DESARROLLO DE UNA APLICACIÓN PARA LA GESTIÓN DE BÚSQUEDAS DE CIENTÍFICOS Y ORGANISMOS

TRABAJO FIN DE GRADO
CURSO 2022-2023

ALBERTO HERRERA GARCÍA
GERALD LIMA MENDIA

DIRECTORES:
ANTONIO SARASA CABEZUELO
COVADONGA DÍEZ SANMARTÍN

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

Septiembre 2023

DEVELOPMENT OF AN APPLICATION FOR THE SEARCH MANAGEMENT OF SCIENTISTS AND ORGANIZATIONS

FINAL DEGREE PROJECT REPORT
COMPUTER ENGINEERING DEGREE

AUTHORS:
ALBERTO HERRERA GARCÍA
GERALD LIMA MENDIA

DIRECTED BY
ANTONIO SARASA CABEZUELO
COVADONGA DÍEZ SANMARTÍN

COMPUTER ENGINEERING DEGREE
FACULTY OF COMPUTER SCIENCE
COMPLUTENSE UNIVERSITY OF MADRID

SEPTEMBER 2023

A todos nuestros seres queridos,
que nos animan a seguir luchando
por nuestros sueños.

AGRADECIMIENTOS

En primer lugar, nos gustaría dar las gracias a nuestros directores de proyecto, Antonio Sarasa y Covadonga Díez, quienes nos dieron la oportunidad de realizar este trabajo.

Así mismo, queremos agradecer a nuestras madres Antonia y Angélica, y a nuestros padres Antonio y Germán todo el apoyo incondicional que nos lleváis dando toda la vida.

A nuestros hermanos Sergio y Jhoel y a nuestras hermanas Carolina y Jessica, que se han interesado y preocupado por nosotros en todo momento.

A nuestras abuelas Rufina y Marta que son nuestros tesoros y siempre quieren lo mejor para nosotros.

Y por último a los abuelos Ginés, Antonio y Gabina quienes, a pesar de no poder estar en este momento de nuestra vida personal y estudiantil, nos dedicaron todo su tiempo, esfuerzo y cariño a convertirnos en las personas que somos y seremos en el futuro.

Resumen

Este proyecto surge de la necesidad de crear un espacio donde los científicos de todo el mundo puedan encontrar proyectos apasionantes en los que poder participar y, al mismo tiempo, donde los organismos puedan publicar sus proyectos y encontrar científicos cualificados para poder hacerlos realidad.

Actualmente, existen una gran cantidad de plataformas donde poder encontrar empleo de cualquier tipo y para cualquier profesión. Sin embargo, en este trabajo de fin de grado se ha desarrollado un sistema con el objetivo de centralizar a todos los científicos y organismos científicos en un entorno donde poder encontrarse y colaborar de forma más accesible. Para ello, cada científico puede realizar una publicación y cada organismo puede publicar sus proyectos. De esta manera, el científico puede recibir recomendaciones de proyectos en los que poder participar y el organismo puede recibir recomendaciones de científicos que pueden incorporarse a sus proyectos.

Palabras clave

Ciencia, científico, organismo, proyecto científico, investigación y aplicación web.

Abstract

This project arises from the need to create a space where scientists from all over the world can find exciting projects in which they can participate and, at the same time, where organizations can publish their projects and find qualified scientists to make them a reality.

Currently, there are a large number of platforms where you can find employment of any kind and for any profession. However, in this final degree project, a system has been developed with the aim of centralizing all scientists and scientific organizations in an environment where they can meet and collaborate in a more accessible way. To do this, each scientist can make a publication and each organism can publish its projects. In this way, the scientist can receive recommendations of projects in which he can participate and the organism can receive recommendations from scientists who can join his projects.

Keywords

Science, scientist, organism, scientific project, research and web application.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1 - Introducción

En este capítulo uno de la memoria, se procede a ofrecer, en primera instancia, una explicación acerca de la motivación de desarrollar la aplicación y cuáles han sido sus objetivos. Posteriormente, se proporciona una descripción tanto de la estructura por capítulos de la memoria como de la metodología realizada para elaborar la aplicación y su consecuente organización del trabajo.

1.1 Motivación

En el entorno científico actual, la identificación y selección de proyectos que se ajusten a las necesidades y metas de los científicos de manera precisa y la búsqueda eficiente de profesionales científicos por parte de los organismos públicos y privados representan un problema para ambas entidades. Este problema se acentúa aún más en la actualidad debido a que las áreas de investigación cada vez son más complejas y los perfiles profesionales en el ámbito científico son cada vez más diversos. Estas dos situaciones realmente dificultan la asociación entre los científicos y los proyectos que ofrecen los organismos.

Una manera efectiva de abordar este problema podría ser mediante la consecución de una plataforma que sirva como punto de encuentro entre los científicos y los organismos.

Para tratar de resolver el problema, se plantea el desarrollo de una plataforma web que actúe como un espacio colaborativo entre científicos y organismos. La plataforma debe integrar una base de datos suficientemente amplia como para poder almacenar publicaciones de científicos y proyectos de los organismos. En ella, los científicos serán capaces de buscar proyectos que se alineen con sus intereses y habilidades y, al mismo tiempo, los organismos podrán publicar sus proyectos detalladamente, especificando sus objetivos y requisitos. La creación de esta plataforma web fomentaría la colaboración interdisciplinaria y contribuiría al avance de la ciencia y las investigaciones científicas.

1.2 Objetivos

El objetivo principal de este proyecto era desarrollar una plataforma que permitiera la gestión de búsquedas de científicos y proyectos.
A continuación se enumeran, de manera más precisa, los objetivos específicos planteados en el proyecto:

1.- Desarrollar una aplicación web que permita a los científicos realizar publicaciones para poder encontrar el proyecto que cubra sus necesidades y que, al mismo tiempo, permita a los organismos localizar efectivos para llevar a cabo sus proyectos.

2.- Desarrollar una API REST que reciba peticiones desde la aplicación web y que incluya toda la lógica de negocio.

3.- Facilitar al administrador de la aplicación el control total sobre los usuarios, publicaciones de los científicos y proyectos de los organismos.

1.3 Estructura de la memoria

Esta memoria se encuentra estructurada por capítulos según cada una de los apartados que conforman la aplicación:

Capítulo 1: Introducción

En este capítulo se realiza una introducción del trabajo efectuado, así como su motivación y sus objetivos. También se realiza una descripción de la estructura de la memoria y se trata la metodología empleada para el desarrollo del proyecto, la planificación del proyecto y la organización del trabajo.

Capítulo 2: Estado del arte

En este capítulo se realiza una revisión y una breve explicación de algunos programas y aplicaciones ya existentes que tengan una funcionalidad similar a la aplicación desarrollada en

este trabajo.

Capítulo 3: Tecnología empleada

En este capítulo se aborda una a una todas las tecnologías que han sido necesarias y que se han empleado para la implementación de la aplicación.

Capítulo 4: Especificación de requisitos

En este capítulo se trata la especificación de requisitos de la aplicación. Este capítulo se centra en los tres actores principales de la aplicación: científico, organismo y administrador, así como en sus casos de uso divididos en módulos funcionales.

Capítulo 5: Arquitectura y modelo de datos

Este capítulo trata sobre la arquitectura empleada para formar la aplicación y los patrones de diseño utilizados para ello. Además, se detalla el proceso mediante el cual se obtiene la persistencia de la información que maneja la aplicación. Se expone el diagrama entidad-relación y se explica su implementación en la base de datos relacional.

Capítulo 6: Diseño y funcionalidades

Este capítulo describe el diseño y las funcionalidades de la aplicación separando la aplicación web del servicio API REST.

Capítulo 7: Conclusiones y trabajo futuro

En este capítulo se exponen las conclusiones de la realización del trabajo y se plantea la dirección para el desarrollo del trabajo futuro.

Capítulo 8: Contribuciones al proyecto

En este capítulo se desarrollan, individualmente, las contribuciones al proyecto por parte de cada uno de los miembros del trabajo.

Capítulo 9: Bibliografía

Este capítulo incluye todos los recursos, enlaces y referencias que han sido necesarias utilizar para la consecución del proyecto.

Capítulo 10: Apéndices

En este último capítulo se proporciona la guía de uso de la aplicación web.

1.4 Metodología

El desarrollo del trabajo se ha realizado según la planificación del proyecto y organización de trabajo que se expone en el siguiente apartado.

1.4.1 Planificación del proyecto

La planificación del proyecto viene dada por un total de cuatro fases.

1. Investigación y captura de requisitos

En esta fase inicial, se ha llevado a cabo una investigación acerca del problema a resolver. Esto implicaba la identificación de características, funcionalidades y restricciones que tenían impacto en el diseño y desarrollo del proyecto. Toda esta información fue recogida en la especificación de requisitos.

2. Desarrollo de la aplicación

En esta fase, el equipo de trabajo desarrolló el código fuente que conforma la aplicación siguiendo las especificaciones y restricciones investigadas en la fase anterior.

El desarrollo del proyecto fue dividido en varios hitos:

- Hito Inicial: Creación del proyecto (front-end y backend), generación de la base de datos y sus entidades, y desarrollo las funcionalidades de: Registro de usuario e Inicio de sesión.
- Primer Hito: Desarrollo de las siguientes funcionalidades de científico: Crear, editar, eliminar, buscar y listar publicaciones.
- Segundo Hito: Desarrollo de las siguientes funcionalidades de organismo: Publicar, editar y eliminar proyecto, buscar científico y recomendaciones de científicos.

- Tercer Hito: Desarrollo de las funcionalidades de científico: Buscar organismo y recomendaciones de proyectos.
- Cuarto Hito: Desarrollo de las funcionalidades de administrador: Listar, editar, eliminar y reactivar publicaciones, usuarios y proyectos.

3. Pruebas

   Esta fase se ha realizado de manera conjunta con el desarrollo del proyecto.
   En ella, los miembros del equipo de desarrollo han adoptado un enfoque proactivo garantizando que cada funcionalidad nueva o corregida funcionara según lo establecido antes de avanzar a la desarrollar la siguiente. Esta estrategia contribuyó a la estabilidad general del software que se estaba desarrollando.

4. Desarrollo de la memoria

   Esta última fase se ha llevado a cabo una vez finalizado el desarrollo de la aplicación y ha implicado la documentación completa de todo el desarrollo del proyecto.

1.4.2 Organización del trabajo

Para la realización del trabajo se ha aplicado una metodología de trabajo iterativa incremental. Cada iteración ha tenido una duración aproximada de dos semanas. Cada dos semanas, se ha realizado una reunión virtual a través de videollamada en la que se han validado y probado las funcionalidades que han conseguido desarrollarse y se han fijado nuevos objetivos para la semana siguiente.

Como el tamaño del equipo de trabajo ha sido únicamente de dos personas, ambos miembros del equipo han estado en contacto directo diariamente a lo largo de las semanas vía mensajería instantánea. De esta manera, la comunicación entre los dos ha sido muy fluida y ha permitido obtener y proporcionar feedback de manera continua. Gracias a esto, ha sido posible tratar en cuestión de horas cualquier tipo de inconveniente o duda que ha surgido durante el desarrollo.

Chapter 1 - Introduction

In this chapter one of the report, we proceed to offer, in the first instance, an explanation about the motivation to develop the application and what its objectives have been. Subsequently, a description is provided both for the structure by chapters of the report and for the methodology used to develop the application and its consequent organization of work.

1.1 Motivation

In the current scientific environment, the identification and selection of projects that fit the needs and goals of scientists in a precise manner and the efficient search for scientific professionals by public and private organizations represent a problem for both entities. This problem is accentuated even more today because the areas of research are increasingly complex and professional profiles in the scientific field are increasingly diverse. These two situations really make it difficult for scientists to partner with the projects offered by organisms.

An effective way to address this problem could be by achieving a platform that serves as a meeting point between scientists and organisms.

To try to solve the problem, the development of a web platform that acts as a collaborative space between scientists and organisms is proposed. The platform should integrate a database large enough to be able to store publications of scientists and projects of the agencies. In it, scientists will be able to search for projects that align with their interests and skills and, at the same time, agencies will be able to publish their projects in detail, specifying their objectives and requirements. The creation of this web platform would foster interdisciplinary collaboration and contribute to the advancement of science and scientific research.

1.2 Goals

The main objective of this project was to develop a platform that would allow the management of searches of scientists and projects.

The specific objectives set out in the project are listed more precisely below:

1.- Develop a web application that allows scientists to make publications in order to find the project that meets their needs and, at the same time, allows agencies to locate personnel to carry out their projects.

2.- Develop a REST API that receives requests from the web application and includes all the business logic.

3.- Provide the administrator of the application with total control over users, publications of scientists and projects of organizations.

1.3 Structure of the report

This report is structured by chapters according to each of the sections that make up the application:

Chapter 1: Introduction

This chapter introduces the work done, as well as its motivation and objectives. It also describes the structure of the report and discusses the methodology used for the development of the project, the planning of the project and the organization of work.

Chapter 2: State of the Art

This chapter reviews and briefly explains some existing programs and applications that have similar functionality to the application developed in this work.

Chapter 3: Technology Used

This chapter addresses one by one all the technologies that have been necessary and that have been used for the implementation of the application.

Chapter 4: Requirements Specification

This chapter discusses the application requirements specification. This chapter focuses on the three main actors of the application: scientific, organization and administrator, as well as their use cases divided into functional modules.

Chapter 5: Architecture and Data Model

This chapter discusses the architecture used to form the application and the design patterns used to form it. In addition, it details the process by which the persistence of the information handled by the application is obtained. The entity-relationship diagram is exposed and its implementation in the relational database is explained.

Chapter 6: Design and Features

This chapter describes the design and capabilities of the application by separating the web application from the REST API service.

Chapter 7: Conclusions and future work

This chapter presents the conclusions of the realization of the work and sets out the direction for the development of future work.

Chapter 8: Contributions to the project

In this chapter, the contributions to the project by each of the members of the work are developed.

Chapter 9: Bibliography

This chapter includes all the resources, links and references that have been necessary to use for the achievement of the project.

Chapter 10: Appendices

This last chapter provides the web application usage guide.

1.4 Methodology

The development of the work has been carried out according to the planning of the project and the work plan that is exposed in the following section.

1.4.1 Project planning

The project planning has been given by a total of four phases.

1. Requirements investigation and capture

In this initial phase, an investigation has been carried out about the problem to be solved. This involved the identification of features, functionalities and constraints that had an impact on the design and development of the project. All this information was collected in the requirements specification.

2. Application development

In this phase, the team developed the source code that makes up the application following the specifications and restrictions investigated in the previous phase.

The development of the project was divided into several milestones:

- Initial Milestone: Creation of the project (front-end and backend), generation of the database and its entities, and development of the functionalities of: User registration and Login.
- First Milestone: Development of the following scientist functionalities: Create, edit, delete, search and list publications.
- Second Milestone: Development of the following organism functionalities: Publish, edit and delete project, search for scientists and recommendations from scientists.
- Third Milestone: Development of scientist functionalities: Search for organism and project recommendations.
- Fourth Milestone: Development of administrator functionalities: List, edit, delete and reactivate publications, users and projects.

3. Tests

This phase has been carried out jointly with the development of the project.

In it, members of the development team have taken a proactive approach ensuring that each new or patched feature works as established before moving on to developing the next. This strategy contributed to the overall stability of the software being developed.

4. Report development

This last phase has been carried out once the development of the application has been completed and has involved the complete documentation of the entire development of the project.

1.4.2 Work plan

To carry out the work, an incremental iterative work methodology has been applied. Each iteration has lasted approximately two weeks. Every two weeks, a virtual meeting has been held through video call in which the functionalities that have been developed have been validated and tested and new objectives have been set for the following week.

As the size of the work team has been only two people, both team members have been in direct contact daily throughout the weeks via instant messaging. In this way, the communication between the two has been very fluid and has allowed them to obtain and provide feedback on a continuous basis. Thanks to this, it has been possible to deal in a matter of hours with any type of inconvenience or doubt that has arisen during development.

Capítulo 2 - Estado del arte

En este capítulo, se destacan las características más significativas sobre los programas y aplicaciones similares que se pueden encontrar en la actualidad:

2.1 ResearchGate

Es una plataforma que reúne a investigadores y científicos de diversas disciplinas en una red social centrada en la academia [1]. Entre sus funcionalidades destacan:

- Posibilidad de compartir y acceder a publicaciones científicas y documentos de investigación.
- Sección de preguntas y respuestas para resolver dudas y discutir conceptos.
- Facilita la búsqueda y establecimiento de colaboraciones entre investigadores.
- Promueve el acceso abierto al conocimiento científico en muchas publicaciones.
- Anuncia eventos académicos relevantes como seminarios y conferencias.
- Opción de unirse a grupos temáticos para discutir y colaborar en proyectos.

2.2 Academia.edu

Es una plataforma en línea que sirve como red social y repositorio académico para científicos y estudiantes [2]. Sus funcionalidades más importantes son:

- Posibilidad de rastrear métricas como vistas, descargas y citas para evaluar el impacto de sus investigaciones y su visibilidad en la comunidad.
- Posibilidad de seguir a otros investigadores para estar al tanto de sus nuevas actividades y contribuciones.
- Ofrece recomendaciones de investigaciones y científicos relevantes según los intereses y la actividad del usuario.
- Posibilidad de que los organismos tengan perfiles en la plataforma. Pueden mostrar sus actividades y sus investigaciones.

2.3 Mendeley

Es una plataforma de gestión de referencias y red social académica que ayuda a los científicos a administrar su investigación de manera eficiente y a conectar con otros académicos en línea [3]. Entre sus funcionalidades más reseñables destacan:

- Brindar la posibilidad de que los científicos puedan organizar y compartir referencias bibliográficas.
- Posibilidad de colaborar en grupos de investigación.
- Posibilidad de generar citas y bibliografías en varios estilos de formato.
- Importación automática de metadatos de bases de datos académicas.
- Posee una versión de pago (*Mendeley Institutional Edition*) que ofrece servicios a nivel institucional para colaborar y acceder a recursos.

2.4 SciProfiles

Es una red social desarrollada por el MDPI (*Multidisciplinary Digital Publishing Institute*, Suiza) [4]. Su objetivo es facilitar el acceso inmediato a resultados de investigaciones y proporcionar oportunidades académicas a los científicos.
Sus funcionalidades más importantes son:

- Proporcionar una visión general completa de todas las contribuciones académicas.
- Mostrar, explicar y compartir las publicaciones científicas con la comunidad.
- Monitorizar estadísticas en tiempo real.
- Mantener informado al científico sobre varios desarrollos en su campo de investigación.
- Poder seguir las actividades de otros científicos.
- Ofrecer la posibilidad de calificar, comentar y recomendar publicaciones.

Capítulo 3 - Tecnología empleada

En este capítulo, se proporciona una descripción acerca de todas las tecnologías que han sido utilizadas durante el desarrollo del trabajo, ordenadas por orden alfabético.

3.1 Angular

Es un framework de desarrollo de aplicaciones web creado por Google [5]. Se basa en el lenguaje de programación TypeScript y ofrece un enfoque modular estructurado para desarrollar aplicaciones web dinámicas de una sóla página (*Single-page application*). Angular ofrece un desarrollo basado en componentes, implementa el patrón Modelo-Vista-Modelo de vista (*Model-View-ViewModel*) y proporciona herramientas para gestionar el estado de la aplicación, enrutamiento, comunicación con servidores y manipulación del DOM (*Document Object Model*).

Angular proporciona también su propio sistema de gestión de peticiones HTTP a través del módulo *HttpClient* que forma parte de *@angular/common/http*. Este módulo permite realizar peticiones HTTP de una forma eficiente a servidores remotos [6]. Además, permite manejar respuestas, gestionar errores y realizar transformaciones de datos.

Otra herramienta que utiliza Angular es Angular CLI. Es una interfaz de línea de comandos (*Command Line Interface*) basada en Node.js que simplifica y agiliza el desarrollo de aplicaciones Angular [7]. Facilita la creación de componentes, módulos y servicios así como la compilación y la ejecución de pruebas.

Para el desarrollo del trabajo se ha utilizado la versión 16.1.8 tanto de Angular como de Angular CLI.

3.2 Bootstrap

Es un framework utilizado en diseño Front-End. Bootstrap proporciona un conjunto de clases y componentes con estilos prediseñados y utilidades que facilitan la creación de interfaces de usuario *responsive* y atractivas [8]. Su principal utilidad es la de estandarizar y agilizar la creación de interfaces web.

La versión de Bootstrap utilizada para hacer las interfaces de la aplicación web del trabajo ha sido la versión 5.3.1.

3.3 CSS

CSS (*Cascading Style Sheets*) es un lenguaje de diseño utilizado para gestionar la presentación y aspecto visual de los documentos HTML y XML de una página web [9] . CSS permite separar la estructura del contenido de la página de su diseño y apariencia. Para ello, utiliza una serie de selectores y reglas que definen cómo se deben visualizar los elementos HTML en cuanto a posicionamiento, márgenes, colores, fuente, etc.

La versión de CSS utilizada en el trabajo ha sido CSS3.

3.4 DBeaver

Es una herramienta de gestión de bases de datos. Permite trabajar con una gran variedad de sistemas de gestión de bases de datos (*SGBD* o *DBMS*, *Database Management System*) ofreciendo una interfaz gráfica e incluyendo soporte multiplataforma [10]. En él, los desarrolladores pueden gestionar bases de datos, realizar consultas SQL, tratar datos, diseñar esquemas, etc.

El sistema de gestión de bases de datos utilizado en el trabajo ha sido PostgreSQL.

3.5 Firebase

Es una plataforma de desarrollo de aplicaciones web y móviles en la nube. Es ofrecida por Google, está integrada y respaldada por Google Cloud Platform (*GCP*) y proporciona una gran variedad de servicios que pueden ser de utilidad durante el desarrollo de una aplicación móvil o de una aplicación web [11]. Algunos ejemplos de estos servicios son: *Authentication* (autenticación de usuarios) [12], *Google Analytics* (análisis de datos) [13] y *Firebase Realtime Database* (bases de datos en tiempo real) [14].

En el caso de la aplicación web, se ha utilizado el servicio *Authentication* en el registro de usuarios e inicio de sesión para gestionar la autenticación y autorización de los usuarios.

3.6 Git

Es un sistema software de control de versiones distribuido (*DVCS, Distributed Control Version System*) que facilita el trabajo en paralelo entre miembros del equipo de desarrollo [15]. Ofrece a cada miembro la posibilidad de trabajar en el código de forma local e independiente a través de una copia del repositorio y, posteriormente, fusionar los cambios realizados en una versión principal ubicada en el repositorio online compartido. Esta estructura descentralizada compuesta por ramas de desarrollo facilita la gestión de versiones de la aplicación y promueve la colaboración entre los miembros del equipo.

La versión de Git que se ha utilizado durante el trabajo ha sido la versión 2.39.1.windows.1.

3.7 GitHub

Es una plataforma en línea que ofrece un lugar donde poder almacenar, gestionar y colaborar en proyectos software [16]. Estos proyectos se sitúan en repositorios que son accesibles por los desarrolladores tanto de manera pública como privada. Está basada en Git y ofrece funcionalidades como integración contínua, seguimiento de problemas, despliegue automático y control de versiones.

3.8 HTML

HTML (*HyperText Markup Language*) es un lenguaje de marcado utilizado para describir el contenido de una página web [17]. A través de sus elementos y etiquetas, permite la inclusión, entre otros, de texto, imágenes, vídeos, enlaces, etc. El código HTML es interpretado por los navegadores web para formar la base de las páginas web. Aunque puede incluir código acerca de los estilos de los elementos, generalmente se combina con una hoja de estilos CSS para definir la apariencia de una página web.

La última versión lanzada y convertida en estándar es HTML5 [18], que es la misma versión que se ha utilizado en la implementación del Front-End de la aplicación web.

3.9 Java Spring Boot

Es un framework de desarrollo utilizado principalmente en el lado Backend de la aplicación. Su función es la de simplificar la creación de servicios web, microservicios, API REST que gestionan la lógica de negocio y permiten la interacción tanto con la base de datos como con el lado Front-End [19].

En la aplicación, se ha utilizado la versión 11 de Java [20] y la versión 2.3.2.RELEASE de Spring Boot.

3.10 Node.js

Es el entorno de ejecución de JavaScript [21] en el lado del servidor. En el contexto de Angular, Node.js [22] se utiliza para desplegar y configurar aplicaciones Angular en servidores, para ejecutar scripts de creación y desarrollo y para gestionar dependencias mediante NPM (*Node Package Manager*). Todas estas funciones hacen que Node.js sea esencial en el conjunto de tecnologías que conforman el desarrollo de aplicaciones Angular.

En el trabajo, se ha utilizado la versión 18.17.0 de Node.js.

3.11 NPM

NPM o *Node Package Manager* [23] es el gestor oficial de paquetes de Node.js. Se encarga de distribuir y gestionar paquetes JavaScript de software. Estos paquetes son una especie de bibliotecas predefinidas de código que se utilizan para ahorrar esfuerzo y tiempo durante el desarrollo. Gracias a NPM, se pueden instalar, administrar y actualizar estas bibliotecas de forma sencilla.

Se ha utilizado en el trabajo la versión 9.8.1 de NPM.

3.12 PostgreSQL

Es un sistema de gestión de bases de datos relacional de código abierto [24]. Su comunidad realiza contribuciones activamente para desarrollar y mejorar la plataforma. Se trata de un sistema capaz de gestionar grandes volúmenes de datos y de realizar consultas SQL complejas. Además, ofrece soporte nativo para datos en formato JSON [25] y para las transacciones ACID (*Atomicidad, Consistencia, Aislamiento y Durabilidad*) [26]. Por estos motivos, PostgreSQL es una gran elección en proyectos que requieran un manejo sólido de la información.

Se ha utilizado la versión 15.2 de PostgreSQL durante el desarrollo de la aplicación.

3.13 Postman

Es una herramienta utilizada por los desarrolladores de aplicaciones para documentar y realizar pruebas en APIs (*Application Programming Interface*) [27]. Dispone de una interfaz gráfica que permite probar, depurar servicios y validar sus respuestas mediante la creación, el envío y la recepción de peticiones HTTP a través de APIs. También ofrece características adicionales como la posibilidad de organizar las peticiones en colecciones y de automatizar flujos de trabajo con scripts.

La versión de Postman [28] utilizada ha sido la versión 10.17.3.

3.14 Visual Studio Code

Es un editor de código fuente desarrollado por Microsoft. Dispone de una amplia gama de extensiones que permiten adaptar el editor a las necesidades del proyecto y del lenguaje de programación. Además, permite la integración con sistemas de control de versiones como Git e incluye la posibilidad de depurar el código directamente desde el editor.

La versión de Visual Studio Code [29] utilizada para realizar el código del trabajo fue la versión 1.81.1.

Capítulo 4 - Especificación de requisitos

En este capítulo se realiza una especificación, en primer lugar, de los actores que intervienen en la aplicación de este trabajo de fin de grado y, en segundo lugar, de cada uno de los módulos funcionales que la componen.

4.1 Actores

Los tres actores que intervienen en la aplicación son:

1. **Científico**: Este actor representa al usuario que utiliza la aplicación con el objetivo de realizar publicaciones para poder incorporarse y participar en proyectos científicos.

2. **Organismo**: Este actor se refiere a las organizaciones e instituciones que utilizan la aplicación para publicar proyectos que quieran realizar con el objetivo de encontrar científicos que quieran participar en ellos.

3. **Administrador**: Este actor es el encargado de gestionar y supervisar la aplicación en su conjunto. Tiene acceso a todo el listado de usuarios de la aplicación, al listado de publicaciones de científicos y al listado de proyectos de organismos. Puede editar, eliminar y reactivar cada elemento de los tres listados.

4.2 Módulos funcionales

Los casos de uso se han agrupado en cuatro módulos según las funcionalidades exclusivas que pueden realizar cada uno de los usuarios, más un módulo adicional llamado Cuenta para recoger todas las operaciones comunes para el científico y el organismo.

- Módulo Científico
- Módulo Organismo
- Módulo Administrador
- Módulo Cuenta

4.2.1 Módulo Científico

El módulo científico envuelve todas las funcionalidades que puede realizar el científico.

Como se observa en la Figura 4.1, estas incluyen: Crear publicación, Listar publicaciones, Editar publicación, Eliminar publicación, Buscar publicación, Buscar organismo, Mi proyecto, Mostrar proyecto y Recomendaciones de proyectos.

