Arties: una web de gestión integral de gimnasios online y presenciales
Arties: a comprehensive management website for online and onsite gyms

Trabajo de Fin de Grado
Curso 2022–2023

Autores
Alberto Pascual Fernández
Andrés Romero Arbáizar

Director
Ramón González del Campo

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Arties: una web de gestión integral de gimnasios online y presenciales

Arties: a comprehensive management website for online and onsite gyms

Trabajo de Fin de Grado en Ingeniería Informática

Autores

Alberto Pascual Fernández
Andrés Romero Arbáizar

Director

Ramón González del Campo

Convocatoria: *Junio 2023*

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

29 de mayo de 2023

Dedicatoria

A nuestros abuelos por confiar en nosotros más que nosotros mismos.

Agradecimientos

A los profesores que a lo largo de la carrera nos han hecho capaces de llevar a cabo este proyecto. A nuestros compañeros que nos han hecho crecer en lo personal y en lo académico. Pero sobre todo, a nuestras familias y amigos por apoyarnos y soportarnos tanto en los momentos fáciles como en los difíciles.

Resumen

Arties es una aplicación web pensada para aquellos gimnasios que busquen dar una gran flexibilidad de servicios a sus clientes. Para ello tiene en cuenta diferentes ámbitos como cursos online creados por los profesionales del sitio, alquiler de material para hacer ejercicio en casa, o la posibilidad de pagar el precio individual de las clases que quiera el cliente sin tener que pagar la suscripción completa del lugar.

Existen tres tipos de usuarios:

- Los clientes pueden elegir entre una suscripción online o una presencial. La online permite acceder a todos los cursos de la web, además de poder alquilar material para realizar ejercicios en casa. La presencial contiene todas las ventajas de la *online*, pero además tiene acceso a las instalaciones y las clases son gratis, a las cuales deberán apuntarse igualmente. Ambas tendrán un perfil donde ver sus datos, poner información relevante para los profesores y ver ciertas estadísticas. También podrán hacer uso del chat para hacer preguntas a los entrenadores
- Los empleados se encargan de gestionar los cursos de la web, al igual que el material que se alquila. También podrán responder preguntas realizadas por los usuarios vía el chat de la aplicación.
- Por último está el administrador, quien tiene permisos totales. Puede gestionar y hacer todo lo que haya en la web, además de poder controlar parámetros del gimnasio y empleados.

Palabras clave

Spring Boot, Java, Thymeleaf, SQL, JPA, Java Script, web, gimasio online, gimnasio presencial, gestión de gimnasios

Abstract

Arties is a web application designed for those gyms that seek to deliver great flexibility to their clients.

Arties is a web application designed for those gyms that seek to provide great flexibility in services to their clients. To achieve this, it takes into account different aspects such as online courses created by the professionals of the site, equipment rental for exercising at home, or the possibility of paying the individual price for the classes that the client wants, without having to pay for the full subscription of the venue.

There are three types of users:

- Clients can choose between an online or onsite subscription. The online subscription allows access to all courses on the website, as well as the ability to rent equipment for home workouts. The in-person subscription includes all the advantages of the online subscription, but also grants access to the facilities and free classes, for which users will need to sign up in advance. Both types of clients will have a profile where they can see their data, provide relevant information to the instructors, and view certain statistics. They can also use the chat toask the trainers any questions they may have.

- The employees are responsible for managing the website's courses and the rental equipment. They can also respond to user questions via the application's chat.

- Finally, there is the administrator, who has full permissions. They can manage and do everything on the website, as well as control parameters of the gym and employees.

Keywords

Spring Boot, Java, Thymeleaf, SQL, JPA, Java Script, web, online gym, onsite gym, gym's management

Índice

Índice de figuras

Capítulo 1

Introducción

Resumen: En este capítulo se detallan los motivos que llevaron a hacer este proyecto, así como los objetivos que se pusieron al comienzo, una explicación de como se ha llevado a cabo el trabajo y la base desde la que se partió.

1.1. Motivación

Arties nace de la idea de hacer el deporte de los gimnasios más accesible para todo tipo de personas, independientemente de factores como su tiempo, dinero o aficiones, haciendo que la experiencia pueda adaptarse a las necesidades de cada persona.

Esta aplicación está pensada para aquellas personas que por cualquier motivo les gustaría hacer ejercicio guiado por profesionales, pero no quieren o no pueden ir presencialmente a un gimnasio. Esto podría darse, por ejemplo, por problemas de tiempo, por no tener ningún gimnasio cerca de casa o por no tener dinero para una suscripción completa de un gimnasio. Además, a raíz del Covid-19, mucha gente ha empezado a disfrutar más del ejercicio en casa, sin tener que ir a gimnasios repletos de personas, en los que hay esperas para usar material, o en los que incluso puede llegar a ser incómoda la gran cantidad de gente en un mismo lugar. Para solucionar estos problemas, Arties presenta la idea de una suscripción *online*, que permitirá a los usuarios tener las ventajas de un gimnasio en su propia casa.

Por otro lado, hay muchas personas que no asisten a un gimnasio por no tener tiempo o dinero para su suscripción. Para afrontar este problema Arties contiene un sistema de reservas de clases abierto a todos los usuarios, tanto internos del gimnasio como externos. De esta forma se permite que personas con menos tiempo que no aprovecharían el coste de una suscripción completa puedan acudir solo a las clases que quieran, pagando únicamente por ellas. También permite que los

usuarios puedan probar diferentes clases y encontrar aquellas que les gusten más, al mismo tiempo que pueden experimentar de primera mano la calidad de las clases impartidas por el gimnasio. Todo esto hace las clases más accesibles, beneficiando tanto a usuarios por poder asistir a clases que de otra forma no hubieran podido, y al gimnasio, el cual puede obtener nuevos clientes.

Por último, existen muchos vídeos en internet sobre ejercicio, pero presentan tres grandes inconvenientes que esta aplicación solucionaría. Los vídeos pueden subirlos cualquier persona (sea más o menos profesional), lo que podría causar que los usuarios realizaran mal los ejercicios, provocando dolores o lesiones. Además, no hay un seguimiento del progreso del usuario ni atención a sus dudas en tiempo real. Por último, el usuario tendría que comprar todo el material necesario para seguir dichos vídeos. Todos estos problemas se pretenden solucionar mediante la aplicación, ya que los vídeos son subidos por los profesionales del gimnasio. Dichos profesores también podrán hacer un seguimiento de los usuarios *online* y responder a todas sus dudas, y por último, los usuarios podrán alquilar y utilizar el material del gimnasio sin necesidad de comprar muchos aparatos que terminarán utilizando solo una o dos veces.

1.2. Objetivos

El principal objetivo de este proyecto es el desarrollo de una aplicación web realista, segura, que sea realmente de utilidad para sus usuarios, con un aspecto visual moderno, similar a aplicaciones utilizadas en el día a día, y desarrollado con tecnologías web actuales.

Para ello la aplicación deberá proporcionar las funcionalidades necesarias para cubrir las necesidades descritas en el apartado 1.1 Motivación, además de otras que permitan la gestión de la web por parte del propietario y empleados del gimnasio.

Para lograr un aspecto moderno y entendible se prestará importancia a la interfaz de usuario, con el objetivo de mejorar la usabilidad y proporcionar una aplicación sencilla de usar y entender.

Para garantizar la seguridad se incluirán sistemas y comprobaciones que aseguren la autenticidad de los usuarios y la integridad de los datos, su confidencialidad y su disponibilidad.

Finalmente, se hará uso de tecnologías web actuales, para permitir así un desarrollo más eficiente y robusto, que permita la creación de una aplicación funcional, eficiente y adaptable.

1.3. Plan de trabajo

A lo largo del Proyecto se ha utilizado la metodología ágil Scrum, adaptada al tamaño del equipo, y haciendo uso de características de otras metodologías como es el tablero Kanban, el cual permite a los integrantes ver de forma rápida y visual el estado del proyecto. Debido a la estrecha coordinación, se han realizado prácticamente todos los Daily Scrums, además de una reunión más extensa todos los fines de semana. En algunas de estas reuniones se revisaba y comentaba el anterior Sprint con mayor profundidad, y se hacía el Sprint Planning del siguiente.

Para las reuniones extensas entre los miembros se ha utilizado Discord, mientras que con el director se ha usado Google Meet. Sin embargo, la mayoría de las reuniones diarias se han llevado a cabo en persona en las instalaciones de la facultad de informática de la Universidad Complutense de Madrid.

La organización de código se ha llevado a cabo con GitHub, el cual también ha servido para organizar las tareas e implementar el tablero Kanban usado, el cual se dividió en tres columnas: *To-do* (por hacer), *In progress* (en progreso) y *Done* (acabadas). A todas las tareas se les asignó una prioridad y una duración aproximada para una mejor organización y reparto.

A continuación se detalla qué se hizo en cada Sprint:

- **Sprint 1:** Se investigó la plantilla para el *back-end* realizada profesor Manuel Freire en su Asignatura de Ingeniería Web. Una vez comprendida lo mejor posible, se pasó a realizar los cambios propicios y borrar todo el código que no iba a ser usado. Por último, se preparó el código para dejar la estructura básica del *Back-end* y del *Front-end*.
- **Sprint 2:** Se diseñó la base de datos al completo, haciendo uso de diagramas y asegurando con el director que cumplía las tres formas normales. Tras esto, se implementó en código. Tras esto, se pobló la base de datos con algunos datos de prueba, y se hicieron conexiones básicas desde el *Front-end*.
- **Sprint 3:** Primero se creó toda la estructura de la web, a la vez que se investigaban elementos necesarios para posteriores implementaciones de código, como es el caso del calendario, las cartas o la *subnavbar*. También se implementó la gestión de empleados y una primera versión de gestión de material.
- **Sprint 4:** Además de correcciones sobre el Sprint anterior, también se implementó el chat entero y se acabó la versión definitiva de la gestión de material y alquileres.
- **Sprint 5:** Se codificó todo lo necesario para la gestión de suscripciones de los usuarios, además de la gestión de clases y del perfil de los usuarios.
- **Sprint 6:** Se implementó la gestión de cursos, además de corregir errores y mejorar aspectos visuales de la aplicación.

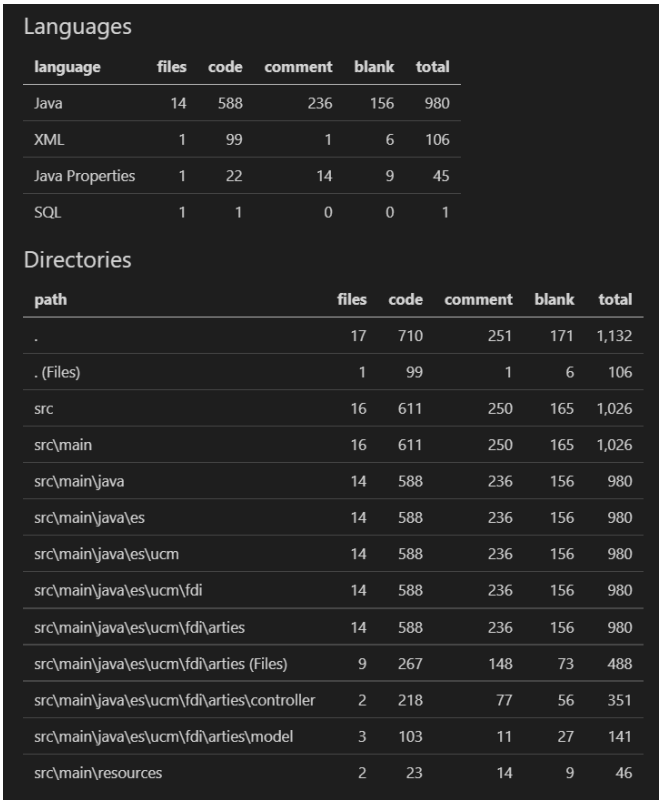
- **Sprint 7:** Este último periodo se dedicó especialmente a la memoria. Aunque la mayoría se fue documentando a la vez que se iba acabando, no fue el caso de todo. Además, se reescribieron algunos apartados y corregido otros. También se corrigieron aspectos del código, tanto funcionales como visuales.

1.4. Punto de partida

Este proyecto parte de una plantilla creada por el profesor Manuel Freire, para la asignatura de Ingeniería Web. La plantilla está subida en su repositorio GitHub (Freire, 2022c)[1]. Se tuvo que investigar su estructura y como se utilizaba, para poder usarla correctamente como base del proyecto. Por otro lado, es una plantilla principalmente para el *back-end*, pues lo fundamental es que tiene las bases de un proyecto con Spring Boot, JPA y H2 funcionando. En el *front end* destaca un método llamado *"go"* que es una especie de interfaz para simplificar el uso de AJAX. También es importante mencionar las funciones para inicializar y suscribirse a los *WebSockets*.

Se ha partido inicialmente de dicha plantilla, no obstante se han realizado numerosos cambios y aumentado el tamaño en gran medida. Para poner en contexto cuánto se ha trabajado se hace una comparación en líneas de código entre la plantilla de la que se partió y el trabajo final. Tras comprender, limpiar y preparar la plantilla para nuestro proyecto, se obtuvo un código de 1.132 líneas, como se ve en la figura 1.1. El proyecto final consta de 19.813 líneas de código, como se aprecia en la figura 1.2. Esto significa que entre los dos miembros del grupo se han escrito 18.681 líneas de código nuevas.

[1]Esta versión no aparece en el repositorio, pero si la de 2022-23 que es prácticamente igual.



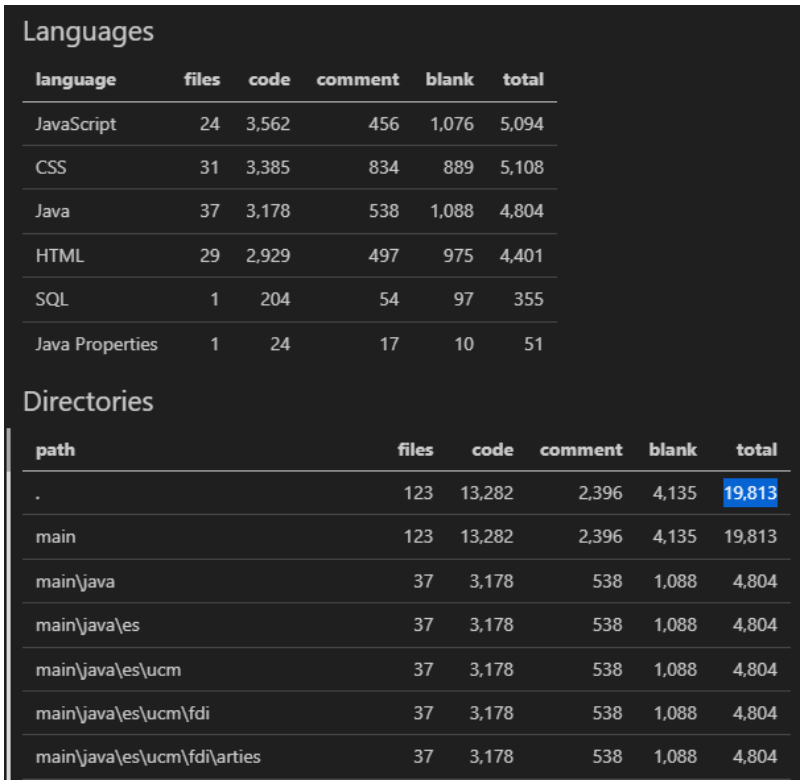
Languages

language	files	code	comment	blank	total
Java	14	588	236	156	980
XML	1	99	1	6	106
Java Properties	1	22	14	9	45
SQL	1	1	0	0	1

Directories

path	files	code	comment	blank	total
.	17	710	251	171	1,132
. (Files)	1	99	1	6	106
src	16	611	250	165	1,026
src\main	16	611	250	165	1,026
src\main\java	14	588	236	156	980
src\main\java\es	14	588	236	156	980
src\main\java\es\ucm	14	588	236	156	980
src\main\java\es\ucm\fdi	14	588	236	156	980
src\main\java\es\ucm\fdi\arties	14	588	236	156	980
src\main\java\es\ucm\fdi\arties (Files)	9	267	148	73	488
src\main\java\es\ucm\fdi\arties\controller	2	218	77	56	351
src\main\java\es\ucm\fdi\arties\model	3	103	11	27	141
src\main\resources	2	23	14	9	46

Figura 1.1: Líneas de código de la plantilla



Languages

language	files	code	comment	blank	total
JavaScript	24	3,562	456	1,076	5,094
CSS	31	3,385	834	889	5,108
Java	37	3,178	538	1,088	4,804
HTML	29	2,929	497	975	4,401
SQL	1	204	54	97	355
Java Properties	1	24	17	10	51

Directories

path	files	code	comment	blank	total
.	123	13,282	2,396	4,135	19,813
main	123	13,282	2,396	4,135	19,813
main\java	37	3,178	538	1,088	4,804
main\java\es	37	3,178	538	1,088	4,804
main\java\es\ucm	37	3,178	538	1,088	4,804
main\java\es\ucm\fdi	37	3,178	538	1,088	4,804
main\java\es\ucm\fdi\arties	37	3,178	538	1,088	4,804

Figura 1.2: Líneas de código del proyecto final

Introduction

Summary: In this chapter, we detail the reasons that led us to undertake this project, as well as the objectives we set at the beginning, an explanation of how we carried out the work, and the starting point we began from.

1.5. Motivation

Arties was born from the idea of making gym sports more accessible to all kinds of people, regardless of factors such as their time, money, or hobbies, making the experience adaptable to each person's needs.

This application is designed for those who would like to do exercises guided by professionals, but either don't want to or can't physically go to a gym for any reason. This could be due to time constraints, not having any gyms nearby, or not having enough money for a full gym subscription. Additionally, due to Covid-19, many people have started to enjoy exercising more at home, without having to go to crowded gyms where there may be waits for equipment or where the large number of people in one place can be overwhelming. To solve these problems, Arties presents the idea of an online subscription, which will allow users to have the benefits of a gym in their own home

On the other hand, many people do not attend a gym due to a lack of time or money for a subscription. To address this problem, Arties includes a class reservation system open to all users, both internal to the gym and external. This allows people with less time who would not take advantage of the cost of a full subscription to only attend the classes they want, paying only for them. It also enables users to try different classes and find those they like most, while simultaneously experiencing first-hand the quality of classes taught by the gym. All of this makes the classes more accessible, benefiting both users by being able to attend classes they otherwise could not, and the gym, which can gain new customers.

Finally, there are many exercise videos on the internet, but they have three major drawbacks that this application would solve. Anyone can upload videos (whether or not they are professionals), which could cause users to perform exercises incorrectly, resulting in pain or injury. Additionally, there is no tracking of the user's progress or real-time attention to their questions. Finally, the user would have to buy all the necessary equipment to follow these videos. All of these problems are intended to be solved by the application, as the videos are uploaded by gym professionals. These teachers will also be able to monitor users online and respond to all their questions, and finally, users will be able to rent and use gym equipment without having to buy many devices that will only be used once or twice.

1.6. Objectives

The main objective of this project is to develop a realistic web application that is truly useful for its users, with a modern visual aspect similar to applications we use in our daily lives, and developed with current web technologies.

To achieve this, the application should provide the necessary functionalities to cover the needs described in the section 1.5, as well as others that allow the management of the website by the gym's owner and employees.

To achieve a modern and understandable appearance, emphasis will be placed on the user interface with the goal of improving usability and providing an easy-to-use and easy-to-understand application.

To ensure safety, systems and checks will be included that guarantee the authenticity of users and the integrity of data, as well as their confidentiality and availability.

Finally, current web technologies will be used to enable more efficient and robust development, allowing the creation of a functional, efficient, and adaptable application.

1.7. Work plan

Throughout the project, the agile Scrum methodology has been used, adapted to the size of the team and making use of features from other methodologies such as the Kanban board, which allows team members to quickly and visually see the project's status. Due to the close coordination, almost all Daily Scrums were held, in addition to a more extensive meeting every weekend. In some of these long meetings, the previous Sprint was reviewed and discussed in more depth, and the Sprint Planning for the next one was done.

Discord has been used for the extended meetings between the members, while Google Meet has been used with the tutor. However, most of the daily meetings

have taken place in face-to-face at the facilities of the Computer Science faculty at the Universidad Complutense de Madrid.

Code organization has been carried out using GitHub, which has also served to organize tasks and implement the Kanban board used. The Kanban board was divided into three columns: To-do, In progress, and Done. All tasks were assigned a priority and an approximate duration for better organization and distribution.

What was done in each Sprint is detailed below:

- **Sprint 1:** The Back-end template developed by Professor Manuel Freire in his Web Engineering course was investigated. Once understood as best as possible, the necessary changes were made and all code that was not going to be used was deleted. Finally, the code was prepared to leave the basic structure of the Back-end and Front-end.

- **Sprint 2:** The complete database was designed, using diagrams and ensuring with the tutor that it met certain quality standards. After this, it was implemented with code. Then, the database was populated with some test data, and basic connections were made from the front-end to ensure everything was working correctly.

- **Sprint 3:** First, the entire web structure was created while researching necessary elements for future code implementations, such as the calendar, cards, or subnavbar. Employee management and a first version of material management were also implemented.

- **Sprint 4:** Apart from corrections from the previous Sprint, the entire chat was implemented and the final version of the material management was completed.

- **Sprint 5:** All the necessary code for managing user subscriptions, as well as class management and user profiles, was coded.

- **Sprint 6:** The course management was implemented, as well as corrected errors and improved visual aspects of the application.

- **Sprint 7:**: During this last period, we focused especially on the report. Although most of it was documented as it was completed, this was not the case for everything. In addition, some sections were rewritten and others were corrected. We also fixed things of the code, both functional and visual.

1.8. Starting point

This project starts from a template created by Professor Manuel Freire, for the Web Engineering course. The template is uploaded on his GitHub repository (Freire,

2022c)[2]. Its structure and use had to be investigated in order to properly use it as the base of the project. On the other hand, it is primarily a template for the back-end, as the key is that it has the foundations of a project with Spring Boot, JPA, and H2 functioning. On the front end, a method called "go" stands out as a kind of interface to simplify the use of AJAX. It is also important to mention the functions to initialize and subscribe to WebSockets.

Initially, the project was based on this template, but numerous changes have been made, and its size has increased significantly. To provide context on how much work has been done, a comparison is made in terms of lines of code between the template we started with and the final work. After understanding, cleaning, and preparing the template for our project, we ended up with 1,132 lines of code, as seen in figure 1.1. The final project consists of 19,813 lines of code, as shown in figure 1.2. This means that between the two members of the group, 18,681 new lines of code have been written.

[2]This version does not appear in the repository, but the 2022-23 one does, which is practically the same.

Capítulo 2

Estado de la Cuestión

Resumen: En este capítulo se analiza la competencia y las alternativas que hay en el sector en el que se encuentra Arties. Además, se explica brevemente qué diferencia a Arties de ellas.

2.1. Análisis de la competencia

Hacer ejercicio en casa o en el gimnasio es más importante que nunca, ya que por lo general llevamos vidas sedentarias, lo cual es negativo para la salud y hay que compensarlo con deporte. Debido a esto, ya se han creado muchas aplicaciones de gimnasios, deporte en casa, cursos... Así que en este apartado se explica lo que ya existe y en qué se diferencia Arties.

En primer lugar existen las *apps* de gimnasios como Body Factory[1], Reto 48[2] o Fitup[3]. Estas son usadas únicamente por los socios de los gimnasios correspondientes y son todas bastante parecidas. Desde la aplicación puedes ver qué clases hay, cuándo son y reservar una plaza si se desea y hay hueco. También suelen tener una página de ayuda, ya sea un chat o información de contacto vía *email* o teléfono. Todas tienen un perfil donde ver la información de la cuenta, y a veces algunos datos extras como altura o peso. Por último, y esto no lo tienen todas, se puede encontrar un apartado con información sobre la rutina que hayan especificado los entrenadores en el primer día de gimnasio. Cabe mencionar que algunos gimnasios tienen vídeos de entrenadores haciendo ejercicios para que el usuario les imite, sin embargo no suelen ser de demasiada calidad ni explicativos.

[1] `https://www.bodyfactory.es/`
[2] `https://reto48.es/`
[3] `https://fitupweb.es/`

Por otro lado están las *apps* de deporte en casa como Home Workout[4], Fitify[5], Hevy[6], 30 Day Fitness[7]... Estas no están asociadas a gimnasios, sino que van por su cuenta. En estas aplicaciones se suele encontrar una página con diferentes rutinas en las que se explica al usuario cómo hacer cada ejercicio, además de cronometrarle. También suele haber una página para el seguimiento de las rutinas, donde suele haber estadísticas de cuántos días las han realizado, supuestas kilocalorías quemadas, pasos realizados, seguimiento del peso... Cada vez es más común que también se incluya un apartado de recetas de comida saludables para complementar los entrenamientos y maximizar resultados. Por lo general estas *apps* son gratuitas, pero la mayoría de rutinas y algunas de las funcionalidades requieren pagar una suscripción *premium*.

Por último, aunque las *apps* de deporte en casa son una buena opción para hacer ejercicio sin dedicar mucho tiempo, la mayoría ofrecen rutinas que no requieren material especializado. Esto está bien, pero acaban siendo siempre los mismos ejercicios, lo cual puede resultar repetitivo y aburrido, ya que no hay excesivas opciones para variar. Para solucionar esto, se han buscado webs que alquilen material de gimnasio para casa, sin embargo no existen. Las webs existentes, como Fitnessrent[8] o Valfit[9] son para alquiler de máquinas pesadas (cintas, bicicletas, remos...) para clientes grandes como hoteles, eventos, comunidades de vecinos. Además tienen un propósito más permanente ya que la unidad normal de alquiler es el año. El resto de negocios que sí tienen material con más sentido para las casas del público general como pesas, bandas elásticas, esterillas, pelotas de goma... no lo alquilan, simplemente lo venden.

2.2. Aspectos diferenciadores

Una vez investigado todo lo comentado en la sección 2.1, se ha llegado a la conclusión de que existen bastantes *apps* con funcionalidades parecidas a las que ofrece Arties. Sin embargo, todas están separadas, no hay ninguna que incluya la mayoría o todas ellas. Además, como se ha mencionado, el alquiler de material de gimnasio ligero no existe, y los gimnasios no suelen ofrecer sus propios cursos para hacer en casa. Aquí es donde Arties se diferencia. Aparte de ofrecer lo que ofrecen todos los gimnasios, les ayuda a expandirse y llegar a más personas con los servicios que ofrece de gimnasio *online* (cursos, alquiler de material, ayuda por chat, seguimiento de los cursos...).

Además, solo por ahora solo se ha mencionado lo que ve el cliente, pero la aplicación también integra lo necesario para que el dueño pueda manejar su negocio. Este

[4] https://play.google.com/store/apps/details?id=homeworkout.homeworkouts.noequipment

[5] https://play.google.com/store/apps/details?id=com.fitifyworkouts.bodyweight.workoutapp

[6] https://play.google.com/store/search?q=hevy&c=apps

[7] https://play.google.com/store/apps/details?id=com.popularapp.thirtydayfitnesschallenge

[8] https://www.entrenamientoyfitness.com/

[9] https://ventalquilerequiposfitness.com/

puede gestionar sus empleados, cursos, clases, material, suscripciones y alquiler, todo desde el mismo lugar. Por otro lado, los empleados podrán llevar a cabo su trabajo de la mejor manera posible, con una interfaz moderna, manejable y facilitando el responder las dudas de los usuarios.

Capítulo 3

Tecnologías

Resumen: En este capítulo se describen las tecnologías que se han utilizado para el desarrollo de la aplicación, así como cual ha sido su utilidad en el proyecto. Por ultimo se hace un resumen de como se realiza la ejecución de la aplicación, ya que esta depende de las tecnologías utilizadas.

3.1. Tecnologías empleadas

En este apartado se hablan de los diferentes lenguajes y tecnologías utilizadas en el desarrollo de la aplicación, destacando sus ventajas y su utilidad.

3.1.1. Spring Boot

El primer paso al desarrollar la aplicación fue elegir que tecnología se utilizaría para desarrollar el *backend*, ya que esto influye en el resto de elementos que se tienen que usar.

Tras investigar diferentes opciones como el uso de Node.js, o PHP con algún *framework* como Larabel, finalmente nos decidimos por utilizar Spring Boot.

Spring Boot es un *framework* diseñado para el desarrollo con Java como lenguaje de programación y empleado para el desarrollo *back-end* (Spring, s.f.b). Se trata de un entorno de desarrollo de código abierto y gratuito. Utilizar este *framework* ofrece una serie de ventajas en el desarrollo de aplicaciones web, como pueden ser las siguientes:

- **Facilidad de configuración:** Proporciona una configuración fácil y rápida, pudiendo dar más importancia y tiempo a la parte del desarrollo en sí.

- **Integración de tecnologías:** Se integra bien con muchas tecnologías diferentes como por ejemplo JPA, Redis, MongoDB, etc. Por tanto ofrece la ventaja de poder añadirle nuevos componentes de forma sencilla y sin preocuparse tanto por la compatibilidad.
- **Patrones de diseño:** Sigue patrones de programación (como por ejemplo el conocido patrón MVC) que permiten un desarrollo de un código mantenible y escalable.
- **Documentación y comunidad:** Al ser una tecnología ampliamente utilizada dispone de una buena documentación y muchos usuarios que ofrecen su ayuda y conocimiento. Esto es de utilidad para poder resolver fácilmente los problemas que vayan surgiendo.

Se decidió utilizar Spring Boot como *framework* para el *back-end* por todas esas ventajas que ofrecía y por algunos motivos más. El primero, es que se trata de una tecnología bastante utilizada hoy en día, siendo muchas webs desarrolladas bajo este *framework*, cumpliendo así uno de los objetivos como era el utilizar tecnologías que se usaran realmente en el desarrollo de *apps*. Por otro lado, al utilizarlo se debe implementar el *back-end* utilizando Java. Este es un lenguaje robusto, muy utilizado y conocido, siendo por tanto un buen punto a favor.

3.1.2. Spring Security

Al utilizar Spring Boot se pudo usar Spring Security, un marco utilizado para el desarrollo de aplicaciones seguras (Security, s.f).

La seguridad es un tema muy importante en el día de hoy, y más aún en una aplicación comercial como la propuesta, que además contiene datos personales de los usuarios. Por tanto, la aplicación debería impedir el acceso a determinados contenidos a los usuarios que no tengan permisos y restringirles ciertas funcionalidades. Por ejemplo, se debe evitar que un atacante pueda modificar datos del gimnasio como el material del que dispone, o los cursos que ofrece. Del mismo modo también se debe evitar que una persona pueda hacerse pasar por otro usuario y acceder a sus datos o hacer acciones en su nombre.

Todo esto relativo a la seguridad puede ser desarrollado fácilmente mediante la integración de Spring Security. Este ofrece un sistema de restricción de accesos personalizable mediante roles de usuarios, pasando todas las peticiones al servidor por su filtro. De esta forma se evita que usuarios sin permisos accedan a funcionalidades que no deberían, véase la gestión del gimnasio.

Por otro lado, implementa un sistema de autenticación con el uso de tokens únicos generados con el *login* que sirven para autenticar a los usuarios y asegurar un acceso correcto a los datos de la aplicación. Así se consigue que un atacante no pueda hacerse pasar por otro usuario.

Por último, también ofrece protección frente ataques csrf y xss, bastante conocidos en el ámbito web.

3.1.3. Java

Este ha sido el principal lenguaje de programación utilizado, principalmente debido a la elección de Spring Boot como *framework*.

Se trata de un lenguaje con bastante recorrido, pues lleva en uso desde 1996, y se trata de uno de los lenguajes más utilizados a día de hoy, ocupando el tercer puesto de los más utilizados.

Es un lenguaje orientado a la programación con objetos, lo que permite desarrollar un código modular, reutilizable, comprensible y escalable (Oracle, s.f.). Además, existe una gran comunidad de desarrolladores que utilizan este lenguaje, permitiendo resolver más fácilmente dudas y problemas.

Además, presenta utilidades para el manejo de fechas, ficheros y eventos que fueron de utilidad en el desarrollo de la aplicación.

3.1.4. Tecnologías web

Para el desarrollo de la interfaz se han utilizado HTML para la estructura, CSS para el formato y estilo, y JavaScript para la detección de eventos en la interfaz y la interacción con el usuario.

Para una mejor organización y mantenibilidad se ha separado el código en diferentes archivos permitiendo así también una mejor modularidad y reutilización de componentes. Por ejemplo, se han creado archivos HTML para la cabecera y el footer que se incluyen en todas las páginas. De la misma manera, existe un archivo JavaScript y CSS común a todas las vistas, además de los propios de cada vista.

