Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior (Jaén)

Trabajo Fin de Grado

APLICACIÓN WEB PARAMETRIZABLE PARA LA GESTIÓN DE RESERVA DE ESPACIOS DEPORTIVOS

Alumno: Peña Cuevas, Jose María

Tutor: Rueda Ruiz, Antonio Jesús
Dpto: Departamento de Informática

Tutor: Luque Luque, Adrián
Dpto: Departamento de Informática

Abril, 2022

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Antonio Jesús Rueda Ruiz y Don Adrián Luque Luque, tutores del Proyecto Fin de Carrera titulado: Aplicación web parametrizable para la gestión de reserva de espacios deportivos, que presenta Jose María Peña Cuevas, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, 22 de Abril de 2022

El alumno:

PEÑA CUEVAS JOSE MARIA - 15520765C
Firmado digitalmente por PEÑA CUEVAS JOSE MARIA - 15520765C
Fecha: 2022.04.26 10:02:05 +02'00'

Jose María Peña Cuevas

Los tutores:

Firmado digitalmente por RUEDA RUIZ ANTONIO JESUS - 33395227D
Fecha: 2022.04.26 09:53:28 +02'00'

Antonio Jesús Rueda Ruiz

LUQUE LUQUE ADRIAN - 77324298T
Firmado digitalmente por LUQUE LUQUE ADRIAN - 77324298T
Fecha: 2022.04.25 16:54:48 +02'00'

Adrián Luque Luque

AGRADECIMIENTOS

Me gustaría darle las gracias a mi familia y amigos que han estado durante los años de carrera apoyándome tanto en lo bueno como en lo malo.

A mis profesores por todo el conocimiento brindado y en especial a mis tutores por la ayuda que me han prestado en todo momento.

Gracias a todos ellos he sido capaz de conseguir mis objetivos.

Índice

Índice de ilustraciones

Índice de tablas

1. INTRODUCCION

El propósito de este capítulo es brindar al lector una visión general de este proyecto, explicando su contexto, propósito, estudio de alternativas existentes, descripción de la aplicación, objetivos que debemos de alcanzar y la estructura de la memoria.

1.1. Contexto

El crecimiento de la práctica deportiva en España es un hecho. Según la última encuesta sobre hábitos deportivos realizada por el Ministerio de Educación, Cultura y Deporte, el 65,1% de la población española mayor de 15 años practicó deporte durante este año. Actualmente, según datos proporcionados por instalaciones públicas y privadas, y a falta de datos oficiales del Ministerio, esta cifra sigue creciendo, lo que confirma el buen estado del sector deportivo en España.

Gracias al último Censo Nacional de Instalaciones Deportivas [1] realizado en 2005, podemos conocer que el número total de instalaciones en aquel año en España era de 79.059. Estas albergaban un total de 176.201 espacios deportivos. Han pasado 17 años desde que se realizara dicho Censo, por tanto, si por aquel entonces ya había un número importante de instalaciones, es de esperar que, hoy en día habrá muchísimas más.

Si hablamos de comunidades autónomas, encontramos en los primeros puestos a Andalucía con 12.831 instalaciones y 26.391 espacios deportivos, y a Cataluña con 12.478 instalaciones y 31.560 espacios deportivos.

En cuanto al impacto económico [2], la industria del deporte aporta un 3,3% del PIB y genera alrededor de 414.000 puestos de trabajo, lo que supone el 2,1% del empleo.

Como podemos ver, el deporte tiene un efecto multiplicador en la economía ya que, por cada euro facturado, el resto de la economía genera 1,50 euros extra. Este crecimiento también tiene un efecto positivo en el empleo: por cada millón de euros que factura la industria del deporte en España se crean 26 puestos de trabajo.

La facturación de las instalaciones deportivas es de 2.508 millones de euros, únicamente superada por clubes con 5.881 millones de euros y tiendas especializadas con 4.988 millones de euros. Sin embargo, cuando se trata de empleo, se sitúan a la cabeza. Estas emplean a 57.400 personas, lo que representa el 29% de todos los puestos de trabajo en la industria deportiva.

Según un estudio [3] realizado en 2017, las reservas de instalaciones a través de apps han crecido más de un 50% respecto de 2016. En datos económicos, este crecimiento representa una facturación que supera el 25%.

Este cambio no solo beneficia a los centros deportivos sino también a los usuarios ya que antes debían de llamar por teléfono, escribir un correo o acercarse al centro deportivo, ahora con las apps únicamente deben acceder, revisar la disponibilidad de las instalaciones y realizar la reserva.

Por ultimo y no menos importante, la pandemia ha impulsado aún más el uso de estas aplicaciones.

1.2. Propósito

El propósito de nuestro proyecto es el desarrollo de una aplicación web accesible desde internet para la gestión de espacios deportivos. Gracias a la parametrización de la configuración, esta podrá ser instalada en diferentes entornos y distribuida con el uso de contenedores.

La aplicación web vendrá complementada con diferentes clientes como: página web o aplicación móvil.

En cuanto a la funcionalidad, contará con un sistema integrado de gestión de usuarios, sistema de alertas/avisos, sistema de pago de reservas, etc.

El objetivo es conseguir un sistema fácilmente distribuible y lo suficientemente flexible para poder usarlo en varios espacios deportivos sin necesidad de codificar personalizaciones para cada uno de ellos.

Lo que buscamos es que personas con limitados conocimientos en informática, guiados a través de la configuración de la web, sean capaces de desplegar una

aplicación totalmente funcional, personalizada y única para el centro deportivo que lo concierne.

Como todos conocemos, las aplicaciones web enfocadas al alquiler de pistas en espacios deportivos tienen funcionalidades en común, como la visualización de pistas disponibles, visualización de horarios, gestión a través de un usuario administrador, etc. Por ello, pensamos que es innecesario desarrollar una aplicación web desde cero para cada espacio deportivo que desee contar con una web de estas características.

Por último, nuestra aplicación a diferencia de las demás, que son entregadas al cliente listas para usar, serán configuradas y personalizadas por los propios clientes, de esta manera conseguiremos que estos que son los que harán uso de la aplicación, se familiaricen con ella y les sea más fácil la utilización de esta para la gestión del centro deportivo.

1.3. Estudio de alternativas existentes

Existen multitud de aplicaciones para la gestión y alquiler de instalaciones en espacios deportivos como MyTurn, Sporttia, Playtomic, R24h, ZCENTER, etc. Por ello, en este apartado realizaremos un estudio acerca de algunas de estas para así poder encontrar carencias existentes, justificar nuestra aplicación a realizar, etc.

1.3.1. R24h

Ilustración 1.1 R24h

Es un software [4] para la gestión de instalaciones deportivas utilizado por universidades, polideportivos, etc. Está basado en una aplicación web compuesta por diferentes módulos configurables que se instalan dependiendo de las necesidades de cada cliente. Los usuarios y administradores pueden hacer uso de esta a través de cualquier dispositivo.

Algunas de las funcionalidades con las que cuenta:

- Gestión de usuarios

- Realización de reservas online
- Inscripción online a actividades, cursos
- Envío de encuestas a usuarios

La Ilustración 1.2 muestra la captura de la pantalla principal de la aplicación:

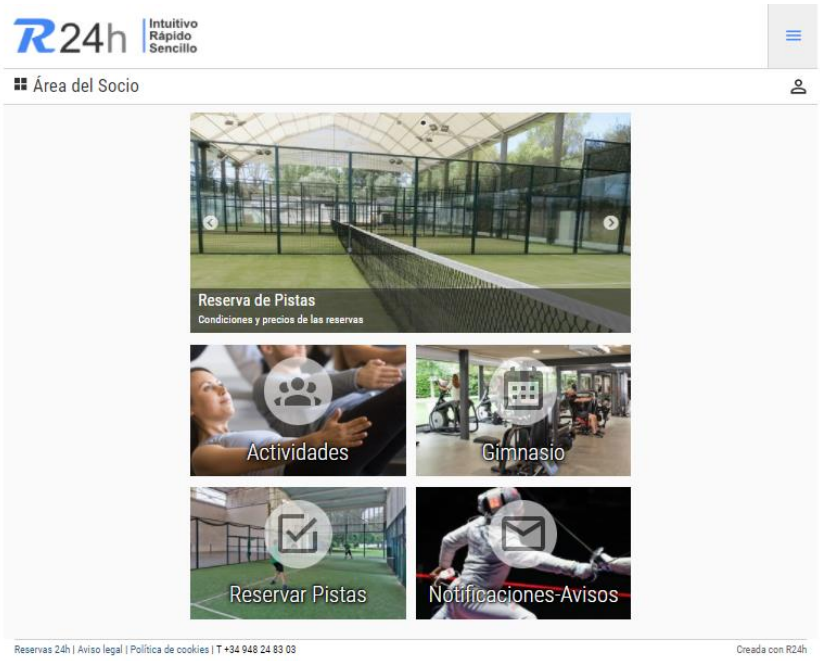


Ilustración 1.2 Pantalla Principal R24h

Como podemos ver cuenta con una interfaz sencilla e intuitiva. Para el caso del cliente móvil, la interfaz es la misma, pero ajustándose a las dimensiones de este.

1.3.2. ZCENTER

Software para la gestión deportiva [5] en centros y entidades de cualquier tamaño. La plataforma está compuesta por diferentes aplicaciones y módulos que permiten adaptarla a las necesidades de cada centro deportivo.

Entre sus funcionalidades cuenta con: aplicación de administración para el centro, tramites automáticos, web y aplicación móvil de reservas, planificación de rutinas personalizadas, control de acceso mediante puertas con cerradura electrónica, etc.

Por último, gracias a la combinación de hardware y software, permite una completa gestión online de los centros deportivos.

1.4. Descripción de la Aplicación

Para empezar a familiarizarnos con la aplicación a diseñar y obtener una idea aproximada, en este apartado realizaremos una descripción acerca de esta.

Nuestra aplicación como ya hemos comentado anteriormente, será accesible a través de internet, los usuarios contarán con dos roles diferentes, rol 'User' y rol 'Admin'.

Los usuarios con rol 'User' podrán realizar una o varias reservas de una instalación determinada, siempre y cuando, esta se encuentre disponible. El usuario podrá hacer efectiva dicha reserva seleccionando el método de pago una vez haya rellenado los datos de la reserva. Si al realizar el pago se produce un error, por ejemplo, de red, podrá pagarla desde su perfil de usuario. Una vez pagada la reserva, se le enviará un correo al usuario con los detalles de esta y se almacenará en la aplicación, pudiendo ser consultada en cualquier momento.

Cada usuario registrado y logueado podrá modificar sus datos de perfil, visualizar sus reservas y, además, toda la funcionalidad con la que usuarios no registrados pueden hacer uso de ella como: realizar consultas al centro deportivo a través de un formulario de contacto, consultar toda la información acerca del centro deportivo, consultar los días que no se pueden reservar las instalaciones, visualizar las instalaciones junto con sus horarios, etc.

El usuario con rol 'Admin' podrá gestionar todo aquello con lo que cuente la aplicación y el centro deportivo: instalaciones, horarios, usuarios, reservas, días no reservables, información detallada del centro deportivo, etc.

Por último, la configuración y personalización de la aplicación se llevará a cabo la primera vez que se inicie esta. El usuario 'Admin' irá rellenando cada uno de los formularios que se le presenten y una vez terminada, obtendrá una aplicación totalmente funcional y única.

Para la aplicación en cuestión, realizaremos la parte del Backend y del Frontend. Todo el diseño y desarrollo se realizará para un cliente web, y que, siguiendo la técnica

‘Responsive Web Design’ podrá ser accedida desde cualquier cliente, ya sea móvil, Tablet, etc.

Cada uno de nuestros clientes será capaz de configurar y personalizar la aplicación únicamente introduciendo los datos necesarios, consiguiendo así familiarizarse con ella y por consecuente, ser capaz desde el primer momento de gestionar en su totalidad el centro deportivo, ya sea modificando las instalaciones, desactivándolas, etc. Sin embargo, en las alternativas que hemos estudiado anteriormente, los clientes que contratan estas aplicaciones, cuentan con un soporte técnico continuo donde cada parte de la aplicación les es explicada, etc. En este caso, estos, deberán contar con muchas horas de práctica para poder entenderla y usarla perfectamente. Por ello, nuestros clientes serán capaces de sacarle el máximo provecho a nuestra aplicación y sin la ayuda de nadie.

En definitiva, nuestra aplicación se distingue de las demás por su sencillez extrema y capacidad de personalización, siendo nuestro objetivo, no la creación de una aplicación supercompleta si no, accesible para cualquier instalación deportiva.

1.5. Objetivos

El objetivo principal es el desarrollo de una aplicación parametrizable para la gestión de espacios deportivos, que cuente con la funcionalidad descrita en el apartado *1.4* y sea fácilmente distribuible.

Para poder lograr nuestro objetivo, hemos fijado los siguientes subobjetivos:

- Realizar el análisis de un sistema que cumpla con los requerimientos marcados.
- Desarrollar una aplicación empresarial con toda la funcionalidad necesaria, tanto en el servidor como en el cliente. Para ello, realizaremos un estudio de los frameworks y metodologías existentes.
- Preparar contenedores con los elementos necesarios para su distribución.
- Testear y documentar todo el desarrollo realizado.

1.6. Estructura de la Memoria

En este apartado mostraremos una breve descripción de cada uno de los capítulos con los que cuenta nuestro TFG:

Comenzaremos con la Introducción en el capítulo 1, en el expondremos una visión general de nuestro proyecto con el contexto, breve descripción de la aplicación, etc.

En el capítulo 2 se realizará un estudio acerca de las metodologías y tecnologías existentes para decidir cuál es la mejor a utilizar. Seguidamente en el capítulo 3 se analizarán los actores y requerimientos del sistema. También realizaremos la planificación temporal del proyecto junto con su coste y, la descripción detallada de sus casos de uso.

En el capítulo 4 pasaremos al diseño, en este expondremos el diagrama de clases y se diseñará la interfaz de nuestra aplicación.

Una vez llegados a este punto, en el capítulo 5 realizaremos la implementación exponiendo los puntos más importantes de esta.

En el capítulo 6, las pruebas de nuestra aplicación y en el 7 las conclusiones.

Por último, incluimos la Bibliografía y apéndices con la documentación del API Restful, el Manual de Instalación y de Usuario.

2. ESTUDIO DE METODOLOGIAS Y TECNOLOGIAS

A la hora de comenzar un proyecto es primordial la planificación de este. Para ello, en este capítulo realizaremos un estudio acerca de las metodologías a seguir para el desarrollo software, entender lo que son las aplicaciones empresariales y el estudio de tecnologías para Base de Datos, Backend , Frontend, HTML y CSS.

2.1. Metodologías de proyectos de desarrollo software

Antes de decidir qué metodología usaremos para el desarrollo de nuestro proyecto, vamos a realizar un estudio de estas para ver sus puntos fuertes y débiles.

2.1.1. Metodología Ágil – Scrum

Scrum [6] es una metodología que pone en marcha los principios ágiles y procesos para mejorar la entrega. Este enfoque se recomienda para proyectos donde el objetivo final no se especifica en detalle y donde los requisitos cambian constantemente o están mal definidos, ya que es un proceso de diseño flexible e iterativo.

En Scrum los proyectos se ejecutan en periodos de tiempo coros y de duración fija, las entregas parciales y periódicas del producto final se determinan de acuerdo con los beneficios que aportan al destinatario del proyecto.

2.1.2. Metodología Waterfall – Cascada

La metodología en Cascada [6] adopta un enfoque muy simple, crea una planificación sólida del proyecto e implementa meticulosamente las diferentes fases del proyecto una vez, correctamente.

Un buen plan se desarrolla e implementa en una secuencia estricta junto con los requisitos definidos antes de que comience el proyecto. Cada etapa debe completarse antes de que comience la siguiente etapa. Por lo general, la salida de una etapa actúa como entrada para la siguiente etapa secuencialmente. Todo el proyecto se entrega en un único ciclo, por lo que no es posible mostrar el progreso del proyecto.

Una de sus desventajas es su limitada capacidad para incorporar nuevos conocimientos y adaptarse a los cambios. Una vez que se aprueba un plan, hay poco margen para adaptarlo a menos que sea absolutamente necesario.

En los casos en que la especificación está bien definida y se esperan pocos o ningún cambio, esta metodología es las más efectiva.

2.1.3. Metodología a usar

Por todo lo comentado anteriormente y teniendo en cuenta el desarrollo de nuestro proyecto, utilizaremos los principios y metodologías ágiles Scrum, ya que gracias a las iteraciones o Sprints, segmentaremos el proyecto en pequeños bloques mucho más gestionables que si tratáramos de abarcarlo entero de principio a fin. De esta manera podremos identificar fácilmente los objetivos de cada etapa e incluso los posibles contratiempos con los que podríamos encontrarnos durante el camino.

Además, al segmentar el objetivo a entregar, las fechas de entrega se ajustarán mucho más a lo planificado.

2.2. Enterprise Applications

Para nuestro proyecto, dotaremos a nuestra aplicación con una arquitectura empresarial [7] debido a que proporciona múltiples ventajas:

- Servidor (Back-end) separado de los clientes (Front-end)
- Servicios para múltiples tipos de clientes (web, móvil) o para otras aplicaciones empresariales
- Capacidad para atender peticiones procedentes de muchos usuarios simultáneos
- Capacidad para soportar gran carga de trabajo
- Mayor escalabilidad y modularidad gracias a su arquitectura.

Con la existencia de un servidor de aplicaciones o Back-end, implementa la inteligencia de negocio y exporta una serie de servicios al exterior, utilizables por cualquier tipo de cliente.

Su arquitectura consta de las siguientes capas:

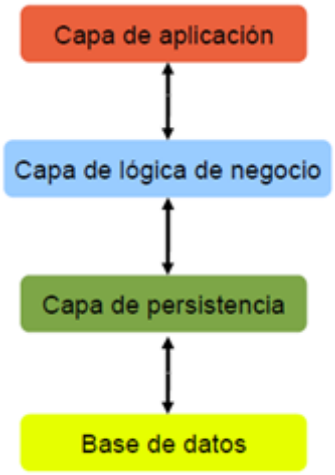


Ilustración 2.3 Arquitectura Enterprise Application

- **Capa de aplicación**: Es como se le conoce al Front-end.
- **Capa de lógica de negocio**: Implementa las reglas reales del negocio, cómo la información es creada, procesada y almacenada mediante una serie de objetos de negocio/dominio.
- **Capa de servicios**: Nueva capa para desacoplar la capa de aplicación. Hace accesible a las aplicaciones la funcionalidad de la lógica de negocio mediante protocolos universales y estandarizados. Implementa políticas de seguridad.
- **Capa de persistencia**: Simplifica el acceso a la base de datos eliminando las características particulares de cada SGBD, así como el uso de lenguajes de consulta como SQL.

Dichas aplicaciones deben incorporar funcionalidades como:

- Implementación de mecanismos de inyección de dependencias
- Acceso a base de datos mediante mapeado O/R
- Transacciones
- Operaciones asíncronas/programadas
- Seguridad con soporte de diferentes proveedores

2.3. Base de Datos

En java, los datos se representan en objetos. Sin embargo, las bases de datos almacenan sus datos en forma relacional. Obviamente existe una brecha entre “objetos-relacional”, para ello existen los frameworks ORM [8], (Object-Relational-Mapping).

Haremos uso de estos ya que así tendremos un código sencillo y legible, un mapeado automático de estructuras de objetos complejos y en un futuro, si necesitamos realizar la migración de un sistema gestor de base de datos a otro, no será complicado.

Java Persistence API (JPA)

Framework que ofrece un conjunto de interfaces y APIs para resolver el problema del almacenamiento de los objetos en una base de datos relacional. No es una implementación en sí misma, sino que proporciona interfaces para ser implementadas por diferentes proveedores. En el código usaremos el API de JPA que será implementado por la librería que más nos convenga.

Una tabla se mapea contra una clase y cada columna contra un atributo de dicha clase. Para ello, usaremos anotaciones de JPA para indicar estas relaciones y así conseguir ocultar la complejidad del acceso a datos, exponiendo únicamente objetos.

Hibernate

ORM bajo licencia GNU LGPL para Java que implementa JPA. Hibernate [9] nos permite detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen mediante anotaciones donde corresponde un atributo de una clase con una columna de una tabla. Con esta información, Hibernate permitirá a la aplicación manipular los datos en la base de datos operando sobre objetos, con todas las características de la POO. Convertirá los datos entre los tipos utilizados por Java y los definidos por SQL y generará las sentencias SQL.

2.4. Frameworks para Enterprise Application

Un framework [10] es un esquema o marco de trabajo que proporciona una estructura para desarrollar un proyecto con objetivos específicos, un tipo de plantilla que sirve como punto de partida para organizar y desarrollar software. Pero ¿por qué usar un Framework? Entre otras, algunas de las ventajas [11] con las que cuentan:

- Ahorra tiempo al disponer del esqueleto sobre el que se desarrollará la aplicación
- Mayor facilidad a la hora de encontrar herramientas, módulos e información para usar, gracias a estar ampliamente extendido
- Existencia de una comunidad, conjunto de desarrolladores que te pueden ayudar en tus consultas

Algunos de los Frameworks más conocidos y usados:

- .Net: framework de Microsoft
- Simphony: proyecto PHP de software libre
- Angular: framework de código abierto desarrollado en TypeScript
- Django: framework de código abierto escrito en Python

A continuación, hablaremos acerca del Back-end y Front-end y de sus respectivas tecnologías.

2.4.1. Back-end

Lo primero que debemos de tener claro es ¿qué es el **Back-end**?

Este término [12] es utilizado para referirse al área lógica de toda página web, es decir, a la arquitectura interna del sitio que modela el negocio/dominio y todos sus procesos internos. No está visible al usuario y no incluye ningún elemento de tipo gráfico.

Además, es responsable de la funcionalidad, seguridad y optimización de los recursos del sitio web.

En resumen, el Back-end es la parte o rama del desarrollo web que se encarga de que todas las lógicas de una aplicación web, móvil o incluso de escritorio tradicional funcione.

Algunas de las funciones que gestiona el Back-end:

- Conexiones con las bases de datos
- Seguridad de los sitios web
- Optimización de los recursos para que las páginas resulten más ligeras
- Uso de librerías del servidor web, ya sea para comprimir las imágenes de la web, implementar temas de caché u otras

Una vez explicado el concepto de Back-end vamos a pasar a explicar la tecnología que usaremos para el desarrollo de este.

Existe una gran variedad de opciones disponibles al elegir con qué framework de Back-end queremos trabajar. En nuestro caso habiendo estudiado previamente los pros y contras de alguno de ellos, hemos decidido trabajar con Spring Boot.

2.4.1.1. Spring Boot

Ilustración 2.4 Icono Spring

Apareció hace unos años con la idea de complementar a Spring [13] . Tal y como podemos ver en la siguiente ilustración, Spring Boot en realidad trabaja por debajo con Spring, pero de una manera más simple.

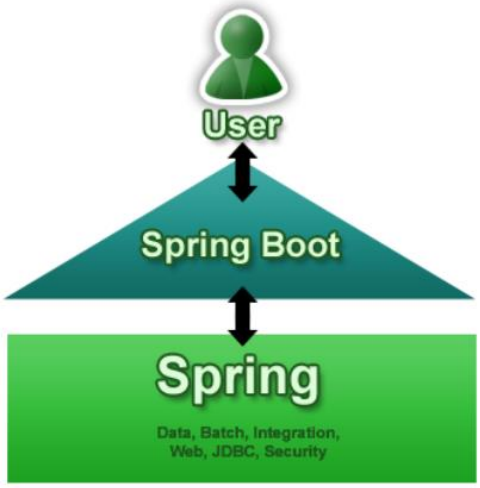


Ilustración 2.5 Spring Boot

Se puede definir como un acelerador para la creación de proyectos Spring, basado principalmente en el concepto de convención antes de configuración. Por lo tanto, su finalidad es crear proyectos de Spring, pero de una manera más ágil, mediante una serie de convenciones que prevalecen sobre la configuración, ahorrándonos el perder tiempo en realizar configuraciones en ficheros.

Debemos de destacar que Spring Boot no se considera Framework como tal, en cambio, Spring sí. Pero con ambos, podemos llegar a tener el mismo desarrollo, aunque la diferencia entre ambos sería el tiempo invertido.

El principio de convención sobre configuración mencionado anteriormente trata de especificar y detallar un conjunto de normas que estarán por defecto a excepción de que se detalle lo contrario. Un ejemplo de esto podría ser cuando creamos una aplicación con Spring, tendremos que realizar configuraciones como la del puerto. Sin embargo, con Spring Boot, gracias a este principio, el puerto por defecto será el 8080, pero si deseamos cambiarlo únicamente tendremos que modificar un archivo y añadir una instrucción donde indicaremos el puerto.

Todo esto trae una gran ventaja ya que nos permite centrarnos directamente en la lógica empresarial que requiere la aplicación, haciendo el proceso más corto, rápido y eficaz, ahorrando líneas de código al evitar tareas repetitivas.

Además de por lo que hemos comentado también lo hemos elegido ya cuenta con [14]:

- Contenedores Java embebidos: Tomcat o Jetty.
- Soporte para la automatización con Maven y Gradle.
- Configuración sugerida para iniciar rápidamente con un proyecto (Starters).
- Cuando sea posible, configura automáticamente Spring.
- Características listas para producción: seguridad, verificación del estatus, externalización de configuración, etc.
- No genera código y no requiere configuración XML.

Por todo esto, es el Framework perfecto para el desarrollo del Backend.

2.4.2. Front-end

El Front-end [12] es la parte que ve el usuario e incluye las líneas de diseño y los elementos gráficos de la web. Aquí será donde incluyamos los estilos, colores, fondos, tamaños, animaciones y más del sitio web.

Trabajaremos con otro tipo de lenguajes, pero más cercanos a la comprensión e interacción del usuario.

Debemos de tener en cuenta que la experiencia del usuario es la prioridad número uno para todas las empresas del mercado. No importa cuán complejas sean las funciones y operaciones en segundo plano, lo que los usuarios ven y experimentan tiene que ser perfecto.

En nuestro caso para el Front-end usaremos Angular y a continuación explicaremos cuáles son las razones.

2.4.2.1. Angular

Ilustración 2.6 Angular

Basado en TypeScript fue lanzado en 2016 [15] y desarrollado por Google para cerrar la brecha entre las crecientes demandas de tecnología y los conceptos convencionales que mostraron resultados.

La funcionalidad a destacar de Angular es su función de enlace de datos bidireccional, la cual nos permite una sincronización en tiempo real entre el modelo y la vista, donde cualquier cambio en el modelo se refleja instantáneamente en la vista y viceversa.

Algunas de las razones que nos han llevado a elegirlo son las siguientes:

- Funcionalidad incorporada para actualizar los cambios realizados en el modelo a la vista y viceversa
- Reduce la cantidad de código, ya que la mayoría de las características destacadas se proporcionan de forma predeterminada
- Desacopla los componentes de las dependencias definiéndolos como elementos externos
- Componentes reutilizables y fáciles de administrar mediante la inyección de dependencias
- Gran comunidad de aprendizaje y apoyo

Para hacer un buen uso de Angular necesitaremos:

Angular CLI

Command Line Interface [16] es una herramienta ofrecida por el propio equipo de Angular. Esta nos facilita el proceso de inicio de cualquier aplicación con Angular, ya que nos ofrece el esqueleto de archivos y carpetas que necesitaremos junto con una gran cantidad de herramientas ya configuradas. Durante el desarrollo nos ayudará generando el "scaffolding" consiguiendo una estructura apropiada para el proyecto. Durante la etapa de producción o testing también nos proporcionará la ayuda necesaria para preparar los archivos que deben ser subidos al servidor, etc.

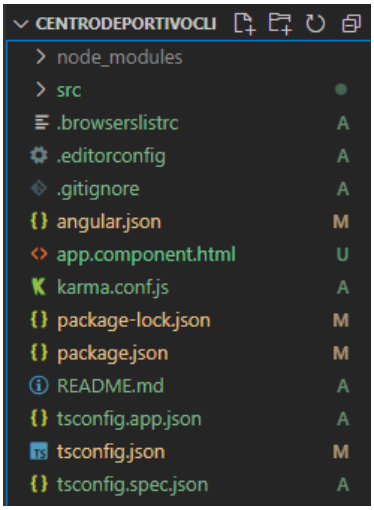


Ilustración 2.7 Scaffolding de nuestro proyecto

Node.js

Para poder instalar Angular CLI necesitaremos contar con Node.js [17] . Este es un entorno de ejecución basado en el lenguaje JavaScript orientado a eventos asíncronos. Está diseñado para crear aplicaciones en red escalables.

Uno de sus objetivos fundamentales es cargar el contenido dinámico de las páginas web antes de que estas se envíen al navegador del usuario. De esta forma se consigue una carga más eficaz y se agiliza la visualización.

Npm

Gestor de paquetes desarrollado [18] bajo el lenguaje JavaScript, gracias a este podemos obtener cualquier librería con tan solo una línea de código, lo cual nos permite agregar dependencias de forma simple, administrar módulos y el proyecto a desarrollar. Cabe mencionar que Node.js se instala automáticamente desde el entorno Npm.

Por todo lo comentado anteriormente, Angular es la mejor opción para aplicaciones empresariales.

2.4.3. HTML y CSS

Ilustración 2.8 Bootstrap

Para el diseño y maquetación de nuestro sitio web elegimos Bootstrap. Este es uno de los Frameworks más populares para desarrollar el Frontend de sitios web. Nos proporciona plantillas CSS y HTML facilitando el posicionamiento y el diseño de la página, las fuentes, los botones y elementos de navegación. De esta forma, implementar un diseño web moderno resulta muy sencillo.

Además de lo comentado anteriormente, algunas de sus ventajas [19] más importantes y por lo que haremos uso de él, son:

- **Fácil de utilizar:** permite crear grandes sitios web en muy poco tiempo debido a su facilidad de uso. Además, es muy intuitivo, lo que ayuda mucho a nuestro trabajo.
- **Responsive Design:** proporciona todas las reglas CSS para que nuestro sitio web se adapte dinámicamente a la mayoría de las pantallas.
- **Comunidad Web:** utiliza componentes y servicios creados por la comunidad. Por ejemplo: HTML5, CSS o Github, entre otros.
- **Extensible:** proporciona diversas herramientas para ampliar nuestro Framework, así como adaptarlo a nuestras necesidades y características.
- **Utiliza Less:** lenguaje de las hojas de estilo CSS que nos permite utilizar funciones, operaciones aritméticas o variables para enriquecer el estilo de nuestra web.
- **Documentación:** cuenta con una gran documentación que puede sacarte de apuros.

3. ANALISIS Y PLANIFICACION DEL PROYECTO

Según los motivos dados en el apartado *2.1.3* y teniendo en cuenta que para aplicar SCRUM hacen falta varios roles y un equipo de desarrollo, nuestra elección para la realización del proyecto será únicamente el uso sus principios y metodologías ya que nos parecen apropiados, pero evidentemente por lo comentado no podríamos desarrollarlo.

En este capítulo, realizaremos un estudio de las funcionalidades con las que debe contar nuestra aplicación y la planificación temporal de nuestro proyecto.

Comenzaremos definiendo los actores y requerimientos del sistema para más tarde realizar la planificación de las tareas previas, el product backlog de desarrollo y, por último, las tareas de testing.

3.1. Actores

Los actores que podrán interactuar con nuestro sistema son los siguientes:

- **Usuario no logueado:** este podrá acceder al sistema y visualizar las características y detalles del centro deportivo, contactar mediante un formulario de contacto y visualizar las instalaciones con sus características y horarios.
- **Usuario logueado:** accederán al sistema iniciando sesión y podrán reservar las pistas disponibles, modificar sus datos personales y ver las reservas realizadas junto con todas las funcionalidades de un usuario no logueado.
- **Usuario Administrador:** usuario con permisos especiales el cual se encargará de realizar la configuración de la aplicación y podrá gestionar los usuarios, instalaciones, horarios, días no seleccionables y reservas. Además, podrá hacer uso de toda la funcionalidad de un usuario normal para así realizar comprobaciones en el sistema.

3.2. Requerimientos del Sistema

Un requerimiento [20] es la condición o capacidad que debe tener un sistema, producto o servicio para satisfacer un contrato, especificación u otros documentos establecidos. Estos indican qué funciones y contenidos se espera que tenga tu producto, y como los usuarios deben interactuar con él.

Como sabemos [21] una de las razones por las que un sistema puede fallar es por una mala definición de requerimientos.

Una comprensión total de la gestión de requerimientos es esencial para el éxito de un desarrollo de software, no importa lo bien diseñado o codificado que este un programa si no se ha analizado correctamente.

En nuestro caso, vamos a diferenciar los requerimientos de nuestro sistema entre funcionales y no funcionales.

3.2.1. Requerimientos Funcionales

Cuando hablamos de la descripción de actividades y servicios que un sistema debe proveer nos estamos refiriendo a los requerimientos funcionales [22]. Estos son declaraciones sobre qué servicios debe proporcionar el sistema, cómo debe reaccionar el sistema ante ciertas entradas y cómo debe comportarse en ciertas situaciones. En algunos casos, los requisitos funcionales también pueden especificar lo que el sistema no puede hacer.