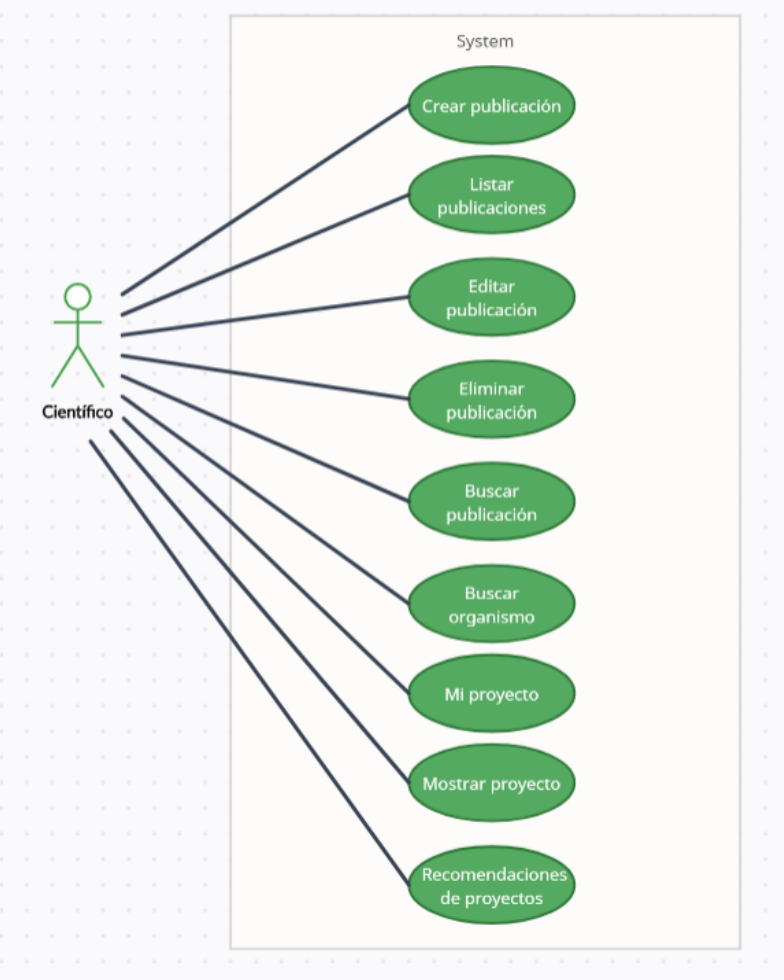

Figura 4.1: Diagrama de casos de uso del módulo Científico

A continuación, se listan los casos de uso asociados a este módulo.
(Desde la Tabla 4.1 hasta la Tabla 4.9)

Requisito	Crear publicación	
Identificador	1.1	
Prioridad	Alta	
Precondición	No debe existir otra publicación con el mismo identificador.	
Descripción	Se da de alta una publicación en la aplicación para que el científico pueda buscar proyectos y recibir recomendaciones.	
Entrada	Título, descripción, especialidad y experiencia (años y meses)	
Salida	Mensaje emergente de que la publicación se ha dado de creado correctamente	
Secuencia normal	Paso	Acción
	1	Se le muestra al usuario un formulario donde debe introducir todos los datos de entrada para crear la publicación.
	2	El usuario rellena el formulario y hace click en “Crear”. Aparecerá un mensaje emergente preguntando si está seguro de querer crear la publicación. Debe hacer click de nuevo en otro botón llamado también “Crear”.
	3	Se valida la información introducida y, si tiene un formato correcto, se realiza el alta de la publicación en la base de datos.
Postcondición	Se habrá creado una publicación en la aplicación.	
Excepciones	Paso	Acción
	2	En el mensaje emergente, aparece también otra opción llamada “Cancelar”. Si se hace click en ella, el mensaje emergente desaparece y se vuelve a la pantalla del formulario.
	3	Si alguno de los datos introducidos no tiene un formato correcto, se muestra un mensaje informativo y se vuelve a mostrar el formulario con los datos introducidos para que cambie los que son incorrectos.
Actores	Científico	

Tabla 4.1: Caso de uso - Científico - Crear publicación

Requisito	Listar publicaciones	
Identificador	1.2	
Prioridad	Media	
Precondición	El científico debe haber iniciado sesión.	
Descripción	Se muestra en forma de tabla un listado de todas las publicaciones creadas por el científico.	
Entrada	-	
Salida	Listado de publicaciones del científico	
Secuencia normal	Paso	Acción
	1	El científico inicia sesión en la aplicación.
	2	El científico hace click en la sección “Mis publicaciones”.
Postcondición	-	
Excepciones	Paso	Acción
	2	En caso de que el científico no haya creado ninguna publicación, en lugar de la tabla se mostrará el siguiente mensaje en pantalla: *“No existen publicaciones.* *Añade una nueva publicación en "Crear publicación".”*
Actores	Científico	

Tabla 4.2: Caso de uso - Científico - Listar publicaciones

Requisito	Editar publicación	
Identificador	1.3	
Prioridad	Media	
Precondición	La publicación que se quiere editar debe existir en la base de datos y debe estar activa.	
Descripción	Se modifican los datos de la publicación de forma parcial o total.	
Entrada	Identificador de la publicación	
Salida	Mensaje emergente confirmando que los datos de la publicación han sido editados correctamente.	
Secuencia normal	Paso	Acción
	1	Se muestra un formulario al científico con todos los datos modificables de la publicación que se desea editar.
	2	El científico modifica aquellos datos que necesite y hace click en el botón Editar. Aparecerá un mensaje emergente preguntando si está seguro de querer editar la publicación. Debe hacer click de nuevo en otro botón llamado también “Editar”.
	3	Los datos modificados se validan y, posteriormente, se actualizan en la base de datos.
Postcondición	Se habrán editado los datos de la publicación en cuestión.	
Excepciones	Paso	Acción
	2	En el mensaje emergente, aparece también otra opción llamada “Cancelar”. Si se hace click en ella, el mensaje emergente desaparece y se vuelve a la pantalla del formulario.
	3	Si alguno de los datos introducidos no tiene un formato correcto, se muestra un mensaje informativo y se vuelve a mostrar el formulario con los datos introducidos para que cambie los que son incorrectos.
Actores	Científico	

Tabla 4.3: Caso de uso - Científico - Editar publicación

Requisito	Eliminar publicación	
Identificador	1.4	
Prioridad	Alta	
Precondición	La publicación debe existir en la base de datos y estar activa.	
Descripción	Se realiza una baja lógica de la publicación, cambiando a falso su estado de activo.	
Entrada	Identificador de la publicación	
Salida	Mensaje emergente confirmando que la publicación se ha dado de baja correctamente.	
Secuencia normal	Paso	Acción
	1	El científico accede a su listado de publicaciones y selecciona aquella publicación que desea eliminar. Aparece un mensaje emergente preguntando si está seguro de querer eliminar esa publicación.
	2	El científico hace click en el botón “Eliminar”.
	3	Se comprueba que exista la publicación y se da de baja en la base de datos.
Postcondición	Se habrá dado de baja una publicación en la aplicación	
Excepciones	Paso	Acción
	1	En el mensaje emergente, aparece también otra opción llamada “Cancelar”. Si se hace click en ella, el mensaje emergente desaparece y se vuelve a mostrar el listado de publicaciones.
Actores	Científico	

Tabla 4.4: Caso de uso - Científico - Eliminar publicación

Requisito	Buscar publicación	
Identificador	1.5	
Prioridad	Media	
Precondición	La publicación debe existir en la base de datos y estar activa.	
Descripción	Se busca la publicación y se muestran todos sus datos.	
Entrada	Identificador de la publicación	
Salida	Título, descripción, especialidad y experiencia de la publicación.	
Secuencia normal	Paso	Acción
	1	El científico introduce el identificador de la publicación.
	2	Se comprueba si existe una publicación con ese identificador. En caso afirmativo, se muestran sus datos.
Postcondición	Se mostrarán los datos de una publicación.	
Excepciones	Paso	Acción
	2	Si no existe ninguna publicación con ese identificador se muestra un mensaje informativo al científico y se vuelve a pedir que introduzca otro identificador válido.
Actores	Científico	

Tabla 4.5: Caso de uso - Científico - Buscar publicación

Requisito	Buscar organismo	
Identificador	1.6	
Prioridad	Media	
Precondición	El organismo que se busca debe existir en la base de datos.	
Descripción	Se busca un organismo y se muestran todos sus proyectos.	
Entrada	Nombre del organismo	
Salida	Todos los proyectos activos del organismo	
Secuencia normal	Paso	Acción
	1	El científico introduce el nombre total o parcial del organismo que desea buscar.
	2	Se muestra una tabla de resultados mostrando todos los organismos encontrados cuyo nombre coincide total o parcialmente con la búsqueda realizada. Además, permite la posibilidad de listar todos los proyectos activos (si tiene) de cada resultado de la búsqueda.
	3	El científico hace click en “Listar proyectos” de aquel organismo que desee de los resultados de la búsqueda y se muestran todos. Además, permite la posibilidad de ver más información acerca de un proyecto en concreto.
Postcondición	Se lista una serie de organismos y se listan todos sus proyectos.	
Excepciones	Paso	Acción
	1	En caso de que el nombre del organismo no coincida con ningún organismo existente y activo en la base de datos, se muestra en pantalla el siguiente mensaje informativo: *“No se encuentran organismos que contengan tu búsqueda en su nombre.”*
	2	Si se hace click en la opción de listar proyectos de un organismo que no tiene ningún proyecto activo, se muestra por pantalla el siguiente mensaje informativo: *“El organismo seleccionado no tiene ningún proyecto en activo.”*
Actores	Científico, Organismo	

Tabla 4.6: Caso de uso - Científico - Buscar organismo

Requisito	Mi Proyecto	
Identificador	1.7	
Prioridad	Media	
Precondición	El científico debe haber iniciado sesión en la aplicación.	
Descripción	El científico puede consultar la información del proyecto en el que está participando.	
Entrada	Identificador del proyecto	
Salida	Título, descripción, ámbito, subámbito, duración e integrantes del proyecto en el que está participando.	
Secuencia normal	Paso	Acción
	1	El científico hace click en “Mi proyecto” en el panel de usuario.
	2	Si el científico está participando en un proyecto, se muestra toda su información.
Postcondición	Se mostrará la información del proyecto en el que participa el científico.	
Excepciones	Paso	Acción
	1	Si el científico no está participando en ningún proyecto, se muestra por pantalla el siguiente mensaje informativo: *“No estás participando en ningún proyecto ¡Busca un proyecto en el que participar!”*
Actores	Científico	

Tabla 4.7: Caso de uso - Científico - Mi proyecto

Requisito	Mostrar proyecto	
Identificador	1.8	
Prioridad	Media	
Precondición	El proyecto debe existir en la base de datos y estar activo.	
Descripción	Se muestra toda la información referente a un proyecto.	
Entrada	Identificador del proyecto	
Salida	Título, descripción, ámbito, subámbito, duración e integrantes del proyecto.	
Secuencia normal	Paso	Acción
	1	El científico hace click en el botón “+Información”.
	2	Se muestran todos los datos del proyecto.
Postcondición	Se mostrarán los datos de un proyecto.	
Excepciones	Paso	Acción
	-	-
Actores	Científico	

Tabla 4.8: Caso de uso - Científico - Mostrar proyecto

Requisito	Recomendaciones de proyectos	
Identificador	1.9	
Prioridad	Alta	
Precondición	El científico debe haber iniciado sesión y debe haber creado al menos una publicación.	
Descripción	Basado en la profesión del científico y en el ámbito y subámbito de los proyectos, se ofrecen recomendaciones de proyectos que pueden ser de su interés.	
Entrada	-	
Salida	Recomendaciones de proyectos activos en los que poder participar.	
Secuencia normal	Paso	Acción
	1	El científico debe hacer click en la sección “Recomendaciones de proyectos”. Al cargar la página, se muestran automáticamente recomendaciones de proyectos. El científico, puede hacer click en el botón “+ Información” que contiene cada recomendación para ver más información sobre ella.
Postcondición	Se mostrarán recomendaciones de proyectos.	
Excepciones	Paso	Acción
	1	Si el científico no ha creado ninguna publicación o no hay proyectos activos relacionados con la profesión del científico, no se mostrará ninguna recomendación por pantalla.
Actores	Científico	

Tabla 4.9: Caso de uso - Científico - Recomendaciones de proyectos

4.2.2 Módulo Organismo

El módulo organismo envuelve todas las funcionalidades que puede realizar el organismo.

Estas incluyen: Publicar proyecto, Listar proyectos, Editar proyecto, Eliminar proyecto, Buscar proyecto, Buscar científico, Mostrar científico y Recomendaciones de científicos.
(Ver Figura 4.2).

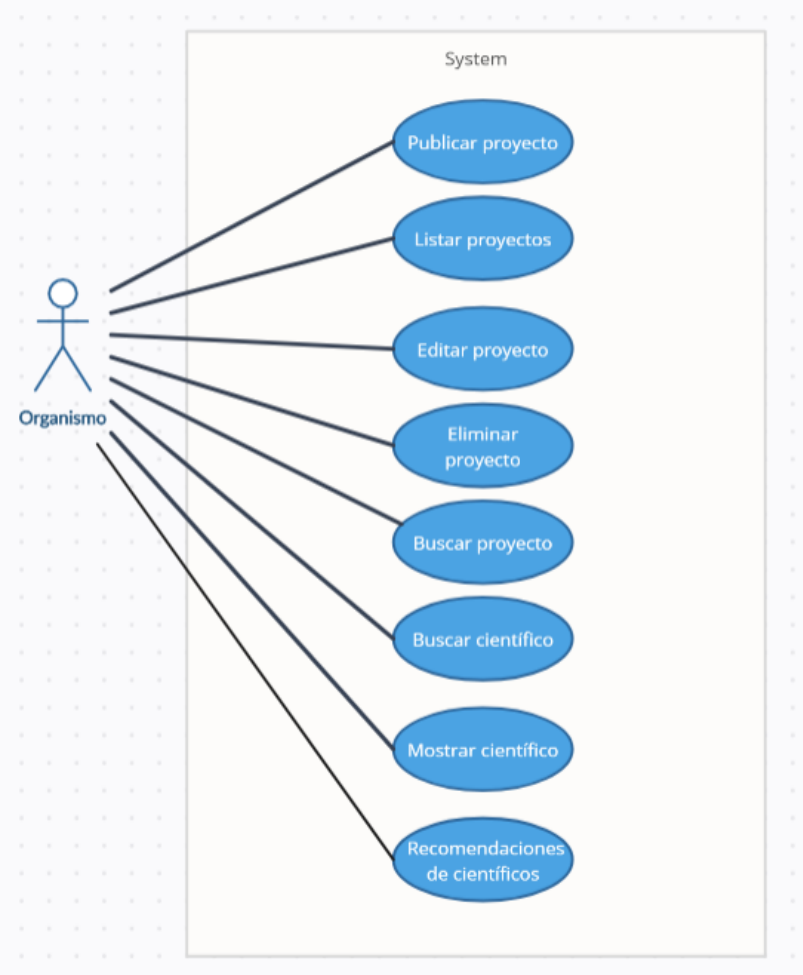

Figura 4.2: Diagrama de casos de uso del módulo Organismo

A continuación, se listan los casos de uso asociados a este módulo.
(Desde la Tabla 4.10 hasta la Tabla 4.17)

Requisito	Publicar proyecto	
Identificador	2.1	
Prioridad	Alta	
Precondición	No debe existir otro proyecto con el mismo identificador.	
Descripción	Se da de alta un proyecto en la aplicación	
Entrada	Título, descripción, capacidad de integrantes, ámbito, subámbito y duración del proyecto.	
Salida	Mensaje emergente de que el proyecto se ha creado correctamente.	
Secuencia normal	Paso	Acción
	1	Se muestra un formulario con toda la información requerida para poder publicar el proyecto.
	2	El organismo rellena el formulario y hace click en “Publicar”. Aparecerá un mensaje emergente preguntando si está seguro de querer publicar el proyecto. Debe hacer click de nuevo en otro botón llamado también “Publicar”.
	3	Se valida la información introducida y, si tiene un formato correcto, se realiza el alta del proyecto en la base de datos
Postcondición	Se habrá publicado un proyecto en la aplicación.	
Excepciones	Paso	Acción
	2	En el mensaje emergente, aparece también otra opción llamada “Cancelar”. Si se hace click en ella, el mensaje emergente desaparece y se vuelve a la pantalla del formulario.
	3	Si alguno de los datos introducidos no tiene un formato correcto, se muestra un mensaje informativo y se vuelve a mostrar el formulario con los datos introducidos para que cambie los que son incorrectos.
Actores	Organismo	

Tabla 4.10: Caso de uso - Organismo - Publicar proyecto

Requisito	Listar proyectos

Identificador	2.2	
Prioridad	Media	
Precondición	El organismo debe haber iniciado sesión.	
Descripción	Se muestra en forma de tabla un listado de todos los proyectos creados por el organismo.	
Entrada	-	
Salida	Listado de proyectos del organismo	
Secuencia normal	Paso	Acción
	1	El organismo inicia sesión en la aplicación
	2	El organismo hace click en la sección “Mis proyectos”.
Postcondición	-	
Excepciones	Paso	Acción
	2	En caso de que el organismo no haya publicado ningún proyecto, en lugar de la tabla se mostrará el siguiente mensaje en pantalla: *“No existen proyectos.* *Añade un nuevo proyecto en "Publicar proyecto".*
Actores	Organismo	

Tabla 4.11: Caso de uso - Organismo - Listar proyectos

Requisito	Editar proyecto

Identificador	2.3	
Prioridad	Media	
Precondición	El proyecto que se quiere editar debe existir en la base de datos y debe estar activo.	
Descripción	Se modifican los datos de un proyecto de manera parcial o total.	
Entrada	Identificador del proyecto.	
Salida	Mensaje emergente confirmando que los datos del proyecto han sido editados correctamente.	
Secuencia normal	Paso	Acción
	1	Se muestra un formulario al organismo con todos los datos modificables del proyecto que se desea editar.
	2	El organismo modifica aquellos datos que necesite y hace click en el botón “Editar”. Aparecerá un mensaje emergente preguntando si está seguro de querer editar el proyecto. Debe hacer click de nuevo en otro botón llamado también “Editar”.
	3	Los datos modificados se validan y, posteriormente, se actualizan en la base de datos.
Postcondición	Se habrán editado los datos del proyecto en cuestión.	
Excepciones	Paso	Acción
	2	En el mensaje emergente, aparece también otra opción llamada “Cancelar”. Si se hace click en ella, el mensaje emergente desaparece y se vuelve a la pantalla del formulario.
	3	Si alguno de los datos introducidos no tiene un formato correcto, se muestra un mensaje informativo y se vuelve a mostrar el formulario con los datos introducidos para que cambie los que son incorrectos.
Actores	Organismo	

Tabla 4.12: Caso de uso - Organismo - Editar proyecto

Requisito	Eliminar proyecto

Identificador	2.4	
Prioridad	Alta	
Precondición	El proyecto debe existir en la base de datos y estar activo.	
Descripción	Se realiza una baja lógica del proyecto, cambiando a falso su estado de activo.	
Entrada	Identificador del proyecto.	
Salida	Mensaje emergente confirmando que el proyecto se ha dado de baja correctamente.	
Secuencia normal	Paso	Acción
	1	El organismo accede a su listado de proyectos y selecciona aquel proyecto que desea eliminar. Aparece un mensaje emergente preguntando si está seguro de querer eliminar ese proyecto.
	2	El organismo hace click en el botón “Eliminar”.
	3	Se comprueba que exista el proyecto y se da de baja de la base de datos.
Postcondición	Se habrá dado de baja un proyecto en la aplicación.	
Excepciones	Paso	Acción
	1	En el mensaje emergente, aparece también otra opción llamada “Cancelar”. Si se hace click en ella, el mensaje emergente desaparece y se vuelve a mostrar el listado de proyectos.
Actores	Organismo	

Tabla 4.13: Caso de uso - Organismo - Eliminar proyecto

Requisito	Buscar proyecto	
Identificador	2.5	
Prioridad	Media	
Precondición	El proyecto debe existir en la base de datos y estar activo.	
Descripción	Se busca el proyecto y se muestran todos sus datos.	
Entrada	Identificador del proyecto	
Salida	Identificador, fecha y hora de la última modificación, título, descripción, capacidad, duración, ámbito y subámbito del proyecto.	
Secuencia normal	Paso	Acción
	1	El organismo introduce el identificador del proyecto.
	2	Se comprueba si existe un proyecto con ese identificador. En caso afirmativo, se muestran sus datos.
Postcondición	Se mostrarán los datos de un proyecto	
Excepciones	Paso	Acción
	2	Si no existe ningún proyecto con ese identificador se muestra un mensaje informativo al organismo y se vuelve a pedir que introduzca un identificador válido.
Actores	Organismo	

Tabla 4.14: Caso de uso - Organismo - Buscar proyecto

Requisito	Buscar científico	
Identificador	2.6	
Prioridad	Media	
Precondición	El científico debe existir en la base de datos y estar activo.	
Descripción	Se busca el científico y se muestran todos sus datos.	
Entrada	ORCID del científico que se desea buscar.	
Salida	ORCID, nombre, profesión, email y disponibilidad del científico.	
Secuencia normal	Paso	Acción
	1	El organismo introduce el ORCID del científico que desea buscar.
	2	Se comprueba si existe un científico con ese ORCID. En caso afirmativo, se muestran sus datos.
Postcondición	Se mostrarán los datos de un científico.	
Excepciones	Paso	Acción
	2	Si no existe ningún científico con el ORCID introducido se muestra un mensaje emergente informativo al organismo y se le pide que introduzca un ORCID válido.
Actores	Organismo, Científico	

Tabla 4.15: Caso de uso - Organismo - Buscar científico

Requisito	Mostrar científico	
Identificador	2.7	
Prioridad	Media	
Precondición	El científico debe existir en la base de datos y estar activo.	
Descripción	Se muestra toda la información referente a un científico.	
Entrada	Identificador del científico	
Salida	ORCID, nombre, profesión, email y disponibilidad del científico.	
Secuencia normal	Paso	Acción
	1	El organismo hace click en el botón “+ Información”
	2	Se muestran todos los datos del científico
Postcondición	Se mostrarán los datos de un científico.	
Excepciones	Paso	Acción
	-	-
Actores	Organismo, Científico	

Tabla 4.16: Caso de uso - Organismo - Mostrar científico

Requisito	Recomendaciones de científicos

Identificador	2.8	
Prioridad	Alta	
Precondición	El organismo debe haber iniciado sesión y debe haber publicado al menos un proyecto.	
Descripción	Basado en el ámbito y subámbito del organismo y en la especialidad de las publicaciones de los científicos y sus profesiones, se ofrecen recomendaciones de los científicos disponibles que podrían participar en un proyecto concreto del organismo.	
Entrada	Identificador del proyecto del que se desea recibir recomendaciones de científicos.	
Salida	Recomendaciones de científicos activos que podrían participar en un proyecto.	
Secuencia normal	Paso	Acción
	1	El organismo debe hacer click en la sección “Recomendaciones de científicos”. Después, debe introducir el identificador del proyecto del que desea recibir recomendaciones de científicos y pulsar en “Recomendar”. Se mostrarán las recomendaciones y el organismo, si lo desea, podrá ver más información acerca de un científico en concreto haciendo click en el botón “+ Información” que contiene cada recomendación.
Postcondición	Se mostrarán recomendaciones de científicos.	
Excepciones	Paso	Acción
	1	Si el organismo no ha creado ningún proyecto o no hay científicos activos relacionados con el proyecto, se mostrará el siguiente mensaje por pantalla: *“No se encuentran científicos que trabajen en el ámbito del proyecto”*.
Actores	Organismo, Cientifico	

Tabla 4.17: Caso de uso - Organismo - Recomendaciones de científicos

4.2.3 Módulo Administrador

El módulo administrador envuelve todas las funcionalidades que puede realizar el administrador.

Como se puede ver en la Figura 4.3, estas incluyen: Listar todas publicaciones, Editar publicación, Eliminar publicación, Reactivar publicación, Listar todos usuarios, Editar usuario, Eliminar usuario, Reactivar usuario, Listar todos proyectos, Editar proyecto, Eliminar proyecto y Reactivar proyecto.

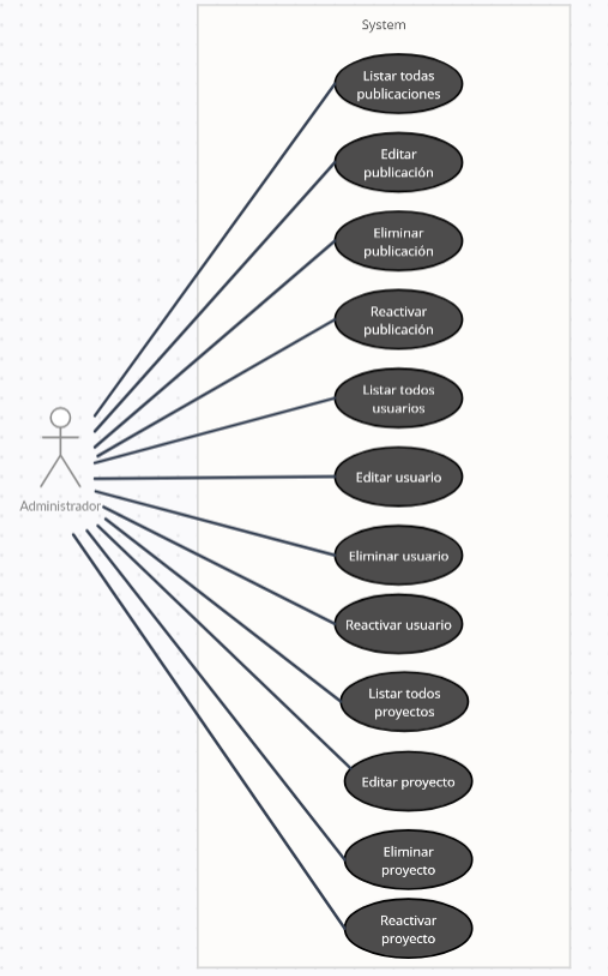

Figura 4.3: Diagrama de casos de uso del módulo Administrador

A continuación, se listan los casos de uso asociados a este módulo
(Desde la Tabla 4.18 hasta la Tabla 4.29)

Requisito	Listar todas publicaciones	
Identificador	3.1	
Prioridad	Alta	
Precondición	El administrador debe haber iniciado sesión.	
Descripción	Se listan en forma de tabla todas las publicaciones de todos los científicos de la aplicación, ya estén activas o no.	
Entrada	-	
Salida	Listado de todas las publicaciones de científicos de la aplicación.	
Secuencia normal	Paso	Acción
	1	El administrador hace click en la sección “Listar publicaciones”. Automáticamente, carga las publicaciones y se muestran listadas en pantalla.
Postcondición	Se mostrará un listado de todas las publicaciones de científicos de la aplicación.	
Excepciones	Paso	Acción
	1	En caso de no existir ninguna publicación en la aplicación se sustituye la tabla por el siguiente texto: *“No existen publicaciones”.*
Actores	Administrador	

Tabla 4.18: Caso de uso - Administrador - Listar todas publicaciones

Requisito	Editar publicación	
Identificador	3.2	
Prioridad	Media	
Precondición	La publicación debe existir en la base de datos y estar activa.	
Descripción	Se modifican los datos de una publicación de científico.	
Entrada	Título, descripción, especialidad y experiencia de la publicación.	
Salida	Mensaje emergente informando que la publicación se ha editado correctamente.	
Secuencia normal	Paso	Acción
	1	El administrador hace click en el botón de Editar disponible en el listado de publicaciones para la publicación que desea editar.
	2	Rellena el formulario editando total o parcialmente los datos de la publicación.
	3	El administrador hace click en el botón Editar. Aparecerá un mensaje emergente preguntando si está seguro de querer editar esa publicación. Por último, selecciona el botón “Editar” del mensaje.
Postcondición	Se editará la información correspondiente a una publicación de científico.	
Excepciones	Paso	Acción
	2	En caso de que alguno de los datos de la publicación que se han editado estén en un formato incorrecto, se mostrará un mensaje emergente.
	3	En el mensaje emergente de confirmación, el administrador puede pulsar también otra opción llamada “Cancelar”. En caso de hacerlo, el mensaje emergente desaparece y se vuelve al formulario de edición.
Actores	Administrador	

Tabla 4.19: Caso de uso - Administrador - Editar publicación

Requisito	Eliminar publicación	
Identificador	3.3	
Prioridad	Media	
Precondición	La publicación debe existir en la base de datos y estar activa.	
Descripción	Se da de baja de forma lógica a una publicación activa en el sistema.	
Entrada	Identificador de la publicación	
Salida	Mensaje de confirmación indicando que se ha realizado correctamente la baja de la publicación.	
Secuencia normal	Paso	Acción
	1	El administrador hace click en el botón de Eliminar disponible en el listado de publicaciones para la publicación que desea eliminar.
	2	Se muestra un mensaje emergente preguntando si desea eliminar esa publicación. Por último, hace click en la opción “Eliminar”.
	3	Se comprueba que exista la publicación y, si es así, se da de baja de forma lógica cambiando su estado de activo.
Postcondición	Se dará de baja una publicación de científico.	
Excepciones	Paso	Acción
	2	En el mensaje emergente de confirmación, el administrador puede pulsar también otra opción llamada “Cancelar”. En caso de hacerlo, el mensaje emergente desaparece y se vuelve al listado de publicaciones.
Actores	Administrador	

Tabla 4.20: Caso de uso - Administrador - Eliminar publicación

Requisito	Reactivar publicación	
Identificador	3.4	
Prioridad	Media	
Precondición	La publicación debe existir en la base de datos y estar dada de baja.	
Descripción	Se reactiva una publicación que ha sido previamente dada de baja.	
Entrada	Identificador de la publicación	
Salida	Mensaje emergente indicando que la publicación ha sido reactivada correctamente.	
Secuencia normal	Paso	
	1	El administrador hace click en el botón de “Reactivar” disponible en el listado de publicaciones para la publicación dada de baja que desea reactivar.
	2	Se muestra un mensaje emergente preguntando si se desea reactivar la publicación. Para hacerlo, debe hacer click en “Reactivar”.
	3	Se comprueba que exista la publicación y que esté dada de baja. Si se cumplen estas condiciones, se reactiva la publicación y se muestra un mensaje emergente de confirmación.
Postcondición	Se reactivará una publicación dada de baja previamente.	
Excepciones	Paso	
	2	En el mensaje emergente, el administrador puede hacer click en la opción “Cancelar”. Si lo hace, el mensaje desaparece y se vuelve al listado de publicaciones.
Actores	Administrador	

Tabla 4.21: Caso de uso - Administrador - Reactivar publicación

Requisito	Listar todos usuarios	
Identificador	3.5	
Prioridad	Alta	
Precondición	El administrador debe haber iniciado sesión.	
Descripción	Se listan en forma de tabla todos los usuarios de la aplicación, estén activos o no. Se muestran en tablas separadas tanto los científicos como los organismos.	
Entrada	-	
Salida	Listados de todos los científicos y organismos de la aplicación.	
Secuencia normal	Paso	Acción
	1	El administrador hace click en la sección “Listar usuarios”. Automáticamente, carga los científicos y los organismos y se muestran listados en pantalla.
Postcondición	Se mostrarán dos listados de todos los usuarios de la aplicación.	
Excepciones	Paso	Acción
	1	En caso de no existir ningún científico o ningún organismo en la aplicación se sustituye la tabla correspondiente por el siguiente texto: “No existen científicos/organismos”.
Actores	Administrador, Científico, Organismo	

Tabla 4.22: Caso de uso - Administrador - Listar todos usuarios

Requisito	Editar usuario	
Identificador	3.6	
Prioridad	Media	
Precondición	El usuario debe existir en la base de datos y estar activo.	
Descripción	Se modifican los datos de un usuario (científico u organismo) de la aplicación.	
Entrada	Si es científico: Nombre y profesión Si es organismo: Nombre y localidad	
Salida	Mensaje de confirmación informando que el usuario (científico u organismo) se ha editado correctamente.	
Secuencia normal	Paso	Acción
	1	El administrador hace click en el botón de Editar disponible en el listado de científicos u organismos para el científico u organismo que desea editar.
	2	Rellena el formulario editando total o parcialmente los datos del científico u organismo.
	3	El administrador hace click en el botón “Editar”. Aparecerá un mensaje emergente preguntando si está seguro de querer editar ese científico u organismo. Por último, selecciona el botón “Editar” del mensaje.
Postcondición	Se editará la información correspondiente a un científico u organismo.	
Excepciones	Paso	Acción
	2	En caso de que alguno de los datos del científico u organismo que se han editado estén en un formato incorrecto, se mostrará un mensaje emergente.
	3	Si alguno de los datos introducidos no tiene un formato correcto, se muestra un mensaje informativo y se vuelve a mostrar el formulario con los datos introducidos para que cambie los que son incorrectos.
Actores	Administrador, Científico, Organismo	