Para el apartado visual se optó por utilizar la librería de Bootstrap. Este fue desarrollado por Twitter y aporta componentes y funcionalidades para un desarrollo de webs *responsive* y con un estilo visual moderno (Bootstrap, s.f.). Su uso permitió implementar la organización del *layout* y hacerlo *responsive*, además de utilizar componentes que se usan en aplicaciones reales como desplegables, *cards*, *navbars*, modales, etc.

3.1.5. Thymeleaf

Thymeleaf es un motor de plantillas de código abierto que se utiliza para generar HTML, XML, JavaScript, CSS y otros tipos de archivos de texto a partir de plantillas. Además, puede integrarse fácilmente con Spring Boot (Thymeleaf, 2022).

Se ejecuta en el lado del servidor, lo que significa que las plantillas se procesan en el servidor y se generan los archivos de salida correspondientes que se envían al navegador del usuario.

Utiliza una sintaxis muy similar a HTML facilitando así el desarrollo. Usando Thymeleaf se puede acceder a los valores cargados por el controlador del servidor, y mostrarlos en el HTML, permitiendo así formar fácilmente archivos HTML con contenidos dinámicos desde el servidor.

3.1.6. AJAX

AJAX (*Asynchronous JavaScript And XML*) se trata de una tecnología que permite realizar peticiones a un servidor y actualizar cierto contenido de la página web sin necesidad de recargarla entera. Es básicamente un uso conjunto de las funcionalidades ofrecidas por el navegador y JavaScript (W3Schools, s.f.).

Su funcionamiento como se indica en el diagrama de la figura 3.1 consiste en lo siguiente:

- **1.** Mediante JavaScript se asocian ciertos elementos de la interfaz a eventos, los cuales se ejecutan al interactuar el usuario con ellos. Dichos eventos realizan una petición asíncrona al servidor.
- **2.** El servidor recibe la petición, la procesa y devuelve un resultado, el cual puede ir desde un texto plano hasta una lista compleja de objetos en formato JSON.
- **3.** El cliente recibe la respuesta, y en función de esta, actualiza la interfaz con los datos recibidos.

El uso de AJAX en aplicaciones actuales es muy frecuente, ya que permite hacer peticiones y modificar únicamente los elementos de la interfaz que deban cambiar. Esto evita que se tenga que recargar la página web por completo, aportando así una mayor velocidad y menos carga de procesamiento tanto al servidor como al cliente.

Para implementarlo en la aplicación se ha hecho uso de la utilidad FETCH que proporciona directamente JavaScript y que permite hacer diferentes tipos de peticiones a la URL que se indique y con los datos que se quieran (Mozilla-Developer-Network, 2023). Está basado en un funcionamiento asíncrono, lo que permite esperar al resultado del servidor y una vez obtenida la respuesta, ejecutar un código que la procese y actualice la interfaz.

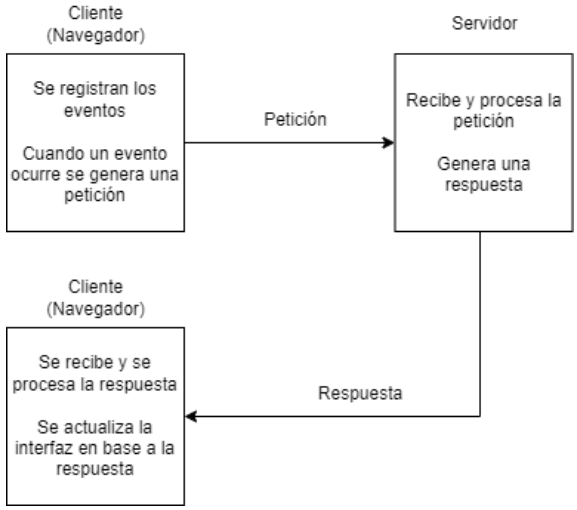


Figura 3.1: Diagrama de AJAX

3.1.7. STOMP y WebSockets

Para que la aplicación sea más realista y útil, es necesario que en algunos casos implemente interacciones en tiempo real, como es el caso del chat. Para lograr esto se ha utilizado Stomp.js y Spring WebSockets (Freire, 2022a), dos tecnologías típicas para el desarrollo de aplicaciones web.

Stomp.js es una librería de JavaScript que proporciona una interfaz simple para comunicar los nodos que estén usando STOMP (*Simple Text Oriented Messaging Protocol*). STOMP es un protocolo orientado a mensajes de texto que trabaja sobre *sockets* TCP, permitiendo mandarlos asíncronamente. Implementa cabeceras, suscripciones y confirmaciones para aumentar la eficiencia y la seguridad.

Para el *back-end* se ha utilizado Spring WebSockets, un módulo de Spring Framework, que se integra perfectamente con los mensajes del protocolo STOMP. Al ser un módulo de Spring, se complementa adecuadamente con el resto, aportando una mayor seguridad y comodidad de uso.

3.1.8. JPA y H2

Para almacenar los datos de la aplicación se decidió implementar una base de datos utilizando H2 y JPA.

H2 es una base de datos relacional escrita en Java y que destaca por su portabilidad, facilidad de uso y rendimiento. Utilizar H2 permite crear una base de datos de forma sencilla y con buenos resultados (Engine, s.f.).

Para realizar la interacción con la base de datos se ha utilizado JPA. Este proporciona una serie de interfaces y anotaciones que se utilizan para definir la relación

entre las clases Java y las tablas de la base de datos, así como para configurar el comportamiento de la persistencia (Hibernate, s.f.). Por tanto mediante su uso se puede indicar como se van a convertir las clases java que se utilizan, en entidades de la base de datos para poder realizar su almacenamiento. También ofrece soporte para las transacciones, lo cual permite realizar cambios y consultas sobre la base de datos de forma simple y segura, garantizando la consistencia.

3.1.9. GitHub

Para poder desarrollar el trabajo en paralelo de la aplicación se ha hecho uso de GitHub. Este permite gestionar proyectos software y proporciona control de versiones mediante Git (GitHub, s.f.).

Gracias su uso se puede trabajar de forma simultánea en diferentes sistemas de la aplicación, juntando después los cambios de forma sencilla y rápida. Además, gracias a su control de versiones se tiene un historial de todos los cambios hechos, pudiendo volver a una versión anterior del proyecto si fuera necesario.

Por otro lado, también incluye herramientas útiles para la organización del proyecto. En este caso se ha hecho uso del tablero Kanban, el cual permite crear las diferentes tareas que se deben realizar en el proyecto (GitHub, s.f). Estas se organizan en una tabla indicando cuales están pendientes de hacer, en progreso o terminadas.

3.1.10. Visual Studio Code

Para el desarrollo del código y la implementación del proyecto se ha utilizado Visual Studio Code. Este se trata de un editor de código, que puede resultar altamente versátil gracias a los *plugins* y extensiones que se pueden instalar para ampliar su funcionalidad (Microsoft, s.f.). Esto ha permitido facilitar el desarrollo del proyecto, pues ha permitido desarrollar de forma cómoda en una única herramienta, código en diferentes lenguajes, como puede ser Java, SQL, JavaScript, HTML o CSS. Gracias al uso de estas extensiones, Visual Studio Code proporciona ayuda a la programación (como sugerencias, tabulados, estructura, etc.) para todos ellos.

3.2. Puesta en marcha

Esta aplicación implementa diversas tecnologías, por lo que la configuración del ordenador para poder probar el funcionamiento no es trivial. Por ello se expone en esta sección un tutorial para poder probarla en Windows. Este tutorial está basado en uno creado por Manuel Freire en la carpeta “qna” de su GitHub (Freire, 2022b).

En primer lugar hay que descargar Visual Studio Code e instalar las siguientes extensiones: Extension Pack for Java y Lombok Annotations Support for VS Code.

El siguiente paso es descargar el OpenSDK 11 de Java y la versión 3.8 de Maven. Para poder usar estos desde la terminal de comandos hay que seguir los siguientes pasos:

- Buscar "editar las variables de entorno del sistema" en la barra de búsqueda de Windows. Esto abrirá la ventana de "Propiedades del sistema".
- Hacer clic en "Variables de entorno...", tal y como se ve en la figura.3.2.
- Seleccionar la variable "Path" en el bloque de "Variables del sistema" y clicar en "Editar", tal y como se muestra en la figura 3.3.
- Una vez dentro de la variable "Path" (figura 3.4), crear una nueva ruta pulsado en "Nuevo" y añadir la ruta donde se encuentre la carpeta "bin" del JDK11.
- Repetir el paso anterior pero agregando la ruta de la carpeta "bin" de Maven.
- Por último, crear o editar la variable "JAVA_HOME", dentro del bloque de "Variables de usuario", esta variable debe apuntar a la ruta donde se encuentre el JDK11, es decir, la dirección utilizada anteriormente, pero quitando el "\bin".

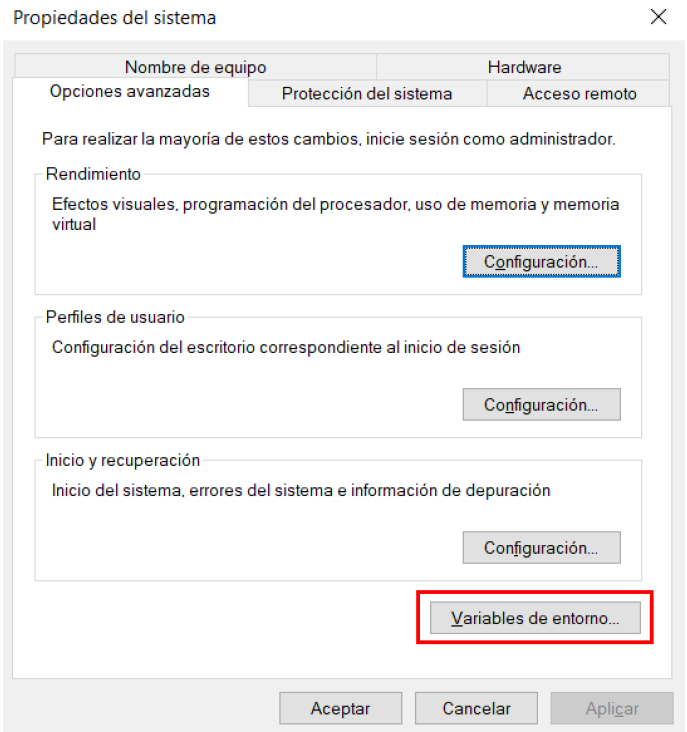


Figura 3.2: Propiedades del sistema

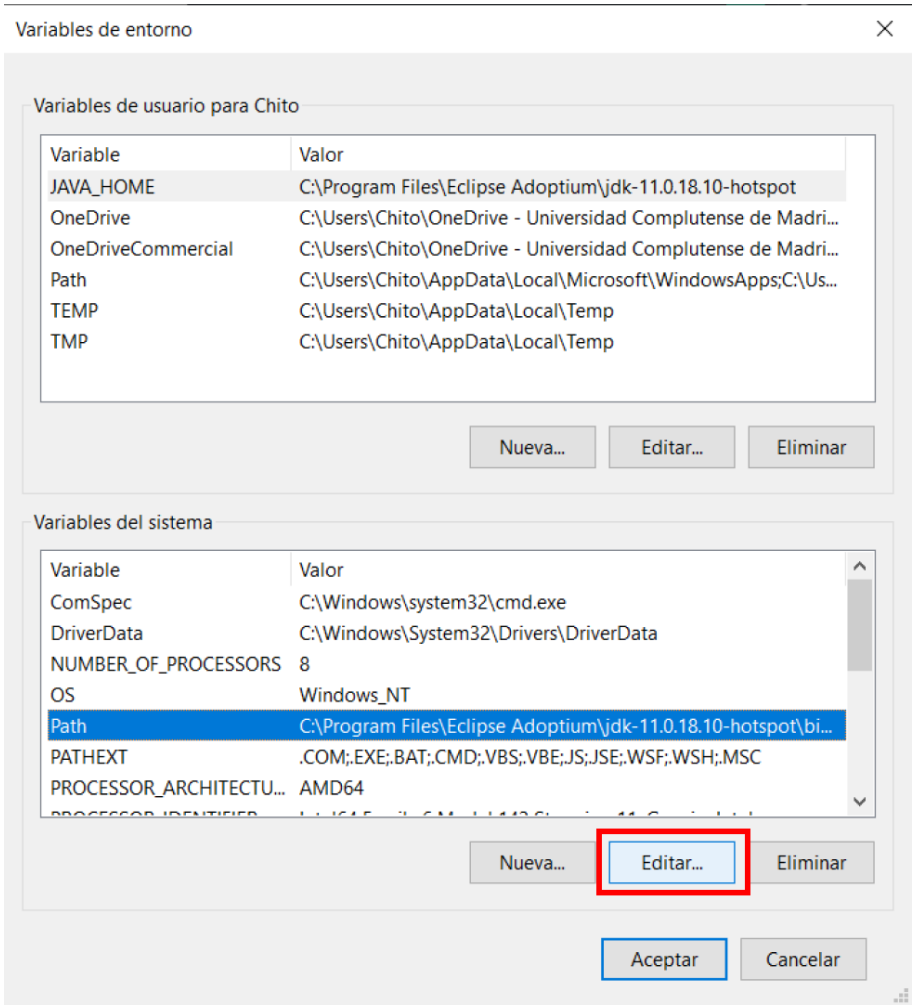


Figura 3.3: Variables de entorno

Ahora ya está configurado el entorno. Solo queda descargar el código, abrir el proyecto con Visual Studio Code y abrir una terminal dentro del proyecto. En la terminal, situarse en la dirección del proyecto y ejecutar el comando: mvn spring-boot:run. Con esto ya estará la web funcionando en la URL: http://localhost:8080/

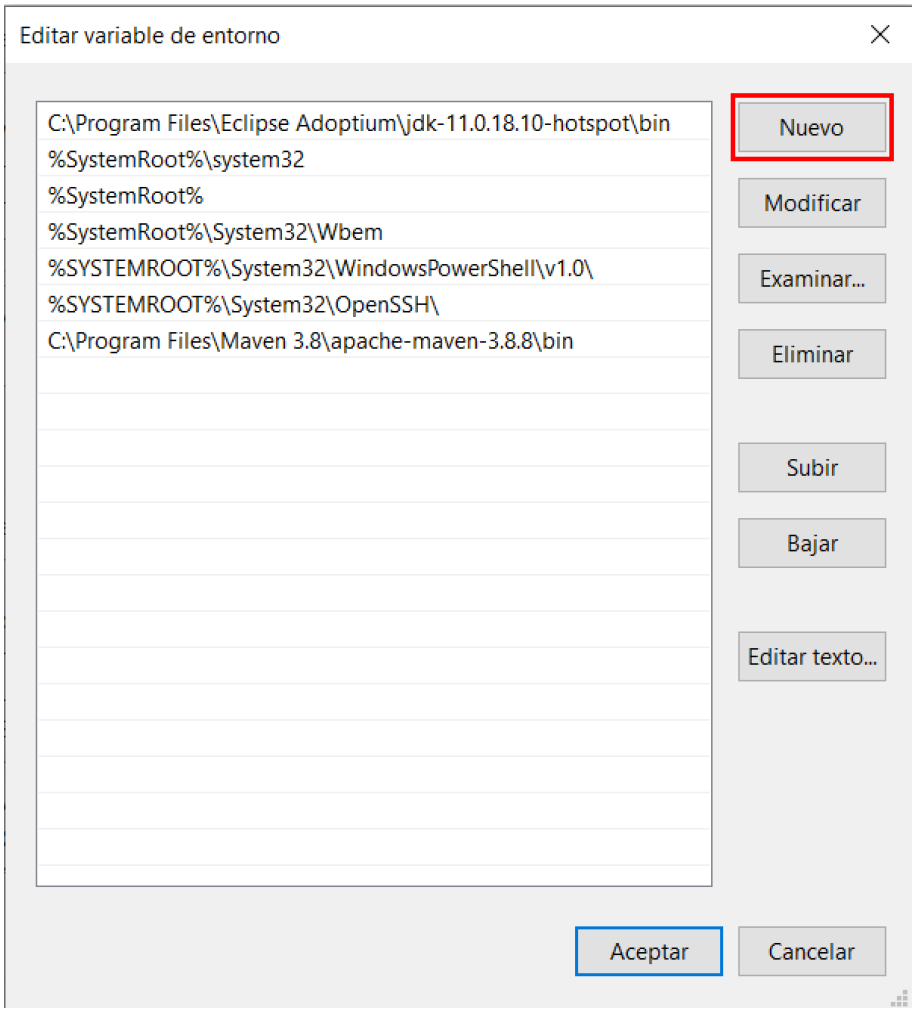


Figura 3.4: Editar la variable Path

Capítulo 4

Arquitectura y modelo de datos

Resumen: En este capítulo se detalla cómo se ha diseñado la estructura de la aplicación, teniendo en cuenta la arquitectura, patrones de programación y diseño de la base de datos.

4.1. Subsistemas

Para poder realizar un desarrollo eficiente y poder trabajar en diferentes apartados de la *app* en paralelo, se dividieron las funcionalidades a implementar en diferentes subsistemas. De esta forma mientras un miembro del grupo esta desarrollando cierta funcionalidad, el otro puede estar trabajando en paralelo en otra sin que haya problemas posteriores para unificar las versiones. Dado los requisitos de la *app* se dividió el sistema en diferentes subsistemas como se puede ver en la figura 4.1.

- **Sistema de material:** Es el encargado de gestionar todos los datos relacionados con el material, así como la funcionalidad del alquiler de material.
- **Sistema de clases:** Se encarga de la gestión de las clases que ofrece el gimnasio además de permitir reservas plazas para asistir a las clases.
- **Sistema de cursos:** Gestiona toda la información referente a los cursos *online* que ofrece el gimnasio, incluyendo la creación y modificación de cursos.
- **Sistema de chat:** Este sistema incluye toda la lógica para implementar un chat a tiempo real entre usuarios y empleados.
- **Sistema de gestión de usuarios:** Su función es restringir los permisos de los usuarios, así como la creación y modificación de nuevos usuarios.

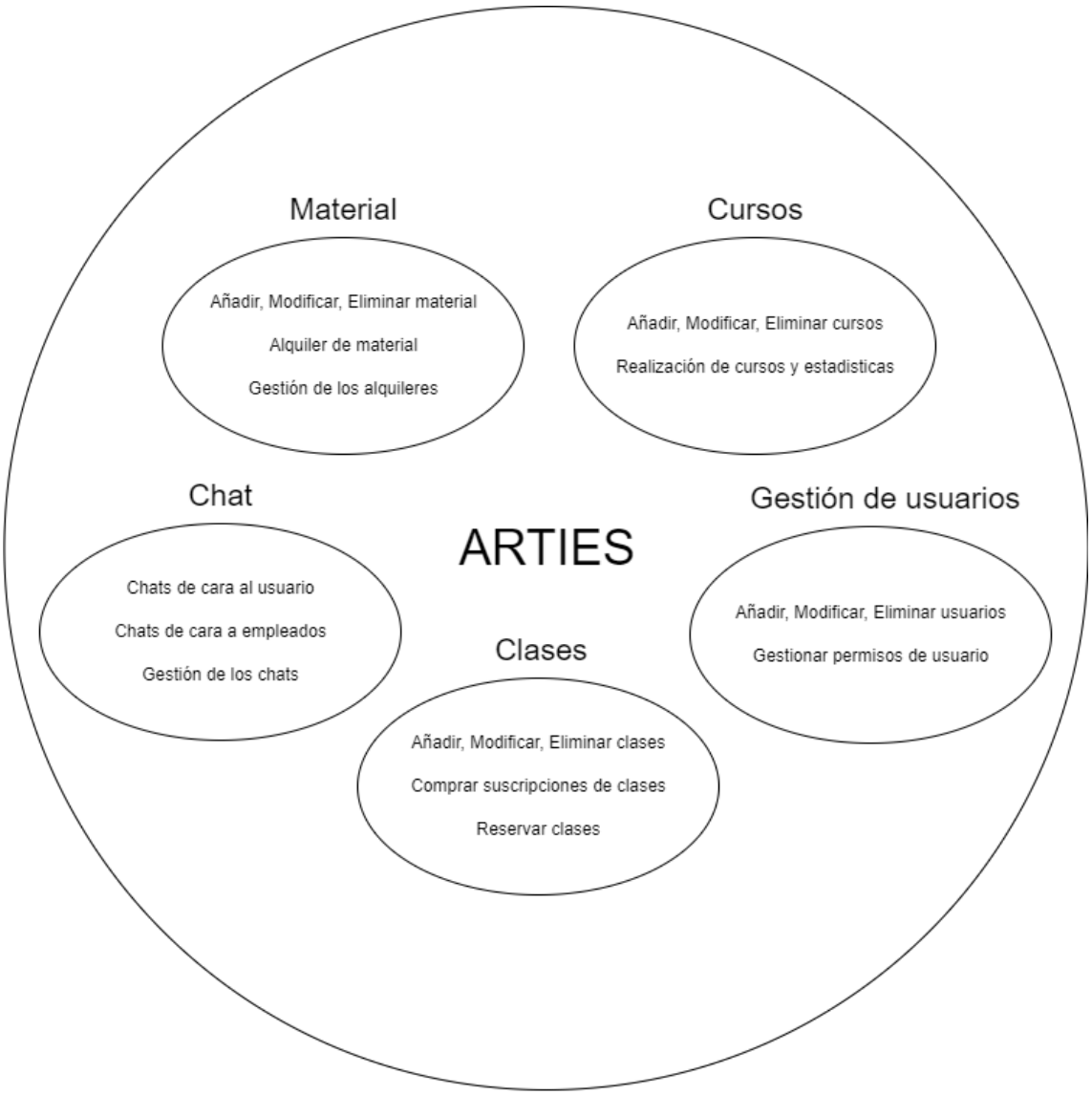


Figura 4.1: Subsistemas de Arties

4.2. Arquitectura de la aplicación

Aprovechando el uso de las tecnologías empleadas como Spring y Thymeleaf se ha optado por implementar el patrón MVC como arquitectura principal de la aplicación.

Este es un patrón muy utilizado para el desarrollo de aplicaciones web debido a los beneficios que aporta su uso, entre los que destaca el desarrollo de un código escalable y mantenible.

Este patrón define como se deben organizar y estructurar los diferentes componentes que forman el sistema, además de sus responsabilidades y la forma en la que se deben relacionar unos con otros. De esta forma se especifican 3 apartados esen-

ciales en los que se agrupan los diferentes elementos del sistema. Dichos apartados son los siguientes:

- **Modelo:** Esta capa incluye el modelo de datos con el que trabaja la aplicación. También incluye la lógica de negocio, así como la gestión del almacenamiento de los datos.
- **Vista:** Son aquellos componentes encargados de generar la interfaz que representan los datos del modelo. Además, contiene aquellos elementos interactuables que permiten al usuario realizar cambios sobre la aplicación.
- **Controlador:** Actúa como intermediario entre el usuario y el sistema. Captura aquellas interacciones del usuario con la vista, interpretándolas y actuando en función de estas.

De esta forma como se puede ver en la figura 4.2 el modelo contiene la lógica de la aplicación. EL usuario interacciona con la vista. El controlador registra dicha interacción y actúa en base a ella, realizando los cambios que sean necesarios sobre el modelo, y pudiendo mostrar los resultados obtenidos comunicándoselos a la vista.

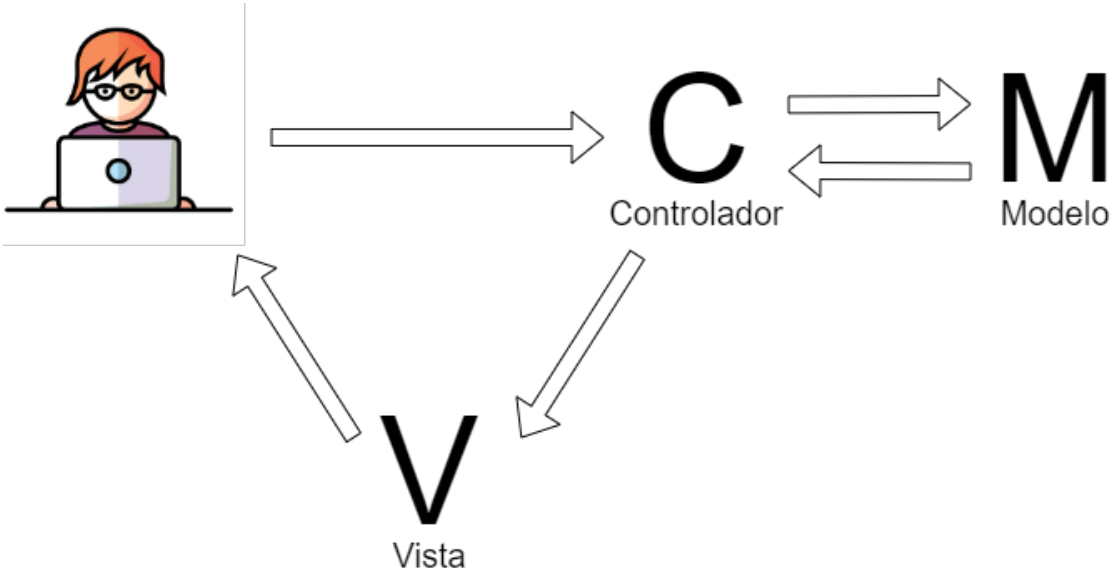


Figura 4.2: Esquema MVC

En el caso de la aplicación desarrollada se puede ver claramente la diferencia entre estas tres partes. Por un lado, todos los archivos HTML, CSS y Javascript son parte de la Vista, y se encargan de mostrar los datos al usuario y de reaccionar ante sus interacciones. Por otro lado, los controladores reciben las peticiones del usuario, las validan y notifican tanto al modelo para que realice las acciones necesarias, como a la vista para que actualice los datos que debe mostrar. Por último, el modelo se corresponde con todas las clases Java que definen los datos de la aplicación y su funcionamiento.

Esta división permite separar claramente las responsabilidades y funcionalidades de cada componente del sistema. De esta forma se garantiza un código mas mantenible y modificable, ya que permite editar componentes concretos sin afectar al funcionamiento del resto de la aplicación.

Por último, es importante destacar que se ha implementado cierta subdivisión a dicho patrón que ha permitido que el trabajo en paralelo de los dos miembros fuera más eficiente. Para ello se ha definido un controlador diferente por cada subsistema de los explicado en el apartado 4.1 Subsistemas.

De esta forma, cada controlador solo contiene la lógica necesaria para modificar los datos y las vistas de dicho subsistema sin interferir en los demás.

Se ha incluido también un controlador común para aquellas páginas que no pertenecen a ningún subsistema (como por ejemplo inicio) o para aquellas que pertenecen a varios (por ejemplo, la página de configuración ofrece funcionalidades sobre todos los subsistemas)

4.3. Base de datos

Uno de los primeros pasos más importantes fue definir que datos necesita almacenar la aplicación para poder ofrecer las funcionalidades propuestas.

Se identificó que se necesitaban datos sobre los diferentes elementos con los que trabaja la *app*, como por ejemplo, datos sobre usuarios, datos sobre objetos, datos sobre cursos, etc.

Tras definir dichos elementos se procedió a definir las relaciones que existían entre estos, tales como, que un usuario alquila objetos, o que un curso pertenece a una categoría. Finalmente, tras estudiar todas las posibles relaciones se obtuvo el siguiente diagrama de Entidad-Relación (figura 4.3) que define la estructura de la base de datos.

Las partes más destacables del diagrama son las relaciones entre las diferentes entidades, que se describen más en detalle a continuación:

- Relación (*loan*) entre usuarios (*User*) y objetos (*Items*): Un usuario puede alquilar varios *items*, y un *item* al disponer de varias unidades, puede ser alquilado por varios usuarios. Esto da lugar a una relación N-N en la que además se deben almacenar los datos del alquiler (fechas de inicio y fin, y cantidad alquilada).
- Relación (*message*) entre usuarios (*User*) y usuarios (*User*): Un usuario puede mandar varios mensajes a diferentes usuarios. Otro usuario puede recibir mensajes de varios usuarios distintos. Por cada mensaje mandado se debe almacenar el id que lo identifica, su texto, asunto y si lo mando un usuario o un

empleado, ya que normalmente la comunicación sera entre usuarios normales y usuarios empleados. Esto da lugar a una relación N:N.

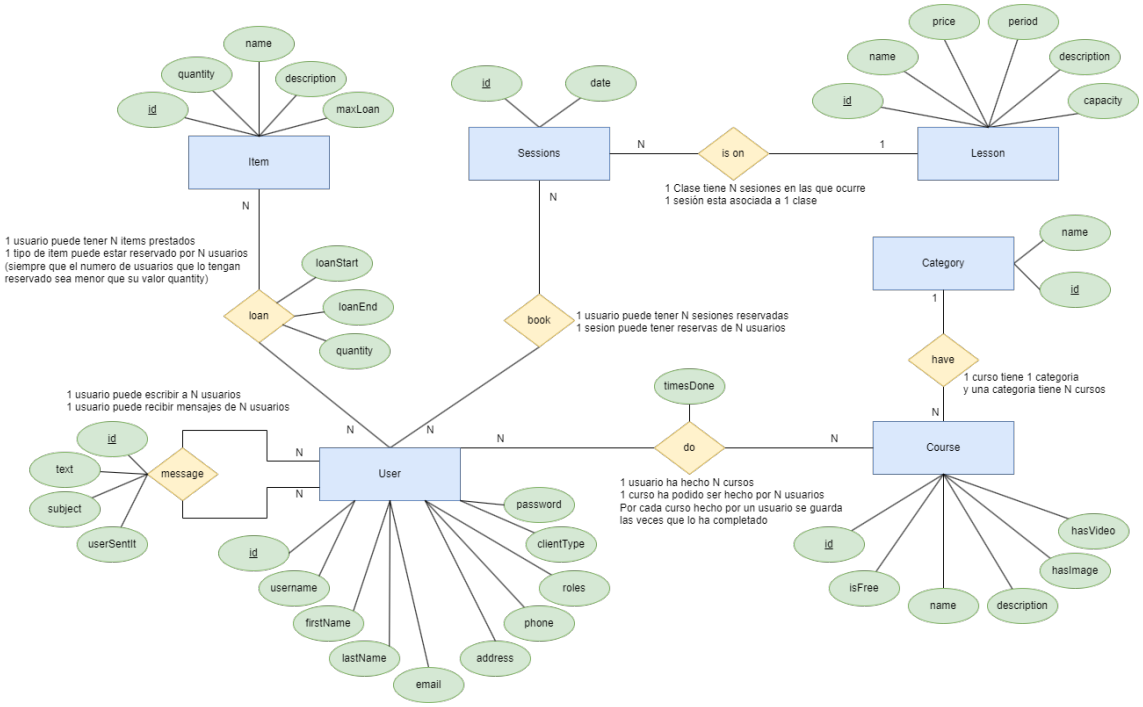


Figura 4.3: Modelo Entidad-Relación

- Relación (*do*) entre usuarios (*User*) y cursos (*Course*): Un usuario puede hacer varios cursos, y un curso puede estar hecho por varios usuarios. Esto define una relación N-N en la que además se guarda cuantas veces hizo el curso cierto usuario, información que será de utilidad para mostrar estadísticas.

- Relación (*have*) entre cursos (*Course*) y categorías (*Category*): Un curso pertenece a una sola categoría y una categoría puede tener varios cursos. Esto da lugar a una relación 1:N que permite una mejor organización de lo cursos al poder agruparlos por categorías.

- Relación (*is on*) entre clases (*Lessons*) y sesiones (*Sessions*): Una clase ocurre periódicamente los mismos días de la semana y a las mismas horas. Las sesiones se tratan del día y hora concretos a los que ocurren las clases. Por tanto una clase tiene varias sesiones en las que ocurre y una sesión concreta esta asociada a una única clase. Así se crea una relación 1:N.

- Relación (*book*) entre usuarios (*User*) y sesiones (*Sessions*): Un usuario puede reservar una plaza para varias sesiones de clases. De la misma manera, dada una sesión, esta puede ser alquilada por varios usuarios. Se genera así una relación N:N

Tras diseñar el anterior diagrama Entidad-Relación se procedió a realizar el diseño en sí de la base de datos, traduciendo dichas entidades y relaciones a tablas que se

almacenan en el servidor. El resultado final se puede ver en el esquema relacional de la figura 4.4.

