez que hemos iniciado los servicios de “Apache” para el servidor y el motor de base de datos “MySQL”, iniciamos al administrador de la herramienta, para ello vamos a nuestro *localhost* y accedemos a *phpMyAdmin:*

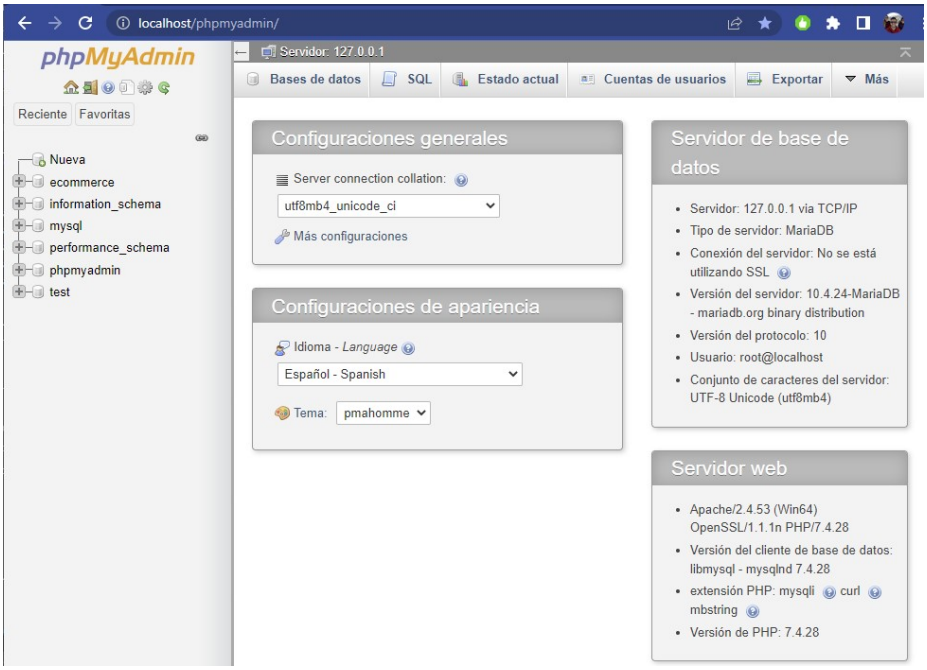


Ilustración 19 PhpMyAdmin

Procederemos a crear nuestra base de datos:

La llamaremos “tfg_adro” y utilizamos la codificación general en UTF8.

Ilustración 20 Base de datos – tfg_adro

Una vez que tenemos nuestra base de datos creada, damos de alta un nuevo usuario, que será el que tenga los roles necesarios para actualizar la información de la base de datos.

Usuario: tfguser

Pass: Uoc@1234

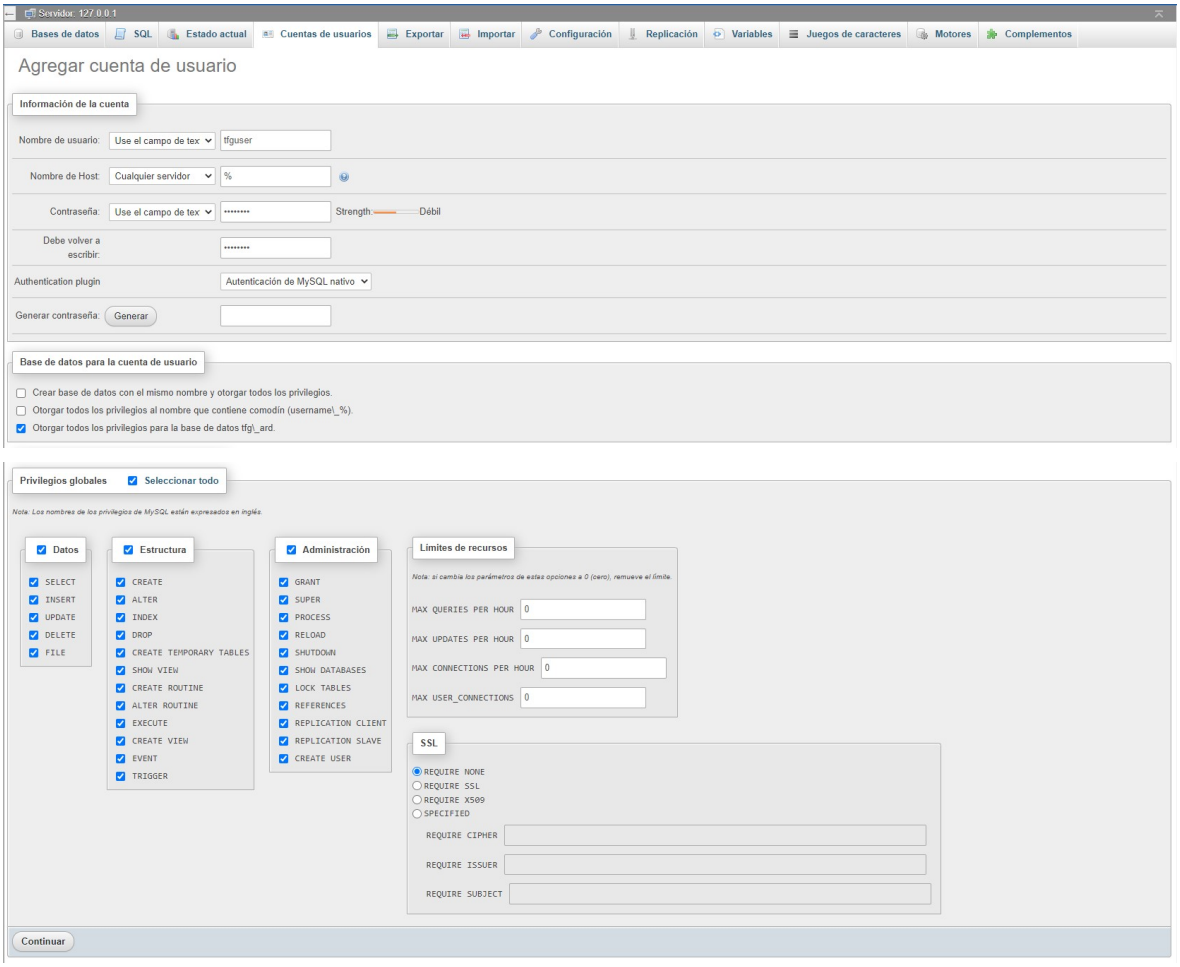


Ilustración 21 Base de datos – creación de usuario

Ya tenemos creada nuestra base de datos y el respositorio sobre el que vamos a ir subiendo los cambios a medida que avanzamos en el desarrollo de la aplicación, procederemos ahora a crear nuestro proyecto Java

3.3 Creación proyecto Spring – Java

Como hemos mencionado, el IDE con el que vamos a trabajar sobre Java será el Spring Tool Suite 4 (la última versión al momento "4.17.0"), que está basado en Eclipse, y viene con todas las herramientas necesarias para trabajar con módulos web mediante Spring.

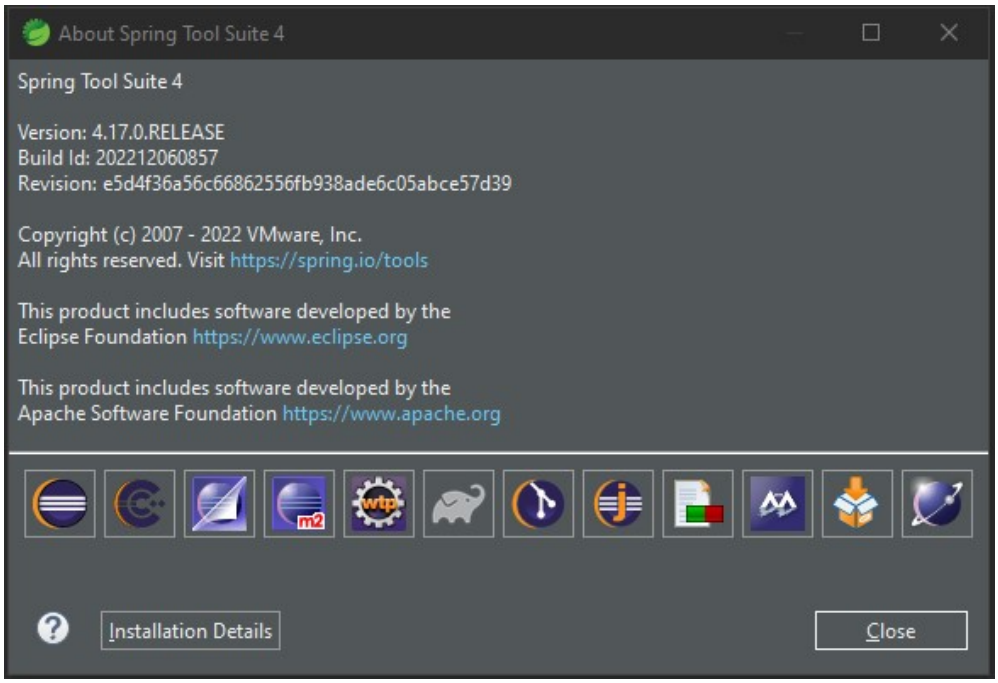


Ilustración 22 Spring Tool Suit 4

Para crear nuestro proyecto, iremos a "*File/new/Spring Starter Project".*

* Una alternativa diferente si estuvieramos trabajando con otro IDE sería crear nuestro proyecto a través de https://start.spring.io/, pero para nuestro proyecto aprovecharemos las facilidades que nos brinda el IDE elegido.

Seleccionamos aquí la versión de Java con la que trabajaremos (17), indicamos nombre y grupo del proyecto, indicamos además que vamos a trabajar con Maven para descargar nuestras librerías necesarias y el tipo de *Packaging*, que en nuestro caso va a ser un fichero Jar.

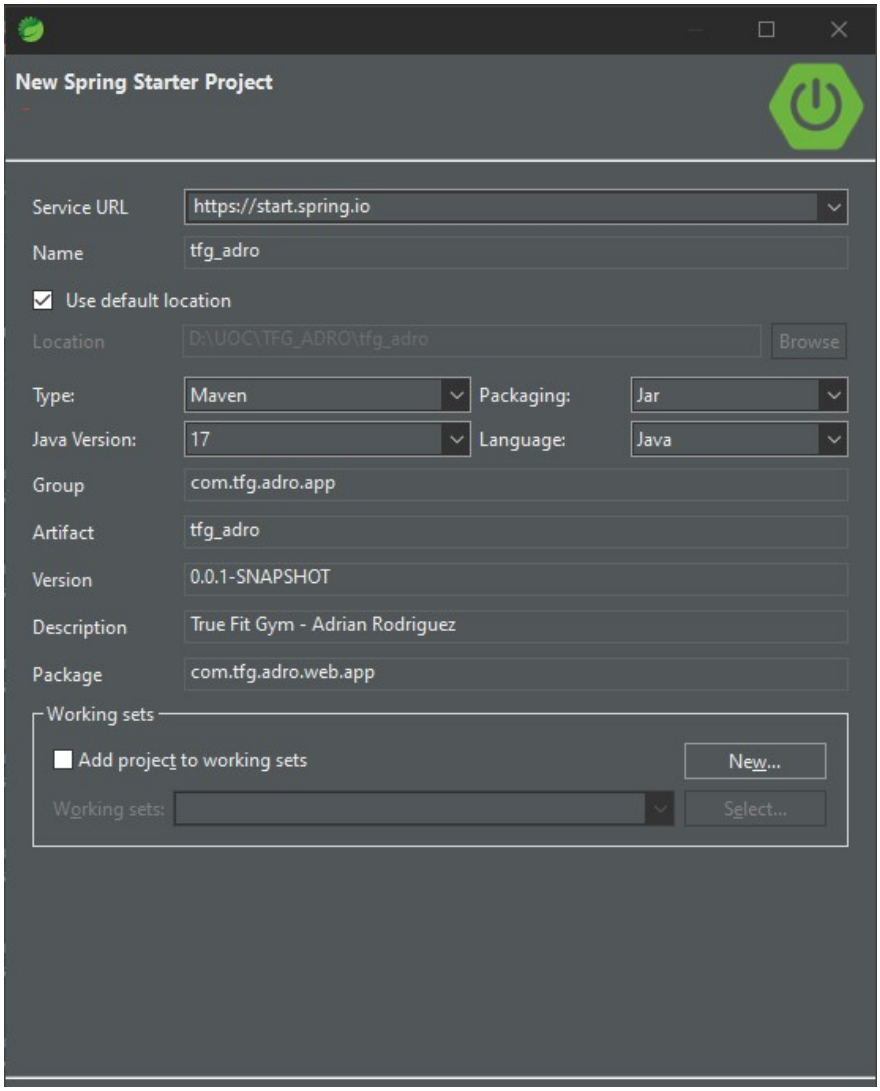


Ilustración 23 Creación del proyecto

En el siguiente paso seleccionamos la versión de Spring Boot que vamos a utilizar (3.0.1) que es la última versión estable; y añadimos aquí todas las dependencias que vamos a necesitar.