Tabla 4.23: Caso de uso - Administrador - Editar usuario

Requisito	Eliminar usuario	
Identificador	3.7	
Prioridad	Media	
Precondición	El usuario debe existir en la base de datos y estar activo.	
Descripción	Se da de baja de forma lógica a un usuario (científico u organismo) activo en el sistema.	
Entrada	Identificador del usuario (científico u organismo)	
Salida	Mensaje de confirmación indicando que se ha realizado correctamente la baja del usuario.	
Secuencia normal	Paso	Acción
	1	El administrador hace click en el botón de Eliminar disponible en el listado de científicos o de organismos para el científico u organismo que desea eliminar.
	2	Se muestra un mensaje emergente preguntando si desea eliminar ese usuario (científico u organismo). Por último, hace click en la opción “Eliminar”.
	3	Se comprueba que exista el usuario y, si es así, se da de baja de forma lógica cambiando su estado de activo.
Postcondición	Se dará de baja un usuario (científico u organismo).	
Excepciones	Paso	Acción
	2	En el mensaje emergente de confirmación, el administrador puede pulsar también otra opción llamada “Cancelar”. En caso de hacerlo, el mensaje emergente desaparece y se vuelve al listado de usuarios.
Actores	Administrador	

Tabla 4.24: Caso de uso - Administrador - Eliminar usuario

Requisito	Reactivar usuario	
Identificador	3.8	
Prioridad	Media	
Precondición	El usuario (científico u organismo) debe existir en la base de datos y estar dado de baja.	
Descripción	Se reactiva un usuario (científico u organismo) que ha sido previamente dado de baja.	
Entrada	Identificador del usuario (científico u organismo).	
Salida	Mensaje emergente indicando que el usuario (científico u organismo) ha sido reactivado correctamente.	
Secuencia normal	Paso	
	1	El administrador hace click en el botón de “Reactivar” disponible en el listado de científicos u organismos para el usuario (científico u organismo) dado de baja que desea reactivar.
	2	Se muestra un mensaje emergente preguntando si se desea reactivar el usuario. Para hacerlo, debe hacer click en “Reactivar”.
	3	Se comprueba que exista el usuario y que esté dado de baja. Si se cumplen estas condiciones, se reactiva el usuario y se muestra un mensaje emergente de confirmación.
Postcondición	Se reactivará un usuario (científico u organismo) dado de baja previamente.	
Excepciones	Paso	
	2	En el mensaje emergente, el administrador puede hacer click en la opción “Cancelar”. Si lo hace, el mensaje desaparece y se vuelve al listado de científicos u organismos.
Actores	Administrador	

Tabla 4.25: Caso de uso - Administrador - Reactivar usuario

Requisito	Listar todos proyectos	
Identificador	3.9	
Prioridad	Alta	
Precondición	El administrador debe haber iniciado sesión.	
Descripción	Se listan en forma de tabla todos los proyectos de todos los organismos de la aplicación, ya estén activos o no.	
Entrada	-	
Salida	Listado de todos los proyectos de todos los organismos de la aplicación.	
Secuencia normal	Paso	Acción
	1	El administrador hace click en la sección “Listar proyectos”. Automáticamente, carga los proyectos y se muestran listados en pantalla.
Postcondición	Se mostrará un listado de todos los proyectos de todos los organismos de la aplicación.	
Excepciones	Paso	Acción
	1	En caso de no existir ningún proyecto en la aplicación se sustituye la tabla por el siguiente texto: *“No existen proyectos”.*
Actores	Administrador	

Tabla 4.26: Caso de uso - Administrador - Listar todos proyectos

Requisito	Editar proyecto	
Identificador	3.10	
Prioridad	Media	
Precondición	El proyecto debe existir en la base de datos y estar activo.	
Descripción	Se modifican los datos de un proyecto de organismo.	
Entrada	Título, descripción, capacidad de integrantes, duración, ámbito y subámbito del proyecto.	
Salida	Mensaje emergente informando que el proyecto se ha editado correctamente.	
Secuencia normal	Paso	Acción
	1	El administrador hace click en el botón de Editar disponible en el listado de proyectos para el proyecto que desea editar.
	2	Rellena el formulario editando total o parcialmente los datos del proyecto.
	3	El administrador hace click en el botón Editar. Aparecerá un mensaje emergente preguntando si está seguro de querer editar ese proyecto. Por último, selecciona el botón “Editar” del mensaje.
Postcondición	Se editará la información correspondiente a un proyecto de organismo.	
Excepciones	Paso	Acción
	2	En caso de que alguno de los datos del proyecto que se han editado estén en un formato incorrecto, se mostrará un mensaje emergente.
	3	En el mensaje emergente de confirmación, el administrador puede pulsar también otra opción llamada “Cancelar”. En caso de hacerlo, el mensaje emergente desaparece y se vuelve al formulario de edición.
Actores	Administrador	

Tabla 4.27: Caso de uso - Administrador - Editar proyecto

Requisito	Eliminar proyecto	
Identificador	3.11	
Prioridad	Media	
Precondición	El proyecto debe existir en la base de datos y estar activo.	
Descripción	Se da de baja de forma lógica a un proyecto activo en el sistema.	
Entrada	Identificador del proyecto.	
Salida	Mensaje de confirmación indicando que se ha realizado correctamente la baja del proyecto.	
Secuencia normal	Paso	Acción
	1	El administrador hace click en el botón de Eliminar disponible en el listado de proyectos para el proyecto que desea eliminar.
	2	Se muestra un mensaje emergente preguntando si desea eliminar ese proyecto. Por último, hace click en la opción “Eliminar”.
	3	Se comprueba que exista el proyecto y, si es así, se da de baja de forma lógica cambiando su estado de activo.
Postcondición	Se dará de baja un proyecto de organismo.	
Excepciones	Paso	Acción
	2	En el mensaje emergente de confirmación, el administrador puede pulsar también otra opción llamada “Cancelar”. En caso de hacerlo, el mensaje emergente desaparece y se vuelve al listado de proyectos.
Actores	Administrador	

Tabla 4.28: Caso de uso - Administrador - Eliminar proyecto

Requisito	Reactivar proyecto	
Identificador	3.12	
Prioridad	Media	
Precondición	El proyecto debe existir en la base de datos y estar dado de baja.	
Descripción	Se reactiva un proyecto que ha sido previamente dado de baja.	
Entrada	Identificador del proyecto	
Salida	Mensaje emergente indicando que el proyecto ha sido reactivado correctamente.	
Secuencia normal	Paso	
	1	El administrador hace click en el botón de “Reactivar” disponible en el listado de proyectos para el proyecto dado de baja que desea reactivar.
	2	Se muestra un mensaje emergente preguntando si se desea reactivar el proyecto. Para hacerlo, debe hacer click en “Reactivar”.
	3	Se comprueba que exista el proyecto y que esté dado de baja. Si se cumplen estas condiciones, se reactiva el proyecto y se muestra un mensaje emergente de confirmación.
Postcondición	Se reactivará un proyecto dado de baja previamente.	
Excepciones	Paso	
	2	En el mensaje emergente, el administrador puede hacer click en la opción “Cancelar”. Si lo hace, el mensaje desaparece y se vuelve al listado de proyectos.
Actores	Administrador	

Tabla 4.29: Caso de uso - Administrador - Reactivar proyecto

4.2.4 Módulo Cuenta

El módulo cuenta envuelve todas las funcionalidades comunes entre los científicos y los organismos relacionadas con la cuenta de usuario. (Ver Figura 4.4).

Estas incluyen: Registrar usuario, Iniciar sesión, Ver perfil, Editar perfil, Mostrar FAQ y Mostrar contacto.

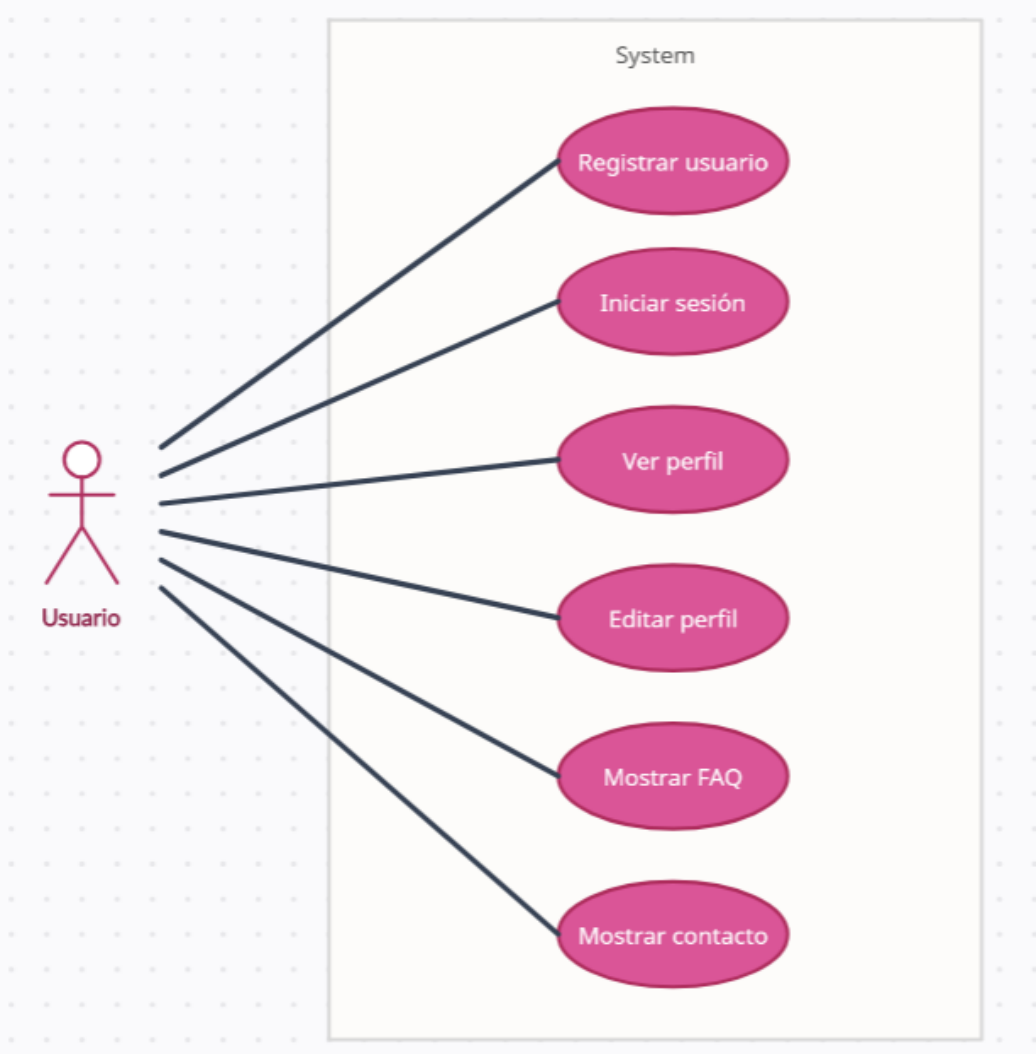

Figura 4.4: Diagrama de casos de uso del módulo Cuenta

A continuación, se listan los casos de uso asociados a este módulo
(Desde la Tabla 4.30 hasta la Tabla 4.35)

Requisito	Registrar usuario	
Identificador	4.1	
Prioridad	Alta	
Precondición	El usuario (científico u organismo) no debe existir en la base de datos.	
Descripción	Se registra un usuario (científico u organismo) en la aplicación.	
Entrada	Si se quiere registrar un científico: Nombre completo, email, ORCID, profesión y contraseña. Si se quiere registrar un organismo público: Código DIR3, nombre, email, localidad y contraseña. Si se quiere registrar un organismo privado: NIF, nombre, email, localidad y contraseña.	
Salida	Mensaje emergente de confirmación indicando que el usuario (científico u organismo) ha sido registrado correctamente.	
Secuencia normal	Paso	Acción
	1	Rellenar el formulario de registro con los datos requeridos (ya sea de científico u organismo). Y hacer click en Registrar.
	2	Se comprueba que los datos introducidos estén en un formato permitido. Si lo están, se da de alta al usuario en la base de datos.
Postcondición	Se registrará un usuario (científico u organismo) en la aplicación.	
Excepciones	Paso	Acción
	2	En caso de que los datos introducidos estén en un formato erróneo, se mostrará un mensaje emergente informando del formato permitido para que se pueda corregir.
Actores	Científico, Organismo	

Tabla 4.30: Caso de uso - Cuenta - Registrar usuario

Requisito	Iniciar sesión	
Identificador	4.2	
Prioridad	Alta	
Precondición	El usuario (científico u organismo) ha sido registrado previamente en la aplicación.	
Descripción	El usuario (científico u organismo) inicia sesión en la aplicación.	
Entrada	Email y contraseña	
Salida	Redirección a la página “Home” del usuario que haya iniciado sesión	
Secuencia normal	Paso	Acción
	1	El usuario (científico u organismo) rellena el formulario de inicio de sesión escribiendo su email y contraseña.
	2	Si los datos son correctos, se redirigirá a la página “Home” correspondiente al usuario que ha iniciado sesión.
Postcondición	Un usuario habrá iniciado sesión en la aplicación.	
Excepciones	Paso	Acción
	2	Si los datos introducidos son erróneos o no corresponden con un usuario registrado, se muestra un mensaje de error.
Actores	Científico, Organismo	

Tabla 4.31: Caso de uso - Cuenta - Iniciar sesión

Requisito	Ver perfil	
Identificador	4.3	
Prioridad	Media	
Precondición	El usuario (científico u organismo) debe haber iniciado sesión en la aplicación y estar activo.	
Descripción	Se muestra toda la información correspondiente al usuario.	
Entrada	-	
Salida	Si lo utiliza un científico: Nombre, ORCID, profesión, email y proyecto en el que está participando. Si lo utiliza un organismo: Nombre, Código DIR3 o NIF, email y localidad.	
Secuencia normal	Paso	Acción
	1	El usuario hace click en la opción “Mi Perfil” del panel de usuario.
	2	Se muestran los datos del usuario.
Postcondición	Se habrán mostrado los datos del usuario que corresponda.	
Excepciones	Paso	Acción
	2	En caso de que el científico vea su perfil y no esté participando en ningún proyecto, se sustituirá el enlace para ver su proyecto por el siguiente texto: *“No participas en ningún proyecto.”*
Actores	Científico, Organismo	

Tabla 4.32: Caso de uso - Cuenta - Ver perfil

Requisito	Editar perfil	
Identificador	4.4	
Prioridad	Media	
Precondición	El usuario (científico u organismo) debe haber iniciado sesión	
Descripción	Se muestra una pantalla en la que se pueden editar los atributos del usuario.	
Entrada	Si lo utiliza un científico: Nombre y profesión. Si lo utiliza un organismo: Nombre y localidad	
Salida	Mensaje emergente indicando que el usuario ha sido editado correctamente.	
Secuencia normal	Paso	Acción
	1	Rellenar el formulario con los datos requeridos del usuario y hacer click en “Editar”.
	2	Se muestra un mensaje emergente preguntando si se desea editar el perfil. En caso afirmativo, hacer click en “Editar”.
Postcondición	Se habrá editado la información de un usuario (científico u organismo) de la aplicación.	
Excepciones	Paso	Acción
	1	En caso de que alguno de los datos del usuario que se han editado estén en un formato incorrecto, se mostrará un mensaje emergente.
	2	En el mensaje emergente, el administrador puede pulsar también otra opción llamada “Cancelar”. En caso de hacerlo, el mensaje emergente desaparece y se vuelve al formulario de edición.
Actores	Científico, Organismo	

Tabla 4.33: Caso de uso - Cuenta - Editar perfil

Requisito	Mostrar FAQ	
Identificador	4.5	
Prioridad	Media	
Precondición	El usuario (científico u organismo) debe haber iniciado sesión y estar activo.	
Descripción	Se muestra una pantalla de preguntas frecuentes (Frequently Asked Questions). Las preguntas y respuestas son diferentes si el usuario es un científico o un organismo.	
Entrada	-	
Salida	Listado de preguntas frecuentes de la aplicación y sus respuestas.	
Secuencia normal	Paso	Acción
	1	El usuario (científico u organismo) abre el menú lateral desplegable y en la sección “Ayuda”, pulsa el botón “FAQ”.
	2	Se redirige a la página de preguntas frecuentes correspondientes, según el tipo de usuario que sea.
Postcondición	Se muestra un listado de preguntas frecuentes de la aplicación que varían según el tipo de usuario que sea.	
Excepciones	Paso	Acción
	-	-
Actores	Científico, Organismo	

Tabla 4.34: Caso de uso - Cuenta - Mostrar FAQ

Requisito	Mostrar contacto	
Identificador	4.6	
Prioridad	Media	
Precondición	El usuario (científico u organismo) debe haber iniciado sesión y estar activo	
Descripción	Se muestra una página de contacto con los detalles del soporte técnico de la aplicación y de sus redes sociales. Esta página es común para ambos usuarios.	
Entrada	-	
Salida	Información de contacto de la aplicación	
Secuencia normal	Paso	Acción
	1	El usuario (científico u organismo) abre el menú lateral desplegable y en la sección “Ayuda”, pulsa el botón “Contacto”.
	2	Se redirige a la página de contacto.
Postcondición	Se habrá mostrado la información de contacto de la aplicación.	
Excepciones	Paso	Acción
	-	-
Actores	Científico, Organismo	

Tabla 4.35: Caso de uso - Cuenta - Mostrar contacto

Capítulo 5 - Arquitectura y modelo de datos

En este capítulo, se realiza una descripción de la arquitectura que compone la aplicación y, posteriormente, se describe la forma en la que se ha realizado la persistencia de datos de la información.

5.1 Arquitectura y modelo de la aplicación

La arquitectura que respalda la aplicación se basa en un modelo cliente-servidor.

El modelo cliente-servidor [30] es un modelo de diseño de software en el que, los usuarios de la aplicación a través del lado del cliente realizan peticiones de una serie de servicios y recursos al servidor. Por su parte, el servidor se encarga de escuchar estas solicitudes y de responder a ellas proporcionando los recursos o servicios solicitados.

En esta aplicación, el lado del cliente está constituido por el navegador web en el que se ejecuta la aplicación, el framework Angular y Firebase.

En el servidor se ha implementado una interfaz de programación de aplicaciones (API). La función de la API es actuar a modo de intermediario entre el cliente (Angular) y la base de datos para realizar operaciones de lectura y escritura en la base de datos.

En la Figura 5.1, se muestra cómo ha sido diseñada la arquitectura de la aplicación utilizando el modelo cliente-servidor.
El navegador se encarga de ejecutar el código HTML y JavaScript proveniente de la compilación de los diversos componentes Angular que constituyen la aplicación, como servicios, plantillas, componentes, entre otros. El lado del cliente es el encargado de realizar peticiones HTTP a los servicios expuestos por la API REST. Ésta se encarga de procesar las solicitudes, extrayendo los datos que sean necesarios de la base de datos. Una vez tratada cada petición, la respuesta del servicio es enviada al cliente y recibida por él.

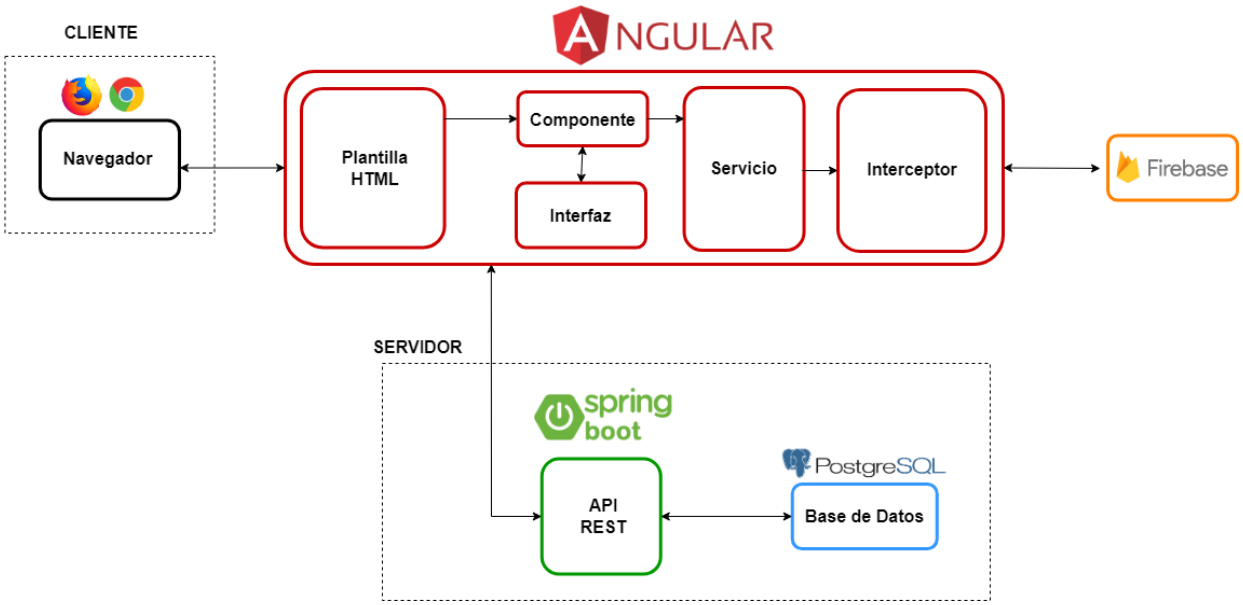

Figura 5.1: Arquitectura de la aplicación

5.2 Patrones de diseño

En esta sección, se realiza una explicación de los patrones de diseño que guían la estructura y el enfoque de los frameworks que se han empleado en la implementación del sistema.

5.2.1 Capa de presentación

La capa de presentación no sigue, en el caso de esta aplicación, el patrón de diseño convencional de modelo-vista-controlador. El patrón utilizado queda definido por la utilización de un framework específico para implementar la capa de presentación: Angular. [31]

Angular hace uso de un concepto denominado *two-way data binding* que genera una dependencia entre el modelo y la vista a la hora de sincronizar los datos. En otras palabras, este concepto permite que desde el modelo se pueda modificar la vista y viceversa. Esto provoca que exista una relación de dependencia muy fuerte entre el modelo y la vista.

Por ello, se dice que Angular sigue un patrón denominado patrón modelo-vista-modelo de vista (Model-View-ViewModel) [32]. Este patrón facilita la mantenibilidad de la aplicación web, la modularización y la reutilización de sus componentes. (Ver Figura 5.2).

- Modelo: Representa a la capa de datos y la lógica de negocio de la aplicación. Contiene la información y las reglas para procesar los datos.

- Vista: Capa que se encarga de la presentación y la interfaz de usuario. Es responsable de mostrar los datos al usuario y recoger sus interacciones.

- Modelo de vista: Es el intermediario entre el Modelo y la Vista. Su utilidad es la de separar la lógica de negocio (situada en el Modelo) de la lógica de presentación (situada en la Vista). Se encarga de solicitar los datos al Modelo y formatearlos para adecuar su presentación en la Vista.

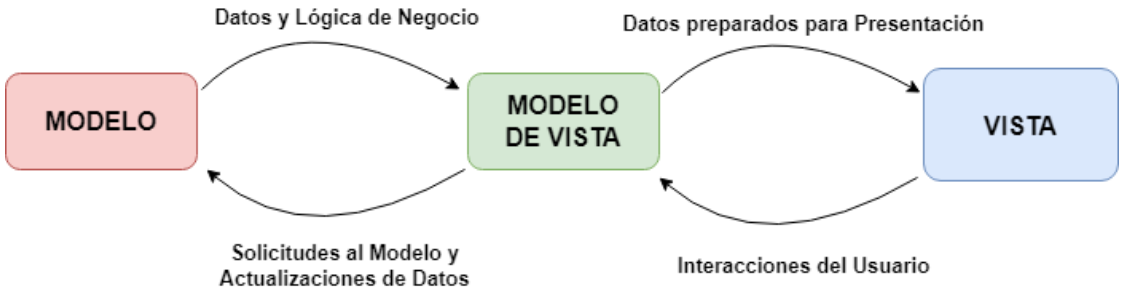

Figura 5.2: Diagrama del Modelo-Vista-Modelo de Vista

5.2.2 Capa de negocio

En el caso de la capa de negocio, para el diseño de software se ha optado por utilizar una arquitectura basada en la Arquitectura en Capas. De esta forma se obtuvo un modelo que permite una mejor organización, una buena modularidad del sistema que facilita el desarrollo y su mantenimiento, y también es escalable según las necesidades del sistema.

Éstas capas o niveles son principalmente:

- La capa *Controller*: En éste nivel se exponen los endpoints para las API Rest.
- La capa *Service*: En éste componente es donde se encapsula toda la lógica de negocio.
- La capa *Repository*: Ésta capa se encarga del sistema de gestión de acceso a la base de datos.
- La capa Model(Entity): Donde están las entidades del negocio.

De ésta forma la división de niveles queda reflejada en la Figura 5.3

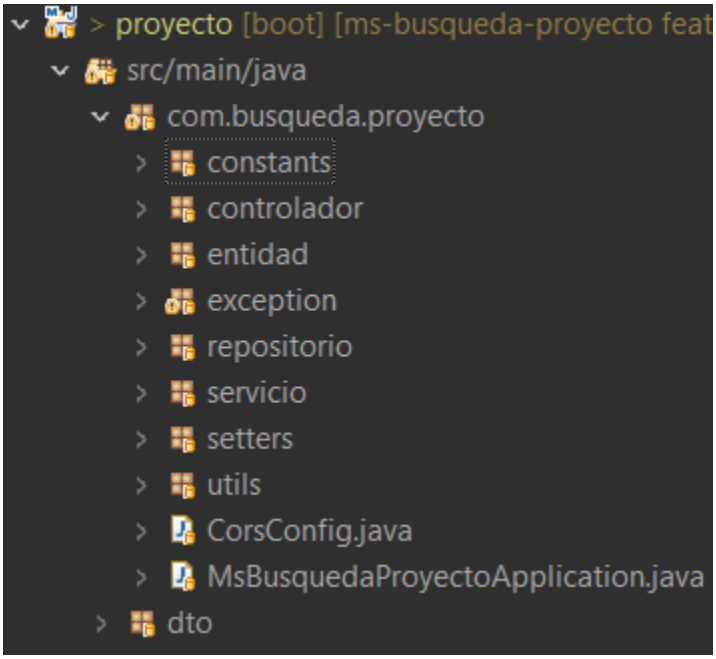

Figura 5.3: Capas o niveles del diseño de Software

Con respecto a los patrones de diseño utilizados en ésta capa de negocio se puede resaltar el siguiente patrón:

El patrón de Inyección de dependencias

Éste patrón es un patrón de diseño orientado a objetos, en el que se suministra objetos a una clase en lugar de ser la propia clase quien vaya a crear dichos objetos, está basada en el principio de Inversión de Control (IoC) para mantener en su "contexto" todas las instancias de la aplicación, y así poder inyectar éstas instancias a quién lo necesite (beans). En ésta aplicación se utilizó un tipo de inyección de dependencia por Atributo, tal y como se muestra en la Figura 5.4.

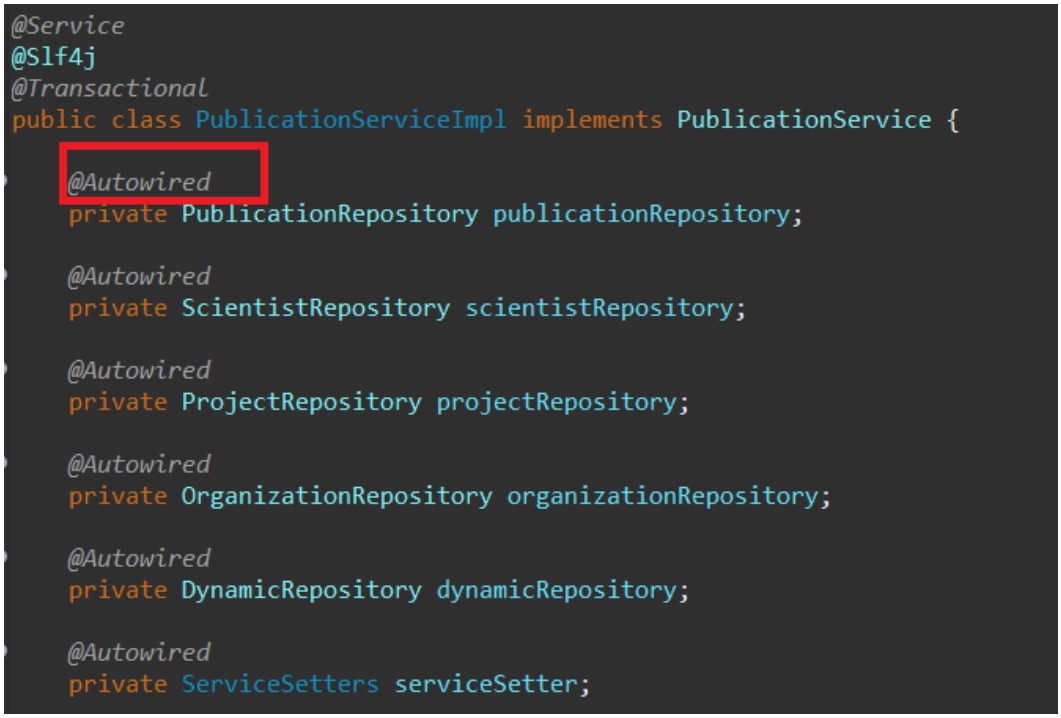

Figura 5.4: IoC por atributo

5.3 Modelo de datos

5.3.1 Modelo Entidad-Relación

En la Figura 5.5 se muestra el diagrama Entidad-Relación que se ha utilizado en el modelado de datos de la aplicación. En este diagrama se presentan las entidades y los atributos definidos para el sistema.