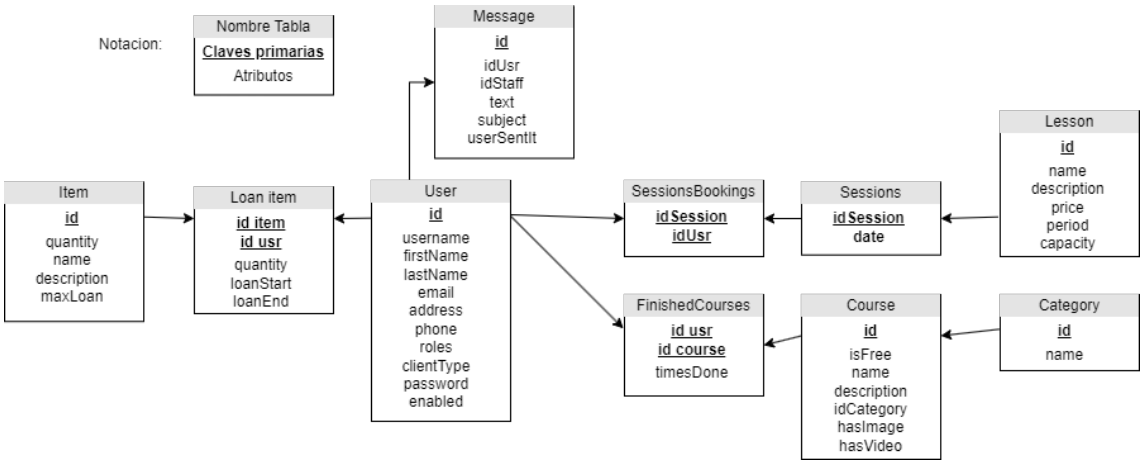


Figura 4.4: Tablas base de datos

Analizando este resultado se puede comprobar que es correcto y que cumple la tercera forma normal de las bases de datos, ya que se cumple que cada tabla contiene una clave primaria única que identifica una línea, todas las columnas de las tablas dependen únicamente de la clave primaria y además no existen relaciones de transitividad entre columnas, dependiendo estas directamente de la clave primaria.

Esta base de datos fue integrada en el proyecto mediante el uso de JPA, con el cual, mediante las anotaciones que proporciona y que se usaron en las diferentes entidades de la aplicación, generó una base de datos con la estructura propuesta.

Finalmente, desde el punto de vista de la seguridad se han realizado dos acciones para garantizar la integridad de la base de datos y la seguridad de los datos almacenados. En primer lugar, todas las contraseñas se almacenan cifradas mediante el uso de BCrypt. Este paquete aporta funcionalidades que permiten realizar el cifrado de textos, personalizar dicho cifrado mediante el uso de *salt* y comparar textos cifrados (SpringSecurity, s.f.). Gracias a su uso las contraseñas no son expuestas y se almacenan de forma segura. Por otro lado, se han tenido también en cuenta los ataques de inyección SQL. Para ello las consultas que dependen de datos que introdujo el usuario son escapadas con el fin de evitar que se incluyan acciones maliciosas a realizar en la base de datos.

Capítulo 5

Descripción del funcionamiento de Arties

Resumen: En este capítulo se explican todas las funcionalidades que implementa Arties, además de su funcionamiento sin entrar mucho en detalles de programación. Destacar que al haber dado mucha importancia al diseño, todos los apartados van acompañados de las respectivas capturas de pantalla.

5.1. Inicio

La página de inicio cobra especial importancia debido a que es la primera impresión que recibe un usuario al utilizar la aplicación. Se ha utilizado un diseño sencillo pero llamativo que capte la atención del usuario. El principal objetivo de la página de inicio es presentar al usuario que funcionalidades le ofrece la aplicación.

De esta forma se estructura en tres grandes secciones que dan información sobre los apartados de cursos, material y clases respectivamente. Respecto a los cursos se ha optado por utilizar un carrusel que muestre una selección de 5 cursos gratuitos ofrecidos por el gimnasio. En cada curso aparece una descripción al pasar el ratón por encima para que el usuario pueda saber fácilmente en que consiste el curso y si le interesa o no. Mediante un click puede ir directamente al curso y así explorar que le ofrece. Por último, el carrusel está animado para cambiar automáticamente cada cierto tiempo el curso mostrado, incentivando así al usuario a que vea alguno de los cursos al destacarle esta información.

La siguiente sección muestra información sobre el sistema de alquileres. Aquí se presentan dos objetos que posee el gimnasio, junto a un apartado que describe el funcionamiento de dicho sistema. Se incluye además un botón que invita al usuario a ver todos los objetos disponibles del gimnasio.

En el siguiente apartado se puede encontrar una lista de 4 clases que ofrece el

gimnasio de forma presencial. Tras ello, se encuentra una descripción que explica el funcionamiento del sistema de reservas junto a un botón que permite ver todas las clases que ofrece el gimnasio.

En todas las secciones se ha optado por un fondo animado con degradados de tonos azules, que sin ser demasiado llamativos, sirven para captar mejor la atención del usuario. El resultado final de esta pantalla se puede ver en las figuras 5.1 y 5.2.

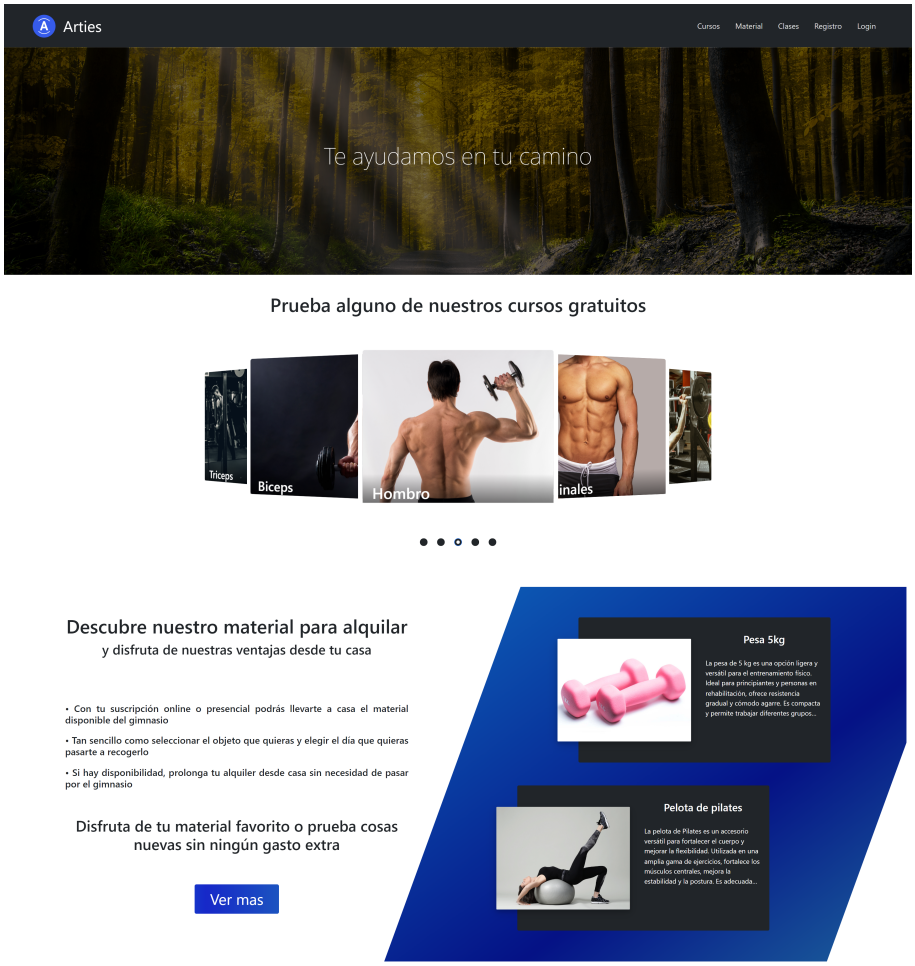


Figura 5.1: Inicio de sesión parte 1

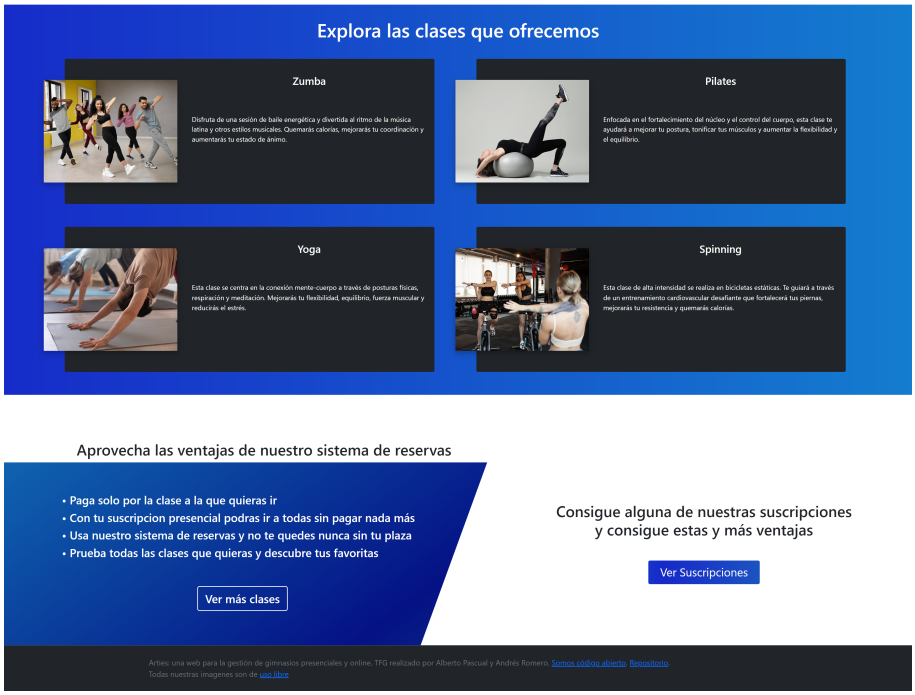


Figura 5.2: Inicio de sesión parte 2

5.2. Inicio de sesión y registro

Arties es una aplicación que se gestiona en función del rol del usuario, que puede ser un administrador, un empleado, un usuario cliente o un usuario que no se ha identificado y que por tanto tendrá muy pocas funcionalidades accesibles en la web. Para saber qué tipo de usuario está usando la web se usa el inicio de sesión (figura 5.3, cuya página se encuentra en la esquina superior derecha, siguiendo el principio de consistencia externa.

Por otro lado está el registro, cuya página se encuentra al lado del botón de inicio de sesión, también en la esquina superior derecha. Aquí el usuario debe rellenar obligatoriamente toda la información pedida, como se ve en la figura 5.4: nombre de usuario, *email*, nombre, apellido, teléfono, dirección y por último la contraseña, la cual tendrá dos campos que deberán coincidir, para asegurar que se haya introducido correctamente y no haya problemas en futuras ocasiones.

Cabe destacar que el nombre de usuario deberá ser único (se avisa al usuario si ya existe), ya que el inicio de sesión se hace a través del nombre de usuario y la contraseña. Se ha elegido esta opción ya que los nombres de usuario suelen ser más cortos, ahorrando tiempo al usuario. El correo se usa para situaciones en las que el gimnasio quiera notificar a sus usuarios mensajes que se quieran hacer ajenos al chat de la aplicación. Esto ocurre por ejemplo con notificaciones por problemas de pago, avisar de que la web estará en mantenimiento, que el gimnasio no será accesible en una fecha determinada...

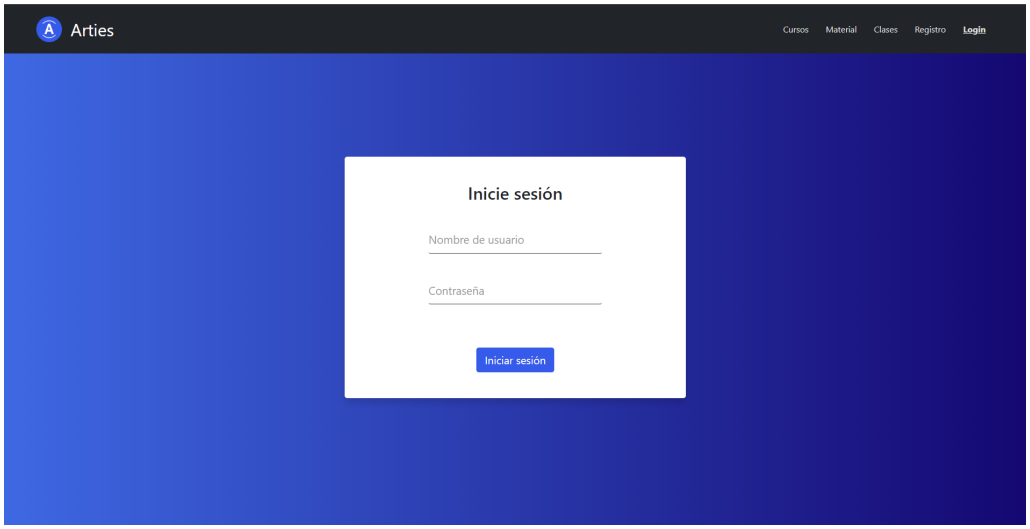


Figura 5.3: Inicio de sesión

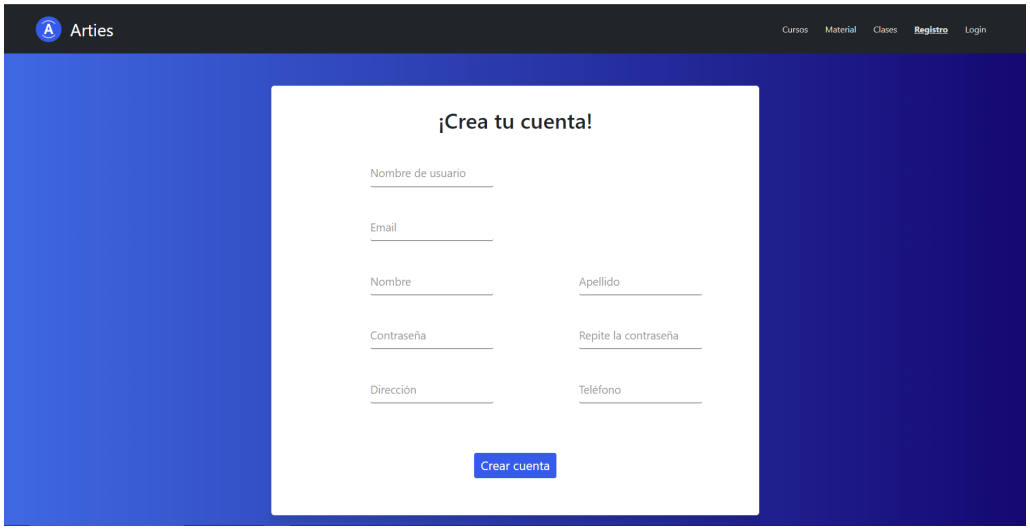


Figura 5.4: Registro del usuario

Desde el punto de vista de la programación, este sistema utiliza parte de las utilidades de Spring Security. De esta forma, Spring Security intercepta las peticiones de Login y las procesa mediante su propio sistema de usuarios (comprobando usuarios y asignado los roles correspondientes). Tras ello la aplicación traduce del usuario de Spring Security a un usuario propio de la aplicación, asignando sus roles y parámetros propios de la aplicación.

Tras la realización del Login, Spring Security genera un X-CSRF-TOKEN, el cual se incluirá a partir de entonces en todas las peticiones que haga dicho usuario, garantizando que las peticiones son realizadas por él y no por otro usuario o página web maliciosa. Así se consigue que el sistema de *Login* pase un filtro de seguridad y que identifique inequívocamente a los usuarios logueados con sus respectivos roles, haciendo así el sistema seguro frente a ataques.

5.3. Gestión de empleados

La gestión de empleados es una sección a la que solo puede acceder el administrador desde el apartado de "Configuración", el cual se accede desde la *navbar* y su contenido se muestra en la figura 5.5. Desde aquí puede ver qué empleados tiene actualmente (nombre y apellido), modificar cualquiera de sus datos: nombre de usuario, nombre, apellido, contraseña, dirección, correo electrónico y móvil. También puede eliminar un empleado o añadir uno nuevo.

Tanto añadir empleado como modificar empleado se realizan desde modales. Esto permite un mejor aprovechamiento del espacio, ya que los respectivos formularios están escondidos hasta que se pulse en el botón de .Añadir." "Modificar". Cabe destacar que se reutiliza el modal de modificar, de forma que solo hay uno para todos los empleados. En su formulario se precargan los datos del empleado seleccionado permitiendo así más rapidez y comodidad a la hora de cambiar los datos. Esto es útil ya que todos los campos son obligatorios y sería tedioso tener que rellenarlos todos los *inputs* cada vez que se quiera modificar solo uno de los datos.

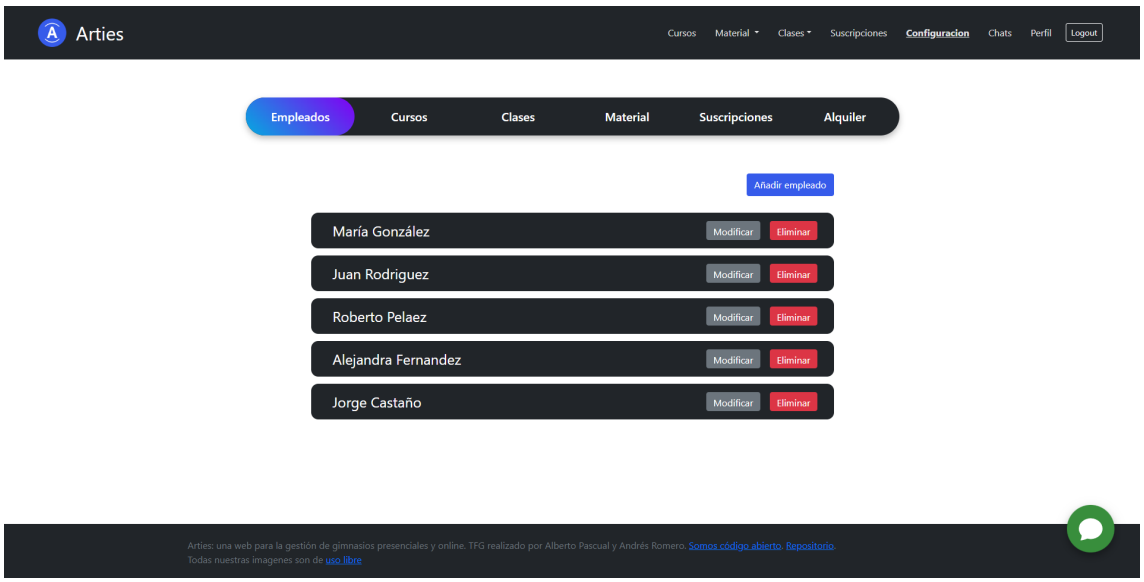


Figura 5.5: Gestión de empleados

5.4. Material

El sistema de gestión de material es la base que permite implementar la funcionalidad de alquiler de material. Gracias a la combinación de estos dos sistemas se cubre el requisito de permitir a los usuarios alquilar material del gimnasio para poder utilizarlo en su casa. Por tanto, el sistema de gestión del material será el que utilice el administrador para personalizar que objetos se pueden alquilar.

Este sistema presenta dos partes diferenciadas. Por un lado, la parte visible a los clientes, en la cual se muestra que material tiene disponible el gimnasio para alquilar como se puede ver en la figura 5.6. Dicha vista es accesible también para usuarios no registrados, de forma que estos puedan conocer de antemano que material ofrece el gimnasio y así poder valorar mejor si les interesa comprar una suscripción o no.

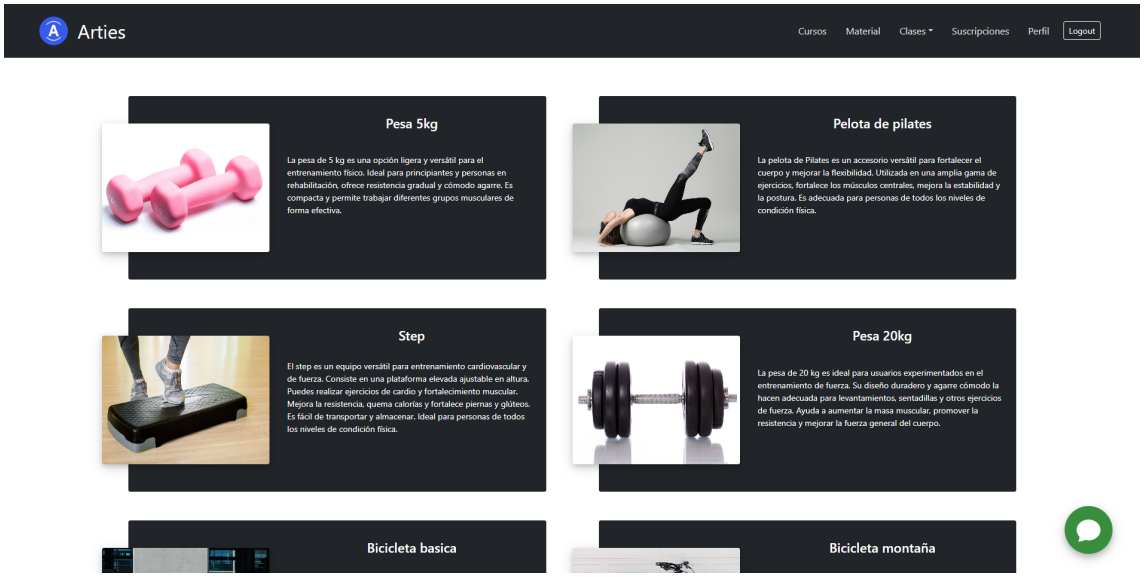


Figura 5.6: Lista de material en la vista de usuario

Por otro lado, la parte más importante de este sistema está dentro del apartado de configuración del gimnasio (figura 5.7). Aquí el propietario podrá administrar el material ofrecido por el gimnasio. Se le presenta una opción que le permite crear un nuevo material (figura 5.8). El propietario debe definir dicho objeto proporcionando sus datos, siendo estos una imagen, un nombre, una descripción (la cual puede incluir desde algún detalle hasta recomendaciones de ejercicios), la cantidad de material que podrá ser alquilada y la cantidad máxima que un cliente puede alquilar en una misma vez.

Este último parámetro resulta ser importante, ya que le permite una mayor personalización y adaptación a diferentes tipos de gimnasio. Un uso básico de este parámetro sería por ejemplo al definir un objeto “Pesas”, pues puede darle al usuario la opción de alquilar como máximo dos pesas a la vez, ya que el uso habitual de muchas pesas suele ser por pares. Sin embargo, si el usuario solo necesita una, también puede alquilar solo una. Por otro lado, si el gimnasio dispone de mucho material, y como aspecto diferenciador quiere permitir alquilar más unidades para que el usuario pueda compartir con su familia o amigos, el propietario puede modificar dicho valor para permitir que se puedan alquilar más unidades a la vez.

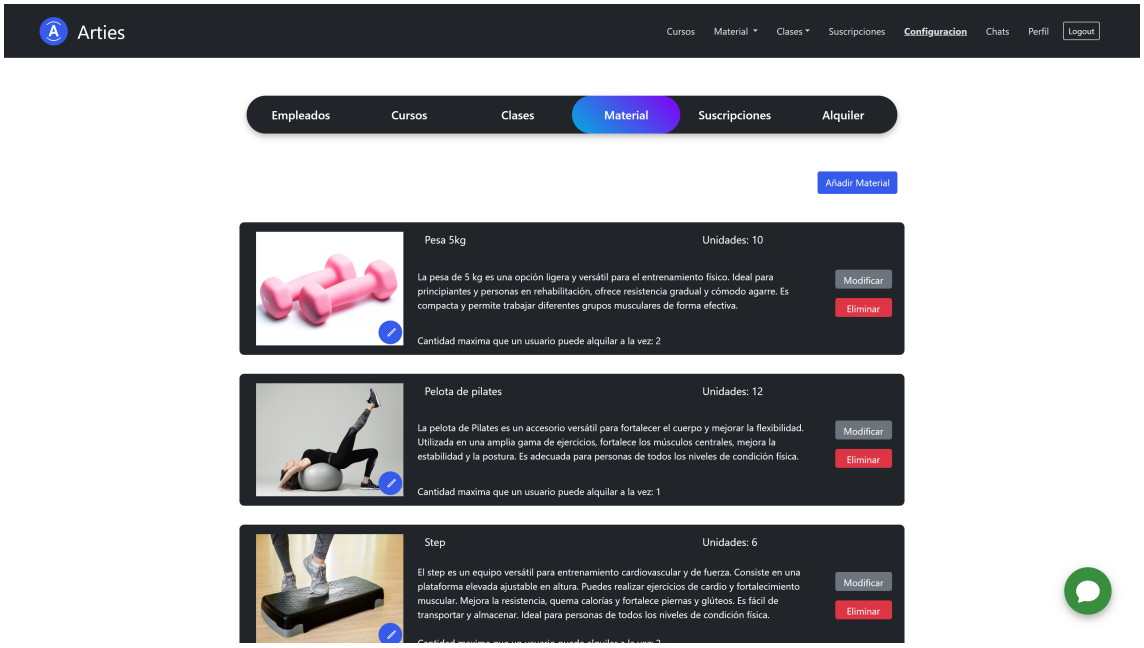


Figura 5.7: Lista de material en la vista del propietario

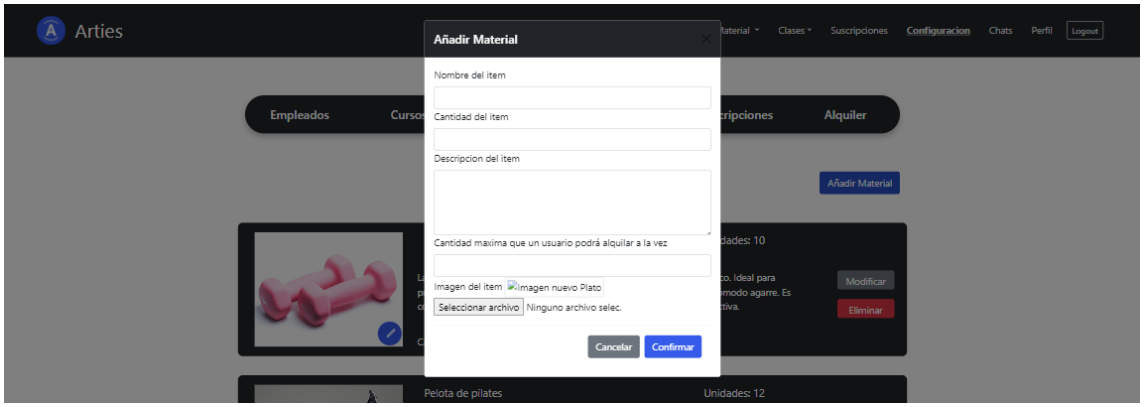


Figura 5.8: Modal para añadir un nuevo material

Desde esta configuración de material el propietario también puede modificar los parámetros de los objetos que ya haya definido, con la particularidad de que si decide reducir la cantidad que tiene de cierto objeto, está siempre deberá ser mayor o igual a la cantidad alquilada actualmente. Esto evita que se produzca la situación errónea en la que tiene más unidades alquiladas de las que realmente tiene disponibles.

Por último, también dispone de la opción de eliminar un material siempre y cuando este no esté alquilado actualmente, evitando así también que se produzcan situaciones erróneas.

5.5. Alquiler

El alquiler de material es una de las partes principales de la aplicación, pues es uno de los pilares que permite a los usuarios poder disfrutar de una mejor forma del deporte en su casa. Este sistema es una de las innovaciones que aportaría la *app* que no muchos gimnasios de hoy en día ofrecen.

Para su implementación se ha partido del concepto de sistemas de alquiler o reservas similares, como podría ser un sistema de préstamos de libros, en los cuales el usuario puede estar en posesión del objeto durante un tiempo predefinido, teniendo una fecha límite para devolverlo y pudiendo ampliar dicha fecha si hay disponibilidad. Además se permite a los usuarios elegir el día que desea que comience su alquiler para que pueda planificarlos.

De forma simplificada, el sistema de alquiler permite al usuario seleccionar un día en el que quiere alquilar un material, permitiéndole recogerlo a partir de ese día y tener dicho objeto durante un periodo de tiempo para utilizarlo en su casa. Para un mejor entendimiento de este sistema se ha diseñado el diagrama de la figura 5.9 que refleja de forma genérica el flujo de acciones que conlleva este sistema.

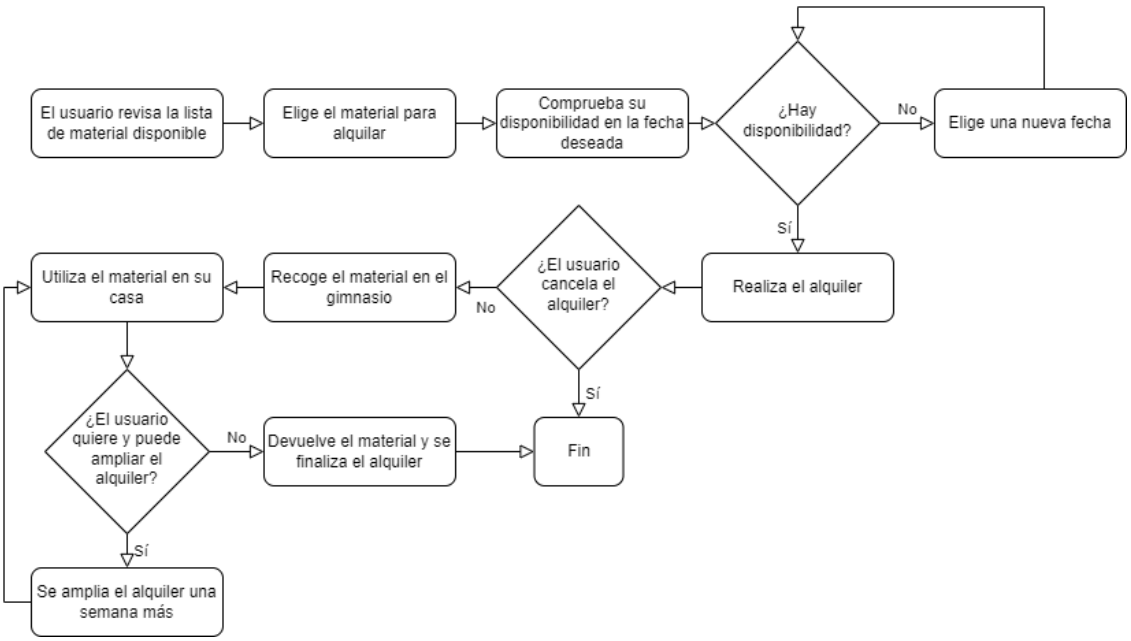


Figura 5.9: Diagrama de flujo sobre el uso genérico del sistema de alquileres

Para entrar más en detalle en cómo funciona, se deben definir cuales son las diferentes restricciones que incluye la aplicación sobre este sistema de alquiler.

En el caso del tiempo de alquiler se ha predefinido a una semana desde el inicio del alquiler. De esta forma se da un tiempo suficiente al usuario para que pueda hacer uso del material, pero al no ser un periodo muy largo, permite una rápida fluctuación del material entre los diferentes clientes.

El sistema de ampliación de alquiler dispone de diferentes restricciones y comprobaciones con el fin de asegurar que un usuario no se queda indefinidamente con un material, y que este pueda ser utilizado de forma equitativa entre los diferentes usuarios que quieran utilizarlo.

La primera restricción establece que un usuario solo podrá ampliar el alquiler de un material si se encuentra en las 48 horas previas a la finalización de su alquiler. En caso de que se encuentre en ese intervalo pueden darse dos situaciones. La primera, en la que en la semana siguiente al fin de su alquiler hay unidades suficientes para ampliar su alquiler (esto teniendo en cuenta que el usuario podía tener más de una unidad alquilada si el material lo permitía). En este caso podrá ampliar el alquiler una semana más desde la fecha de fin del actual. La segunda situación es que no haya unidades suficientes en la semana siguiente al fin de su alquiler (debido a que haya reservas para alquiler de otros usuarios). En ese caso no se permitirá ampliar la fecha y tendrá que esperar a devolver el objeto para poder volver a alquilarlo de nuevo.

Por último, si un usuario no devolvió un objeto a tiempo, no se le permitirá ampliar su alquiler ni hacer nuevos alquileres sobre él hasta que lo devuelva.

Todas estas condiciones se pueden ver de forma visual en el diagrama de flujo de la figura 5.10.