En cuanto a estos requerimientos, vamos a mostrar un acercamiento y más adelante detallaremos cada uno de ellos en forma de historias de usuario:

- El sistema permitirá únicamente la existencia de 2 perfiles diferentes, ADMIN y USER.
- El sistema dotará a la web con los siguientes elementos:
    - Calendario informativo para el usuario.
    - Ubicación Google Maps del recinto deportivo.
    - Widget de Previsión meteorológica.
    - Carrusel con las imágenes de las instalaciones.
    - Vídeo del centro deportivo

- El sistema permitirá al usuario USER:
  - Registrarse introduciendo sus datos personales.
  - Iniciar sesión introduciendo sus credenciales.

- El sistema permitirá al usuario USER visualizar sin necesidad de registrarse o iniciar sesión:
  - Las instalaciones.
  - Horario correspondiente a cada una de las pistas
  - Formulario de contacto.
  - Información acerca del centro deportivo.
- El sistema permitirá al usuario ADMIN visualizar un listado de las instalaciones creadas. Además, podrá crear, modificar y desactivarlas. Las pistas podrán ser desactivadas en el caso de necesitar mantenimiento, no pudiendo ser alquiladas por un tiempo.
- El sistema permitirá al usuario ADMIN visualizar un listado de los usuarios registrados en la aplicación pudiendo ser desactivados o activados. Se le dará esta opción al administrador por si en cualquier momento existe algún problema con el usuario y hay que desactivarlo para que no pueda acceder al sitio web.
- El sistema permitirá al usuario ADMIN introducir el horario para cada una de las pistas de forma automática indicando la hora de inicio y hora fin, generándose así diversas franjas horarias con una duración máxima de 1 hora. Además, podrá desactivar cada una de las franjas horarias debido a los cambios de temporalidad estacionales.
- El sistema permitirá al usuario ADMIN seleccionar los días los cuales no se podrán realizar reservas de las instalaciones.
- El sistema permitirá al usuario ADMIN visualizar un listado de todas las reservas realizadas y de aquellas pendientes de pago.
- El sistema permitirá al usuario USER una vez haya iniciado sesión en la aplicación web, realizar una reserva, visualizarlas, realizar el pago de estas y visualizar/modificar sus datos personales.
- Para la realización de la reserva el sistema deberá permitirle al usuario USER, seleccionar la pista que desee alquilar, fecha y hora.

- El sistema permitirá al usuario USER reservar únicamente una franja horaria por reserva, debido a que puede darse la posibilidad de que existan usuarios que quieran reservar multitud de horas una misma pista y por consecuente dejen a otros usuarios sin la opción de reservar dicha pista. Como ampliación en el futuro ya que queda fuera del ámbito de nuestro TFG, podríamos realizar un filtrado (sistema de detección de uso abusivo), de esta forma un usuario no podrá reservar, por ejemplo, más de 2 franjas horarias al día o 4 semanales, quitándonos así esta problemática.
- El sistema permitirá al usuario USER pagar la reserva eligiendo entre tarjeta bancaria, PayPal o Sofort.
- El sistema permitirá al usuario ADMIN cancelar una reserva en el caso de que el usuario no haga efectivo el pago de la reserva por cualquier error.
- El sistema permitirá hacer efectiva la reserva a un único componente del grupo, teniéndose que pagar la totalidad de esta.
- El sistema permitirá una vez realizada la reserva, que el usuario reciba un email de confirmación con los datos detallados de esta.
- La primera vez que se inicie la aplicación web, el sistema permitirá al usuario ADMIN rellenar toda la información acerca de esta, y de manera automática y sencilla obtener una aplicación web totalmente funcional.
- Para el apartado de la **parametrización**, el sistema deberá permitir parametrizar:
    - Título.
    - Descripción del Sitio Web.
    - Contacto:
        - Dirección
        - Correo Electrónico
        - Número de Teléfono
    - Introducción de las características de cada pista.
    - Selección de las fechas no disponibles para alquilar las instalaciones.
    - Horario de alquiler de cada pista.
    - Creación de un usuario Administrador.
    - Vídeo de la página principal.

3.2.2. Requerimientos No Funcionales

Por otro lado, los requerimientos no funcionales [22] describen otras características y restricciones que debe tener un sistema para tener éxito. Estos engloban características como el rendimiento, la facilidad de uso, el presupuesto, el tiempo de entrega, la documentación, la seguridad y auditorías internas. Por lo general, se aplican a todo el sistema.

Para nuestro proyecto contamos con los siguientes requerimientos no funcionales siendo algunos de ellos muy importantes [23]:

- El sitio web de la aplicación deberá poderse explotar y administrar empleando cualquier navegador web y cualquier dispositivo, ya sea Tablet, Smartphone u Ordenador. Seguirá la técnica “Responsive Web Design”, adaptando y optimizando los contenidos de esta al tamaño y resolución de pantalla permitiendo así su correcta visualización.
- Los datos de la aplicación deberán estar almacenados en un sistema gestor de bases de datos, sobre el cual podrán realizarse consultas.
- Los datos de la aplicación solo podrán ser modificados por aquellas personas autorizadas para ello, en nuestro caso el perfil “Admin”.
- La web deberá tener una estructura clara, ordenando el contenido y las funciones de la aplicación en apartados que abarquen todas las funcionalidades disponibles según el perfil del usuario conectado.
- Facilidad en la puesta a punto de la aplicación web por parte de un usuario sin amplios conocimientos informáticos.
- Facilidad de despliegue

3.3. Planificación Temporal

La planificación de nuestro proyecto estará dividida en tres etapas, la etapa inicial que cuenta con las tareas que debemos de realizar previamente, como es el caso del análisis, estudio, etc, etapa intermedia, donde haciendo uso de elementos SCRUM [24] se realizará el desarrollo de nuestra aplicación, y, por último, la etapa final con las tareas de evaluación y testing, donde pondremos a prueba nuestra aplicación y algunos usuarios finales rellenarán un cuestionario haciendo uso de la aplicación.

3.3.1. Tareas Previas

En un proyecto real, con un equipo formado por diferentes roles, estas tareas serían realizadas por el jefe de proyecto o analista.

Nombre de la Tarea	Días (3 horas)	Horas
Análisis del problema	5	15
Estudio de aplicaciones existentes	3	9
Definición de objetivos, propósito del proyecto, descripción de la aplicación y estructura de la memoria	7	21
Estudio de metodologías y tecnologías	3	9
Identificación de actores y requerimientos del sistema	5	15
Historias de usuario	2	6
Gestión de costes	2	6
Redacción de casos de uso	3	9
Diseño de diagrama de clases	3	9
Diseño de la interfaz	3	9
Total	**36**	**108**

Tabla 3.1 Duración Tareas Previas

Como podemos observar en la tabla anterior, la etapa inicial contará con un total de 36 días y 108 horas.

3.3.2. Product Backlog de Desarrollo

Para el desarrollo, el cual se debería de encargar el programador, se hará uso del Product Backlog. Este es el inventario donde todas las características o requisitos se almacenan como una lista jerárquica. Estos requisitos serán los que tendrá el producto o los que tendrá en iteraciones sucesivas. Estos se suelen expresar a través de **historias de usuario**.

La estimación de cada historia de usuario la realizaremos siguiendo la técnica de "**estimación por talla de camisetas**", la cual dependiendo de su dificultad le asociaremos unos puntos de historia u otros. La correspondencia de esta será la siguiente:

Talla	Puntos de Historia
XS	1
S	2
M	3
L	5
XL	8
XXL	13

Tabla 3.2 Correspondencia de tallas de camiseta con puntos de historia

Para nuestro proyecto tendremos el siguiente Product Backlog:

Id	Título	Tallas	Estimación
01	Registro de Usuario	XL	8
02	Login de Usuario	XL	8
03	Visualización de instalaciones	M	3
04	Visualización detalles pista	M	3
05	Visualización horario pista	M	3
06	Formulario de Contacto	L	5
07	Visualización Información Centro Deportivo	M	3
08	Visualización/modificación/desactivación pistas	XL	8
09	Visualización/desactivación Usuarios registrados	XL	8
10	Activación/desactivación de horarios	L	5
11	Visualización fechas no disponibles	M	3
12	Listado Reservas	M	3
13	Cancelación/Visualización de Reservas Pendientes	XL	8
14	Realización de Reserva	XXL	13
15	Listado mis Reservas	L	5
16	Visualización/modificación datos personales	L	5
17	Selección Método de Pago	XL	8
18	Confirmación de Reserva	XL	8
19	Ubicación Google Maps	M	3
20	Previsión Meteorológica	M	3
21	Calendario consulta Fechas	XL	8
22	Navegadores Web y Dispositivos	L	5
23	Interfaz Intuitiva	L	5
24	Seguridad Sitio Web	XXL	13

25	Sistema Gestor de Base de Datos	XXL	13
26	Facilidad Puesta a punto sitio web	XXL	13
27	Facilidad de despliegue	XXL	13

Tabla 3.3 Product Backlog

Para el apartado de la **parametrización** también tendremos las siguientes historias de usuario en nuestro Product Backlog:

Id	Título	Tallas	Estimación
28	Información Centro Deportivo	L	5
29	Selección fechas no disponibles	L	5
30	Creación Instalaciones	L	5
31	Creación Horarios	L	5
32	Creación usuario Administrador	L	5

Tabla 3.4 Product Backlog Parametrización

La estimación que obtenemos de nuestro Product Backlog es de **208 puntos de historia** en total.

3.3.2.1. Historias de Usuario

Una vez realizado el Product Backlog, pasaremos a especificar cada una de las historias de usuario.

Las historias de usuario [24] son descripciones de las funcionalidades que tendrá nuestro producto. Estas evolucionarán a lo largo de la vida del proyecto.

Se componen de tres fases conocidas como “Las 3 C”:

- Card: breve descripción escrita para usar como recordatorio.
- Conversation: conversación que servirá para asegurar que todo se entienda, y concretar el objetivo.
- Confirmation: tests funcionales para fijar detalles relevantes e indicar cuál va a ser el límite.

Cada historia de usuario está compuesta de:

- Id
- Título
- Descripción
- Estimación, Puntos de Historia

- Criterios de Aceptación

Para nuestro producto tendremos las siguientes historias de usuario.

Como **Usuario**:

Id	01
Título	**Registro de Usuario**
Descripción	Como usuario quiero poder registrarme en la aplicación para poder disfrutar de todas sus funcionalidades
Puntos de Historia	8
Criterios de Aceptación	• El usuario se dirigirá a la pestaña “Acceder” y seleccionará “Registrarse” • Introducirá sus datos personales como: nombre, apellidos, teléfono, email, etc • Los datos introducidos deben ser correctos en cuanto a las restricciones, si no, aparecerá un mensaje de error y no se hará efectivo el registro • Una vez registrado correctamente el sistema le llevará al Login para iniciar sesión

Tabla 3.5 HU Registro de Usuario

Id	02
Título	**Login de Usuario**
Descripción	Como usuario registrado quiero poder iniciar sesión en la aplicación para poder hacer uso de todas las funcionalidades
Puntos de Historia	8
Criterios de Aceptación	• El usuario se dirigirá a la pestaña “Acceder” y seleccionará “Iniciar Sesión” • El usuario introducirá sus credenciales de usuario • Las credenciales del usuario deben ser correctas, en caso contrario, el sistema mostrará un mensaje de error indicando el campo incorrecto • Una vez logueado correctamente, el sistema le llevará a su página de perfil

Tabla 3.6 HU Login de Usuario

Id	03
Título	**Visualización de instalaciones**
Descripción	Como usuario quiero poder visualizar el listado de instalaciones para asegurarme de querer realizar una reserva de estas
Puntos de Historia	3
Criterios de Aceptación	• El usuario se dirigirá a la sección “Instalaciones” • El usuario podrá visualizar todas las instalaciones de las que dispone el centro deportivo

Tabla 3.7 HU Visualización de instalaciones

Id	04
Título	**Visualización detalles Pista**
Descripción	Como usuario quiero poder visualizar las características de una pista para decantarme por cuál reservar
Puntos de Historia	3
Criterios de Aceptación	• El usuario se dirigirá a la sección “Instalaciones” • El usuario seleccionará el botón “ver” de la pista en cuestión

Tabla 3.8 HU Visualización detalles Pista

Id	05
Título	**Visualización Horario Pista**
Descripción	Como usuario quiero poder visualizar el horario de una pista para saber si puedo reservarla
Puntos de Historia	3
Criterios de Aceptación	• El usuario se dirigirá a la sección “Instalaciones” • El usuario seleccionará para la pista en cuestión el botón “Horario” y así poder visualizarlo

Tabla 3.9 HU Visualización Horario Pista

Id	06
Título	**Formulario de Contacto**
Descripción	Como usuario quiero disponer de un formulario de contacto para poder resolver mis dudas
Puntos de Historia	5
Criterios de Aceptación	• El usuario se dirigirá a la sección “Contacto” • El usuario rellenará dicho formulario • Los datos introducidos deben ser correctos en cuanto a las restricciones, si no, aparecerá un mensaje de error indicando el campo incorrecto y no se permitirá enviar la consulta

Tabla 3.10 HU Formulario de Contacto

Id	07
Título	**Visualización Información Centro Deportivo**
Descripción	Como usuario quiero poder visualizar la información del recinto para así conocer sus datos
Puntos de Historia	3
Criterios de Aceptación	• El usuario se dirigirá a la sección “Contacto” • El usuario podrá visualizar todos los datos del centro deportivo

Tabla 3.11 HU Información Centro Deportivo

Id	11
Título	**Visualización fechas no disponibles**
Descripción	Como usuario quiero visualizar las fechas en las que no se pueden reservar las pistas para una correcta reserva
Puntos de Historia	3
Criterios de Aceptación	• El usuario se dirigirá a la sección “Perfil” o seleccionará el botón “Reservar” de la página principal • El usuario podrá visualizar las fechas en el caso de que existan fechas no disponibles

Tabla 3.12 HU Visualización Fechas No Disponibles

Id	14
Título	**Realización de Reserva**
Descripción	Como usuario logueado quiero poder realizar una reserva para alquilar una pista
Puntos de Historia	13
Criterios de Aceptación	• El usuario se dirigirá a la sección “Perfil” o seleccionará el botón “Reserva” de la página principal • El usuario introducirá los datos necesarios, si alguno de ellos no es correcto se le mostrará un mensaje de error indicándolo, en el caso de que todos lo sean, la reserva se hará efectiva

Tabla 3.13 HU Realización de Reserva

Id	15
Título	**Listado mis Reservas**
Descripción	Como usuario logueado quiero poder disponer de un listado de mis reservas realizadas para consultarlo
Puntos de Historia	5
Criterios de Aceptación	• El usuario se dirigirá a la sección “Perfil” • El usuario seleccionará el botón “Mis Reservas” donde podrá visualizarlas siempre y cuando, haya realizado alguna anteriormente

Tabla 3.14 HU Listado mis Reservas

Id	16
Título	**Visualización/Modificación Datos Personales**
Descripción	Como usuario logueado quiero poder visualizar y modificar mis datos personales
Puntos de Historia	5
Criterios de Aceptación	• El usuario se dirigirá a la sección “Perfil” • En dicha sección podrá visualizar sus datos personales • Si desea modificarlos deberá seleccionar el botón “Modifica tus Datos” • Si los nuevos datos introducidos no cumplen con las restricciones se le mostrará un mensaje de error indicando el campo incorrecto, en caso contrario, los datos serán modificados correctamente

Tabla 3.15 HU Visualización/Modificación Datos Personales

Id	17
Título	**Selección Método de Pago**
Descripción	Como usuario quiero seleccionar un método de pago para poder hacer efectiva la reserva
Puntos de Historia	8
Criterios de Aceptación	• Una vez el usuario haya seleccionado los datos necesarios para realizar la reserva, este, podrá seleccionar el método de pago que desee • En caso de que al realizar el pago de la reserva se produzca un error, el usuario deberá dirigirse a la sección “Perfil”, “Mis Reservas” y seleccionar el pago deseado para hacer efectiva la reserva en cuestión

Tabla 3.16 HU Selección Método de Pago

Id	18
Título	**Confirmación de Reserva**
Descripción	Como usuario quiero recibir un correo electrónico para confirmar la reserva realizada y comprobar que los datos son correctos
Puntos de Historia	8
Criterios de Aceptación	• Al realizar una reserva el usuario correspondiente recibirá un mensaje a su correo electrónico con la confirmación y todos los datos acerca de esta

Tabla 3.17 HU Confirmación de Reserva

Id	19
Título	**Ubicación Google Maps**
Descripción	Como usuario quiero consultar Google Maps para conocer la ubicación del centro deportivo
Puntos de Historia	3
Criterios de Aceptación	• El usuario accederá a la sección “Contacto” y así podrá consultar a través de Google Maps la ubicación de las instalaciones del recinto deportivo

Tabla 3.18 HU Ubicación Google Maps

Id	20
Título	**Previsión Meteorológica**
Descripción	Como usuario quiero consultar la previsión del tiempo para saber si puedo desarrollar el deporte que deseo
Puntos de Historia	3
Criterios de Aceptación	• El usuario accederá a la página de Inicio y podrá consultar la previsión del tiempo a través de un widget

Tabla 3.19 HU Previsión Meteorológica

Id	21
Título	**Calendario consulta Fechas**
Descripción	Como usuario quiero poder visualizar los días que se pueden alquilar las instalaciones del recinto para realizar una reserva
Puntos de Historia	8
Criterios de Aceptación	• El usuario se dirigirá a la página principal y podrá visualizar un calendario con las fechas disponibles

Tabla 3.20 HU Calendario Fechas

Id	22
Título	**Navegadores Web y Dispositivos**
Descripción	Como usuario o administrador quiero acceder a la aplicación desde cualquier navegador web y dispositivo para hacer uso de sus funcionalidades
Puntos de Historia	5
Criterios de Aceptación	• Seguirán la técnica “Responsive Web Design”

Tabla 3.21 HU Navegadores Web y Dispositivos

Id	23
Título	**Interfaz Intuitiva**
Descripción	Como usuario quiero visualizar una interfaz intuitiva para poder optimizar el tiempo empleado en la web
Puntos de Historia	5
Criterios de Aceptación	• La web contará con una estructura claramente definida

Tabla 3.22 HU Interfaz Intuitiva

Como **Administrador**:

Id	08
Título	**Visualización/Modificación/Desactivación Pistas**
Descripción	Como administrador quiero poder visualizar/modificar y desactivar las pistas del recinto para su correcta visualización
Puntos de Historia	8
Criterios de Aceptación	• El administrador se dirigirá a la sección “Admin” • Seleccionará el botón “Instalaciones” • El administrador podrá visualizar un listado con las pistas creadas • Si desea modificar los datos de una pista, deberá seleccionar el botón “Editar” de la pista deseada • Si desea desactivar una pista deberá seleccionar el botón “Editar” y marcar la casilla “desactivar”

Tabla 3.23 HU Visualización/Modificación/Desactivación Pistas

Id	09
Título	**Visualización/Desactivación Usuarios Registrados**
Descripción	Como administrador quiero poder disponer de un listado de usuarios registrados y de una opción para poder activarlos o desactivarlos para poder gestionarlos
Puntos de Historia	8
Criterios de Aceptación	• El administrador se dirigirá a la sección “Admin” • Seleccionará el botón “Usuarios” donde podrá visualizar el listado de los usuarios registrados junto con sus datos • Si el administrador desea activar/desactivar a dicho usuario únicamente tendrá que seleccionar el botón “Activar” o “Desactivar”

Tabla 3.24 HU Visualización/Desactivación Usuarios Registrados

Id	10
Título	**Activación/Desactivación de Horarios**
Descripción	Como administrador quiero poder activar o desactivar las franjas horarias correspondientes a cada pista para poder ofrecerlas correctamente
Puntos de Historia	5
Criterios de Aceptación	• El administrador se dirigirá a la sección “Admin” • Seleccionará el botón “Franjas Horarias” • El administrador desactivará o activará una franja horaria seleccionando el botón en cuestión

Tabla 3.25 Activación/Desactivación de Horarios

Id	12
Título	**Listado Reservas**
Descripción	Como administrador quiero poder visualizar un listado completo de las reservas para consultarlo
Puntos de Historia	3
Criterios de Aceptación	• El administrador se dirigirá a la sección “Admin” • Seleccionará el botón “Reservas” donde podrá visualizar el listado completo de Reservas realizadas

Tabla 3.26 HU Listado Reservas

Id	13
Título	**Cancelación/Visualización de Reservas Pendientes**
Descripción	Como administrador quiero poder visualizar un listado completo de las reservas pendientes para consultarlo y poder cancelarlas
Puntos de Historia	8
Criterios de Aceptación	• El administrador se dirigirá a la sección “Admin” • Seleccionará el botón “Reservas” • Seleccionará el botón “Reservas Pendientes” donde podrá visualizarlas y cancelarlas

Tabla 3.27 HU Listado de Reservas Pendientes

Id	24
Título	**Seguridad Sitio Web**
Descripción	Como administrador quiero denegar el acceso a usuarios cuando intenten acceder a urls o pantallas no permitidas para tener una web segura
Puntos de Historia	13
Criterios de Aceptación	• Cuando un usuario intente acceder a ciertas urls o pantallas en las que no tenga autorización se le mostrará un mensaje indicándolo

Tabla 3.28 HU Seguridad Sitio Web

Id	25
Título	**Sistema Gestor de Base de Datos**
Descripción	Como administrador quiero tener una base de datos para poder realizar consultas
Puntos de Historia	13
Criterios de Aceptación	

Tabla 3.29 HU Sistema Gestor de Base de Datos

Id	26
Título	**Facilidad Puesta a Punto Sitio Web**
Descripción	Como administrador quiero crear y personalizar el sitio web de forma sencilla e intuitiva para su correcto desarrollo
Puntos de Historia	13
Criterios de Aceptación	• El administrador deberá rellenar los datos de ciertos formularios para crear la aplicación web de manera sencilla y obtener esta totalmente personalizada.

Tabla 3.30 HU Facilidad Puesta a Punto Sitio Web

Id	27
Título	**Facilidad de Despliegue**
Descripción	Como administrador quiero contar con una aplicación fácil de desplegar para poder hacer uso de sus funcionalidades
Puntos de Historia	13
Criterios de Aceptación	

Tabla 3.31 HU Facilidad de Despliegue

En cuanto a la **parametrización** de la aplicación web, tenemos diferentes historias de usuario de las cuales su finalidad es dicha parametrización y el actor principal el administrador:

Debemos de tener en cuenta que, para dicha parametrización, el administrador únicamente tendrá que ir rellenando los formularios que se le presenten con los datos necesarios e ir seleccionando el botón “Siguiente”.

Id	28
Título	**Información Centro Deportivo**
Descripción	Como administrador quiero poder incluir toda la información acerca del recinto para su correcta visualización
Puntos de Historia	5
Criterios de Aceptación	• Al usuario administrador se le presentará un formulario para introducir los datos • Deberá introducir las características del recinto junto con la información de contacto • Si los datos introducidos no son correctos se mostrará un mensaje de error indicándolo

Tabla 3.32 HU Parametrización Información Centro Deportivo

Id	29
Título	**Selección Fechas No Disponibles**
Descripción	Como administrador quiero seleccionar las fechas no disponibles de las pistas para la correcta realización de las reservas
Puntos de Historia	5
Criterios de Aceptación	• Al usuario se le presentará un calendario donde podrá seleccionar las fechas para su correcta visualización

Tabla 3.33 HU Parametrización Selección Fechas No Disponibles

Id	30
Título	**Creación Instalaciones**
Descripción	Como administrador quiero añadir las pistas que componen el recinto para su correcta visualización
Puntos de Historia	5
Criterios de Aceptación	• Al usuario administrador se le presentará un formulario donde podrá introducir los datos necesarios para las pistas • Si los datos introducidos no son correctos se mostrará un mensaje de error indicándolo, en caso contrario se hará efectiva la creación

Tabla 3.34 HU Parametrización Creación Instalaciones

Id	31
Título	**Creación Horarios**
Descripción	Como administrador quiero crear los horarios de cada pista para su correcta visualización
Puntos de Historia	5
Criterios de Aceptación	• Al administrador se le presentará un formulario donde introducirá los datos necesarios • Si el administrador desea seguir creando pistas deberá seleccionar el botón “Crear otra pista”, en caso contrario el botón “Confirmar”

Tabla 3.35 HU Parametrización Creación Horarios

Id	32
Título	**Creación Usuario Administrador**
Descripción	Como administrador quiero crear un usuario con el rol “Admin” para el mantenimiento correcto de las funcionalidades de la web
Puntos de Historia	5
Criterios de Aceptación	• Al administrador se le presentará un formulario donde podrá introducir los datos necesarios para la creación del usuario • Si los datos introducidos no son correctos, se mostrará un mensaje de error indicándolo, en caso contrario, se hará efectivo

Tabla 3.36 HU Parametrización Creación Usuario Administrador

A continuación, vamos a presentar la estimación del Backlog y Spring Backlog.

3.3.2.2. Estimaciones del Backlog

Antes de comenzar con el Sprint Backlog, debemos de saber en cuantos Sprints vamos a dividir la realización de nuestro Product Backlog y cuántas historias de usuario vamos a incluir en cada Sprint.

Para poder realizar las estimaciones, primeramente, le hemos asociado a cada historia sus correspondientes puntos de historia según su dificultad a la hora de realizarla, por lo tanto, sumándolos obtenemos un total de **208 puntos de historia**.

Si suponemos que realizamos **30 puntos de historia** por cada Sprint, realizando la siguiente operación, 208 ÷ 30 obtenemos como resultado que nuestro proyecto lo dividiremos en 6,93 **≈ 7 Sprints**.

Lo siguiente que queremos calcular son los días de trabajo que tardaremos en realizar cada historia de usuario según la estimación que le hemos dado a cada una. Al realizar esto obtendremos una estimación de los días que tardaremos en tener el proyecto con todas las historias de usuario acabadas. Para ello haremos la siguiente operación:

XXL (13) →13 × 5 (días de una semana sin contar fines de semana) ÷ 30

=2,16 ≈ 2 días × 5 (historias de usuario que tenemos) = **10,83 días**

30puntos → 5 días

X →Y días

XL (8)-> (8×5) ÷ 30 = 1,3 × 8 = **10,6 días**

L (5)-> (5×5) ÷ 30 = 0,83 × 11 = **9,16 días**

M (3)-> (3×5) ÷ 30 = 0,5 × 8 = **4 días**

34,66 días ÷ 7 Sprints = 4,9523 **≈ 5 días**

De esta forma sabemos que cada Sprint durará 5 días y como tenemos 7 Sprints, el desarrollo de nuestro Product Backlog siempre y cuando no nos encontremos con retrasos, será de 35 días.

	Puntos de Historia	Días (3 horas)	Horas
Total	208	**35**	**105**

Tabla 3.37 Duración Product Backlog Desarrollo

En el siguiente apartado decidiremos qué historias de usuario incluiremos en cada uno de los Sprints.

3.3.2.3. Sprint Backlog

Tal y como sabemos el Sprint Backlog [25] es el conjunto de tareas seleccionadas del Product Backlog para el sprint actual, es decir, las tareas necesarias para realizar un incremento de producto.

El incremento es la suma de todos los elementos del Product Backlog completados durante un Sprint más el de todos los Sprints anteriores.

Es fundamental seleccionar las tareas y determinar su tamaño para así comprometernos en la finalización del Sprint.

A continuación, incluiremos en cada Sprint las historias de usuario que vamos a realizar en cada uno de estos:

Sprint	Historias de Usuario	Puntos de Historia
1	HU25(13), HU01(8), HU09(8)	29
2	HU16(5), HU19(3), HU20(3), HU30(5), HU08(8), HU04(3), HU03(3)	30
3	HU31(5), HU05(3), HU10(5), HU23(5), HU29(5), HU11(3), HU06(5)	31
4	HU17(8), HU28(5), HU22(5), HU32(5), HU02(8)	31
5	HU14(13), HU18(8), HU21(8)	29
6	HU13(8), HU15(5), HU12(3), HU24(13)	29
7	HU07(3), H26(13), H27(13)	29

Tabla 3.38 División de Sprints

3.3.3. Tareas de Validación, Testing

Igual que hemos mencionado en el apartado *3.3*, estas tareas serían realizadas por el equipo encargado del testing y validación de la aplicación.

Nombre de la Tarea	Días (3 horas)	Horas
Documentación de la API	2	6
Evaluación de pruebas	7	21
Pruebas con usuarios finales	6	18
Finalización de la documentación	5	15
Total	**20**	**60**

Tabla 3.39 Duración Tareas Testing

Esta etapa realizarla nos llevará un total de 20 días y 60 horas.

3.3.4. Diagrama de Gantt

En este apartado vamos a mostrar el Diagrama de Gantt realizado. Este contendrá las tres etapas anteriormente mencionadas, tareas previas, de desarrollo y de testing para así obtener la planificación completa del proyecto. En este, podremos observar la distribución que se ha hecho de las tareas con la fecha de inicio y fin de cada una de ellas.

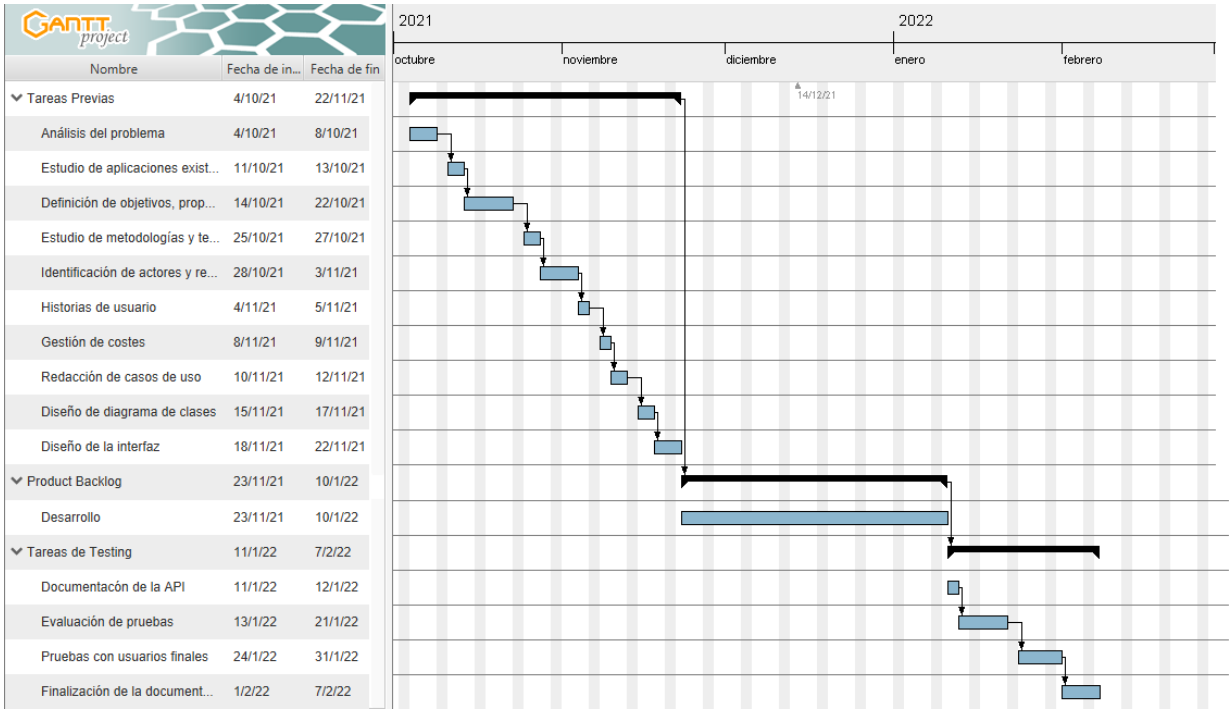


Ilustración 3.9 Diagrama de Gantt

Llegados a este punto estamos ya en condiciones de poder realizar una estimación de costes.

3.4. Coste del Proyecto

A continuación, realizaremos un estudio del coste de nuestro proyecto. Debemos de tener en cuenta que el equipo de desarrollo estará formado por un único miembro y que la duración estimada de dicho proyecto estará alrededor de 4 meses, ya que pueden aparecer imprevistos que retrasen y cambien nuestra planificación.

Gastos de Personal:

El presupuesto del analista programador para este proyecto tomando como referencia los datos consultados en el BOE [26] asciende a:

Analista Programador: 1.248,60€ al mes.

Meses de trabajo: 4 meses.