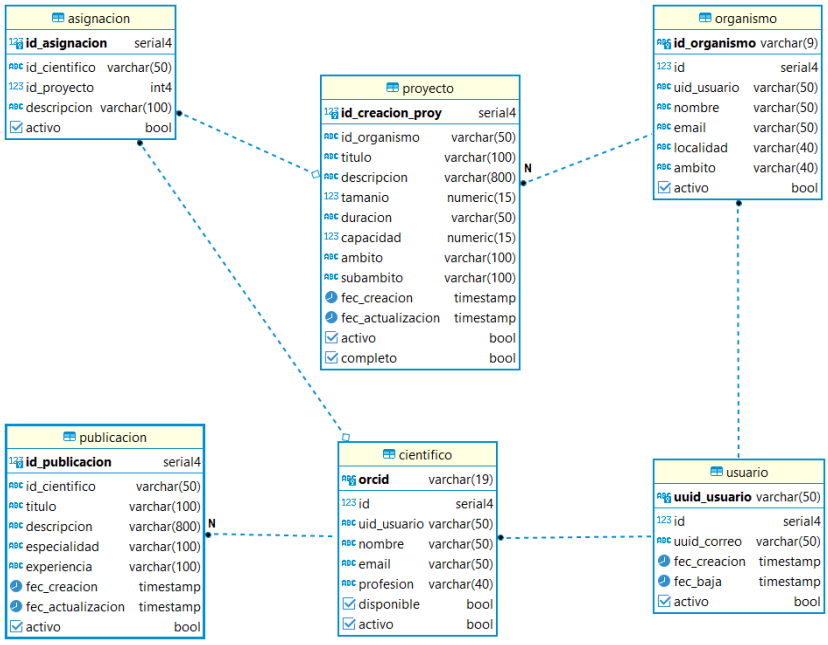

Figura 5.5: Diagrama Entidad-Relación

5.3.2 Implementación de la base de datos

Para poder implementar el modelo de datos descrito anteriormente se ha utilizado un sistema de gestión de base de datos relacional orientado a objetos llamado PostgreSQL que se integra de forma fluida con Spring Boot JPA. Así las consultas de operaciones CRUD [33] resultan más sencillas. A continuación, se muestran las tablas utilizadas en este modelo de datos ordenados por módulos funcionales.

5.3.2.1 Usuario

En este apartado se describe la tabla de Usuario. Se puede resaltar los siguientes atributos; UUID_usuario y UUID_correo. Esto es que una vez se haya registrado el usuario en la aplicación se guardarán en ésta tabla los identificadores UUID de usuario y UUID de correo electrónico generados por Firebase.

id: Valor autoincremental de un usuario

uuid_usuario: Identificador primario de un usuario

uuid_correo: Identificador del correo electrónico de un usuario

fec_creacion: Fecha de creación de un usuario

fec_baja: Fecha de baja de un usuario

activo: Flag de activo/desactivo de un usuario

La relación de la tabla de Usuario con otras tablas se puede apreciar en la siguiente figura 5.6:

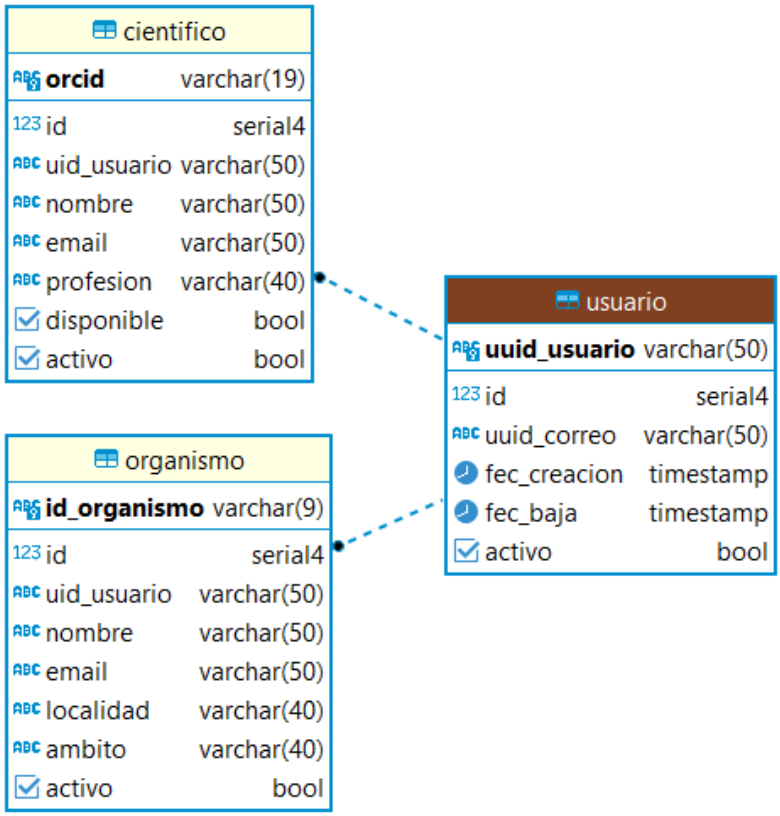

Figura 5.6: Tabla Usuario

5.3.2.2 Científico

En este apartado se describe la tabla de Científico, que representa el rol de usuario de científico quien es capaz de crear publicaciones. Se puede resaltar los atributos uid_usuario que relaciona con la tabla de usuario.

orcid: Identificador primario de un científico

uid_usuario: Identificador foráneo de la relación con la tabla de usuario

nombre: Nombre de un científico

email: Email de un científico

profesion: Profesión de un científico

disponible: Flag de disponible/no disponible de un científico

activo: Flag de activo/desactivo de un científico

Como queda relacionada la tabla de Científico con otra tablas se puede visualizar en la siguiente figura 5.7:

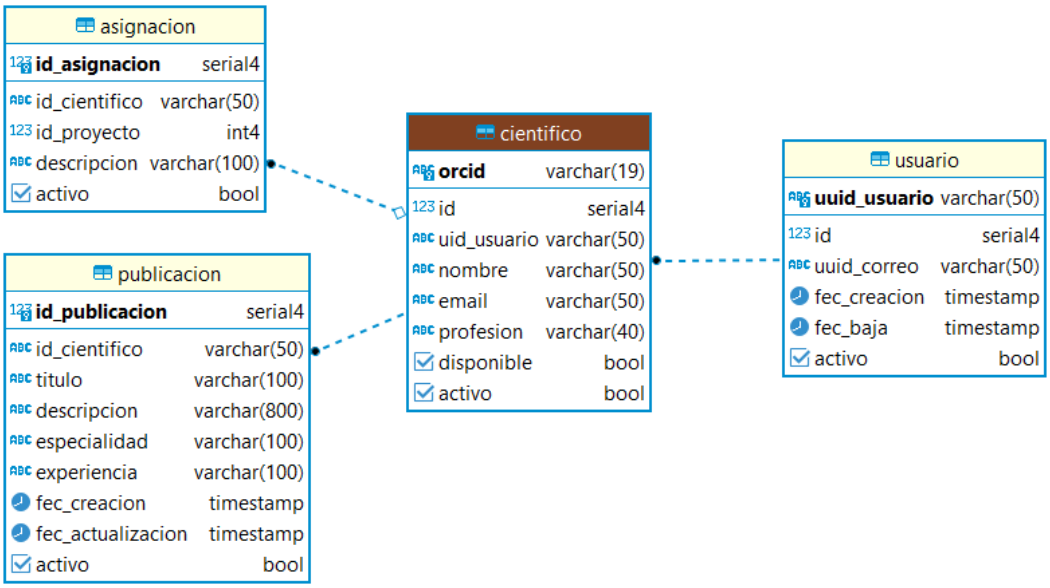

Figura 5.7: Tabla Científico

5.3.2.3 Organismo

En este apartado se describe la tabla de Organismo, que representa el rol de usuario de organismo quien se encarga de crear los proyectos. Al igual que en el anterior apartado ésta tabla está relacionada con la tabla de usuario, mediante el uid_usuario.

id: Valor autoincremental de un organismo

id_organismo: Identificador primario de un organismo

uid_usuario: Identificador foráneo de la relación con la tabla de usuario

nombre: Nombre de un organismo

email: Email de un organismo

localidad: Localidad de un organismo

ambito: Flag de ámbito público/privado de un organismo

activo: Flag de activo/desactivo de un organismo

El diagrama de relación de la tabla de Organismo quedaría de la siguiente forma: (Ver Figura 5.8)

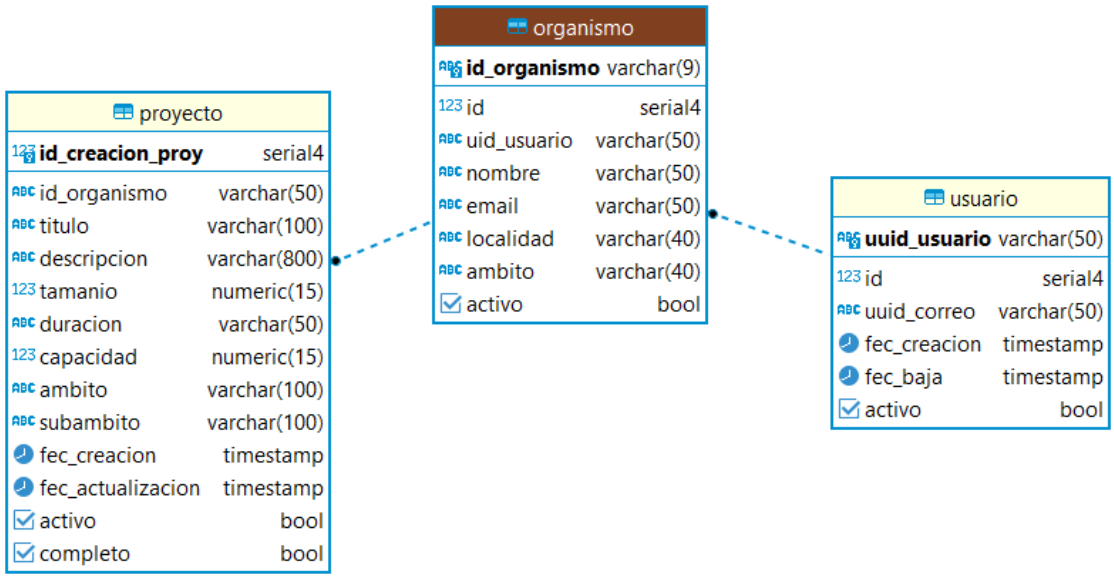

Figura 5.8: Tabla Organismo

5.3.2.4 Publicación

En este apartado se describe la tabla de Publicación, y como dijimos en anteriores apartados son creadas por un usuario científico.

id_publicacion: Identificador primario de una publicación

id_cientifico: Identificación foráneo de la relación con la tabla de científico

titulo: Título de una publicación

descripcion: Descripción de una publicación

especialidad: Especialidad del científico sobre una publicación

experiencia: Experiencia del científico sobre una publicación

fec_creacion: Fecha de creación de una publicación

fec_actualizacion: Fecha de actualización de una publicación

activo: Flag de activo/desactivo de una publicación

La relación de ésta tabla de Publicación con la tabla de científico queda como se muestra en la siguiente figura 5.9:

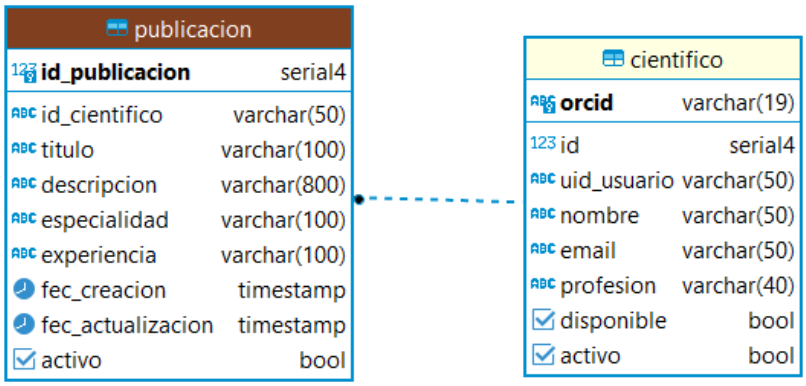

Figura 5.9: Tabla Publicación

5.3.2.5 Proyecto

En este apartado se detallan los atributos que componen la tabla de Proyecto. Éstos proyectos los crean los propios usuarios organismo.

id_creacion_proy: Identificador primario de un proyecto

id_organismo: Identificación foráneo de la relación con la tabla de organismo

titulo: Título de un proyecto

descripcion: Descripción de un proyecto

tamanio: Tamaño acumulado de un proyecto

duracion: Duración de un proyecto

capacidad: Tamaño máximo de un proyecto

ambito: Ámbito de especialidad de un proyecto

subambito: Subámbito de especialidad de un proyecto

fec_creacion: Fecha y hora de creación del proyecto

fec_actualizacion: Fecha y hora de última actualización del proyecto.

activo: Flag activo/no activo de un proyecto

completo: Flag de completo/no completo de un proyecto

A continuación las relaciones de ésta tabla de Proyecto con las tablas de organismo y asignación se muestran en la figura 5.10:

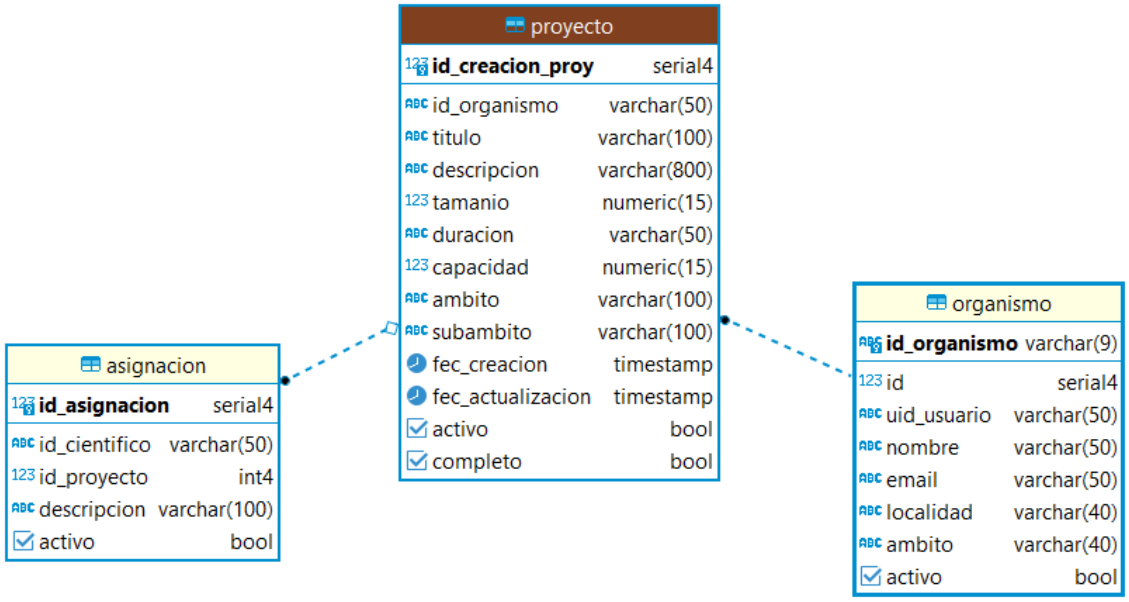

Figura 5.10: Tabla Proyecto

5.3.2.6 Asignación

En este apartado se detalla los atributos de la tabla de Asignación que es la que hace de nexo entre las tablas de científicos y proyectos.

id_asignacion: Identificador primario de una asignación científico-proyecto

id_cientifico: Identificación foráneo de la relación con la tabla de científico

id_proyecto: Identificación foráneo de la relación con la tabla de proyecto

descripcion: Descripción opcional de una asignación

activo: Flag de activo/ no activo de una asignación

En la siguiente figura 5.11 se muestra como queda relacionada la tabla de Asignación:

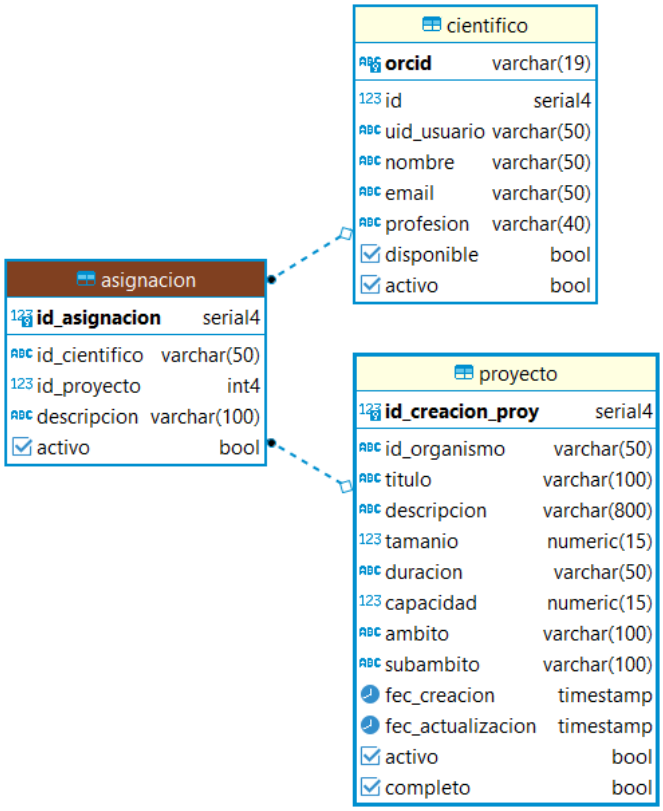

Figura 5.11: Tabla Asignación

Capítulo 6 - Diseño y funcionalidades

En este capítulo se explica el diseño de la aplicación y sus funcionalidades.
En primer lugar, se tratan los estilos utilizados a lo largo de la aplicación y, después, se realiza una disertación sobre las distintas funcionalidades de la aplicación web y de la API.

Como los científicos y organismos son los principales usuarios a los que está dirigida esta aplicación web, se ha optado por hacer un diseño igualmente entendible tanto para aquellas personas familiarizadas con el uso de sistemas informáticos como para aquellas menos experimentadas en este aspecto. Se ha tratado de mantener un diseño bastante simple y amigable para el usuario, garantizando que no exista un gran número de elementos en pantalla. Además, la utilización de iconos ha sido una de las principales técnicas empleadas para hacer la aplicación más accesible. Los iconos empleados en la aplicación web provienen tanto de Font Awesome [34] como, sobre todo, de Bootstrap Icons [35].

6.1 Estilos

Respecto a los estilos utilizados en la aplicación se puede realizar una división entre el logotipo, la paleta de colores y la tipografía empleada.

Logotipo:

En la Figura 6.1 se muestra el logotipo de la aplicación, que está compuesto por el nombre de la aplicación, llamada “ScienceHub” y un icono de un matraz redondo. Este instrumento es utilizado por los científicos en los laboratorios y se suele asociar comúnmente con la ciencia.

Figura 6.1: Logotipo de ScienceHub

Paleta de colores:

La paleta de colores de la aplicación viene dada por la combinación entre varios colores según el usuario que utiliza la aplicación y el módulo en el que se encuentra.

- Para los científicos, se ha utilizado una combinación de colores que incluye el verde, el negro y el blanco. El verde ha sido asociado a lo largo de la aplicación al científico ya que es un color comúnmente asociado a la ciencia.
  Su código hexadecimal es el: #53B200.

- Para los organismos, la paleta de colores se compone de tonos azules, negros y blancos. El color azul ha sido elegido por su facilidad de contraste con el color verde del científico. Además es un color asociado de forma general con la confianza, serenidad y profesionalismo de un organismo.
  Su código hexadecimal es el: #00A2FF

- Para las páginas comunes para los usuarios científico y organismo, se han empleado los colores: turquesa, negro y blanco. La elección del uso del color turquesa viene dada por su propia naturaleza al tratarse de un color combinado entre el color verde y el azul.
  Su código hexadecimal es el: #40E0D0

- Para el administrador, el abanico de colores comprende el gris, el negro y el blanco. El color gris transmite autoridad y control, facultades necesarias para desempeñar el rol de administrador de la aplicación.
  Su código hexadecimal es el: #808080

Tipografía:

En la aplicación se han utilizado distintos tipos de tipografías según el carácter e importancia del texto en la pantalla.

Todos los tipos de fuentes empleados en la aplicación pertenecen a Google Fonts [36] y, en la aplicación, pueden variar de formas ya sea porque estén en negrita, en cursiva, subrayadas o porque, simplemente, no se les aplique ningún estilo adicional.

- Para el título del módulo presente en la cabecera de cada página, se ha utilizado la fuente '*Staatliches*'.

- Para la sección de enlaces de cada página, se ha empleado la fuente '*Nunito*'.

- Para el título de los casos de uso de cada página, se ha empleado la fuente '*Raleway*'.

- Para los campos de los formularios y párrafos, se ha empleado la fuente '*Source Sans Pro*'.

6.2 Funcionalidades de la aplicación

En esta sección se realiza una descripción detallada de las funcionalidades más importantes de la aplicación.

En la Figura 6.2 se muestra un diagrama de todos los módulos que componen la aplicación y sus componentes y servicios: *AppModule* (módulo raíz), *ScientistModule* (módulo de científico), *OrganizationModule* (módulo de organismo), *AccountModule* (módulo de operaciones comunes a científico y organismo) y *AdministratorModule* (módulo de administrador).

En el diagrama, se muestran en color verde los módulos, en color amarillo los componentes (excepto el componente principal de la aplicación AppComponent, que se muestra en color azul) y en color naranja los servicios.

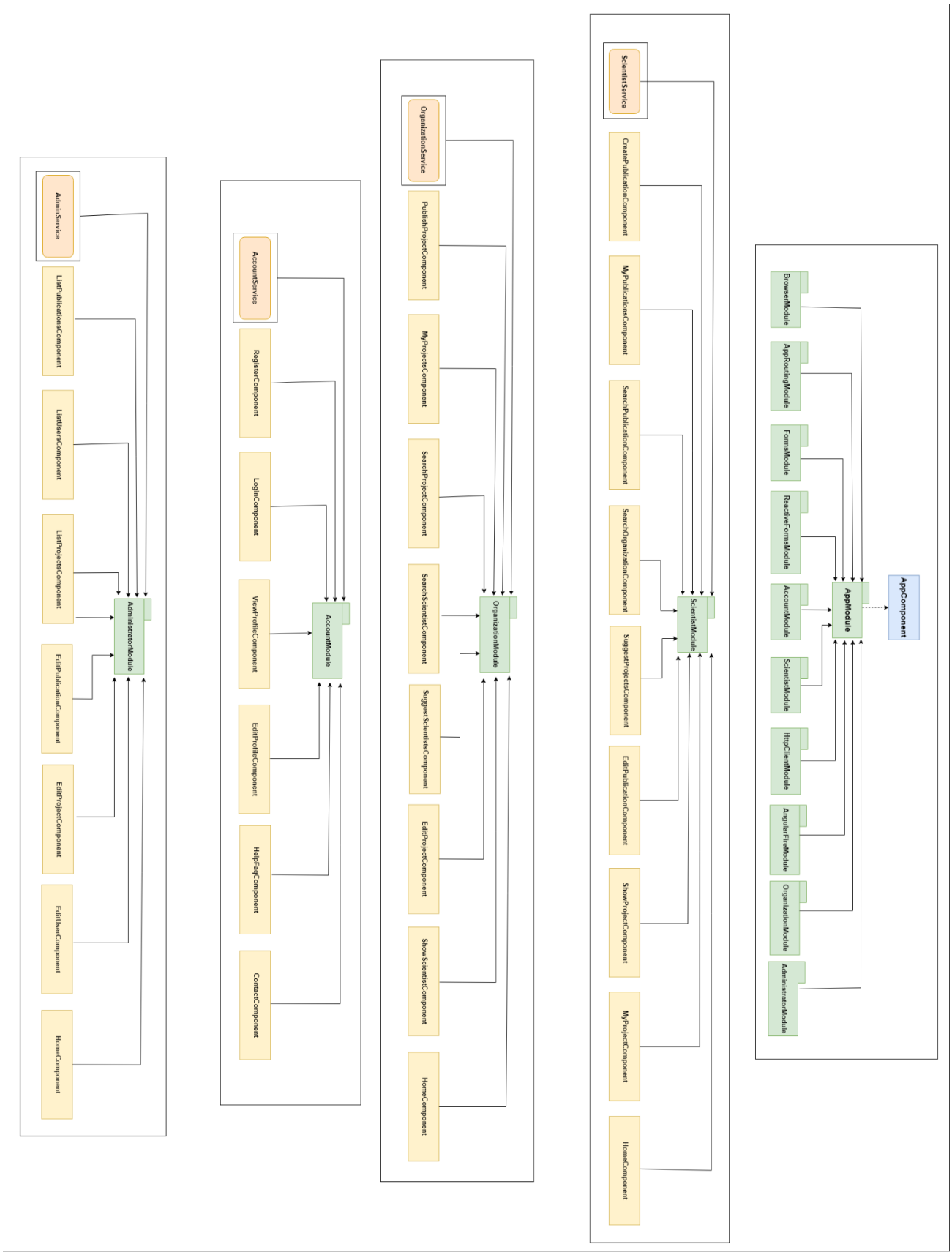

Figura 6.2: Diagrama de módulos de la aplicación

6.2.1 Registro de usuario

El registro de usuario es la primera funcionalidad que debe utilizar el usuario para poder utilizar la aplicación. Esta funcionalidad le permite darse de alta como usuario según su tipo (científico u organismo).

A pesar de que los formularios de cada usuario contienen datos diferentes, el procedimiento de registro para cada uno de ellos sigue los mismos pasos.

En primer lugar, se recogen los datos introducidos en el formulario y se validan. Estas validaciones incluyen:

1. Validación de los campos vacíos: Esta validación consiste en comprobar que ninguno de los campos del formulario se encuentre sin rellenar.
2. Validación de la longitud de la contraseña: Esta validación consiste en comprobar la longitud de la contraseña escrita. En la aplicación, la contraseña de los usuarios debe tener una longitud de, al menos, seis caracteres.
3. Validación de formato: Esta validación consiste en comprobar el formato de algunos campos que requieren de un formato concreto. En la aplicación, estos campos incluyen el email, el ORCID del científico, el Código DIR3 del organismo público y el NIF del organismo privado. La validación de estos campos se realiza mediante el uso de expresiones regulares.

   En segundo lugar, una vez que los datos han sido validados se realizan tres operaciones asíncronas de registro. En caso de que ocurra un error en alguna de ellas, la operación se revierte y el error se captura.

   La primera operación consiste en registrar el usuario en Firebase. Para ello, se llama al método de la Figura 6.1, proporcionando las credenciales que se quieren dar de alta.

```
//1º. Registrar usuario en Firebase
const userCredential = await this.afAuth.createUserWithEmailAndPassword(emailElement.value, passwordElement.value);
uid = userCredential.user?.uid;
```

Figura 6.1: Registro de usuario en Firebase

A continuación, se realiza la segunda operación que consiste en registrar al usuario en la tabla Usuario de la base de datos local. Esta operación, como se puede observar en la Figura 6.2, realiza una petición HTTP POST a la parte backend de la aplicación.

```
// 2º. UID del usuario existente, se registra usuario en la BD local
if(uid != null){

  const formUser = {
    uuId: uid,
    uuidEmail: email,
    active: true
  }

  const jsonUser = JSON.stringify(formUser);

  await fetch('http://localhost:8080/api/searchProject/user', {
    method: 'POST',
    body: jsonUser,
    headers: {
      'Content-Type': 'application/json'
    }
  })
  .then(response => response.json())
  .then(data => {
    console.log('Respuesta del servidor:', data);
  })
  .catch(error => {
    ok=false;
    console.error('Error al enviar los datos:', error);
    alert(error.message);
  });
```

Figura 6.2: Registro de usuario en la base de datos local tabla Usuario

En última instancia, se realiza la tercera operación que consiste en registrar al usuario en su tabla correspondiente de la base de datos local, según su tipo: científico u organismo. Para ello, se realiza otra petición HTTP POST al backend de la aplicación.

En la Figura 6.3, se proporciona el código de registro de un científico en la tabla Científico de la base de datos.

Por su parte, en la Figura 6.4, se proporciona el código de registro de un organismo público en la tabla Organismo de la base de datos. Para registrar un organismo privado el procedimiento sería igual al del organismo público pero cambiando sus atributos.

```
// 3º. Se inserta el cientifico en la BD local.
const formData ={
  orcid: orcid,
  userUuid: uid,
  name: nombre,
  email: email,
  profession: profesion,
  active: true
}

const jsonData = JSON.stringify(formData);

await fetch('http://localhost:8080/api/searchProject/scientist', {
  method: 'POST',
  body: jsonData,
  headers: {
    'Content-Type': 'application/json'
  }
})
.then(response => response.json())
.then(data => {
  console.log('Respuesta del servidor:', data);

})
.catch(error => {
  ok=false;
  console.error('Error al enviar los datos:', error);
  alert(error.message);
});

if(ok){alert('¡El científico ha sido registrado correctamente! \n ');}
```

Figura 6.3: Registro de usuario en la base de datos tabla Científico.

```
// 3º. Se inserta el organismo público en la BD local.
const formDataPublic ={
  idOrganization: dir3,
  userUuid: uid,
  name: nombre,
  email: email,
  location: localidad,
  area: "Público",
  active: true
}

const jsonData = JSON.stringify(formDataPublic);
console.log(jsonData);

await fetch('http://localhost:8080/api/searchProject/organization', {
  method: 'POST',
  body: jsonData,
  headers: {
    'Content-Type': 'application/json'
  }
})
.then(response => response.json())
.then(data => {
  console.log('Respuesta del servidor:', data);
})
.catch(error => {
  ok = false;
  console.error('Error al enviar los datos:', error);
  alert(error.message);
});

if(ok){alert('¡El organismo público ha sido registrado correctamente! \n ');}
```

Figura 6.4: Registro de usuario en la base de datos tabla Organismo

6.2.2 Inicio de sesión

El inicio de sesión es la funcionalidad que permite al usuario acceder a la aplicación utilizando las credenciales (email y contraseña) elegidas durante el registro.

Para lograr iniciar sesión, en primer lugar se llama al método *signInWithEmailAndPassword* presente en la Figura 6.5 y perteneciente a Firebase, proporcionando las credenciales del usuario que quiere iniciar sesión. Este método permite comprobar si las credenciales introducidas están registradas en Firebase o no.
En caso de estarlo, se obtiene el UID de usuario para su posterior almacenaje en la base de datos y se realiza una petición HTTP GET a la parte backend que devuelve el identificador del usuario y el tipo de usuario que es (científico u organismo).
En caso contrario, se detecta que las credenciales introducidas no son válidas y lanza un error que impide iniciar sesión.

```
login(email: string, password: string): Observable<any> {
  return from(this.afAuth.signInWithEmailAndPassword(email, password)).pipe(
    switchMap((userCredential) => {

      if (userCredential.user) {
        const uid = userCredential.user.uid;
        const url = `${this.apiUrl}/login/${uid}`;

        return this.http.get(url).pipe(
          catchError((error: any) => {
            console.error('Error en la petición GET:', error);
            return throwError('Error en la petición GET');
          })
        );
      } else {
        return throwError('No se pudo obtener el UID de usuario.');
      }
    }),
    catchError((error: any) => {
      console.error('Error al autenticar usuario:', error);
      return throwError(error);
    })
  );
}
```

Figura 6.5: Inicio de sesión en Firebase

Acto seguido, como se muestra en la Figura 6.6 para el caso del científico y en la Figura 6.7 para el caso del organismo, según el tipo de usuario que quiere iniciar sesión, se realiza una petición HTTP GET para obtener toda la información del usuario, almacenarla en un servicio compartido en toda la aplicación llamado *userService* y redirigir al usuario a la pantalla

“Home” del módulo que le corresponda. Este servicio utiliza la propiedad sessionStorage que almacena la información de forma temporal en el navegador web mientras la pestaña donde se utilice esté abierta.