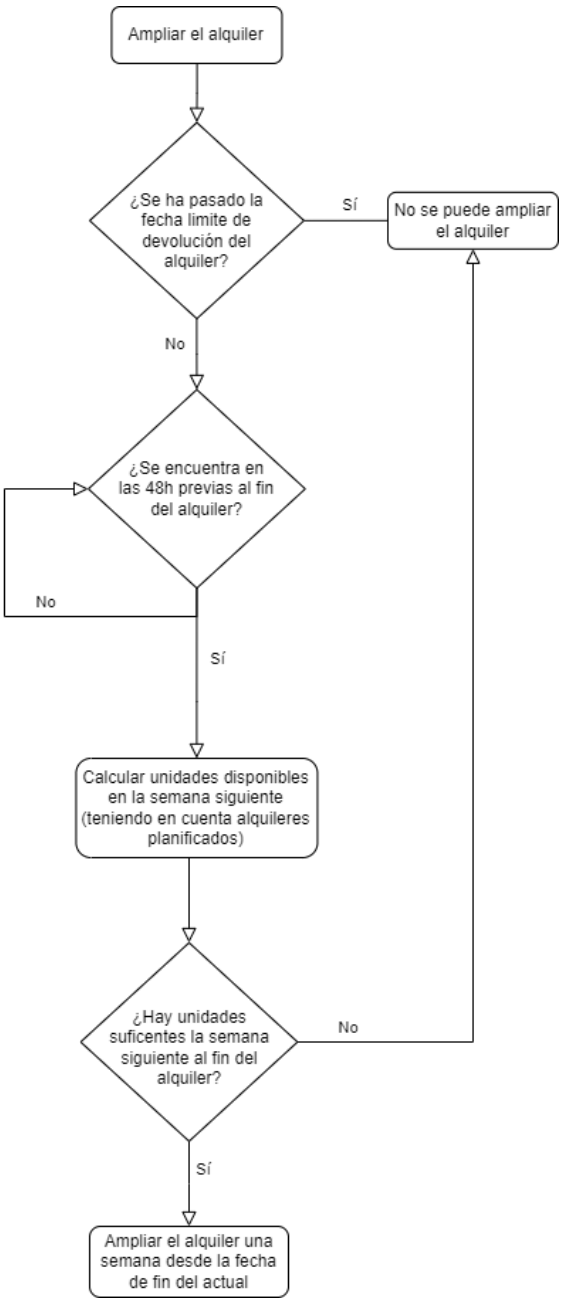


Figura 5.10: Diagrama de flujo sobre ampliar un alquiler

Desde el punto de vista visual se han desarrollado varias pantallas que permiten realizar los alquileres y administrarlos.

El usuario utiliza dos pantallas diferentes. Una de ellas es la pantalla para hacer la reserva de alquiler de un material (figura 5.11). En esta se le muestra la información del objeto a alquilar, un calendario para que seleccione el día a realizar el alquiler, información sobre cuantas unidades hay disponibles el día seleccionado, un campo donde puede indicar cuantas unidades quiere alquilar el día seleccionado y el botón de confirmar el alquiler. La información de cuantas unidades hay disponibles el día seleccionado se carga dinámicamente mediante el uso de AJAX.

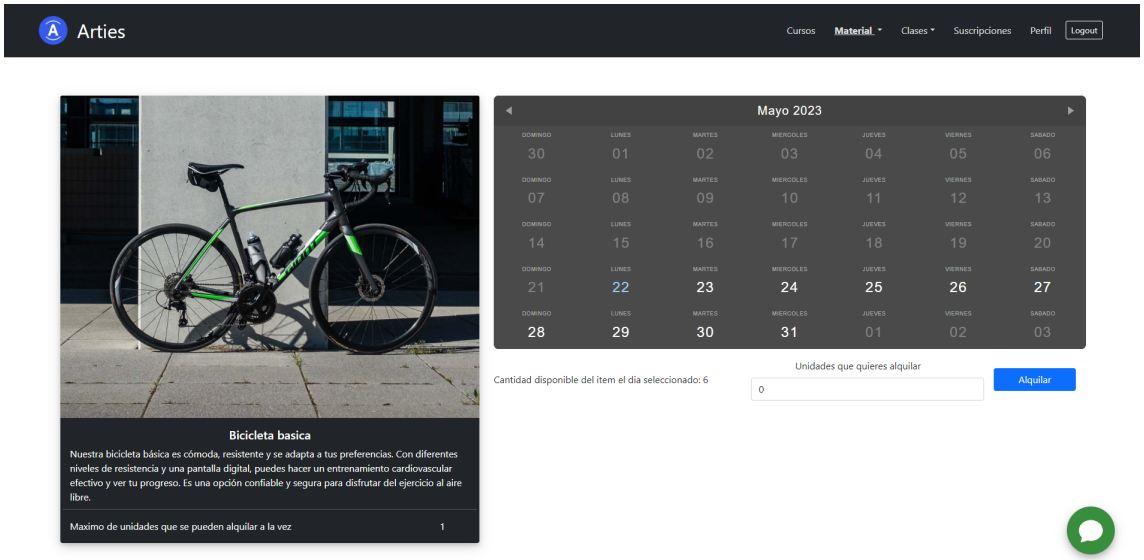


Figura 5.11: Vista para hacer el alquiler de un material

Sobre esta pantalla hay que destacar que su contenido cambia en función de si el usuario tenia ya alquilado ese objeto, si ya lo tenia y puede ampliar su alquiler o si la fecha de devolución se ha pasado, como se puede ver en la figura 5.12.

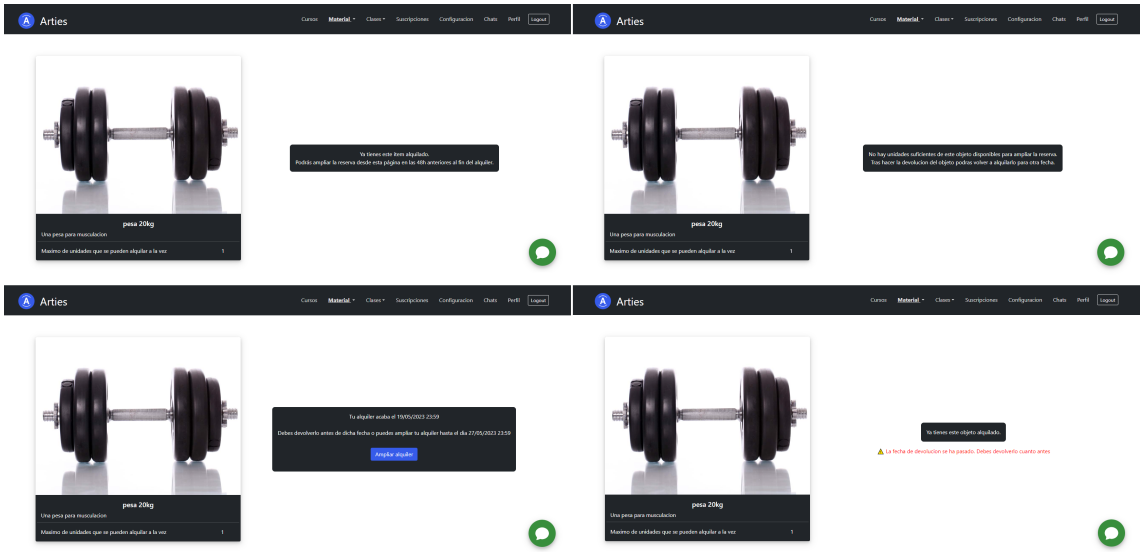


Figura 5.12: Variaciones de la vista de alquilar material

La otra pantalla que utiliza el usuario sobre el sistema de alquiler es aquella en la que puede ver sus alquileres actuales (figura 5.13). En ella puede ver que objetos tiene en alquiler, además de información útil como las fechas del alquiler o información sobre cuando podrá ampliar el alquiler. Además, cada objeto incluye un botón que se activará cuando el usuario pueda ampliar el alquiler de dicho material. Destacar, que cuando la fecha de devolución se ha pasado se muestra un mensaje avisando al usuario.

En esta pantalla también se permite al usuario cancelar un alquiler en caso de que se haya confundido o haya cambiado de opinión. Sin embargo, existe una restricción, y es que la cancelación solo se puede realizar si se hace en una fecha anterior al comienzo del alquiler. Esto se ha definido así para evitar que los usuarios puedan cancelar sus propios alquileres poseyendo ya el objeto y así quedarse indefinidamente con él. De esta forma, el botón para cancelar el alquiler de un material solo esta disponible hasta la fecha de comienzo del alquiler, siendo sustituido tras esa fecha por el de ampliar alquiler (el cual estará desactivado hasta 48 horas antes del final del alquiler como se ha explicado anteriormente).

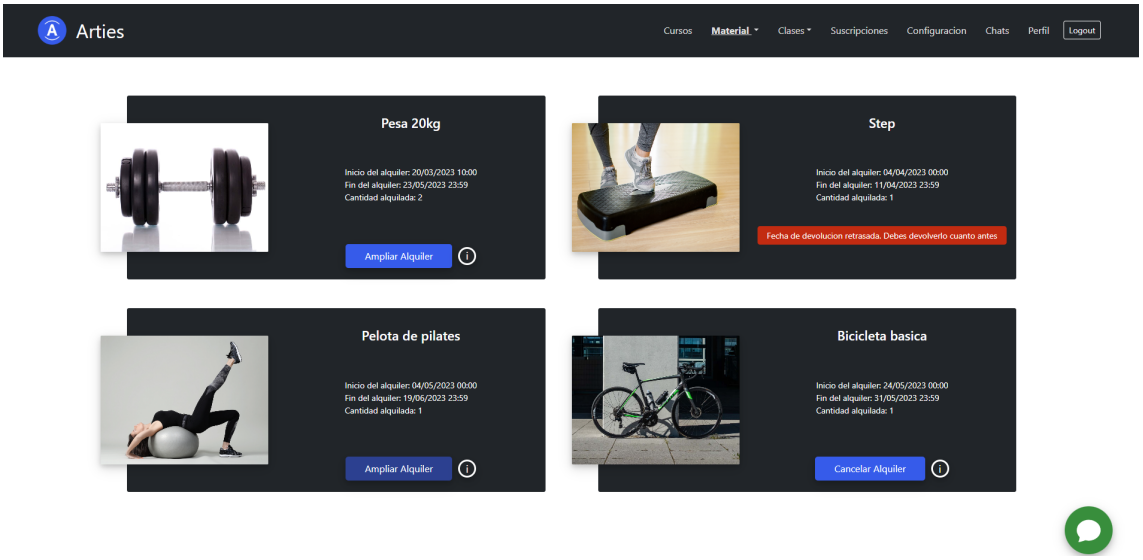


Figura 5.13: Vista de los materiales alquilados por el usuario

De cara al propietario, este dispone de un apartado de Alquiler dentro de configuración donde puede ver todos los alquileres organizados por material (figura 5.14). En ellos se le muestra tanto información del propio alquiler (como fechas o cantidades alquiladas), como información del usuario (nombre y datos de contacto). Además, se incluye un botón para confirmar la devolución de un objeto por parte de un usuario. También destacar, que para aquellos alquileres que se hayan pasado la fecha de devolución, el campo de fecha del alquiler saldrá marcado en rojo para avisar al propietario.

Si se pulsa en finalizar alquiler se muestra un mensaje mediante un *Toast* confirmando el fin del alquiler, y con un botón que permite deshacer dicha acción (figura 5.15). Si se pulsa, se restablece el alquiler finalizado y se cambia el botón por un texto informativo que indica que se ha deshecho la ultima acción (figura 5.16).

En este apartado se encuentra también un botón que permite ver al administrador el historial de todos los alquileres que se han realizado en el gimnasio.

Los empleados también tienen acceso a la funcionalidad de gestión de alquileres, sin embargo la realizan a través de una página propia con formato similar a la de configuración del administrador y accesible directamente desde la *navbar*.

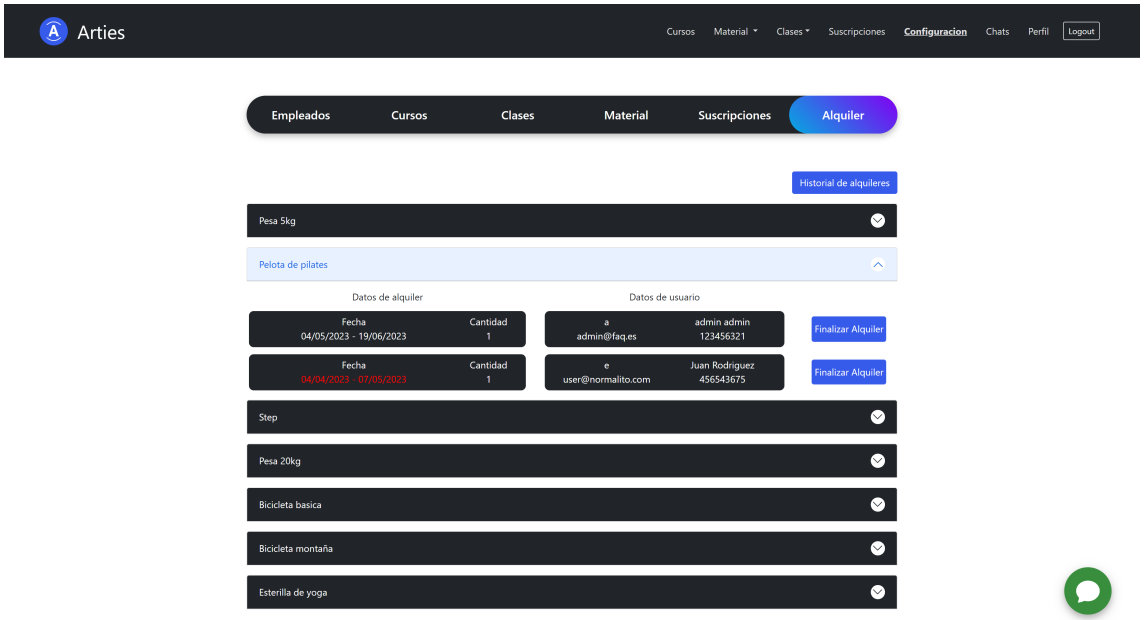


Figura 5.14: Vista de gestión de alquileres

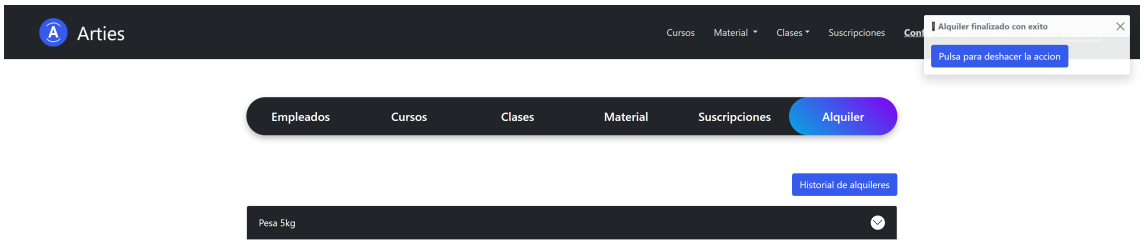


Figura 5.15: Mensaje informativo de finalización de alquiler

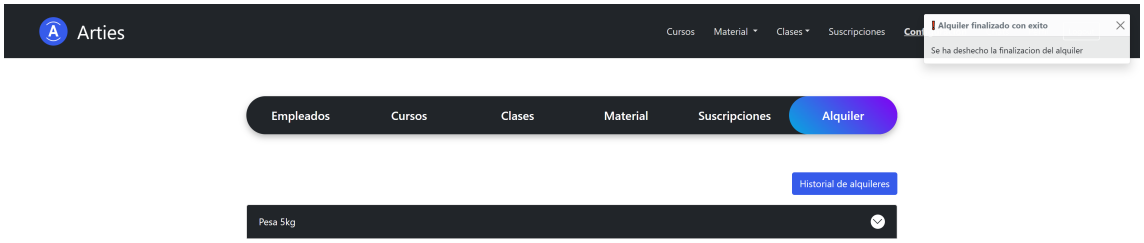


Figura 5.16: Mensaje informativo de deshacer finalizar un alquiler

5.6. Chat

El chat es un apartado muy importante para la aplicación, ya que el proyecto trata de simplificar la vida del usuario, haciendo su experiencia más dinámica y flexible. Además, al estar la opción del gimnasio *online*, es de vital importancia que los usuarios puedan preguntar desde casa todas sus dudas a los profesionales, de la manera más rápida y cómoda posible.

El chat funciona de manera ligeramente diferente dependiendo del tipo de usuario. Los clientes pueden acceder al chat a través del icono flotante que aparece en la esquina inferior derecha, en cualquiera de las pantallas. Al clicarlo se abre una ventana flotante de tamaño reducido que da al usuario tres opciones, tal y como se ve en la figura 5.17 y que se analiza posteriormente.

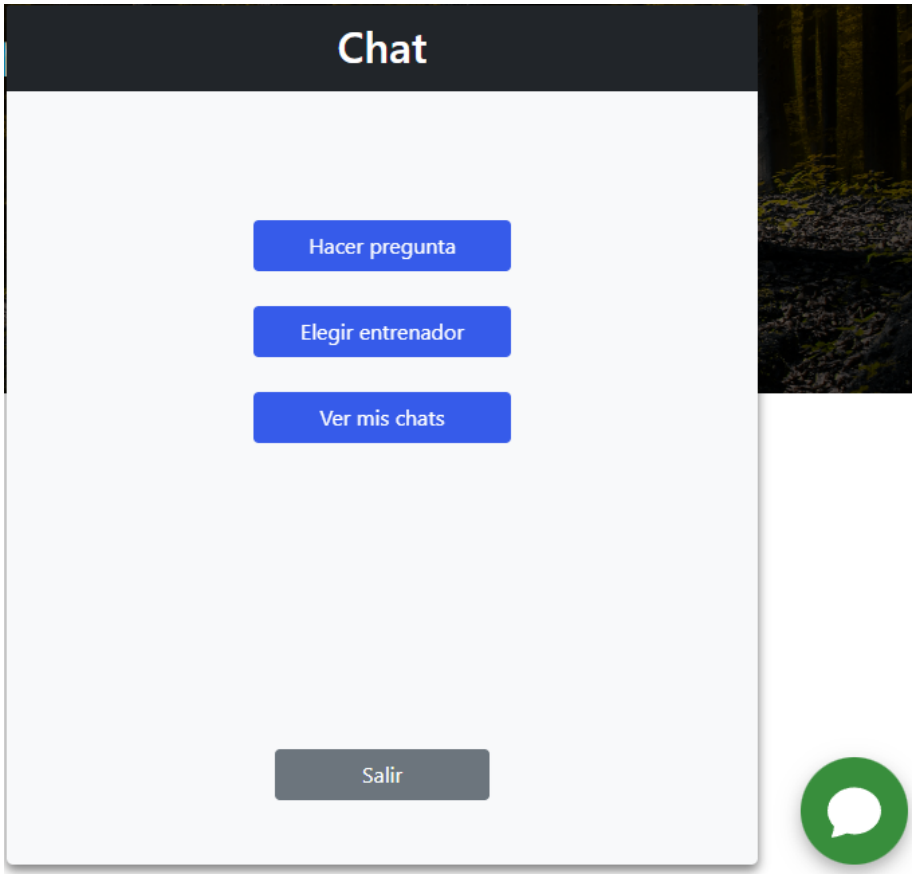


Figura 5.17: Vista inicial del chat del cliente

En primer lugar, se puede hacer una pregunta general. Esto quiere decir que el usuario especifica un asunto para la pregunta y posteriormente desarrolla el problema. Esta pregunta no tendrá un destinatario en concreto, sino que llegará a una página que comparten todos los empleados del lugar, permitiendo que responda el instructor disponible en el momento, o un instructor especializado en el tema de la pregunta. Esta es una buena vía para cuando el cliente quiere obtener la respuesta lo más rápido posible, o no sabe a qué empleados tiene que dirigir la pregunta.

En segundo lugar, el cliente puede ver una lista de entrenadores y elegir con quién quiere iniciar o continuar una conversación. De esta forma, por ejemplo, si sabe que un entrenador está especializado en nutrición, podrá contactar con él directamente. Por otro lado, si ha hablado con un instructor que no estaba especializado en el tema de su pregunta, pero le ha recomendado con quién hablar, el usuario podrá abrir un chat directamente con la persona recomendada.

En último lugar, el cliente puede acceder a sus chats. Desde aquí puede revisar conversaciones que haya tenido en el pasado, además de continuarlas y hacer más preguntas. Este chat tiene consistencia externa, ya que tiene una lista de chats y en función del chat seleccionado aparece una conversación u otra, tal y como se muestra en la figura 5.18 siendo esto el formato habitual de las aplicaciones de chat.

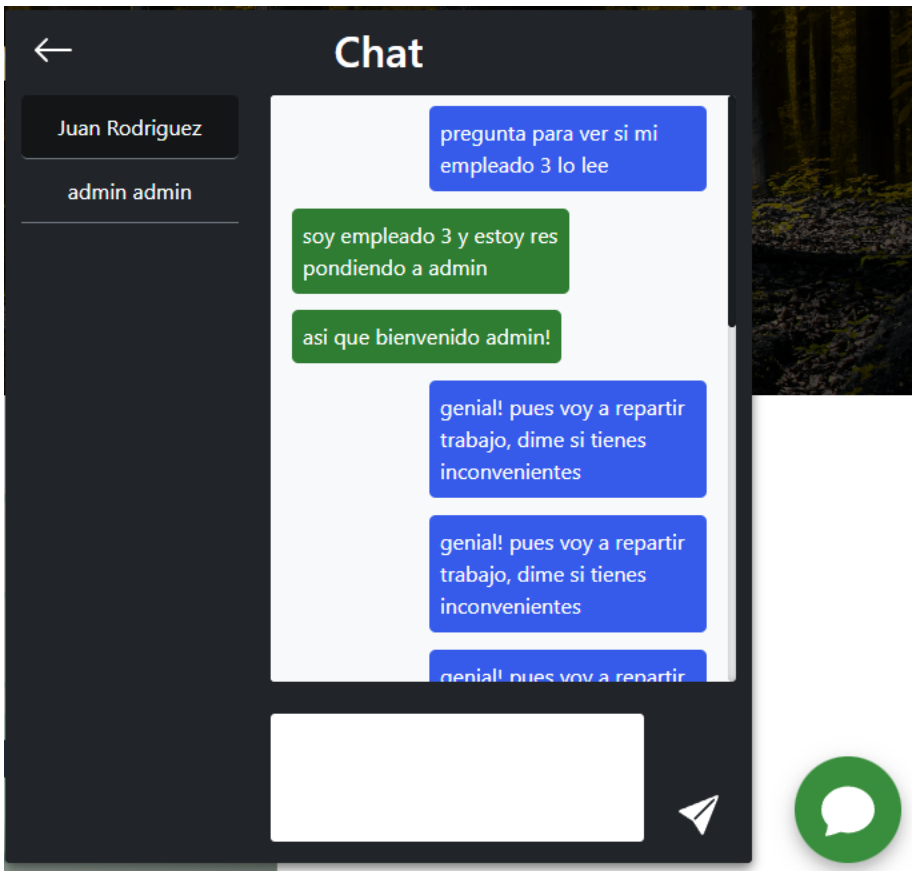


Figura 5.18: Vista de los chats del cliente

Para los empleados el chat funciona de manera algo diferente. Estos tendrán una página dedicada al chat, a la cual se accede desde la *navbar*. Esta página tendrá dos vistas diferentes, a las cuales se accede a través de una *subnavbar*, como se ve en la figura 5.19.

En la primera ventana aparece una lista en forma de acordeón con todas las preguntas generales que han hecho los usuarios. En cada entrada se indica el nombre y apellido del usuario, el asunto de la pregunta y tres botones: ver pregunta, con el cual se despliega la entrada del acordeón mostrando el mensaje; aceptar el chat, con

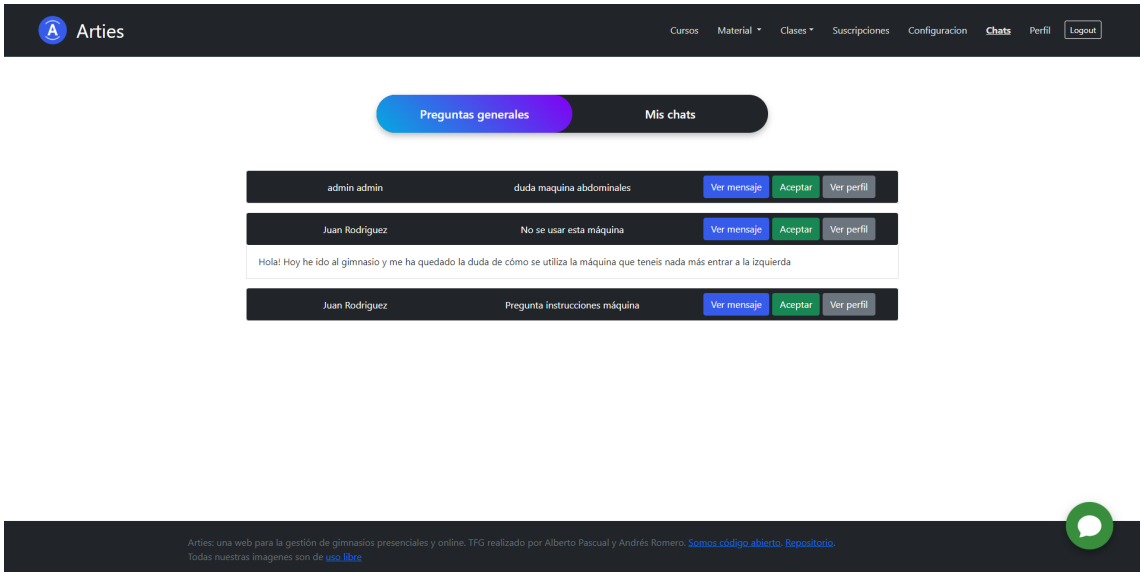


Figura 5.19: Vista de preguntas generales para empleados

el que se vincula la conversación entre el cliente y el entrenador que haya aceptado la pregunta, desapareciendo así la pregunta general para el resto de entrenadores; y ver el perfil del usuario, para que el entrenador pueda tener un contexto del cliente y poder responderle de una manera más personalizada. Si el empleado quiere responder la pregunta, deberá aceptar el chat, y entonces la pregunta general pasará a tener destinatario, que será el empleado, y pasará a ser una conversación normal.

En la segunda ventana aparecen los chats específicos del empleado. Desde aquí el personal puede ver y responder los mensajes privados que le llegan, además de también poder ver el perfil de los usuarios, para poder responder las preguntas de una manera más personalizada.

Cabe destacar que para la implementación del chat se ha hecho uso de *Websockets*, los cuales permiten conversaciones y actualizaciones de partes de la página en tiempo real, sin tener que recargar la web, dando así una experiencia mucho más fluida, cómoda y realista.

5.7. Suscripciones

Como se menciona en otras partes del documento, Arties es una aplicación web para la gestión de gimnasios tanto *online* como presencial. Por esto, hay que saber qué servicio ha contratado el usuario, ya que la opción de gimnasio *online* no otorga todas las ventajas que tiene el servicio presencial. Esto se hace mediante las suscripciones.

Hay dos suscripciones: *online* y presencial, de tal forma que el usuario solo puede estar suscrito a una de las dos. Las ventajas que incluye cada opción corresponde

al administrador definirlas.Por ejemplo, como se ve en la figura 5.20, la suscripción *online* incluye acceso a todos los cursos, alquiler de material y ayuda a través del chat, mientras que la presencial, además de lo que incluye la *online*, también otorga acceso al gimnasio y acceso gratis a las clases impartidas en el local. También se puede cancelar la suscripción de cara al mes siguiente haciendo clic en el botón rojo de "Cancelar suscripción" que aparece debajo de las tarjetas cuando el usuario está actualmente suscrito a una.

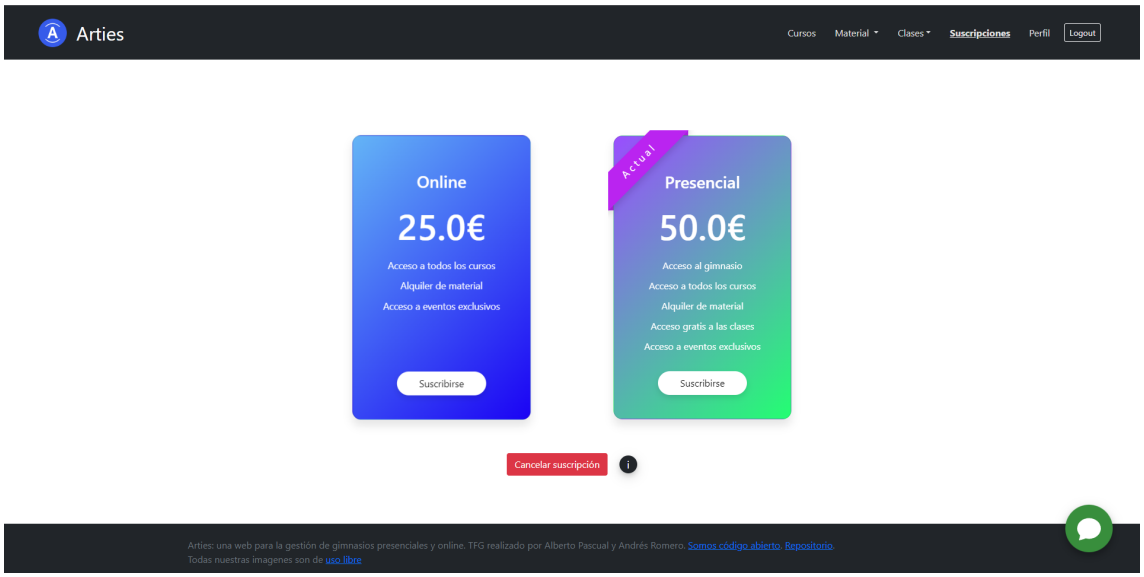


Figura 5.20: Vista de las suscripciones al gimnasio

El administrador puede controlar las características de las suscripciones a través de una interfaz muy cómoda y rápida (figura 5.21) situada en la página de "Configuración", en el apartado de "Suscripciones". Aquí se muestran dos columnas, una con la información de la suscripción *online* y la otra con la de la presencial . En cada columna se muestra el precio actual, para que el administrador sea consciente en todo momento de qué precios hay establecidos, aunque también lo puede mirar en la página de "Suscripciones". Sin embargo desde ahí no se puede editar el precio así que el proceso conllevaría más clics. También se presentan las frases de la opción correspondiente, con un botón de eliminar al lado de cada una, además de la posibilidad de añadir nuevas.

Para gestionar sus suscripciones, el cliente accede desde la *navbar*, al apartado "Suscripciones", y selecciona una de las dos (figura 5.20). Tras esto se le redirige a una pantalla de pago (figura 5.22), la cual simula una pasarela de pago que no es funcional. A continuación el usuario debe rellenar los datos que son pedidos: titular de la tarjeta, número de tarjeta, fecha de caducidad, código de seguridad y dirección de la vivienda. Por último, tras çonfirmar el pago", se redirige al usuario a la pantalla de suscripciones, donde puede ver claramente cuál es la suya actual.

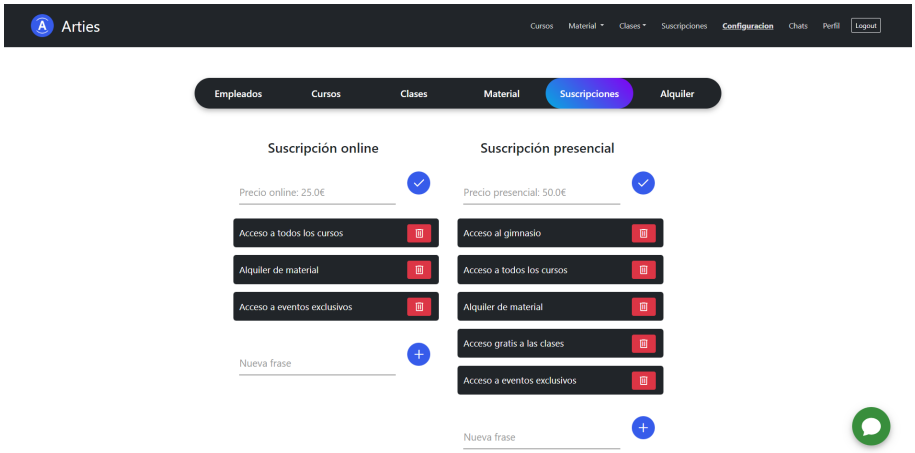


Figura 5.21: Vista de gestión de suscripciones

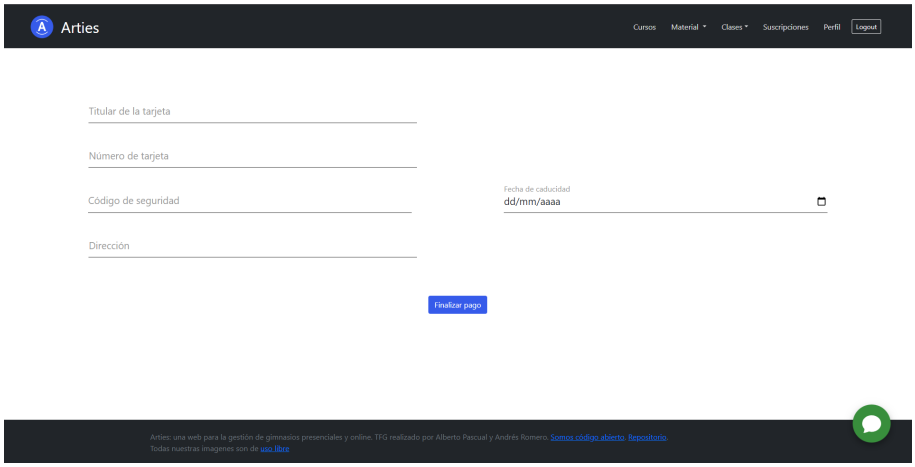


Figura 5.22: Vista de la pantalla de pago

5.8. Perfil

La aplicación tendrá perfiles de usuario. En el perfil se distinguen tres bloques. Por un lado, los datos personales y de inicio de sesión del usuario, por otro lado una descripción personal escrita por el usuario, aunque es opcional, y por último unas estadísticas básicas sobre cuántas veces ha completado cada curso, cuántas veces ha alquilado cada material y cuántas veces ha asistido a cada clase . También hay una foto de perfil, que viene por defecto con un icono de un avatar como se ve en la figura 5.23, sin embargo, el usuario la podrá cambiar por la foto que desee, siempre que no sea excesivamente grande y cumpla con los formatos admitidos. Esta foto no solo permite una mayor personalización y satisfacción por parte del usuario, si no que también permite a entrenadores identificar más fácilmente a los usuarios.