Sueldo al mes	**2.376,84€**
Total	**4.994,39€**

Tabla 3.40 Coste Gastos de Personal

Equipamiento Informático

A la hora de calcular el coste del equipamiento informático utilizado para el desarrollo de nuestro proyecto hemos tenido en cuenta: la vida útil de estos que normalmente es de 5 años y la duración de nuestro proyecto que estimamos en 4 meses, por lo tanto:

El equipamiento necesario para la realización de nuestro proyecto es el siguiente:

Dispositivo	Coste Inicial	Coste Asociado
Portátil ASUS X550LD Intel Core i7 4510U Almacenamiento 500 RAM 12GB Pantalla 15.6”	590,00€	39,33€
L-Link LL-KB-816-COMBO Kit Teclado + Ratón USB	5,35€	0,35666€
Tablet Samsung Galaxy Tab A7 Lite 32 GB WIFI Gris	148,99€	9,9326€
	Total	**49,61926€**

Tabla 3.41 Coste Equipamiento Informático

Gastos Inmateriales

Componente	Coste
Internet Jazztel 100Mbps	50€/mes
Total	**200€**

Tabla 3.42 Coste Gastos Inmateriales

Software

Como podemos ver en la siguiente tabla, únicamente debemos de hacer frente al coste de Microsoft Office ya que la mayoría de software utilizado ha sido gratuito gracias a las licencias que la Universidad de Jaén nos ha proporcionado y o licencias educativas como es el caso de IntelliJ IDEA. También contamos con programas con versiones gratuitas como Visual Studio Code, MySQL WorkBench o Postman y, por último, Gitlab, software de código abierto.

Software	Coste
IntelliJ IDEA Ultimate 2020.2	0,00€
Visual Studio Code 1.63.2	0,00€
MySQL Workbench 8.0	0,00€
Postman 9.0.9	0,00€
Visual Paradigm 16.0	0,00€
Gitlab Enterprise Edition 14.8.0-pre	0,00€
Microsoft Office Profesional Plus2016	69,95€
Total	**69,95€**

Tabla 3.43 Coste Software

Para finalizar este apartado mostraremos en *Tabla 3.44* un resumen de los costes totales.

Componente	Coste
Gastos de Personal	4.994,39€
Equipamiento Informático	49,61926€
Gastos Inmateriales	200€
Software	69,95€
Total	**5.313,95€**

Tabla 3.44 Coste Total

El último apartado de este capítulo trata de la descripción de cada uno de los casos de uso con los que cuenta nuestro proyecto para así tener una idea totalmente clara de la funcionalidad de nuestra aplicación.

3.5. Casos de Uso

Los Casos de Uso [27] son una lista de los pasos que realiza el actor para interactuar con el sistema. Esta es una lista de opciones básicas donde se muestran los pasos dados en el escenario principal, además tienen otra lista de alternativas, donde aparecen los errores o excepciones durante la ejecución del caso de uso.

Por todo ello, el usar casos de uso tiene un objetivo principal que es el de representar los distintos escenarios que pueden darse.

Estos son muy importantes para el análisis de sistemas por las siguientes razones:

- Describen el comportamiento de un sistema desde el punto de vista del usuario mediante reacciones y acciones.
- Establecen los límites del sistema
- Estos son descripciones de la funcionalidad del sistema
- Basados en lenguaje natural

Un caso de uso está formado por los siguientes elementos:

- Nombre
- Actor Primario
- El sistema al que pertenece el caso de uso
- Nivel del caso de uso: objeto usuario o subfunción
- Condiciones previas
- Operaciones básicas del escenario principal
- Alternativas

En cuanto a los casos de uso con los que contamos son los siguientes:

Caso de Uso	Registro de Usuario
Actor Primario	Usuario
Sistema	Sistema del Centro Deportivo
Nivel	Objetivo usuario
Condición Previa	El usuario no tenga ya una cuenta creada en la aplicación web
Operaciones Básicas	
1	El usuario se dirige a la pestaña Acceder
2	El usuario selecciona registrarse
3	El usuario introduce sus datos personales
4	Comprobación datos personales
5	Redireccionamiento a Login
Alternativas	
4.A	¿Son correctos los datos introducidos?
4.A.1	Si sí, continuar
4.A.2	Si no, mostrar mensaje de error y volver a 3

Tabla 3.45 Caso de Uso Registro de Usuario

Caso de Uso	Login de Usuario
Actor Primario	Usuario
Sistema	Sistema del Centro Deportivo
Nivel	Objetivo usuario
Condición Previa	El usuario tenga ya una cuenta creada en la aplicación web
Operaciones Básicas	
1	El usuario se dirige a la pestaña Acceder
2	El usuario selecciona Iniciar Sesión
3	El usuario introduce sus credenciales
4	Comprobación datos personales
5	Redireccionamiento a Página de Perfil
Alternativas	
4.A	¿Son correctos los datos introducidos?
4.A.1	Si sí, continuar
4.A.2	Si no, mostrar mensaje de error y volver a 3

Tabla 3.46 Caso de Uso Login de Usuario

Caso de Uso	Visualización de Instalaciones
Actor Primario	Usuario
Sistema	Sistema del Centro Deportivo
Nivel	Objetivo usuario
Condición Previa	Ninguna
Operaciones Básicas	
1	El usuario se dirige a la sección Instalaciones
2	Visualiza las pistas con las que cuenta el centro deportivo
Alternativas	Ninguna

Tabla 3.47 Caso de Uso Visualización de Instalaciones

Caso de Uso	Visualización Detalles Pista
Actor Primario	Usuario
Sistema	Sistema del Centro Deportivo
Nivel	Objetivo Usuario
Condición Previa	Ninguna
Operaciones Básicas	
1	El usuario se dirige a la sección Instalaciones
2	Selecciona el botón “ver” de la pista deseada
Alternativas	Ninguna

Tabla 3.48 Caso de Uso Visualización Detalles Pista

Caso de Uso	Visualización Horario Pista
Actor Primario	Usuario
Sistema	Sistema del Centro Deportivo
Nivel	Objetivo Usuario
Condición Previa	Ninguna
Operaciones Básicas	
1	El usuario se dirige a la sección Instalaciones
2	Selecciona el botón “Horario” de la pista deseada
Alternativas	Ninguna

Tabla 3.49 Caso de Uso Visualización Horario Pista

Caso de Uso	Formulario de Contacto
Actor Primario	Usuario
Sistema	Sistema del Centro Deportivo
Nivel	Objetivo Usuario
Condición Previa	Ninguna
Operaciones Básicas	
1	El usuario se dirige a la sección Contacto
2	El usuario rellena el formulario de contacto con los datos necesarios
3	El usuario selecciona “Enviar”
Alternativas	
2.A	¿Son correctos los datos introducidos?
2.A.1	Si sí, mostrar mensaje de confirmación y continuar
2.A.2	Si no, mostrar mensaje de error y volver a 2

Tabla 3.50 Caso de Uso Formulario de Contacto

Caso de Uso	Visualización Información Centro Deportivo
Actor Primario	Usuario
Sistema	Sistema del Centro Deportivo
Nivel	Objetivo Usuario
Condición Previa	Ninguna
Operaciones Básicas	
1	El usuario se dirige a la sección Contacto
2	El usuario visualiza la información de Contacto del centro deportivo
Alternativas	Ninguna

Tabla 3.51 Caso de Uso Información Centro Deportivo

Caso de Uso	Creación/Visualización/Modificación/ Desactivación Pistas
Actor Primario	Administrador
Sistema	Sistema del Centro Deportivo
Nivel	Objetivo Usuario
Condición Previa	Haber iniciado sesión como Administrador
Operaciones Básicas	
1	El administrador se dirige a la pestaña Admin y selecciona Instalaciones
2	Selecciona la opción deseada
2.A	Visualiza listado de pistas
2.B	Modifica la pista deseada
2.B.1	Introduce los datos necesarios
2.C	Activa/Desactiva la pista deseada
2.D	Crea una nueva pista
2.D.1	Introduce los datos necesarios
3	Finalizar
Alternativas	
2.B.1.A	¿Los datos introducidos son correctos?
2.B.1.A.1	Si sí, continuar
2.B.1.A.2	Si no, volver a 2.B.1
2.D.1.A	¿Los datos introducidos son correctos?
2.D.1.A.1	Si sí, continuar
2.D.1.A.2	Si no, volver a 2.D.1

Tabla 3.52 Caso de Uso Visualización/Modificación/Desactivación Pistas

Caso de Uso	Selección Fechas No Seleccionables
Actor Primario	Administrador
Sistema	Sistema del Centro Deportivo
Nivel	Objetivo Usuario
Condición Previa	Haber iniciado sesión como Administrador
Operaciones Básicas	
1	El administrador se dirige a la sección Admin
2	Selecciona Fechas No Seleccionables
3	Selecciona las fechas deseadas a través del calendario
Alternativas	Ninguna

Tabla 3.53 Caso de Uso Selección Fechas No Seleccionables

Caso de Uso	Visualización/Desactivación Usuarios Registrados
Actor Primario	Administrador
Sistema	Sistema del Centro Deportivo
Nivel	Objetivo Usuario
Condición Previa	Haber iniciado sesión como Administrador
Operaciones Básicas	
1	El administrador se dirige a la sección Admin
2	Selecciona el botón “Usuarios”
3	Selecciona la opción deseada
3.A	Visualiza el listado de usuarios registrados
3.B	Activa/Desactiva el usuario deseado
Alternativas	Ninguna

Tabla 3.54 Caso de Uso Visualización/Desactivación Usuarios Registrados

Caso de Uso	Creación/Activación/Desactivación Horarios
Actor Primario	Administrador
Sistema	Sistema del Centro Deportivo
Nivel	Objetivo Usuario
Condición Previa	Haber iniciado sesión como Administrador
Operaciones Básicas	
1	El administrador se dirige a la sección Admin
2	Selecciona el botón “Franjas Horarias”
3	Selecciona la opción deseada
3.A	Crea una nueva franja horaria seleccionando el botón “Nueva Franja”
3.A.1	Introduce los datos necesarios
3.B	El usuario Activa o Desactiva la franja horaria deseada seleccionando dicho botón
4	Finalizar
Alternativas	
3.A.1.A	¿Los datos introducidos son correctos?
3.A.1.A.1	Si sí, continuar
3.A.1.A.2	Si no, volver a 3.A.1

Tabla 3.55 Caso de Uso Activación/Desactivación Horarios

Caso de Uso	Visualización Fechas No Seleccionables
Actor Primario	Usuario
Sistema	Sistema del Centro Deportivo
Nivel	Subfunción
Condición Previa	Existan fechas no disponibles
Operaciones Básicas	
1	El usuario se dirige a la sección Perfil o seleccionará el botón “Reservar” de la página principal
2	El usuario visualiza las fechas en las que no se pueden reservar las instalaciones
Alternativas	Ninguna

Tabla 3.56 Caso de Uso Visualización Fechas No Seleccionables

Caso de Uso	Listado Reservas
Actor Primario	Administrador
Sistema	Sistema del Centro Deportivo
Nivel	Objetivo Usuario
Condición Previa	Haber iniciado sesión como Administrador
Operaciones Básicas	
1	El administrador se dirige a la sección Admin
2	Selecciona el botón “Reservas”
3	Visualiza el listado completo de reservas realizadas
Alternativas	Ninguna

Tabla 3.57 Caso de Uso Listado Reservas

Caso de Uso	Cancelación/Visualización de Reservas Pendientes
Actor Primario	Administrador
Sistema	Sistema del Centro Deportivo
Nivel	Objetivo Usuario
Condición Previa	Haber iniciado sesión como Administrador
Operaciones Básicas	
1	El administrador se dirige a la sección Admin
2	Selecciona el botón “Reservas”
3	Selecciona el botón “Reservas Pendientes”
4	Selecciona la opción deseada
4.A	Visualiza las reservas pendientes
4.B	Cancela la reserva deseada
5	Finalizar
Alternativas	Ninguna

Tabla 3.58 Caso de Uso Listado Reservas Pendientes

Caso de Uso	Listado Mis Reservas
Actor Primario	Usuario
Sistema	Sistema del Centro Deportivo
Nivel	Objetivo Usuario
Condición Previa	El usuario haya iniciado sesión
Operaciones Básicas	
1	El usuario se dirige a la sección Perfil
2	Selecciona el botón “Mis Reservas”
3	Visualiza las reservas realizadas
Alternativas	Ninguna

Tabla 3.59 Caso de Uso Listado Mis Reservas

Caso de Uso	Realización de Reserva
Actor Primario	Usuario
Sistema	Sistema del Centro Deportivo
Nivel	Objetivo usuario
Condición Previa	El usuario haya iniciado sesión
Operaciones Básicas	
1	El usuario se dirige a la sección Perfil o seleccionará el botón “Reservar” de la página principal
2	El usuario selecciona el día a reservar
3	El usuario selecciona la pista
4	El usuario selecciona la hora
5	El usuario selecciona el método de pago
6	Finalizar
Alternativas	
2.A	¿Está disponible el día?
2.A.1	Si sí, continuar
2.A.2	Si no, mostrar mensaje de error y volver a 2
4.A	¿Quedan horas disponibles?
4.A.1	Si sí, continuar
4.A.2	Si no, mostrar mensaje de error y volver al 2
5.A	¿Se ha realizado el pago correctamente?
5.A.1	Si sí, continuar
5.A.2	Si no, el usuario se dirige a su Perfil, “Mis Reservas” y hará efectivo el pago
6.A	¿El usuario desea volver a reservar?
6.A.1	Si sí, seleccionar botón “Volver a Reservar”
6.A.2	Si no, seleccionar botón “Perfil”

Tabla 3.60 Caso de Uso Realización de Reserva

Caso de Uso	Visualización/Modificación datos personales
Actor Primario	Usuario
Sistema	Sistema del Centro Deportivo
Nivel	Objetivo Usuario
Condición Previa	El usuario haya iniciado sesión
Operaciones Básicas	
1	El usuario se dirige a la sección Perfil
2	Selecciona la opción deseada
2.A	Visualiza sus datos personales
2.B	Modifica sus datos personales seleccionando el botón “Modifica tus Datos”
2.B.1	Introduce los datos necesarios
2.B.2	Selecciona “Actualizar Datos”
3	Finalizar
Alternativas	
2.B.1.A	¿Los datos modificados son correctos?
2.B.1.A.1	Si sí, continuar
2.B.1.A.2	Si no, volver a 2.B.1

Tabla 3.61 Caso de Uso Visualización/Modificación Datos Personales

Caso de Uso	Selección Método de Pago
Actor Primario	Usuario
Sistema	Sistema del Centro Deportivo
Nivel	Subfunción
Condición Previa	El usuario haya iniciado sesión y rellenado los datos de la reserva
Operaciones Básicas	
1	El usuario selecciona el método de pago deseado
2	Finalizar
Alternativas	
1.A	¿Pago se ha realizado correctamente?
1.A.1	Si sí, continuar
1.A.2	Si no, se dirigirá a Perfil, “Mis Reservas” y seleccionará el método de pago deseado de la reserva en cuestión

Tabla 3.62 Caso de Uso Selección Método de Pago

Caso de Uso	Confirmación de Reserva
Actor Primario	Usuario
Sistema	Sistema del Centro Deportivo
Nivel	Subfunción
Condición Previa	Haber realizado y pagado una reserva
Operaciones Básicas	
1	El usuario recibirá un email con la confirmación de la reserva y todos sus datos asociados
Alternativas	Ninguna

Tabla 3.63 Caso de Uso Confirmación de Reserva

Caso de Uso	Ubicación Google Maps
Actor Primario	Usuario
Sistema	Sistema del Centro Deportivo
Nivel	Subfunción
Condición Previa	Ninguna
Operaciones Básicas	
1	El usuario se dirige a la sección Contacto
2	Visualiza la ubicación del centro deportivo
Alternativas	Ninguna

Tabla 3.64: Caso de Uso Ubicación Google Maps

Caso de Uso	Previsión Meteorológica
Actor Primario	Usuario
Sistema	Sistema del Centro Deportivo
Nivel	Subfunción
Condición Previa	Ninguna
Operaciones Básicas	
1	El usuario se dirige a la Página de Inicio
2	Visualiza la previsión meteorológica de este a través de un widget
Alternativas	Ninguna

Tabla 3.65: Caso de Uso Previsión Meteorológica

Para el apartado de la parametrización, juntaremos todas las historias de usuario relacionadas con esta en un único caso de uso, en el que se seguirá la secuencia que el administrador debe seguir para crear y personalizar a su gusto el sitio web.

Caso de Uso	Parametrización
Actor Primario	Administrador
Sistema	Sistema del Centro Deportivo
Nivel	Objetivo Usuario
Condición Previa	Sea la primera vez que se arranque la aplicación web
Operaciones Básicas	
1	El administrador rellenará el formulario de configuración general con la información acerca del centro deportivo
2	El administrador se registrará con los datos necesarios como usuario Administrador
3	El administrador seleccionará las fechas las cuales no pueden reservarse las pistas
4	El administrador rellenará el formulario para la creación de pistas
5	El administrador registrará los horarios para cada pista
6	Finalizar
Alternativas	
1.A	¿Los datos introducidos son correctos?
1.A.1	Si sí, continuar
1.A.2	Si no, volver a 1
2.A	¿Los datos introducidos son correctos?
2.A.1	Si sí, continuar
2.A.2	Si no, volver a 2
3.A	¿Los datos introducidos son correctos?
3.A.1	Si sí, continuar
3.A.2	Si no, volver a 3
4.A	¿Los datos introducidos son correctos?
4.A.1	Si sí, continuar
4.A.2	Si no, volver a 4
5.A	¿Desea añadir más pistas?
5.A.1	Si sí, seleccionar Crear otra pista y volver a 4
5.A.2	Si no, confirmar y continuar

Tabla 3.66 Caso de Uso Parametrización

4. DISEÑO

En este capítulo comenzaremos diseñando el diagrama de clases de nuestra aplicación para seguidamente realizar el diseño inicial de la interfaz de usuario haciendo uso de Wireframes.

4.1. Diagrama de Clases

Los diagramas de clases se corresponden con la lógica de negocio de la aplicación, en estos se definen las clases a implementar junto con sus atributos y métodos.

Para la realización de nuestro diagrama de clases separaremos a este en dos, uno específico para la funcionalidad de nuestra aplicación y otro para el apartado de la parametrización.

En la siguiente imagen aparece el diagrama de clases que hemos obtenido en las primeras iteraciones de nuestro proyecto.

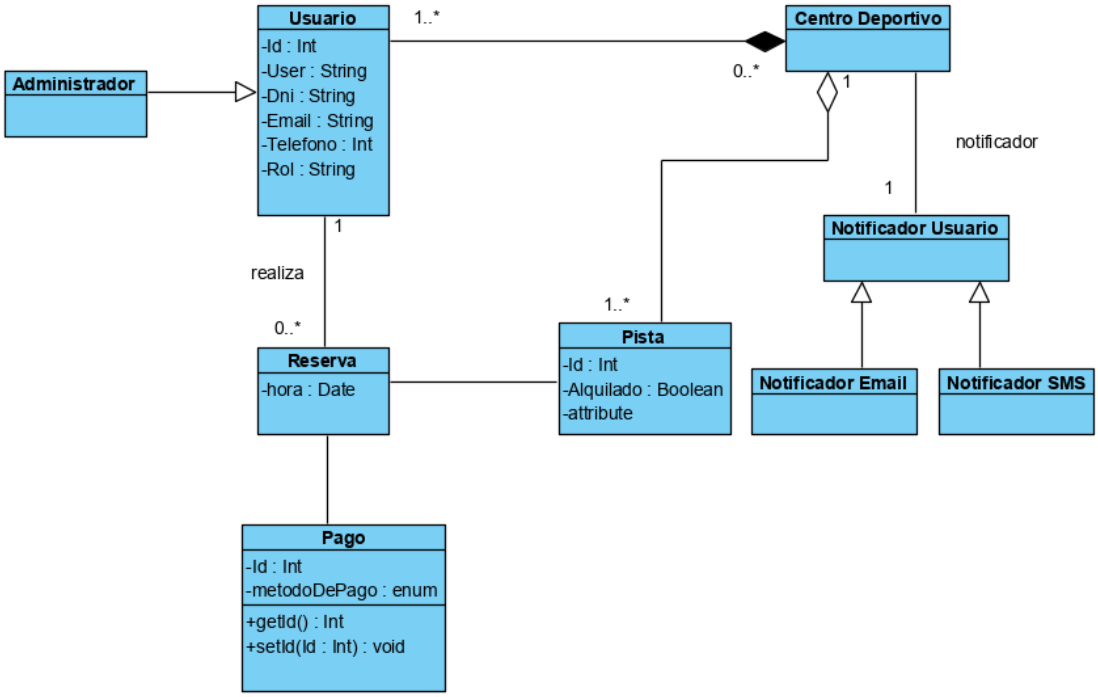


Ilustración 4.10 Diagrama de Clases Primeras Iteraciones

A continuación, mostraremos el diagrama de clases final obtenido en las últimas iteraciones y una breve descripción de la función de cada una de las clases que forman dicho diagrama.

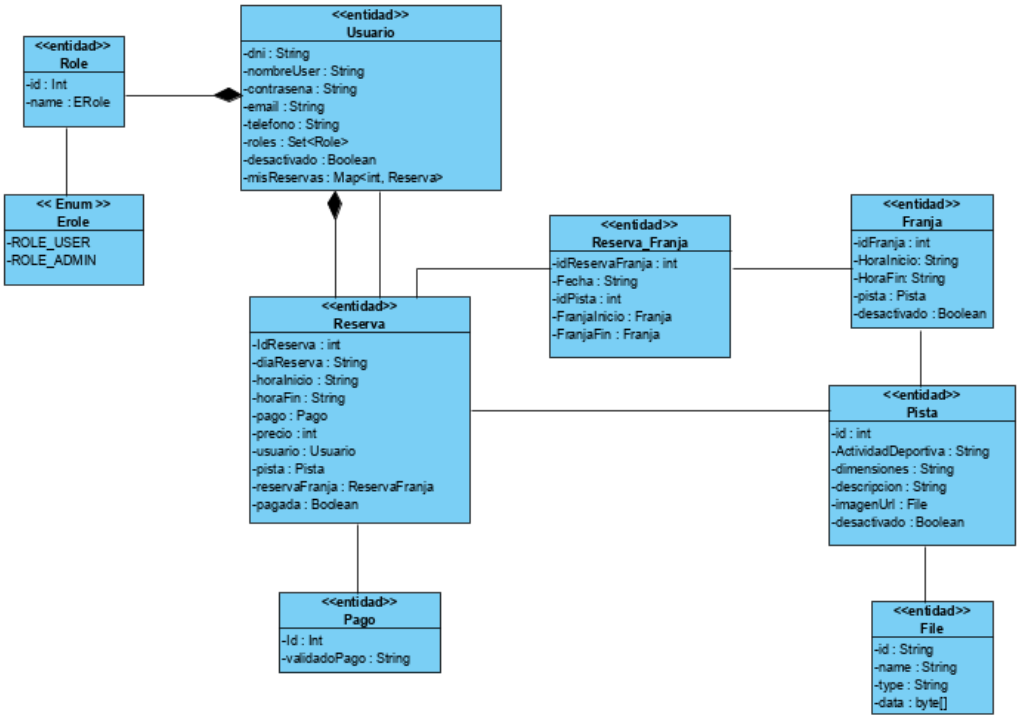


Ilustración 4.11 Diagrama de Clases Final

Las clases finales con las que cuenta son las siguientes:

La clase **Usuario** representa a los usuarios del sistema. Cada uno contiene los atributos necesarios para su identificación. Esta tiene dos relaciones de composición, una con la clase **Reserva** para que quedan registradas todas las reservas realizadas por un usuario, para su posterior visualización, y otra con la clase **Role** para conocer los roles de cada usuario.

La clase **Pago** representa el pago de una reserva, esta contiene un código aleatorio único de validación que se genera al realizar el pago.

La clase **Pista** contiene toda la información de cada una de las pistas del centro deportivo. Esta cuenta con una relación con la clase **File**, necesaria para la imagen asociada a cada pista.

En la clase **Franja** nos encontramos con el horario de cada pista, cada una contiene la hora de inicio y de fin ya que como hemos comentado anteriormente, cada franja horaria debe ser de una hora de duración. Además, una relación de asociación con la clase **Pista** para saber a cuál pertenece.

La clase más compleja es la de **Reserva**, en esta tenemos la fecha seleccionada, la hora de inicio y fin, el precio correspondiente y un atributo para conocer si está pagada o no. Además, una relación de asociación para obtener los datos de las clases **Usuario** y **Pago**, y una clase intermedia llamada **Reserva_Franja** de la cuál hablaremos a continuación.

La clase **Reserva_Franja** ha sido creada con cierta información de la clase **Reserva** y **Franja** para así poder acceder a través de ciertos métodos y consultas y validar que no se puedan realizar dos reservas de una misma pista a la misma hora y fecha.

Para la realización del diagrama de clases de la parametrización, este fue diseñado una vez contábamos con la funcionalidad total de nuestra aplicación, por ello a continuación mostraremos dicho diagrama obtenido en las últimas iteraciones.

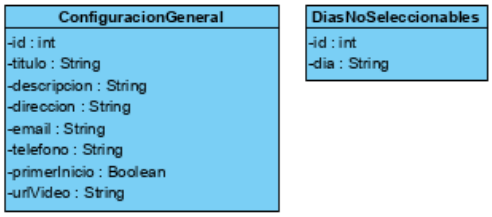


Ilustración 4.12 Diagrama de Clases Parametrización Final

Como podemos ver, las clases con las que cuenta son las siguientes:

ConfiguracionGeneral: contiene toda la información para la personalización de nuestro sitio web. Gracias a ella, el usuario podrá introducir campos como el título, descripción, teléfono de contacto, email, etc. Además, contará con una variable booleana para conocer si es la primera vez que se arranca el sitio web y poder realizar la personalización de esta.

DiasNoSeleccionables: en esta tabla se almacenarán los días los cuales no se pueden alquilar las pistas de las que dispone el espacio deportivo.

4.2. Diseño de la Interfaz

En este apartado mostraremos los Wireframes que explicarán el diseño que se ha llevado a cabo para obtener una idea de cómo debe quedar nuestra aplicación.

Un Wireframe es un boceto en el que se representa visualmente la estructura de un sitio web de forma muy sencilla y esquemática.

Una vez realizado el estudio acerca de los requisitos con los que debe contar nuestra aplicación, vamos a diseñar y mostrar cada una de las páginas/componentes con las que contará dicha aplicación.

En cada iteración realizaremos el diseño de una página/componente mostrando las imágenes creadas en iteraciones tempranas, quedando así una estimación de cómo debería ser nuestra página/componente y, para finalizar, habiendo realizado un estudio más a fondo, la iteración final que nos servirá de esqueleto para nuestra aplicación real. Cabe destacar que algunas páginas/componentes tiene el mismo diseño que al crearlas en las iteraciones iniciales ya que no se han producido cambios.

Mostraremos únicamente los Wireframe desde la vista de un ordenador, esto se debe a que como ya hemos comentado, nuestra aplicación web será accesible desde cualquier dispositivo ya sea ordenador, móvil o tablet haciendo uso de la técnica “Responsive Web Design”, de esta forma el diseño se ajustará automáticamente a diferentes tamaños de pantalla.

Para realizar los Wireframes hemos hecho uso de la web **Mockflow**.

4.2.1. Página Principal

En la iteración 0 para el diseño de la Página Principal nos quedó de la siguiente manera:

Ilustración 4.13: Wireframe Inicial Página de Inicio

Esta cuenta con una barra de navegación para poder movernos, un widget para ver la previsión meteorológica, una descripción de nuestro recinto deportivo y los métodos de pago permitidos para realizar una reserva.

Tras varias iteraciones y diversos cambios, el Wireframe final de la Página Principal es el siguiente:

Ilustración 4.14 Wireframe Final Página de Inicio

Como podemos observar, además de con los elementos que ya contaba, incluimos un carrusel con las imágenes de nuestras instalaciones, un video donde de nuestro centro deportivo, un calendario para consultar los días que se pueden reservar y, diversos botones.

4.2.2. Página de Contacto

En la iteración 0, contaba con un formulario de contacto para dudas o consultas de los usuarios y la localización de Google Maps de nuestro recinto.

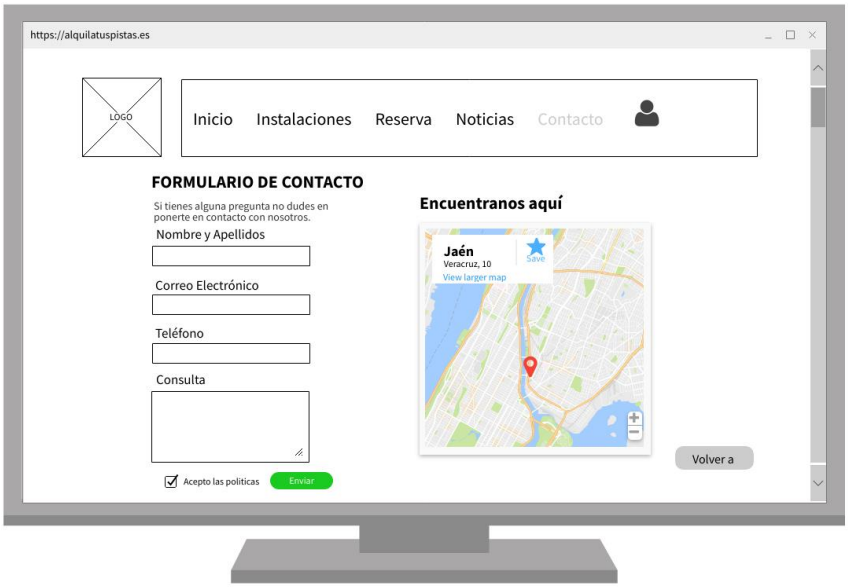


Ilustración 4.15 Wireframe Inicial Página de Contacto

El Wireframe final del Formulario de contacto queda así:

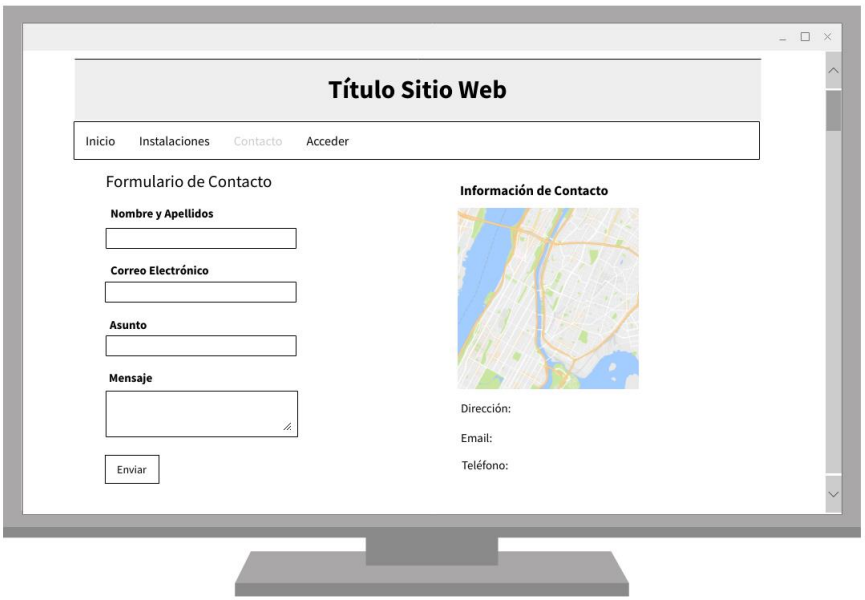


Ilustración 4.16 Wireframe Final Página de Contacto

Después de varias iteraciones, además de seguir contando con un formulario para dudas o consultas y la localización, hemos añadido información adicional acerca del reciento.

4.2.3. Pistas

En cuanto a las pistas de nuestro centro deportivo, las páginas las cuales podrán ser visualizadas por los usuarios con rol “USER” son, Visualización Pistas, Detalle de Pista y Horario Pista. Por otro lado, las páginas para la gestión de la aplicación y que únicamente debe acceder el usuario con rol “ADMIN” son, Crear Pista, Editar Pista y Listado Pistas

Cabe destacar que algunas de las siguientes páginas fueron diseñadas tras varias iteraciones y cambios, y otras únicamente durante la iteración final de diseño ya que no se produjeron cambios.

1. ***Visualización Pistas***

Para ver las pistas con la que cuenta nuestro centro deportivo tenemos la pestaña **Instalaciones**, durante la iteración 0, contábamos con un filtro para seleccionar de qué actividad deportiva queríamos ver sus pistas. En este ejemplo, tenemos seleccionado futbol y cómo podemos ver nos aparecerían dichas pistas, además de una pequeña descripción y el precio/hora de cada una de ellas.

Ilustración 4.17 Wireframe Inicial Visualización Pistas

En la iteración final las modificaciones que se produjeron son las siguientes:

En lugar de mostrar las pistas filtradas por la actividad deportiva, decidimos eliminar dicho filtro y mostrar todas las pistas con las que cuenta el recinto, además,

para cada una de ellas añadimos unos botones para ver su información y horario correspondiente.