Gracias al uso de este servicio compartido, será posible realizar funciones en los módulos como mostrar el nombre del usuario que ha iniciado sesión en el panel de usuario o cargar directamente el proyecto en el que participa un científico, entre otras.

```
this.accountService.login(email, password).subscribe(
  async (response: LoginResponse) => {

    // Científico u Organismo
     const userType = response.userType;
     const idUser = response.idUser;

     if (userType === 'cientifico') {

       this.userService.setUserId(idUser);

       try {
         const response = await fetch('http://localhost:8080/api/searchProject/scientist/' + idUser, {
           method: 'GET',
           headers: {
             'Content-Type': 'application/json'
           }
         });

         if (response.ok) {

           const data = await response.json();

           // Reset de atributos de Organismo
           this.userService.setOrgId('');
           this.userService.setUserUuid('');
           this.userService.setName('');
           this.userService.setEmail('');
           this.userService.setLocation('');
           this.userService.setArea('');


           // Set de los atributos de Científico
           this.userService.setOrcid(data.orcid);
           this.userService.setUserUuid(data.userUuid);
           this.userService.setName(data.name);
           this.userService.setEmail(data.email);
           this.userService.setProfession(data.profession);
           this.userService.setAvailable(data.available);

           this.obtenerIdProyecto(data.orcid);

           this.router.navigate(['/scientist/home']);

         } else {
           console.error('Error al obtener los datos:', response.status);
           alert('Error al obtener los datos del usuario.');
         }
       } catch (error) {
         console.error('Error al enviar los datos:', error);
```

Figura 6.6: Inicio de sesión de un científico

```
}else if (userType === 'organismo') {

  this.userService.setUserId(idUser);

  try {
    const response = await fetch('http://localhost:8080/api/searchProject/organization/' + idUser, {
      method: 'GET',
      headers: {
        'Content-Type': 'application/json'
      }
    });

    if (response.ok) {

      const data = await response.json();

      //Reset de los atributos de Científico
      this.userService.setOrcid('');
      this.userService.setUserUuid('');
      this.userService.setIdProject(0);
      this.userService.setName('');
      this.userService.setEmail('');
      this.userService.setProfession('');
      this.userService.setAvailable(true);


      //Set de los atributos de Organismo
      this.userService.setOrgId(data.idOrganization);
      this.userService.setUserUuid(data.userUuid);
      this.userService.setName(data.name);
      this.userService.setEmail(data.email);
      this.userService.setLocation(data.location);
      this.userService.setArea(data.area);


      this.router.navigate(['/organization/home']);

    } else {
      console.error('Error al obtener los datos:', response.status);
      alert('Error al obtener los datos del usuario.');
    }
```

Figura 6.7: Inicio de sesión de un organismo

Por último, en la aplicación, se considera al administrador como un usuario que no requiere de registro y que existe de por sí tanto en la base de datos como en Firebase.
Por ello, para poder iniciar sesión como administrador, se debe comprobar la existencia del usuario en la base de datos y en Firebase y escribir las siguientes credenciales en el inicio de sesión:

Email: admin@correo.com
Contraseña: administrador

6.2.3 Módulo Científico

6.2.3.1 Crear publicación

Esta funcionalidad permite al científico crear una publicación en la aplicación rellenando un formulario.

En el formulario, se debe especificar el título y descripción de la publicación así como la especialidad y la experiencia del científico.

Para implementarla, se recogen los datos introducidos en el formulario, se validan y se forma un objeto con ellos.

Acto seguido, se añade el código para poder mostrar una alerta emergente utilizando la biblioteca de JavaScript de código abierto llamada: *SweetAlert2* [37]. (Ver Figura 6.8).

```
const nuevaPublicacion = {
  "idScientist": idScientist,
  "title": titulo,
  "expertise": especialidad,
  "profExperience": profExpertise,
  "description": descripcion,
  "active": true
};


Swal.fire({
  icon: 'question',
  title: '¿Seguro que quieres crear una publicación?',
  showCancelButton: true,
  confirmButtonText: 'Crear',
  cancelButtonText: 'Cancelar',
}).then((result) => {
  if (result.isConfirmed) {
    this.scientistService.realizarPublicacion(nuevaPublicacion).subscribe(
      (response) => {

        Swal.fire({
          icon: 'success',
          title: '¡Publicación realizada con éxito!',
          text: 'La publicación se ha creado satisfactoriamente.',
          confirmButtonText: 'Vale'
        });

        (document.getElementById('tituloPublicacion') as HTMLInputElement).value = '';
        (document.getElementById('descripcionPublicacion') as HTMLTextAreaElement).value = '';
        (document.getElementById('especialidadPublicacion') as HTMLInputElement).value = '';
        (document.getElementById('experienciaAñosPublicacion') as HTMLInputElement).value = '0';
        (document.getElementById('experienciaMesesPublicacion') as HTMLInputElement).value = '0';
      },
      (error) => {
        console.error('Error al crear la publicación:', error);
      }
    );
  }
});
```

Figura 6.8: Crear publicación

En caso de que en la alerta emergente se seleccione la opción “Crear”, se realiza una llamada al método del servicio *realizarPublicacion* pasándole por parámetro el objeto que contiene la información del formulario.

Como se puede observar en la Figura 6.9, en esta función se realiza la petición HTTP POST a la API para registrar la nueva publicación en la base de datos.

```
private apiUrl = 'http://localhost:8080/api/searchProject';

constructor(private http: HttpClient) { }

realizarPublicacion(publicacion: any): Observable<any>{
  return this.http.post(this.apiUrl + '/publication', publicacion);
}
```

Figura 6.9: Función realizarPublicación

6.2.3.2 Mis publicaciones

Esta funcionalidad se encarga de mostrar un listado en forma de tabla de todas las publicaciones realizadas por el usuario. En caso de que no exista ninguna publicación se sustituye la tabla por un texto que indica que no existen publicaciones creadas.

El listado de publicaciones se carga automáticamente al entrar en la sección “Mis publicaciones” de la aplicación. Para ello, se realiza una llamada a la función *cargarPublicaciones* mostrada en la Figura 6.10. En esta función llama al servicio que hace la petición a la API para obtener el listado de publicaciones que hay que mostrar, pasándole el ORCID del científico que está utilizando el módulo.

```
cargarPublicaciones() {
  this.scientistService.obtenerPublicaciones(this.userService.getOrcid()).subscribe(
    (data: Publicacion[]) => {
      this.publicaciones = data;
    },
    error => {
      console.error('Error al obtener las publicaciones:', error);
    }
  );
}
```

Figura 6.10: Función cargarPublicaciones

A continuación, se muestra en la Figura 6.11 la función obtenerPublicaciones que se encarga de realizar la petición HTTP GET para obtener las publicaciones a listar.

```
obtenerPublicaciones(orcid:string): Observable<Publicacion[]> {
  return this.http.get<Publicacion[]>(this.apiUrl + '/publication/all/'+ orcid);
}
```

Figura 6.11: Función obtenerPublicaciones

6.2.3.3 Editar publicación

Esta funcionalidad permite al científico realizar modificaciones en los campos que conforman una publicación. Para poder utilizar esta funcionalidad, debe haberse creado la publicación que se desea editar.

El acceso a esta funcionalidad se realiza desde la sección “Mis publicaciones” pasando por la URL el identificador de la publicación que se desea editar.

Para cargar la información de la publicación que se desea editar, se obtiene el identificador pasado como parámetro y se llama a la función *obtenerPublicacion* presente en el servicio (ver Figura 6.12).
Ésta se encarga de realizar una petición HTTP GET para obtener la información de la publicación a editar, como se muestra en la Figura 6.13.

```
this.publicationId = parseInt(publicationId);

this.scientistService.obtenerPublicacion(parseInt(publicationId)).subscribe(data => {
  this.tituloPublicacion = data.title;
  this.descripcionPublicacion = data.description;
  this.especialidadPublicacion = data.expertise;
  this.stringExperiencia = data.profExperience;
```

Figura 6.12: Función obtenerPublicacion

```
obtenerPublicacion(idPublicacion:number):Observable<Publicacion>{
  return this.http.get<Publicacion>(this.apiUrl+ '/publication/'+idPublicacion);
}
```

Figura 6.13: Petición GET obtenerPublicacion

Una vez obtenida la información de la publicación, se muestra por pantalla.

Cuando el científico haya editado los campos que sean necesarios, se obtienen sus valores y se validan para formar un objeto. Posteriormente, se muestra la alerta emergente de *SweetAlert2* para preguntar por última vez si se desea editar la publicación. (ver Figura 6.14).

```
const nuevaPublicacion = {
  id: this.publicationId,
  idScientist: this.userService.getUserId(),
  title: titulo,
  expertise: especialidad,
  profExperience: profExperience,
  description: descripcion,
  active: true,
  initLifeDate: "",
  updateLife: ""
}

Swal.fire({
  icon: 'question',
  title: '¿Seguro que quieres editar esta publicación?',
  showCancelButton: true,
  confirmButtonText: 'Editar',
  cancelButtonText: 'Cancelar',
}).then((result) => {
  if (result.isConfirmed) {
    this.scientistService.editarPublicacion(this.publicationId, nuevaPublicacion ).subscribe(
      (data: Publicacion) => {

        Swal.fire({
          icon: 'success',
          title: '¡Publicación editada con éxito!',
          text: 'La publicación se ha editado satisfactoriamente.',
          confirmButtonText: 'Vale'
        }).then((result) => {
          if (result.isConfirmed) {
            this.router.navigate(['scientist/my-publications']);
          }
        });
      },
      error => {
        console.error('Error al editar la publicacion:', error);
      }
    );
  }
});
```

Figura 6.14: Editar publicación

Si la respuesta es afirmativa, se realiza una llamada a la función *editarPublicacion* presente en el servicio que se encarga de realizar la petición HTTP PUT que edita la publicación en la base de datos. (ver Figura 6.15)

```
editarPublicacion(idPublicacion: number, publicacion: Publicacion): Observable<Publicacion> {
  const url = `${this.apiUrl}/publication/${idPublicacion}`;
  return this.http.put<Publicacion>(url, publicacion);
}
```

Figura 6.15: Petición PUT editarPublicacion

Una vez editada la publicación se redirige al usuario a la página “Mis Publicaciones”.

6.2.3.4 Eliminar publicación

Esta funcionalidad permite al científico dar de baja una publicación creada anteriormente.

El acceso a esta funcionalidad se realiza desde la sección “Mis publicaciones” en el que se pasa el identificador de la publicación que se desea eliminar. (ver Figura 6.16).
Se vuelve a utilizar *SweetAlert2* para generar una alerta emergente que pregunte al usuario.

```
eliminarPublicacion(idPublicacion: number) {
  Swal.fire({
    title: '¿Seguro que quieres eliminar esta publicación?',
    icon: 'question',
    showCancelButton: true,
    confirmButtonText: 'Eliminar',
    cancelButtonText: 'Cancelar',
    confirmButtonColor: '#dc3545',
    cancelButtonColor: '#6c757d',
  }).then((result) => {
    if (result.isConfirmed) {
      this.scientistService.eliminarPublicacion(idPublicacion).subscribe(
        () => {
          Swal.fire({
            icon: 'success',
            title: '¡Publicación eliminada con éxito!',
            text: 'La publicación se ha eliminado satisfactoriamente.',
            confirmButtonText: 'Vale'
          }).then((result) => {
            if (result.isConfirmed) {
              window.location.reload();
            }
          });
        },
        error => {
          console.error('Error al eliminar la publicación:', error);
          Swal.fire({
            icon: 'error',
            title: '¡Error!',
            text: 'Ha ocurrido un error al eliminar la publicación',
            confirmButtonText: 'Entendido'
          });
        }
      );
    }
  });
}
```

Figura 6.16: Eliminar publicación

Si la respuesta es afirmativa, se realiza una llamada a la función *eliminarPublicacion* presente en el servicio que se encarga de realizar la petición HTTP DELETE que elimina la publicación en la base de datos. (ver Figura 6.17)

```
eliminarPublicacion(idPublicacion:number):Observable<Publicacion>{
  return this.http.delete<Publicacion>(this.apiUrl+ '/publication/'+idPublicacion);
}
```

Figura 6.17: Petición DELETE eliminarPublicacion

Una vez que la publicación es eliminada se redirige al científico a la página “Mis Publicaciones”.

6.2.3.5 Buscar organismo

Esta funcionalidad permite al científico buscar un organismo escribiendo su nombre total o parcialmente. Una vez encontrado, permite listar sus proyectos.

En primer lugar, se obtiene el nombre introducido por el científico y se valida.

A continuación se llama a la función *buscarOrganismo* presente en el servicio pasándole como parámetro el nombre escrito en la búsqueda. (ver Figura 6.18)

```
buscarOrganismo(){
  const nombreOrg = (document.getElementById('nombreOrganismo') as HTMLInputElement).value;
  if (nombreOrg.trim().length === 0) {
    Swal.fire({
      icon: 'error',
      title: '¡Error!',
      text: 'Por favor, rellena el campo requerido.',
      confirmButtonText: 'Entendido'
    });

    this.mostrarResultados = false;
    this.listaProyectos = false;

    return;
  }

  this.scientistService.buscarOrganismo(nombreOrg).subscribe(
    (data: Organismo[]) => {

      this.mostrarResultados = true;
      this.listaProyectos = false;

      if(data.length == 0){
        this.noResultados = true;
      }else{
        this.noResultados = false;
        this.organismos = data;
      }

      if (nombreOrg.trim().length > 0) {
        this.router.navigate([], {
          relativeTo: this.route,
          queryParams: { search: nombreOrg },
          queryParamsHandling: 'merge'
        });
      }


    },
    error => {
      console.error('Error al obtener los resultados:', error);
```

Figura 6.18: Buscar organismo

Como muestra la Figura 6.19, la función *buscarOrganismo* es la encargada de realizar la petición HTTP GET que obtiene un listado de organismos cuyo nombre coincide total o parcialmente con el del organismo buscado.

```
buscarOrganismo(nombreOrg: string): Observable<Organismo[]>{
  return this.http.get<Organismo[]>(this.apiUrl + '/organization/findByName/' + nombreOrg);
}
```

Figura 6.19: Petición GET buscarOrganismo

Los resultados de la petición GET serán mostrados por pantalla como “Resultados” de la búsqueda. Si no devuelve ningún organismo, se mostrará un mensaje por pantalla indicando que la búsqueda realizada no ha encontrado resultados.

En caso de encontrarlos, cada resultado tendrá disponible la posibilidad de listar todos sus proyectos.

Para realizar esto, se realiza una llamada a la función *listarProyectos* que se muestra en la Figura 6.20, pasándole el identificador del organismo para el cual el científico ha elegido la opción “Listar proyectos”. Esta función realiza la llamada a la función *obtenerProyectos* que se encuentra en el servicio del científico. (ver Figura 6.20).

```
listarProyectos(orgId: number){

  this.scientistService.obtenerProyectos(orgId.toString()).subscribe(
    (data: Proyecto[]) => {

      this.listaProyectos = true;

      if(data.length == 0){
        this.noProyectos = true;
      }else{
        this.noProyectos = false;
        this.proyectos = data;
      }

    },
    error => {
      console.error('Error al obtener los resultados:', error);
      this.listaProyectos = false;
    }
  );
```

Figura 6.20: Listar proyectos

Como se muestra en la Figura 6.21, la función *obtenerProyectos* se encarga de realizar la petición HTTP GET al backend que devuelve el listado de proyectos de un organismo en concreto.

```
obtenerProyectos(orgId:string){
  return this.http.get<Proyecto[]>(this.apiUrl + '/project/all/' + orgId);
}
```

Figura 6.21: Petición GET obtenerProyectos

Los proyectos obtenidos se mostrarán en pantalla. En caso de no obtener ningún proyecto, se muestra un mensaje indicando que el organismo seleccionado no ha creado ningún proyecto.

6.2.3.6 Recomendaciones de proyectos

Esta funcionalidad permite al científico recibir recomendaciones de proyectos en los que poder participar basándose en la profesión que indicó a la hora de registrarse y en el ámbito y subámbito de los proyectos creados.

Esta funcionalidad se realiza automáticamente al entrar en la sección "Recomendaciones de proyectos".

Para implementarla, se realiza una llamada a la función *recomendarProyectos* del servicio de científico pasándole como parámetro el ORCID del científico que ha iniciado sesión y que está utilizando el módulo. (ver Figura 6.22).

```
recomendarProyectos(){

  this.scientistService.recomendarProyectos(this.userService.getOrcid()).subscribe(
    (response: any) => {
      console.log(response);
      this.encontrados = true;
      this.proyectos = response.content;

    },
    (error) => {
      this.encontrados = false;
      console.error('Error al recomendar proyectos:', error);
    }
  );
}
```

Figura 6.22: Recomendaciones de proyectos

Como se muestra en la Figura 6.23, la función *recomendarProyectos* se encarga de realizar la petición HTTP GET al backend para obtener los proyectos recomendados.

```
recomendarProyectos(orcid:string){
  return this.http.get<any>(this.apiUrl + '/scientist/recommendation/'+ orcid);
}
```

Figura 6.23: Petición GET recomendarProyectos

Si se obtienen recomendaciones, se mostrarán en forma de tarjetas al científico.
En caso contrario, se muestra un mensaje indicando que no se han podido obtener recomendaciones.

6.2.4 Organismo

6.2.4.1 Publicar proyecto

Esta funcionalidad permite al organismo publicar un proyecto en la aplicación.

En el formulario, se debe especificar el título, descripción, duración, capacidad de integrantes, ámbito y subámbito del proyecto.

Para implementarla, se recogen los datos introducidos en el formulario, se validan y se forma un objeto con ellos.

Acto seguido, se añade el código para poder mostrar una alerta emergente utilizando *SweetAlert2*. (Ver Figura 6.24).

```
const nuevoProyecto = {
  "idOrganization": idOrganization,
  "title": titulo,
  "description": descripcion,
  "capacity": capacidad,
  "scope": ambito,
  "subscope": subambito,
  "duration": duration,
  "active": true
};

Swal.fire({
  icon: 'question',
  title: '¿Seguro que quieres publicar un proyecto?',
  showCancelButton: true,
  confirmButtonText: 'Publicar',
  cancelButtonText: 'Cancelar',
}).then((result) => {
  if (result.isConfirmed) {
    this.organizationService.publicarProyecto(nuevoProyecto).subscribe(
      (response) => {

        (document.getElementById('tituloProyecto') as HTMLInputElement).value = '';
        (document.getElementById('descripcionProyecto') as HTMLTextAreaElement).value = '';
        (document.getElementById('capacidadProyecto') as HTMLInputElement).value = '2';
        (document.getElementById('duracionAñosProyecto') as HTMLInputElement).value = '0';
        (document.getElementById('duracionMesesProyecto') as HTMLInputElement).value = '1';
        (document.getElementById('ambitoProyecto') as HTMLInputElement).value = '';
        (document.getElementById('subambitoProyecto') as HTMLInputElement).value = '';

        Swal.fire({
          icon: 'success',
          title: '¡Proyecto publicado con éxito!',
          text: 'El proyecto se ha publicado correctamente.',
          confirmButtonText: 'Vale'
        });
      },
      (error) => {
        console.error('Error al publicar el proyecto:', error);
      }
    );
```

Figura 6.24: Publicar proyecto

En caso de que en la alerta emergente se seleccione la opción “Publicar”, se realiza una llamada al método del servicio *publicarProyecto* pasándole por parámetro el objeto que contiene la información del formulario.

Como se puede observar en la Figura 6.25, en esta función se realiza la petición HTTP POST a la API para registrar el nuevo proyecto en la base de datos.

```
private apiUrl = 'http://localhost:8080/api/searchProject';

constructor(private http: HttpClient) { }

publicarProyecto(proyecto: any): Observable<any>{
  return this.http.post(this.apiUrl + '/project', proyecto);
}
```

Figura 6.25: Petición POST publicarProyecto

Una vez publicado el proyecto, se vacían los campos del formulario para que pueda ser utilizado de nuevo.

6.2.4.2 Mis proyectos

Esta funcionalidad se encarga de mostrar un listado en forma de tabla de todos los proyectos realizados por el usuario. En caso de que no exista ningún proyecto se sustituye la tabla por un texto que indica que no existen proyectos creados.

El listado de proyectos se carga automáticamente al entrar en la sección “Mis proyectos” de la aplicación. Para ello, se realiza una llamada a la función *cargarProyectos* mostrada en la Figura 6.26. En esta función se llama al servicio que hace la petición a la API para obtener el listado de proyectos que hay que mostrar, pasándole el identificador del organismo que está utilizando el módulo.

```
cargarProyectos() {

  this.organizationService.obtenerProyectos(this.userService.getOrgId()).subscribe(
    (data: Proyecto[]) => {
      this.proyectos = data;
    },
    error => {
      console.error('Error al obtener los proyectos:', error);
    }
  );
}
```

Figura 6.26: Función cargarProyectos

A continuación, se muestra en la Figura 6.27 la función *obtenerProyectos* que se encarga de realizar la petición HTTP GET para obtener los proyectos a listar.

```
obtenerProyectos(orgId: string): Observable<Proyecto[]> {
  return this.http.get<Proyecto[]>(this.apiUrl + '/project/all/'+ orgId);
}
```

Figura 6.27: Petición GET obtenerProyectos

6.2.4.3 Editar proyecto

Esta funcionalidad permite al organismo realizar modificaciones en los campos que conforman un proyecto. Para poder utilizar esta funcionalidad, debe haberse publicado el proyecto que se desea editar.

El acceso a esta funcionalidad se realiza desde la sección “Mis proyectos” pasando por la URL el identificador del proyecto que se desea editar.

Para cargar la información del proyecto que se desea editar, se obtiene el identificador pasado como parámetro y se llama a la función *obtenerProyecto* presente en el servicio (ver Figura 6.28).

```
this.projectId = parseInt(projectId);

this.organizationService.obtenerProyecto(parseInt(projectId)).subscribe(data => {
  this.tituloProyecto = data.title;
  this.descripcionProyecto = data.description;
  this.capacidadProyecto = data.capacity;
  this.ambitoProyecto = data.scope;
  this.subambitoProyecto = data.subscope;
  this.stringDuracion = data.duration;
```

Figura 6.28: Función obtenerProyecto

Como se muestra en la Figura 6.29, la función *obtenerProyecto* se encarga de realizar una petición HTTP GET para obtener la información del proyecto a editar.

```
obtenerProyecto(idProyecto:number):Observable<Proyecto>{
  return this.http.get<Proyecto>(this.apiUrl+ '/project/'+ idProyecto);
}
```

Figura 6.29: Petición GET obtenerProyecto

Una vez obtenida la información del proyecto, se muestra por pantalla.

Cuando el organismo haya editado los campos que sean necesarios, se obtienen sus valores y se validan para formar un objeto. Posteriormente, se muestra la alerta emergente de SweetAlert2 para preguntar por última vez si se desea editar el proyecto. (ver Figura 6.30).

```
const nuevoProyecto = {
  id: this.projectId,
  idOrganization: this.userService.getOrgId(),
  title: titulo,
  description: descripcion,
  duration: duracion,
  capacity: capacidad,
  scope: ambito,
  subscope: subambito,
  active: true,
  initLifeDate: "",
  updateLife: ""
}

Swal.fire({
  icon: 'question',
  title: '¿Seguro que quieres editar este proyecto?',
  showCancelButton: true,
  confirmButtonText: 'Editar',
  cancelButtonText: 'Cancelar',
}).then((result) => {
  if (result.isConfirmed) {
    this.organizationService.editarProyecto(this.projectId, nuevoProyecto ).subscribe(
      (data: Proyecto) => {
        Swal.fire({
          icon: 'success',
          title: '¡Proyecto editado con éxito!',
          text: 'El proyecto se ha editado satisfactoriamente.',
          confirmButtonText: 'Vale'
        }).then((result) => {
          if (result.isConfirmed) {
            this.router.navigate(['organization/my-projects']);
          }
        });
      },
      error => {
        console.error('Error al editar el proyecto:', error);
      }
    );
  }
});
```

Figura 6.30: Editar proyecto

Si la respuesta es afirmativa, se realiza una llamada a la función *editarProyecto* presente en el servicio que se encarga de realizar la petición HTTP PUT que edita el proyecto en la base de datos. (ver Figura 6.31).

```
editarProyecto(idProyecto: number, proyecto: Proyecto): Observable<Proyecto> {
  const url = `${this.apiUrl}/project/${idProyecto}`;
  return this.http.put<Proyecto>(url, proyecto);
}
```

Figura 6.31: Petición PUT editarProyecto

Una vez editado el proyecto se redirige al organismo a la página “Mis Proyectos”.

6.2.4.4 Eliminar proyecto

Esta funcionalidad permite al organismo dar de baja un proyecto creado anteriormente.

El acceso a esta funcionalidad se realiza desde la sección “Mis proyectos” en el que se pasa el identificador del proyecto que se desea eliminar.
Se vuelve a utilizar *SweetAlert2* para generar una alerta emergente que pregunte al usuario. (ver Figura 6.32).

```
eliminarProyecto(idProyecto: number) {
  Swal.fire({
    title: '¿Seguro que quieres eliminar este proyecto?',
    icon: 'warning',
    showCancelButton: true,
    confirmButtonText: 'Eliminar',
    cancelButtonText: 'Cancelar',
    confirmButtonColor: '#dc3545',
    cancelButtonColor: '#6c757d',
  }).then((result) => {
    if (result.isConfirmed) {
      this.organizationService.eliminarProyecto(idProyecto).subscribe(
        () => {
          Swal.fire({
            icon: 'success',
            title: '¡Proyecto eliminado con éxito!',
            text: 'El proyecto se ha eliminado satisfactoriamente.',
            confirmButtonText: 'Vale'
          }).then((result) => {
            if (result.isConfirmed) {
              window.location.reload();
            }
          });
        },
        error => {
          console.error('Error al eliminar el proyecto:', error);
          Swal.fire({
            icon: 'error',
            title: '¡Error!',
            text: 'Ha ocurrido un error al eliminar el proyecto',
            confirmButtonText: 'Entendido'
          });
        }
      );
    }
  });
}
```

Figura 6.32: Eliminar proyecto

Si la respuesta es afirmativa, se realiza una llamada a la función *eliminarProyecto* presente en el servicio que se encarga de realizar la petición HTTP DELETE que elimina el proyecto en la base de datos. (ver Figura 6.33).

```
eliminarProyecto(idProyecto:number):Observable<Proyecto>{
  return this.http.delete<Proyecto>(this.apiUrl+ '/project/'+ idProyecto);
}
```

Figura 6.33: Petición DELETE eliminarProyecto

Una vez que el proyecto es eliminado se redirige al organismo a la página “Mis Proyectos”.

6.2.4.5 Buscar científico

Esta funcionalidad permite al organismo buscar un científico para poder ver toda su información.

Para implementarla, se recoge el ORCID escrito por el organismo en el campo habilitado para ello. Se valida si está o no en el formato correcto y se pasa como parámetro a la función *obtenerCientífico* del servicio. (ver Figura 6.34).

```
this.organizationService.obtenerCientifico(orcid).subscribe(
  (response)=> {

    //Resetear el ORCID escrito
    (document.getElementById('orcidCientifico') as HTMLInputElement).value = '';

    // Actualizar los elementos <span> con la información de la publicación
  const orcidSpan = document.getElementById('orcidSpan');
  if (orcidSpan) {
    orcidSpan.innerText = response.orcid;
  }


  const nombreSpan = document.getElementById('nombreSpan');
  if (nombreSpan) {
    nombreSpan.innerText = response.name;
  }

  const profesionSpan = document.getElementById('profesionSpan');
  if (profesionSpan) {
    profesionSpan.innerText = response.profession;
  }

  const emailSpan = document.getElementById('emailSpan');
  if (emailSpan) {
    emailSpan.innerText = response.email;
  }

  const disponibleSpan = document.getElementById('disponibleSpan');
  if ((response.available) && disponibleSpan) {
    disponibleSpan.innerText = "Libre para incorporarse a un proyecto";
  }else if ((!response.available) && disponibleSpan){
    disponibleSpan.innerText = "Ocupado participando en un proyecto";
  }
```

Figura 6.34: Buscar científico

Como se muestra en la Figura 6.35, la función obtenerCientífico se encarga de realizar la petición HTTP GET para obtener el científico.

```
obtenerCientifico(orcid:string):Observable<Cientifico>{
  return this.http.get<Cientifico>(this.apiUrl+ '/scientist/findBy/'+ orcid);
}
```

Figura 6.35: Petición GET obtenerCientifico

Una vez obtenido el científico, se muestra su información por pantalla.

6.2.4.6 Recomendaciones de científicos

Esta funcionalidad permite al organismo recibir recomendaciones de científicos que pueden participar en un proyecto en concreto basándose en el ámbito y subámbito del proyecto seleccionado y en la especialidad de las publicaciones de los científicos y sus profesiones.

Esta funcionalidad requiere que el organismo introduzca el identificador del proyecto para el cual desea recibir recomendaciones.

El identificador es validado y pasado como parámetro a la función *recomendarProyectos* del servicio. (ver Figura 6.36).

```
this.organizationService.recomendarCientificos(idProyecto).subscribe(
  (response: any) => {
    this.encontrados = true;
    this.buscados = true;
    this.cientificos = response.content;

  },
  (error) => {
    console.error('Error al recomendar cientificos:', error);
    this.encontrados = false;
    this.buscados = true;
  }
);
```

Figura 6.36: Recomendaciones de científicos

Como se muestra en la Figura 6.37, la función *recomendarCientificos* se encarga de realizar la petición HTTP GET para obtener los científicos recomendados.

```
recomendarCientificos(idProject:number){
  return this.http.get<any>(this.apiUrl + '/project/recommendation/'+ idProject);
}
```

Figura 6.37: Petición GET recomendarCientificos

Si se obtienen recomendaciones de científicos, se mostrarán en forma de tarjetas al organismo.
En caso contrario, se muestra un mensaje indicando que no se han podido obtener recomendaciones.

6.2.9 Administrador

6.2.9.1 Listar publicaciones

Esta funcionalidad se encarga de mostrar un listado en forma de tabla de todas las publicaciones realizadas por los científicos. En caso de que no exista ninguna publicación se sustituye la tabla por un texto que indica que no existen publicaciones.