El primer bloque, de datos personales y de inicio de sesión (figura 5.23), consiste en un formulario con los datos precargados, excepto las contraseñas. Gracias a esto se permite al usuario cambiar la información deseada de una manera más rápida y

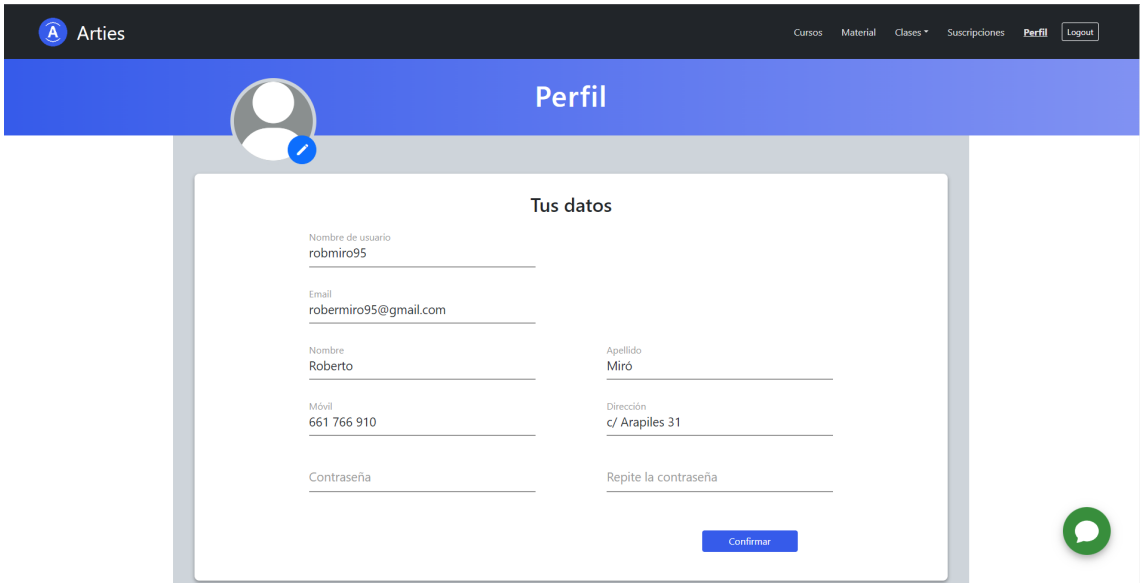


Figura 5.23: Vista inicial del perfil

cómoda, ya que si no estuviesen los datos precargados, tendría que rellenar todos los campos pues son obligatorios, menos las contraseñas que son un caso especial. La contraseña tiene dos *inputs*: "Contraseña" y "Repite la contraseña" permitiendo así cambiarla muy rápidamente. Para aplicar los cambios se deberá clicar en el botón de "Confirmar" el cual comprueba la validez de los datos, y si la contraseña se ha dejado en blanco, se mantendrá la que tuviese el usuario anteriormente.

El segundo bloque contiene un campo para que el usuario escriba una descripción sobre sí mismo, indicando lo que considere importante, que generalmente será qué deportes hace, qué *hobbies* tiene, qué le gustaría hacer, si ha tenido o tiene lesiones, si es intolerante a algo o tiene alergias... Esto permitirá que los entrenadores puedan responder de manera más acertada y personalizada al cliente, en caso de que este haya realizado una pregunta. Para ayudar al usuario a saber qué tiene que escribir en este apartado, el bloque tiene un subtítulo indicando qué se recomienda contar sobre uno mismo, siempre orientado hacia la utilidad de la información respecto al gimnasio o el ámbito deportivo o de bienestar.

Como se ha mencionado anteriormente, el tercer bloque contiene las estadísticas. Tanto el segundo como el tercero se pueden ver en la figura 5.24, aunque en esta no aparecen las últimas estadísticas sobre cuántas veces se ha alquilado cada material.

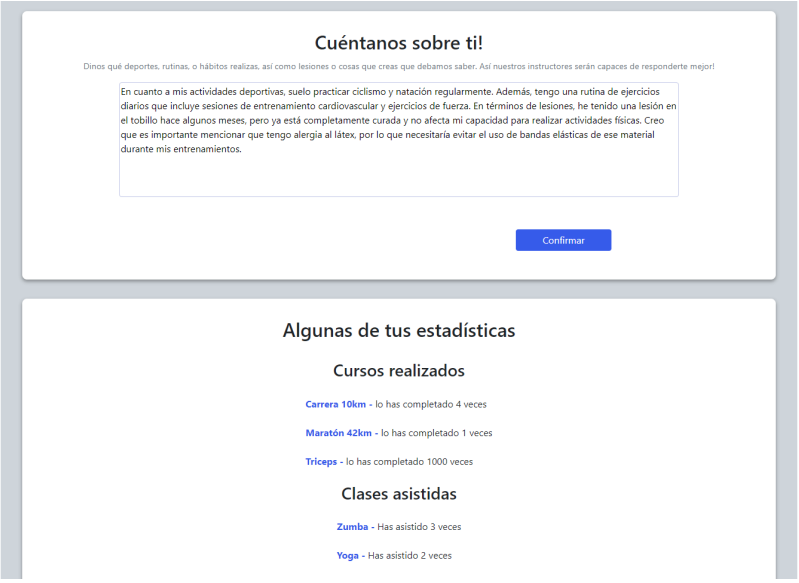


Figura 5.24: Bloques segundo y tercero del perfil

5.9. Clases

Otro de los sistemas principales de la aplicación es la reserva de clases. Este sistema permite reservar plaza para una sesión de una de las clases que ofrece el gimnasio y presenta dos principales beneficios. Por un lado, permite que usuarios con suscripción presencial puedan reservar su plaza para las clases. De esta forma se aseguran de que puedan ir a dicha clase. Por otro lado, permite que usuarios que no tengan suscripción presencial puedan acudir a una clase, pagando únicamente por dicha clase. Esto permite que usuarios que estén pensando en apuntarse al gimnasio puedan probar fácilmente la calidad de sus clases, y permite que usuarios sin tanto tiempo o dinero puedan disfrutar de las clases del gimnasio de vez en cuando.

Antes de explicar en detalle este sistema es necesario aclarar los conceptos de clases y sus sesiones. Una clase es definida por el gimnasio y la imparte un profesor (por ejemplo, clase de yoga). Dicha clase ocurre periódicamente, los mismos días de la semana a la misma hora (por ejemplo, los lunes a las 10:00 y los miércoles a las 11:00 y a las 17:30). La fecha y hora concreta a la que ocurre una clase se trata de una sesión (por ejemplo, la clase de Yoga del día 17/04/2023 a las 10:00).

Este sistema al ser uno de los principales, se divide en funcionalidad de cara a los clientes y funcionalidad de cara al administrador para su gestión. A continuación, se explicará qué ofrece para cada uno de estos usuarios.

Desde el punto de vista de los usuarios, este sistema les permite ver todas las clases que ofrece el gimnasio (figura 5.25) y reservar plazas para dichas clases.

Una vez seleccionada una clase, aparecerá en la mitad izquierda toda su información, siendo esta su nombre, descripción completa, aforo de la clase, y si se trata

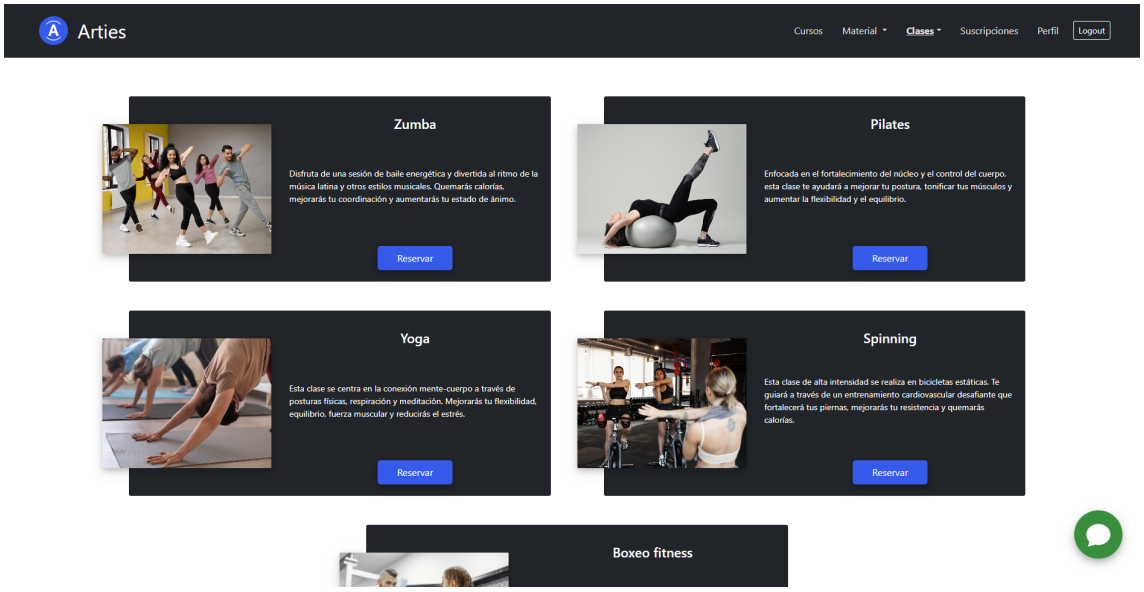


Figura 5.25: Vista de lista de clases

de un usuario sin suscripción presencial, el precio de la clase. En la otra mitad se presenta un calendario en el cual solo están activados los días en los que hay alguna sesión de la clase seleccionada.

Una vez elija una de las fechas seleccionables, el usuario podrá elegir una hora de las disponibles ese día. Tras seleccionar una hora, se cargará la información de cuantas plazas quedan disponibles para dicha sesión, informando al usuario, y este podrá reservar una plaza si hay al menos una disponible. Todo esto se puede ver en la figura 5.26.

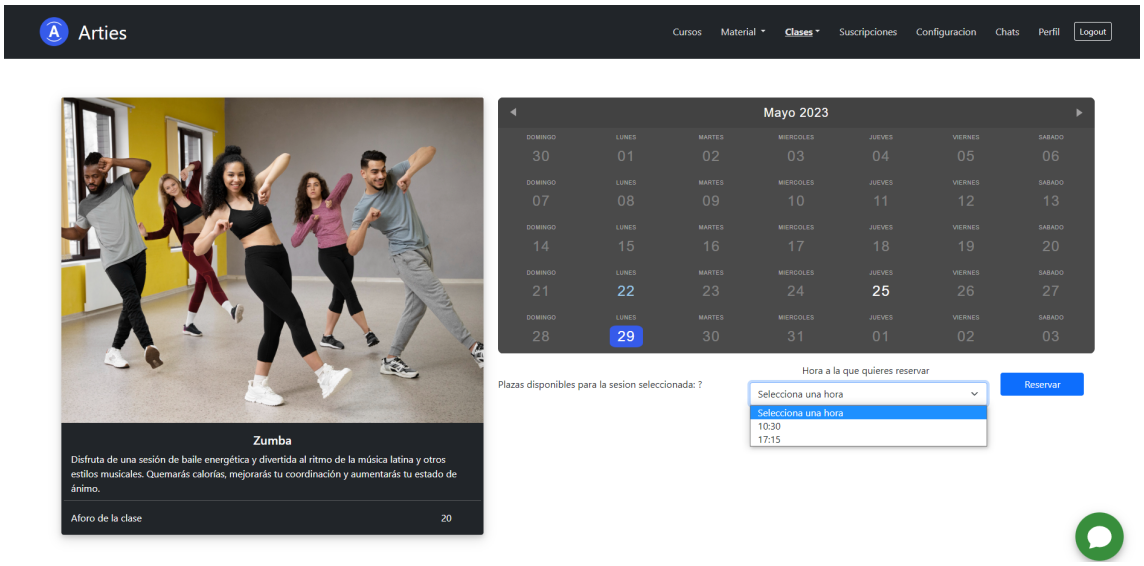


Figura 5.26: Reserva de una clase en un día y hora concretos

Si el usuario no tiene suscripción presencial se le redirigirá a la página de pago y tras realizarlo se confirmará la reserva. Si el usuario tiene suscripción presencial

se confirmará la reserva directamente. En ambos casos, finalmente, se redirigirá al usuario a la página de sus reservas (figura 5.27), donde podrá ver qué reservas para próximas clases tiene. Además, cada reserva incluye un botón que permite cancelarla en caso de que el usuario quiera, dejando así su plaza libre.

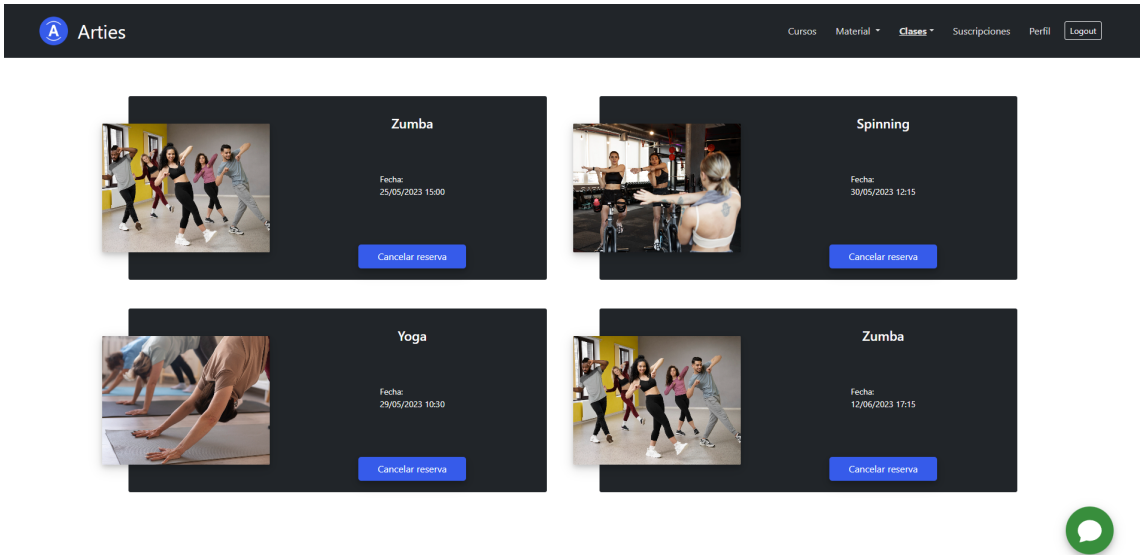


Figura 5.27: Vista de la lista de clases reservadas

Desde el punto de vista del administrador este sistema le permite gestionar toda la información relativa a las clases. Puede acceder a este sistema desde la página de "Configuración", dentro del subapartado "Clases". Aquí se presenta un botón para añadir una nueva clase y un listado de todas las clases disponibles con sus datos y botones para modificarlas y eliminarlas (figura 5.28).

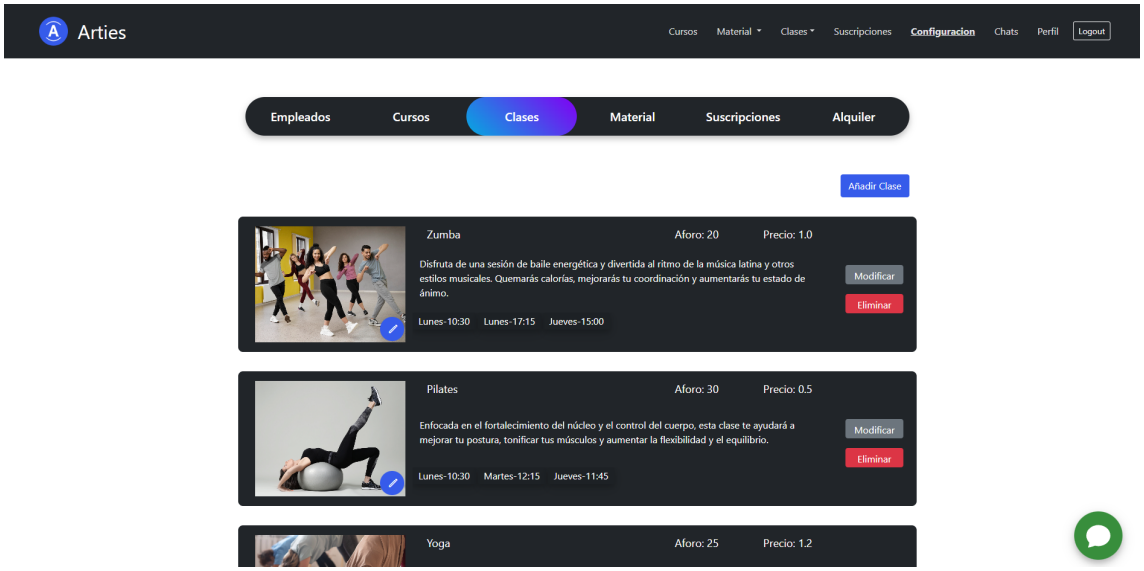


Figura 5.28: Vista de clases en configuración

Al definir una clase (ya sea creándola o modificándola) se abre un modal donde puede introducir sus datos (figura 5.29). Destacar que desde aquí puede definir la

lista de días y horas de la semana a las que ocurre una clase, pudiendo añadir nuevas fechas o eliminar las ya existentes. Al pulsar añadir sesión se activa una pequeña sección donde puede indicar el día de la semana y la hora de la sesión (figura 5.30). También se desactivan los botones de añadir sesión y confirmar para evitar que el usuario cometa errores. En el caso de que se borre alguna sesión se avisa al administrador que todas las reservas asociadas a dicho horario serán canceladas automáticamente.

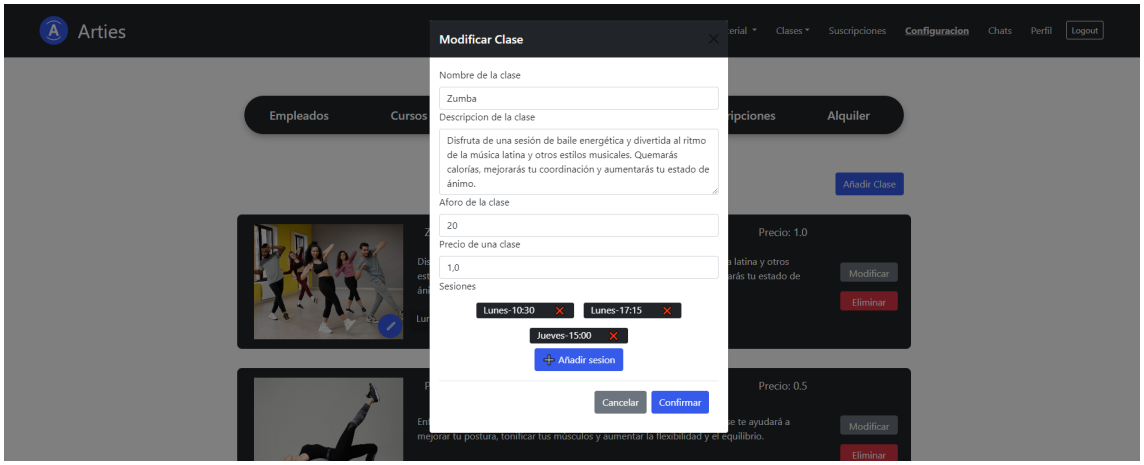


Figura 5.29: Modal para añadir y modificar clases

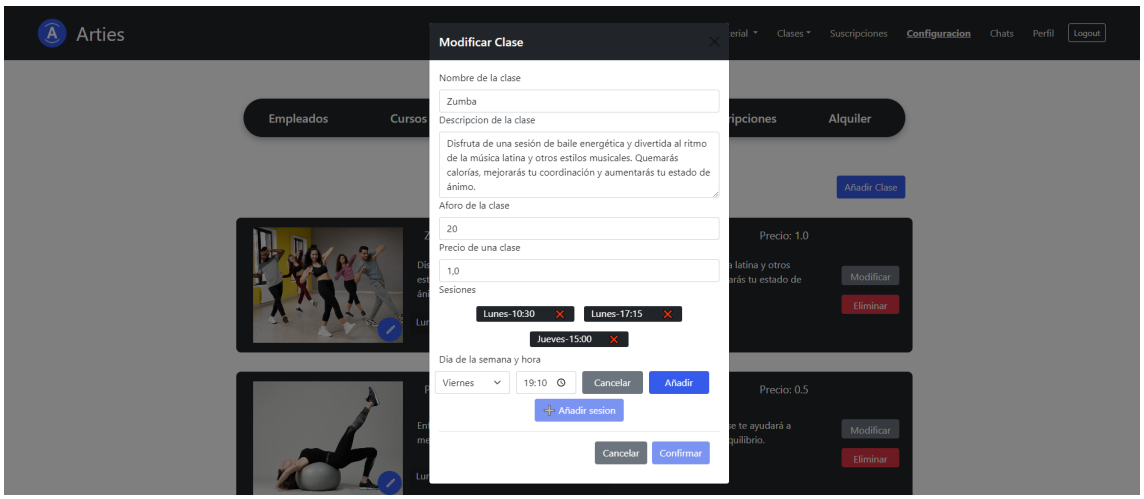


Figura 5.30: Detalle al añadir una sesión

Profundizando más en el funcionamiento de este sistema, destaca la gestión de las sesiones que los usuarios pueden reservar para asistir a las clases. Para evitar problemas de que los usuarios puedan reservar clases para una fecha muy lejana (por ejemplo, para una clase en 3 meses desde la fecha actual) o que no tengan tiempo para hacer las reservas (por ejemplo, que solo puedan reservar para los 5 días siguientes), se ha propuesto una solución intermedia. De esta forma, en todo momento solo hay sesiones disponibles para reservar desde la fecha actual hasta un mes en adelante como mucho. El cliente puede observar esto en como al reservar una plaza para una clase a partir de un mes desde la fecha actual, ya no hay días seleccionables

(figura 5.31). Esta solución permite a los usuarios tener un tiempo más que suficiente para reservar sus clases, y que el gimnasio no deba tener preocupaciones sobre estar gestionando clases que ocurrirán en un tiempo lejano.

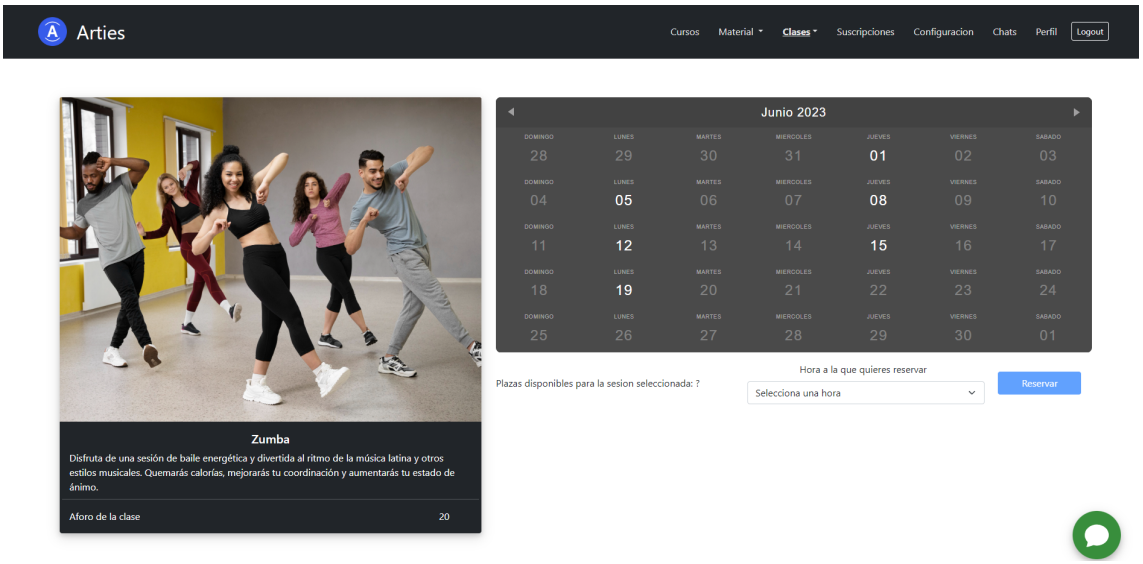


Figura 5.31: Limite visual de días que se pueden reservar

Desde el punto de vista de la programación este aspecto ha necesitado un tratamiento diferente al resto del sistema. En primera instancia, se ha añadido un código que, al crear una clase nueva, automáticamente crea todas las sesiones correspondientes a dicha clase. Para ello se basa en los días y horas de la semana a la que ocurre la clase y crea sesiones los días concretos que toque. Dicha creación de sesiones se limita hasta un mes desde la fecha actual. Para un mejor entendimiento de su funcionamiento se ha creado el diagrama de la figura 5.32.

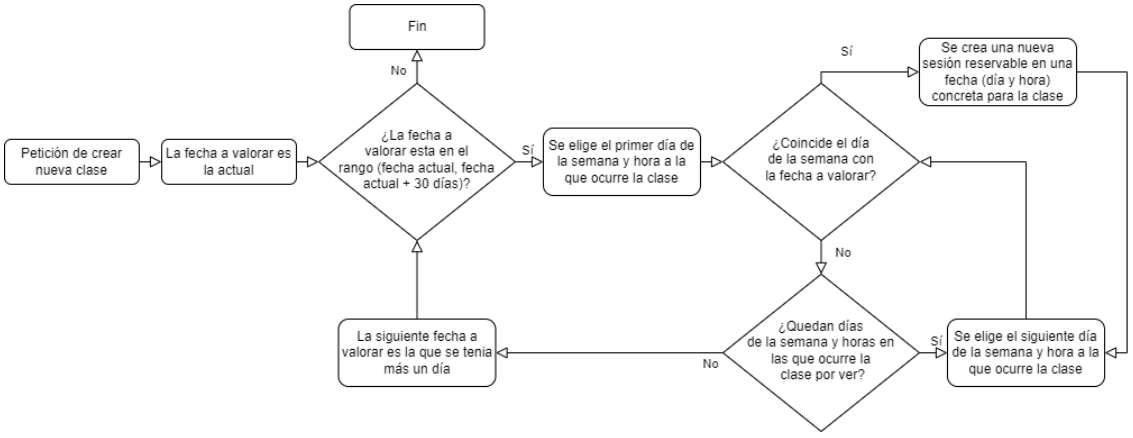


Figura 5.32: Diagrama de creación de sesiones de nuevas clases

Además, las clases siempre deben tener una ventana de tiempo de un mes que permita sus reservas. Esto implica que cada día se deben crear sesiones nuevas para mantener el tamaño de dicha ventana. Para ello se debe registrar un evento que se ejecute cada día y cree las sesiones que toquen en el día cuya fecha será la actual más

un mes. Esto se ha conseguido mediante el uso del paquete *scheduling.annotation* de Spring. Dicho paquete ofrece diferentes funcionalidades para registrar eventos y ejecutarlos bajo determinadas condiciones (Spring, s.f.a). Por ejemplo, se pueden registrar eventos que se ejecuten en un día concreto, que se ejecuten cada cierto tiempo o que se ejecuten a determinadas horas del día entre otros. Esto se consigue mediante la utilización de la anotación *Scheduled* seguida de la expresión que define cuando debe ejecutarse. En el caso del sistema de la aplicación, se definió que debía ejecutarse todos los días a las 00:00, es decir, al comenzar cada día. De esta forma cada día se ejecuta el código encargado de crear las sesiones necesarias para mantener la ventana de tiempo de un mes. Este funcionamiento se puede ver reflejado en el diagrama de la figura 5.33.

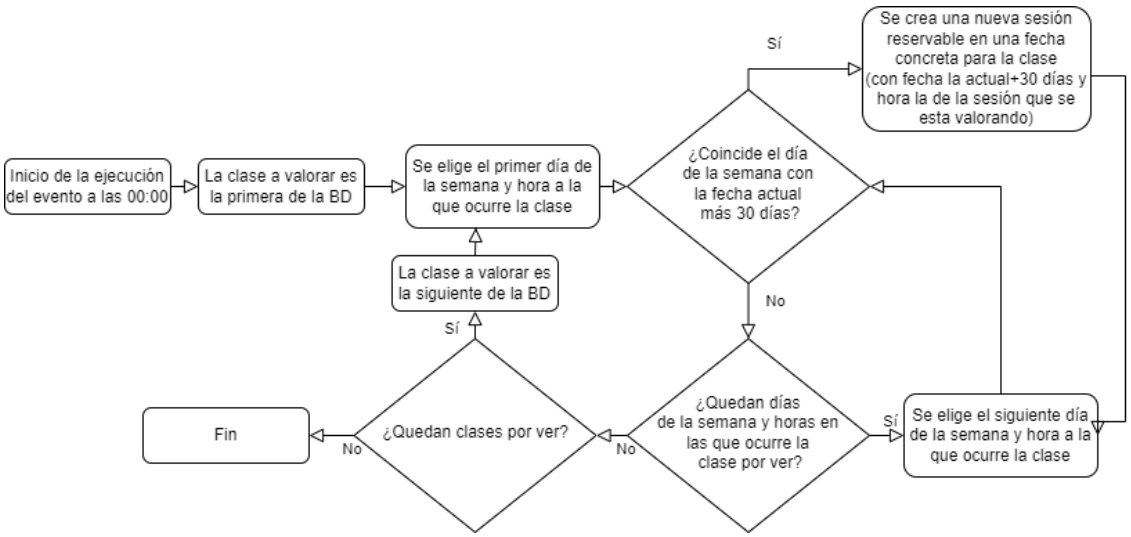


Figura 5.33: Diagrama de evento periódico para crear sesiones

Por último, también se ha añadido un tratamiento especial al sistema de modificar las sesiones de un curso. El administrador al modificar una clase puede eliminar días y horas de la semana a la que ocurre dicha clase y también añadir nuevas. El servidor recibe la lista de días y horas nuevos y la de eliminados. Para los días y horas nuevos se crean sesiones desde la fecha actual hasta un mes de distancia en los días que toque, funcionando de forma similar a la lógica de añadir clase para crear las nuevas sesiones. Para los días y horas eliminados se borran todas las sesiones que ocurrieran en esos días y horas, borrando así también las reservas que hubiera. De esta forma los días y horas que no se hayan modificado se mantienen igual conservando sus respectivas reservas, se crean nuevas sesiones en los días y horas nuevos que se hayan definido, y se eliminan las sesiones de los días y horas borrados, manteniendo los datos de la aplicación un estado consistente.

5.10. Cursos

Una de las funcionalidades principales de Arties es dar servicio a sus clientes de forma *online*. Para esto, entre otras cosas, están los cursos, ya que el cliente no siempre querrá ir al gimnasio o ponerse en contacto con los profesores, además de que puede ser que quiera ir probando diferentes cosas que no conozca hasta encontrar una que le gusta. Una vez más, esta es una funcionalidad que da mucha flexibilidad al usuario pues a través de los cursos podrá elegir qué hacer, aunque no supiese hacerlo, y a qué ritmo.

Los usuarios con algún tipo de suscripción podrán ver todos los cursos dentro del apartado de "Cursos" situado en la *navbar*. Para facilitar al usuario encontrar el curso deseado, o ver qué cursos hay para una temática en concreto, existe un menú desplegable con las categorías de los cursos, establecidas por el gimnasio, como se ve en la figura 5.34; también se pueden ver todos los cursos a la vez por si el usuario lo requiere. De cada curso aparece una tarjeta con su nombre, su descripción, una miniatura y un botón para acceder a su página y ver más información sobre él.