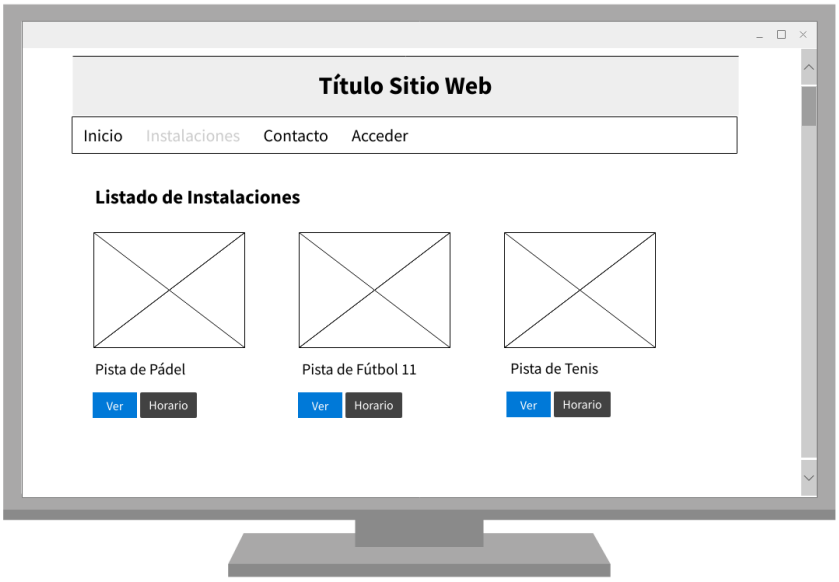


Ilustración 4.18 Wireframe Final Visualización Pistas

2. Detalle de Pista

Para la página de Detalle de una Pista, el diseño ha sido el mismo que se realizó durante la iteración 0 e iteración final ya que no ha habido ningún cambio.

En esta contamos con una imagen de la pista junto con información acerca de esta y un botón para volver a la página de inicio.

Ilustración 4.19 Wireframe Final Detalle Pista

3. Horario Pista

Como es el caso del Wireframe anterior, durante la iteración 0 y final no se producen cambios.

En dicha página, contamos con una tabla donde se muestra para cada una de las pistas las franjas horarias que componen el horario de estas, con una hora de inicio y fin.

Ilustración 4.20 Wireframe Final Horario Pista

4. Crear Pista

En la iteración Final, para la creación de una pista contamos con un formulario con los campos actividad deportiva, descripción, dimensiones y un fichero de imagen. Además, los botones para Guardar y Salir.

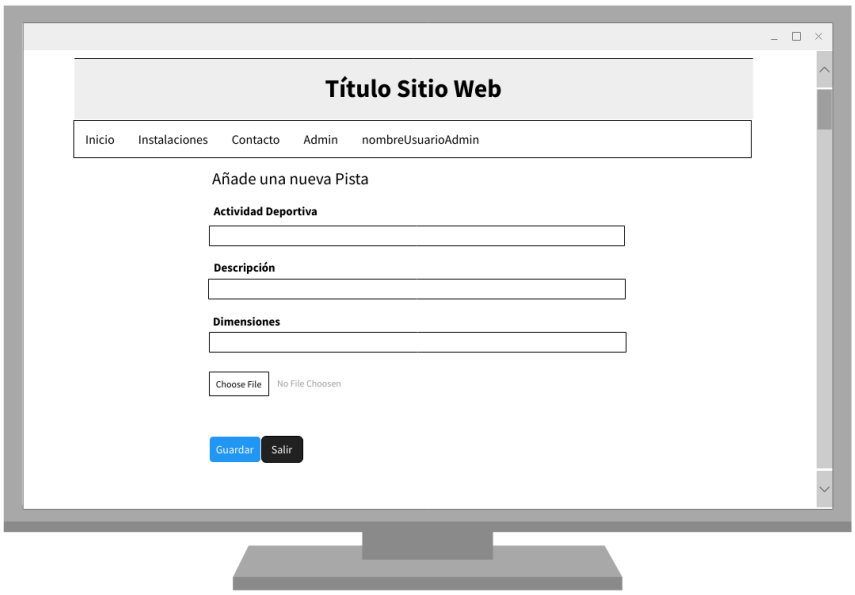


Ilustración 4.21 Wireframe Final Crea Pista

5. Editar Pista

A la hora de editar una pista, en la iteración final, como podemos observar, tenemos un formulario con los mismos campos que la página Crear Pista, la única diferencia es que dichos campos aparecerán rellenos con la información de la pista a modificar.

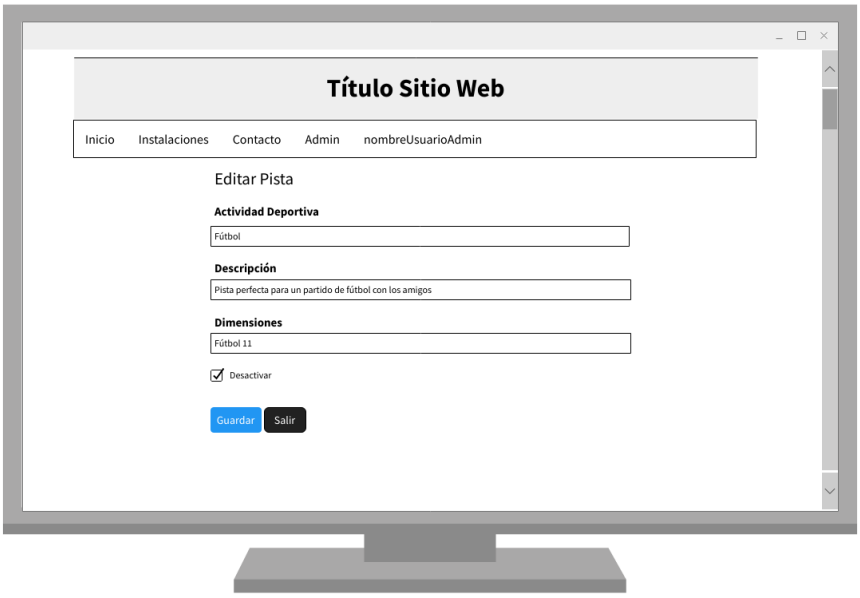


Ilustración 4.22 Wireframe Final Edita Pista

6. Listado Pistas

Dicha página contará con una tabla con los datos de las pistas, un botón para añadir una nueva pista y un botón para editar dicha pista.

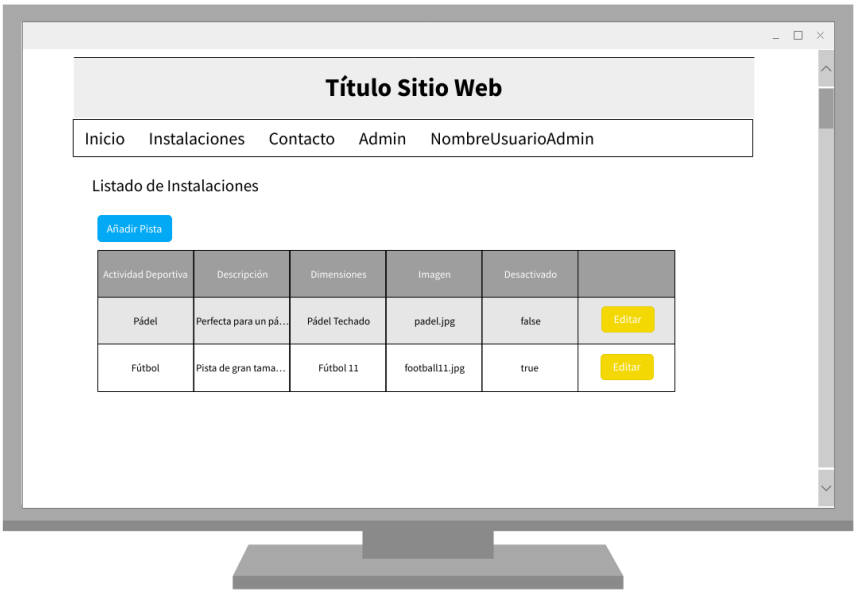


Ilustración 4.23 Wireframe Final Listado Pista

4.2.4. Reservas

1. Realiza Reserva

En la iteración 0, a la hora de realizar una reserva nos dirigíamos a la pestaña **Reserva** y una vez allí seleccionábamos la actividad deportiva deseada y la pista. Después de la selección anterior, en el calendario aparecían los días y un cuadro con las horas disponibles.

Ilustración 4.24: Wireframe Inicial Reserva 1

Con los datos introducidos para la reserva, en esta página aparecía un cuadro con todos los datos acerca de esta y con el precio final. Si todo era correcto, para finalizar, seleccionábamos el método de pago para hacer esta reserva efectiva.

Ilustración 4.25: Wireframe Inicial Reserva 2

Una vez realizada la reserva, el usuario recibiría un email para confirmar esta.

Ilustración 4.26: Wireframe Inicial Reserva 3

Durante la iteración final de dicha página, hemos realizado los siguientes cambios:

A la hora de realizar una reserva, en la parte derecha aparecerán los días que no se pueden reservas las pistas. En la parte izquierda, tenemos un calendario para seleccionar la fecha a reservar y la pista deseada.

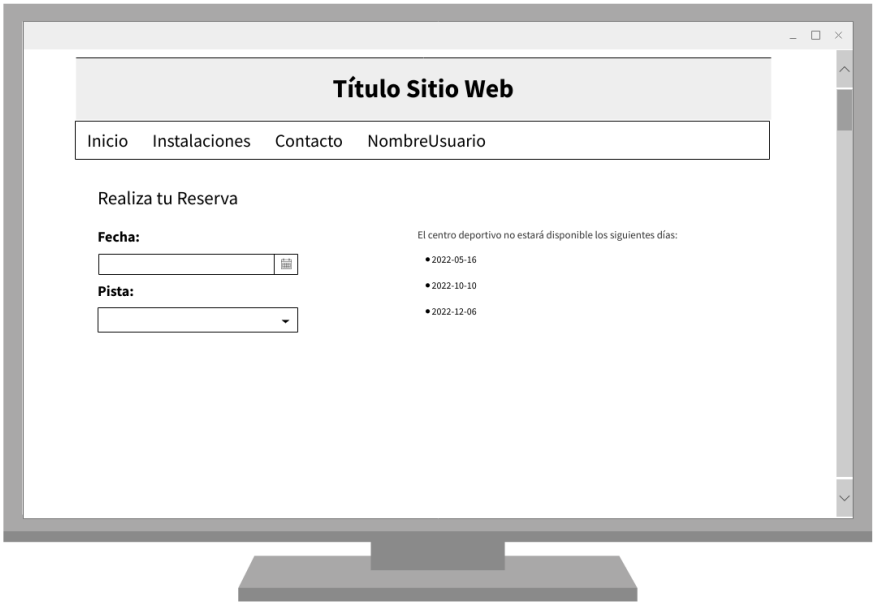


Ilustración 4.27 Wireframe Final Reserva 1

Una vez introducida la información anterior nos aparece un nuevo campo desplegable con las horas de inicio disponibles, al haber seleccionado esta, aparecerá la hora final de forma automática ya que como bien sabemos únicamente se pueden reservar franjas de una 1 hora.

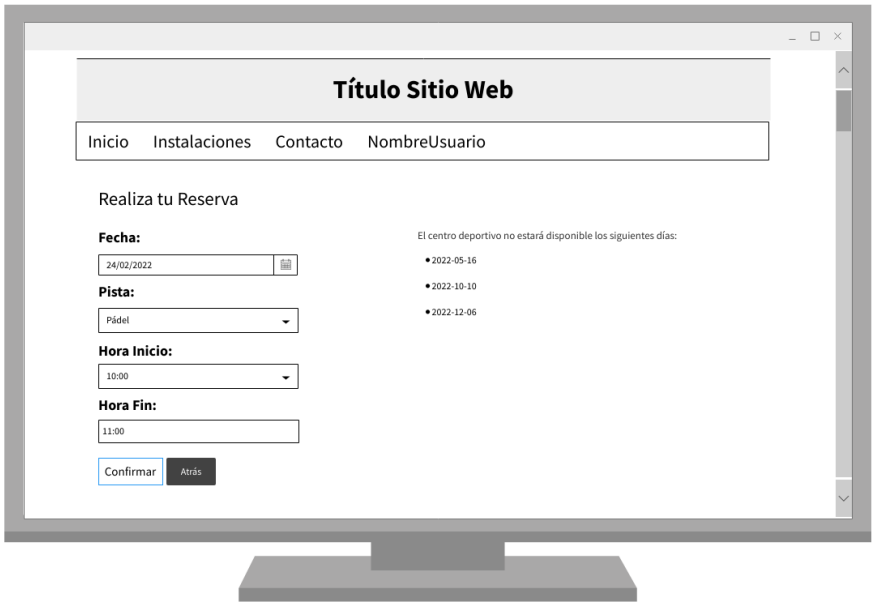


Ilustración 4.28 Wireframe Final Reserva 2

Al completar toda la información para la reserva, se mostrará el precio final de la reserva y un botón para realizar el pago.

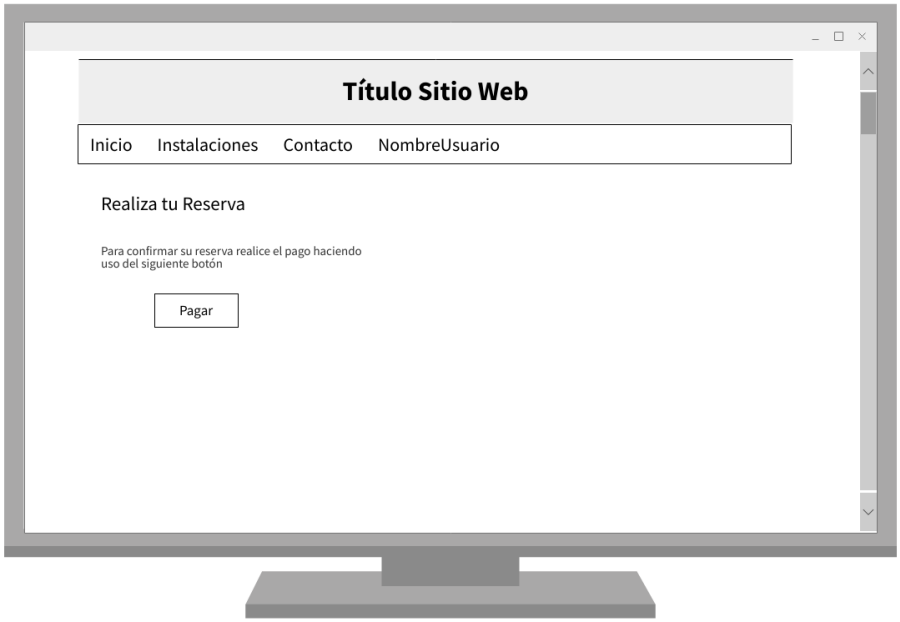


Ilustración 4.29 Wireframe Final Reserva 3

Por último, si el pago se ha realizado correctamente, se mostrará un mensaje con la confirmación de la reserva y unos botones para volver a reservar o redirigirnos a nuestro perfil.

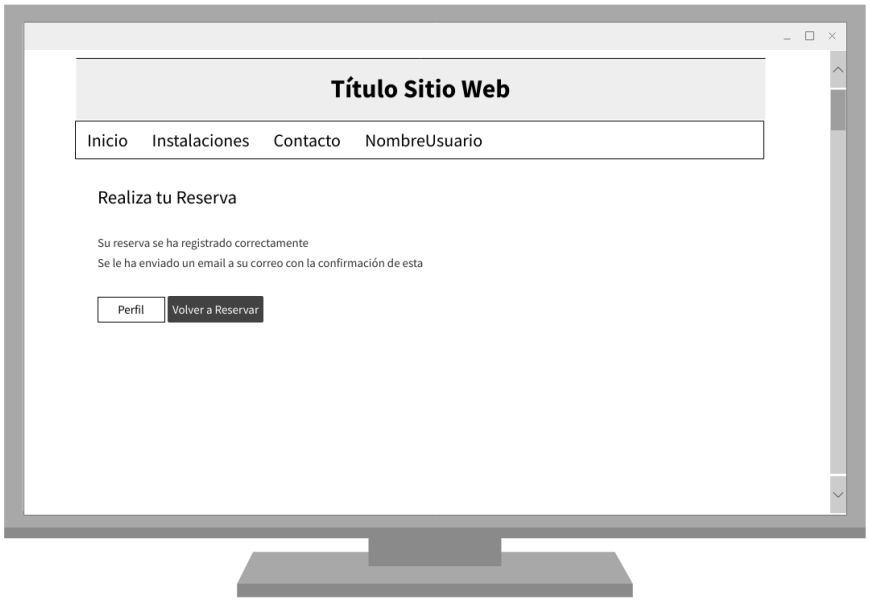


Ilustración 4.30 Wireframe Final Reserva 4

2. Listado Reservas

Dicha página consta de una tabla con los datos de las reservas, un botón para redirigirte a las reservas pendientes de pago y un botón para volver a atrás.

Ilustración 4.31 Wireframe Final Listado Reservas

La imagen anterior es igual para la visualización de las reservas realizadas por un usuario y para las reservas pendientes de pago, pero con sus datos correspondientes.

La única diferencia con respecto de las reservas pendientes de pago es el botón para que el administrador pueda cancelarlas.

4.2.5. Administrador

Durante la iteración 0 nuestra idea era que el **Administrador** pudiera gestionar los datos personales de los usuarios ya registrados, y las reservas realizadas.

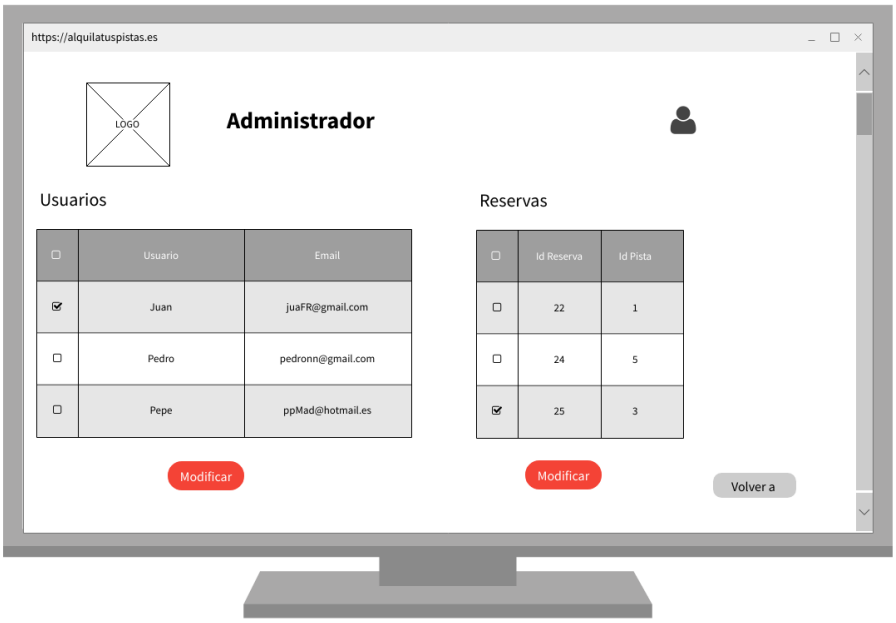


Ilustración 4.32: Wireframe Inicial Administrador

Finalmente, tras los cambios realizados, el Wireframe de la iteración final cuenta con un panel para gestionar usuarios, instalaciones, franjas horarias, la selección de fechas no disponibles y reservas.

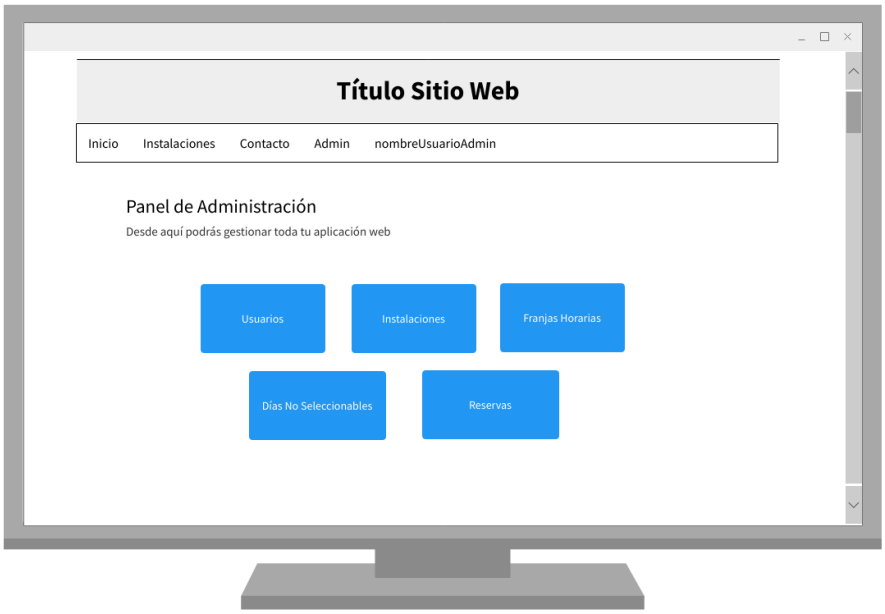


Ilustración 4.33 Wireframe Final Administrador

4.2.6. Usuarios

Todas las páginas relacionas con usuarios han sido diseñadas durante la iteración final ya que como hemos comentado en otros componentes anteriormente, no se han producido cambios.

1. ***Registra Usuario***

Cuenta con un formulario con todos los datos necesarios para la creación de un nuevo usuario, además de los botones para Guardar y Salir.

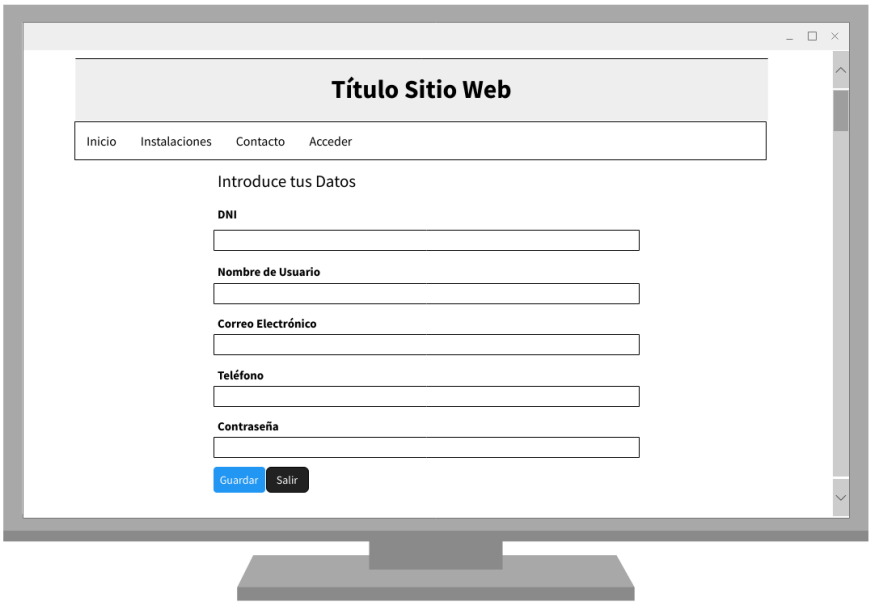


Ilustración 4.34 Wireframe Final Registra Usuario

2. Listado Usuarios

Para que el usuario con rol “ADMIN” pueda gestionar a todos los usuarios, cuenta con una tabla con los datos de estos, un botón para activarlos o desactivarlos y otro para añadir nuevos usuarios en caso de prueba.

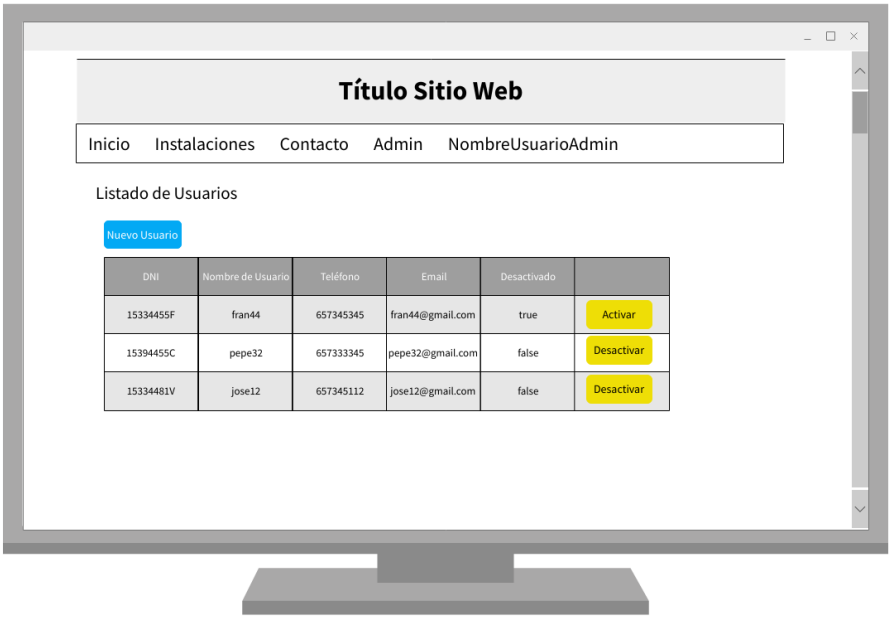


Ilustración 4.35 Wireframe Final Listado Usuarios

3. Modifica Datos Usuario

A la hora de modificar los datos personales de un usuario, encontramos un formulario relleno con sus datos los cuales podrán ser modificados a su gusto. En esta página debemos de destacar que el usuario también podrá modificar la contraseña.

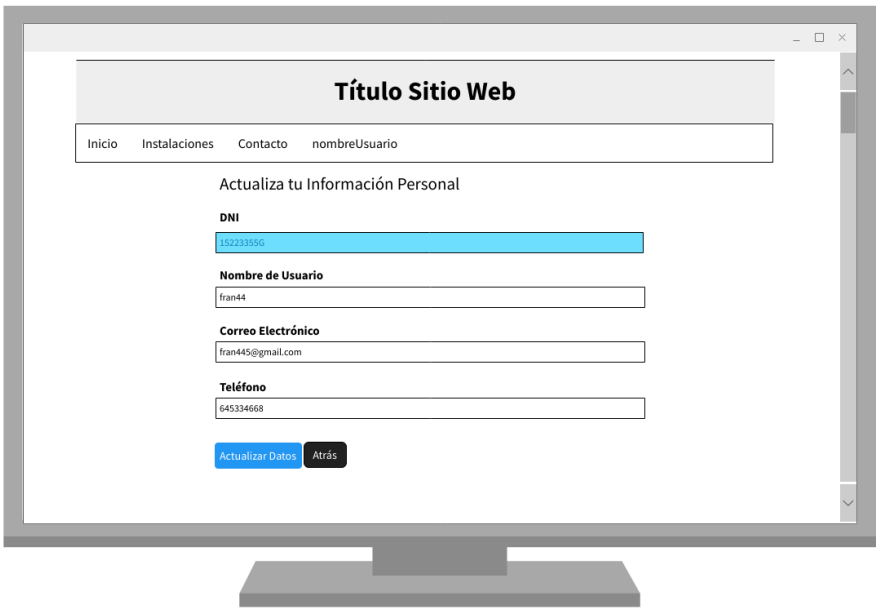


Ilustración 4.36 Wireframe Final Modifica Datos Usuario

4.2.7. Franjas Horarias

1. Nueva Franja Horaria

Para dicha página, en la iteración final, se decidió tener un formulario con los campos de hora de inicio, hora de fin y la pista a la que asignarle dicho horario.

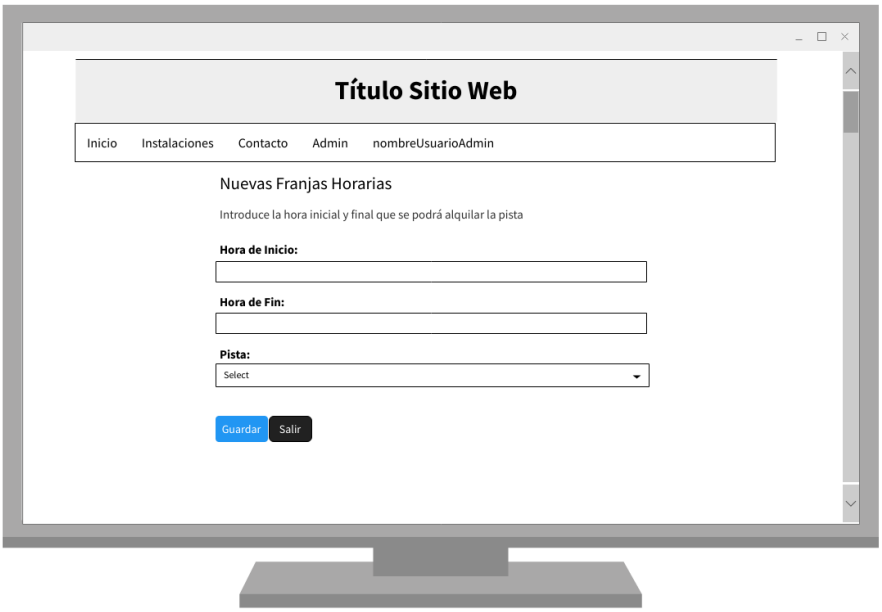


Ilustración 4.37 Wireframe Final Nueva Franja

2. *Listado Franjas Horarias*

El usuario con rol “ADMIN” contará con una tabla con la información completa de los horarios del recinto, divido en franjas horarias de una hora de duración. Además, podrá Activarlas/Desactivarlas a su gusto y crear nuevas.

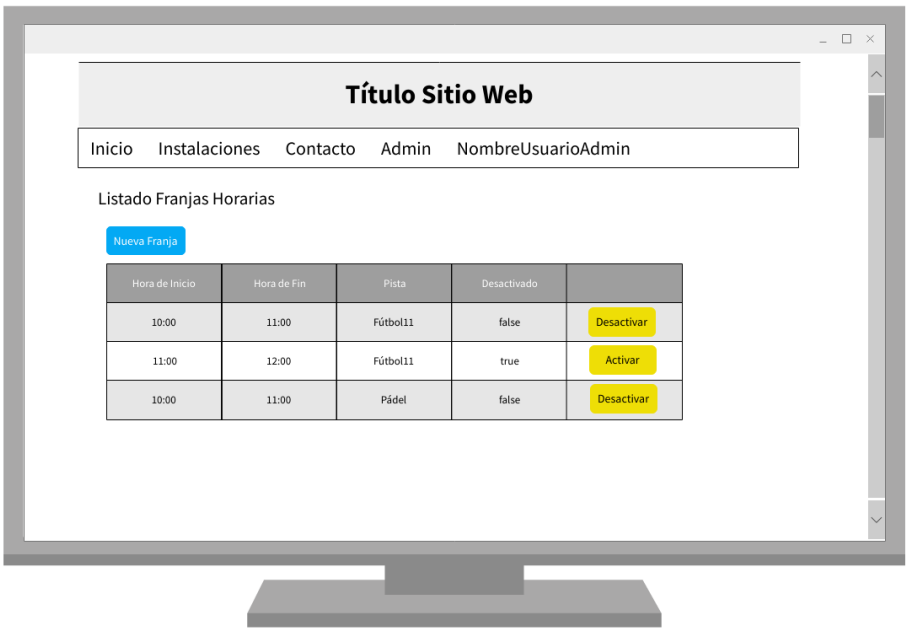


Ilustración 4.38 Wireframe Final Listado Franjas Horarias

4.2.8. Login

Para el Login únicamente en la iteración final decidimos insertar un pequeño formulario donde el usuario debe introducir su nombre de usuario y contraseña.

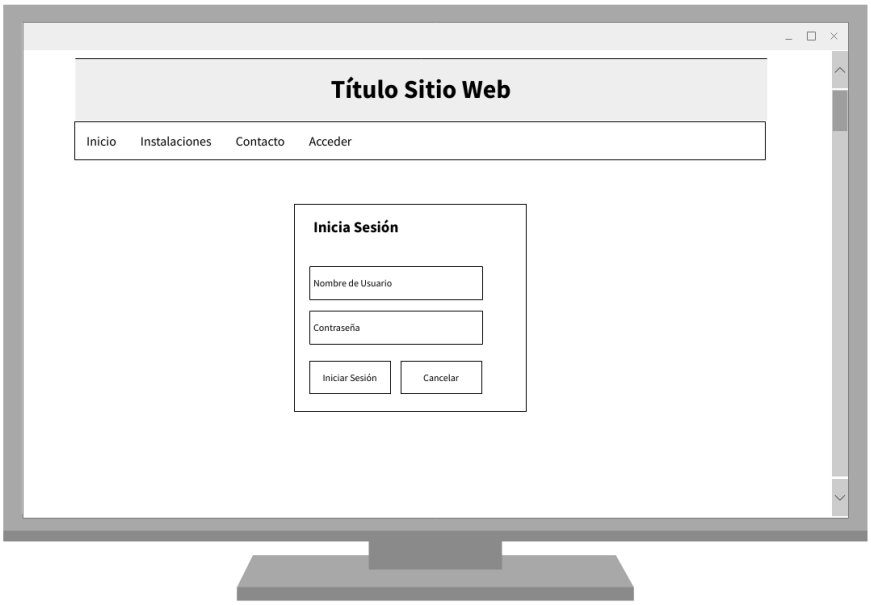


Ilustración 4.39 Wireframe Final Login

4.2.9. Fechas No Seleccionables

Para la selección de fechas que no podrán reservarse, en la iteración final, el usuario con rol “ADMIN” desde la parte izquierda podrá añadirlas seleccionándolas directamente desde el calendario. En la parte derecha podrá ver los días que no se pueden reservar y eliminarlos seleccionando el botón Reestablecer.