El listado de publicaciones se carga automáticamente al entrar en la sección.
Para ello, se realiza una llamada a la función *cargarPublicaciones* mostrada en la Figura 6.38. En esta función simplemente llama a la función *obtenerTodasPublicaciones* del servicio administrador.

```
cargarPublicaciones() {
  this.adminService.obtenerTodasPublicaciones().subscribe(
    (data: any) => {
      this.publicaciones = data.content;
    },
    error => {
      console.error('Error al obtener las publicaciones:', error);
    }
  );
}
```

Figura 6.38: Listar publicaciones

Como se muestra en la Figura 6.39, la función *obtenerTodasPublicaciones* se encarga de realizar la solicitud HTTP GET al backend para obtener todas las publicaciones de los científicos.

```
obtenerTodasPublicaciones(){
  return this.http.get<Publicacion[]>(`${this.apiUrl}/publication/all`);
}
```

Figura 6.39: Petición GET obtenerTodasPublicaciones

6.2.9.2 Editar publicación

Esta funcionalidad permite al administrador realizar modificaciones en los campos de una publicación de un científico. Para poder utilizar esta funcionalidad, deben existir publicaciones que editar. En caso de no existir, se muestra un mensaje por pantalla indicándolo.

El acceso a esta funcionalidad se realiza desde la sección “Listar publicaciones” pasando por la URL el identificador de la publicación que se desea editar.

Para cargar la información de la publicación que se desea editar, se obtiene el identificador pasado como parámetro y se llama a la función *obtenerPublicacion* presente en el servicio (ver Figura 6.40).
Ésta se encarga de realizar una petición HTTP GET para obtener la información de la publicación a editar, como se muestra en la Figura 6.41.

```
const publicationId = params.get('publicationId');

if (publicationId) {
  this.publicationId = parseInt(publicationId);

  this.adminService.obtenerPublicacion(parseInt(publicationId)).subscribe(data => {
    this.tituloPublicacion = data.title;
    this.descripcionPublicacion = data.description;
    this.especialidadPublicacion = data.expertise;
    this.stringExperiencia = data.profExperience;
```

Figura 6.40: Función obtenerPublicacion

```
private apiUrl = 'http://localhost:8080/api/searchProject';

constructor(private http: HttpClient) { }

//Publicaciones
obtenerPublicacion(idPublicacion:number):Observable<Publicacion>{
  return this.http.get<Publicacion>(this.apiUrl+ '/publication/'+idPublicacion);
}
```

Figura 6.41: Petición GET obtenerPublicación

Una vez obtenida la información de la publicación, se muestra por pantalla.

Cuando el administrador haya editado los campos que sean necesarios, se obtienen sus valores y se validan para formar un objeto. Posteriormente, se muestra la alerta emergente de *SweetAlert2* para preguntar por última vez si se desea editar la publicación. (ver Figura 6.42).

```
const nuevaPublicacion = {
  id: this.publicationId,
  title: titulo,
  expertise: especialidad,
  profExperience: profExperience,
  description: descripcion,
  active: true,
  initLifeDate: "",
  updateLife: ""
}


Swal.fire({
  icon: 'question',
  title: '¿Seguro que quieres editar esta publicación?',
  showCancelButton: true,
  confirmButtonText: 'Editar',
  cancelButtonText: 'Cancelar',
}).then((result) => {
  if (result.isConfirmed) {
    this.adminService.editarPublicacion(this.publicationId, nuevaPublicacion ).subscribe(
      (data: Publicacion) => {

        Swal.fire({
          icon: 'success',
          title: '¡Publicación editada con éxito!',
          text: 'La publicación se ha editado satisfactoriamente.',
          confirmButtonText: 'Vale'
        }).then((result) => {
          if (result.isConfirmed) {
            this.router.navigate(['administrator/list-publications']);
          }
        });
      },
      error => {
        console.error('Error al editar la publicacion:', error);
      }
    );
  }
});
```

Figura 6.42: Editar publicación

Si la respuesta es afirmativa, se realiza una llamada a la función *editarPublicacion* presente en el servicio que se encarga de realizar la petición HTTP PUT que edita la publicación en la base de datos. (ver Figura 6.43).

```
editarPublicacion(idPublicacion: number, publicacion: Publicacion): Observable<Publicacion> {
  const url = `${this.apiUrl}/publication/${idPublicacion}`;
  return this.http.put<Publicacion>(url, publicacion);
}
```

Figura 6.43: Petición PUT editarPublicacion

Una vez editada la publicación se redirige al usuario a la página “Listar publicaciones”.

6.2.9.3 Eliminar publicación

Esta funcionalidad permite al administrador dar de baja una publicación.

El acceso a esta funcionalidad se realiza desde la sección “Listar publicaciones” en el que se pasa el identificador de la publicación que se desea eliminar. (ver Figura 6.44).
Se vuelve a utilizar *SweetAlert2* para generar una alerta emergente que pregunte al usuario.

```
eliminarPublicacion(idPublicacion: number) {
  Swal.fire({
    title: '¿Seguro que quieres eliminar esta publicación?',
    icon: 'question',
    showCancelButton: true,
    confirmButtonText: 'Eliminar',
    cancelButtonText: 'Cancelar',
    confirmButtonColor: '#dc3545',
    cancelButtonColor: '#6c757d',
  }).then((result) => {
    if (result.isConfirmed) {
      this.adminService.eliminarPublicacion(idPublicacion).subscribe(
        () => {
          Swal.fire({
            icon: 'success',
            title: '¡Publicación eliminada con éxito!',
            text: 'La publicación se ha eliminado satisfactoriamente.',
            confirmButtonText: 'Vale'
          }).then((result) => {
            if (result.isConfirmed) {
              window.location.reload();
            }
          });
        },
        error => {
          console.error('Error al eliminar la publicación:', error);
          Swal.fire({
            icon: 'error',
            title: '¡Error!',
            text: 'Ha ocurrido un error al eliminar la publicación',
            confirmButtonText: 'Entendido'
          });
        }
      );
    }
  });
}
```

Figura 6.44: Eliminar publicación

Si la respuesta es afirmativa, se realiza una llamada a la función *eliminarPublicacion* presente en el servicio que se encarga de realizar la petición HTTP DELETE que elimina la publicación en la base de datos. (ver Figura 6.45).

```
eliminarPublicacion(idPublicacion:number):Observable<Publicacion>{
  return this.http.delete<Publicacion>(this.apiUrl+ '/publication/'+idPublicacion);
}
```

Figura 6.45: Petición DELETE eliminarPublicacion

Una vez que la publicación es eliminada se redirige al científico a la página “Listar publicaciones”.

6.2.9.4 Reactivar publicación

Esta funcionalidad permite al administrador reactivar una publicación que estaba dada de baja.

El acceso a esta funcionalidad se realiza desde la sección “Listar publicaciones” en el que se pasa el identificador de la publicación que se desea reactivar. (ver Figura 6.46).
Se vuelve a utilizar *SweetAlert2* para generar una alerta emergente que pregunte al usuario.

```
reactivarPublicacion(idPublicacion: number) {
  Swal.fire({
    title: '¿Seguro que quieres reactivar esta publicación?',
    icon: 'question',
    showCancelButton: true,
    confirmButtonText: 'Reactivar',
    cancelButtonText: 'Cancelar',
    confirmButtonColor: '#53B200',
    cancelButtonColor: '#6c757d',
  }).then((result) => {
    if (result.isConfirmed) {
      this.adminService.reactivarPublicacion(idPublicacion).subscribe(
        () => {
          Swal.fire({
            icon: 'success',
            title: '¡Publicación reactivada con éxito!',
            text: 'La publicación se ha reactivado satisfactoriamente.',
            confirmButtonText: 'Vale'
          }).then((result) => {
            if (result.isConfirmed) {
              window.location.reload();
            }
          });
        },
        error => {
          console.error('Error al reactivar la publicación:', error);
          Swal.fire({
            icon: 'error',
            title: '¡Error!',
            text: 'Ha ocurrido un error al reactivar la publicación',
            confirmButtonText: 'Entendido'
          });
        }
      );
    }
  });
}
```

Figura 6.46: Reactivar publicación

Si la respuesta es afirmativa, se realiza una llamada a la función *reactivarPublicacion* presente en el servicio que se encarga de realizar la petición HTTP GET que reactiva la publicación en la base de datos. (ver Figura 6.47).

```
reactivarPublicacion(idPublicacion:number):Observable<Publicacion>{
  return this.http.get<Publicacion>(this.apiUrl+ '/publication/reactivate/'+ idPublicacion);
}
```

Figura 6.47: Petición GET reactivarPublicacion

Una vez que la publicación es reactivada se redirige al administrador a la página “Listar publicaciones”.

6.2.9.5 Listar usuarios

Esta funcionalidad se encarga de mostrar un listado en forma de tabla de todos los científicos registrados en la aplicación y otro listado de todos los organismos. En caso de que no exista ningún usuario se sustituye la tabla correspondiente por un texto que indica que no existen usuarios.

Los listados de científicos y organismos se cargan automáticamente al entrar en la sección.
Para ello, se realiza una llamada a la función *cargarUsuarios* mostrada en la Figura 6.48. En esta función llama a las funciones *obtenerTodosCientificos* y *obtenerTodosOrganismos* del servicio administrador.

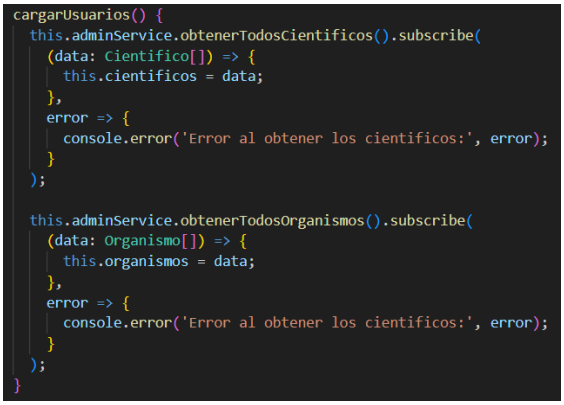

Figura 6.48: Función cargarUsuarios

Como se muestra en la Figura 6.49, la función *obtenerTodosCientificos* se encarga de realizar la solicitud HTTP GET al backend para obtener todos los científicos.

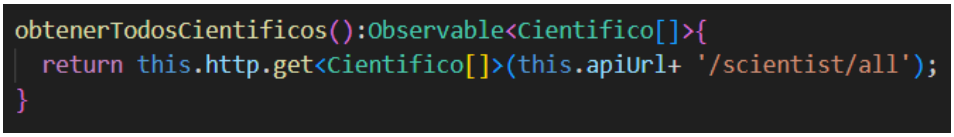

Figura 6.49: Petición GET obtenerTodosCientificos

En la Figura 6.50, se puede observar que la función *obtenerTodosOrganismos* se encarga de realizar la solicitud HTTP GET al backend para obtener todos los organismos.

```
obtenerTodosOrganismos():Observable<Organismo[]>{
  return this.http.get<Organismo[]>(this.apiUrl+ '/organization/all');
}
```

Figura 6.50: Petición GET obtenerTodosOrganismos

6.2.9.6 Editar usuario

Esta funcionalidad permite al administrador realizar modificaciones en los atributos de un científico u organismo. En caso de no existir, se muestra un mensaje por pantalla indicándolo.

El acceso a esta funcionalidad se realiza desde la sección “Listar usuarios” pasando por la URL el identificador del usuario (científico u organismo) que se desea editar.

Para cargar la información del usuario que se desea editar, se obtiene el identificador pasado como parámetro y se llama, o bien a la función *obtenerCientifico*, o bien a la función *obtenerOrganismo* presentes en el servicio (ver Figura 6.51).

```
this.route.paramMap.subscribe(params => {

  const type = params.get('type');
  const userId = params.get('id');


  if (type == "scientist" && userId) {
    this.esCientifico = true;
    this.userId = parseInt(userId);

    this.adminService.obtenerCientifico(parseInt(userId)).subscribe(data => {

      this.orcid = data.orcid;
      this.userUuid = data.userUuid;
      this.name = data.name;
      this.email = data.email;
      this.profession = data.profession;
      this.available = data.available;
      this.active = data.active;

    });
  }else if(type == "organization" && userId){
    this.esCientifico = false;
    this.userId = parseInt(userId);

    this.adminService.obtenerOrganismo(parseInt(userId)).subscribe(data => {

      this.idOrganization = data.idOrganization.toString();
      this.userUuid = data.userUuid;
      this.name = data.name;
      this.email = data.email;
      this.location = data.location;
      this.area = data.area;
      this.active = data.active;
    });
  }
});
```

Figura 6.51: Obtener usuario

Como se muestra en la Figura 6.52, la función *obtenerCientifico* se encarga de realizar la petición HTTP GET al backend para obtener los datos del científico.

```
obtenerCientifico(id:number):Observable<Cientifico>{
  return this.http.get<Cientifico>(this.apiUrl + '/scientist/' + id);
}
```

Figura 6.52: Petición GET obtenerCientifico

En la Figura 6.53 se muestra que la función *obtenerOrganismo* se encarga de realizar la petición HTTP GET al backend para obtener los datos del organismo.

```
obtenerOrganismo(id:number):Observable<Organismo>{
  return this.http.get<Organismo>(this.apiUrl + '/organization/' + id);
}
```

Figura 6.53: Petición GET obtenerOrganismo

Una vez obtenida la información del usuario, se muestra por pantalla.

Cuando el administrador haya editado los campos que sean necesarios, se obtienen sus valores y se validan para formar un objeto. Posteriormente, se muestra la alerta emergente de *SweetAlert2* para preguntar por última vez si se desea editar el usuario. (ver Figura 6.54 para el científico y la Figura 6.55 para el organismo).

```
const nuevoCientifico = {
  id: this.userId,
  orcid: this.orcid,
  userUuid: this.userUuid,
  name: nombre,
  email: this.email,
  profession: profesion,
  available: this.available,
  active: this.active
}

Swal.fire({
  icon: 'question',
  title: '¿Seguro que quieres editar este científico?',
  showCancelButton: true,
  confirmButtonText: 'Editar',
  cancelButtonText: 'Cancelar',
}).then((result) => {
  if (result.isConfirmed) {
    this.adminService.editarCientifico(this.userId, nuevoCientifico ).subscribe(
      (data: any) => {

        Swal.fire({
          icon: 'success',
          title: '¡Científico editado con éxito!',
          text: 'El científico se ha editado satisfactoriamente.',
          confirmButtonText: 'Vale'
        }).then((result) => {
          if (result.isConfirmed) {
            this.router.navigate(['administrator/list-users']);
          }
        });
      },
      error => {
        console.error('Error al editar el científico:', error);
      }
    );
  }
});
```

Figura 6.54: Editar científico

```
const nuevoOrganismo = {
  id: this.userId,
  idOrganization: parseInt(this.idOrganization),
  userUuid: this.userUuid,
  name: nombre,
  email: this.email,
  location: localidad,
  area: this.area,
  active: this.active
}

Swal.fire({
  icon: 'question',
  title: '¿Seguro que quieres editar este organismo?',
  showCancelButton: true,
  confirmButtonText: 'Editar',
  cancelButtonText: 'Cancelar',
}).then((result) => {
  if (result.isConfirmed) {
    this.adminService.editarOrganismo(this.userId, nuevoOrganismo ).subscribe(
      (data: any) => {

        Swal.fire({
          icon: 'success',
          title: '¡Organismo editado con éxito!',
          text: 'El organismo se ha editado satisfactoriamente.',
          confirmButtonText: 'Vale'
        }).then((result) => {
          if (result.isConfirmed) {
            this.router.navigate(['administrator/list-users']);
          }
        });
      },
      error => {
        console.error('Error al editar el organismo:', error);
      }
    );
  }
});
```

Figura 6.55: Editar organismo

Si la respuesta es afirmativa, se realiza una llamada a la función *editarCientifico* o a la función *editarOrganismo* presentes en el servicio que se encargan de realizar la petición HTTP PUT que edita el científico o el organismo en la base de datos. (ver Figuras 6.56 y 6.57).

```
editarCientifico(idCientifico: number, cientifico: Cientifico):Observable<Cientifico>{
  const url = `${this.apiUrl}/scientist/${idCientifico}`;
  return this.http.put<Cientifico>(url, cientifico);
}
```

Figura 6.56: Petición PUT editarCientifico

```
editarOrganismo(idOrganismo: number, organismo: Organismo):Observable<Organismo>{
  const url = `${this.apiUrl}/organization/${idOrganismo}`;
  return this.http.put<Organismo>(url, organismo);
}
```

Figura 6.57: Petición PUT editarOrganismo

Una vez editado el usuario se redirige al administrador a la página “Listar usuarios”.

6.2.9.7 Eliminar usuario

Esta funcionalidad permite al administrador dar de baja un usuario (científico u organismo) de forma lógica en la aplicación.

El acceso a esta funcionalidad se realiza desde la sección “Listar usuario” en el que pasa el identificador del usuario que se desea eliminar. (ver Figura 6.58 para el científico y Figura 6.59 para el organismo).

Se vuelve a utilizar *SweetAlert2* para generar una alerta emergente que pregunte al usuario.

```
eliminarCientifico(orcid: string) {
  Swal.fire({
    title: '¿Seguro que quieres eliminar este científico?',
    icon: 'question',
    showCancelButton: true,
    confirmButtonText: 'Eliminar',
    cancelButtonText: 'Cancelar',
    confirmButtonColor: '#dc3545',
    cancelButtonColor: '#6c757d',
  }).then((result) => {
    if (result.isConfirmed) {
      this.adminService.eliminarCientifico(orcid).subscribe(
        () => {
          Swal.fire({
            icon: 'success',
            title: '¡Científico eliminado con éxito!',
            text: 'El científico se ha eliminado satisfactoriamente.',
            confirmButtonText: 'Vale'
          }).then((result) => {
            if (result.isConfirmed) {
              window.location.reload();
            }
          });
        },
        error => {
          console.error('Error al eliminar el científico:', error);
          Swal.fire({
            icon: 'error',
            title: '¡Error!',
            text: 'Ha ocurrido un error al eliminar el científico',
            confirmButtonText: 'Entendido'
          });
        }
      );
    }
  });
}
```

Figura 6.58: Eliminar científico

Figura 6.59: Eliminar organismo

Si la respuesta es afirmativa, se realiza una llamada a la función *eliminarCientifico* o a la función *eliminarOrganismo* presentes en el servicio que se encargan de realizar la petición HTTP DELETE que elimina al científico u organismo de forma lógica en la base de datos. (ver Figuras 6.60 y 6.61).

```
eliminarCientifico(orcid: string):Observable<Cientifico>{
  return this.http.delete<Cientifico>(this.apiUrl + '/scientist/'+ orcid);
}
```

Figura 6.60: Petición DELETE eliminarCientifico

```
eliminarOrganismo(idOrganismo: number):Observable<Organismo>{
  return this.http.delete<Organismo>(this.apiUrl + '/organization/'+ idOrganismo);
}
```

Figura 6.61: Petición DELETE eliminarOrganismo

Una vez que el usuario es eliminado se redirige al administrador a la página “Listar usuarios”.

6.2.9.8 Reactivar usuario

Esta funcionalidad permite al administrador reactivar un usuario (científico u organismo) que estaba dado de baja.

El acceso a esta funcionalidad se realiza desde la sección “Listar usuarios” en el que pasa el identificador del usuario que se desea reactivar. (ver Figuras 6.62 para el científico y Figura 6.63 para el organismo).

Se vuelve a utilizar *SweetAlert2* para generar una alerta emergente que pregunte al usuario.

```
reactivarCientifico(orcid: string) {
  Swal.fire({
    title: '¿Seguro que quieres reactivar este científico?',
    icon: 'question',
    showCancelButton: true,
    confirmButtonText: 'Reactivar',
    cancelButtonText: 'Cancelar',
    confirmButtonColor: '#53B200',
    cancelButtonColor: '#6c757d',
  }).then((result) => {
    if (result.isConfirmed) {
      this.adminService.reactivarCientifico(orcid).subscribe(
        () => {
          Swal.fire({
            icon: 'success',
            title: '¡Científico reactivado con éxito!',
            text: 'El científico se ha reactivado satisfactoriamente.',
            confirmButtonText: 'Vale'
          }).then((result) => {
            if (result.isConfirmed) {
              window.location.reload();
            }
          });
        },
        error => {
          console.error('Error al reactivar el científico:', error);
          Swal.fire({
            icon: 'error',
            title: '¡Error!',
            text: 'Ha ocurrido un error al reactivar el científico',
            confirmButtonText: 'Entendido'
          });
        }
      );
    }
  });
}
```

Figura 6.62: Reactivar científico

```
reactivarOrganismo(idOrganization: string) {
  Swal.fire({
    title: '¿Seguro que quieres reactivar este organismo?',
    icon: 'question',
    showCancelButton: true,
    confirmButtonText: 'Reactivar',
    cancelButtonText: 'Cancelar',
    confirmButtonColor: '#53B200',
    cancelButtonColor: '#6c757d',
  }).then((result) => {
    if (result.isConfirmed) {
      this.adminService.reactivarOrganismo(idOrganization).subscribe(
        () => {
          Swal.fire({
            icon: 'success',
            title: '¡Organismo reactivado con éxito!',
            text: 'El organismo se ha reactivado satisfactoriamente.',
            confirmButtonText: 'Vale'
          }).then((result) => {
            if (result.isConfirmed) {
              window.location.reload();
            }
          });
        },
        error => {
          console.error('Error al reactivar el organismo:', error);
          Swal.fire({
            icon: 'error',
            title: '¡Error!',
            text: 'Ha ocurrido un error al reactivar el organismo',
            confirmButtonText: 'Entendido'
          });
        }
      );
    }
  });
}
```

Figura 6.63: Reactivar organismo

Si la respuesta es afirmativa, se realiza una llamada a la función *reactivarCientifico* o a la función *reactivarOrganismo* presentes en el servicio que se encargan de realizar la petición HTTP GET que reactiva al científico u organismo en la base de datos. (ver Figuras 6.64 y 6.65).

```
reactivarCientifico(orcid: string):Observable<Cientifico>{
  return this.http.get<Cientifico>(this.apiUrl+ '/scientist/reactivate/'+ orcid);
}
```

Figura 6.64: Petición GET reactivarCientifico

```
reactivarOrganismo(idProyecto: string):Observable<Organismo>{
  return this.http.get<Organismo>(this.apiUrl+ '/organization/reactivate/'+ idProyecto);
}
```

Figura 6.65: Petición GET reactivarOrganismo

Una vez que el usuario es reactivado se redirige al administrador a la página “Listar usuarios”.

6.2.9.9 Listar proyectos

Esta funcionalidad se encarga de mostrar un listado en forma de tabla de todos los proyectos realizados por los organismos. En caso de que no exista ningún proyecto se sustituye la tabla por un texto que indica que no existen proyectos.

El listado de proyectos se carga automáticamente al entrar en la sección.
Para ello, se realiza una llamada a la función *cargarProyectos* mostrada en la Figura 6.66. En esta función simplemente llama a la función *obtenerTodosProyectos* del servicio administrador.

```
cargarProyectos() {
  this.adminService.obtenerTodosProyectos().subscribe(
    (data: any) => {
      this.proyectos = data.content;
    },
    error => {
      console.error('Error al obtener los proyectos:', error);
    }
  );
}
```

Figura 6.66: Listar proyectos

Como se muestra en la Figura 6.67, la función *obtenerTodosProyectos* se encarga de realizar la solicitud HTTP GET para obtener todos los proyectos de los organismos.

```
obtenerTodosProyectos():Observable<Proyecto[]>{
  return this.http.get<Proyecto[]>(this.apiUrl + '/project/all/');
}
```

Figura 6.67: Petición GET obtenerTodosProyectos

6.2.9.10 Editar proyecto

Esta funcionalidad permite al administrador realizar modificaciones en los campos de un proyecto de un organismo. Para poder utilizar esta funcionalidad, deben existir proyectos que editar. En caso de no existir, se muestra un mensaje por pantalla indicándolo.

El acceso a esta funcionalidad se realiza desde la sección “Listar proyectos” pasando por la URL el identificador del proyecto que se desea editar.

Para cargar la información del proyecto que se desea editar, se obtiene el identificador pasado como parámetro y se llama a la función *obtenerProyecto* presente en el servicio (ver Figura 6.68).
Ésta se encarga de realizar una petición HTTP GET para obtener la información del proyecto a editar, como se muestra en la Figura 6.69.

```
const projectId = params.get('projectId');

if (projectId) {
  this.projectId = parseInt(projectId);

  this.adminService.obtenerProyecto(parseInt(projectId)).subscribe(data => {

    this.tituloProyecto = data.title;
    this.idOrganization = data.idOrganization;
    this.descripcionProyecto = data.description;
    this.size = data.size;
    this.capacidadProyecto = data.capacity.toString();
    this.ambitoProyecto = data.scope;
    this.subambitoProyecto = data.subscope;
    this.full = data.full;
    this.stringDuracion = data.duration;
```

Figura 6.68: Función obtenerProyecto

```
obtenerProyecto(idProyecto:number):Observable<Proyecto>{
  return this.http.get<Proyecto>(this.apiUrl+ '/project/'+ idProyecto);
}
```

Figura 6.69: Petición GET obtenerProyecto

Una vez obtenida la información del proyecto, se muestra por pantalla.

Cuando el administrador haya editado los campos que sean necesarios, se obtienen sus valores y se validan para formar un objeto. Posteriormente, se muestra la alerta emergente de *SweetAlert2* para preguntar por última vez si se desea editar el proyecto. (ver Figura 6.70).

```
const nuevoProyecto = {
  id: this.projectId,
  idOrganization: this.idOrganization,
  title: titulo,
  description: descripcion,
  size: this.size,
  duration: duracion,
  capacity: parseInt(capacidad),
  scope: ambito,
  subscope: subambito,
  full: this.full,
  active: true,
  initLifeDate: "",
  updateLife: ""
}

Swal.fire({
  icon: 'question',
  title: '¿Seguro que quieres editar este proyecto?',
  showCancelButton: true,
  confirmButtonText: 'Editar',
  cancelButtonText: 'Cancelar',
}).then((result) => {
  if (result.isConfirmed) {
    this.adminService.editarProyecto(this.projectId, nuevoProyecto ).subscribe(
      (data: Proyecto) => {

        Swal.fire({
          icon: 'success',
          title: '¡Proyecto editado con éxito!',
          text: 'El proyecto se ha editado satisfactoriamente.',
          confirmButtonText: 'Vale'
        }).then((result) => {
          if (result.isConfirmed) {
            this.router.navigate(['administrator/list-projects']);
          }
        });
      },
      error => {
        console.error('Error al editar el proyecto:', error);
      }
```

Figura 6.70: Editar proyecto

Si la respuesta es afirmativa, se realiza una llamada a la función *editarProyecto* presente en el servicio que se encarga de realizar la petición HTTP PUT que edita el proyecto en la base de datos. (ver Figura 6.71).

```
editarProyecto(idProyecto: number, proyecto: Proyecto): Observable<Proyecto> {
  const url = `${this.apiUrl}/project/${idProyecto}`;
  return this.http.put<Proyecto>(url, proyecto);
}
```

Figura 6.71: Petición PUT editarProyecto

Una vez editado el proyecto se redirige al usuario a la página “Listar proyectos”.

6.2.9.11 Eliminar proyecto

Esta funcionalidad permite al administrador dar de baja un proyecto de forma lógica en la aplicación.

El acceso a esta funcionalidad se realiza desde la sección “Listar proyectos” en el que se pasa el identificador del proyecto que se desea eliminar. (ver Figura 6.72).
Se vuelve a utilizar *SweetAlert2* para generar una alerta emergente que pregunte al usuario.

```
eliminarProyecto(idPublicacion: number) {
  Swal.fire({
    title: '¿Seguro que quieres eliminar este proyecto?',
    icon: 'question',
    showCancelButton: true,
    confirmButtonText: 'Eliminar',
    cancelButtonText: 'Cancelar',
    confirmButtonColor: '#dc3545',
    cancelButtonColor: '#6c757d',
  }).then((result) => {
    if (result.isConfirmed) {
      this.adminService.eliminarProyecto(idPublicacion).subscribe(
        () => {
          Swal.fire({
            icon: 'success',
            title: '¡Proyecto eliminado con éxito!',
            text: 'El proyecto se ha eliminado satisfactoriamente.',
            confirmButtonText: 'Vale'
          }).then((result) => {
            if (result.isConfirmed) {
              window.location.reload();
            }
          });
        },
        error => {
          console.error('Error al eliminar el proyecto:', error);
          Swal.fire({
            icon: 'error',
            title: '¡Error!',
            text: 'Ha ocurrido un error al eliminar el proyecto',
            confirmButtonText: 'Entendido'
          });
        }
      );
    }
  });
}
```

Figura 6.72: Eliminar proyecto

Si la respuesta es afirmativa, se realiza una llamada a la función *eliminarProyecto* presente en el servicio que se encarga de realizar la petición HTTP DELETE que elimina el proyecto de forma lógica en la base de datos. (ver Figura 6.73).

```
eliminarProyecto(idProyecto:number):Observable<Proyecto>{
  return this.http.delete<Proyecto>(this.apiUrl+ '/project/'+ idProyecto);
}
```

Figura 6.73: Petición DELETE eliminarProyecto

Una vez que el proyecto es eliminado se redirige al administrador a la página “Listar proyectos”.

6.2.9.12 Reactivar proyecto

Esta funcionalidad permite al administrador reactivar un proyecto que estaba dado de baja.

El acceso a esta funcionalidad se realiza desde la sección “Listar proyectos” en el que se pasa el identificador del proyecto que se desea reactivar. (ver Figura 6.74).
Se vuelve a utilizar *SweetAlert2* para generar una alerta emergente que pregunte al usuario.

Figura 6.74: Reactivar proyecto

Si la respuesta es afirmativa, se realiza una llamada a la función *reactivarProyecto* presente en el servicio que se encarga de realizar la petición HTTP GET que reactiva el proyecto en la base de datos. (ver Figura 6.75).

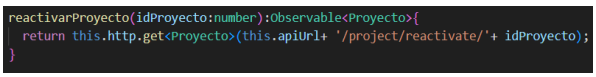

Figura 6.75: Petición GET reactivarProyecto

Una vez que el proyecto es reactivado se redirige al administrador a la página “Listar proyectos”.

6.3 API REST

En este apartado se especificará el diseño de la API Rest implementada. Éstas funcionalidades se pueden dividir por módulos. Además de seguir unas buenas prácticas para diseñar una API RESTful, se ha documentado el desarrollo de la API con el framework de *Swagger (OpenAPI 3*). Éste framework es de código abierto y respaldado por un gran ecosistema de herramientas que nos ayudan a diseñar, construir, documentar y consumir un servicio RESTful.

De esta manera, en conjunto, la interfaz que presenta Swagger al usuario con las API Rest expuestas se muestra en la siguiente Figura 6.76.