En nuestro caso:

- *Spring Boot DevTools*: Para que se reinicie la aplicación de forma automática cada vez que realizamos un cambio sobre nuestro código
- *Validation*: Servirá para facilitarnos la eliminación de código verboso y *boilerplate* de nuestras aplicaciones y nos simplificará su construcción.
- *Spring Data JPA*: Para nuestra persistencia en la base de datos
- *MySQL Driver*: Servirá para conectar nuestro projecto con la base de datos
- *Spring Security:* Será el framework de apoyo para nuestra configuración de seguridad de Spring.

- *Thymeleaf*: Como mencionamos anteriormente, esta librería será la que nos ayude con la comunicación entre back y front; será nuestro motor de plantillas.

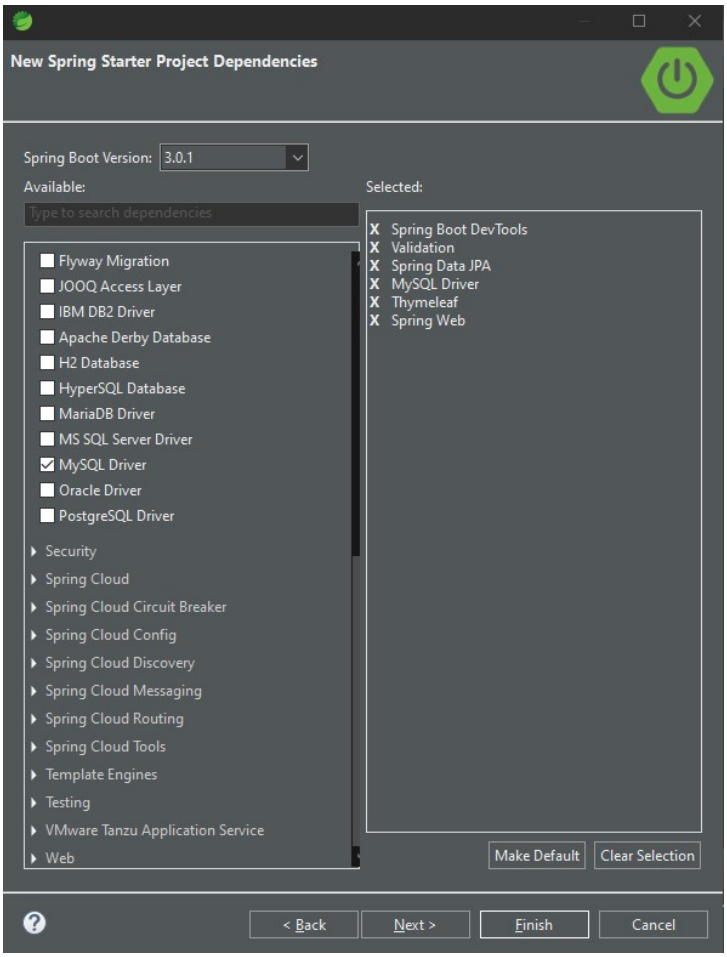


Ilustración 24 Selección de dependencias

Si fuese necesario añadir alguna nueva dependencia se puede incluír en el fichero pom.xml de nuestro proyecto en cuanto queramos.

Una vez finalizada la construcción y descargadas todas las dependencias, la estructura de nuestro proyecto es la siguiente:

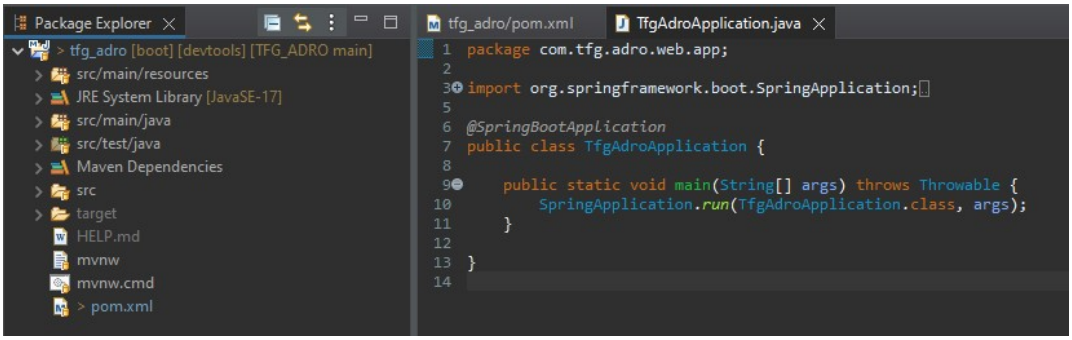


Ilustración 25 Estructura de proyecto Java

Cremos un controlador básico para nuestra página inicial y nuestro Index.html, para comprobar que el servidor se inicia correctamente, y que Spring está correctamente configurado:

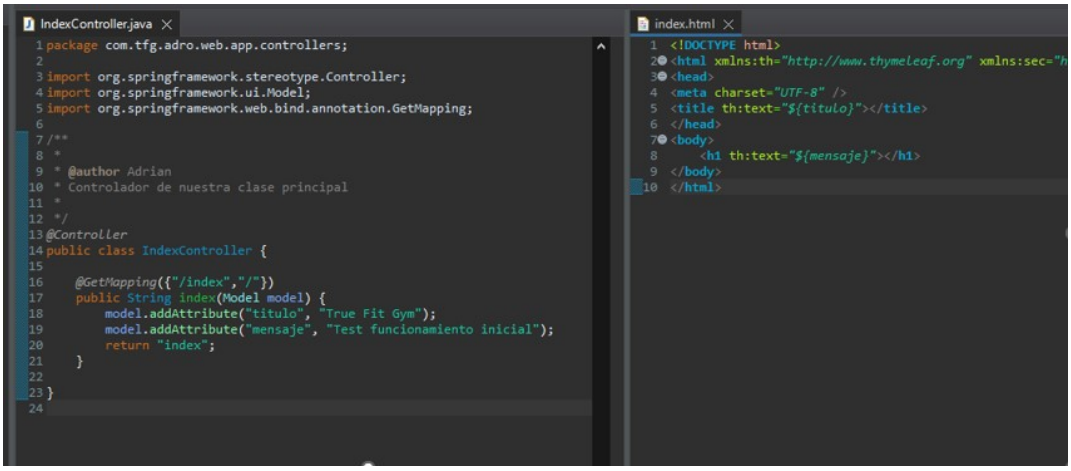


Ilustración 26 Configuración de Index y Controller - Prueba inicial

Iniciamos el servidor y comprobamos que al acceder a nuestro index local vemos el mensaje de test:

Ilustración 27 Test funcionamiento inicial

El siguiente paso será configurar nuestra base de datos creada anteriormente para que haya comunicación entra ambas partes.

Para ello, abrimos el fichero “application.properties” y configuramos las propiedades de nuestra base de datos.

Necesitamos indicar el *datasource*, que será la url donde se encuentra alojada nuestra base de datos, el puerto, su nombre, y definimos los parámetros de seguridad y de ubicación:

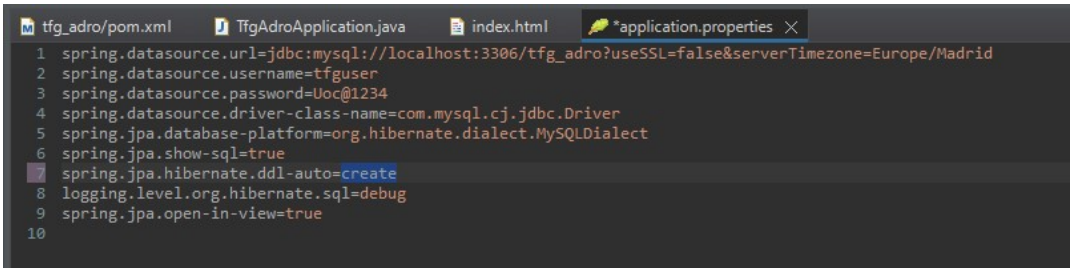


Ilustración 28 Configuración BBDD – IDE

Entre los parámetros a destacar están los de *username* y *password*, que son los que definimos anteriormente para nuestra BBDD, el *driver* que utilizaremos para conectarnos, el "dialecto" de MySQL que utilizaremos.

Además, indicamos la traza *"spring.jpa.hibernate.ddl-auto=create"*, para decir que queremos que cree nuestras tablas a partir de las entidades que crearemos más adelante. Y por otro lado indicamos el *logging* en modo *debug*, para poder ver en consola los movimientos que se realizan sobre la base de datos.

Una vez que tenemos el proyecto funcionando, procedemos a sincronizarlo con el repositorio Git configurado inicialmente. Para ello, realizamos un primer merge con la estructura básica de nuestro proyecto y lo subimos al repositorio una vez configurada la URL y las credenciales.

3.4 Sincronización IDE – GitHub

Procederemos ahora a sincronizar nuestro proyecto con el repositorio Git configurado inicialmente. Para ello, realizamos un primer merge con la estructura básica de nuestro proyecto y lo subimos al repositorio una vez configurada la URL y las credenciales:

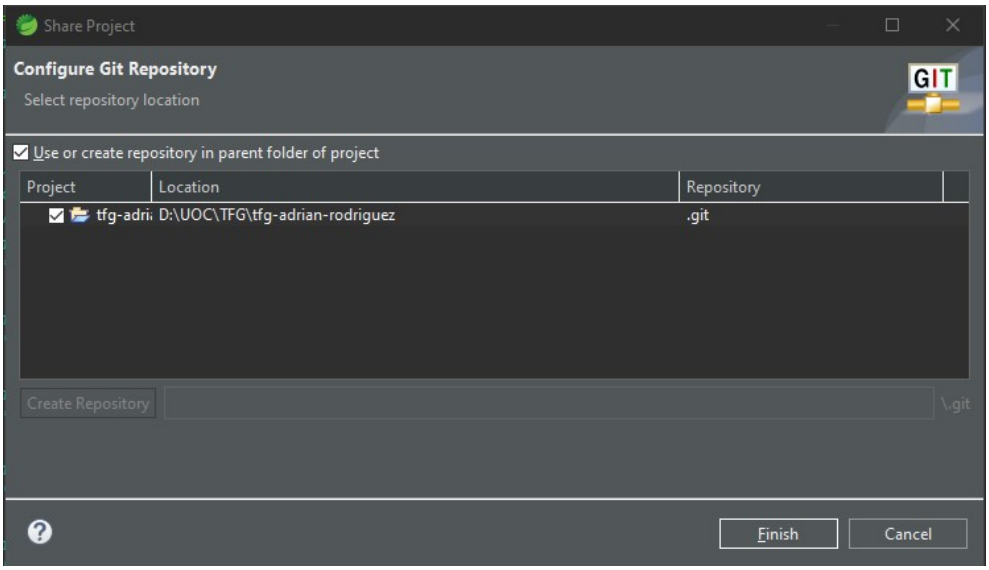


Ilustración 29 Configuración GIT

Copiamos la URI que obtenemos desde GIT, indicamos las credenciales y avanzamos

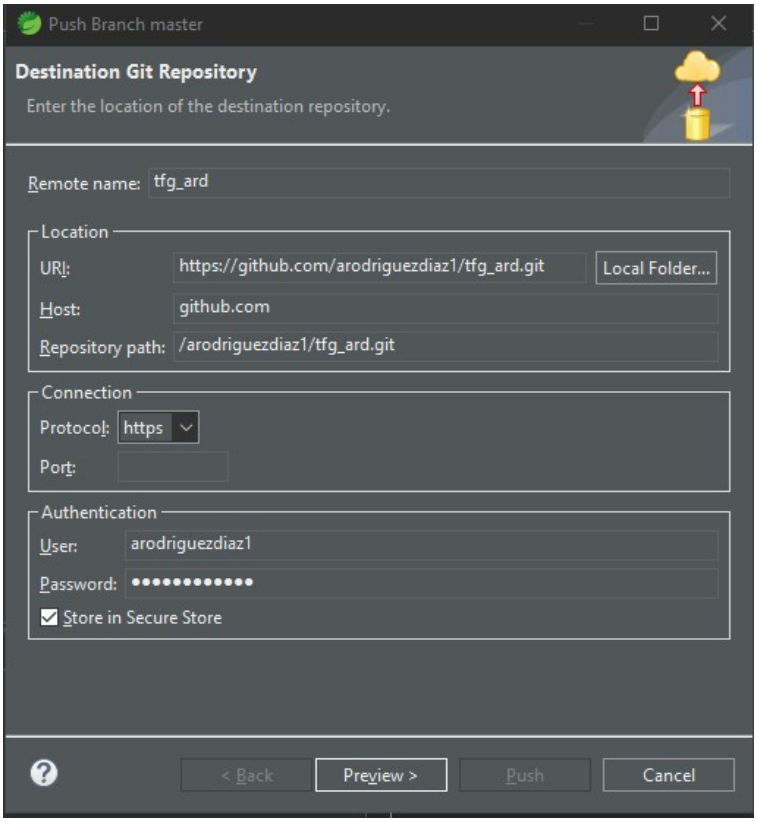


Ilustración 30 Parametros GitHub

Una vez finalizado el Push, verificamos en GIT que el proyecto esté subido:

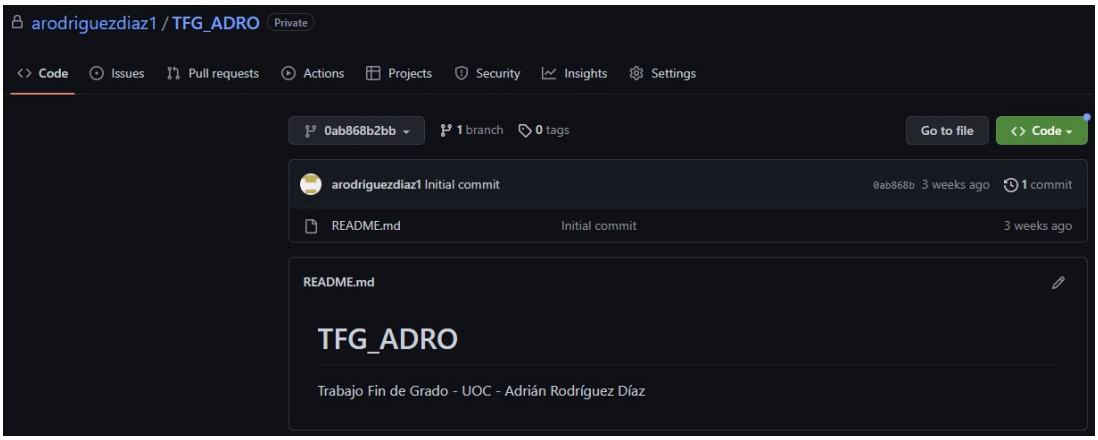


Ilustración 31 Primer Commit GitHub

3.5 Estructura del proyecto

Ya tenemos configurado nuestro proyecto y sincronizado con GitHub y con nuestra base de datos MySQL, por lo tanto, ya estamos preparados para comenzar el desarrollo de nuestra aplicación.

El primer paso será crear la estructura de nuestro proyecto, organizando las clases por paquetes.

La estructura de paquetes que tendremos será la siguiente:

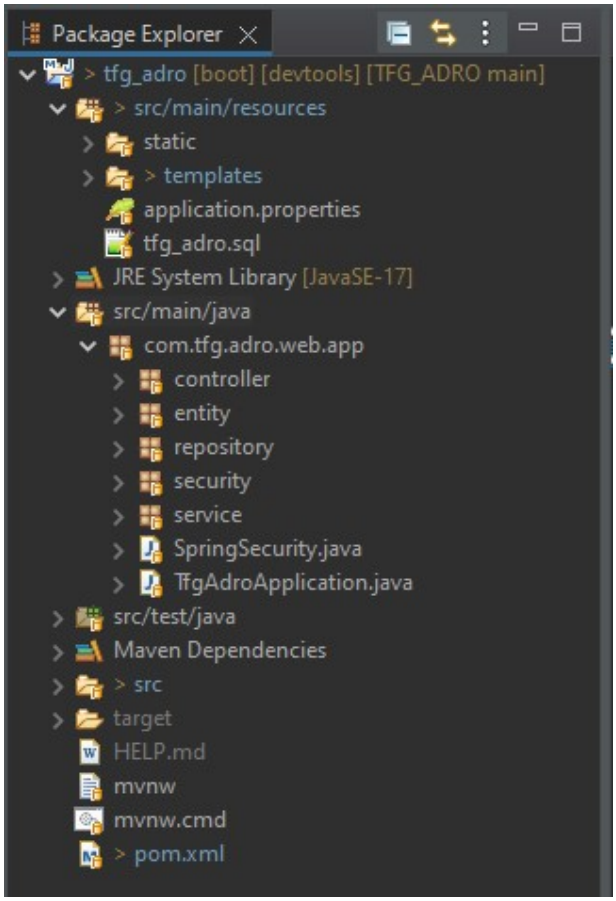


Ilustración 32 Estructura proyecto Java

Como se vé en la ilustración 32, hemos dividido la aplicación en varios paquetes:

- **Controller**: Aquí se guardarán los controladores, que serán los encargados de recibir las peticiones HTTP y responderlas, según las instrucciones que indiquemos en el código.
- **Entity**: En este paquete guardaremos las entidades, que serán los modelos de nuestros objetos, que luego los veremos en BBDD como nuestras tablas (Usuario, Actividades, Salas, etc.).

- **Repository**: Aquí definiremos los repositorios, serán las interfaces que extienden las propiedades de *JPARepository*, que básicamente se encargará de realizar las operaciones CRUD que necesitemos aplicar sobre la base de datos.
- **Security**: Definiremos nuestra clase de seguridad que implemente la interface *UserDetailsService*, que será donde definamos nuestras opciones de seguridad de Spring.
- **Service:** Dentro de este paquete, definiremos los servicios y su correspondiente implementación, estos serán los que comuniquen con las clases creadas en Repository, y se encargarán de indicar las operaciones que se van a realizar (Listar todos, obtener algún elemento por su Id, Guardar nuevos datos, borrar datos, etc.)

3.5 Codificación Java

En este punto definiremos los puntos más relevantes a la hora de la codificación, pero no entraremos en detalle ya que se haría un documento muy extenso.

3.5.1 Entidades

El primer paso de la codificación será configurar las entidades con las que vamos a trabajar, siguiendo el diseño de nuestro diagrama ERD realizado en el primer capítulo.

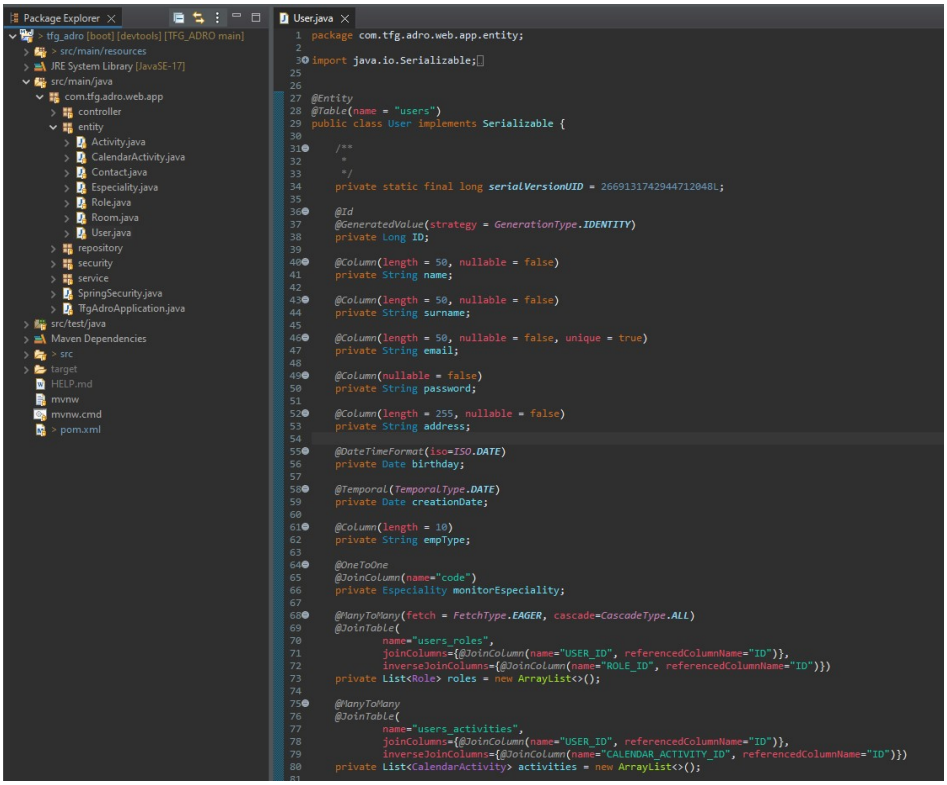


Ilustración 33 Entidad User – Ejemplo

Como vemos en la ilustración, indicamos que se nuestra clase se trata de una entidad a través de “@Entity”, y definimos también el nombre de la tabla que queremos en nuestra BBDD a través de “@Table(name=”users”)). Por convención, los nombres de las tablas de una BBDD tienen que estár en minúsculas y en plural.

Indicamos también cuál será el identificador de la tabla a través de “@Id” en la variable indicada y también definimos los parámetros que tienen que tener el resto de columnas.

Además, está entidad tiene una particularidad, y es que estará relacionada con 3 entidades más, que son *Roles*, *CalendarActivities* y *Especiality*. Estas 3 entidades representan los roles que tendrán (Administrador, Usuario, Monitor), las actividades a las que se ha registrado el usuario y la especialidad (especifica para los monitores) respectivamente.

```
@OneToOne
@JoinColumn(name="code")
private Especiality monitorEspeciality;

@ManyToMany(fetch = FetchType.EAGER, cascade=CascadeType.ALL)
@JoinTable(
        name="users_roles",
        joinColumns={@JoinColumn(name="USER_ID", referencedColumnName="ID")},
        inverseJoinColumns={@JoinColumn(name="ROLE_ID", referencedColumnName="ID")})
private List<Role> roles = new ArrayList<>();

@ManyToMany
@JoinTable(
        name="users_activities",
        joinColumns={@JoinColumn(name="USER_ID", referencedColumnName="ID")},
        inverseJoinColumns={@JoinColumn(name="CALENDAR_ACTIVITY_ID", referencedColumnName="ID")})
private List<CalendarActivity> activities = new ArrayList<>();
```

Ilustración 34 Relaciones entre entidades

Como vemos en la ilustración, hay 2 tipos diferentes de *joins*, el primero que es *OneToOne*, y es que la relación entre usuario y especialidad será 1 a 1, cada monitor tendrá una especialidad. Por otro lado tenemos dos relaciones de tipo *ManyToMany*, que ambas incluyen una nueva tabla que relaciona las entidades; por ejemplo, para la relación entre User y Role, se creará una tabla “users_roles” que contendrá 2 columnas, “USER_ID” y “ROLE_ID”, en la que se guardará el ID del usuario y el ID del rol que tenga asignado, al ser *ManyToMany*, los usuarios pueden tener más de 1 rol.

Una vez creadas todas las entidades, procedemos a iniciar el la aplicación. Como mencionamos anteriormente, veremos el log de creación y modificación de las tablas en la consola del IDE; al tener configurada la opción de “*spring.jpa.hibernate.ddl-auto=create*” se van a crear nuestras tablas y a actualizar sus propiedades.