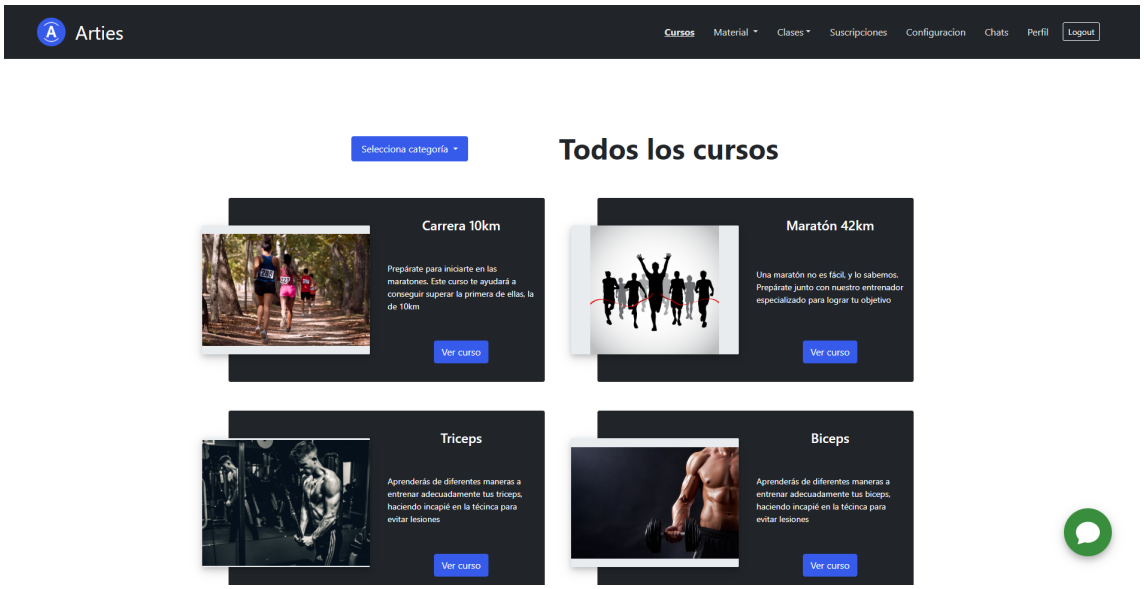


Figura 5.34: Cursos para usuarios con suscripción

Como se puede ver en la figura 5.35, dentro de cada curso se puede ver su nombre, si es exclusivo para socios o no, la descripción del curso, la miniatura y un vídeo donde el usuario podrá seguir a tiempo real los pasos que deberá llevar a cabo. Cabe mencionar que los usuarios que no tengan suscripción actualmente, o que directamente no han iniciado sesión, podrán ver los cursos que el gimnasio haya definido previamente como gratuitos, a lo cual también se accederá desde el apartado "Cursos" de la *navbar*.

Por otro lado, el administrador podrá gestionar todo lo relacionado con los cursos desde el apartado de "Configuración", concretamente desde el panel de "Cursos" de la *subnavbar* (figura 5.36). Desde aquí podrá añadir un curso, lo cuál le redirigirá a

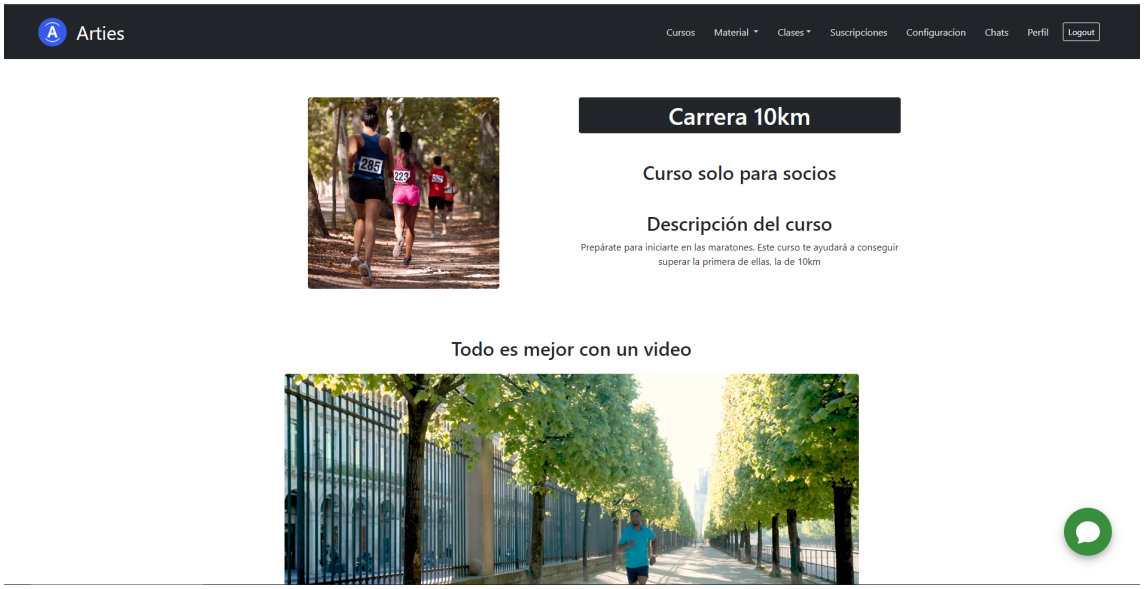


Figura 5.35: Información de un curso

una página como se muestra en la figura 5.37, donde deberá dar obligatoriamente un nombre, una categoría y una descripción al curso, el resto de atributos son opcionales. Estos últimos son la foto de miniatura, un *switch* para indicar si el curso es gratis o no, y otro *switch* que si se activa, creará una nueva sección en la página, para que se pueda agregar el vídeo deseado en formato ".mp4". Los empleados tienen exactamente la misma interfaz y permisos para gestionar los cursos, pero en vez de acceder desde configuración, que solo tiene acceso el administrador, tienen una página llamada "Gestión de cursos" en la *navbar*.

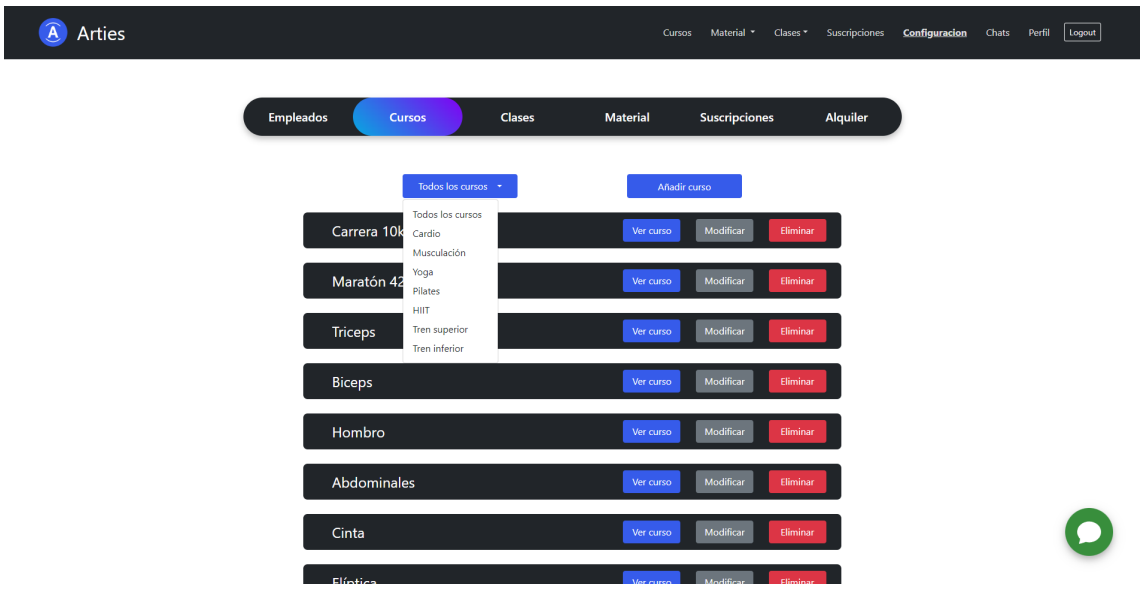


Figura 5.36: Gestión de cursos en Configuración

También en el panel de "Cursos" hay la misma facilidad que se le da al cliente para poder filtrar cursos por categorías. Dentro de cada categoría aparece una lista

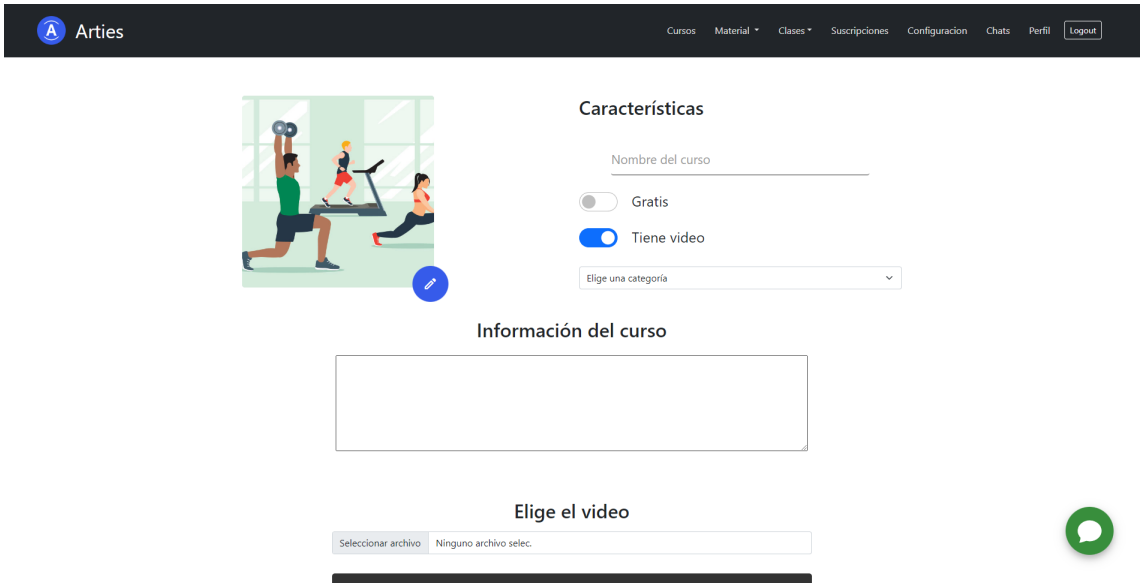


Figura 5.37: Creación de un curso

de los cursos; cada entrada contiene el nombre y tres botones. El primer botón sirve para ir a ver el curso tal cual cómo lo ve el cliente. El segundo redirige a una vista igual que la de añadir curso pero con la información precargada para facilitar al administrador y personal cambiar la información y las características lo más rápido posible. En último lugar se encuentra el botón de eliminar.

5.11. Gestión de usuarios y permisos

La aplicación puede ser utilizada por usuarios con diferentes roles, los cuales les dan acceso a unas u otras funcionalidades. Para ofrecer dicha funcionalidad se utiliza un sistema de permisos que garantiza que las funcionalidades son accedidas solo por aquellos usuarios que tengan acceso a ellas.

Para implementar dicho sistema se hizo uso de las ventajas que proporciona Spring Security. Como se explicó en el apartado 3.1.2 Spring Security además de utilidades de seguridad, ofrece una gestión de permisos basadas en roles. Permite definir aquellas partes de la aplicación a las que tendrán acceso únicamente usuarios con ciertos roles, devolviendo la aplicación un error de acceso en caso de que se intentara acceder a apartados sin permiso. De esta forma se garantiza que cada usuario solo puede acceder a lo que este definido, haciendo la aplicación segura frente a ataques.

La aplicación presenta tres roles principales, siendo estos “Usuario”, “Empleado” y ”Administrador”. Además, la aplicación también diferencia entre usuarios no registrados y sí registrados (que poseerán alguno de los tres roles anteriores). Finalmente, dentro del rol “Usuario” se diferencia entre usuarios sin ninguna suscripción, con suscripción *online* o con suscripción presencial. A continuación, se explicará a

qué funcionalidades tiene acceso cada tipo de usuario, junto a un diagrama de casos de uso que lo representa de forma gráfica.

- **Usuario no registrado:** Este usuario tiene acceso a una funcionalidad básica de la aplicación como se puede ver en el diagrama de la figura 5.38. Se le permite ver la lista de cursos, clases y material que tiene disponible el gimnasio, con el fin de que el usuario sepa lo que se le ofrece y pueda valorar si le interesa apuntarse a dicho gimnasio. Además, para conocer mejor la calidad del gimnasio puede ver aquellos cursos *online* que estuvieran marcados como gratuitos.

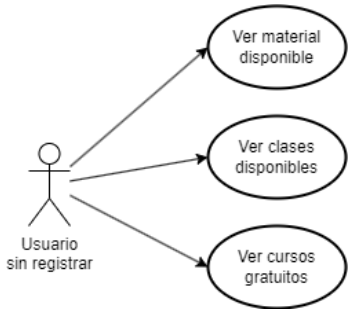


Figura 5.38: Diagrama de permisos usuario no registrado

- **Usuario registrado:** Este usuario tiene acceso a diferentes funcionalidades en función de si tiene o no una suscripción comprada. Al registrarse por primera vez un usuario no tiene ninguna suscripción, teniendo acceso a las mismas funcionalidades de un usuario no registrado, pero incluyendo reserva de clases con previo pago y uso del chat del gimnasio para dudas. Esto se puede ver en la figura 5.39.

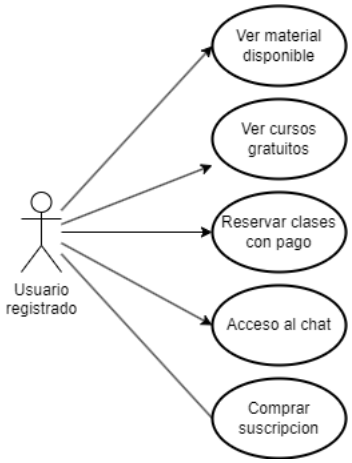


Figura 5.39: Diagrama de permisos de un usuario registrado

- **Suscriptor online:** Se trata de un usuario registrado que ha comprado una suscripción *online*. Como se puede ver en la figura 5.40, esto le da acceso a

todo lo que ya tenía como usuario registrado, más alquiler de material y poder ver todos los cursos *online* que ofrece el gimnasio.

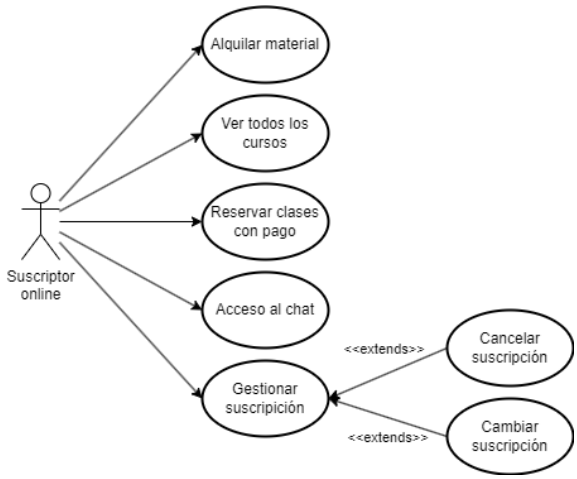


Figura 5.40: Diagrama de permisos de un suscriptor *online*

- **Suscriptor presencial:** Respecto a los permisos, tiene los mismos que el suscriptor online, con la diferencia de que puede hacer reservas de clases sin necesidad de realizar ningún pago como se ve en la figura 5.41.

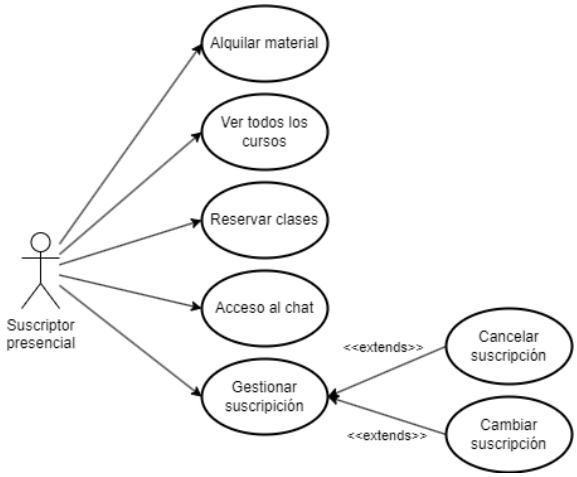


Figura 5.41: Diagrama de permisos de un suscriptor presencial

- **Empleado:** Este usuario es personal del propio gimnasio. Tiene acceso a funcionalidades especiales que usuarios normales no tienen como se ve en la figura 5.42. Esto incluye la gestión de alquileres, la gestión de cursos y la gestión de chats y sus respuestas.

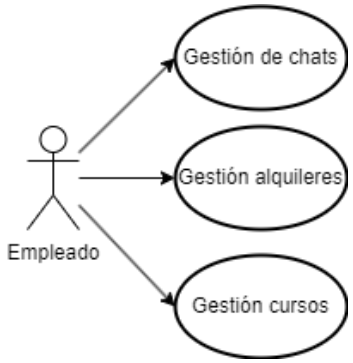


Figura 5.42: Diagrama de permisos de un empleado

- **Administrador:** Se trata del propietario del gimnasio y es el usuario con más permisos, ya que se le permite modificar diferentes apartados del propio gimnasio (figura 5.43). Este usuario tiene acceso a toda la gestión del gimnasio, lo que incluye gestión de material, cursos, clases, suscripciones y de alquileres. Además, se le permite crear, modificar y eliminar usuarios con el rol de "Empleado", por tanto, puede gestionar directamente y en todo momento este tipo de usuarios.

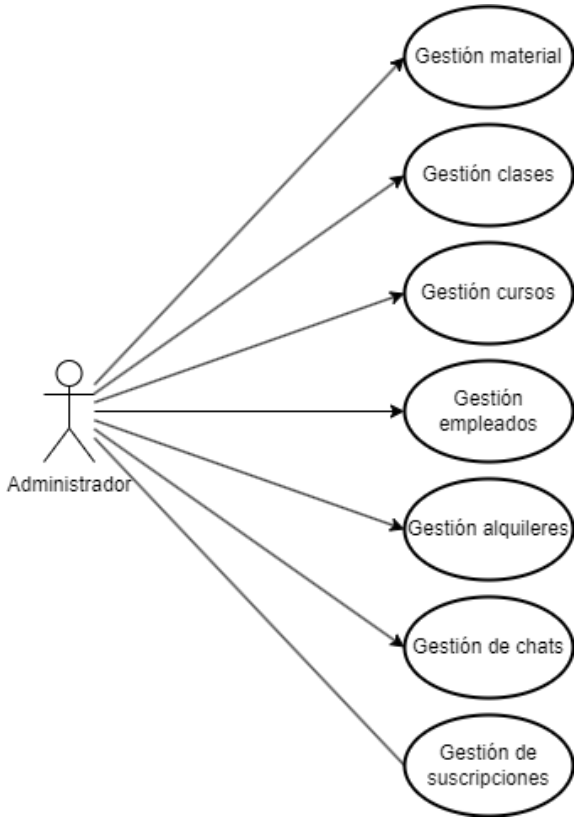


Figura 5.43: Diagrama de permisos del administrador

Aunque todas las comprobaciones de seguridad respecto a los permisos se hacen en el servidor para garantizar la seguridad de la aplicación, la interfaz de usuario

en el cliente se adapta al rol del usuario, haciendo así su navegación más cómoda y evitando posibles errores. Esto se consigue principalmente mediante una *navbar* (menú superior de navegación) cuyos accesos cambian dependiendo del rol (figura 5.44). De esta forma a cada usuario solo le aparecen aquellas funcionalidades a las que tiene acceso. También hay otros componentes que se adaptan a estos permisos, como por ejemplo los botones de alquilar un objeto, los cuales no aparecen si es un usuario sin registrar, o registrado sin suscripción, pero sí aparecen si tiene suscripción *online* o presencial (figura 5.45).

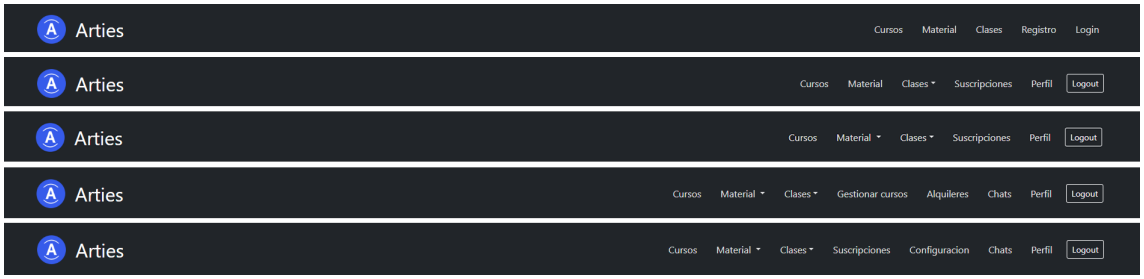


Figura 5.44: Variaciones de la navbar según el tipo de usuario: usuario no registrado, usuario registrado, suscriptor, empleado y administrador)

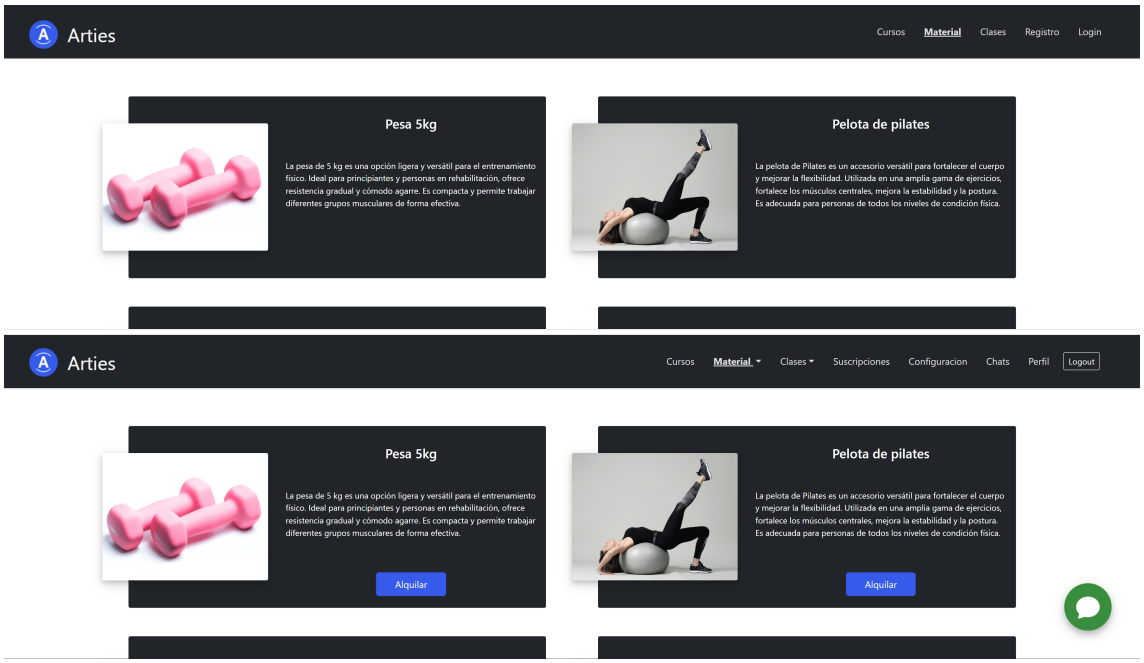


Figura 5.45: Variaciones de botones según los permisos

Finalmente, se incluye un diagrama de casos de uso más detallado con todas las funcionalidades que aporta la aplicación y con los usuarios que pueden acceder a ellas (figura 5.46). En este diagrama quedan definidos a grandes rasgos todas las acciones que los diferentes usuarios pueden realizar en la aplicación como se ha ido explicando a lo largo de este capítulo 5 Descripción del funcionamiento de Arties.

Para reducir el tamaño del diagrama y así facilitar un poco su comprensión se ha utilizado el término “gestión” en material, alquiler, cursos, clases y chat entre otros.

Este término incluye varias funcionalidades que permiten administrar cada uno de esos sistemas. Esto incluye eliminar, añadir y modificar los elementos de ese sistema, además de funcionalidades propias que pueda necesitar para su gestión.

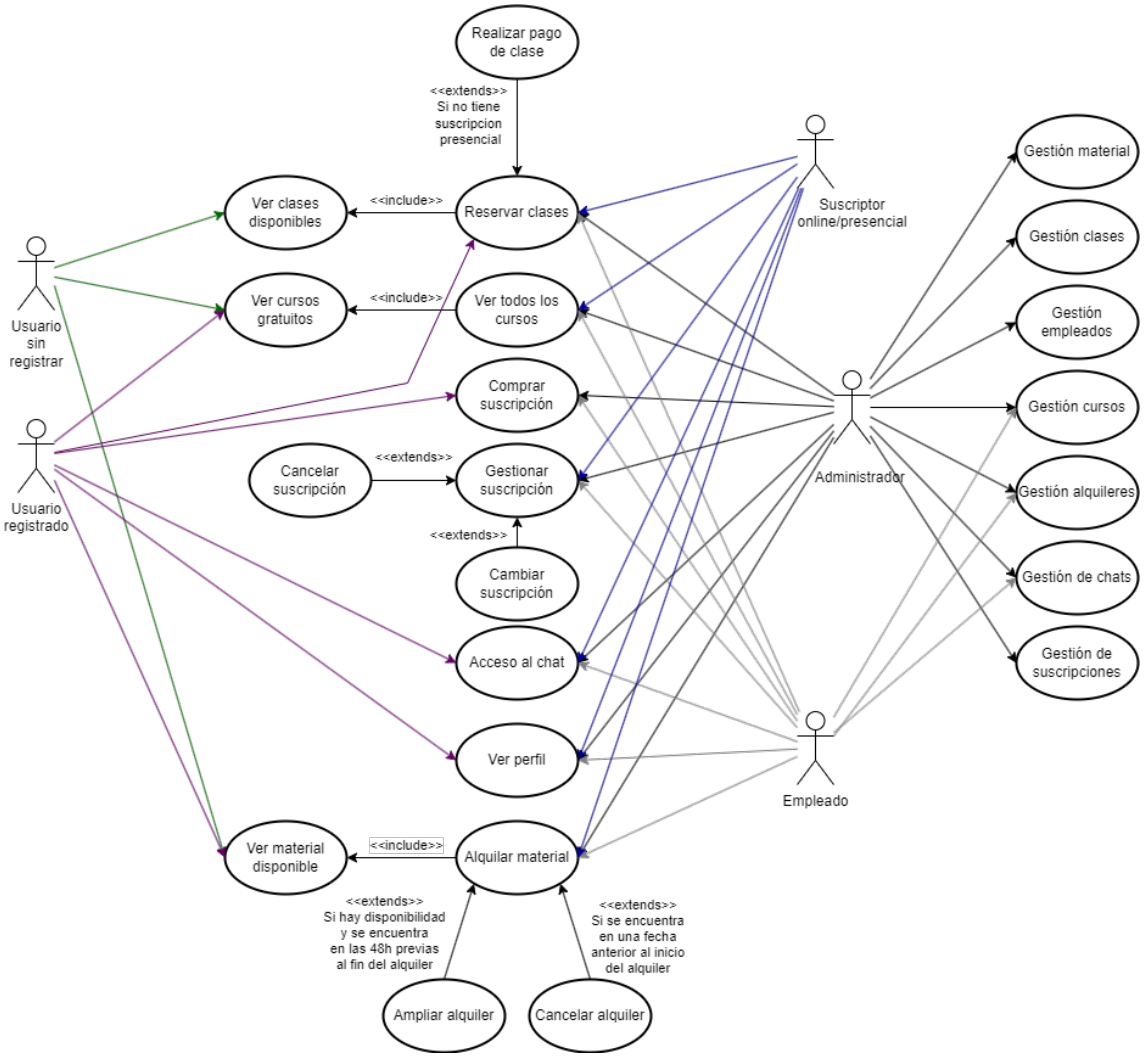


Figura 5.46: Diagrama completo de casos de uso

5.12. Resultado final

Dado que ejecutar la aplicación para probarla requiere de una configuración algo compleja como se explicó en el apartado 3.2 Puesta en marcha, a lo largo de los apartados anteriores se ha hecho una descripción lo más detallada posible para mostrar lo que ofrece la aplicación.

Sin embargo, mediante esos apartados no se puede valorar bien el resultado final del conjunto de todos ellos. Por ello, se ha incluido un vídeo en el que se muestra de forma interactiva todas las funcionalidades que ofrece la aplicación.

En el siguiente enlace se puede acceder a dicho vídeo: `https://www.youtube.com/watch?v=wVsT7tJqjOE&ab_channel=ALBERTOPASCUALFERNANDEZ`

En el vídeo se van recorriendo las diferentes funcionalidades y apartados de la aplicación desde los puntos de vista de los diferentes usuarios que pueden utilizarla. A continuación se incluye también un listado con las marcas de tiempo de las interacciones importantes realizadas en el vídeo, con el fin de facilitar su visionado y permitir revisar más rápido aquellos apartados concretos que se quieran.

Además, el propio vídeo esta dividido en capítulos con dichas marcas de tiempo para poder interactuar directamente con él sin necesidad de estar observando constantemente esta lista.

Listado de marcas de tiempo:

Usuario sin registrar

- **00:00** Inicio.
- **00:14** Ver clases, cursos y material sin estar registrado.
- **00:28** Registro con control de errores.
- **00:59** Login.

Usuario registrado

- **01:10** Uso del sistema de clases.
- **01:25** Pantalla de pago.
- **02:00** Suscripción al gimnasio.
- **02:36** Uso del sistema de cursos.
- **02:56** Uso del sistema de alquiler.
- **03:14** Reservar clase siendo suscriptor presencial.
- **03:45** Uso del sistema del chat a tiempo real.

Empleado

- **05:22** Gestión de cursos (vista empleado) con ejemplo de modificación.
- **05:58** Gestión de alquileres (vista empleado) con ejemplo de finalización y deshacer dicha acción.
- **06:11** Historial de alquileres.

Administrador

- **06:30** Uso del sistema de ampliación de alquiler.
- **06:40** Perfil con ejemplos de modificación de datos e imagen.
- **07:01** Gestión de empleados.
- **07:33** Gestión cursos (vista administrador).
- **07:37** Gestión de clases.
- **07:48** Quitar y añadir una sesión a una clase.
- **08:10** Gestión de material.
- **08:18** Gestión de suscripciones con ejemplo de modificación.
- **08:45** Gestión alquileres (vista administrador).

Por último, se incluye a continuación un enlace al proyecto de GitHub. En él se puede ver como fue el desarrollo del mismo, y se puede descargar el código fuente para aquellas personas que quieran probar directamente la aplicación siguiendo los pasos del apartado 3.2 Puesta en marcha.

Enlace del proyecto: `https://github.com/AndresRomero01/TFG`

Capítulo 6

Diseño y usabilidad

Resumen: En este capítulo se explican las decisiones de diseño tanto visuales como estructurales. También se exponen y justifican los principios de diseño seguidos para conseguir una buena usabilidad.

6.1. Diseño

En este apartado se desarrollan aquellos aspectos generales sobre la identidad visual de la aplicación, así como la estructura de la interfaz y las bases de su diseño.

6.1.1. Identidad visual

Esta aplicación se presenta como una solución moderna que permite hacer un deporte de calidad más accesible a todo tipo de usuarios, al mismo tiempo que proporciona nuevos beneficios a los gimnasios.

Por este motivo se decide utilizar el nombre de Arties. De origen vasco y con significado "llano entre dos aguas" representa el concepto de la aplicación. Esta se sitúa entre dos extremos (clientes y gimnasios) con el fin de unirlos mediante un sistema que les proporcione un bien común. De esta forma, la aplicación no pretende beneficiar a uno a costa del otro, sino situarse en un punto intermedio que pueda ofrecer ventajas a ambos, mejorando así también la relación entre clientes y gimnasio.

Al tratarse de una solución innovadora se han utilizado como colores principales el azul y el negro, ya que su combinación es utilizada en aspectos tecnológicos y modernos. Además, el color azul, desde el punto de vista de la psicología del color

(Heller, 2008), transmite confianza y estabilidad, dos valores importantes en la aplicación, pues uno de los aspectos principales es mejorar la relación entre gimnasio y clientes.