Figura 6.76: API del Controlador de Usuarios

En la lista anterior se exponen los endpoints relativos a los *usuarios* de la aplicación con las respectivas operaciones CRUD.

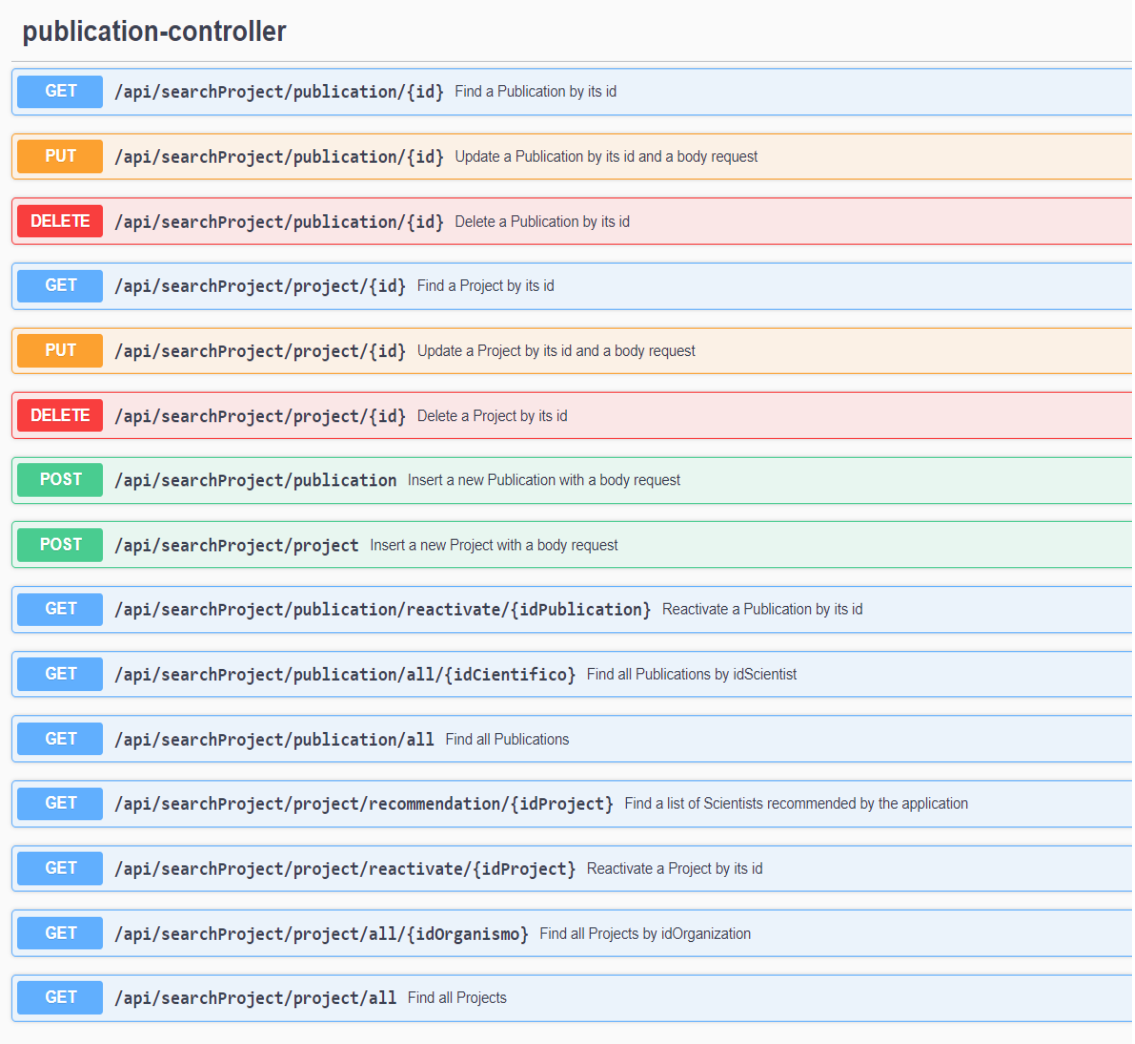

Figura 6.77: API del Controlador de Publicaciones/Proyectos

A su vez en la Figura 6.77 se exponen los endpoints de los servicios relativos a los Proyecto y Publicaciones entendiéndolo como entidades.

A continuación se describen en las siguientes tablas, divididos por módulos, la relación de los endpoints y las operaciones CRUD que se pueden realizar mediante métodos HTTP.

Dependiendo del método HTTP utilizado en cada endpoint se invoca una operación u otra. El método GET se puede utilizar en un recurso individual o un conjunto de recursos, en el primer caso se realiza una operación de lectura (read), y en el segundo una operación de listado (list).

El método POST (write) es la operación de creación de recursos y se puede invocar sobre un recurso individual. Por otro lado, el método PUT (update) sustituye o modifica un recurso por otro. Y el método DELETE es la operación de borrado que permite eliminar los recursos individuales.

6.3.1 Registro

Los endpoints disponibles para el módulo Registro se muestran en la siguiente Tabla 6.1.

Descripción	Result	Endpoint
Método de inserción (POST) en el que un usuario se registra en la aplicación	Long	*api/searchProject/user*
Método de consulta (GET) en el que un usuario inicia sesión dado un uuid	Object	*api/searchProject/login/{uuidUser}*

Tabla 6.1: Endpoint API Registro

6.3.2 Científico

En la Tabla 6.2 se muestran la lista de endpoints expuestos en el módulo de Científico.

Descripción	Result	Endpoint
Método de consulta (GET) en el que se busca a un científico por su clave primaria	Object	*api/searchProject/scientist/findBy/{orcid}*
Método de consulta (GET) en el que se busca a un científico por la clave autoincremental	Object	*api/searchProject/scientist/{id}*

Método de inserción (POST) en el que se da de alta a un científico	Long	*api/searchProject/scientist*
Método de actualización (PUT) en el que se actualizan los datos de un científico dado la clave primaria	Object	*api/searchProject/scientist/{id}*
Método de consulta (GET) en el que se busca a todos los científicos	List	*api/searchProject/scientist/all*
Método de borrado (DELETE) en el que se da de baja a un científico dado su clave primaria	Boolean	*api/searchProject/scientist/{orcid}*
Método de consulta (GET) en el que se comprueba si un científico existe dado su clave primaria	Boolean	*api/searchProject/scientist/isExists/{orcid}*
Método de modificación en el que se busca a un científico para reactivar su estado dado su clave primaria	Boolean	*api/searchProject/scientist/reactivate/{idScientist}*
Método de modificación en el que se asigna un científico a un proyecto dado la clave primaria del científico	Boolean	*api/searchProject/scientist/assignment/{orcid}*
Método de consulta (GET) en el que se busca una lista de proyectos	List	*api/searchProject/scientist/recommendation/{orcid}*

recomendados para un científico dado su clave primaria		
Método de consulta (GET) en el que se busca el proyecto en el que participa un científico dado su clave primaria	Object	*api/searchProject/scientist/project/{orcid}*
Método de modificación en el que un científico se desapunta de un proyecto dado la clave primaria del científico	Boolean	*api/searchProject/scientist/assignment/getOut/{orcid}*

Tabla 6.2: Endpoints API Científicos

6.3.3 Organismo

En éste apartado se muestran en la Tabla 6.3 los endpoints expuestos para el módulo de Organismo.

Descripción	**Result**	**Endpoint**
Método de consulta (GET) en el que se busca a un organismo por su clave autoincremental	Object	*api/searchProject/organization/{id}*
Método de consulta (GET) en el que se busca a un organismo por su clave primaria	Object	*api/searchProject/findBy/{idOrganization}*
Método de inserción (POST) en el que se da de alta a un organismo	Long	*api/searchProject/organization*

Método de actualización (PUT) en el que se actualizan los datos de un organismo dado la clave primaria	Oject	*api/searchProject/organization/{id}*
Método de consulta (GET) en el que se busca a todos los organismos	List	*api/searchProject/organization/all*
Método de consulta (GET) en el que se busca a organismos por el nombre que contengan la palabra introducida	List	*api/searchProject/organization/findByName/{nameOrg}*
Método de borrado (DELETE) en el que se da de baja a un organismo dado su clave primaria	Boolean	*api/searchProject/organization/{idOrganization}*
Método de consulta (GET) en el que se comprueba si un organismo existe dado su clave primaria	Boolean	*api/searchProject/organization/isExists/{idOrganization}*
Método de modificación en el que se busca a un organismo para reactivar su estado dado su clave primaria	Boolean	*api/searchProject/organization/reactivate/{idOrganization}*

Tabla 6.3: Endpoints API Organismos

6.3.4 Publicación

Los endpoint disponibles para el módulo de Publicación se muestran en la siguiente Tabla 6.4.

Descripción	**Result**	**Endpoint**

Método de consulta (GET) en el que se busca una publicación por su clave primaria	Object	*api/searchProject/publication/{id}*
Método de inserción (POST) en el que se crea una publicación	Object	*api/searchProject/publication*
Método de actualización (PUT) en el que se actualizan los datos de una publicación dado su clave primaria		*api/searchProject/publication/{id}*
Método de consulta (GET) en el que se busca todas las publicaciones que tiene un científico dado la clave primaria de éste	List	*api/searchProject/publication/all/{idCientifico}*
Método de consulta (GET) en el que se busca todas las publicaciones	List	*api/searchProject/publication/all*
Método de borrado (DELETE) en el que se da de baja a una publicación dado su clave primaria	Boolean	*api/searchProject/publication/{id}*
Método de modificación en el que se busca una publicación para reactivar su estado dado su clave primaria	Boolean	*api/searchProject/publication/reactivate/{idPublication}*

Tabla 6.4: Endpoints API Publicaciones

6.3.5 Proyectos

En éste apartado se muestran en la Tabla 6.5 los endpoints relativos al módulo de proyecto.

Descripción	Result	Endpoint
Método de consulta (GET) en el que se busca un proyecto por su clave primaria	Object	*api/searchProject/project/{id}*
Método de inserción (POST) en el que se crea un proyecto	Object	*api/searchProject/project*
Método de actualización (PUT) en el que se actualizan los datos de un proyecto dado su clave primaria	Object	*api/searchProject/project/{id}*
Método de consulta (GET) en el que se busca todos los proyectos que tiene un organismo dado la clave primaria de éste	List	*api/searchProject/project/all/{idOrganismo}*
Método de borrado (DELETE) en el que se da de baja a un proyecto dado su clave primaria	Boolean	*api/searchProject/project/{id}*
Método de consulta (GET) en el que se busca todos los proyectos	List	*api/searchProject/project/all*

Método de modificación en el que se busca un proyecto para reactivar su estado dado su clave primaria	Boolean	*api/searchProject/project/reactivate/{idProject}*
Método de consulta (GET) en el que se busca una lista de científicos recomendados para el proyecto de un organismo dado su clave primaria	List	*api/searchProject/project/recommendation/{idProject}*

Tabla 6.5: Endpoint API Proyectos

Capítulo 7 - Conclusiones y Trabajo futuro

7.1 Conclusiones

En este trabajo se ha logrado desarrollar una aplicación web para la gestión de búsquedas de científicos y proyectos. A través de esta aplicación, los científicos son capaces de poder explorar una amplia gama de proyectos que se alineen con sus intereses y conocimientos. Del mismo modo, los organismos pueden difundir sus proyectos y encontrar científicos cualificados para llevarlos a cabo.

Esta aplicación ha conseguido establecer, no sólo una plataforma tecnológica, sino un puente entre la comunidad científica y los proyectos de investigación. Ha sido una demostración de que la tecnología y la ciencia pueden trabajar mano a mano para desarrollar soluciones efectivas a los complejos problemas de nuestros tiempos. El desarrollo de esta aplicación ha supuesto una contribución para fomentar la colaboración interdisciplinar y promover e impulsar la innovación y el avance en las investigaciones científicas.

Por ello se han aplicado conocimientos en ingeniería del software, bases de datos, programación y otros conceptos para diseñar y desarrollar una aplicación que satisfaga las necesidades de la comunidad científica. Se puede destacar que en la parte de ingeniería del software se pasaron por las cuatro etapas (ciclo de vida del software):

La etapa de la *concepción* para determinar el alcance del proyecto y tener una versión inicial del modelo de negocio. Esto queda reflejado en las propuestas iniciales de proyecto de TFG presentadas a nuestros directores.

La etapa de *elaboración* con la planificación del proyecto y el desarrollo de las especificaciones funcionales del proyecto. Así se refleja en la versión inicial de los hitos del proyecto y el documento SRS presentados a los directores.

La etapa de *construcción* en el que elaboramos todas las características técnicas del producto, reflejado en el desarrollo backend y frontend.

Y finalmente la etapa de *transición* para la entrega del producto terminado. Así culmina con la subida del proyecto en un repositorio git con su documentación y versión final.

Por otro lado, en el desarrollo de la base de datos se han tenido en cuenta los conceptos aprendidos a la hora de elegir un modelo de datos relacional que sea capaz de mostrarnos, a través de un conjunto de tablas, la relación que hay entre los principales módulos.

Así también los conceptos de programación orientada a objetos se han aplicado en la elección del diseño de software (modelo basado en la arquitectura en capas), también se ven reflejados en las buenas prácticas a la hora de implementar código; el formato y tamaño del código de desarrollo (número de líneas de código), el empaquetado y división de las clases e interfaces para tener una mejor organización y para que el código sea mantenible.

Por último, en pro de la transparencia y el acceso abierto a la información, se proporcionan los enlaces a los repositorios Github donde se encuentra el código fuente de la aplicación para que, aquel que esté interesado, pueda consultarlos.

- Front-end: https://github.com/aherre08/Science-Hub-FrontEnd.git
- Backend: https://github.com/GeraldLima/ms-busqueda-proyecto.git

7.2 Trabajo futuro

A pesar de que se han conseguido implementar la gran mayoría de las funcionalidades propuestas inicialmente, se pueden plantear varias líneas de trabajo futuro:

- Mejorar el algoritmo de recomendaciones: En la aplicación el sistema de recomendaciones es funcional pero poco avanzado. Esta línea de trabajo futuro se podría centrar en seguir desarrollando el algoritmo de recomendación. Por ejemplo, se podrían incorporar sistemas de recomendación basados en el comportamiento del usuario utilizando algoritmos de aprendizaje automático, con la finalidad de ofrecer recomendaciones más precisas.

- Integración de herramientas de colaboración: Esta línea consistiría en agregar funcionalidades que faciliten la colaboración en línea, como la compartición de documentos o sistemas de videoconferencia. Estos dos sistemas mencionados podrían ser utilizados, por ejemplo, para poder firmar y enviar contratos de los proyectos o para realizar reuniones del proyecto en el que se esté participando.

- Funcionalidad de generar un equipo de trabajo: Esta línea se centraría en desarrollar una funcionalidad que permitiese generar equipos de trabajo para un proyecto. Se indicaría el número de integrantes necesario, sus perfiles y la aplicación formaría automáticamente un equipo de trabajo.

- Mejoras y correcciones en las funcionalidades actuales: Esta línea se centraría en mejorar las funcionalidades ya implementadas en la aplicación, como por ejemplo, añadir la posibilidad de filtrado de información en los listados. Además se centraría en la corrección de errores, optimización de la aplicación y eliminación de aquellas funcionalidades obsoletas.

- Análisis de impacto: En esta línea se podrían desarrollar métricas para evaluar el impacto de las publicaciones y de los proyectos. Con estas métricas, se podrían obtener gráficas y realizar estudios y comparaciones.

- Colaboración con instituciones científicas: Esta línea consistiría en establecer asociaciones con instituciones científicas, laboratorios de investigación y universidades para publicar sus proyectos y promover la colaboración científica.

- Integración de financiamiento: Esta línea se centraría en integrar funcionalidades para que los organismos puedan buscar financiación y subvenciones para sus proyectos.

- Remuneración: Esta línea trataría la idea de remunerar a los organismos por publicar proyectos en la aplicación para que actúe como foco de interés para los científicos.

Chapter 7 - Conclusions and future work

7.1 Conclusions

In this work it has been possible to develop a web application for the management of searches of scientists and projects. Through this app, scientists are able to explore a wide range of projects that align with their interests and knowledge. In the same way, agencies can disseminate their projects and find qualified scientists to carry them out.

This application has managed to establish, not only a technological platform, but a bridge between the scientific community and research projects. It has been a demonstration that technology and science can work hand in hand to develop effective solutions to the complex problems of our times. The development of this application has been a contribution to foster interdisciplinary collaboration and promote and boost innovation and progress in scientific research.

Knowledge in software engineering, databases, programming and other concepts have been applied to design and develop an application that meets the needs of the scientific community. It can be noted that in the software engineering part the project went through the four stages (software life cycle):

The *conception* stage to determine the scope of the project and have an initial version of the business model. This is reflected in the initial TFG project proposals presented to our directors.

The *elaboration* stage with the planning of the project and the development of the functional specifications of the project. This is reflected in the initial version of the project milestones and the SRS document presented to managers.

The *construction* stage in which we elaborate all the technical characteristics of the product, reflected in the backend and frontend development.

And finally the *transition* stage for the delivery of the finished product. This culminates with the upload of the project in a git repository with its documentation and final version.

On the other hand, in the development of the database we have taken into account the concepts learned when choosing a relational data model that is capable of showing us, through a set of tables, the relationship between the main modules.

Likewise, the concepts of object-oriented programming we have been applied in the choice of software design (model based on layered architecture), they are also reflected in good practices when we have implemented code; the format and size of the development code (number of line of code), the packaging and division of the classes and interfaces in order to have a better organization and to make the code maintainable.

Finally, for the sake of transparency and open access to information, links are provided to the Github repositories where the source code of the application is located so that anyone who is interested can consult them.

- Front-end: https://github.com/aherre08/Science-Hub-FrontEnd.git
- Backend: https://github.com/GeraldLima/ms-busqueda-proyecto.git

7.2 Future work

Although the vast majority of the functionalities initially proposed have been implemented, several lines of future work can be proposed:

- Improve the recommendation algorithm: In the application the recommendation system is functional but not very advanced. This future line of work could focus on further developing the recommendation algorithm. For example, recommendation systems based on user behavior could be incorporated using machine learning algorithms, in order to offer more accurate recommendations.

- Integration of collaboration tools: This line would consist of adding functionalities that facilitate online collaboration, such as document sharing or videoconferencing systems. These two systems mentioned could be used, for example, to be able to sign and send project contracts or to hold meetings of the project in which you are participating.

- Functionality of generating a work team: This line would focus on developing a functionality that would allow generating work teams for a project. The number of members needed, their profiles would be indicated and the application would automatically form a work team.

- Improvements and corrections in the current functionalities: This line would focus on improving the functionalities already implemented in the application, such as adding the possibility of filtering information in the lists. It would also focus on the correction of errors, optimization of the application and elimination of those obsolete functionalities.

- Impact analysis: In this line, metrics could be developed to evaluate the impact of publications and projects. With these metrics, graphs could be obtained and studies and comparisons could be made.

- Collaboration with scientific institutions: This line would consist of establishing partnerships with scientific institutions, research laboratories and universities to publish their projects and promote scientific collaboration.

- Integration of financing: This line would focus on integrating functionalities so that organizations can seek funding and grants for their projects.

- Remuneration: This line would deal with the idea of remunerating organizations for publishing projects in the application so that it acts as a focus of interest for scientists.

Capítulo 8 - Contribuciones personales

8.1 Alberto Herrera García

La contribución de Alberto al trabajo puede clasificarse en las distintas fases del desarrollo:

1. **Fase de Ingeniería de requisitos**:

   La fase inicial del trabajo consistió en la recopilación y análisis de requisitos aplicando Ingeniería de Requisitos. Su participación fue activa no sólo en la identificación y captura de requisitos sino también en la labor de recopilación de información y elaborar la documentación. Además, participó en la redacción de la especificación de requisitos, elaborando las tablas de casos de uso para los actores que intervienen en la aplicación.

2. **Fase de Modelado de datos:**

   El modelado de datos se realizó, inicialmente, en una reunión junto con Gerald, en la que se logró establecer un modelo de datos de bastante alto nivel. El modelado de datos ha sido una tarea que se ha realizado de forma conjunta a lo largo del desarrollo de la aplicación ya que ha afectado tanto en la implementación front-end de la aplicación web como en el diseño back-end. La labor de Alberto en esta fase ha consistido en detectar y tratar junto con Gerald cualquier tipo de ajuste que ha sido necesario realizar para que el modelo de datos tomara un enfoque cada vez más de bajo nivel.

3. Diseño e implementación de la capa de presentación

El diseño e implementación de la capa de presentación de la aplicación ha sido realizada principal y únicamente por Alberto.

Sus aportaciones más destacadas en esta fase incluyen:

- Confección del logotipo y nombre de la aplicación.
- Decisión de la paleta de colores y tipos de fuentes a utilizar.
- Estudio de la herramienta Angular y creación y gestión del proyecto front-end.
- Desarrollo del código HTML, CSS y TypeScript de las distintas vistas de la aplicación.
- Gestión de la navegación de la aplicación y del enrutamiento de enlaces.
- Ajuste de las vistas para tres tipos de tamaños de dispositivos.
  Portátil L: (1440px x 776px), Portátil estándar: (1024px x 776px) y Tablet: (772px x 776px).
- Creación y desarrollo de los servicios del lado del cliente que comunican la capa de presentación con el backend.
- Control y gestión de errores HTTP.

4. **Ajuste de la representación de información de la API**

En los casos de uso se tuvo que adaptar la representación de información que se enviaba desde el backend y se recibía en el front-end y viceversa. Las respuestas de la API se han realizado en formato JSON y ha sido necesario adaptar esa información a los tipos de datos utilizados en Angular. De la misma manera, a la hora de realizar una petición desde la capa de presentación a la API, ha sido necesario adaptar la información en el otro sentido, de los tipos empleados en Angular al formato JSON.

5. Pruebas y corrección de errores

En cada avance en los casos de uso en la capa de presentación, se han realizado pruebas con el objetivo de verificar su correcto funcionamiento y abordar cualquier error

que pudiese surgir. Estas pruebas no sólo se enfocaron en identificar posibles problemas propios de la capa de presentación, sino también en identificar aquellos que pudieran surgir como resultado de la interacción entre el front-end y el backend.

6. Memoria

La redacción de la memoria ha sido realizada conjuntamente con Gerald.

8.2 Gerald Lima Mendia

1. Ingeniería de requisitos y selección de tecnologías

Durante la fase inicial se investigó la documentación oficial y se estudió varios cursos, tutoriales sobre las tecnologías a utilizar como Spring boot, versión de Java, versión de Angular, el uso de las API Rest para una aplicación con microservicios. Y con las sucesivas reuniones se obtuvieron los requisitos necesarios que debía satisfacer la aplicación.

2. Modelado de los datos

Para la elaboración del modelado de datos, junto con Alberto, se realizaron distintas versiones conceptuales basándose en los requisitos funcionales que se obtenían en las reuniones periódicas. A medida que avanzaba la implementación de la aplicación el enfoque del modelado de datos evolucionaba desde un enfoque de alto nivel a un enfoque de más bajo nivel. Así la base de datos elegida fue PostgreSQL que nos permitía tener un modelo de tablas relacional.

3. Diseño e implementación de la API REST

Se ha diseñado e implementado una API Restful con Spring boot, ya que nos proporcionaba herramientas y librerías para crear microservicios.

Se ha decidido utilizar un diseño del software basado en una Arquitectura en Capas, teniendo como niveles principales: La capa *Controller*, la capa *Service*, la capa *Repository* y la capa *Model.*

Para documentar y diseñar y consumir las API Rest se ha hecho uso de “Swagger (OpenAPI)” en su versión 3. Éste framework es muy útil y visual a la hora de consumir peticiones HTTP.

Además, para algunas peticiones HTTP y pruebas de fallo/error con manejo de excepciones se ha hecho uso de “Postman”, ya que nos permitía depurar los microservicios y validar las respuestas de las mismas.

4. Corrección de errores

Se corrigieron los errores detectados durante la fase de integración de los servicios de la capa de negocio con la aplicación de la capa de presentación. Y eran muy necesarios porque estos afectan ambas partes tanto backend como frontend.

5. Memoria

Se ha redactado la memoria con Alberto conjuntamente.

Capítulo 9 - Bibliografía

[1] «ResearchGate». [En línea]. Disponible en: https://www.researchgate.net/

[2] «Academia.edu». [En línea]. Disponible en: https://www.academia.edu/

[3] «Mendeley». [En línea]. Disponible en: https://www.mendeley.com/

[4] «MDPI SciProfiles». [En línea]. Disponible en: https://sciprofiles.com/

[5] «Angular». [En línea]. Disponible en: https://angular.io/

[6] «Angular, HttpClient». [En línea]. Disponible en: https://angular.io/api/common/http/HttpClient

[7] «Angular, Angular CLI». [En línea]. Disponible en: https://angular.io/cli

[8] «Bootstrap». [En línea]. Disponible en: https://getbootstrap.com/

[9] «MDN Web Docs, CSS». [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Web/CSS

[10] «DBeaver Community». [En línea]. Disponible en:
https://dbeaver.io/

[11] «Firebase». [En línea]. Disponible en:
https://firebase.google.com/?hl=es

[12] «Firebase Authentication». [En línea]. Disponible en:
https://firebase.google.com/docs/auth?hl=es-419

[13] «Google Analytics for Firebase». [En línea]. Disponible en:
https://firebase.google.com/docs/analytics?hl=es-419

[14] «Firebase Realtime Database». [En línea]. Disponible en:
https://firebase.google.com/docs/database?hl=es-419

[15] «Git». [En línea]. Disponible en:
https://git-scm.com/

[16] «Github». [En línea]. Disponible en:
https://github.com/

[17] «MDN Web Docs, HTML». [En línea]. Disponible en:
https://developer.mozilla.org/es/docs/Web/HTML

[18] «MDN Web Docs, HTML5». [En línea]. Disponible en:
https://developer.mozilla.org/es/docs/Glossary/HTML5

[19] «Spring Boot ». [En línea]. Disponible en:
https://spring.io/projects/spring-boot

[20] «Java». [En línea]. Disponible en:
https://www.java.com/es/

[21] «MDN Web Docs, JavaScript». [En línea]. Disponible en:
https://developer.mozilla.org/es/docs/Web/JavaScript

[22] «Node.js». [En línea]. Disponible en:
https://nodejs.org/es

[23] «NPM». [En línea]. Disponible en:
https://www.npmjs.com/

[24] «PostgreSQL». [En línea]. Disponible en:
https://www.postgresql.org/

[25] «Introducción a JSON». [En línea]. Disponible en:
https://www.json.org/json-es.html

[26] «IBM, Transaction Properties». [En línea]. Disponible en:
https://www.ibm.com/docs/es/iis/11.7?topic=transactions-transaction-properties

[27] «IBM, ¿Qué es una interfaz de programación de aplicaciones (API)?». [En línea]. Disponible en: https://www.ibm.com/mx-es/topics/api

[28] «Postman». [En línea]. Disponible en:
https://www.postman.com/

[29] «Visual Studio Code». [En línea]. Disponible en:
https://code.visualstudio.com/

[30] «Wikipedia, Cliente-servidor». [En línea]. Disponible en:
https://es.wikipedia.org/wiki/Cliente-servidor

[31] «OpenWebinars, ¿Qué patrón usa Angular? MVC o MVVM». [En línea]. Disponible en:
https://openwebinars.net/blog/que-patron-usa-angular-mvc-o-mvvm/

[32] «Microsoft, Modelo-Vista-Modelo de vista». [En línea]. Disponible en:
https://learn.microsoft.com/es-es/dotnet/architecture/maui/mvvm

[33] «Appmaster, Operaciones CRUD». [En línea]. Disponible en:
https://appmaster.io/es/blog/operaciones-crud-que-es-crud

[34] «Font Awesome Icons». [En línea]. Disponible en:
https://fontawesome.com/icons

[35] «Bootstrap Icons». [En línea]. Disponible en:
https://icons.getbootstrap.com/

[36] «Google Fonts». [En línea]. Disponible en:
https://fonts.google.com/

[37] «SweetAlert2». [En línea]. Disponible en:
https://sweetalert2.github.io/

Apéndices

Apéndice 1 - Guía de uso

En este apéndice se presenta una guía detallada de uso que proporciona una explicación sobre cómo utilizar y aprovechar todas las funcionalidades de la aplicación.

En primer lugar, se comienzan explicando los primeros pasos para cualquier usuario que vaya a hacer uso de la aplicación, como son el registro de usuario y el inicio de sesión. A partir de ahí, la guía se sumerge en los módulos especializados que atienden a las necesidades de científicos, organismos y administradores, en ese orden.

1. Registro de usuario

Para poder utilizar la aplicación es necesario registrarse como usuario.
En primer lugar, se debe elegir qué tipo de usuario se quiere registrar en la aplicación, si un científico o un organismo.

En caso de seleccionar el científico, se mostrará un formulario que se debe rellenar. (Figura 1).

Una vez relleno, se debe hacer click en el botón “Registrarse”.

En caso de no rellenar con un formato válido alguno de los campos, se mostrará una pantalla emergente que indica cuál de ellos está mal y que explica cuál es el formato correcto.

Figura 1: Registro de científico

En caso de seleccionar el organismo, se debe elegir el ámbito del mismo, ya sea público o privado.

En ambos casos, se mostrará un formulario diferente para cada uno que debe ser rellenado. En la Figura 2, se muestra el formulario para un organismo público mientras que en la Figura 3 se muestra el formulario para un organismo privado.

Una vez relleno, se debe hacer click en el botón “Registrarse”.

En caso de no rellenar con un formato válido alguno de los campos, se mostrará una pantalla emergente que indica cuál de ellos está mal y que explica cuál es el formato correcto.

Figura 2: Registro de organismo público

Figura 3: Registro de organismo privado

2. Inicio de sesión

Una vez registrado, el usuario debe dirigirse al inicio de sesión (Ver Figura 4) donde deberá escribir el email y la contraseña utilizados previamente durante el registro y pulsar el botón “Iniciar Sesión”. Una vez pulsado, la aplicación reconocerá que tipo de usuario es y le redirigirá a la página Home del módulo que corresponda.

Figura 4: Inicio de sesión

3. Módulo científico

Si el usuario que ha iniciado sesión es un científico, se mostrará la siguiente pantalla de Home (Ver Figura 5) en el que se le da la bienvenida a la aplicación y una introducción a las funcionalidades que se pueden realizar en el módulo. Se debe pulsar el botón “Comenzar”.