Observamos el log de Spring:

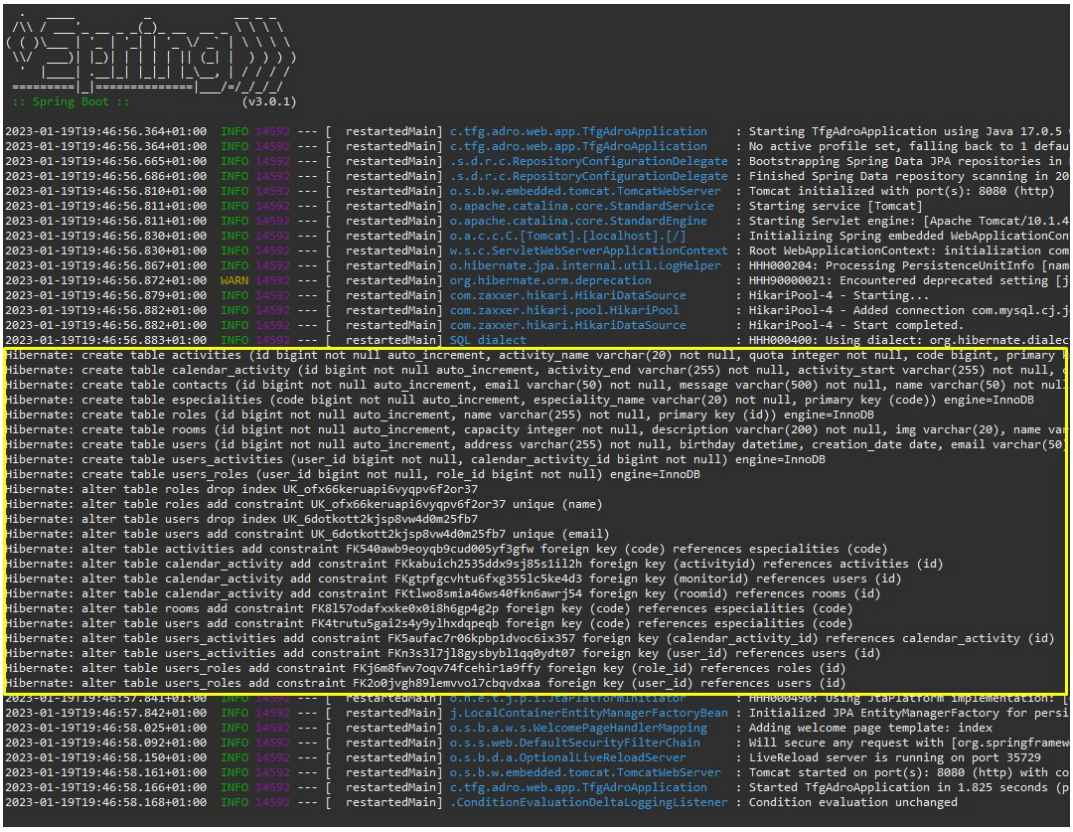


Ilustración 35 Creación de tablas a través de MySqlDialect

3.5.2 Repositorios

El paquete de repositorios contiene como mencionamos anteriormente a las interfaces que extenderán de *JpaRepository*. Simplemente contendrá la definición de los repositorios, y en caso de ser necesario, se creará una operación diferente a las que se encuentran en *JpaRepository*, como es el caso de los usuarios, ya que realizar la búsqueda por ID sería dificil, lo más adecuado sería buscarlos por ejemplo por el correo electrónico, ya que hemos definido en la entidad que debe ser único para cada usuario:

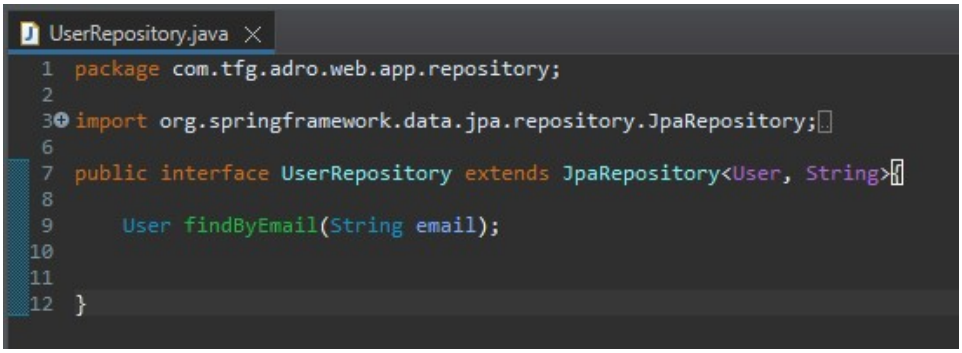


Ilustración 36 Repositorio UserRepository ejemplo

3.5.3 Seguridad

En esta carpeta en particular, definiremos la clase "*CustomUserDetailService"*, que extenderá de *UserDetailsService* y definirá las propiedades de los usuarios que se autentiquen en nuestra aplicación. Tendrá 2 operaciones que son las de *loadUserByUsername*, que en nuestro caso recibirán como mencionamos anteriormente el correo electrónico, lo buscará en la base de datos a través la operación "*findByEmail"* que hemos definido en el paso anterior y si existe, devolverá la información de este, de lo contrario arrojará una excepción de tipo *UsernameNotFoundException.*

La otra operación es un mapeo de los roles, que será llamada por la operación anterior para obtener el rol del usuario que busca.

CustomUserDetailsService.java | SpringSecurity.java

```
1  package com.tfg.adro.web.app.security;
2
3
4
5  import java.util.Collection;
18
19 @Service
20 public class CustomUserDetailsService implements UserDetailsService {
21
22     private UserRepository userRepository;
23
24     public CustomUserDetailsService(UserRepository userRepository) {
25         this.userRepository = userRepository;
26     }
27
28     @Override
29     public UserDetails loadUserByUsername(String email) throws UsernameNotFoundException {
30         User user = userRepository.findByEmail(email);
31
32         if (user != null) {
33             return new org.springframework.security.core.userdetails.User(user.getEmail(),
34                     user.getPassword(),
35                     mapRolesToAuthorities(user.getRoles()));
36         }else{
37             throw new UsernameNotFoundException("Nombre de usuario o contraseña incorrecta");
38         }
39     }
40
41     private Collection < ? extends GrantedAuthority> mapRolesToAuthorities(Collection <Role> roles) {
42         Collection < ? extends GrantedAuthority> mapRoles = roles.stream()
43                 .map(role -> new SimpleGrantedAuthority(role.getName()))
44                 .collect(Collectors.toList());
45         return mapRoles;
46     }
47 }
48
```

Ilustración 37 Seguridad CustomUserDetailsService

Por otro lado, tenemos otra clase que define la seguridad de Spring de nuestra aplicación. Esta clase se encuentra fuera del paquete de Security ya que tiene que estar al mismo nivel que la clase *main* principal.

Se trata de la clase que definimos como *SpringSecurity* y es donde definimos qué acceso tiene cada uno de los usuarios, según el rol que tengan asignado. Esta clase además, es la encargada de codificar la contraseña del usuario al momento de registrarse:

CustomUserDetailsService.java SpringSecurity.java

```java
package com.tfg.adro.web.app;

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

@Configuration
@EnableWebSecurity
public class SpringSecurity {

    @Autowired
    private UserDetailsService userDetailsService;

    @Bean
    static PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http.csrf().disable().authorizeHttpRequests((authorize) -> authorize.requestMatchers("/register/**").permitAll()
                .requestMatchers("/images/**").permitAll().requestMatchers("/index", "/").permitAll()
                .requestMatchers("/about").permitAll().requestMatchers("/salas").permitAll()
                .requestMatchers("/salas/**").permitAll().requestMatchers("/calendar").authenticated()
                .requestMatchers("/calendar/**").authenticated().requestMatchers("/calendar_date").authenticated()
                .requestMatchers("/calendar_date/**").authenticated().requestMatchers("/contact").permitAll()
                .requestMatchers("/contact_list").permitAll().requestMatchers("/contact_list/**").permitAll()
                .requestMatchers("/newactivity").permitAll().requestMatchers("/nueva_actividad").permitAll()
                .requestMatchers("/activity/save").permitAll().requestMatchers("/actividades_usuario").hasRole("USER")
                .requestMatchers("/actividades_monitor").hasRole("MONITOR").requestMatchers("/actividades_monitor/**")
                .hasRole("MONITOR").requestMatchers("/**").hasRole("ADMIN"))
                .formLogin(form -> form.loginPage("/login").usernameParameter("email").loginProcessingUrl("/login")
                        .defaultSuccessUrl("/").permitAll())
                .logout(logout -> logout.logoutRequestMatcher(new AntPathRequestMatcher("/logout")).permitAll());
        return http.build();
    }

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
    }

}
```

Ilustración 38 SpringSecurity.java

3.5.4 Servicios

Esta carpeta cuenta con 2 tipos de clases, por un lado tenemos las interfaces, que definiran las operaciones CRUD que vamos a utilizar sobre cada entidad; y por otro lado tendremos a la implementación de las operaciones de estas interfaces.

Por ejemplo, para la entidad User, definimos la interface *IUserService.java* y su implementación *UserServiceImpl.java:*

IUserService.java UserServiceImpl.java

```java
package com.tfg.adro.web.app.service;

import java.util.List;

public interface IUserService {

    public List<User> findAll();

    public void save(User user);

    public User findByEmail(String email);

    public void delete(String id);

    public void update(User user);

}
```

Ilustración 39 IUserService.java

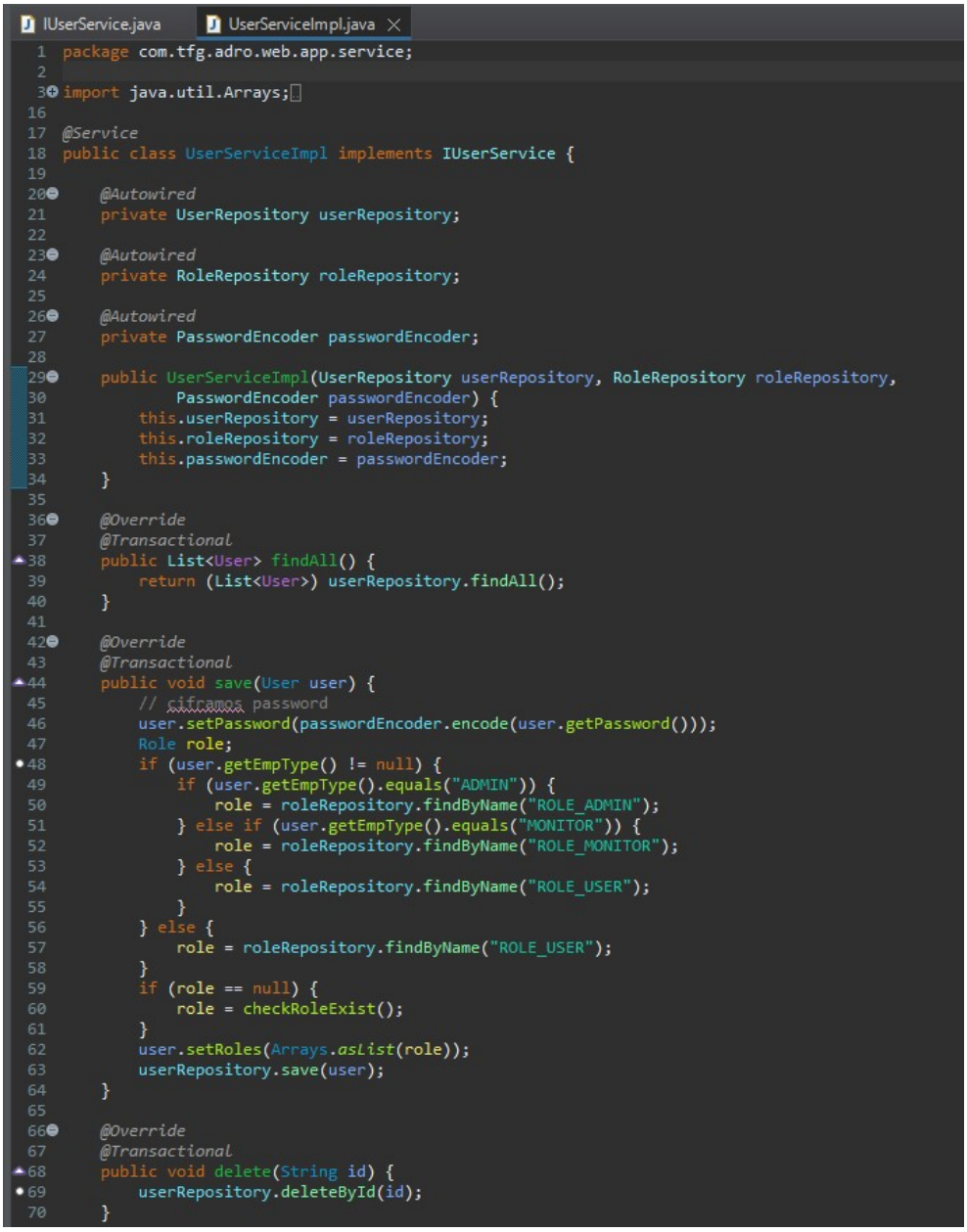


Ilustración 40 UserServiceImpl.java

3.5.4 Controladores

Por último, a nivel de java, tenemos a los controladores, estos serán los encargados de recibir las peticiones http (GET y POST), y según lo que reciban actuarán de una forma u otra.