Los códigos de colores utilizados han sido #365BEA para el azul principal, #212529 para los fondos de los elementos como la *navbar*, *footer*... y #6C757D para acciones secundarias como los botones de modificar, ver perfil... Para tener una mejor idea de estos colores, se muestran en la figura 6.1.

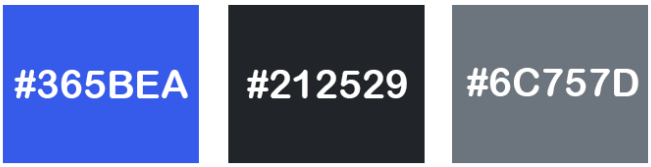


Figura 6.1: Colores seleccionados para Arties

Respecto al diseño del logo se ha optado por un aspecto simple, minimalista y fácil de recordar. El logo se ha diseñado de forma que siga el concepto principal de la aplicación y de su identidad (conexión entre gimnasios y clientes). De esta forma la A de Arties se encuentra entre dos marcas (una superior y otra inferior) que representan ambos extremos, conectados mediante esta A. Como colores se ha escogido el azul por ser el principal de la aplicación, y el blanco, que ofrece un buen contraste con el azul y con los colores oscuros utilizados. El resultado final se puede ver con más detalle en la figura 6.2.

Figura 6.2: Logo de la aplicación

6.1.2. Uso de imágenes

Para poder mostrar todo el potencial que ofrece la aplicación ha sido necesario incluir imágenes de prueba. De esta forma elementos como cada curso, clase y material tiene una imagen que lo representa.

Al no disponer de los medios necesarios para la creación de dichas imágenes se ha optado por utilizar imágenes de internet, siempre respetando los derechos de

propiedad intelectual. Para ello se han utilizado páginas web que contienen imágenes de uso libre.

Además, se han referenciado correctamente las imágenes mediante una página que contiene todas las menciones de las imágenes utilizadas y sus autores, accesible desde el *footer* de la aplicación.

6.1.3. Estructura de la web

Lo más importante en cuanto a la estructura de la web es que mantenga consistencia, tanto universal como interna, además de que sea eficiente. Por ello se ha escogido un modelo de navegación *Fully connected*. Este consiste en que haya una página de inicio y todas las páginas sean accesibles desde todos lados (generalmente a través de una *navbar*). Cabe destacar que la página de Configuración tiene diferentes funcionalidades que no son accesibles desde el resto de páginas, por lo que aumenta en uno el número de clics necesarios. No obstante, hemos tomado esta decisión porque así todo lo relativo a la gestión del negocio está en un mismo lugar, aumentando la facilidad de uso y memorabilidad para el administrador.

6.1.4. Diseño adaptativo

Esta aplicación está enfocada al uso tanto del personal del gimnasio como de los propios clientes. Los primeros la utilizan en un contexto de gestión, como es el típico de las aplicaciones de escritorio, siendo el formato habitual una pantalla grande como la de un portátil o monitor. Los clientes sin embargo pueden utilizarla en un entorno más distendido desde sus casas, siendo habitual el uso de ella mediante el teléfono móvil. Por ello se ha tenido en cuenta este aspecto y se han implementado funcionalidades extras que permiten que la aplicación se adapte a pantallas pequeñas en todas las partes posibles.

De esta manera, ciertos componentes y pantallas se han adaptado para ser *responsive*, es decir, que los tamaños y disposición de los elementos se adaptan en función del tamaño de la pantalla que los visualiza.

Para lograr esto se ha recurrido a diferentes métodos que se explicarán en los siguientes subapartados.

6.1.4.1. Uso de las ventajas de Bootstrap

Bootstrap incluye utilidades que facilitan el desarrollo de una aplicación *responsive*. Los principales componentes para lograr esto es el uso de filas y columnas. Estas ultimas si no entran en una fila, se van apilando, de forma que una fila de varios elementos en una pantalla grande, pude convertirse en una fila de uno o dos

elementos en una pantalla pequeña. Para lograr un correcto resultado se ha hecho uso también de los *breakpoints* que incluye Bootstrap. Con ellos se puede indicar hasta que tamaño un componente puede reducirse como máximo, de forma que cuando llegue a ese tamaño, pase a colocarse en la siguiente fila. Esto, en combinación con las filas y columnas, permite que una interfaz que presenta una lista de elementos se adapte dinámicamente a los tamaños de las pantallas como se puede ver en la figura 6.3.

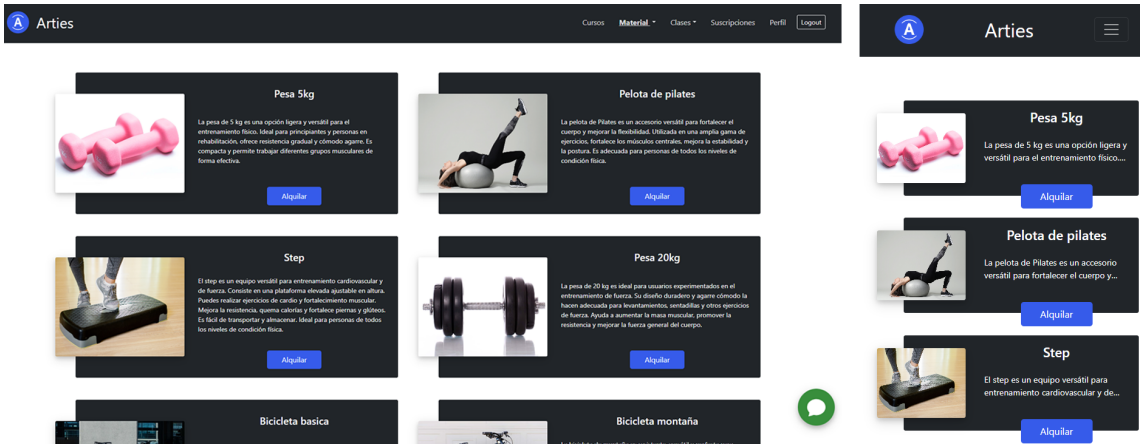


Figura 6.3: Comparación listas *responsive*

Otra de las utilidades que aporta Bootstrap es el uso de una *navbar responsive*. De esta manera se ha diseñado una *navbar* que al visualizarse en pantallas de dispositivos móviles pasa a convertirse en un menú hamburguesa, típico de aplicaciones móviles, y que permite una cómoda navegación por los diferentes apartados de la aplicación (figura 6.4.)

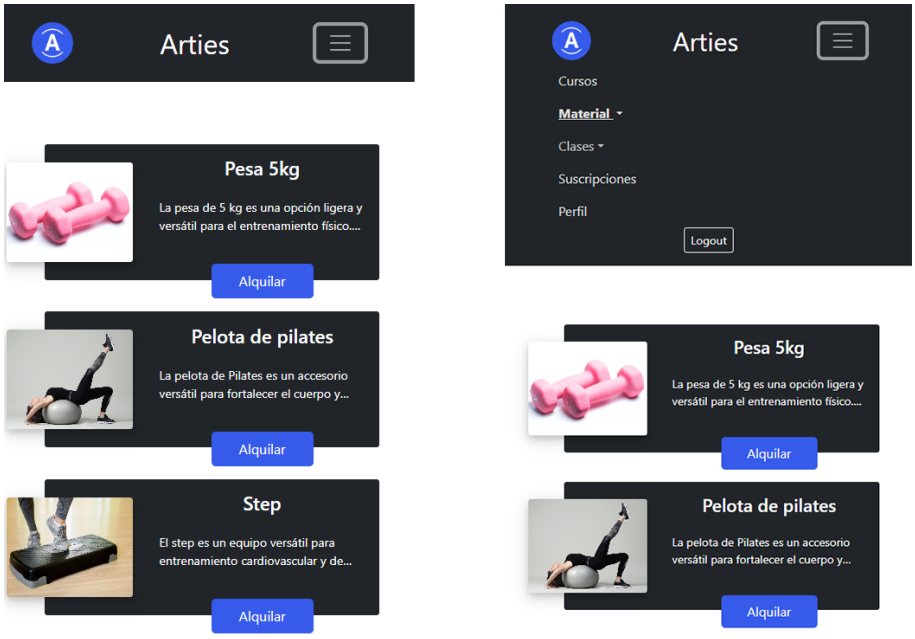


Figura 6.4: Comparación navbar responsive

6.1.4.2. Creación de elementos adaptativos

Hay elementos que aparecen repetidos en diferentes lugares de la aplicación como puede ser las tarjetas o el calendario. Para asegurar su correcta visualización en los diferentes lugares donde se utilicen, dichos componentes deben funcionar de manera *responsive* independientemente del contexto en el que se incluyan. Por ello el código que define su aspecto visual, también incluye código condicional que se incluye o no dependiendo del tamaño de la pantalla. Esto se ha conseguido mediante el uso de las *media queries* de CSS, las cuales permiten aplicar código CSS extra si se cumplen ciertas características, como por ejemplo que se aplique en función de cierto tamaño de pantalla. En algunos casos también ha sido necesario código JavaScript extra que modificara los elementos en tiempo real en función del tamaño de la pantalla.

Este código extra modifica el aspecto original para que el elemento se adapte correctamente. Por ejemplo, en el caso de las tarjetas, se reduce la cantidad de texto que se puede mostrar y se desplaza el botón verticalmente para aprovechar más el espacio disponible en una pantalla de teléfono móvil como se puede ver en la figura 6.5. En el caso del calendario, se reduce el tamaño de las letras y de los números, y se sustituyen los días de la semana de palabras completas a las versiones abreviadas de una letra (figura 6.6)

Figura 6.5: Comparación tarjeta *responsive*

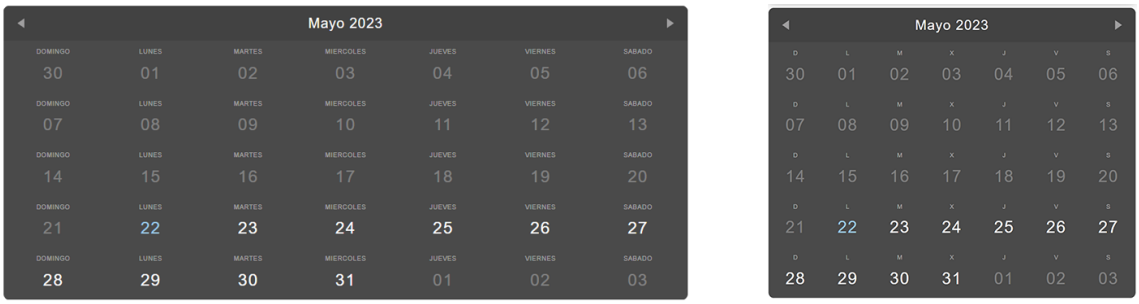


Figura 6.6: Comparación calendario *responsive*

El uso de estos componentes en combinación con las utilidades de Bootstrap ha permitido que ciertas vistas de la aplicación pasen a ser *responsive* automáticamente sin necesidad de cambios extra, como es el caso de la vista de la lista de material o la lista de clases entre otras (por ejemplo, como se puede ver en la anterior figura 6.3).

6.1.4.3. Adaptación personalizada de pantallas

Determinadas pantallas debido a su diseño y estructura no se adaptan automáticamente mediante el uso de Bootstrap o elementos propios *responsive*.

En dichos casos se ha tenido que añadir código extra exclusivo para dichas pantallas, que permita que se adapten a pantallas de dispositivos móviles.

Esto ocurre en las páginas de inicio, alquiler de material, visualización de cursos, reserva de clases y suscripciones. En general en dichas páginas se han realizado cambios similares. Se ha reducido el margen horizontal de la pagina y se han dispuesto los elementos en un formato vertical en vez de horizontal. Eso sumado a detalles como ajustes en los tamaños de texto y botones o ajuste de márgenes y rellenos ha permitido que se adapten correctamente a pantallas de móviles. En la figura 6.7 se puede ver el resultado final de dichas pantallas.

Figura 6.7: Pantallas versiones móvil

6.2. Usabilidad

Arties es una web que busca la facilidad de uso, eficiencia y perfección. Para ello se han seguido los principios de Nielsen (1994) y las Reglas de oro de Shneiderman (2016). Estas son dos de las principales referencias en cuanto a diseño de interfaces de usuario, y aunque son parecidas, Shneiderman se centra más en la interacción entre el usuario y la máquina, mientras que Nielsen proporciona unas pautas más genéricas de diseño para una mejor usabilidad. De esta forma se consigue una aplicación más usable y amigable. A continuación se exponen, justifican y se da ejemplos de los principales principios de diseño seguidos.

6.2.1. Visibilidad del estado del sistema

Este principio establece que se debe mantener siempre informado al usuario de lo que esta ocurriendo en el sistema. De esta forma el usuario no se siente perdido y conoce en todo momento las implicaciones de sus interacciones y su repercusión en el sistema. Para lograr esto se han implementado diferentes soluciones que informan al usuario de lo que está ocurriendo.

Por un lado, se ha hecho uso de JavaScript y de AJAX para actualizar la interfaz de forma dinámica en función de las acciones del usuario. Por ejemplo, en el caso del administrador al eliminar una clase, esta desaparece de la lista de clases disponibles, dejando claro así que dicho elemento ha sido eliminado.

Otro aspecto que sigue este principio es previsualizar las imágenes y vídeos que el personal sube a la aplicación. De esta forma, por ejemplo, cuando va a crear o modificar un curso y elige el vídeo y la imagen del mismo, estos elementos se previsualizan antes de enviarlos (figura 6.8). Así, el administrador puede saber que el vídeo y la imagen están ya listos para enviarse, además de poder comprobar que no se ha equivocado seleccionando el archivo.

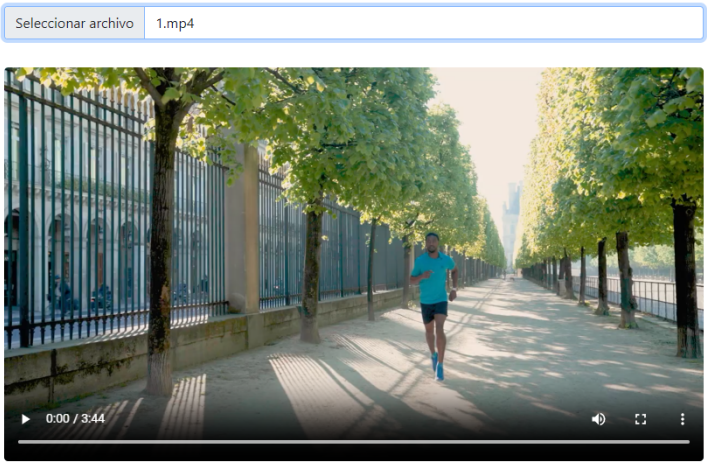


Figura 6.8: Previsualización del contenido subido

Otro elemento utilizado para complementar a los dos anteriores es el uso de los *toast* de Bootstrap. Estos son mensajes superpuestos que son visibles solo durante cierto tiempo y que informan de la realización de alguna acción. Se utilizan en diferentes apartados de la aplicación, como, por ejemplo, al mandar una pregunta por el chat genérico confirmando su envío, o al crear y modificar los cursos (figura 6.9). En estos casos son especialmente importantes, ya que al realizar dichas acciones no se producen cambios en la interfaz, por tanto, estos mensajes sirven para indicar al usuario que todo ha funcionado correctamente.

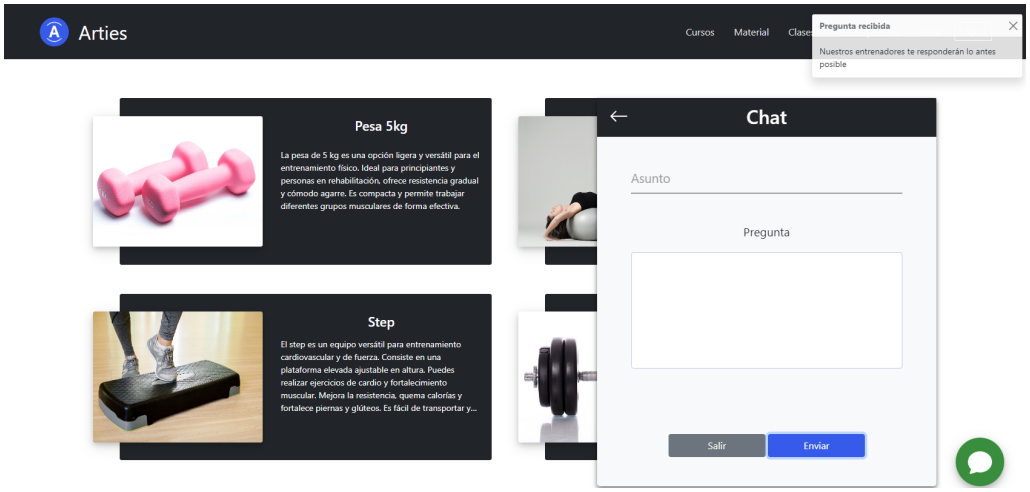


Figura 6.9: Ejemplo de mensaje de confirmación

Por último, se incluye un mensaje informativo en las páginas donde el usuario puede ver sus alquileres y sus reservas de clases cuando no tiene ninguna. De esta forma, en vez de aparecer una página vacía en la que el usuario podría pensar que algo no esta funcionando bien, se muestra un mensaje indicando que no tiene ningún alquiler o reserva, y se le invita a realizar uno (figura 6.10).

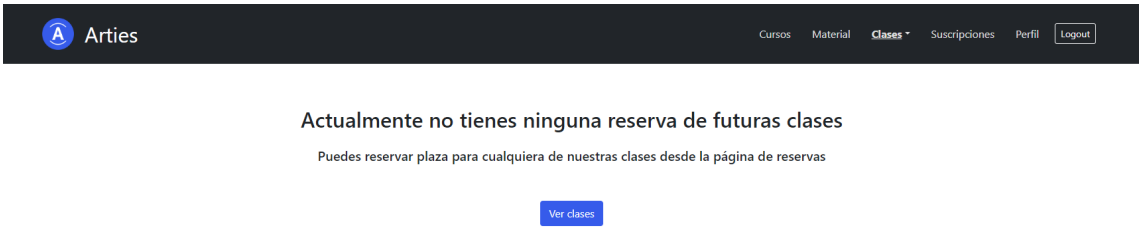


Figura 6.10: Mensaje informativo indicando que no hay reservas

6.2.2. Prevención de errores

Los errores no solo son una pérdida de tiempo para el usuario, sino que también empeoran su experiencia usando la aplicación. Por lo tanto es mejor tratar de prevenir al usuario de cometer errores. Algunos ejemplos de cómo se ha hecho en la *app* son los siguientes:

Cuando el usuario tiene que escoger un archivo, ya sea foto o vídeo, solo se le permite escoger de su sistema aquellos archivos que tengan la extensión adecuada. Esto se hace mediante el atributo *"accept"* de HTML, evitando así que el usuario suba un archivo que el servidor va a rechazar y obtenga un error.

Otro apartado en el que se ha tenido cuidado en este aspecto es en el diseño del calendario al realizar alquileres de materiales y reservas de clases. Para evitar que los usuarios puedan seleccionar fechas sin sentido se han desactivado en ambos casos todos los días anteriores al actual. (figura 6.11). Además, en el caso de reservas de plazas para clases, solo salen activados aquellos días en los que hay sesiones de la clase a reservar. Esto incluye los días de la semana en la que ocurre hasta llegar a un mes desde el día actual. A partir de esa fecha todos los días siguientes salen desactivados (figura 6.12).

Figura 6.11: Calendario con fechas anteriores a la actual desactivadas

Figura 6.12: Calendario con solo las fechas activadas en los días que hay sesiones

6.2.3. Identificación y recuperación de errores

Diseñar un sistema para prevenir errores del usuario es fundamental, pero no siempre es posible evitar todos los errores. Por esto también es de vital importancia que en cuanto suceda uno, se informe al usuario claramente, con lenguaje que entienda y evitando códigos de error. Esto se ha tenido en el diseño de la aplicación como se explicará a continuación.

Cuando el usuario intenta usar un nombre de usuario existente, ya sea desde registro o desde cambiar datos en el perfil, aparece un error que informa al usuario. Este impide que se realicen los cambios, pero no se reinician los campos del formulario, haciendo que sea lo más cómodo posible. Lo mismo sucede si escribe las contraseñas de forma que no coincide la una con la otra. Estos errores se pueden ver en la figura 6.13.

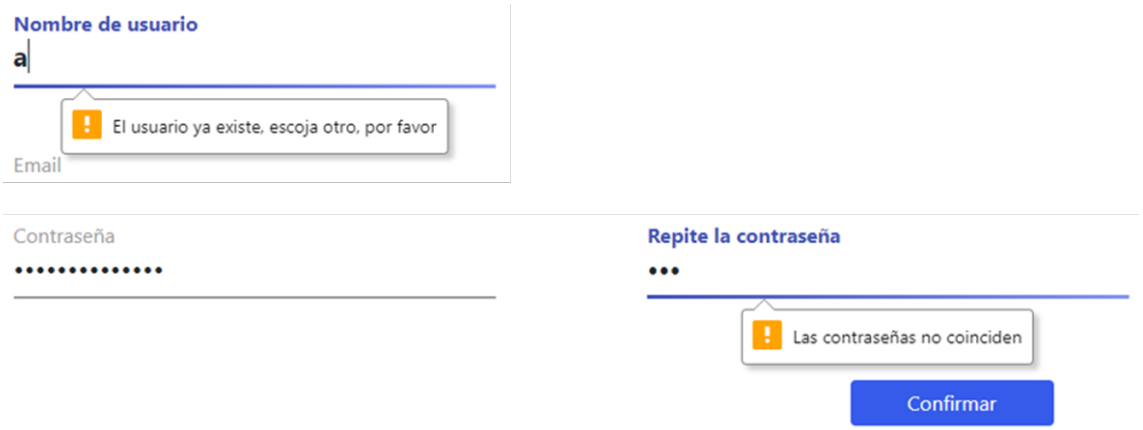


Figura 6.13: Errores en datos de usuario

Otro sistema donde se ha tenido en cuenta la importancia de este principio de usabilidad es en el de gestión de alquileres.

Finalizar un alquiler se trata de una acción que los empleados del gimnasio realizan muy a menudo, pero con unas implicaciones importantes en la aplicación, ya que finalizarlo supone que el usuario ha devuelto el objeto y que podrá alquilarlo de nuevo más adelante. Si se finalizara un alquiler por error podría suponer que un usuario pueda quedarse indefinidamente con un objeto y además alquilar nuevos.

Por tanto, la acción de finalizar un alquiler debe situarse en un punto intermedio entre ser una acción rápida pero que pueda asegurarse que se hace sin errores. Para ello, al finalizar un alquiler se muestra un mensaje informativo, el cual contiene un botón para deshacer dicha acción (figura 6.14). Si se pulsa dicho botón, el alquiler volverá a estar en vigor como si nada hubiera pasado. Esta implementación permite que finalizar un alquiler sea una acción rápida, al no tener diálogos de confirmación, pero también segura al permitir deshacer la acción fácilmente.

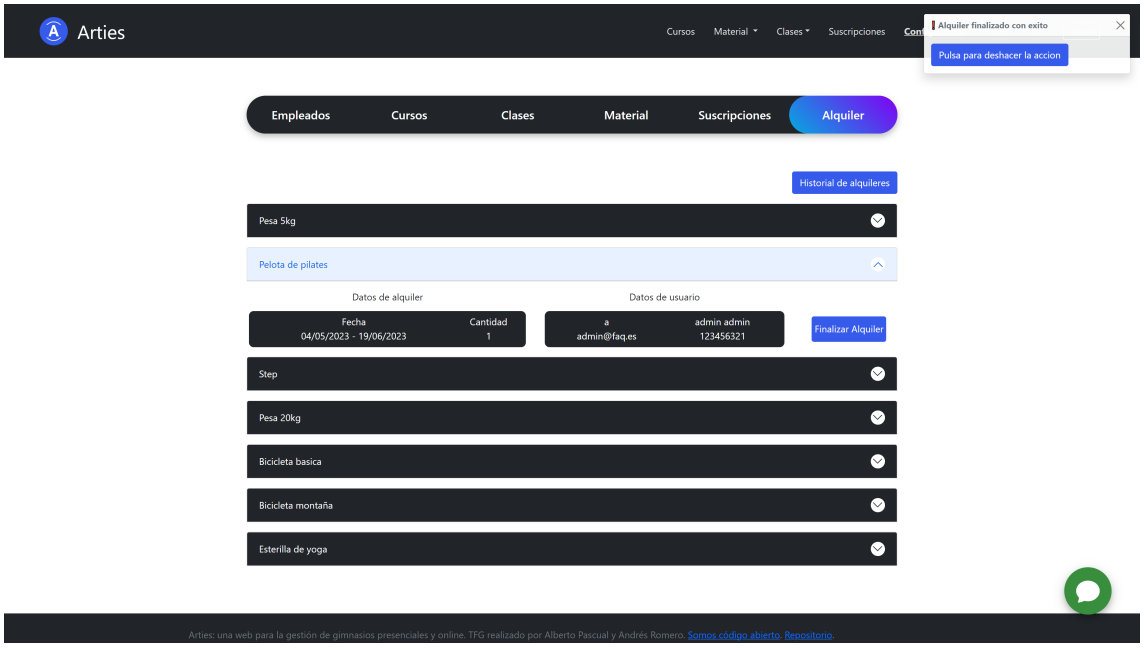


Figura 6.14: Deshacer finalización de un alquiler

6.2.4. Comentarios informativos

Es necesario informar claramente al usuario de todo lo que ocurre en la aplicación y de las opciones que tiene sobre ella. Para ello cobran vital importancia el uso de descripciones y comentarios que ayuden al usuario y le aporten información útil.

Eso ocurre por ejemplo en la pantalla donde un usuario puede ver sus alquileres. Un alquiler tiene diferentes funcionalidades dependiendo de la fecha en la que se vea (puede ser o no cancelable, o puede ser o no ampliable). Para informar de dichas opciones se incluye un botón de información en cada alquiler que dependiendo de la situación en la que se encuentre el usuario, mostrara un mensaje u otro (figura 6.15). De esta forma, el usuario conoce en todo momento las opciones que tiene disponibles en cada uno de sus alquileres.

Figura 6.15: Variaciones del mensaje informativo de alquileres

Respecto a esto, hay que destacar que dichos mensajes también aparecen si el usuario entra a la página de alquiler de algunos de los objetos que ya tenía (figura 6.16).

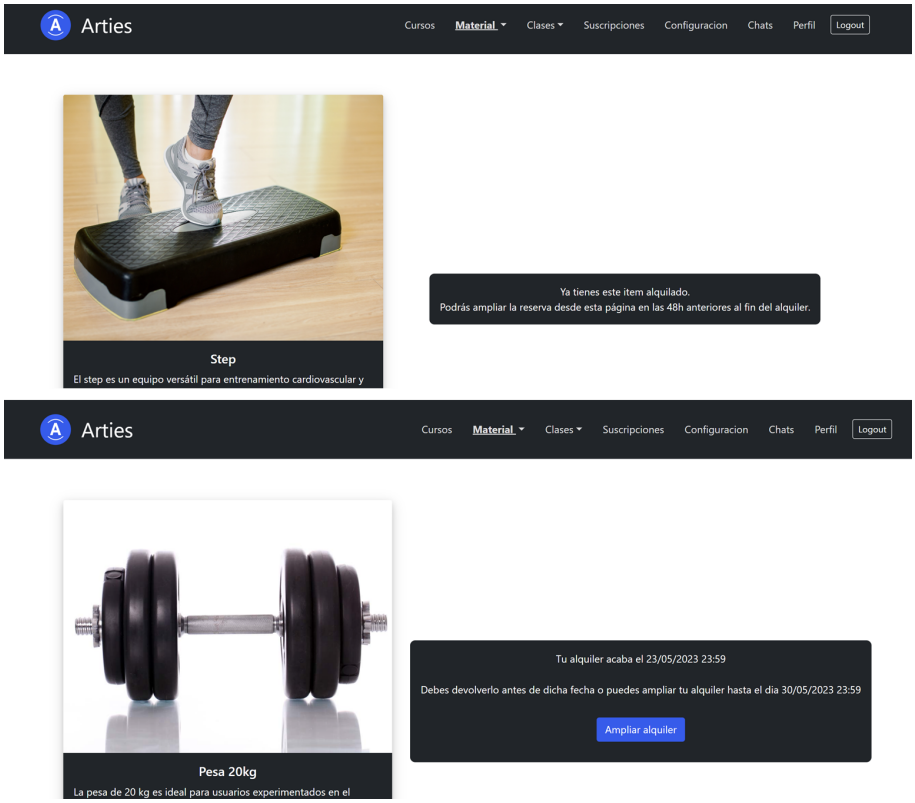


Figura 6.16: Variaciones del mensaje informativo en la página de alquilar

Otro aspecto relacionado con este principio es resaltar cierta información importante para el usuario, de forma que pueda acceder a dicha información rápidamente. Esto se aplica, por ejemplo, en los alquileres, en los cuales, cuando un alquiler no ha sido devuelto, sus fechas aparecen en rojo remarcando dicho alquiler para que el usuario sepa que es importante (figura 6.17). También se aplica en la vista de administrador, al crear o modificar una clase cuando se elimina una sesión. Como dicha acción conlleva eliminar las reservas de dichas sesiones, siendo por tanto una acción con gran importancia, se remarca al usuario las consecuencias mediante un llamativo mensaje en rojo (figura 6.18).

Figura 6.17: Mensaje de advertencia de alquiler vencido

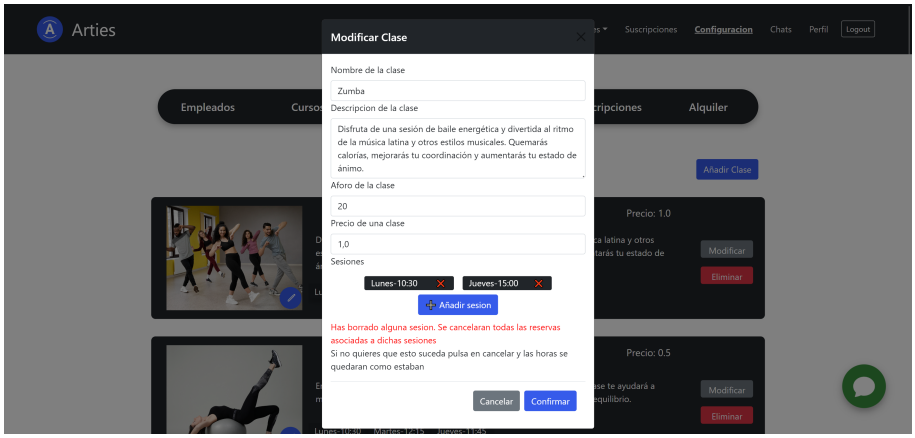


Figura 6.18: Mensaje de advertencia al eliminar sesiones

Por último, este principio también se aplica en la pantalla de perfil de usuario. En esta página se presenta al usuario un apartado para que explique cosas sobre él. La finalidad de esta información, es que los profesores del gimnasio puedan saber sobre los hábitos y estado físico del usuario con el fin de hacerle mejores recomendaciones. Por ello, se incluye una pequeña descripción que indica al usuario qué tipo de información debe introducir, y que así esta pueda ser realmente de utilidad para el personal del gimnasio (figura 6.19).

Figura 6.19: Mensaje informativo en el perfil

6.2.5. Consistencia

En Arties se ha procurado mantener tanto la consistencia externa (similitudes con otras webs) como la interna (similitudes dentro de nuestra página).

La consistencia externa es una característica muy importante para facilitar al usuario aprender a usar la aplicación y que no le cause rechazo. Para lograrlo, se han tenido en cuenta varios factores, como los siguientes patrones de navegación:

- **Escape Hatch:** Cada pantalla tiene un botón o enlace que permita claramente volver a una página bien conocida por el usuario, como ocurre con el logo de la esquina superior izquierda que lleva a inicio.
- **Sign-in Tools:** Consiste en poner todo lo relacionado a inicio de sesión, registro, perfil y cerrar sesión en la esquina superior derecha.

Otros elementos que proporcionan consistencia externa son el icono de chat en la esquina inferior derecha, los iconos flotantes de cambiar imagen en las respectivas esquinas inferiores derecha de las fotos, el uso de una *navbar* para la navegación...

La consistencia interna permite al usuario familiarizarse con la aplicación con mayor rapidez, consiguiendo realizar tareas similares o entender lo que está viendo más fácilmente. Un ejemplo sería que tanto el sistema de reservar clases como el de reservar material son muy parecidos, ya que mantienen la misma estructura de la página, con el mismo calendario para elegir el día... Otros ejemplos es la consistencia en el uso de colores dentro de la app, sabiendo que el azul es un elemento clicable, que el rojo es para eliminar, el gris es para modificar...