Ilustración 4.40 Wireframe Final Fechas No Seleccionables

4.2.10. Perfil

La página Perfil fue diseñada de la siguiente manera: en la parte izquierda se muestra la información del usuario junto con el botón para cerrar sesión y en la parte derecha, 3 botones para crear una nueva reserva, visualizar las reservas realizas y modificar los datos personales del usuario.

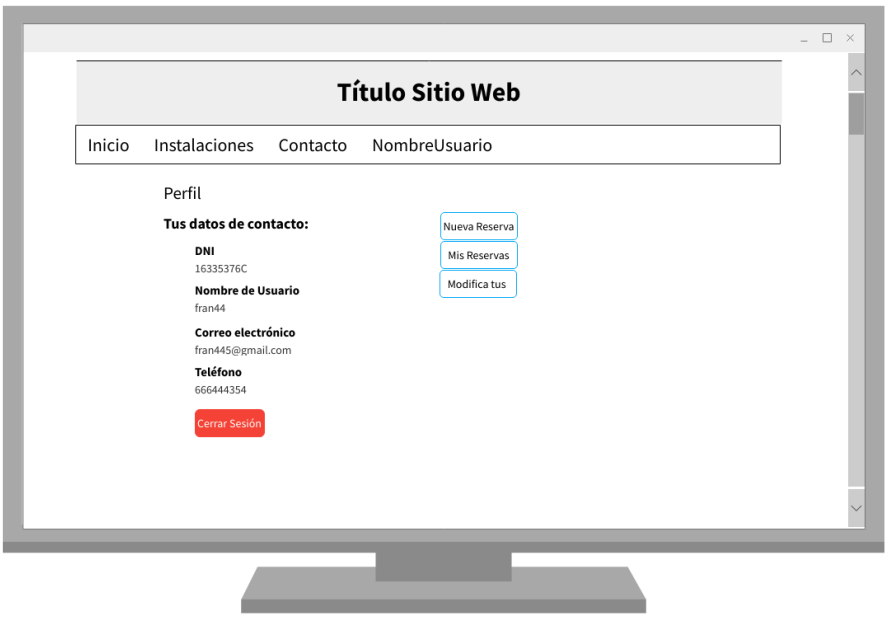


Ilustración 4.41 Wireframe Final Perfil

4.2.11. Parametrización

A continuación, mostraremos los Wireframe acerca de la parametrización de nuestra página, donde el usuario podrá modificar a su gusto los diferentes parámetros comentados anteriormente en *Requerimientos Funcionales* en el apartado parametrización.

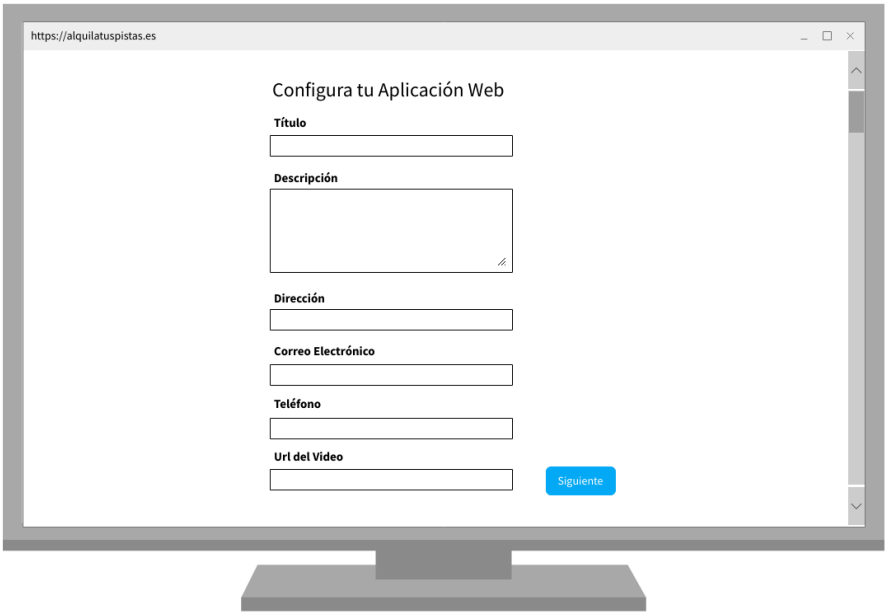


Ilustración 4.42 Wireframe Final Parametrización

La imagen anterior se corresponde con la iteración final, en ella podemos observar un formulario con los campos necesarios para detallar la información acerca del recinto deportivo.

Para la configuración y parametrización de la aplicación, únicamente hemos mostrado en este apartado el formulario para añadir la información del centro deportivo ya que la demás información como instalaciones, horarios, usuarios, ya ha sido mostrada de forma individual en los Wireframes anteriores.

El diseño final y la funcionalidad de nuestra aplicación se mostrará en el *Apéndice C: Manual de Usuario* más adelante.

5. IMPLEMENTACION

En este apartado nos centraremos en algunos de los aspectos más importantes en cuanto a la implementación. Para ello describiremos cuál ha sido el proceso para conseguir la autenticación, autorización, pasarela de pago, etc.

Comenzaremos con la base de datos, ya que es uno de los puntos por los que debemos de comenzar en la implementación, comentaremos también la autenticación y autorización, las Guards de Angular y pasarela de pago. Además, realizaremos la documentación de la Api y la estructura de nuestro Backend y Frontend, para, por último, y no menos importante, la dockerización de nuestra aplicación.

5.1. Base de Datos

Como hemos comentado en el apartado de Base de Datos del capítulo ESTUDIO DE METODOLOGIAS Y TECNOLOGIAS, haremos uso de JPA Hibernate.

A continuación, vamos a mostrar una clase de nuestro backend donde haciendo uso de las anotaciones de Hibernate mapearemos nuestra entidad “Usuario”:

```
@Entity
public class Usuario{
    @Id
    @NotNull
    @Pattern(regexp=Formato.DNI)
    private String dni;
    @NotBlank(message = "El nombre de Usuario no puede estar vacio")
    @NotNull
    @Column(name ="nombre_user" ,unique = true)
    private String nombreUser;
    @ManyToMany(fetch = FetchType.LAZY)
    private Set<Role> roles = new HashSet<>();
    @NotNull
    @NotBlank(message = "Contraseña es obligatoria")
    @Lob
    private String contrasena;
    @NotNull
    @NotBlank(message = "Telefono no puede estar vacio")
    @Pattern(regexp ="(^$|[0-9]{9})",message = "Formato de telefono invalido")
    private String telefono;
    @NotNull
    @Email
    private String email;
    private boolean desactivado;
    @OneToMany(fetch = FetchType.EAGER)
    @MapKey
    Map<Integer, Reserva> misReservas;
```

Ilustración 5.43 Entidad Usuario

Como podemos ver en la imagen anterior usamos algunas de las anotaciones más importantes como: **@Entity** para indicar que la clase Java es una clase entidad, **@Id** para la clave primaria, **@Email** para correo electrónico y **@OneToMany** para una relación de 1 a muchos. Las demás anotaciones son para la validación y restricción de datos.

Como resultado en la base de datos se genera automáticamente la siguiente tabla:

Ilustración 5.44 Campos Tabla BBDD Usuario

dni	contrasena	desactivado	email	nombre_user	telefono
15555555G	$2a$10$jfi2ak0Y4pWE7QEUZlJCg.miy3ubI4l63R...	0	pepe@gmail.com	pepepe	666455333
15556688V	$2a$10$E58UChoClqoLU6kCFXk3q.LU4wB0d9m...	0	jose7madridista@gmail.com	fran66	678654666
NULL	NULL	NULL	NULL	NULL	NULL

Ilustración 5.45 Tabla BBDD Usuario

Hemos generado las tablas mediante el sistema de generación de tablas que tiene MYSQL y el esquema resultante es el siguiente:

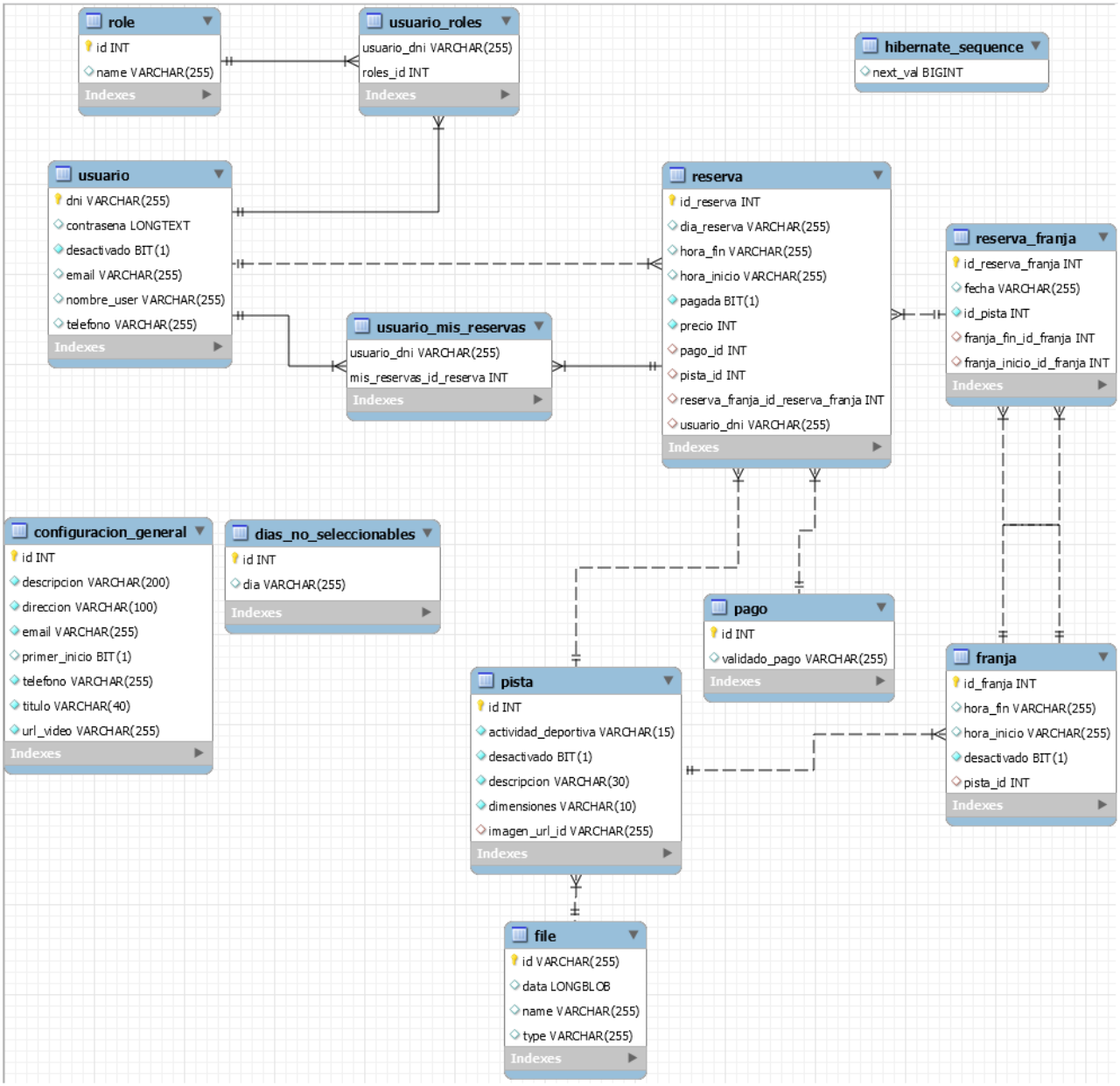


Ilustración 5.46 Esquema Tablas Generado

5.2. Autenticación y Autorización

Para nuestro servidor Backend con Spring Boot haremos uso de Spring Security para autenticación JWT [28] y autorización basada en roles. Por otro lado, para nuestro Frontend, HttpInterceptor y validación de formularios.

Lo primero que haremos será explicar de que trata JWT.

JWT (token web JSON)

Popular para la autenticación e intercambio de información. El servidor codificará los datos en un token web JSON, los enviará al cliente y los guardará. Cada solicitud

del cliente a recursos protegidos o a rutas debe adjuntarse en el encabezado ese JWT. Por último, el servidor validará dicho JWT y devolverá la respuesta.

JWT cuenta con tres partes importantes: encabezado, carga útil y firma. Cada uno de ellos se combinan en una estructura estándar: header.payload.signature.

5.2.1. En el lado del servidor:

Spring Security

- WebSecurityConfigurerAdapter: es el punto central de nuestra implementación de seguridad. Proporciona configuraciones de HttpSecurity para configurar cors, csrf, gestión de sesiones y reglas para recursos protegidos.
- Interfaz UserDetailsService: contiene un método para cargar un usuario por nombre usuario y devolver un objeto de tipo UserDetails que Spring Boot utilizará para la autenticación y validación.
- UserDetails: contiene la información necesaria como, nombre de usuario, contraseña y autoridades para construir un objeto Authentication.
- UsernamePasswordAuthenticationToken: obtiene nombre de usuario y contraseña de la solicitud de inicio de sesión, AuthenticationManager lo utilizará para autenticar una cuenta de inicio de sesión.
- AuthenticationManager: valida el objeto UsernamePasswordAuthenticationToken. Si tiene éxito devuelve un objeto Authentication completamente rellenado.
- OncePerRequestFilte: proporciona un método “doFilterInternal()” que implementaremos analizando y validando el JWT, cargando los detalles del usuario, comprobando la autorización.
- AuthenticationEntryPoint: cuando los clientes intenten acceder a recursos protegidos sin autenticación, este capturará el error y devolverá un 401.

Repositorios

Para trabajar con la base de datos y será importado en el controlador.

Servicio API Rest

Maneja las solicitudes de registro/inicio sesión.

Para asegurar cada uno de los métodos que componen nuestro API Rest, haremos uso de la anotación “@PreAuthorize”. En ella indicamos qué rol ya sea ‘USER’ o y ‘ADMIN’ debe tener el usuario para poder hacer uso de un determinado método. En la siguiente imagen podremos ver como el método solo está permitido para el Administrador.

```
@GetMapping("/reservasPendientes")
@PreAuthorize("hasRole('ADMIN')")
@ResponseStatus(HttpStatus.OK)
public List<ReservaDTO> listaReservasNoPagadas(){
    return centrodeportivo.lista_Reservas_Pendientes().stream().map(reservas -> mapper.aReservaDTO(reservas))
            .collect(Collectors.toList());
}
```

Ilustración 5.47 Método Ver Reservas No Pagadas

5.2.2. En el lado del Cliente:

- Componente App: Obtiene el token y la información del usuario desde el almacenamiento de la sesión del navegador a través de TokenStorageService. Entonces la barra de navegación se muestra según el estado de inicio de sesión y rol del usuario.

```
ngOnInit(){

  this.isLoggedIn = !!this.tokenStorageService.getToken();

  if(this.isLoggedIn){
    const user = this.tokenStorageService.getUser();
    this.roles = user.roles;
    this.rol = user.roles[0];
    this.username = user.username;
  }
}
```

Ilustración 5.48 ngOnInit() App

- Componentes Login y UsuariosAdd: contienen un formulario para el envío de datos (con soporte de validación de formularios). Utilizan TokenStorageService para comprobar el estado y AuthService para enviar las solicitudes de inicio de sesión/registro.
- UsuariosAdd: Como vemos a continuación, cada campo del formulario cuenta con soporte de validación. Todos los campos excepto rol deben ser obligatorios; Para DNI, email y teléfono deben cumplir con el patrón especificado y para el resto con una determinada longitud.

```
this.addForm = this.formBuilder.group({
    dni:['',[Validators.required,Validators.pattern("^[0-9]{8}[TRWAGMYFPDXBNJZSQVHLCKE]$")]],
    password: ['',[Validators.required,Validators.maxLength(9),Validators.minLength(4)]],
    email: ['',[Validators.required,Validators.pattern("^[a-z0-9._%+-]+@[a-z0-9.-]+\\.[a-z]{2,4}$")]],
    username:['',[Validators.required,Validators.minLength(5)]],
    role: [''],
    telefono: ['',[Validators.required,Validators.pattern("^((\\+91-?)|0)?[0-9]{9}$")]],
})
```

Ilustración 5.49 Formulario UsuariosAdd

```
this.authService.register(this.addForm.getRawValue()).subscribe(data => {

    this._router.navigate(['/login']);
    this.isSuccesful = true;
    this.isSignUpFailed = false;

}, (error) => {
    this.errorMessage = error.error.message;
    this.isSignUpFailed = true;
});
```

Ilustración 5.50 Método onSubmit() Clase UsuariosAdd

- Login: Igual que en el formulario para el registro, en el formulario para el Login también contamos con validación de datos. Ambos campos son obligatorios y además deben cumplir con las longitudes especificadas.

```
this.loginForm = this.formBuilder.group({
  username: ['',[Validators.required,Validators.minLength(5)]],
  password : ['',[Validators.required,Validators.maxLength(9),Validators.minLength(4)]]
})
```

Ilustración 5.51 Formulario Login

```
onSubmit() {
  this.authService.login(this.loginForm.getRawValue()).subscribe((data: any) => {
    this.tokenStorage.saveToken(data.accessToken);
    this.tokenStorage.saveUser(data);

    this.isLoginFailed = false;
    this.isLoggedIn = true;
    this.roles = this.tokenStorage.getUser().roles;
    if (this.primerInicioVar == true) {
      this._router.navigate(['perfil']).then(() => {
        window.location.reload();
      });

    } else if (this.primerInicioVar == false && this.roles[0] == "ROLE_ADMIN") {

      this._router.navigate(['/calendar']).then(() => {
        window.location.reload();
      });
    }
  }, error => {

    this.errorMessage = error.error.message;
    this.isLoginFailed = true;
  })
}
```

Ilustración 5.52 Método onSubmit() Clase Login

- TokenStorageService: gestiona el token y la información del usuario dentro del almacenamiento de la sesión del navegador. Para cerrar sesión únicamente tendremos que borrar este almacenamiento de sesión.

```
export class TokenStorageService{

    constructor(){
    }

    signOut():void{
        window.sessionStorage.clear();
    }

    public saveToken(token: string): void {
        window.sessionStorage.removeItem(TOKEN_KEY);
        window.sessionStorage.setItem(TOKEN_KEY, token);
      }

      public getToken(): string | null {
        return window.sessionStorage.getItem(TOKEN_KEY);
      }

      public saveUser(user: any): void {
        window.sessionStorage.removeItem(USER_KEY);
        window.sessionStorage.setItem(USER_KEY, JSON.stringify(user));
      }

      public getUser(): any {
        const user = window.sessionStorage.getItem(USER_KEY);
        if (user) {
          return JSON.parse(user);
        }
        return {};
      }
}
```

Ilustración 5.53 Clase TokenStorageService

- AuthService: utiliza HttpClient para realizar las peticiones HTTP Post de registro e inicio de sesión al Backend.

```
export class AuthService{

    public url:string;

    constructor(private http:HttpClient){
        this.url = GLOBAL.url;
    }

    login(form:any):Observable<any>{
        let direccion = this.url+'signin';
        return this.http.post(direccion,form,httpOptions);
    }

    register(form:any):Observable<any>{
        let direccion = this.url+'signup';
        return this.http.post(direccion,form,httpOptions);
    }

}
```

Ilustración 5.54 Clase AuthService

- AuthInterceptor: cuenta con el método intercept() para inspeccionar y transformar cada petición HTTP antes de ser enviada al servidor. Implementa HttpInterceptor y le añade al token la cabecera Authorization con el prefijo "Bearer".

```
const TOKEN_HEADER_KEY = 'Authorization';
@Injectable()
export class AuthInterceptor implements HttpInterceptor{
    constructor(private token: TokenStorageService){
    }

    intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
        let authReq = req;
        const token = this.token.getToken();
        if(token != null){
            authReq = req.clone({headers: req.headers.set(TOKEN_HEADER_KEY,'Bearer ' + token)});
        }
        return next.handle(authReq);
    }
}

    export const authInterceptorProviders = [
        { provide: HTTP_INTERCEPTORS, useClass: AuthInterceptor, multi: true }
      ];
```

Ilustración 5.55 Clase AuthInterceptor

- Componente Perfil: obtiene los datos del usuario desde el almacenamiento de la sesión.

```
ngOnInit(): void{
    console.log("perfil-component cargado correctamente");
    this.currentUser = this.token.getUser();
}
```

Ilustración 5.56 ngOnInit Clase Perfil

5.3. Guards Angular

Además de la seguridad implementada en el Backend la cual es la más importante y que anteriormente hemos mencionado, en el Frontend también será implementada haciendo uso de las Guards de Angular. Con esto vamos un paso más allá en cuanto a la seguridad de nuestra aplicación.

Las Guards de Angular nos permitirán otorgar o denegar el acceso a usuarios no autorizados a ciertas partes de la navegación.

Existen 4 tipos de Guards pero, en nuestro caso utilizaremos "CanActivate" para controlar si se puede activar una determinada ruta.

Lo primero que debemos de hacer es crear una clase Guard, en nuestro caso tenemos dos, una para los usuarios con rol 'User' y otra para 'Admin'. A continuación, vamos a mostrar la clase Guard para el usuario Admin:

```
canActivate(next: ActivatedRouteSnapshot,
  state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean{

  this.currentUser = this.token.getUser();

    if(this.token.getToken() != null) {
      if(this.currentUser.roles[0] == "ROLE_ADMIN"){
        return true;
      }else{
        window.alert('Acceso Denegado, No tienes permisos')
        this.router.navigate(['home'])
      }
    }else{
        window.alert('Acceso Denegado, es necesario iniciar sesión para acceder a la página requerida!')
        this.router.navigate(['login'])
    }
}
```

Ilustración 5.57 Clase Guard Admin

Como podemos ver en la imagen anterior, en el método “canActivate” que es donde debemos implementar nuestra lógica, comprobamos que un usuario está logueado, si no es así mostramos un mensaje indicando que es necesario iniciar sesión, si ha iniciado sesión, pero no es Administrador, mostramos otro mensaje indicado que no está autorizado para acceder a esa parte de la navegación. Por último, si es administrador devolvemos true y podrá acceder.

Una vez creada dicha clase debemos de configurar el enrutamiento modificando el archivo “app.routing.ts” mostrado a continuación:

```
//Reservas
{path: 'reservas',component:ReservaListComponent,canActivate : [GuardAdmin]},
{path: 'usuarios/:dniUsuario/reservas',component:ReservaListUsersComponent, canActivate : [GuardUser]},
{path: 'addReserva',component:ReservaAddComponent,canActivate : [GuardUser]},
{path: 'reservasPendientes', component: ReservasNoPagadas, canActivate : [GuardAdmin]},
```

Ilustración 5.58 app.routing.ts Guards

En este fichero para cada una de las rutas que queremos dotar de seguridad le asociaremos ya sea, la clase Guard anteriormente mencionada para el Administrador o para un usuario normal.

Por ejemplo, en nuestro fichero, el usuario administrador es el único que podrá visualizar el listado total de reservas y reservas pendientes. Por otro lado, para visualizar las reservas realizadas por el propio usuario y crearlas, tanto el usuario normal como el administrador podrán acceder, por ello en la clase Guard de un usuario normal permitimos tanto a dicho usuario normal como al administrador.

5.4. Pasarela de Pago

Para nuestra aplicación vamos a integrar diferentes métodos de pago, todo se hará del lado del cliente con Angular.

Contaremos con **PayPal**, **Sofort**, donde el pago se realiza directamente desde tu cuenta bancaria hasta la cuenta del comercio donde has efectuado la compra, sin necesidad de intermediarios, y **tarjeta de débito o crédito**.

5.4.1. Script

Lo primero que debemos de hacer es colocar el siguiente script en nuestra página "index.html" para que el objeto PayPal esté disponible cuando se necesite en el pago.

<script src="https://www.paypal.com/sdk/js?client-id=YOUR_CLIENT_ID"></script>

Debemos de sustituir "YOUR_CLIENT_ID" por nuestro ID de cliente y la divisa debe ser Europa (EUR).

Para obtener nuestro ID de cliente, debemos de crear en Paypal una nueva App.

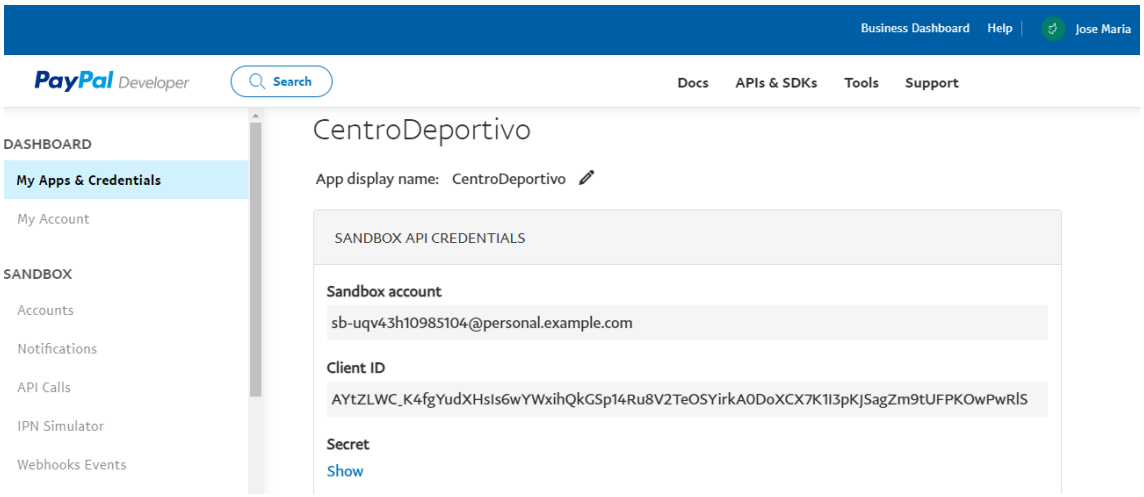


Ilustración 5.59 Client ID

Nuestro script queda de la siguiente manera:

```
<script
  src="https://www.paypal.com/sdk/js?client-id=AYtZLWC_K4fgYudXHsIs6wYWxihQkGSp14Ru8V2TeOSYirkA0DoXCX7K1I3pKJSagZm9tUFPKOwPwRlS&currency=EUR">
</script>
```

Ilustración 5.60 Script PayPal

5.4.2. Componente PayPal

Declararemos una variable PayPal, un objeto @ViewChield para monitorear un elemento Html y a este añadirle toda la lógica de la pasarela de pagos y un objeto Pista con ciertos campos como el precio y descripción.

Ahora con la variable PayPal creada anteriormente podemos renderizar los botones.

Con el método “createOrder()” creamos la orden pasándole los parámetros necesarios como descripción de la pista y precio del objeto creado anteriormente.

Para validar que el pago ha llegado hacemos uso del método “onApprove()” capturando el evento de creación de orden. De esta forma, comprobamos que se ha realizado el pago correctamente y si es así, actualizamos la reserva a ‘Pagada’ y enviamos el email de confirmación al usuario.

```
ngOnInit() {
  this.idReserva = this.datas
  paypal
    .Buttons({
      createOrder: (data, actions) => {
        return actions.order.create({
          purchase_units: [
            {
              description: this.pistA.descripcion,
              amount: {
                currency_code: 'EUR',
                value: this.pistA.precio
              }
            }
          ]
        })
      },
      onApprove: async (data, actions) => {
        const order = await actions.order.capture();

        if (order.status == 'COMPLETED') {

          this.reserva_Api.getReserva(this.idReserva).subscribe(response => {
            this.reserva = response;
            this.reserva_Api.editReserva(this.reserva.id, order.id).subscribe(data => {
              this.enviaEmail(this.reserva.idPista);
            }, error => {
              console.log(error)
            })
          }
            , error => {
              console.log(error)
            })
        }
      },
      onError: err => {
        console.log(err);
      },
    })

    .render(this.paypalElement.nativeElement);
}
```

Ilustración 5.61 ngOnInit() Componente PayPal

De esta forma ya contamos con nuestros diferentes métodos de pago, todos ellos totalmente funcionales. Además, podemos comprobar que nuestros pagos se realizan correctamente desde la aplicación de PayPal.

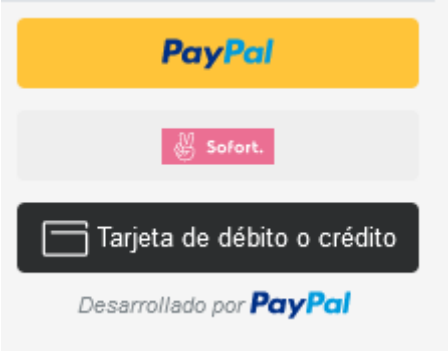


Ilustración 5.62 Componente Paypal

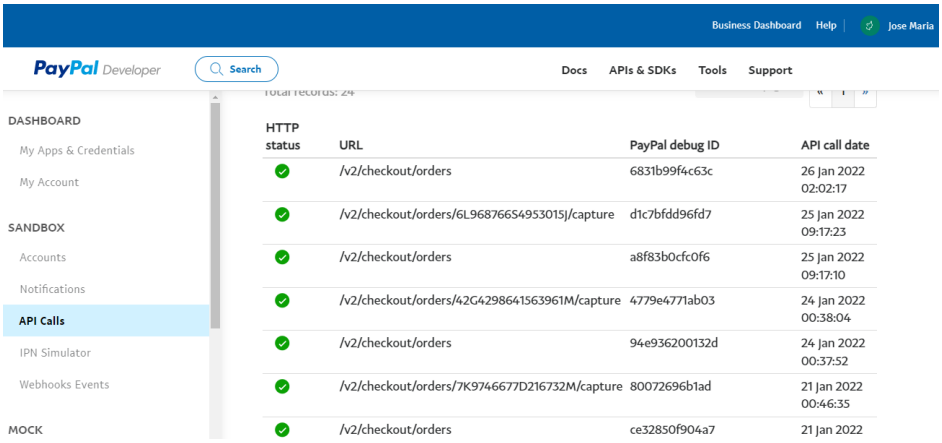


Ilustración 5.63 Creación Pagos

5.5. API RestFul

Disponer de una API REST bien documentada es fundamental ya que facilita el proceso de aprendizaje sobre el nuevo entorno que se desarrollará basándose en nuestro servicio API. Para ello, es importante dar a conocer los endpoints que dispone nuestra API, métodos HTTP, parámetros, esquema de los cuerpos de la petición y de los resultados y códigos de retorno devueltos.

Para la generación de la documentación utilizaremos Swagger. Esta herramienta analiza todo el proyecto y auto documenta todos los servicios Rest con los que cuenta.

Para poder hacer uso de esta únicamente hemos tenido que incluir las siguientes librerías en nuestro fichero “pom.xml”:

```xml
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.9.2</version>
</dependency>

<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.9.2</version>
</dependency>
```

Ilustración 5.64 Librerias Swagger

Por último, creamos el siguiente fichero de configuración, donde indicaremos el paquete que debe analizar:

```java
@Configuration
@EnableSwagger2
public class SwaggerConfig {

    @Bean
    public Docket apiDocker(){
        return new Docket(DocumentationType.SWAGGER_2)
                .select()
                .apis(RequestHandlerSelectors.basePackage("es.ujaen.TFG.beans"))
                .paths(PathSelectors.any())
                .build()
                .apiInfo(getApiInfo());
    }

    private ApiInfo getApiInfo() {
        return new ApiInfo(
                title: "API Centro Deportivo",
                description: "Service API Description",
                version: "1.0",
                termsOfServiceUrl: "http://codmind.com/terms",
                new Contact( name: "Codmind",  url: "https://codmind.com",  email: "apis@codmind.com"),
                license: "LICENSE",
                licenseUrl: "LICENSE URL",
                Collections.emptyList()
        );
    }
```

Ilustración 5.65 SwaggerConfig

A continuación, mostraremos un resumen con el título, URL, método y descripción, para más adelante en *Apéndice A: API Restful* mostrar ya toda la documentación de la API generada con más detalle.