Por ejemplo, para continuar con los usuarios, tenemos un controlador llamado “UserController”, que será el que utilice las operaciones definidas en los servicios definidos en el paso anterior. En este caso de ejemplo, tenemos una solicitud http de tipo “GET”, que indicará en la URL el valor “/listar”, y dentro de esta operación se recorrerán todos los usuarios que estén registrados en

nuestra base de datos y serán devueltas en un HTML llamado “listar”, difrenciados por su rol:

IUserService.java | UserServiceImpl.java | UserController.java

```java
package com.tfg.adro.web.app.controller;

import java.util.ArrayList;

@Controller
public class UserController {

    @Autowired
    private IUserService userService;

    @Autowired
    private IEspecialityService especialities;

    /*
     * Listado de usuarios (Administradores - Monitores - Usuarios)
     */
    @GetMapping("/listar")
    public String listar(Model model) {

        List<User> allUsers = userService.findAll();
        List<User> userMonitor = new ArrayList<>();
        List<User> userAdmin = new ArrayList<>();
        List<User> userClient = new ArrayList<>();

        for (int i=0; i<allUsers.size(); i++) {
            if(allUsers.get(i).getEmpType()!=null) {
                if(allUsers.get(i).getEmpType().equals("ADMIN")) {
                    userAdmin.add(allUsers.get(i));
                } else {
                    userMonitor.add(allUsers.get(i));
                }
            } else {
                userClient.add(allUsers.get(i));
            }
        }

        model.addAttribute("titulo", "Listado de clientes");
        model.addAttribute("userMonitor", userMonitor);
        model.addAttribute("userAdmin", userAdmin);
        model.addAttribute("userClient", userClient);
        return "listar";
    }
```

Ilustración 41 Controller - Listar clientes

Como podemos apreciar, tenemos en primer lugar la definición de “@Controller”, que indicará a Spring que se va a tratar de un controlador. Tenemos la definición de una variable de tipo “IUserService” que como mencionamos anteriormente tiene las operaciones CRUD. Y por último tenemos una operación indicada con la etiqueta “*@GetMapping*”, que indica la ruta que va a leer y el tipo de petición (get). Vemos que utiliza como parámetro de entrada un tipo “Model”, que será un mapa de tipo clave-valor, que se encargará de comunicarle al html los atributos que le definamos (en el ejemplo vemos que tenemos entre otros “titulo” con el valor “Listado de clientes”).

3.6 Codificación Front (Html + Bootstrap + Thymeleaf)

En este módulo tampoco nos exteneremos mucho, simplemente mostrar un ejemplo de nuestra aplicación y cómo se comunican la parte Frontal con el Back-End a través de las herramientas indicadas.

Comentaré la clase principal index.htm y mostraré un ejemplo de cada una de las partes:

En primer lugar veremos la cabecera definida en el html:

index.html

```
<head>
<title th:text="${titulo}"></title>
<meta charset="utf-8" />
<link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
    rel="stylesheet">
<script
    src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"></script>
<style type="text/css">
body {
    background-color: #212429;
    background-image: url("/images/background_login.jpg");
    opacity: 0.95;
    background-size: cover;
    background-repeat: no-repeat;
}
</style>
</head>
```

Ilustración 42 Index.html - head

En primer lugar se define el titulo. En este ejemplo vemos como se necesita la etiqueta “th:..” para indicar que es una etiqueta de Thymeleaf. Vemos que va a recibir una propiedad de tipo texto, y que ser informará mediante la variable “titulo”. Para indicar que el html va a recibir una variable lo indicamos con el símbolo de “$” y el nombre entre llaves: “*${titulo}*”.

Por otro lado tenemos un link, con la URL del css de Bootstrap. Es necesario indicarlo si queremos utilizar esta herramienta en nuestro diseño.

Además también tenemos un link hacia un script también de Bootstrap, que será necesario para utilizar las “animaciones” de esta herramienta, como la barra de navegación o el carrusel (que veremos más adelante).

Además, he añadido un estilo css a mi página, para indicar una imagen de fondo y algunas propiedades de diseño.

A continuación, se muestra la forma en la que se puede asignar un fragmento de tipo *header* y otro *footer* a través de *Thymeleaf* que puede estar definido en otras plantillas html sin tener la necesidad de codificarlo en cada una de las páginas que utilicemos. En nuestro caso la barra de navegación y el pie de página se utilizarán durante toda la navegación.

```
</head>
<div th:replace="~{fragments/header :: header}"></div>
<body>
<div th:replace="~{fragments/footer :: footer}"></div>
</html>
```

Ilustración 43Fragments header/footer

Y por último veremos el cuerpo del html:

```
<body>
    <div class="container-fluid bg-dark">
        <div id="carouselControls" class="carousel slide carousel-fade"
            data-bs-ride="carousel">
            <div class="carousel-inner">
                <div class="carousel-item active">
                    <img th:src="${'/images/background-img.jpg'}"
                        style="width: auto; object-fit: cover;">
                    <div class="carousel-caption"
                        style="width: auto; object-fit: cover;">
                        <h1>True Fit Gym</h1>
                        <h3>Sé más fuerte que tus excusas</h3>
                        <br> <br>
                    </div>
                </div>
                <div class="carousel-item">
                    <img th:src="${'/images/background2-img.jpg'}"
                        style="width: auto; object-fit: cover; position: center;">
                    <div class="carousel-caption"
                        style="width: auto; object-fit: cover;">
                        <h1>Welcome to True Fit Gym</h1>
                        <h3>Enjoy the Adventure!</h3>
                        <br> <br>
                    </div>
                </div>
            </div>
            <button class="carousel-control-prev" type="button"
                data-bs-target="#carouselControls" data-bs-slide="prev">
                <span class="carousel-control-prev-icon" aria-hidden="true"></span>
                <span class="visually-hidden">Previous</span>
            </button>
            <button class="carousel-control-next" type="button"
                data-bs-target="#carouselControls" data-bs-slide="next">
                <span class="carousel-control-next-icon" aria-hidden="true"></span>
                <span class="visually-hidden">Next</span>
            </button>
        </div>
    </div>
</body>
```

Ilustración 44 Index.html body

Como podemos ver, en nuestro body tenemos definido un contenedor de tipo *carousel*, que muestra varias imágenes de forma dinámica con un control para deslizarse hacia los lados.

Dentro de este carrousel tenemos 2 imágenes, definidas mediante las etiquetas de thymeleaf “th:src=..” para indicar la ruta donde están alojadas las imágenes, y además tenemos un titulo y un texto para cada una de ellas.

La página en acción la podemos ver a continuación:

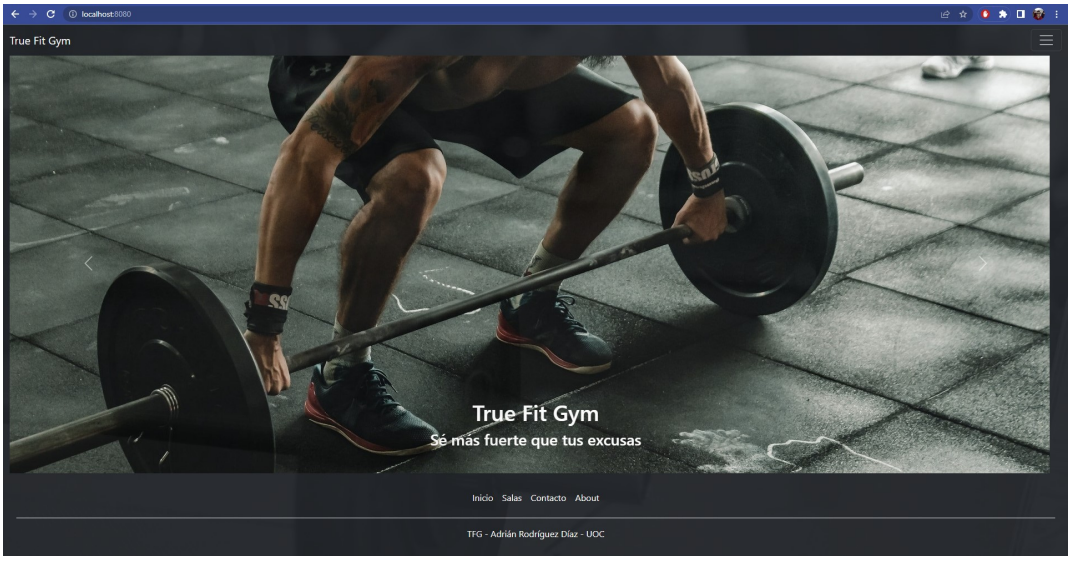


Ilustración 45 Index.html iniciado

4. Funcionamiento

4.1 Barra de navegación

Como podemos apreciar en la ilustración del capítulo anterior, vemos que tenemos la barra de navegación superior “header” y el pie de página “footer”.

La barra de navegación contiene un menú dinámico, que se contrae y se expande al precionar sobre el botón. Este menú cambiará según las diferentes condiciones:

- Usuario anónimo
- Usuario autenticado
- Monitor
- Administrador

En primer lugar veremos el menú de un usuario que no está autenticado (anónimo):

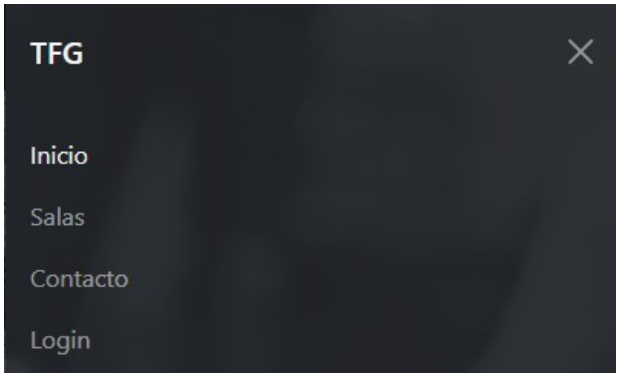


Ilustración 46 Navbar anónimo

Como podemos apreciar, las opciones que tienen los usuarios no autenticados son mínimas, pueden acceder al menú principal, a ver las salas del gimnasio y su descripción, autenticarse o enviar un mensaje a través de la herramienta “Contacto”.