6.2.6. Diseño estético

Este principio establece que la interfaz debe ser limpia, atractiva, entendible y minimalista.

Esto se ha aplicado en los diferentes apartados de la aplicación, entre los cuales por ejemplo se podría destacar la implementación de un calendario personalizado que favorece la interacción con el usuario.

Se ha utilizado como sustituto al *"input date"* de HTML (figura 6.20) para conseguir una interacción con el usuario más atractiva y agradable.

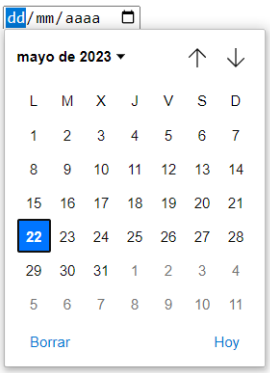


Figura 6.20: Formato de *input date* de HTML

Este calendario permite cargar dinámicamente los días de los meses que seleccione el usuario y ofrece varias ventajas respecto al *"input date"* de HTML. Por un lado, presenta las fechas en un formato más grande, haciéndolo más fácil de utilizar y

ver (figura 6.21). Por otro lado, es personalizable, lo que permite definir qué días se pueden seleccionar o no, además de destacar días concretos.

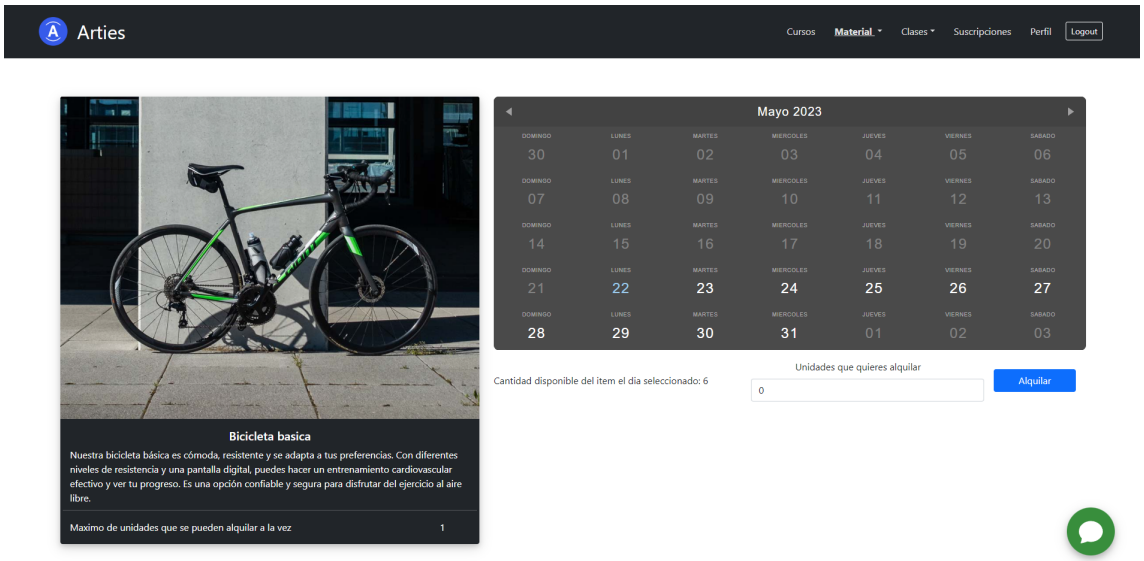


Figura 6.21: Resultado del calendario diseñado

De esta forma se consigue una interacción más natural con el usuario mediante este elemento, que, aun siguiendo un estilo simple y minimalista, ofrece una interacción sencilla y agradable.

6.2.7. Reconocer antes que recordar

Tanto Shneiderman como Nielsen coinciden en que se debe reducir al máximo posible la carga de memoria para el usuario, haciendo bien visibles los elementos que provoquen acciones. La forma más típica de conseguir esto es con menús constantemente visibles, como es el caso de la *navbar*. Además, como se ha ido sugiriendo en algunos de los principios anteriores, cuanto más consistencia haya, mayor número de patrones de diseño implementados, mejor estructura... menor número de cosas tendrá que memorizar el usuario y por tanto mayor será la usabilidad. Así, al usuario le valdrá con reconocer los elementos que está viendo para saber qué hacen y cómo lo hacen.

Capítulo 7

Conclusiones y Trabajo Futuro

Resumen: En este capítulo se comentan las conclusiones sacadas por los miembros del equipo tras la realización del proyecto. Además se proponen mejoras que se podrían realizar en la aplicación.

7.1. Conclusiones

Se ha realizado un proyecto que cubre todas las necesidades de los usuarios planteadas en el apartado 1.1 Motivación, y por tanto consiguiendo implementar una aplicación innovadora y que sea realmente de utilidad para gimnasios y usuarios.

Su realización se ha hecho mediante tecnologías actuales y de uso profesional a día de hoy como se ha explicado en el apartado 3.1 Tecnologías empleadas. A esto se suma una metodología y plan de trabajo muy utilizado hoy en día como es Scrum, consiguiendo así simular correctamente el desarrollo de una aplicación web profesional.

Una especial atención a la seguridad de la aplicación mediante el uso de Spring Security y los permisos ha permitido el desarrollo de una aplicación segura y factible de utilizar en un entorno real.

Además, gracias a la importancia dada al diseño de la interfaz y a la aplicación de principios de usabilidad, se ha desarrollado una aplicación con un aspecto moderno y atractivo, con una alta usabilidad.

Se han conseguido cumplir todos los objetivos propuestos en el apartado 1.2 Objetivos para el desarrollo del proyecto, quedando reflejados en su desarrollo los diferentes conocimientos obtenidos en el grado, como son metodologías de desarrollo software, patrones de programación, importancia de la usabilidad, diseño de bases de datos...

Por todo ello se considera que el desarrollo del proyecto de Arties ha sido un éxito, cumpliendo con todos los objetivos y requisitos necesarios.

7.2. Trabajo futuro

Una web como Arties conlleva mucho trabajo y por tanto hay secciones que se pueden llevar a cabo mejor, solo es cuestión de tiempo. Además todo siempre se puede mejorar. Aunque el resultado final ha sido bueno, a continuación se exponen funcionalidades y detalles que mejorarían en mayor o menor medida la aplicación.

- **Editor para la creación de cursos:** La interfaz para crear cursos es algo básica y no permite excesiva personalización. Una buena aportación sería que un curso pudiese tener un número dinámico de secciones, de forma que el administrador pueda ir añadiendo nuevos subapartados intercalando fotos, texto y vídeos en el orden que quisiera. Además esto sería interesante de implementar con una interfaz que presente tres elementos genéricos (texto, imagen y vídeo) y que se puedan arrastrar, parecido a como funciona WordPress.
- **Notificaciones de mensajes:** Arties presenta un sistema de chats a tiempo real, no obstante, estaría bien que se notificase también a tiempo real al usuario que reciba el mensaje, para que este no tenga que estar revisando si ha recibido mensajes nuevos o no. Además sería interesante que la lista de chats, tanto para usuarios como para empleados, se ordenase en tiempo real según el orden de llegada de los mensajes.
- **Estadísticas:** Cada usuario puede ver algunas estadísticas sobre cuántas veces ha realizado un curso, cuántas veces ha reservado un material o cuántas veces ha asistido a una clase. Lo ideal sería que hubiese un sistema que permitiese contabilizar más tipos de datos, como el tiempo invertido en hacer un curso, tiempo total asistiendo a una clase, número de meses que lleva siendo socio del gimnasio... Además estaría bien presentar tanto las estadísticas actuales como las nuevas en formatos más visuales como podrían ser gráficas en forma de tarta.
- **Suscripciones mensuales de clases:** Actualmente hay dos opciones para el sistema de clases: o bien que el usuario tenga la suscripción presencial y por tanto pueda ir gratuitamente a todas las clases, o bien que el usuario pague por cada clase individual. Sin embargo, otra opción interesante a implementar es que un usuario sin suscripción al gimnasio, pueda suscribirse mensualmente a un tipo de clase, y así no tenga que comprar una plaza para cada clase individualmente.
- **Aplicación totalmente responsive:** La mayor parte de Arties es *responsive* y por tanto se ve bien en dispositivos más pequeños. Sin embargo, no es totalmente *responsive*. Por ejemplo, el sistema de chat para los usuarios se ha

desactivado para dispositivos que no sean ordenadores, ya que el diseño cambiaría en gran medida. Esto también ocurre en la página de configuración ya que agrupa varias funcionalidades y habría que rediseñarla al completo.

Conclusions and Future Work

Summary: This chapter discusses the conclusions drawn by the team members after the completion of the project. Additionally, it proposes improvements that could be made to the application.

7.3. Conclusions

The project addresses all user needs set forth in section 1.5 Motivation, resulting in the implementation of an innovative application offering real utility for gyms and users.

Its development has been done using current and professionally used technologies as explained in section 3.1. In addition, a widely used methodology and work plan like Scrum has been adopted, accurately simulating the development of a professional web application.

Special attention has been given to the security of the application through the use of Spring Security and permissions, resulting in a secure application suitable for real-world usage.

Moreover, with special attention to interface design and the application of usability principles, the development results in an application with a modern and attractive appearance, featuring high usability.

All the objectives set forth in section 1.6 Objectives for the project's development have been successfully achieved, reflecting the various knowledge acquired during the degree program, such as software development methodologies, programming patterns, the importance of usability, and database design.

Therefore, it is considered that the development of the Arties project has been a success, meeting all the necessary objectives and requirements.

7.4. Future Work

A website like Arties entails a lot of work, and therefore, there are sections that can be improved. It's just a matter of time. Furthermore, everything can always be enhanced, including our website. Although we are satisfied with the work done, we would like to highlight functionalities and details that would improve the application to a greater or lesser extent.

- Course creation editor: Our interface for creating courses is quite basic and does not allow for excessive customization. We believe it would be beneficial for a course to have a dynamic number of sections, allowing the administrator to add new subsections while intermixing photos, text, and videos in the desired order. It would also be interesting to implement an interface that presents three generic elements (text, image, and video) that can be dragged and dropped, similar to how WordPress works.
- Message notifications: Arties features a real-time chat system; however, it would be convenient to notify users in real-time when they receive a new message, so they don't have to keep checking for new messages manually. Additionally, it would be helpful if the chat list, for both users and employees, could be dynamically sorted based on the arrival order of messages.
- Statistics: Each user can see some statistics on how many times they have completed a course, reserved equipment, or attended a class. Ideally, there should be a system that allows for the tracking of more types of data, such as the time invested in completing a course, the total time spent attending classes, or the number of months they have been a gym member. Additionally, it would be nice to present both the current and new statistics in more visual formats, such as pie charts.
- Monthly class subscriptions: Currently, there are two options for the class system: either the user has a physical subscription and can attend all classes for free, or the user pays for each individual class. However, another interesting option to implement is allowing non-subscribing users to have a monthly subscription for a specific class, so they don't have to purchase a spot for each class individually.
- Fully responsive application: Most of Arties is responsive and displays well on smaller devices. However, it is not fully responsive. For example, the user chat system has been disabled for non-computer devices as the design would need significant changes. The same applies to the configuration page, which combines several functionalities and would require a complete redesign.

Capítulo 8

Contribuciones personales

Resumen: En este capítulo se detalla qué ha hecho cada miembro de la pareja, aunque cabe destacar que el reparto del trabajo ha estado muy equilibrado gracias a la metodología seguida.

8.1. Alberto Pascual Fernández

El primer paso al comienzo del proyecto fue investigar la plantilla a utilizar y las tecnologías que utilizaba. Para ello ambos miembros investigamos sobre el uso de tecnologías como Spring, viendo sus capacidades y utilidades. Tras ello, se preparó una primera versión ejecutable del proyecto lista para comenzar a desarrollar.

El siguiente paso fue definir qué datos se necesitaban almacenar para poder implementar las funcionalidades propuestas. Ambos miembros propusimos ideas hasta llegar a una versión inicial de la base de datos con la que poder comenzar. Respecto a esto, yo me ocupe de diseñar los diagramas de entidad relación y de la base de datos, definiendo ya de forma mas concreta la estructura de la base de datos. Además, me encargué de pensar y diseñar las entidades y relaciones necesarias para la implementación del sistema de alquiler y del sistema de reserva de clases.

Una vez conseguida una base sobre la que empezar a desarrollar nos distribuimos el trabajo y comenzamos a desarrollar diferentes partes de la aplicación. La forma de trabajo consistió en que cada uno elegía uno de los sistemas que quería hacer y pasaba a implementarlo, por tanto, a partir de tener la base se maximizó el trabajo en paralelo.

A continuación, pasare a explicar las aportaciones personales concretas que hice al desarrollo del proyecto.

Investigación de Spring Security

En primer lugar, hice una investigación sobre el funcionamiento de Spring Security, teniendo en cuenta sus utilidades y la gestión de los usuarios que define, ya que serían aspectos importantes de cara al registro y *login*, y gestión de permisos. Aunque finalmente no ha sido necesario aplicar muchos cambios extra dado que la plantilla utilizada ya contenía la información que necesitábamos para su uso, esta investigación si fue útil para conocer mejor lo que esta tecnología nos aportaba.

Sistema de material

Fue el primer sistema que desarrollé, el cual, aunque era grande, no era uno de los más complejos, lo que me permitió utilizar su desarrollo para conocer mejor las utilidades de Spring y el desarrollo web en general. Este sistema incluía las funcionalidades de mostrar los objetos del gimnasio y poder añadir, modificar y editar material. También se hizo una primera versión de código para subir imágenes al servidor.

Sistema de alquileres

Utilizando el sistema de material como base, pasé al desarrollo del sistema de alquileres. Este ya se trataba de un sistema más complejo, principalmente por necesitar un uso avanzado de fechas y tener muchos condicionantes.

El primer paso fue pensar y definir como funcionaría de forma genérica de cara al usuario. Se valoraron diferentes opciones y al final se decidió implementar un sistema que permitiera alquilar en cualquier fecha desde la actual, y en el que fuera posible ampliar el alquiler de un material siempre y cuando te encontraras en las 48 horas previas a su final. Esto complicó bastante su desarrollo ya que en función de la fecha actual un alquiler podía encontrase en varios estados diferentes. Podía estar retrasado, que aun no hubiera empezado, que hubiera empezado y estuviera antes de las 48 horas previas al fin del alquiler o que hubiera empezado, pero no estuviera en esas 48 horas. Todas estas situaciones debían comprobarse mediante operaciones sobre fechas y comparaciones de las mismas. Para ello investigué las utilidades de los paquetes *Localdate, LocalTime y LocalDateTime* de Java que permitieron su correcta implementación.

Finalmente, respecto a este sistema hubo que hacer un pequeño trabajo de refactorización ya que se decidió permitir deshacer la finalización de un alquiler y poder obtener estadísticas sobre los alquileres.

Calendario

En relación con el alquiler, el siguiente paso fue desarrollar un calendario para permitir al usuario hacer los alquileres de forma más visual. Se partió de un código inicial que cargaba la información de los días del mes seleccionado, todo en inglés. Por ello, se necesitó añadir código extra que, una vez obtenida la información de forma dinámica, tradujera los días y el nombre del mes.

Además, se preparó para funcionar como un componente independiente que pudiera ser utilizado en diferentes lugares de la aplicación. Para conseguirlo se implementó una especie de patrón *listener* que detectaba cuando el calendario cambiaba de mes, y almacenaba cuales eran las fechas seleccionables. Con este listado de fechas, se podía añadir un código independiente al calendario en otro archivo JavaScript, que asignara un evento al pulsar en cada una de esas fechas. Usando esto, por ejemplo, en el caso de alquiler, se implementó que al pulsar un día se obtenga la cantidad disponible de material para alquilar ese día.

Por último, se preparó para que fuera personalizable. De esta forma se podían decidir qué días no querías permitir seleccionar, cosa que fue de gran utilidad en el sistema de reservas de clases, ya que permitió dejar solo como días seleccionables aquellos en los que ocurría la clase.

Tarjetas

También respecto al diseño de la aplicación, otro de los primeros pasos fue diseñar las tarjetas que hemos utilizado en diferentes apartados de la aplicación (material, cursos y clases). Este era un elemento importante ya que se utilizaría mucho, por tanto, se optó por un diseño que, aunque más complejo de implementar y modificar, resultaba más atractivo y moderno visualmente.

También se le añadió código especial que modificaba su aspecto en pantallas más pequeñas haciéndolo *responsive* y se diseñó alguna variación con solo texto o con dos botones.

Relacionado con esto, se decidió un formato estándar para usar en las imágenes en toda la web. Esto permitió que las imágenes se vieran siempre igual en todos los apartados de la web independientemente del tamaño de la imagen original. Se decidió una relación de aspecto 1:3 que se adaptaba correctamente a las tarjetas.

Sistema de clases

Este sistema fue uno de los que más problemas dio a la hora de implementarlo. El principal problema fue que las clases tienen definido un periodo de tiempo en el que ocurren y que se repiten (ciertos días de las semana y ciertas horas). Sin embargo, un

usuario hace una reserva para una clase en un día concreto. Tras valorar diferentes opciones como por ejemplo permitirle reservar solo semana a semana e ir limpiando la base de datos cada semana, se llegó a una solución mejor gracias al uso de eventos programados.

Se ideó un sistema que permitiera siempre reservar desde la fecha actual hasta un mes, y para mantener esa ventana de tiempo era necesario que cada día se crearan nuevas clases reservables. Para ello investigué el paquete *annotation.Scheduled* de Spring que permitía registrar código bajo ciertas condiciones. Se estableció que cada día a las 00:00 se generarían las clases correspondientes en el día con fecha la actual más un mes, almacenándolas en la base de datos. De la misma manera, al crear una clase nueva se debían añadir a la base de datos todas sus sesiones correspondientes desde la fecha actual hasta un mes de distancia.

Como complicación extra, este sistema requirió trabajar con días de la semana que representaban el periodo en el que ocurrían las clases (*LocaDate*), mezclándolos con fechas concretas en la que un usuario hacia una reserva (*LocalDateTime*). Gracias a tener ya práctica con el manejo de fechas al implementar previamente el sistema de alquiler, se pudo realizar estas operaciones sobre fechas con menos problemas.

Una vez resuelto este problema este sistema no presentó muchas más complicaciones, ya que el resto de uso de fechas era más sencillo que en el sistema de alquileres, y el resto de acciones como modificar, añadir y eliminar eran muy parecidos al resto. La única complicación extra fue el diseño de la interfaz para añadir y modificar sesiones (días de la semana y sus horas) a las que ocurría una clase, ya que necesitaba actualizar la interfaz dinámicamente (modificando una lista y ocultando y mostrando elementos) al mismo tiempo que se iban almacenando los datos para enviar al servidor. Además, para modificar las sesiones de una clase hubo que tener especial cuidado, de forma que se eliminaran solo las reservas de clases de las sesiones modificadas o eliminadas, pero que de aquellas que siguieran igual se mantuvieran.

Inicio

También me encargué de realizar la página de inicio. Si bien no es una página compleja de implementar, pues no requiere de ninguna funcionalidad, sí le dediqué tiempo para intentar conseguir un aspecto moderno y realista. Se hizo uso de alguna utilidad más compleja de CSS para lograr ciertos efectos, y se adaptó para que fuera *responsive*.

Pantallas Responsive

Por último respecto a código, me encargué de adaptar el diseño de algunas páginas para que se adaptaran a pantallas de teléfonos móviles. Algunas ya se han

mencionado, como son la lista de material y de clases por hacer uso de las tarjetas *responsive* que había diseñado, además de la página de inicio. También, hice *responsive* las páginas de alquiler de material y reserva de clases.

Memoria

Respecto a la redacción de la memoria me encargué de documentar aquellas partes que había desarrollado yo, siendo estas todo lo mencionado en el apartado 8.1. Además me encargué de documentar otros apartados como se explicará a continuación:

- **Motivación:** Redacté la motivación del proyecto, destacando aquellas partes que lo hacen útil y especial.
- **Objetivos:** A partir de las ideas que teníamos me encargué de plasmarlas en este apartado, definiendo todo aquello que queríamos conseguir.
- **Tecnologías empleadas:** Me encargué de la redacción de todo este apartado a excepción de STOMP y puesta en marcha. Para poder realizar este apartado tuve que hacer una búsqueda más exhaustiva de todas aquellas tecnologías que habíamos empleado en el desarrollo del proyecto.
- **Arquitectura y modelo de datos:** Realicé la redacción completa de este capítulo. Dado que me había encargado de la definición formal de entidad relación de la aplicación era el más indicado para explicar mejor este capítulo. Necesité hacer cierta investigación para explicar correctamente el patrón MVC y diseñé un diagrama para facilitar su explicación.
- **Gestión de usuarios y permisos:** Dado que este es un sistema que va implícito en muchos otros cualquiera de los dos podíamos realizar su documentación. En este caso me ocupé yo para tener una mejor distribución del trabajo y dado que me manejaba mejor con la creación de los diagramas. También diseñé e incluí un diagrama de casos de uso con todas las funcionalidades y usuarios de la aplicación.
- **Diseño:** Me encargué de la redacción de los apartados de diseño a excepción de estructura de la web. Dado que había diseñado bastantes elementos y pantallas *responsive* y que el nombre elegido fue una de las opciones que había propuesto, era el mas indicado para redactar estos apartados.
- **Usabilidad:** Este apartado fue redactado de forma conjunta por los dos. Yo me ocupé concretamente de los apartados de Visibilidad y estado del sistema, comentarios informativos y diseño estético. También he realizado aportaciones a los apartados de prevención de errores y de recuperación de errores.
- **Conclusiones y trabajo futuro:** En trabajo futuro se han registrado las diferentes propuestas que fuimos viendo a lo largo del desarrollo del proyecto

y en conclusiones se ha definido si se han conseguido los objetivos del proyecto. Yo me encargué más de la parte de conclusiones mientras que Andrés redactó la de trabajo futuro.

- **Vídeo:** Se hizo una reunión conjunta para ver que aspectos de la aplicación mostrar en el vídeo. Tras ello yo me encargué de juntar las partes grabadas en un único vídeo, subiéndolo a Youtube y poniendo las marcas de tiempo interactivas creadas por Andrés. Respecto a esto, también redacté el apartado de Resultado final.

- **Extras:** Para la redacción y diseño de la memoria también he realizado cierto trabajo extra. Por un lado, me he encargado de editar algunas capturas de pantalla para poder mostrar de forma más directa diferentes comparaciones entre pantallas o elementos. Además, me encargué del diseño de algunos diagramas de flujo, como los de ampliar alquiler o el genérico de alquiler, y de los diagramas de caso de uso, incluyendo aquellos sencillos de permisos y el genérico de toda la aplicación.

8.2. Andrés Romero Arbáizar

La primera fase del proyecto fue entender la plantilla escogida, para lo cual hubo que investigar. Los dos miembros del grupo fuimos aportando información mutuamente para acelerar el proceso. Una vez entendida lo mejor posible, lo siguiente fue limpiar todo el código que no se iba a usar y hacer algunos cambios para adaptarla a nuestras necesidades. Después se procedió a pensar la estructura de la base de datos y a modelarla en java. Todo esto se hizo en común, aunque la documentación de la base de datos y el sistema de alquileres y reservas los realizó Alberto exclusivamente.

Una vez teniendo ya la base para empezar a trabajar, dividimos todo el trabajo en tareas que pusimos en el tablero Kanban. A continuación contaré en detalle qué tareas fueron realizadas por mí.

Estructura de la página

Lo primero fue crear una estructura para la página, de tal forma que se pudiese ir avanzando en el proyecto de manera paralela. Para esto se crearon todas las páginas en blanco y se puso acceso a ellas a través de una *navbar* creada manualmente. También se creó el *footer*, aunque este no era necesario para el desarrollo.

No obstante, la página de configuración tiene subapartados, por lo cual se tuvo que desarrollar un sistema de paneles que se ocultan y muestran en función del subapartado elegido en la *subnavbar*, la cual también se tuvo que hacer manualmente. Este sistema se ha utilizado en más partes de la aplicación, como en mostrar los cursos en función de la categoría.

Inicio de sesión y registro

Para la aplicación es primordial diferenciar entre los usuarios, ya que en función del rol tendrán acceso a unas funcionalidades y contenidos que otros no. Por esto le dimos bastante prioridad. Primero se pobló la base de datos con algunos usuarios de prueba con diferentes roles. A continuación se desarrolló tanto la lógica del *back-end* como la del *front-end* tanto de inicio de sesión como de registro, a la vez que se construía el apartado visual. No obstante, el apartado visual es algo que ha ido cambiando a lo largo del proyecto hasta llegar al actual.

Gestión de empleados

Una vez implementados los sistemas de *login* y registro, se continuó en la misma línea con la gestión de empleados. Primero se creó la parte visual de forma que se viese la lista de empleados con botones al lado de cada uno para gestionarlos, y un botón general para abrir un modal de crear empleados. La parte lógica y del *back-end* se desarrolló lo último.

Chat

Teniendo ya el sistema de usuarios, se procedió a realizar el chat, ya que el funcionamiento de este depende del rol del usuario. Para implementarlo primero se tuvo que investigar cómo se usaban los *WebSockets*, después se procedió a crear una estructura básica en el *front-end* para poder ir haciendo pruebas y desarrollando el chat poco a poco en base a cada una de las funcionalidades que tiene. A continuación se desarrolló toda la lógica y finalmente se le dio un aspecto visual moderno y cuidado, aunque este se fue mejorando a lo largo del proyecto.

Suscripciones

Para ir probando las diferentes funcionalidades de la aplicación, se tenía que hacer escribiendo los roles del usuario manualmente, así que para evitarlo lo siguiente que se desarrolló fue el sistema de suscripciones. Primero se desarrolló la parte visual para el cliente, después toda la lógica necesaria para que el usuario pueda cambiar de suscripción correctamente, tras confirmar la compra en la pantalla de pago, la cual no es una pasarela de pago real. Después se procedió a desarrollar toda la parte de gestión de suscripciones que lleva a cabo el administrador. Esta se complicó en cierta medida ya que para que el cliente siempre vea las tarjetas perfectas, hubo que hacerlas dinámicas en función de la información que quisiese mostrar el administrador.

Perfil

Llegando ya a la parte final en cuanto al desarrollo de la aplicación, se procedió a hacer un perfil, siguiendo una estética totalmente moderna, minimalista y que sigue los principios del *Google Material Design*. El perfil presenta tres bloques, la información de la cuenta del usuario, una descripción del cliente para conocerle mejor y que pueda ser mejor aconsejado y finalmente una sección de estadísticas. Los dos primeros bloques se desarrollaron lo primero, comenzando con la parte visual y finalmente con la lógica para cambiar los datos del usuario como contraseña, correo, dirección... Se tuvo especial cuidado en ayudar al usuario a recuperarse de los errores y a que el proceso fuese lo más eficiente posible. En último lugar está el bloque de estadísticas, que se fue desarrollando según se fueron acabando las funcionalidades de la aplicación que permitían extraer dichos los datos.

Cursos

Aunque es una de las funcionalidades más importantes, se dejó para el final permitiendo así dedicarla todo el tiempo que se pudiese. Finalmente se creó una versión de complejidad media, que implementa una estética cuidada y agradable tanto para los empleados como para los clientes. Lo primero fue crear tanto en la pantalla de "Cursos" como en el apartado de "Cursos" dentro de "Configuración" una vista donde se mostrase una lista con todos los cursos y que se pudiesen filtrar según categorías. Después se realizó la parte de lógica de añadir, modificar y eliminar cursos, se siguió con la parte visual para añadir y modificar un curso, ya que difieren ligeramente. Finalmente se hizo la página de ver un curso para que los clientes puedan visualizar toda la información del mismo.

Memoria

Todo lo mencionado en el apartado 8.2 ha sido escrito y explicado en la memoria por mí, por lo que no hace falta repetirlas en este apartado. No obstante, también he escrito otras secciones de la memoria que no tienen tanto que ver con el código. Estas son las siguientes:

- **Resumen y palabras claves:** Realización del resumen y de las palabras claves tanto en español como en inglés.
- **Plan de trabajo y Punto de partida:** En el primer punto se explica la metodología seguida y el trabajo realizado en cada hito. En el segundo se comenta de dónde se empezó a construir nuestro proyecto. Además, se cuentan las líneas de código para ver la diferencia, poniendo así en contexto cuánto ha cambiado respecto a la plantilla original.

- **Puesta en marcha:** Desarrollo de un tutorial para la configuración del sistema necesaria para desplegar Arties en Windows, ya que al integrar diversas tecnologías no es algo trivial.
- **Estado de la cuestión:** Se llevó a cabo un análisis de la competencia enfocado a cada una de las funcionalidades que ofrece Arties, pues no existe una aplicación que integre todo al igual que la nuestra. Esto último se explica también en el apartado 2.2, donde se comenta en qué nos diferenciamos.
- **Conclusiones y trabajo futuro:** Redacción de todas las ideas que han ido surgiendo a lo largo del proyecto que podrían mejorar la aplicación. También algunas aportaciones a conclusiones.
- **Diseño y usabilidad:** Estos puntos también se realizaron conjuntamente, aunque cada uno se encargó de redactar en solitario ciertos apartados. En mi caso fueron los siguientes: introducción, estructura de la web, principio de cierre, estado de la navegación, consistencia y reconocer antes que recordar.
- **Vídeo:** Grabación de los vídeos que muestran el funcionamiento de la aplicación. También creación de la lista de tiempos para encontrar rápidamente en el vídeo alguna funcionalidad en específico.
- **Ajustes memoria:** Algunos detalles como ajustar las portadas, los interlineados, quitar, añadir y mover capítulos, configurar las constantes del proyecto...

Bibliografía

Bootstrap. Página oficial bootstrap. s.f. Disponible en https://getbootstrap.com/ (último acceso, Mayo, 2023).

Engine, H. D. Página oficial de h2. s.f. Disponible en http://www.h2database.com/html/main.html (último acceso, Mayo, 2023).

Freire, M. Ajax, stomp y websockets. 2022a. Disponible en https://github.com/manuel-freire/iw/blob/main/doc/11-ajax-y-websockets.md (último acceso, Mayo, 2023).

Freire, M. Guía de instalación de la plantilla. 2022b. Disponible en https://github.com/manuel-freire/iw/tree/main/qna (último acceso, Febrero, 2023).

Freire, M. Plantilla para proyectos de ingeniería web. 2022c. Disponible en https://github.com/manuel-freire/iw/tree/main/plantilla (último acceso, Febrero, 2023).

GitHub. Página de información de tableros kanban en github. s.f. Disponible en https://docs.github.com/es/issues/organizing-your-work-with-project-boards/managing-project-boards/about-project-boards (último acceso, Mayo, 2023).

GitHub. Página oficial de github. s.f. Disponible en https://github.com/ (último acceso, Mayo, 2023).

Heller, E. Psicología del color. 2008.

Hibernate. Página de documentación de jpa utilizado con hibernate. s.f. Disponible en https://docs.jboss.org/hibernate/orm/current/userguide/html_single/Hibernate_User_Guide.html (último acceso, Mayo, 2023).

Microsoft. Página oficial de visual studio code. s.f. Disponible en https://code.visualstudio.com/ (último acceso, Mayo, 2023).

Mozilla-Developer-Network. Página de documentación del uso de fetch. 2023. Disponible en https://developer.mozilla.org/es/docs/Web/API/Fetch_API/Using_Fetch (último acceso, Mayo, 2023).

Nielsen, J. Los 10 principios de nielsen. 1994. Disponible en https://www.nngroup.com/articles/ten-usability-heuristics/ (último acceso, Mayo, 2023).

Oracle. Página oficial java. s.f. Disponible en https://www.oracle.com/java/ (último acceso, Mayo, 2023).

Security, S. Página oficial spring security. s.f. Disponible en https://spring.io/projects/spring-security (último acceso, Mayo, 2023).

Shneiderman, B. Las 8 reglas de oro de shneiderman. 2016. Disponible en https://www.cs.umd.edu/users/ben/goldenrules.html (último acceso, Mayo, 2023).

Spring. Página de documentación del paquete scheduling.annotation. s.f.a. Disponible en https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/scheduling/annotation/Scheduled.html (último acceso, Mayo, 2023).

Spring. Página oficial spring. s.f.b. Disponible en https://spring.io/ (último acceso, Mayo, 2023).

SpringSecurity. Página de documentación del paquete bcrypt. s.f. Disponible en https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/crypto/bcrypt/BCrypt.html (último acceso, Mayo, 2023).

Thymeleaf. Página oficial thymeleaf. 2022. Disponible en https://www.thymeleaf.org/ (último acceso, Mayo, 2023).

W3Schools. Página de documentación de ajax. s.f. Disponible en https://www.w3schools.com/js/js_ajax_intro.asp (último acceso, Mayo, 2023).