Título	URL	Método	Descripción
Ingresa Configuración Inicial	/centrodeportivo/configuracionInicial	POST	Introduce la información acerca del centro deportivo
Visualiza Información Inicial	/centrodeportivo/configuracionInicial	GET	Visualiza la información del centro deportivo
Ingresa Días No Reservables	/centrodeportivo/diasNS	POST	Ingresa los días que no se pueden

			reservas las instalaciones
Listado Días No Reservables	/centrodeportivo/diasNS	GET	Visualiza un listado con los días que no se pueden reservas las instalaciones
Elimina Días No Reservables	/centrodeportivo/diasNS	DELETE	Reestablece los días no reservables
Envía Email	/centrodeportivo/email	POST	Envía un email de consulta al centro deportivo
Envía Confirmación Reserva	/centrodeportivo/emailConfirmacion	POST	Envía un email de confirmación de la reserva con todos los datos al usuario
Lista Imágenes	/centrodeportivo/files	GET	Visualiza el listado de todas las imágenes
Visualiza Imagen	/centrodeportivo/files/{id}	GET	Visualiza una determinada imagen
Sube Imagen	/centrodeportivo/upload	POST	Almacena una imagen
Actualiza Primer Inicio	/centrodeportivo/primerInicio	PUT	Establece como finalizada la configuración de la aplicación
Consulta Primer Inicio	/centrodeportivo/primerInicio	GET	Consulta si se ha realizado la configuración de la aplicación
Ingresa Franja Horaria	/centrodeportivo/horario	POST	Ingresa el horario de una pista
Actualiza Estado Franja Horaria	/centrodeportivo/franjaEstado/{id}	PUT	Activa/Desactiva una franja horaria
Visualiza Franja	/centrodeportivo/horario/{id}	GET	Visualiza una franja horaria
Lista Franjas Horarias	/centrodeportivo/horarios	GET	Visualiza un listado de todas las

			franjas horarias
Lista Franjas Horarias Disponibles	/centrodeportivo/horarios/{idPista}/{fecha}	GET	Visualiza las franjas horarias disponibles para alquilar una pista
Ingresa Pista	/centrodeportivo/pista	POST	Ingresa una pista nueva en la aplicación
Visualiza Pista	/centrodeportivo/pista/{id}	GET	Visualiza los datos de una pista
Actualiza Datos Pista	/centrodeportivo/pista/{id}	PUT	Modifica los datos de una pista
Visualiza Horario Pista	/centrodeportivo/pista/{id}/horario	GET	Visualiza el horario de una determinada pista
Lista Pistas	/centrodeportivo/pistas	GET	Visualiza todas las pistas
Lista Pistas Activas	/centrodeportivo/pistasActivas	GET	Visualiza las pistas que se encuentran activas
Crea Reserva	/centrodeportivo/reserva	POST	Realiza una reserva de una pista
Ver Reserva	/centrodeportivo/reserva/{id}	GET	Visualiza los datos de una reserva
Cancela Reserva No Pagada	/centrodeportivo/reserva/{id}	DELETE	Cancela una reserva pendiente de pago
Actualiza Reserva Pendiente	/centrodeportivo/reservaPendiente/{idReserva}/{idPago}	PUT	Actualiza a pagada una reserva
Lista Reservas	/centrodeportivo/reservas	GET	Visualiza un listado de todas las reservas
Lista Reservas Pendientes	/centrodeportivo/reservasPendientes	GET	Visualiza un listado de las reservas pendientes de pago
Inicia Sesión	/centrodeportivo/signin	POST	Operación para iniciar sesión en la aplicación

Registra Usuario	/centrodeportivo/signup	POST	Operación para registrarse en la aplicación
Visualiza Usuario	/centrodeportivo/usuario/{dni}	GET	Visualiza los datos de un usuario
Actualiza Usuario	/centrodeportivo/usuario/{dni}	PUT	Actualiza los datos de un usuario
Actualiza Estado Usuario	/centrodeportivo/usuarioEstado/{dni}	PUT	Activa/Desactiva a un usuario
Lista Usuarios	/centrodeportivo/usuarios	GET	Visualiza un listado de los usuarios registrados
Visualiza Mis Reservas	/centrodeportivo/usuarios/{dni}/reservas	GET	Visualiza las reservas realizadas por un usuario

Tabla 5.67 API RestFul

5.6. Backend

En este apartado expondremos la estructuración de nuestro servidor mostrando los diferentes paquetes y mencionando alguno de los archivos que lo forman. Con dichos paquetes y ficheros obtendremos la estructura necesaria para llevar a cabo la funcionalidad relacionada con el servidor.

Los paquetes que forman nuestro servidor son los siguientes:

- **Entidades**. Contiene elementos importantes del dominio(entidades) que mantienen una identidad única y permanente que los diferencia del resto. Entre los archivos que forman este paquete, encontramos, por ejemplo, el archivo “Reserva.java” el cual representa a la entidad Reserva, “Usuario.java”, “Pista.java”, etc.
- **EntidadesDTO**. Está formado por archivos DTO (Data Transfer Object), aquellos utilizados para la transmisión y modificación de datos entre las diferentes capas. Al igual que en el paquete entidades, aquí algunos de los archivos que encontramos son: “UsuarioDTO.java”, “ReservaDTO.java”, etc.
- **Beans**. Estamos ante el paquete más importante de nuestro servidor. Este está formado por los Servicios y el fichero de configuración “SwaggerConfig.java” para la documentación del API. En cuanto a los servicios

encontramos, “Servicio_CentroDeportivo.java” el cual cuenta con la mayoría de métodos para la funcionalidad del servidor, los servicios “EmailService.java” para el correo electrónico y “FileStorageService.java” para las imágenes de las instalaciones y “ServicioRestAPI.java” con los métodos Rest.

- **Repositorios**. En él encontramos todos los repositorios que necesitamos como, “Repositorio_Reserva” donde encontramos métodos para consultar la existencia de una reserva a realizar, etc.
- **Security**. En este paquete encontramos el servicio “ServicioSeguridad.java” donde implementamos las políticas de seguridad y el resto de ficheros mencionados en el apartado *5.2*.
- **Aplicación**. Encontramos un único archivo, “CentroDeportivoApp.java” con el que lanzamos la aplicación.
- **Util**. Paquete donde contamos con ficheros para la validez de un DNI y el formato que deben tener los mensajes o respuestas que recibamos.
- **Tests**. Formado por los archivos con los cuales comprobamos el correcto funcionamiento acerca de las funcionalidades de nuestro servidor, por ejemplo, en “ReservaTest.java” realizamos pruebas acerca de dichas reservas.

5.7. FrontEnd

Para la creación de la estructura del Frontend hemos hecho uso de Angular CLI comentando anteriormente.

El directorio más importante y donde se encuentra la implementación de los componentes creados y usados para dotar al cliente de la funcionalidad deseada es /src/app. En él encontramos:

- **Models**. Formado por aquellas interfaces que tendrá cada componte como por ejemplo, “pista.interface.ts”, “usuario.interface.ts”, etc.
- **Services**. En dicho paquete encontramos un servicio por componente creado, por ejemplo, “reserva.service.ts”, “admin.service.ts”, etc. Cada uno de estos contiene la funcionalidad para su respectivo componente.
- **Componentes**. Cuenta con todos los componentes que hemos creado para nuestro Frontend. Cada uno de ellos está formado por dos ficheros con extensiones, “.ts” donde se almacena todas las funciones relacionadas con la

lógica del componente, y ".html" que almacena las funciones que proporcionan estilos al componente. Entre ellos encontramos componentes para la pestaña del Administrador, para las Instalaciones, Perfil, Login, etc.

Entre algunos de los ficheros más usados e importantes encontramos:

- **app.routing.ts**. Fichero para la configuración de las rutas de nuestra aplicación.
- **app.component**. Es el componente principal, este cuenta con la barra de navegación, el encabezado y el pie.
- **app.module**. En él importamos los componentes creados y librerías.
- **styles.css**. Archivo donde definimos los estilos que le daremos a cada uno de nuestros componentes.
- **index.html**. Archivo de la página principal del proyecto, a partir de este se cargará el resto de la aplicación.

5.8. Docker

Docker [29] es una plataforma software que permite crear y empaquetar una aplicación junto con sus dependencias y librerías en un contenedor con el fin de ejecutarse en otras máquinas. De esta manera, son independientes del sistema operativo o versiones de librerías y dependencias, garantizando así el aislamiento perfecto de dicho software.

Para nuestro proyecto como sabemos implementaremos una aplicación completa, Base de datos, Backend y Frontend. De esta forma en vez de iniciar cada uno de ellos por separado, escribiremos un único archivo YAML para desplegarlos todos a la vez.

Empezaremos por la base de datos, para esta, únicamente debemos de usar su imagen con el siguiente comando:

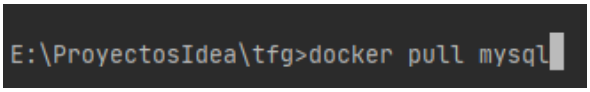


Ilustración 5.66 Docker BD

Para el Backend, crearemos su fichero Docker con la siguiente información:

```
FROM maven:3.8.4-jdk-8 AS build
COPY src /home/app/src
copy pom.xml /home/app
RUN mvn -f /home/app/pom.xml clean package

FROM openjdk:8-alpine
COPY --from=build "home/app/target/tfg-1.0-SNAPSHOT.jar" "app.jar"
EXPOSE 8080
ENTRYPOINT ["java","-jar","app.jar"]
```

Ilustración 5.67 Docker File Backend

En la primera parte del fichero, creamos un contenedor de la imagen Maven llamado build, copiamos la carpeta src que contiene el código de nuestro proyecto en “home/app/src” y lo mismo con el archivo pom.xml. Por último, se compila.

En la segunda parte de dicho fichero, creamos otro contenedor para la imagen openjdk, copiamos él “.jar” desde el contenedor build, exponemos el puerto 8080 y procedemos a hacer el deploy dentro del contendor.

Antes de crear la imagen de este, debemos de conectar el contenedor con el deploy de la aplicación al contenedor que contenga la Base de datos MYSQL, para ello, modificaremos en el fichero “properties” la línea donde indicamos la url de la base de datos, cambiándola por el nombre del contenedor MYSQL.

```
spring.jpa.hibernate.ddl-auto= update
spring.datasource.url=jdbc:mysql://myapp-mysql:3306/centrodeportivo?allowP
spring.datasource.driver-class-name = com.mysql.cj.jdbc.Driver
spring.datasource.username = root
spring.datasource.password = Dae2020
```

Ilustración 5.68 Properties File

Lo siguiente y ultimo será crear la imagen ejecutando el siguiente comando:

```
E:\ProyectosIdea\tfg>docker build -t backendimage .
```

Ilustración 5.69 Comando Crear Imagen Backend

Para el caso del Frontend, lo primero será crear un archivo de configuración para Nginx, para así alojar la compilación angular dentro del contenedor.

Como podemos ver a continuación, le indicamos que escuche en el puerto 80, dónde se encuentran los archivos y las cors.

```
nginx.conf
worker_processes  1;

events {
    worker_connections  1024;
}

http {
    server {
        listen 80;
        server_name  localhost;
        root   /usr/share/nginx/html;
        index  index.html index.htm;
        include /etc/nginx/mime.types;
        gzip on;
        gzip_min_length 1000;
        gzip_proxied expired no-cache no-store private auth;
        gzip_types text/plain text/css application/json application/javascript application/x-javascript text/xml

        location / {
            try_files $uri $uri/ /index.html;

            if ($request_method = 'OPTIONS') {
            add_header 'Access-Control-Max-Age' 1728000;
            add_header 'Access-Control-Allow-Origin' '*';
            add_header 'Access-Control-Allow-Headers' 'Authorization,Accept,Origin,DNT,X-CustomHeader,Keep-Alive,
            X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Content-Range,Range';
            add_header 'Access-Control-Allow-Methods' 'GET,POST,OPTIONS,PUT,DELETE,PATCH';
            add_header 'Content-Type' 'application/json';
            add_header 'Content-Length' 0;
            return 204;
            }

            add_header 'Access-Control-Allow-Origin' '*';
            add_header 'Access-Control-Allow-Headers' 'Authorization,Accept,Origin,DNT,X-CustomHeader,Keep-Alive,
            X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Content-Range,Range';
            add_header 'Access-Control-Allow-Methods' 'GET,POST,OPTIONS,PUT,DELETE,PATCH';

        }
    }
}
```

Ilustración 5.70 File nginx.conf

Ahora ya con el archivo “nginx.conf”, creamos también su fichero Docker:

```
Dockerfile > ...
#Primera Etapa
FROM node:14-alpine as build-step
RUN mkdir -p /app

WORKDIR /app
COPY package.json /app

RUN npm install

COPY . /app

RUN npm run build --prod

#Segunda Etapa
FROM nginx:1.17.1-alpine
COPY --from=build-step /app/dist/centroDeportivoCLI/ /usr/share/nginx/html
COPY nginx.conf /etc/nginx/nginx.conf
```

Ilustración 5.71 Docker File Frontend

Al igual que en el Backend, aquí utilizamos un contenedor de la imagen de Node al cual llamamos build-step, copiamos el código de la aplicación en “/app”, instalamos las dependencias del archivo “package.json” y creamos los archivos de producción usando dicha imagen. Por último, en la parte final del archivo, usamos la imagen de

Nginx para poder desplegar la aplicación en este, copiamos los archivos en “/usr/share/ngnix/html” y su archivo de configuración anteriormente creado.

Una vez ya tengamos las tres imágenes necesarias, nos queda crear el archivo docker-compose.yml

En este llamamos a los tres servicios, Base de datos, Backend y Frontend.

Para la base de datos, indicamos el nombre de su imagen, las variables de entorno, el atributo volumen, el puerto y la red.

Para el Backend, su imagen, puerto, variables de entorno, la red que debe ser la misma y, que depende de la base de datos, es decir, este debe comenzar una vez la base de datos esté lista. Por último, para el Frontend, imagen, red que también debe ser la misma que para los demás, el puerto y que depende, en este caso del Backend.

```yaml
version: '3'

services:

  myapp-mysql:
    image: mysql:latest
    environment:
      - MYSQL_ROOT_PASSWORD=Dae2020
      - MYSQL_DATABASE=centrodeportivo
      - MYSQL_PASSWORD=Dae2020
    cap_add:
      - SYS_NICE  # CAP_SYS_NICE
    ports:
      - 3306:3306
    volumes:
      - /home/hans/Escritorio/git_proyect/bd_spring_docker:/var/lib/mysql
    networks:
      - employee-mysql

  myapp-main:
    image: backendimage:latest
    networks:
      - employee-mysql
    depends_on:
      - myapp-mysql
    ports:
      - 8080:8080
    environment:
      - DATABASE_HOST=myapp-mysql
      - DATABASE_USER=root
      - DATABASE_PASSWORD=Dae2020
      - DATABASE_NAME=centrodeportivo
      - DATABASE_PORT=3306
  frontend:
    image: frontendimage:latest
    networks:
      - employee-mysql
    depends_on:
      - myapp-main
    ports:
      - 4200:80

networks:
  employee-mysql:
```

Ilustración 5.72 File Docker-compose

Para lanzar este fichero usamos el siguiente comando y cómo podemos ver los tres servicios se han creado y se encuentran activos.

```
E:\ProyectosIdea\tfg>docker-compose up -d
[+] Running 4/4
 - Network tfg_employee-mysql   Created
 - Container tfg-myapp-mysql-1  Started
 - Container tfg-myapp-main-1   Started
 - Container tfg-frontend-1     Started
```

Ilustración 5.73 Comando Docker-compose

Ahora que ya tenemos el contenedor con todos los servicios funcionando, lo siguiente será subir las imágenes creadas de cada servicio al repositorio Docker Hub.

Para ello, crearemos las etiquetas para las imágenes en cuestión con el nombre de nuestro repositorio Docker y nombre de usuario:

```
docker tag backendimage jmpc0016/tfg:backendimage
```

Ilustración 5.74 Comando etiqueta imagen Backend

Por último, la subimos al repositorio con la etiqueta creada. Por ejemplo, para la imagen de nuestro Backend usaremos el siguiente comando:

```
docker push jmpc0016/tfg:backendimage
```

Ilustración 5.75 Comando subir imagen Backend

Como podemos ver se ha creado en nuestro repositorio, además, nos aparece el comando para descargar la imagen para usarla.

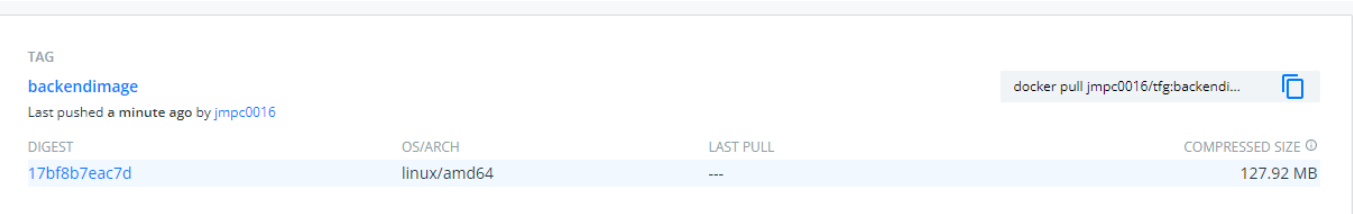


Ilustración 5.76 Imagen Backend Repositorio

Al igual que para el Backend, lo haremos para el Frontend. Para el caso de MYSQL no hace falta ya que en el repositorio de Docker ya existe su imagen.

6. PRUEBAS

Una vez terminada la implementación de nuestra aplicación, pasamos a la fase de pruebas donde deberemos realizar un testeo del software implementado.

Para realizar el testeo de nuestra aplicación entregaremos un cuestionario a un grupo de personas junto con nuestra aplicación web corriendo. Contamos con dos cuestionarios diferentes, uno para usuarios que pueden ser administrador y otro para usuarios normales.

6.1. Administrador

Cuestionario

Las preguntas son las siguientes:

Cuestión 1. Inicie la aplicación por primera vez y realice la configuración y personalización. ¿Es sencillo montar la aplicación?

Cuestión 2. ¿El panel de Administrador es intuitivo?

Cuestión 3. Desactive a un usuario, instalación y franja horaria.

Cuestión 4. Cancele una reserva pendiente de pago

Cuestión 5. Valore la gestión de la aplicación.

Resultados

	Persona 1	Persona 2	Persona 3	Persona 4	Persona 5
Cuestión 1	8	8	8,5	7,75	9
Cuestión 2	9	9	9,25	8,75	8,5
Cuestión 3	9	9	7,75	8	8,5
Cuestión 4	9,5	9	9	8,5	9,25
Cuestión 5	9,5	9	9,25	9	9
Media Total	**8,76**				

Tabla 6.68 Resultados Cuestionario Administrador

Los usuarios administradores han calificado la aplicación con un 8,76. Los resultados han sido satisfactorios, aunque siempre existe la posibilidad de poder mejorar ciertos aspectos.

6.2. Usuario

Cuestionario

Las preguntas y acciones que deberán responder y realizar los usuarios son las siguientes:

Cuestión 1. Regístrese en la aplicación e inicie sesión.

Cuestión 2. Realice una reserva.

Cuestión 3. Valore la confirmación de la reserva por correo

Cuestión 4. Envíe una consulta al centro deportivo a través del formulario de contacto.

Cuestión 5. Visualice sus reservas realizadas

Cuestión 6. Visualice las instalaciones junto con sus horarios

Cuestión 7. ¿La navegación por la web es sencilla?

Resultados

	Persona 1	Persona 2	Persona 3	Persona 4	Persona 5
Cuestión 1	9	9	8,75	9	8,5
Cuestión 2	9	8,5	9	7,5	8,5
Cuestión 3	9,5	8	9	9	9
Cuestión 4	9	9	8,5	8,75	8,5
Cuestión 5	9	8	8,5	8	9
Cuestión 6	9	9	9	9	9
Cuestión 7	9,5	10	8,75	9	8,75
Media total	**8,81428571**				

Tabla 6.69 Resultados Cuestionario Usuario

Los usuarios consideran que la aplicación es intuitiva y que tiene todo lo necesario para un centro deportivo, obteniendo de esta forma un 8,81.

7. CONCLUSIONES

Después de haber finalizado el proyecto, es hora de reflexionar sobre el trabajo realizado.

Como hemos comentado en capítulos anteriores, la tecnología cada vez va evolucionando más y por lo tanto con el paso de los años se irá perdiendo la gestión y reservas de centros deportivos a papel, con esto queremos decir que, cada vez más existirán más aplicaciones de este tipo. Por ello, este proyecto busca ayudar a facilitar estas tareas, tanto por parte de los administradores como de los clientes.

Gracias al trabajo realizado, he conseguido profundizar en los conocimientos de metodologías y tecnologías para el desarrollo de aplicaciones web. Cabe destacar que

me he encontrado con diversos problemas que he sido capaz de solventar y de esta forma ampliar aún más mis conocimientos.

En cuanto a los resultados obtenidos, han sido satisfactorios ya que se han resuelto los objetivos y complicaciones que fueron surgiendo, obteniendo de esta forma una aplicación que cumple con todos los requerimientos marcados.

Para el trabajo futuro, existen múltiples ampliaciones, como puede ser el caso de un sistema de pago por recompensas, donde un usuario después de realizar ciertas reservas pueda reservar gratuitamente, filtros o estadísticas de reservas realizadas en la última semana, comentarios por parte de los clientes en las pistas, etc.

Por último, quiero señalar que ha sido un gran trabajo del cual me siento muy orgullo y he intentado demostrar los conocimientos adquiridos.

Bibliografía

[1] A. E. B. O. d. Estado, «Instalaciones y espacios deportivos.,» de *Anuario de Estadísticas Deportivas 2021*, 2021.

[2] A. Muñoz Vita, «Cincodias Elpais,» 2020. [En línea]. Available: https://cincodias.elpais.com/cincodias/2020/11/23/fortunas/1606151043_866311.html#:~:text=La%20industria%20del%20deporte%20aporta,del%20Consejo%20Superior%20de%20Deportes..

[3] D. Sillari, «Comunicae,» 2017. [En línea]. Available: https://www.comunicae.es/nota/las-reservas-de-pistas-deportivas-a-traves-de-las-apps-crece-en-un-50-0-1189094/.

[4] R24h, «reservas24h,» [En línea]. Available: https://reservas24h.es/Nuestra-aplicacion/software-gestion-instalaciones-deportivas/n39.aspx. [Último acceso: Octubre 2021].

[5] ZCENTER, «gestion-centros-deportivos,» [En línea]. Available: https://www.gestion-centros-deportivos.es/software-de-gestion-deportiva/. [Último acceso: Octubre 2021].

[6] DecideSoluciones, «decidesoluciones,» 2020. [En línea]. Available: https://decidesoluciones.es/metodologias-proyectos-desarrollo-software/.

[7] A. J. Ruíz Rueda, «Apuntes de Desarrollo de Aplicaciones Empresariales,» 2020/21.

[8] D. Ideas, «Dos Ideas,» [En línea]. Available: https://dosideas.com/cursos/course/introduccion-al-desarrollo-en-java-con-spring-

framework-y-spring-boot/introduccion-a-hibernate. [Último acceso: Octubre 2021].

[9] E. Geek, «ifgeekthen,» 2019. [En línea]. Available: https://ifgeekthen.nttdata.com/es/que-es-java-hibernate-por-que-usarlo.

[10] Edix, «Edix Digital Workers,» 2021. [En línea]. Available: https://www.edix.com/es/instituto/framework/#:~:text=Un%20framework%20es%20un%20esquema,organizaci%C3%B3n%20y%20desarrollo%20de%20software..

[11] Arimetrics, «Arimetrics,» [En línea]. Available: https://www.arimetrics.com/glosario-digital/framework. [Último acceso: Noviembre 2021].

[12] Nestrategia, «Nestrategia,» [En línea]. Available: https://nestrategia.com/desarrollo-web-back-end-front-end/. [Último acceso: Octubre 2021].

[13] admin, «javadesde0,» 2019. [En línea]. Available: http://javadesde0.com/diferencias-entre-spring-y-spring-boot/.

[14] J. Aguilar, «softwareevolutivo,» 2017. [En línea]. Available: http://softwareevolutivo.com.ec/espanol-desarrollo-de-aplicaciones-web-con-spring-boot/.

[15] H. Dhaduk, «Simform,» 2021. [En línea]. Available: https://www.simform.com/best-frontend-frameworks/.

[16] A. Basalo, «Desarrollo Web,» 2020. [En línea]. Available: https://desarrolloweb.com/articulos/angular-cli.html.

[17] NodeJS, «nodejs,» [En línea]. Available: https://nodejs.org/es/about/. [Último acceso: Octubre 2021].

[18] Y. Muradas, «OpenWebinars,» 2019. [En línea]. Available: https://openwebinars.net/blog/que-es-node-package-manager/.

[19] J. J. Fernández, «tooltyp,» 2015. [En línea]. Available: https://www.tooltyp.com/8-razones-por-las-que-debes-utilizar-bootstrap-para-tu-web/.

[20] SaraClip, «saraclip,» 2017. [En línea]. Available: https://www.saraclip.com/requerimientos-de-un-proyecto/.

[21] EvaluandoSoftware, «evaluandosoftware,» 2022. [En línea]. Available: https://www.evaluandosoftware.com/gestion-requerimientos-proyecto-software-empresarial/.

[22] rvillarroel16, «PORQUERIA, Requisitos Funcionales y No Funcionales,» 2017. [En línea]. Available: https://ingenieriadesoftwareutmachala.wordpress.com/2017/01/20/requerimientos-funcionales-y-no-funcionales/.

[23] midder, «Assembla,» 2013. [En línea]. Available: https://app.assembla.com/wiki/show/iw2013-tavira/Requisitos_no_funcionales.

[24] G. T. Manuel, «GESTION DE PROYECTOS INFORMÁTICOS Metodología Scrum,» Octubre.

[25] BeAgileMyFriend, «beagilemyfriend,» [En línea]. Available: https://beagilemyfriend.com/en-que-consiste-el-sprint-backlog/. [Último acceso: Octubre 2021].

[26] A. E. B. O. d. Estado, «Capítulo V Salarios y estructura retributiva Sección Segunda,» de *BOLETÍN OFICIAL DEL ESTADO*, 2021, p. 57159.

[27] J. I. G. E. L. Alfonso Ureña López, «Apuntes de Tema 4. Representación de Requisitos,» de *Software e Ingeniería del Software*, 2017/18.

[28] Bezkoder, «bezkoder,» 2021. [En línea]. Available: https://www.bezkoder.com/angular-12-spring-boot-jwt-auth/.

[29] A. Blanch, «Arsys,» 2019. [En línea]. Available: https://www.arsys.es/blog/docker-ventajas-contenedores.

Apéndice A: API Restful

1. Configuración de la Aplicación

Ingresa configuración inicial

```
POST /centrodeportivo/configuracionInicial
```

Descripción

Operación para introducir la información acerca del centro deportivo

Parámetros

Type	Name	Descripción	Schema
Body	**config** *required*	config	ConfiguracionGeneralDTO

Responses

HTTP Code	Descripción	Schema
201	**Created**	ConfiguracionGeneralDTO
400	**Bad Request**. El parámetro se	No Content

	encuentra vacío	
409	**Conflict**. La Configuración Inicial no se ha creado	No Content

Visualiza información inicial

```
GET /centrodeportivo/configuracionInicial
```

Descripción

Operación para visualizar la información del centro deportivo

Responses

HTTP Code	Descripción	Schema
200	**OK**	ConfiguracionGeneralDTO
404	**Not Found**. No se encuentra el contenido solicitado	No Content

Actualiza Primer Inicio

```
PUT /centrodeportivo/primerInicio
```

Descripción

Operación para establecer como finalizada la configuración de la aplicación

Responses

HTTP Code	Descripción	Schema
200	**OK**	ConfiguracionGeneralDTO
404	**Not Found**. No se encuentra el contenido solicitado	No Contest

Consulta Primer Inicio

```
GET /centrodeportivo/primerInicio
```

Descripción

Operación para conocer si se ha realizado la configuración de la aplicación

Responses

HTTP Code	Descripción	Schema
200	**OK**	boolean
404	**Not Found**. No se ha encontrado el contenido solicitado	No Content

2. Días no Reservables

Ingresa días no reservables

```
POST /centrodeportivo/diasNS
```

Descripción

Operación para que el administrador ingrese los días que no se pueden reservar

Parámetros

Type	Name	Descripción	Schema
Body	**dias** *required*	dias	<DiasNoSeleccionablesDTO> array

Responses

HTTP Code	Descripción	Schema
201	**Created**	DiasNoSeleccionablesDTO
400	**Bad Request**. El parámetro se encuentra vacío	No Content
401	**Unauthorized**. El usuario debe iniciar sesión	No Content
403	**Forbidden**. El usuario debe ser administrador	No Content
409	**Conflict**. Los días no reservables no se han creado	No Content
500	**Internal Server Error**. El día ya se encuentra introducido	No Content

Listado Días no reservables

```
GET /centrodeportivo/diasNS
```

Descripción

Operación para que se visualice un listado con los días que no se pueden reservar las pistas

Responses

HTTP Code	Descripción	Schema
200	**OK**	<DiasNoSeleccionablesDTO> array
404	**Not Found.** No se encuentra el contenido solicitado	No Content

Elimina días no reservables

```
DELETE /centrodeportivo/diasNS
```

Descripción

Operación para el administrador reestablezca los días no reservables

Responses

HTTP Code	Descripción	Schema
200	**OK**	ResponseEntity
400	**Bad Request**. El parámetro se encuentra vacío	No Content
401	**Unauthorized**. El usuario debe iniciar sesión	No Content
403	**Forbidden**. El usuario debe ser administrador	No Content
404	**Not Found**. No se encuentra el contenido solicitado	No Content

3. Email

Envía Email

```
POST /centrodeportivo/email
```

Descripción

Operación para que un usuario envíe un email de consulta al centro deportivo

Parámetros

Type	Name	Descripción	Schema
Body	**formulario** *required*	formulario	FormularioContactoDTO

Responses

HTTP Code	Descripción	Schema
201	**Created**	No Content
400	**Bad Request**. El email se encuentra vacío	No Content
404	**Not Found**. No se encuentra el contenido solicitado	No Content

Envía Confirmación Reserva

POST /centrodeportivo/emailConfirmacion

Descripción

Operación para enviar un email de confirmación de la reserva con todos sus datos al usuario

Parámetros

Type	Name	Descripción	Schema
Body	**conf** *required*	conf	confirmacionReserva

Responses

HTTP Code	Descripción	Schema
201	**Created**	No Content
401	**Unauthorized**. El usuario debe iniciar sesión	No Content
400	**Bad Request**. El parámetro se encuentra vacío	No Content
404	**Not Found**. No se encuentra el contenido solicitado	No Content

4. Imágenes

Lista Imágenes

GET /centrodeportivo/files

Descripción

Operación para visualizar el listado de todas las imágenes en la aplicación

Responses

HTTP Code	Descripción	Schema
200	**OK**	<FileDTO> array
404	**Not Found**. No se encuentra el contenido solicitado	No Content

Visualiza Imagen

GET /centrodeportivo/files/{id}

Descripción

Operación para visualizar una determinada imagen en la aplicación

Parámetros

Type	Name	Descripción	Schema
Path	**id** *required*	id	string

Responses

HTTP Code	Descripción	Schema
200	**OK**	File
400	**Bad Request**. El id se encuentra vacío	No content
404	**Not Found**. No se encuentra el contenido solicitado	No Content

Sube Imagen

POST /centrodeportivo/upload

Descripción

Operación para almacenar una imagen

Parámetros

Type	Name	Descripción	Schema
FormData	**file** *required*	file	file

Responses

HTTP Code	Descripción	Schema
201	**Created**	No Content
404	**Not Found**. No se encuentra el contenido solicitado	No Content
500	**Internal Server Error**. El tamaño de la imagen excede el máximo permitido	No Content

5. Franjas Horarias

Ingresa Franja Horaria

POST /centrodeportivo/horario

Descripción

Operación para que el administrador ingrese el horario de una pista

Parámetros

Type	Name	Descripción	Schema
Body	**horario** *required*	horario	FranjaDTO

Responses

HTTP Code	Descripción	Schema
201	**Created**	No Content
400	**Bad Request**. El parámetro se encuentra vacío	No Content
401	**Unauthorized**. El usuario debe iniciar sesión	No Content
403	**Forbidden**. El usuario debe ser administrador	No Content

500	**Internal Server Error**. El horario ya se encuentra introducido o la pista no existe	No Content

Actualiza Estado Franja Horaria

PUT /centrodeportivo/franjaEstado/{id}

Descripción

Operación para activar/desactivar una franja horaria

Parámetros

Type	Name	Descripción	Schema
Path	**id** *required*	id	integer (int32)

Responses

HTTP Code	Descripción	Schema
200	**OK**	FranjaDTO
400	**Bad Request**. El id se encuentra vacío	No Content
401	**Unauthorized**. El usuario debe iniciar sesión	No Content
403	**Forbidden**. El usuario debe ser administrador	No Content
404	**Not Found**. No se encuentra el contenido solicitado	No Content

verFranja

GET /centrodeportivo/horario/{id}

Descripción

Operación para visualizar una franja horaria

Parámetros

Type	Name	Descripción	Schema
Path	**id** *required*	id	integer (int32)

Responses

HTTP Code	Descripción	Schema
200	**OK**	FranjaDTO
404	**Not Found**. No se encuentra el contenido solicitado	No Content

Lista Horarios

```
GET /centrodeportivo/horarios
```

Descripción

Operación para visualizar por parte del administrador el listado de todos los horarios

Responses

HTTP Code	Descripción	Schema
200	**OK**	<FranjaDTO>array
401	**Unauthorized**. El usuario debe iniciar sesión	No Content
403	**Forbidden**. El usuario debe ser administrador	No Content
404	**Not Found**. No se encuentra el contenido solicitado	No Content