Cuando el usuario ya está autenticado y se trata de un usuario con rol de tipo “User” (cliente del gimnasio), verá las siguientes opciones:

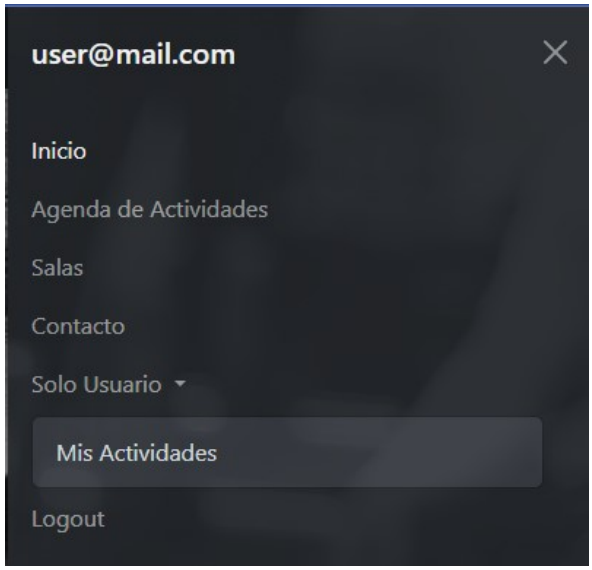


Ilustración 47 navbar rol "User"

Como podemos apreciar ya tiene más opciones. En primer lugar vemos que aparece la opción “agenda de actividades”, que nos guiará hacia la página que contenga esta información.

Además, puede acceder a todas las actividades a la que se ha registrado a través de “Solo Usuario > Mis Actividades”.

En tercer lugar tenemos a los usuarios con rol de “Monitor”, este tipo de usuario verá la siguiente información:

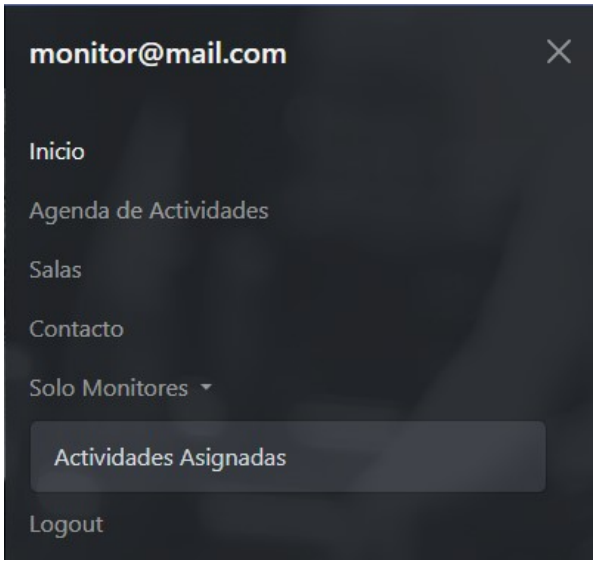


Ilustración 48 navbar rol Monitor

Como vemos, tiene la misma información que un cliente, pero en este caso tiene un menú que es sólo para los usuarios con este rol, al que pueden acceder a las actividades que se les ha asignado.

Por último, tenemos al rol más completo que es el de los administradores, en este caso tienen acceso a todas las herramientas de nuestro sitio:

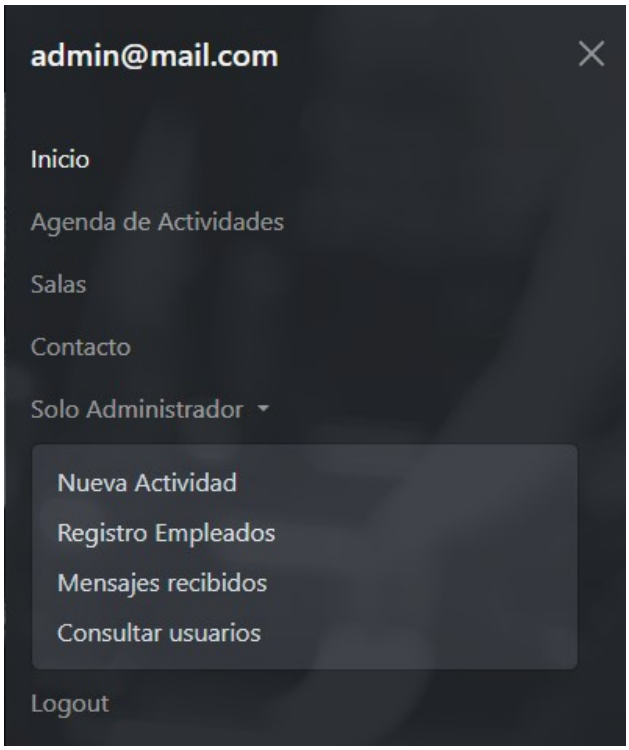


Ilustración 49 navbar rol Admin

Las opciones a destactar son:

- Nueva Actividad: Aquí es donde tendremos un formulario donde se van a registrar las nuevas actividades que realizará el gimnasio. Dentro de la misma se elige también la sala y el monitor de la actividad.

- Registro Empleados: Aquí es donde el administrador puede crear a nuevos administradores y monitores, además a estos últimos le puede asignar una especialidad.

- Mensajes Recibidos: En este apartado, el administrador podrá leer los mensajes que ha recibido el centro mediante el formulario de “Contacto”. Puede además borrarlos una vez que se haya contactado con el usuario.

- Consultar Usuarios: Aquí verá un listado con todos los usuarios registrados en la página, diferenciados por su rol.

4.2 Login de usuarios

El login de usuarios se trata de un formulario con 2 datos, el Email del usuario y su contraseña, en caso de que el usuario no esté ingresando los datos correctamente aparecerá un mensaje indicando el fallo.

Además, si el usuario no se ha registrado aún, tiene la opción de registrarse a través de el enlace "*Registrese aquí*".

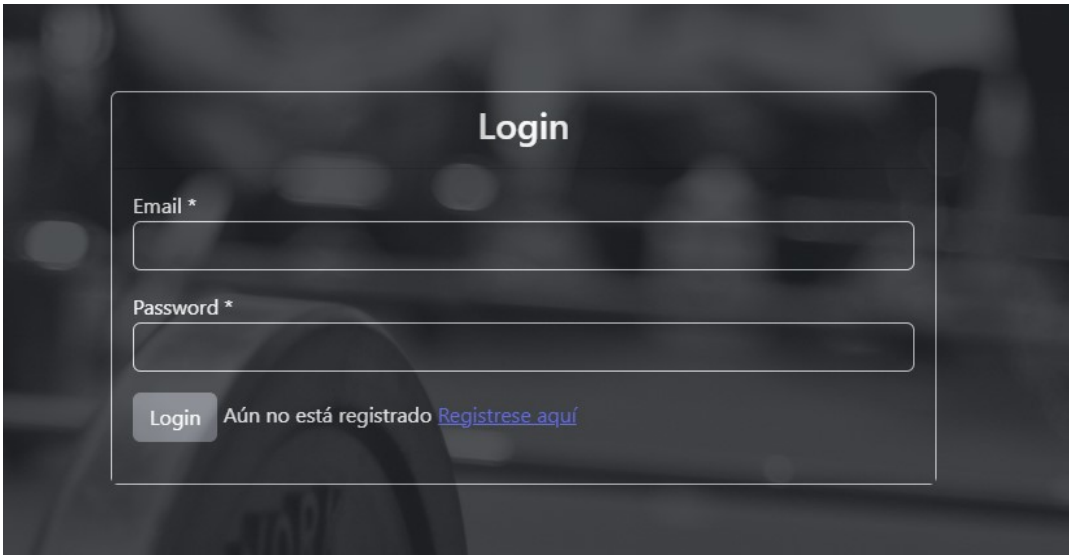


Ilustración 50 Formulario Login

Ejemplo de login incorrecto:

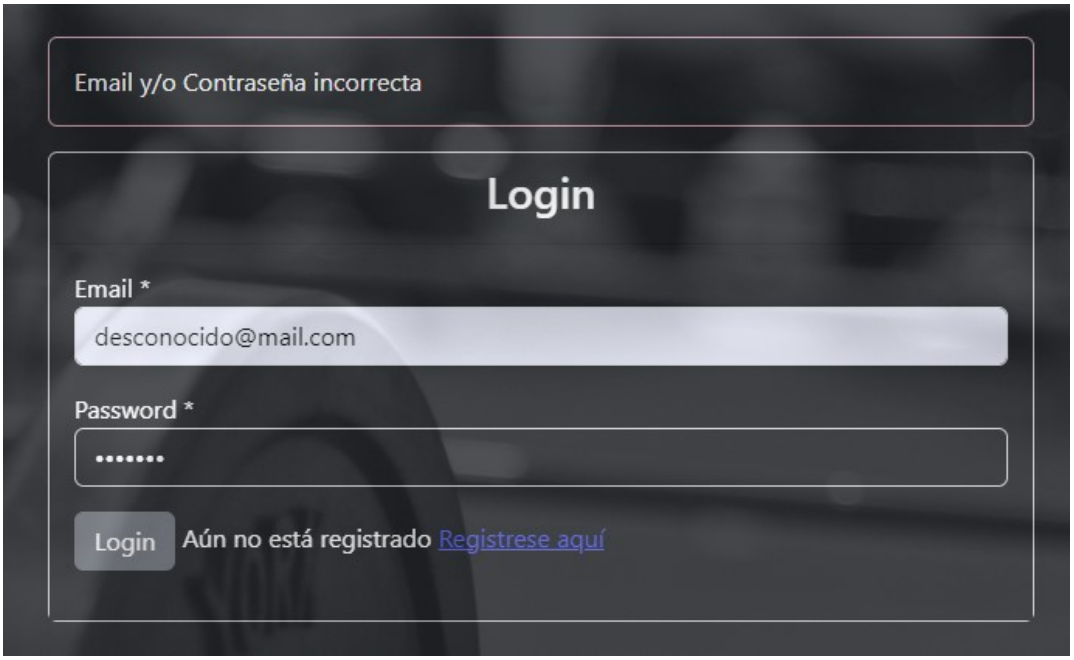


Ilustración 51 Login incorrecto

4.3 Registro de usuarios

En caso de que en el paso anterior se haya seleccionado la opción de registro, el formulario que verá será el siguiente:

Ilustración 52 Formulario de registro

Una vez registrado ya puede proceder a loguearse a través del enlace del punto anterior.

4.4 Salas

En este apartado, el usuario podrá navegar entre las diferentes salas que ofrece el gimnasio, consultar su capacidad y ver la descripción de esta.

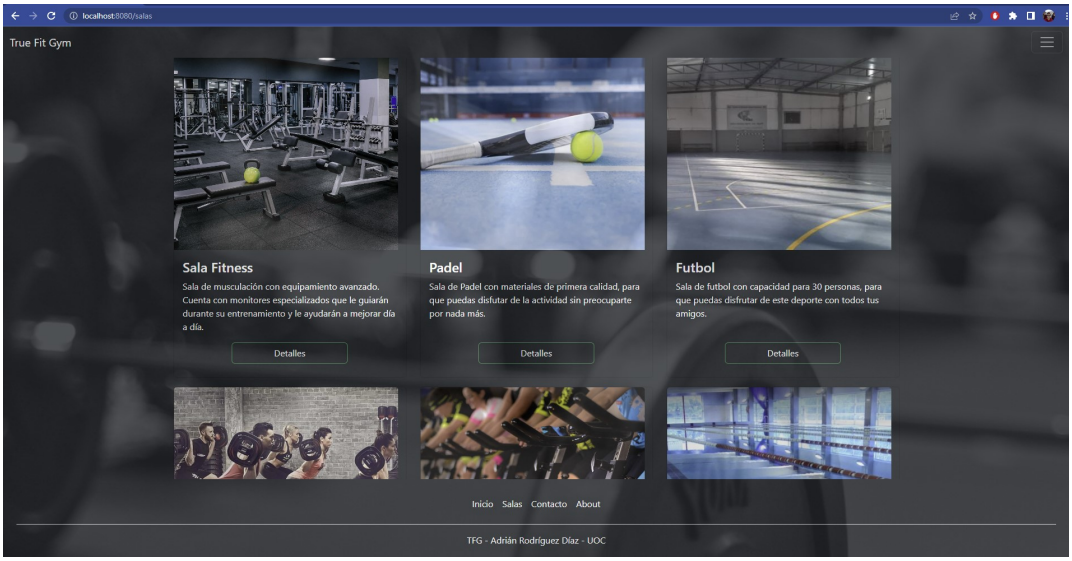


Ilustración 53 Salas

Si hacemos clic sobre “Detalles” en cada una de las salas, se abrirá una nueva página con la información de la sala seleccionada:

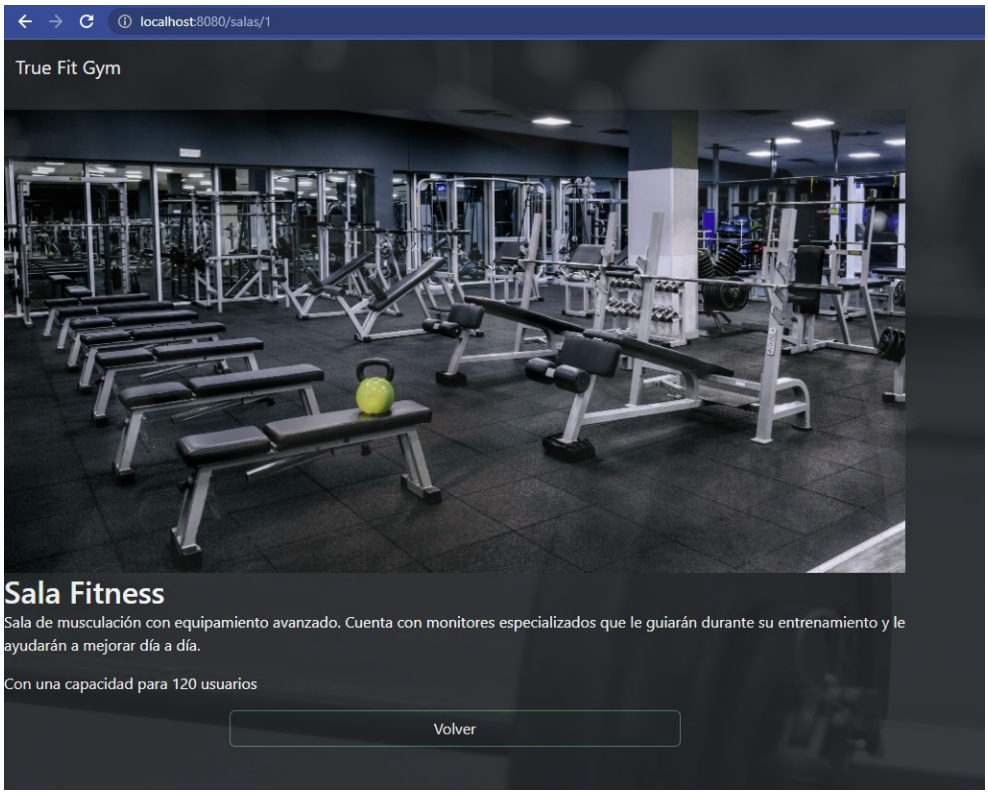


Ilustración 54 Salas – Detalles

Como podemos apreciar en la URL vemos que se indica “/salas/..” + el ID de la sala. Esto se mapea en el controlador correspondiente y se indica que lo que va a recibir como parámetro será un ID. El código se muestra a continuación:

```
    @GetMapping("/salas/{id}")
    public String roomDetail(@PathVariable("id") Long roomId, Model model) {

        model.addAttribute("titulo", "Detalle de sala");
        model.addAttribute("room", room.findById(roomId));
        return "sala_detail";
    }


}
```

Ilustración 55 Controlador de Sala

Vemos que devuelve la página html “sala_detail” y que le envía dentro de un parámetro “room” el objeto de tipo Room que obtiene a través del ID.

4.5 Formulario de contacto:

El formulario de contacto es sencillo, simplemente el usuario debe ingresar los datos y éstos serán grabados en la base de datos para que luego el administrador los pueda ver:

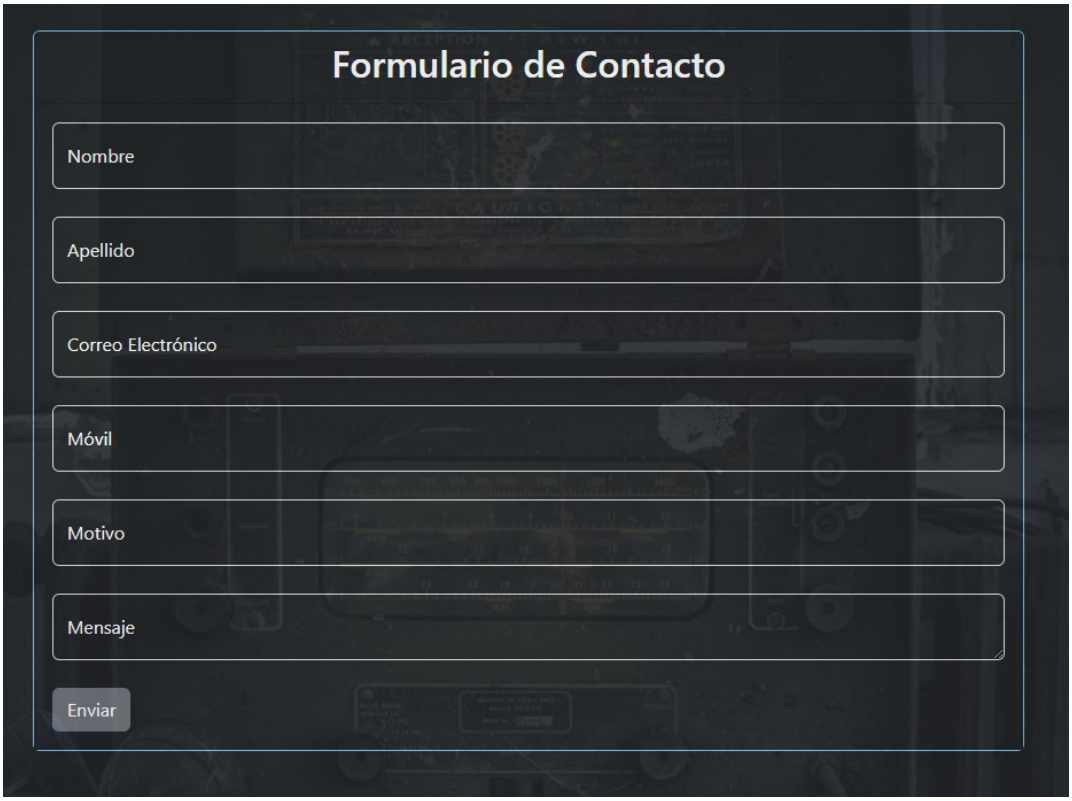


Ilustración 56 Formulario de contacto

4.6 Alta de Actividad (Solo Administrador):

En este caso nos encontramos con un formulario, en el que el administrador dará de alta una actividad para una fecha en concreto y podrá seleccionar la sala y el monitor que aparecerán en el menú de selección.

Por ejemplo:

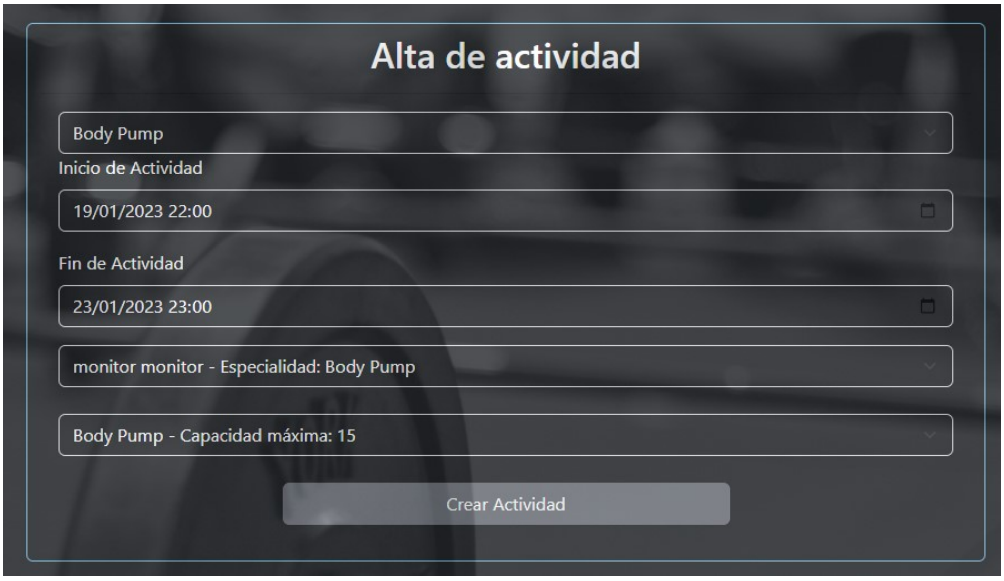


Ilustración 57 Alta de Actividad

Una vez creada la actividad ya se puede ver dentro de la agenda de actividades que veremos a continuación.

4.7 Agenda de actividades:

Dentro de la agenda de actividades podemos ver en primer lugar las actividades del día en el que nos encontramos, su información de horario, sala y monitor asignado:

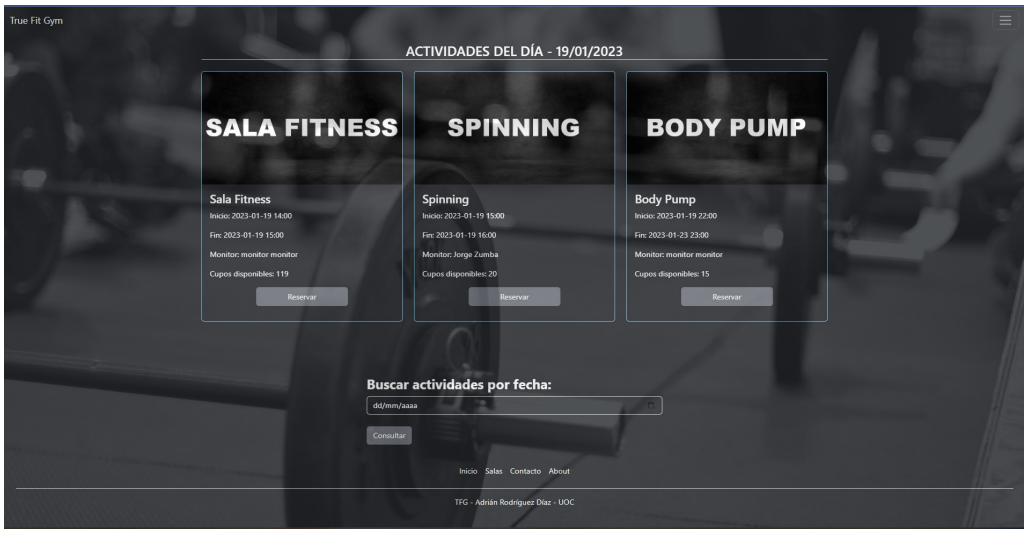


Ilustración 58 Agenda de actividades

Además en esta página nos podemos agendar en las distintas actividades que se ven, y podemos también realizar una búsqueda de actividades por fecha en el menú inferior, para poder realizar reservas unicamente en actividades futuras.