Lista Franjas Horarias Disponibles

```
GET /centrodeportivo/horarios/{idPista}/{fecha}
```

Descripción

Operación para que se visualicen únicamente las franjas horarias disponibles para alquilar una pista

Parámetros

Type	Name	Descripción	Schema
Path	**fecha** *required*	fecha	string
Path	**idPista** *required*	idPista	integer (int32)

Responses

HTTP Code	Descripción	Schema
200	**OK**	<FranjaDTO> array
404	**Not Found**. No se encuentra el Contenido solicitado	No Content

6. Pistas

Ingresa Pista nueva

POST /centrodeportivo/pista

Descripción

Operación para que un administrador ingrese una pista en la aplicación

Parámetros

Type	Name	Descripción	Schema
Body	**pista** *required*	pista	PistaDTO

Responses

HTTP Code	Descripción	Schema
201	**Created**	No Content
400	**Bad Request**. El parámetro pista se encuentra vacío	
401	**Unauthorized**. El usuario debe iniciar sesión	No Content
403	**Forbidden**. El usuario debe ser administrador	No Content
404	**Not Found**. No se ha creado la pista	No Content

Visualiza Pista

GET /centrodeportivo/pista/{id}

Descripción

Operación para visualizar los datos de una pista

Parámetros

Type	Name	Descripción	Schema
Path	**id** *required*	id	integer (int32)

Responses

HTTP Code	Descripción	Schema
200	**OK**	PistaDTO
400	**Bad Request**. El parámetro id se encuentra vacío	No Content
404	**Not Found**. No se encuentra el contenido solicitado	No Content

Actualiza datos pista

```
PUT /centrodeportivo/pista/{id}
```

Descripción

Operación para modificar los datos de una pista

Parámetros

Type	Name	Descripción	Schema
Path	**id** *required*	id	integer (int32)
Body	**pista** *required*	pista	PistaDTO

Responses

HTTP Code	Descripción	Schema
200	**OK**	PistaDTO
400	**Bad Request**. El parámetro id se encuentra vacío	No Content
401	**Unauthorized**. El usuario debe iniciar sesión	No Content
403	**Forbidden**. El usuario debe ser administrador	No Content

404	**Not Found**. La pista se encuentra vacía	No Content

Visualiza Horario Pista

```
GET /centrodeportivo/pista/{id}/horario
```

Descripción

Operación para visualizar el horario de una determinada pista

Parámetros

Type	Name	Descripción	Schema
Path	**id** *required*	id	integer (int32)

Responses

HTTP Code	Descripción	Schema
200	**OK**	<FranjaDTO> array
404	**Not Found**. No se encuentra el contenido solicitado	No Content

Lista Pistas

```
GET /centrodeportivo/pistas
```

Descripción

Operación para visualizar todas las pistas

Responses

HTTP Code	Descripción	Schema
200	**OK**	<PistaDTO> array
404	**Not Found**. No se encuentra el contenido solicitado	No Content

Lista Pistas Activas

```
GET /centrodeportivo/pistasActivas
```

Descripción

Operación para visualizar las pistas que se encuentran activas

Responses

HTTP Code	Descripción	Schema
200	**OK**	<PistaDTO> array
404	**Not Found**. No se encuentra el contenido solicitado	No Content

7. Reservas

Crea Reserva

```
POST /centrodeportivo/reserva
```

Descripción

Operación para que un usuario pueda realizar una reserva en la aplicación

Parámetros

Type	Name	Descripción	Schema
Body	**reserva** *required*	reserva	ReservaDTO

Responses

HTTP Code	Descripción	Schema
201	**Created**	object
400	**Bad Request**. El parámetro reserva se encuentra vacío	No Content
401	**Unauthorized**. El usuario debe iniciar sesión	No Content
500	**Internal Server Error**. La pista ya se encuentra reservada ese día a esa hora	No Content

ver_Reserva

```
GET /centrodeportivo/reserva/{id}
```

Descripción

Operación para visualizar los datos de una reserva

Parámetros

Type	Name	Descripción	Schema
Path	**id** *required*	id	integer (int32)

Responses

HTTP Code	Descripción	Schema
200	**OK**	ReservaDTO
400	**Bad Request**. El parámetro se encuentra vacío	No Content
401	**Unauthorized**. El usuario debe iniciar sesión	No Content
404	**Not Found**. No se encuentra el contenido solicitado	No Content

verReserva

```
GET /centrodeportivo/usuarios/{dni}/reserva/{id}
```

Descripción

Operación para visualizar los datos de una reserva

Parámetros

Type	Name	Descripción	Schema
Path	**dni** *required*	dni	string
Path	**id** *required*	id	integer (int32)

Responses

HTTP Code	Descripción	Schema
200	**OK**	ReservaDTO
400	**Bad Request**. El parámetro se encuentra vacío	No Content
401	**Unauthorized**. El usuario debe iniciar sesión	No Content
404	**Not Found**. Los parámetros se encuentran vacíos	No Content

Cancela reserva no pagada

```
DELETE /centrodeportivo/reserva/{id}
```

Descripción

Operación para que una reserva pendiente de pago sea cancelada por parte del administrador

Parámetros

Type	Name	Descripción	Schema
Path	**id** *required*	id	integer (int32)

Responses

HTTP Code	Descripción	Schema
200	**OK**	ResponseEntity
400	**Bad Request**. El parámetro se encuentra vacío	No Content
401	**Unauthorized**. El usuario debe iniciar sesión	No Content
403	**Forbidden**. El usuario debe ser administrador	No Content
404	**Not Found**. No se ha encontrado el contenido solicitado	No Content
500	**Internal Server Error**. La reserva se encuentra pagada	No Content

Actualiza Reserva Pendiente

```
PUT /centrodeportivo/reservaPendiente/{idReserva}/{idPago}
```

Descripción

Operación para pagar una reserva pendiente

Parámetros

Type	Name	Descripción	Schema
Path	**idPago** *required*	idPago	string
Path	**idReserva**	idReserva	integer (int32)

	required		

Responses

HTTP Code	Descripción	Schema
200	**OK**	No Content
400	**Bad Request**. El parámetro se encuentra vacío	No Content
401	**Unauthorized**. El usuario debe iniciar sesión	No Content

Lista Reservas

```
GET /centrodeportivo/reservas
```

Descripción

Operación para visualizar un listado de todas las reservas

Responses

HTTP Code	Descripción	Schema
200	**OK**	<ReservaDTO>array
401	**Unauthorized**. El usuario debe iniciar sesión	No Content
404	**Not Found**. No se encuentra el contenido solicitado	No Content

Lista reservas no pagadas

```
GET /centrodeportivo/reservasPendientes
```

Descripción

Operación para que el administrador visualice el listado de las reservas pendientes

Responses

HTTP Code	Descripción	Schema
200	**OK**	<ReservaDTO>array

401	**Unauthorized**. El usuario debe iniciar sesión	No Content
403	**Forbidden**. El usuario debe ser administrador	No Content
404	**Not Found**. No se encuentra el contenido solicitado	No Content

8. Usuarios

Inicia Sesión

POST /centrodeportivo/signin

Descripción

Operación para que un usuario inicie sesión en la aplicación

Parámetros

Type	Name	Descripción	Schema
Body	**loginRequest** *required*	loginRequest	LoginRequest

Responses

HTTP Code	Descripción	Schema
200	**OK**	object
400	**Bad Request**. El usuario no puede Iniciar sesión porque está desactivado	No Content
401	**Unauthorized**. Credenciales Erróneas	
404	**Not Found.** No se encuentra el contenido solicitado	No Content

Registra Usuario

POST /centrodeportivo/signup

Descripción

Operación para que un usuario se registre en la aplicación

Parámetros

Type	Name	Descripción	Schema
	signUpReques		

Body	**t** *required*	signUpRequest	SignupRequest

Responses

HTTP Code	Descripción	Schema
200	**OK**	object
400	**Bad Request**. El nombre de usuario o email ya existe	No Content
404	**Not Found**. No se encuentra el contenido solicitado	No Content

Visualiza Usuario

```
GET /centrodeportivo/usuario/{dni}
```

Descripción

Operación para visualizar los datos de un usuario

Parámetros

Type	Name	Descripción	Schema
Path	**dni** *required*	dni	string

Responses

HTTP Code	Descripción	Schema
200	**OK**	UsuarioDTO
400	**Bad Request**. El parámetro se encuentra vacío	No Content
401	**Unauthorized**. El usuario debe iniciar sesión	No Content
404	**Not Found**. No se encuentra el contenido disponible	No Content

Actualiza Usuario

```
PUT /centrodeportivo/usuario/{dni}
```

Descripción

Operación para actualizar los datos de un usuario

Parámetros

Type	Name	Descripción	Schema
Path	**dni** *required*	dni	string
Body	**usuario** *required*	usuario	UsuarioDTO

Responses

HTTP Code	Descripción	Schema
200	**OK**	UsuarioDTO
400	**Bad Request**. El parámetro se encuentra vacío	No Content
401	**Unauthorized**. El usuario debe iniciar sesión	No Content
404	**Not Found**. No se encuentra el contenido solicitado	No Content

Actualiza el estado de un usuario

```
PUT /centrodeportivo/usuarioEstado/{dni}
```

Descripción

Operación para activar/desactivar a un usuario

Parámetros

Type	Name	Descripción	Schema
Path	**dni** *required*	dni	string

Responses

HTTP Code	Descripción	Schema
200	**OK**	UsuarioDTO
400	**Bad Request**. El parámetro se encuentra vacío	No Content
401	**Unauthorized**. El usuario debe iniciar sesión	No Content
403	**Forbidden**. El usuario debe ser administrador	No Content

404	**Not Found**. No se encuentra el contenido solicitado	No Content

Lista Usuarios

GET /centrodeportivo/usuarios

Descripción

Operación para visualizar un listado de los usuarios por parte del Administrador

Responses

HTTP Code	Descripción	Schema
200	**OK**	<UsuarioDTO>array
401	**Unauthorized**. El usuario debe iniciar sesión	No Content
403	**Forbidden**. El usuario debe ser administrador	No Content
404	**Not Found**. No se encuentra el contenido solicitado	No Content

Visualiza mis reservas

GET /centrodeportivo/usuarios/{dni}/reservas

Descripción

Operación para que un usuario pueda visualizar sus reservas realizadas

Parámetros

Type	Name	Descripción	Schema
Path	**dni** *required*	dni	string

Responses

HTTP Code	Descripción	Schema
200	**OK**	<ReservaDTO>array
401	**Unauthorized**. El usuario debe	No Content

	iniciar sesión	
404	**Not Found**. No se encuentra el contenido solicitado	No Content

Apéndice B: Manual de Instalación

A continuación, mostraremos cómo debemos de instalar nuestra aplicación para que pueda ser utilizada en cualquier ordenador, únicamente haciendo uso de Docker.

Lo primero que debemos de hacer es irnos al siguiente enlace y descargar la aplicación Docker Desktop.

https://docs.docker.com/desktop/windows/install/

Como podemos ver en la imagen, una vez instalado y ejecutándose, lo siguiente será descargar desde el repositorio las imágenes de los servicios en cuestión.

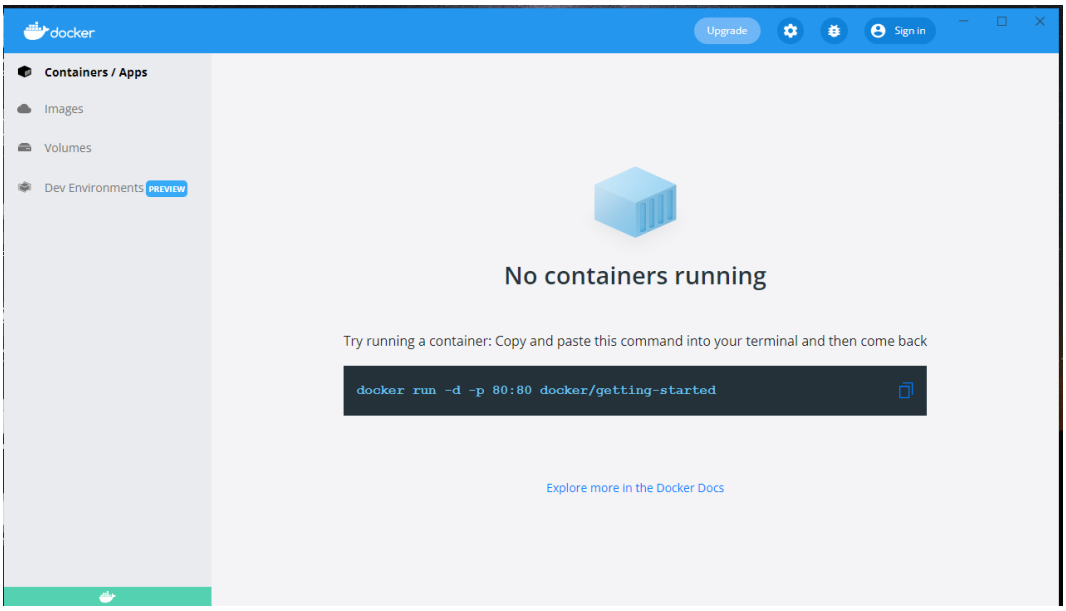


Ilustración 0.77 Docker Desktop

El comando para descargar la imagen del Backend es el siguiente:

```
docker pull jmpc0016/tfg:backendimage
```

Ilustración 0.78 Comando descargar Backend

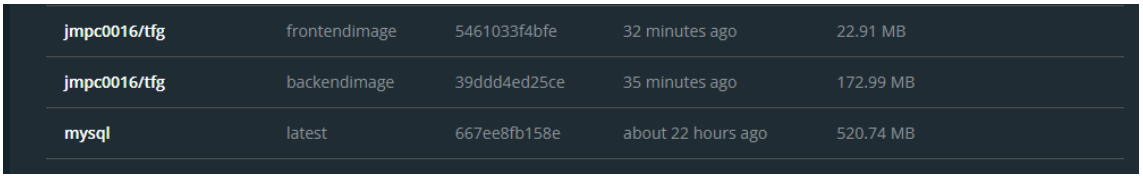
Para el Frontend:

```
docker pull jmpc0016/tfg:frontendimage
```

Ilustración 0.79 Comando descargar Frontend

Para la base de datos es el mismo comando que anteriormente hemos comentado.

Como podemos ver ya tenemos en nuestro Docker las imágenes que vamos a utilizar.



jmpc0016/tfg	frontendimage	5461033f4bfe	32 minutes ago	22.91 MB
jmpc0016/tfg	backendimage	39ddd4ed25ce	35 minutes ago	172.99 MB
mysql	latest	667ee8fb158e	about 22 hours ago	520.74 MB

Ilustración 0.80 Docker Desktop Imágenes

El archivo Docker-compose deberá ser modificado con los nombres que tienen las nuevas imágenes.

```
myapp-mysql:
  image: mysql:latest
  environment:
    - MYSQL_ROOT_PASSWORD=Dae2020
    - MYSQL_DATABASE=centrodeportivo
    - MYSQL_PASSWORD=Dae2020
  cap_add:
    - SYS_NICE  # CAP_SYS_NICE
  ports:
    - 3306:3306
  volumes:
    - /home/hans/Escritorio/git_proyect/bd
  networks:
    - employee-mysql

myapp-main:
  image: jmpc0016/tfg:backendimage
  networks:
    - employee-mysql
  depends_on:
    - myapp-mysql
  ports:
    - 8080:8080
  environment:
    - DATABASE_HOST=myapp-mysql
    - DATABASE_USER=root
    - DATABASE_PASSWORD=Dae2020
    - DATABASE_NAME=centrodeportivo
    - DATABASE_PORT=3306
frontend:
  image: jmpc0016/tfg:frontendimage
```

Ilustración 0.81 Nuevo docker-compose

Por último, ejecutamos dicho archivo con el siguiente comando:

```
docker-compose up -d
```

Ilustración 0.82 Comando docker-compose

Como vemos, todos se han lanzado correctamente y podemos acceder a la aplicación únicamente con el Docker ejecutándose.

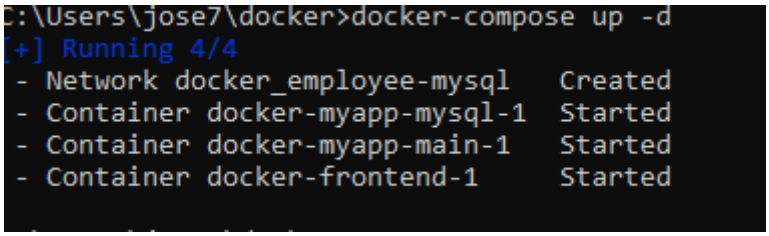


Ilustración 0.83 Docker Lanzado

Ilustración 0.84 Página corriendo con Docker

Cabe destacar que estos comandos estarán contenidos dentro de un fichero .bat para que únicamente debamos de ejecutar dicho fichero para descargar y lanzar nuestra aplicación junto con el fichero docker-compose.

Apéndice C: Manual de Usuario

En este manual de usuario explicaremos el funcionamiento de nuestra aplicación.

La explicación de esta estará dividida en 3 apartados, uno para un **usuario no registrado** y que únicamente desee navegar por la aplicación para conocer las instalaciones con las que cuenta el centro deportivo, otro para los **usuarios registrados** y, por último, para el **administrador**, el cual tendrá que realizar la configuración de la aplicación y gestión del centro deportivo.

Administrador

Una vez el administrador arranque la aplicación por primera vez, deberá configurar y personalizar esta.

La aplicación le mostrará un mensaje indicándole que va a comenzar la configuración:

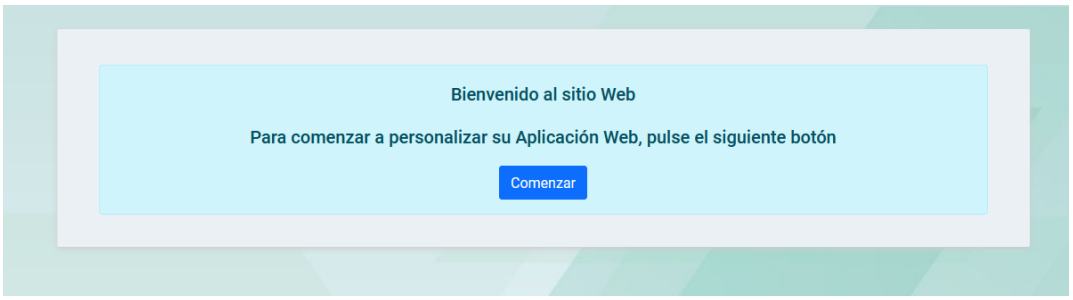


Ilustración 0.85 Mensaje Inicio Configuración

En el caso de que ya se haya personalizado la aplicación, mostrará el siguiente mensaje:

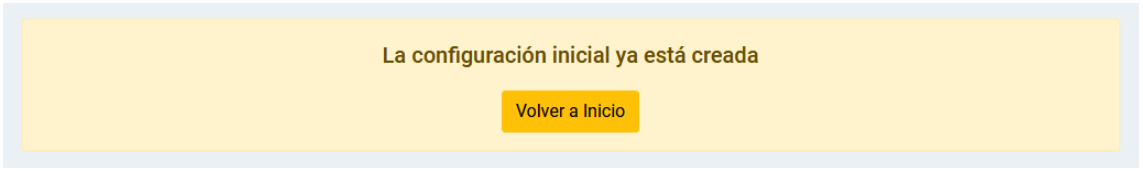


Ilustración 0.86 Mensaje Configuración ya creada

Lo primero que deberá introducir será la información acerca del centro deportivo, título, descripción, dirección, correo electrónico, etc.

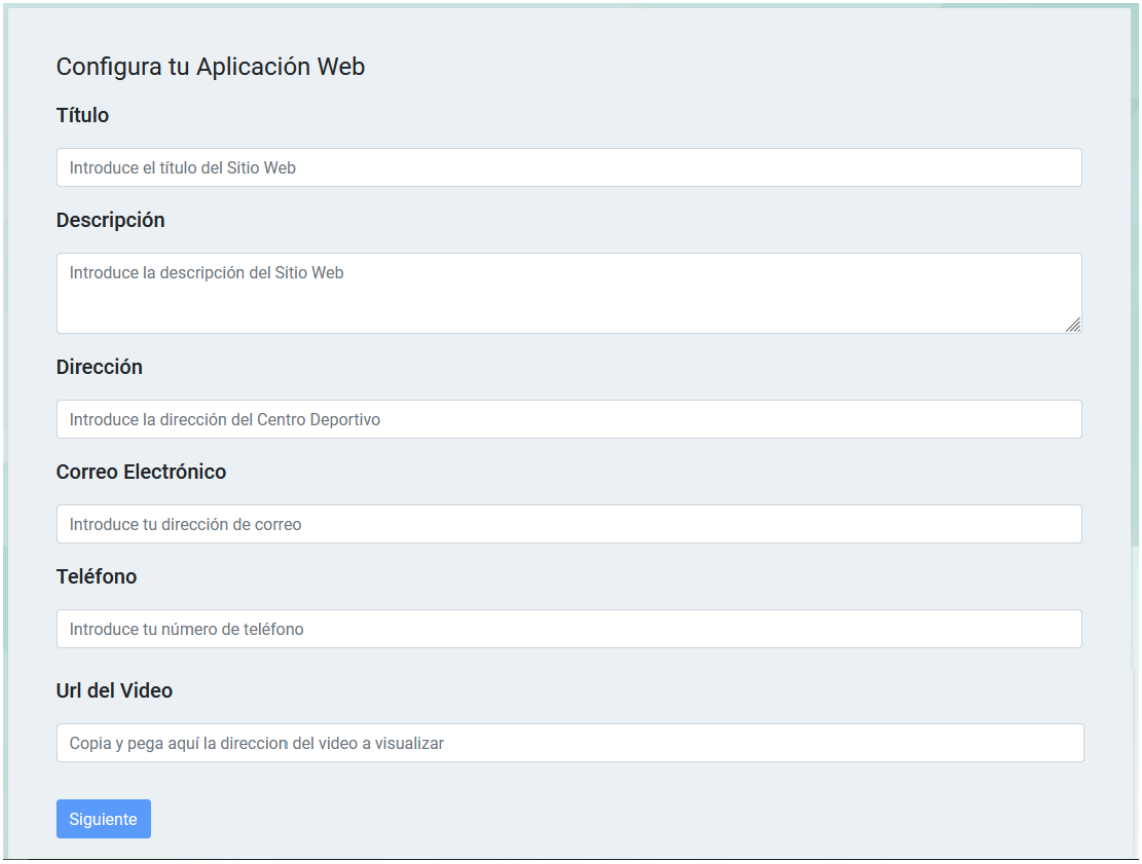


Ilustración 0.87 Formulario Información Centro Deportivo

En cada uno de los formularios, los datos a introducir son validados, ya sea por patrones, longitud, campos obligatorios, etc.

Como podemos ver en la siguiente imagen si no se cumple alguna de estas restricciones se mostrará un mensaje de error indicándolo.

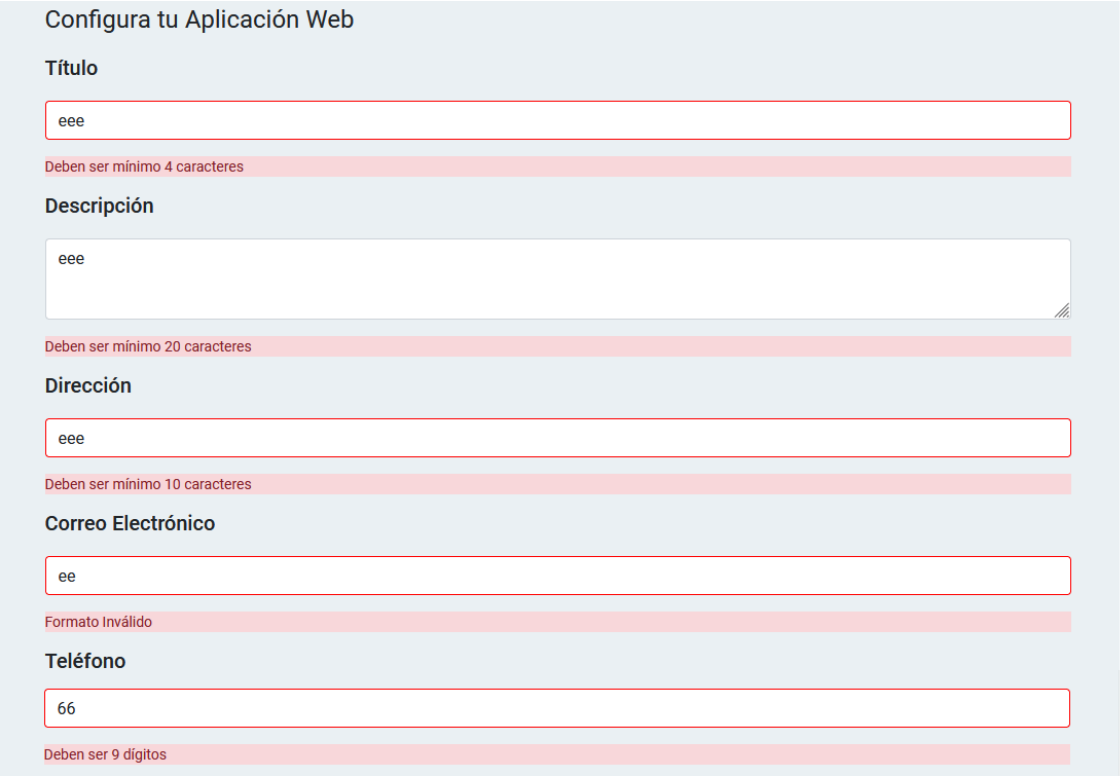


Ilustración 0.88 Errores Formulario Información Centro Deportivo

Al pulsar “Siguiente” se generará la barra de navegación y la cabecera de la aplicación que aparecerá en cada una de las distintas páginas, el título de esta será el valor introducido en el campo Título del formulario anterior. En las siguientes imágenes no mostraremos la cabecera y navegación generada ya que preferimos centrarnos en los formularios para la configuración.

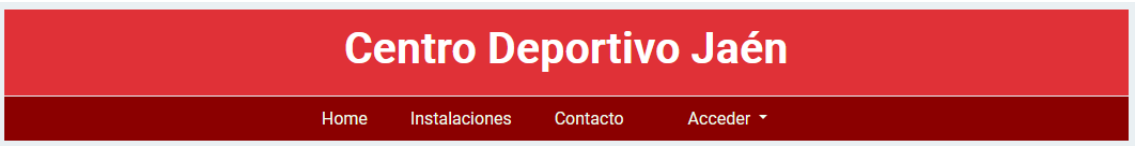


Ilustración 0.89 Barra de Navegación

Si el usuario cuenta con alguna reserva que no ha pagado, se mostrará en la barra de navegación un icono indicándolo, si pulsa sobre este, será redirigido al apartado de “Mis Reservas” en “Perfil” para pagarla.

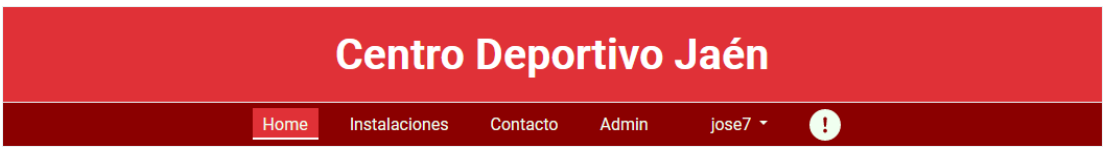


Ilustración 0.90 Barra de Navegación Reservas Pendientes

El siguiente paso en la configuración será la creación del usuario Administrador.

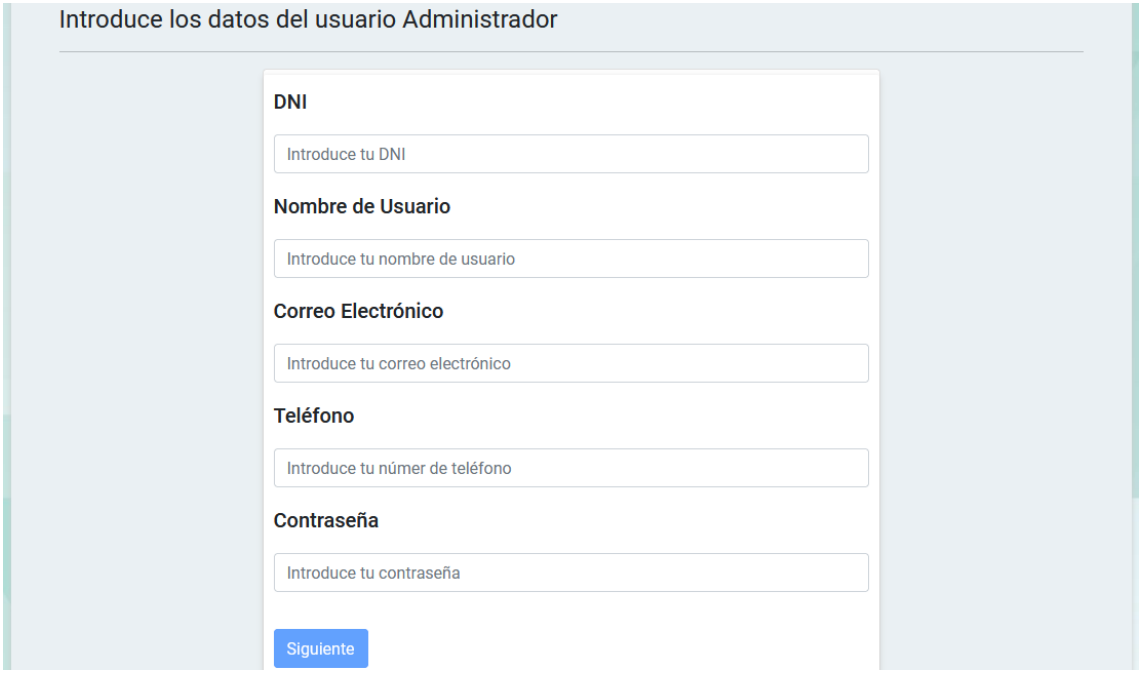


Ilustración 0.91 Formulario Usuario Administrador

También con las restricciones para cada campo:

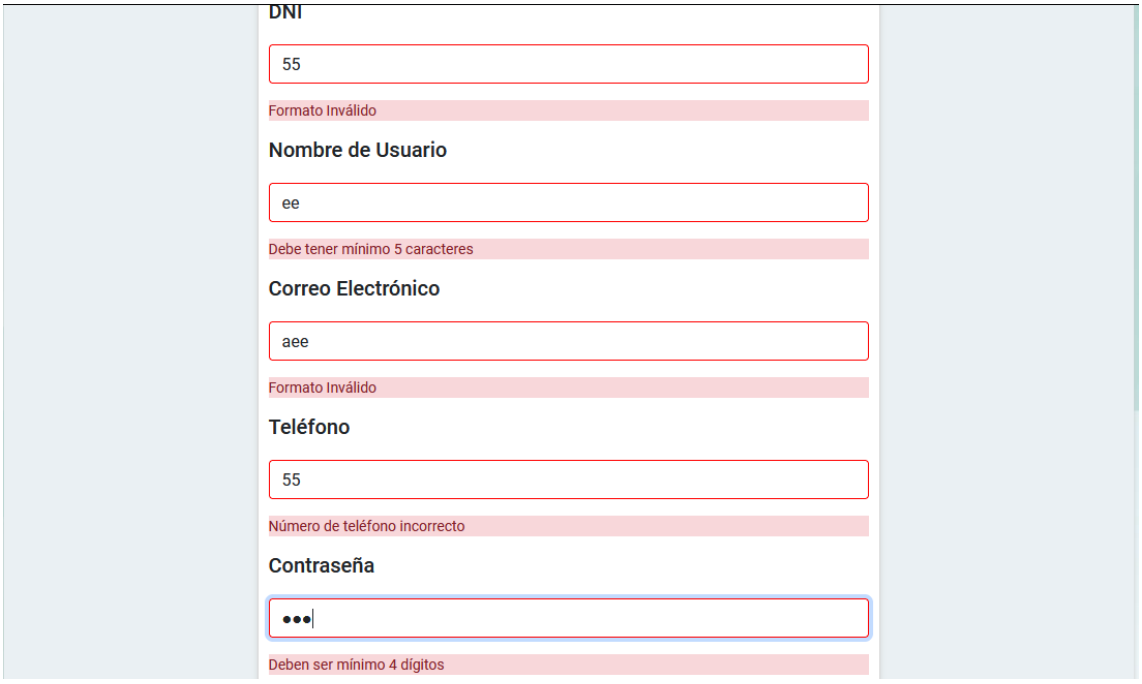


Ilustración 0.92 Errores Formulario Administrador

Una vez se hayan introducido los datos, el administrador deberá iniciar sesión introduciendo su nombre de usuario y contraseña para continuar con la configuración.

Ilustración 0.93 Login

Lo siguiente será decidir qué días no se podrán reservas las instalaciones, para ello pinchará en el icono del date picker y seleccionará los días deseados. En caso de que todos los días se puedan realizar reservas se selecciona “Siguiente”, no seleccionándose así ninguno.

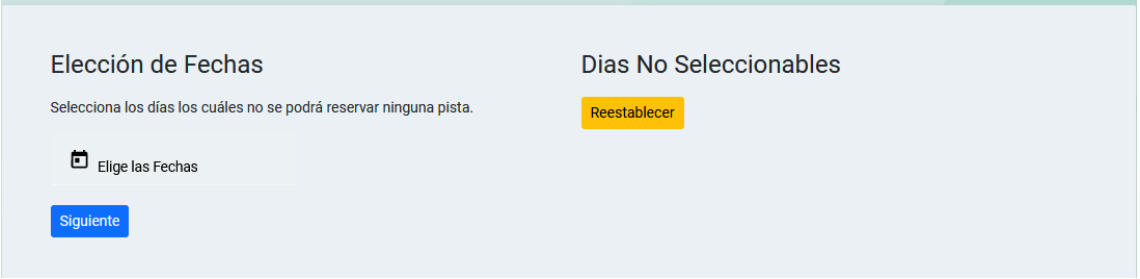


Ilustración 0.94 Selección Fechas No Reservables

Una vez seleccionado los días, pasamos a la creación de cada una de las pistas con las que cuenta nuestro centro deportivo, no podemos olvidar que debemos seguir cumpliendo con las restricciones.

Para la creación de cada una de las pistas deberemos de introducir la actividad deportiva a desarrollar, descripción y dimensiones de la pista y la imagen asociada.

Ilustración 0.95 Formulario Creación Pista

Lo siguiente será asociarle a cada pista su horario correspondiente. Para ello introduciremos la hora de inicio y la hora de fin que podrá ser reservada, generándose cada una de ellas de manera automática. No podemos olvidar que cada franja tendrá una duración de 1 hora.

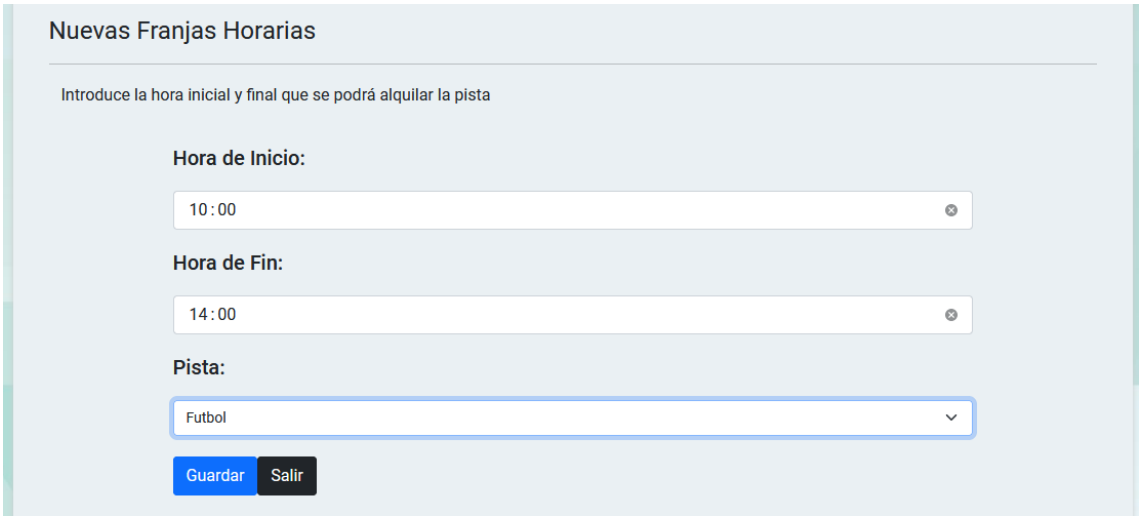


Ilustración 0.96 Formulario Horario Pista

Si alguna de las franjas horarias a introducir ya se encontrara creada, la aplicación mostrará un mensaje de error indicándolo.

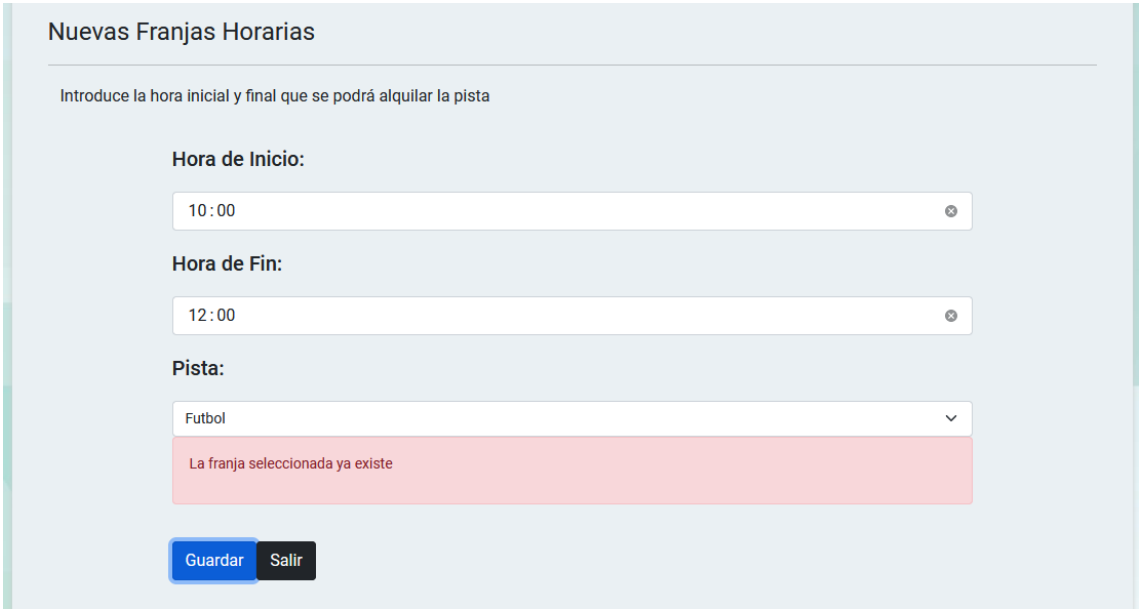


Ilustración 0.97 Errores Formulario Horario Pista

En el caso de que el centro deportivo cuente con más pistas y tengamos que seguir creándolas, pulsaremos el botón de “Crear otra pista”, si no, “Confirmar”.

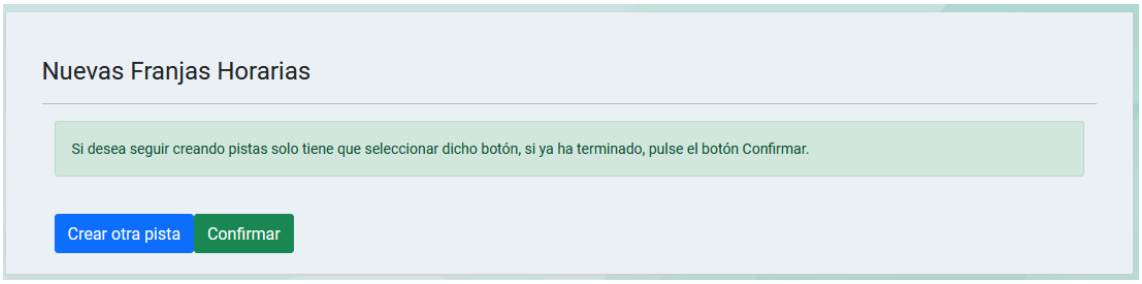


Ilustración 0.98 Mensaje Creación Pistas

Si hemos pulsado el botón de “Confirmar” nos aparecerá un mensaje indicándonos que la aplicación ha sido configurada con éxito y nos redirigirá a la página principal.

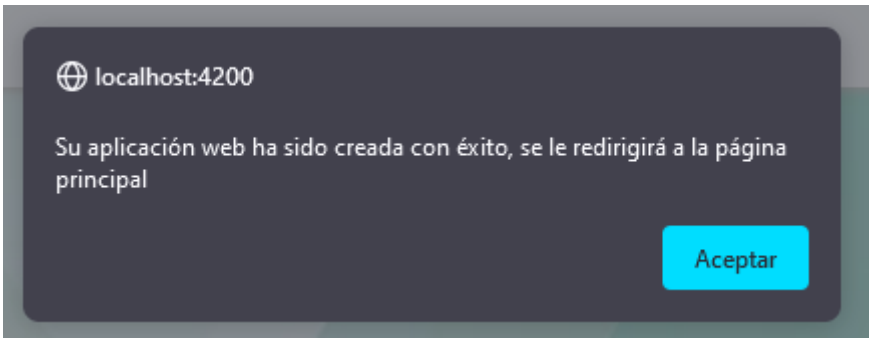


Ilustración 0.99 Mensaje Configuración Finalizada

Como podemos ver en la siguiente imagen, la página principal cuenta con información acerca del centro deportivo, un carrusel con las imágenes de las pistas, un calendario para poder consultar los días que se pueden reservas las instalaciones, un widget para conocer el tiempo y un video donde se mostrará el recinto deportivo.

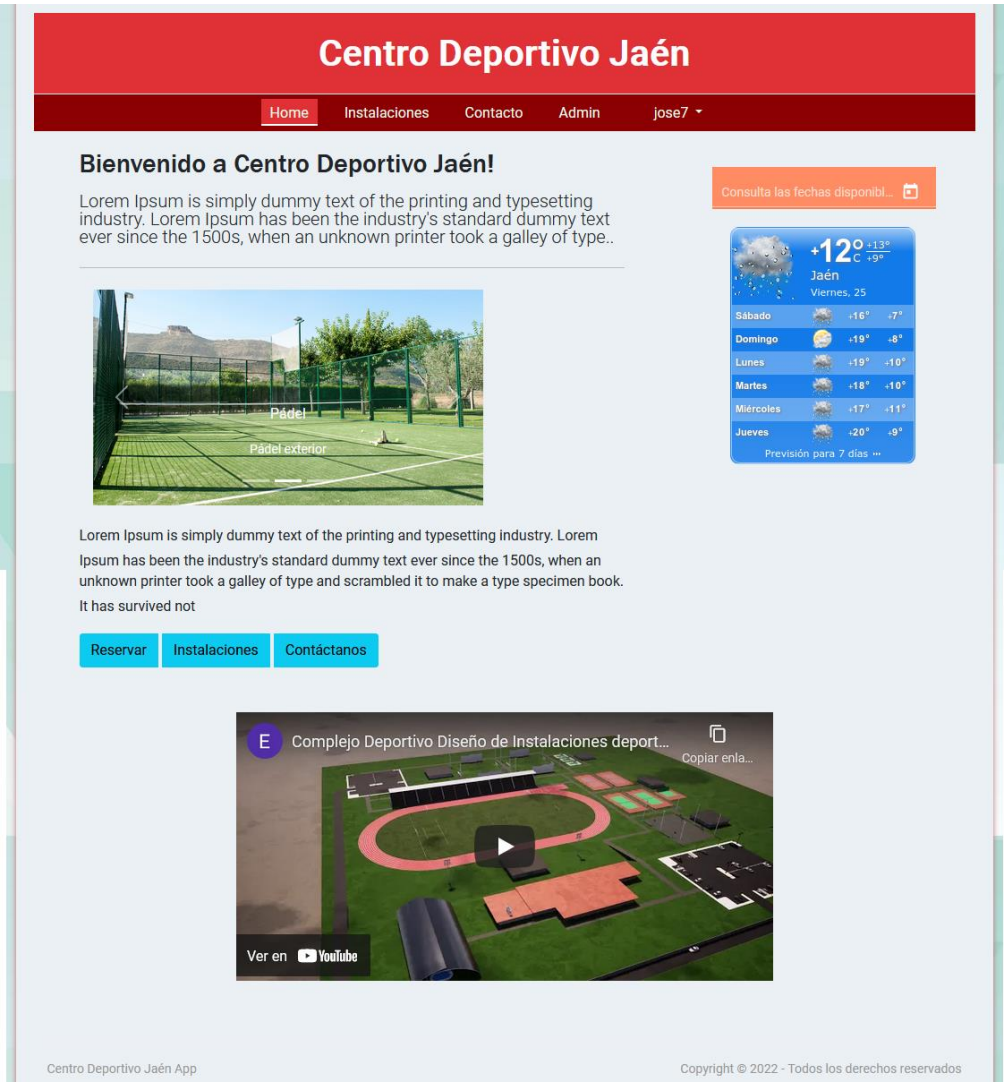


Ilustración 0.100 Página Home

Al igual que en la barra de navegación aparece un icono cuando alguna reserva no se ha pagado, en la página principal aparecerá un mensaje indicándolo:

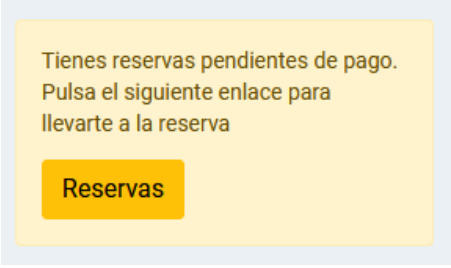


Ilustración 0.101 Mensaje Reserva Pendiente de Pago

Una vez inicie sesión el administrador, en la navegación aparecerá una pestaña “Admin”, a partir de la cual se podrá gestionar toda la aplicación.

Ilustración 0.102 Panel Administrador

Como podemos ver en la imagen anterior, en el panel del Administrador, se podrá gestionar, usuarios, instalaciones, franjas horarias, días no seleccionables y reservas. A continuación, entraremos en detalle en cada uno de ellos.

Si pinchamos en **Usuarios**, aparecerá un listado con todos los usuarios registrados en la aplicación, incluido el administrador. Este podrá crear un nuevo usuario para pruebas y activar/desactivar usuarios para que no puedan iniciar sesión en la aplicación en el caso de que no hagan efectivo el pago de alguna reserva, etc. El administrador podrá activar/desactivar a un usuario siempre y cuando este no sea Administrador.

En la siguiente imagen vemos como un usuario se encuentra desactivado y el usuario jose7 que es el administrador no.

Ilustración 0.103 Panel Administrador Usuarios

Para las **Instalaciones**, el administrador podrá visualizar un listado con la información de las pistas creadas, además podrá ingresar una nueva pista o editar la información de estas.

Ilustración 0.104 Listado Pistas Administrador

Como podemos ver en la siguiente imagen, el administrador al pulsar el botón “Editar” en la página anterior, podrá desactivar una pista seleccionando el botón “Desactivar”. De esta manera si una pista se encuentra desactivada no aparecerá en la pestaña de Instalaciones.

Ilustración 0.105 Formulario Editar Pista

Si accedemos a **Franjas Horarias**, se visualizará un listado con cada una de las franjas horarias existentes. Cada una de estas puede ser desactivada en caso de mantenimiento, etc. De esta forma no aparecerán a la hora de realizar una reserva ni en el horario de una pista.

Ilustración 0.106 Listado Franjas Horarias Administrador

Para la creación del horario de una pista, la aplicación cuenta con dos maneras de hacerlo. Si necesitamos ingresar una nueva pista, al introducir la información de esta y confirmar, la aplicación redirigirá al administrador al formulario para introducir su horario. Por otro lado, si únicamente queremos introducir una franja horaria individual, pincharemos en el botón “Nueva Franja”.

En cuanto a la sección **Días No Reservables**, como vemos en la siguiente imagen, el administrador visualizará en la parte derecha los días que no se pueden reservar junto con un botón para reestablecerlos y en la parte izquierda, el date picker para seleccionarlos.

Ilustración 0.107 Días No Reservables Administrador

Por último, para la gestión de **Reservas**, el administrador al igual que en los anteriores apartados, visualizará un listado con todas las reservas realizadas.

Ilustración 0.108 Listado Reservas Administrador

Además, si pincha el botón “Pendientes de Pago”, visualizará otro listado, pero con las reservas que no han sido pagadas por el momento.

Ilustración 0.109 Listado Reservas Pendientes Administrador

También contará con la opción de cancelar las reservas pendientes de pago. Si desea cancelar alguna de estas, únicamente pulsará dicho botón y se le mostrará un mensaje de confirmación donde deberá cancelar o aceptar.

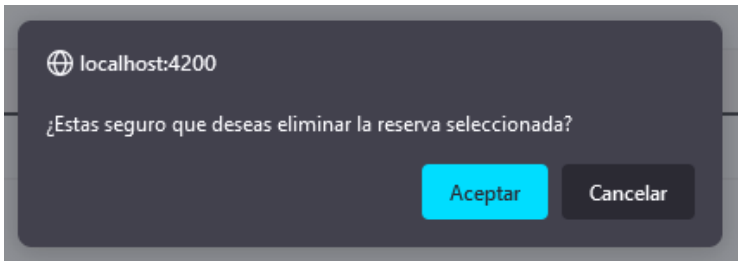


Ilustración 0.110 Mensaje Confirmación Cancelar Reserva

Usuario No Registrado

Para los usuarios que no se han registrado en la aplicación podrán hacer uso de las siguientes funcionalidades.

Podrán visualizar la página principal junto con toda su funcionalidad.

En la pestaña **Instalaciones** podrán visualizar las pistas con las que cuenta el centro deportivo. Además, cada pista tiene dos botones, uno para ver su información y otro para el horario.

Ilustración 0.111 Instalaciones

Si el usuario desea ver la información de una pista, este podrá visualizar su descripción, dimensiones, etc.

Ilustración 0.112 Pista Detalle

Para el horario, se muestra un listado con las franjas horarias de las que dispone. Si recordamos, anteriormente en el listado que el administrador tenía para listar las franjas horarias, dos de estas se encontraban desactivadas, como podemos ver en la imagen siguiente, no aparecen.

Ilustración 0.113 Horario Pista

Por último, un usuario no registrado podrá enviar sus consultas al centro deportivo y visualizar la información de este. Para ello, deberá acceder a la pestaña **Contacto**, en ella rellenará el formulario para que así el centro deportivo reciba su consulta. Además, para la información del centro deportivo, el usuario podrá visualizar la dirección, el correo electrónico, etc.

Ilustración 0.114 Contacto

El correo que recibirá el centro deportivo tendrá el siguiente formato.

Nombre: Jose Maria Peña Fernández
Direccion de Email: carpinteriajosemari@hotmail.es
Asunto: Consulta Pista

Para la pista de futbol 7 podríamos jugar 8 personas

Ilustración 0.115 Formato Email Consulta

Usuarios Registrados

Para que un usuario se registre en la aplicación, deberá pinchar en la pestaña **Acceder** y, **Registrarse**, rellenará el siguiente formulario con sus datos e iniciará sesión en el Login anteriormente mostrado. En el caso de que ya esté registrado, seleccionará **Iniciar Sesión**.

Ilustración 0.116 Registro Usuario

Si el usuario introduce un nombre de usuario o correo electrónico que ya existe, se le mostrará un mensaje de error indicándolo.

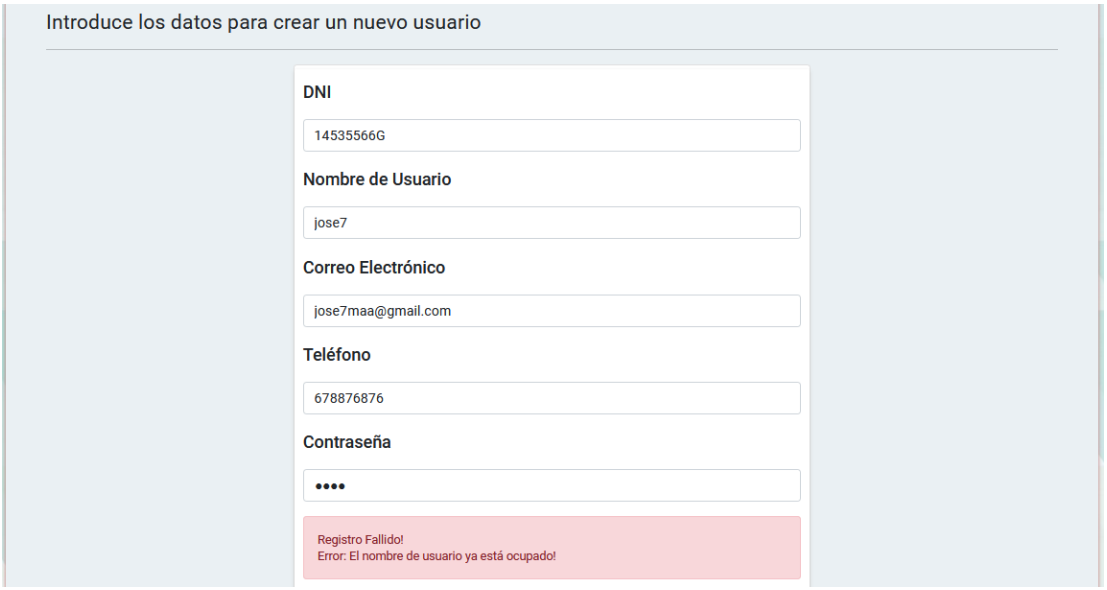


Ilustración 0.117 Mensaje Error Nombre Usuario Existente

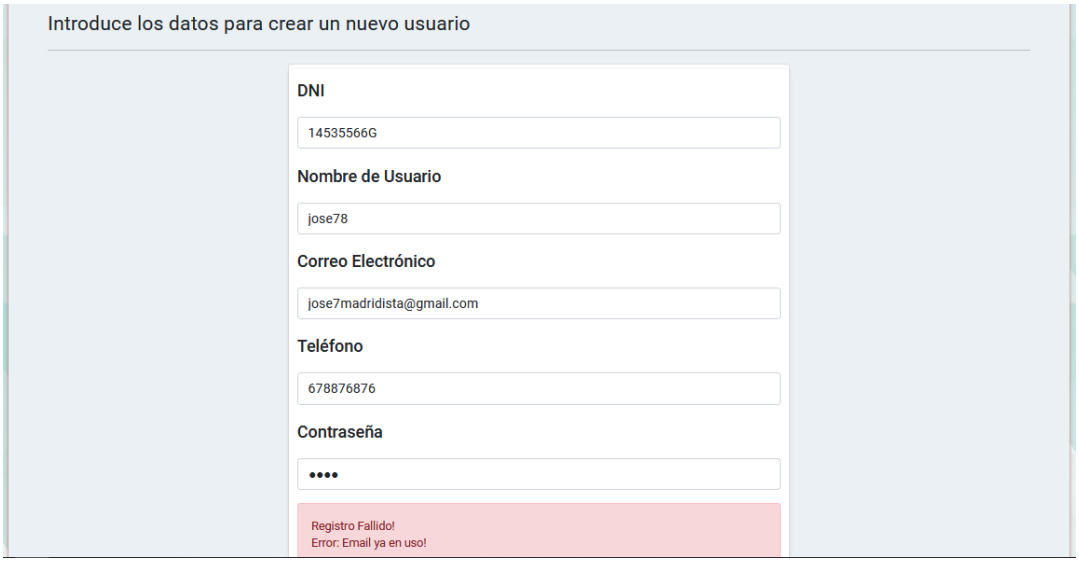


Ilustración 0.118 Mensaje Error Email Existente

Si en algún momento el usuario desea acceder a ciertas rutas o funcionalidades las cuales no le están permitidas, se le mostrará el siguiente mensaje:

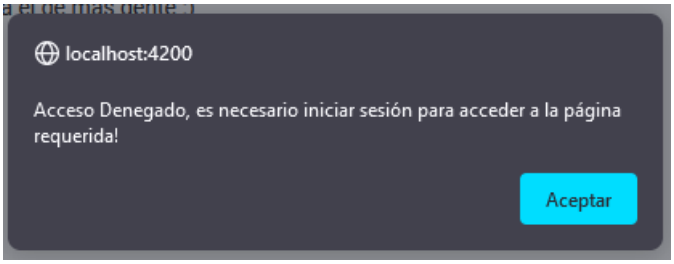


Ilustración 0.119 Mensaje Error

Si el usuario logueado intenta acceder a funcionalidades del administrador:

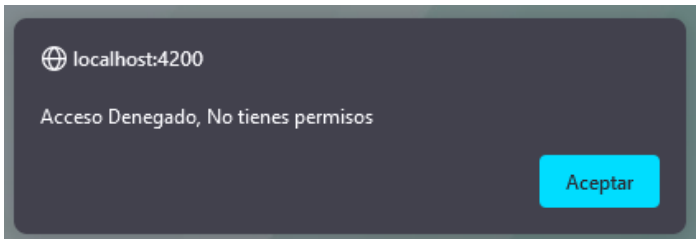


Ilustración 0.120 Mensaje Error Permisos

Si el usuario al iniciar sesión introduce mal sus credenciales, se le mostrará un mensaje de error indicándolo y si su usuario se encuentra desactivado por el administrador, también se le indicará y no le dejará iniciar sesión hasta que vuelva a ser activado.

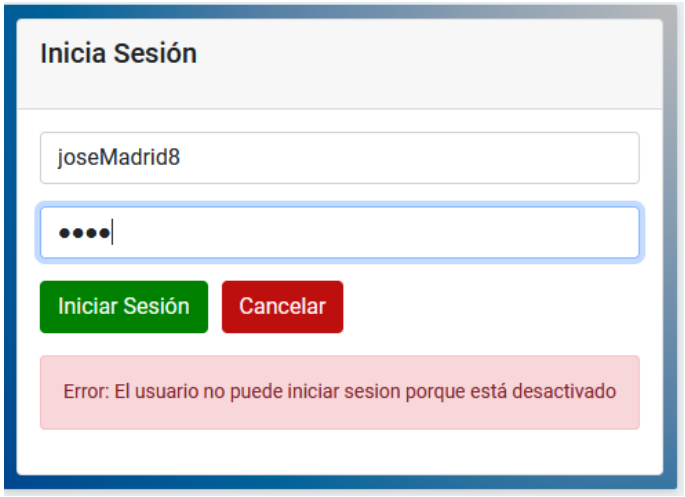


Ilustración 0.121 Mensaje Error Usuario Desactivado

Una vez el usuario inicie sesión correctamente, la aplicación le redirigirá a su **Perfil**. En el podrá visualizar sus datos y contará con tres botones que explicaremos a continuación.

Ilustración 0.122 Perfil Usuario

El usuario podrá modificar sus datos como la contraseña, correo electrónico y teléfono, siempre cumpliendo con las restricciones.

Ilustración 0.123 Modifica Datos Usuario

Desde el **Perfil** también podrá realizar una reserva y visualizarlas.

Al Pinchar en “Nueva Reserva”, el usuario podrá visualizar en la parte derecha los días que no se pueden reservas, estos los añadimos anteriormente al configurar la aplicación.

Ilustración 0.124 Realizar Reserva 1

Para realizar la reserva, como vemos en la imagen anterior, deberá seleccionar el día que desea, en el caso de seleccionar alguno de los días que no se pueden reservar, se mostrará un mensaje de error indicándolo.

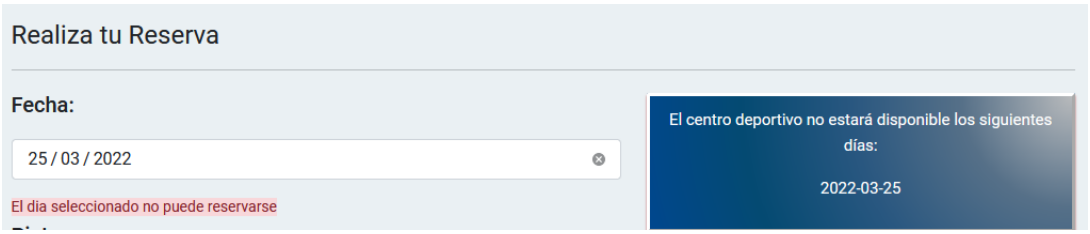


Ilustración 0.125 Mensaje Error Día No Reservable

Al seleccionar la pista, únicamente se mostrarán las pistas que se encuentren activas. Una vez hayamos seleccionado estos campos, aparecerán las horas de inicio que se encuentren activas y no se hayan reservado ya.

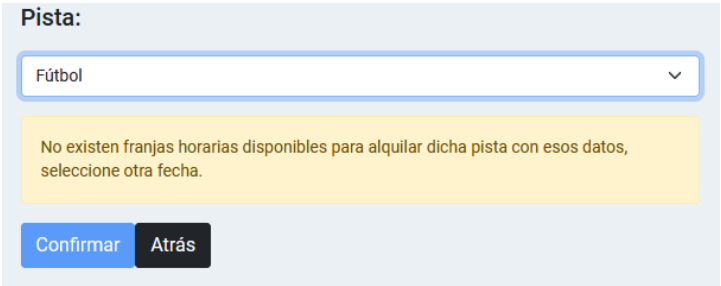


Ilustración 0.126 Mensaje Error No Existen Horas

Por último, se generará automáticamente la hora de fin ya que solo se puede reservar por cada reserva una franja horaria de una hora de duración.

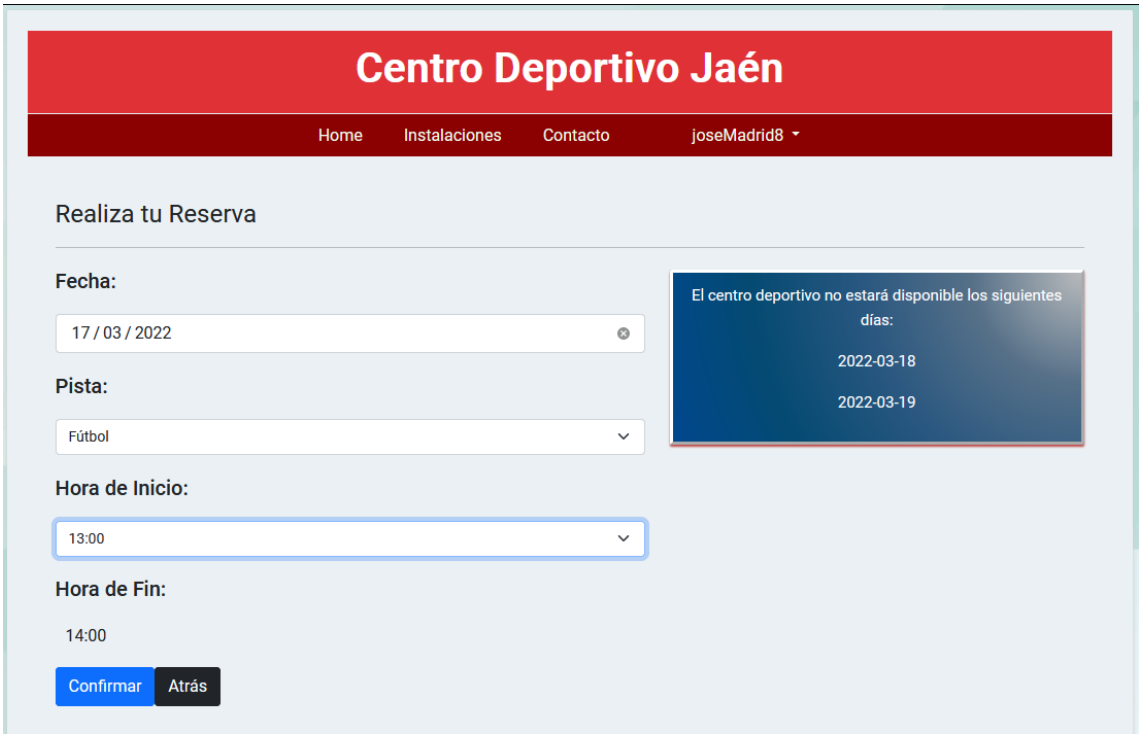


Ilustración 0.127 Realizar Reserva 2

Al confirmar los datos de la reserva, nos aparecerá el precio y un botón para poder realizar el pago, seleccionando entre Paypal, Sofort o Tarjeta de Crédito.

Ilustración 0.128 Realizar Reserva 3

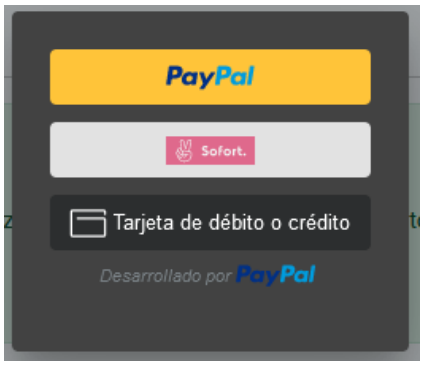


Ilustración 0.129 Métodos de Pago

Si el pago se ha realizado con éxito, se mostrará un mensaje y se dará la opción de volver a reservar por si se desea alquilar una pista más de una hora, etc.

Ilustración 0.130 Reserva 4

Además, el usuario recibirá un correo electrónico de confirmación con los datos de la reserva. El formato de este será similar al siguiente:

Su reserva con los siguientes datos se ha realizado correctamente
DNI del usuario encargado: 16334455C
Fecha de la Reserva: 2022-03-17
Hora de Inicio: 12:00
Hora de Fin: 13:00
Pista: Fútbol
Dimensiones de la Pista: Fútbol 7

Ilustración 0.131 Formato Correo Confirmación reserva

Para finalizar, el último botón que aparece en **Perfil** es “Mis Reservas”, en él se podrá visualizar las reservas realizadas y pagar las pendientes de pago por cualquier error.

Ilustración 0.132 Listado Mis Reservas