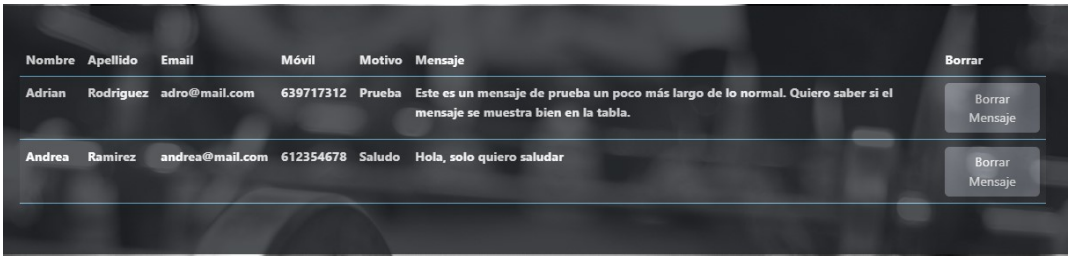
4.8 Registro de empleados:

A este apartado solamente pueden acceder los administradores, y lo utilizarán para registrar nuevos empleados de tipo “Administrador” o “Monitor”. Para los monitores aparecerá una nueva opción que será su especialidad:

Ilustración 59 Registro de empleado - Administrador

Ilustración 60 Registro de empleado - Monitor

4.9 Mensajes Recibidos:



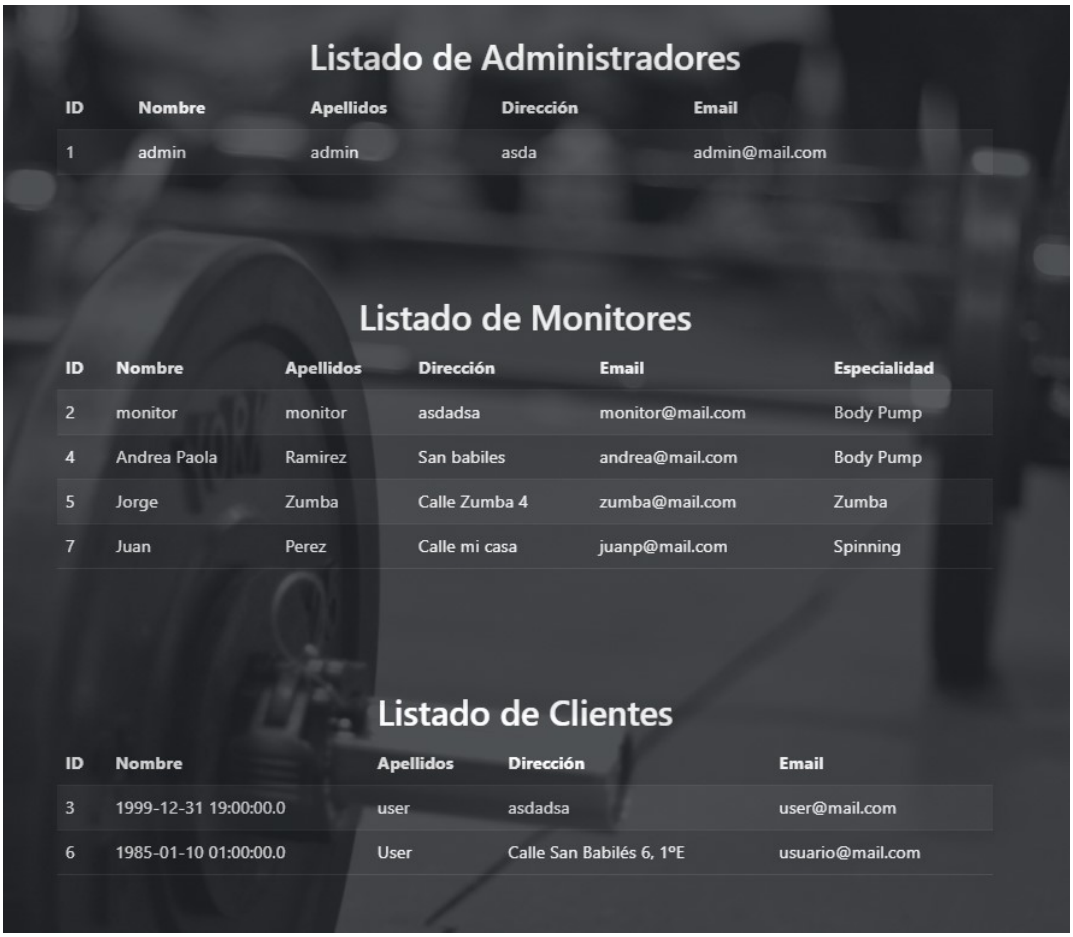
Nombre	Apellido	Email	Móvil	Motivo	Mensaje	Borrar
Adrian	Rodriguez	adro@mail.com	639717312	Prueba	Este es un mensaje de prueba un poco más largo de lo normal. Quiero saber si el mensaje se muestra bien en la tabla.	Borrar Mensaje
Andrea	Ramirez	andrea@mail.com	612354678	Saludo	Hola, solo quiero saludar	Borrar Mensaje

Ilustración 61 Mensajes recibidos

En esta página se podrán consultar los mensajes que se reciben a través del formulario de contacto. Tiene la opción de borrarlos una vez consultados.

4.10 Consultar usuarios:

Como mencionamos anteriormente, esta opción es unicamente para los administradores, y se obtienen tres listas con los usuarios asignados a cada uno de los roles



Listado de Administradores

ID	Nombre	Apellidos	Dirección	Email
1	admin	admin	asda	admin@mail.com

Listado de Monitores

ID	Nombre	Apellidos	Dirección	Email	Especialidad
2	monitor	monitor	asdadsa	monitor@mail.com	Body Pump
4	Andrea Paola	Ramirez	San babiles	andrea@mail.com	Body Pump
5	Jorge	Zumba	Calle Zumba 4	zumba@mail.com	Zumba
7	Juan	Perez	Calle mi casa	juanp@mail.com	Spinning

Listado de Clientes

ID	Nombre	Apellidos	Dirección	Email
3	1999-12-31 19:00:00.0	user	asdadsa	user@mail.com
6	1985-01-10 01:00:00.0	User	Calle San Babilés 6, 1ºE	usuario@mail.com

Ilustración 62 Listado de Usuarios

4.11 Mis Actividades:

Esta página es para que los clientes puedan acceder al historico de actividades a las que ha participado:

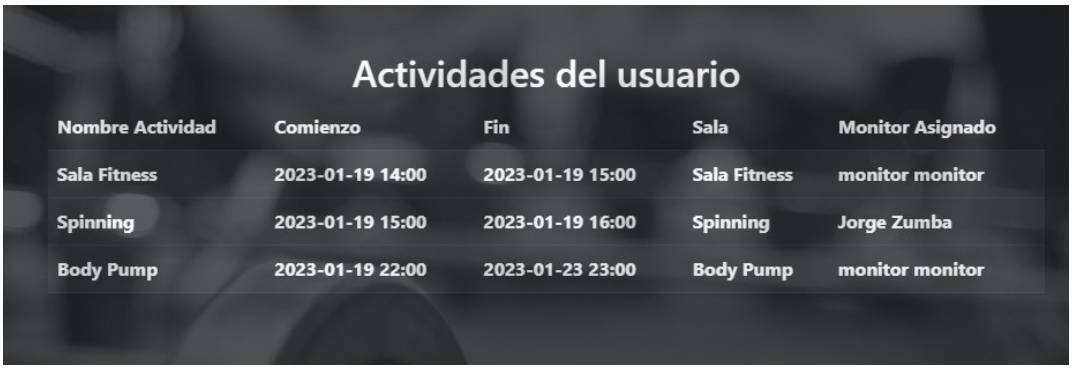


Ilustración 63 Listado actividades usuario

4.12 Actividades asignadas al monitor:

En este caso, veremos un listado con todas las actividades a las que ha sido asignado un monitor (el que esté autenticado). Podrá ver la fecha, la cantidad de clientes que participaron, el nombre de la actividad y además, tendrá la opción de ver los detalles de la actividad, que lo redirigirá a un nuevo listado en el que aparecen los clientes que participaron:

Ilustración 64 Actividades del Monitor

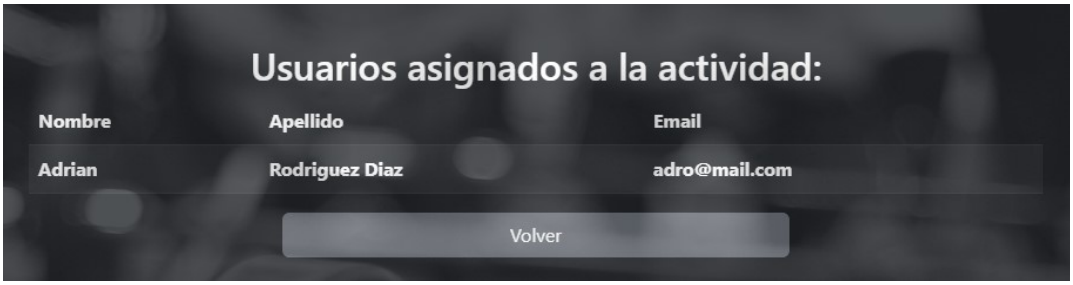


Ilustración 65 Usuarios dentro de la actividad

5. Conclusiones

5.1 Reflexiones sobre el trabajo

Como conclusión sobre la realización de este trabajo de fin de grado, puedo decir que significa la finalización de una carrera que ha abarcado gran parte de mi vida, un camino que ha comenzado en mi país (Uruguay), y que ha pasado por muchos cambios a lo largo de mi vida, en los que he tenido que adaptarme a muchas situaciones diferentes, buenas y malas, pero en la que nunca he perdido el foco que era llegar a la meta.

Durante la realización del proyecto he aprendido muchas cosas nuevas, en primer lugar a realizar un análisis en profundidad de lo que quería conseguir, realizar una planificación temporal con varios hitos a cumplir durante el recorrido.

También he aprendido acerca de estas nuevas tecnologías, he intentado y creo que conseguido trabajar con las últimas versiones de las tecnologías planteadas, en las que me ha costado mucho conseguir información cada vez que me topaba con algún error. Me ha llevado más tiempo de dedicación del que he estimado en un comienzo. Me ha dejado con varias malas noches en las que por algún momento se cruzó por mi cabeza arrojar la toalla y volver a intentarlo más adelante, pero gracias a el apoyo de mi familia y la confianza que depositaron en mi no podía permitirmelo.

5.2 Planificación y metodología

En este punto creo que he comenzado con las entregas a tiempo y he cumplido con las tareas requeridas, pero no he contado con que al final del proyecto nos encontrabamos con fechas complicadas (fiestas de navidad, fin de año, entregas de otras asignaturas, etc.) y me he confiado con el tiempo que tenía. Al final he necesitado una ampliación de tiempo de entrega de la implementación para poder cumplir con los requerimientos planteados de la aplicación.

5.3 Trabajo futuro

Si bien estoy satisfecho con el resultado final, creo que existen puntos de mejora que podrían añadir valor al producto final. Por ejemplo algunos detalles como añadir imagenes a los usuarios para hacer la web más dinámica y ta,bién que se puedan gestionar las cuotas pagas e impagas de los usuarios, con posibilidad de inhabilitarlos en caso de incumplimiento.

También que exista la posibilidad de crear actividades fijas en el tiempo, para que el administrador no tenga la necesidad de crearlas todos los días.

Además me gustaría mejorar el manejo de salas y monitores al momento de crear las actividades, ya que actualmente al seleccionar la actividad se puede elegir una sala que no corresponde con la de la actividad y asignar un monitor cuya especialidad no es la indicada para dicha actividad.

6. Glosario

CRUD: Corresponde con las siglas Create + Read + Update + Delete, que son las operaciones básicas que se pueden realizar sobre una entidad en este caso.

IDE: Corresponde a el “Entorno de Desarrollo Integrado”, que es la aplicación informática sobre la que desarrollaremos nuestro código, que se encargará de compilar y ejecutar el proyecto.

UML: Corresponde con los diagramas de “Lenguaje Unificado de Modelado”, cuyo proposito es proporcionar una forma estándar de visualizar el diseño de un sistema.

XAMPP: Es el nombre de la herramienta que utilizamos para crear nuestra base de datos en nuestro entorno local. La **X** corresponde a que se puede utilizar bajo cualquier sistema operativo, la **A** es de Apache, **M** corresponde a la base de datos que puede ser “MariaDB” o “MySQL”, la **P** de PHP (lenguaje de programación) y la última **P** de Pearl (lenguaje de programación).

7. Bibliografía

https://blog.ganttpro.com/es/ganttproject-caracteristicas-ventajas-y-desventajas/

(Consultada durante la realización del plan de trabajo 28/09/2022 al 11/10/2022)

https://www.w3schools.com/bootstrap5/index.php (Consultado durante la etapa de implementación 12/11/2022 al 02/01/2023)

https://getbootstrap.com/docs/5.3/getting-started/introduction/ (Consultado durante la etapa de implementación 12/11/2022 al 02/01/2023)

https://www.thymeleaf.org/documentation.html (Consultado durante la etapa de implementación 12/11/2022 al 02/01/2023)

https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/
(Consultado durante la etapa de implementación 12/11/2022 al 02/01/2023)

https://stackoverflow.com/ (Consultado durante la etapa de implementación para obtener ayuda acerca de errores conocidos 12/11/2022 al 02/01/2023)

8. Anexos

Contenido del fichero .zip de la entrega:

1. Código fuente de la última versión de la aplicación
2. Memoria en formato PDF
3. Video de presentación de la aplicación.
4. Manual de despliegue local de la aplicación