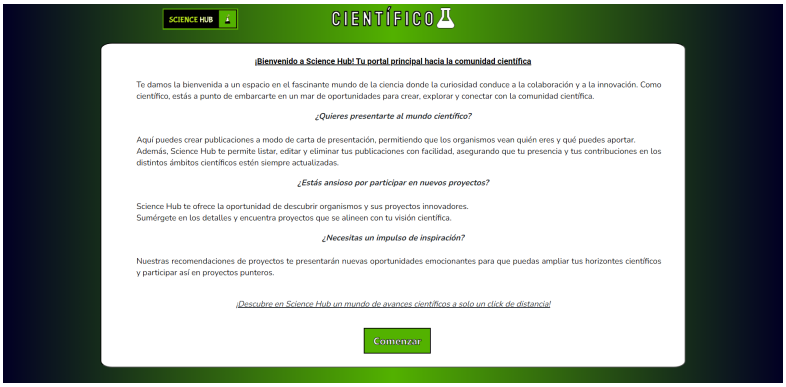

Figura 5: Home de científico

Para navegar por las secciones del módulo científico, tiene disponible en todas las secciones un menú superior con enlaces (Figura 6) y un menú lateral desplegable (Figura 7).

Figura 6: Menú superior con enlaces

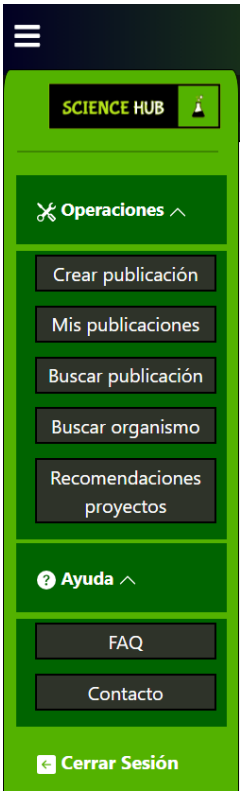

Figura 7: Menú lateral desplegable

3.1 Crear publicación

Para crear una publicación el científico debe rellenar el formulario que aparece en la Figura 8 con la información requerida y pulsar el botón “Crear”.

Figura 8: Crear publicación

3.2 Mis publicaciones

Para listar las publicaciones del científico simplemente se tiene que acceder a la sección.

Si hay publicaciones creadas, se cargarán y se mostrarán en forma de tabla automáticamente, como se muestra en la Figura 9.

Figura 9: Mis publicaciones

3.3 Editar publicación

Para editar una publicación, el científico debe dirigirse a “Mis publicaciones” y hacer click en el botón azul de “Editar” de la publicación que desea editar. Al pulsar el botón, la aplicación le redirigirá a la página “Editar publicación” donde se mostrará un formulario con la información de la publicación a editar. (Ver Figura 10).

Una vez que el científico haya editado los campos de la publicación, debe hacer click en el botón de “Editar” situado en la parte inferior del formulario.

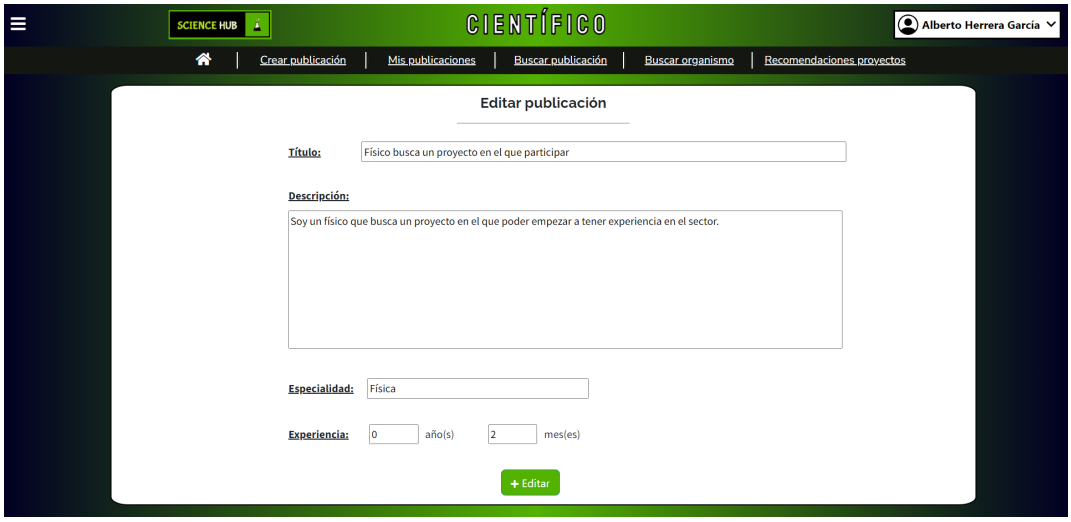

Figura 10: Editar publicación

3.4 Eliminar publicación

Para eliminar una publicación, el científico debe dirigirse a “Mis publicaciones” y hacer click en el botón rojo de “Eliminar” de la publicación que desea eliminar.

En la Figura 11 se muestra el mensaje emergente que aparecerá preguntando si desea eliminar la publicación. Para eliminarla, debe hacer click en el botón “Eliminar”.

Figura 11: Eliminar publicación

3.5 Buscar publicación

Para buscar una publicación, el científico debe escribir el identificador de la publicación que desea buscar y, acto seguido, pulsar el botón “Buscar”.

Puede consultar el identificador de la publicación en “Mis publicaciones”.

Si la publicación existe, se mostrará toda su información por pantalla. (Ver Figura 12)

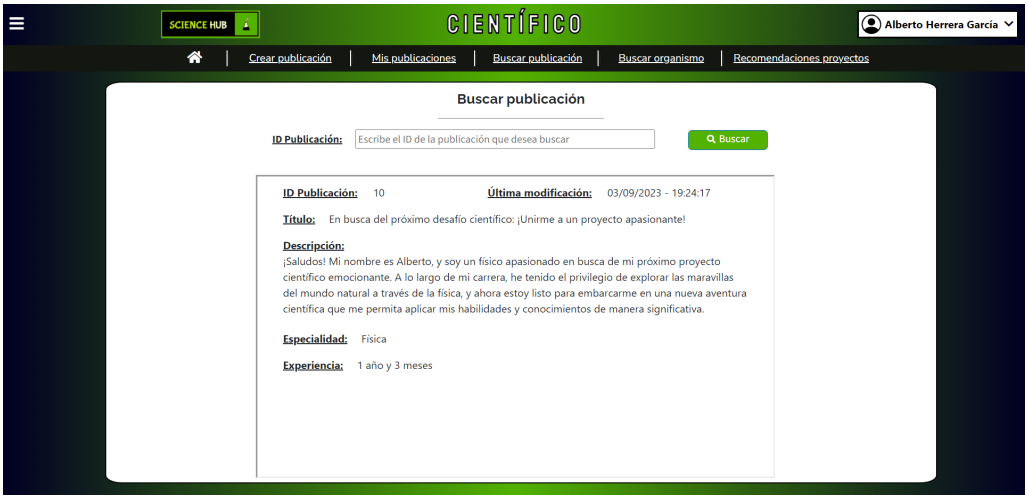

Figura 12: Buscar publicación

3.6 Buscar organismo

Para buscar un organismo, el científico debe escribir total o parcialmente el nombre del organismo que desea encontrar y pulsar en el botón “Buscar”.

Si existen organismos que contengan ese nombre, se mostrarán como “Resultados” en una tabla en la parte izquierda y se habilitará la opción de “Listar proyectos”. En caso de tener proyectos, se listarán en una tabla en el lado derecho de la pantalla y, por cada proyecto, se podrá ver más información de él pulsando en el botón “+ Info”. (Ver Figura 13).

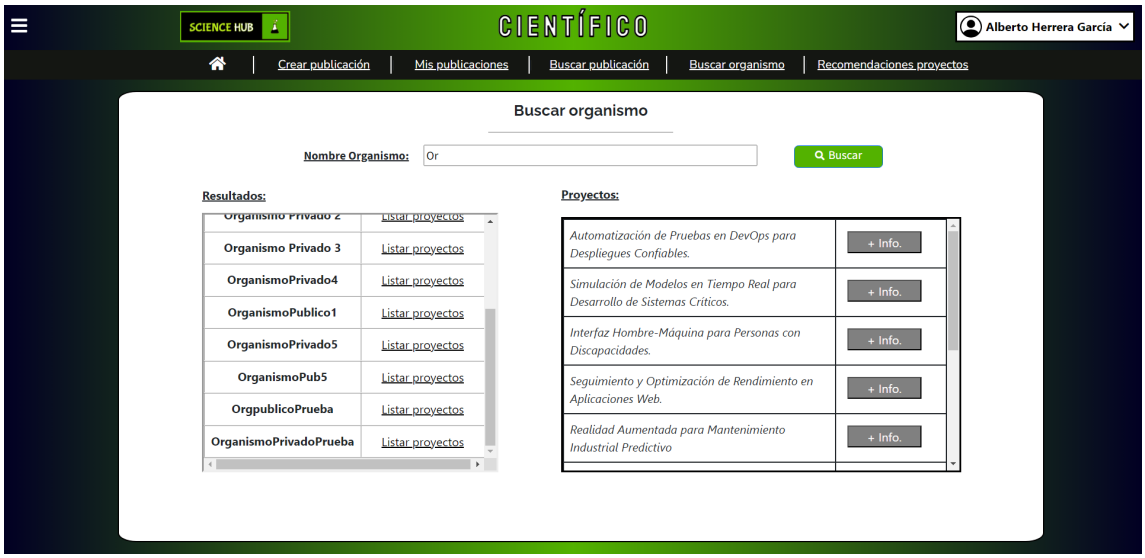

Figura 13: Buscar organismo

3.7 Recomendaciones de proyectos

Para recibir recomendaciones de proyectos, el científico deberá simplemente acceder a la sección. Si existen proyectos que recomendar, se mostrarán automáticamente en forma de tarjetas como se muestra en la Figura 14. En caso de querer ver más información acerca de un proyecto en concreto, el científico puede pulsar el botón “+ Información” que se encuentra en el interior de cada tarjeta.

Figura 14: Recomendaciones de proyectos

4. Módulo organismo

Si el usuario que ha iniciado sesión es un organismo, se mostrará la pantalla de Home que muestra la Figura 15. En ella, se le da la bienvenida a la aplicación y una introducción a las funcionalidades que se pueden realizar en el módulo. Se debe pulsar el botón “Comenzar”.

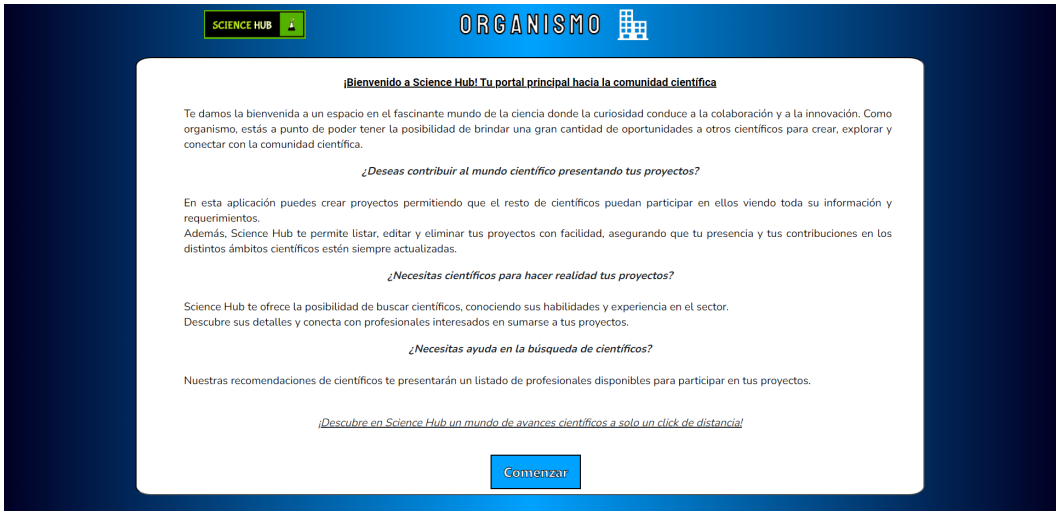

Figura 15: Home del organismo

Para navegar por las secciones del módulo organismo, tiene disponible en todas las secciones un menú superior con enlaces (Figura 16) y un menú lateral desplegable (Figura 17).

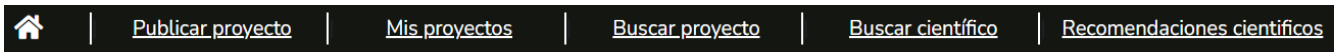

Figura 16: Menú superior con enlaces

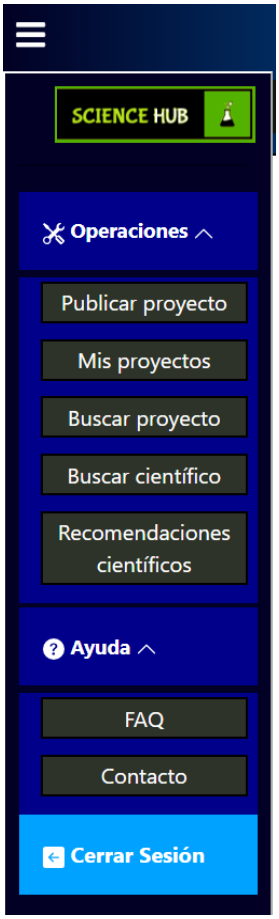

Figura 17: Menú lateral desplegable

4.1 Publicar proyecto

Para publicar un proyecto el organismo debe rellenar el formulario que se muestra en la Figura 18 con la información requerida y pulsar el botón “Publicar”.

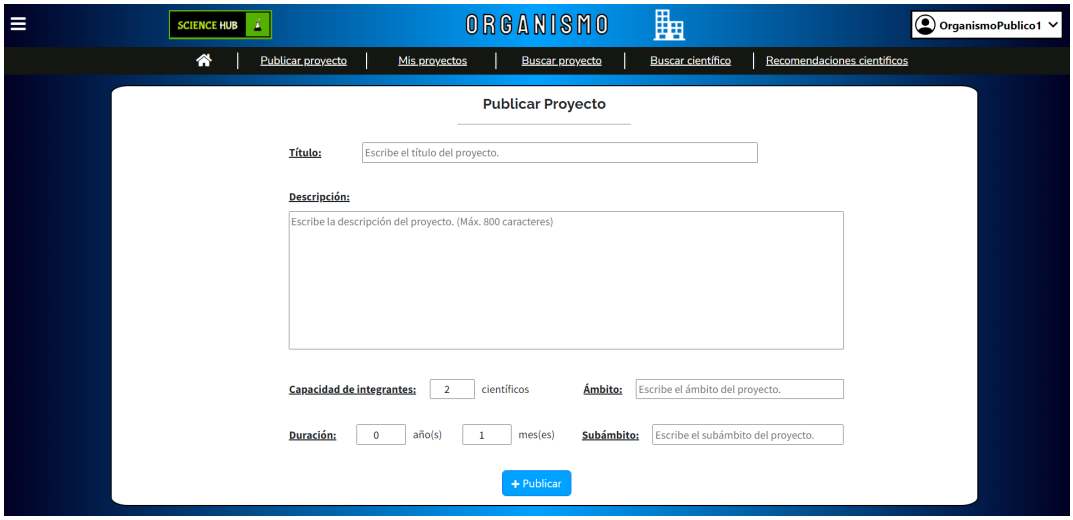

Figura 18: Publicar proyecto

4.2 Mis proyectos

Para listar los proyectos del organismo simplemente se tiene que acceder a la sección.

Si hay proyectos creados, se cargarán y se mostrarán en forma de tabla automáticamente, como se muestra en la Figura 19.

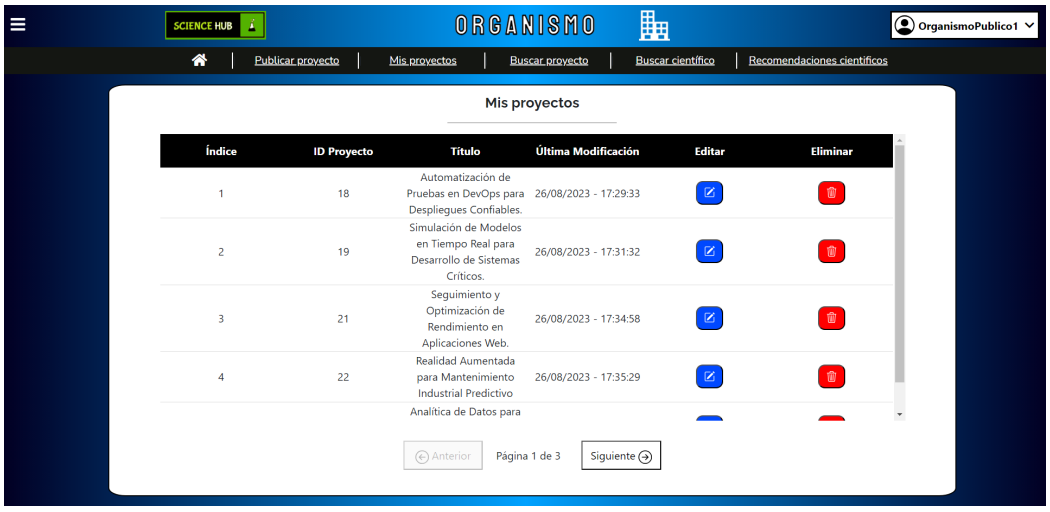

Figura 19: Mis proyectos

4.3 Editar proyecto

Para editar un proyecto, el organismo debe dirigirse a “Mis proyectos” y hacer click en el botón azul de “Editar” del proyecto que desea editar. Al pulsar el botón, la aplicación le redirigirá a la página “Editar proyecto” donde se mostrará un formulario con la información del proyecto a editar. (Ver Figura 20).

Una vez que el organismo haya editado los campos del proyecto, debe hacer click en el botón de “Editar” situado en la parte inferior del formulario.

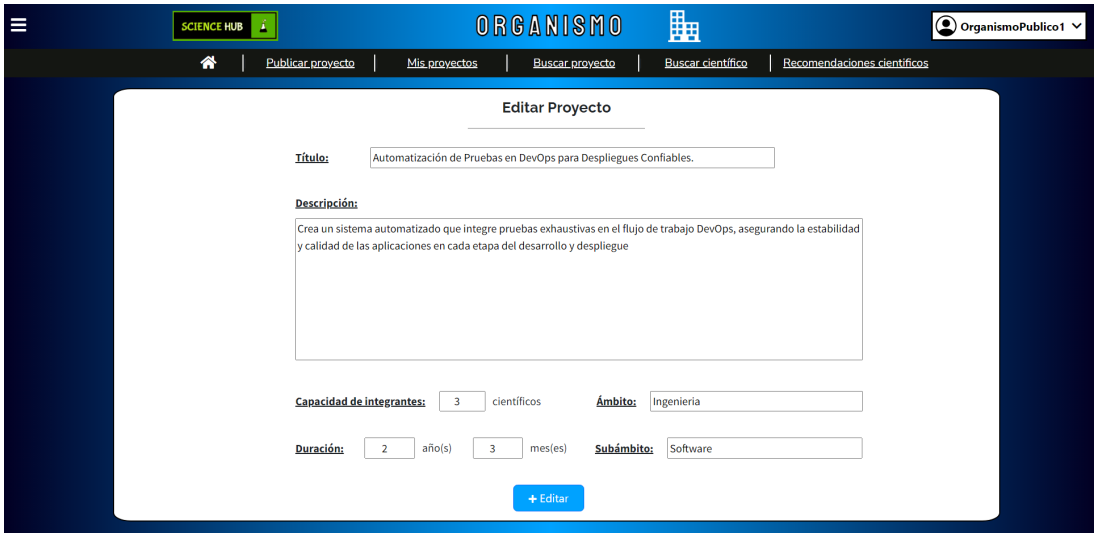

Figura 20: Editar proyecto

4.4 Eliminar proyecto

Para eliminar un proyecto, el organismo debe dirigirse a “Mis proyectos” y hacer click en el botón rojo de “Eliminar” del proyecto que desea eliminar.

En la Figura 21 se muestra el mensaje emergente que aparecerá preguntando si desea eliminar el proyecto. Para eliminarlo, debe hacer click en el botón “Eliminar”.

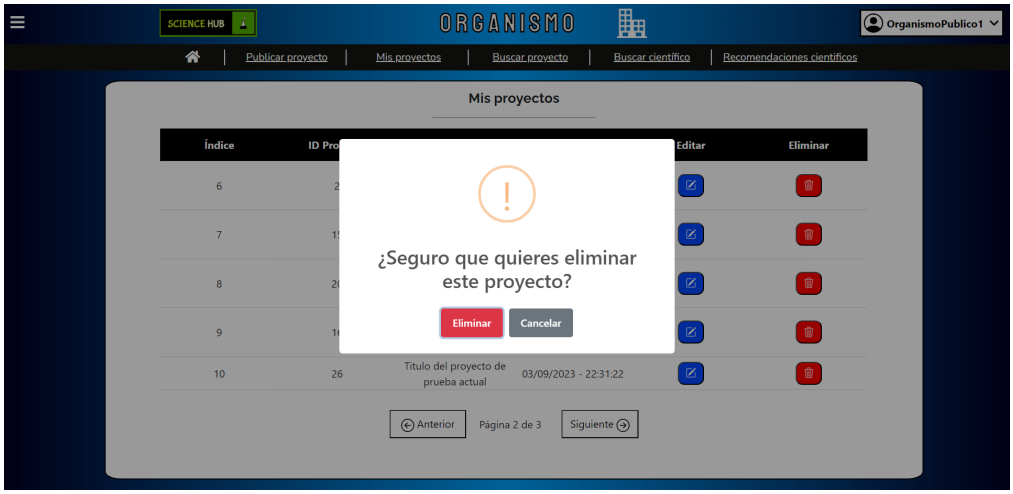

Figura 21: Eliminar proyecto

4.5 Buscar proyecto

Para buscar un proyecto, el organismo debe escribir el identificador del proyecto que desea buscar y, acto seguido, pulsar el botón “Buscar”.

Puede consultar el identificador del proyecto en “Mis proyectos”.

Si el proyecto existe, se mostrará toda su información por pantalla, tal y como se muestra en la Figura 22.

Figura 22: Buscar proyecto

4.6 Buscar científico

Para buscar un científico, el organismo debe escribir el ORCID del científico que desea buscar y, acto seguido, pulsar el botón “Buscar”.

Si el científico existe, se mostrará toda su información por pantalla, como se puede apreciar en la Figura 23.

Figura 23: Buscar científico

4.7 Recomendaciones de científicos

Para recibir recomendaciones de científicos, el organismo deberá acceder a la sección, escribir el identificador del proyecto para el cual desea recibir recomendaciones y pulsar en el botón “Recomendar”. Si existen científicos que recomendar, se mostrarán automáticamente en forma de tarjetas como se muestra en la Figura 24. En caso de querer ver más información acerca de un científico en concreto, el organismo puede pulsar el botón “+ Información” que se encuentra en el interior de cada tarjeta.

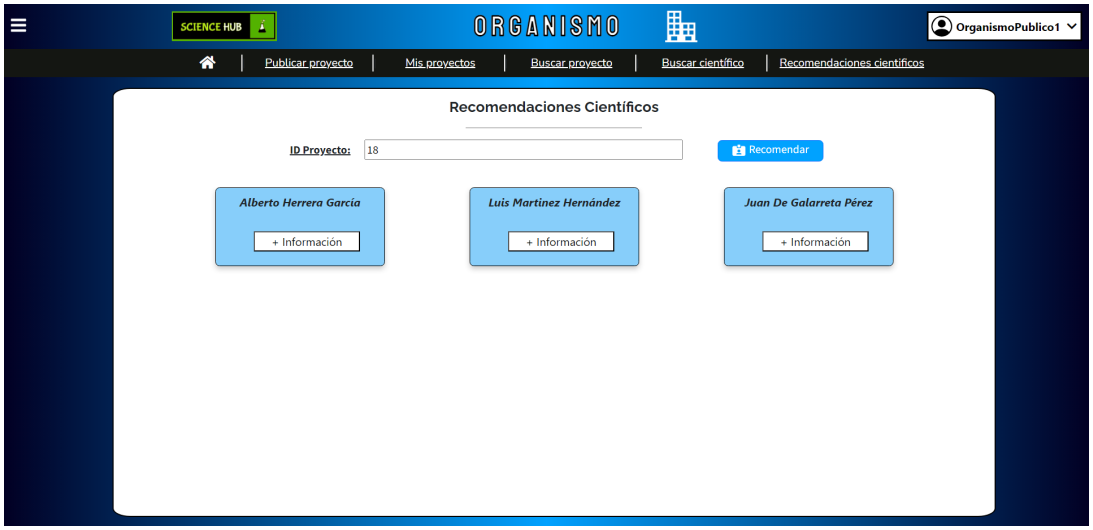

Figura 24: Recomendaciones de científicos

5. Módulo administrador

Si el usuario que ha iniciado sesión es el administrador, se mostrará la siguiente pantalla de Home (Ver Figura 25) en el que se le da la bienvenida a la aplicación y una introducción a las funcionalidades que se pueden realizar en el módulo. Debe pulsar el botón “Comenzar”.

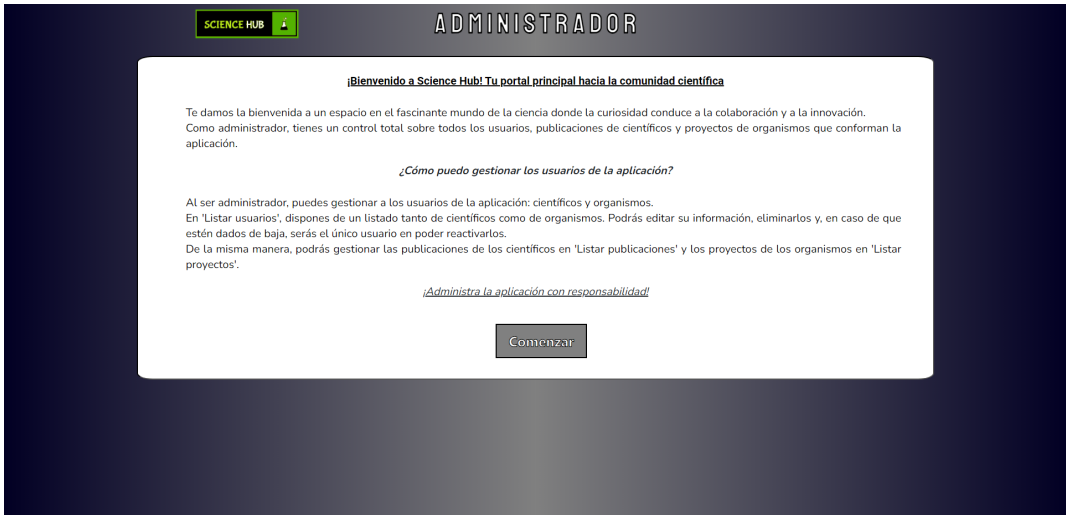

Figura 25: Home del administrador

5.1 Listar todas publicaciones

Para listar todas las publicaciones de los científicos simplemente se tiene que acceder a la sección.

Si hay publicaciones creadas, se cargarán y se mostrarán en forma de tabla automáticamente, como se puede apreciar en la Figura 26.

Figura 26: Listar todas publicaciones

5.2 Editar publicación

Para editar una publicación, el administrador debe dirigirse a “Listar publicaciones” y hacer click en el botón azul de “Editar” de la publicación que desea editar. Al pulsar el botón, la aplicación le redirigirá a la página “Editar publicación” donde se mostrará un formulario con la información de la publicación a editar. (Ver Figura 27).

Una vez que el administrador haya editado los campos de la publicación, debe hacer click en el botón de “Editar” situado en la parte inferior del formulario.

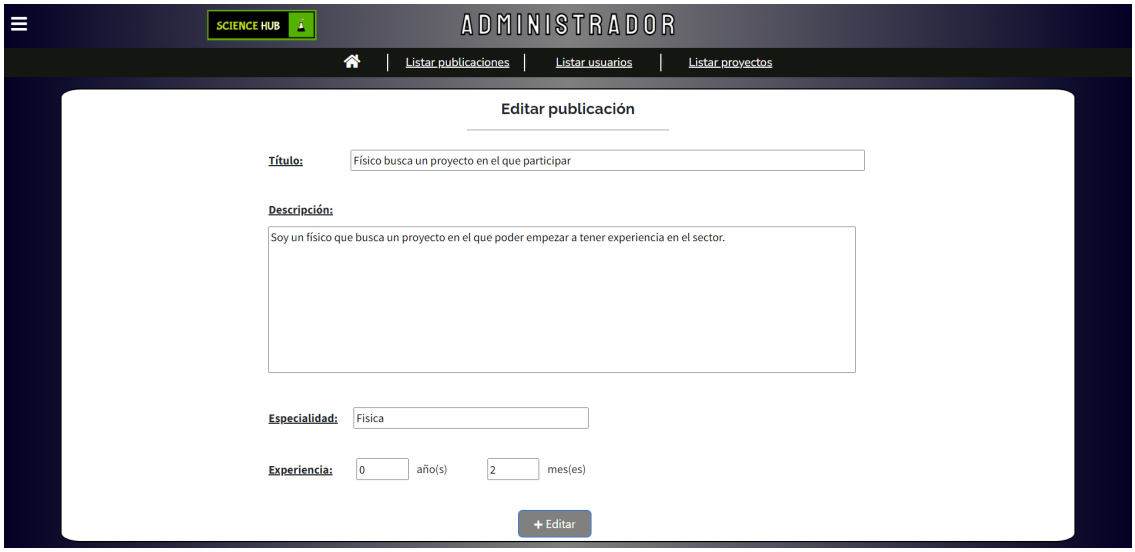

Figura 27: Editar publicación

5.3 Eliminar publicación

Para eliminar una publicación, el administrador debe dirigirse a “Listar publicaciones” y hacer click en el botón rojo de “Eliminar” de la publicación que desea eliminar.

En la Figura 28 se muestra el mensaje emergente que aparecerá preguntando si desea eliminar la publicación. Para eliminarla, debe hacer click en el botón “Eliminar”.

Figura 28: Eliminar publicación

5.4 Reactivar publicación

Para reactivar una publicación, el administrador debe dirigirse a “Listar publicaciones” y hacer click en el botón verde de “Reactivar” de la publicación dada de baja que desea reactivar.

En la Figura 29 se muestra el mensaje emergente que aparecerá preguntando si desea reactivar la publicación. Para reactivarla, debe hacer click en el botón “Reactivar”.

Figura 29: Reactivar publicación

5.5 Listar todos usuarios

Para listar todos los usuarios de la aplicación simplemente se tiene que acceder a la sección. Los científicos se listarán en una tabla en la parte izquierda y, en otra tabla, los organismos en la parte derecha de la pantalla. (Ver Figura 30).

Si hay usuarios creados, se cargarán y se mostrarán en forma de tabla automáticamente.

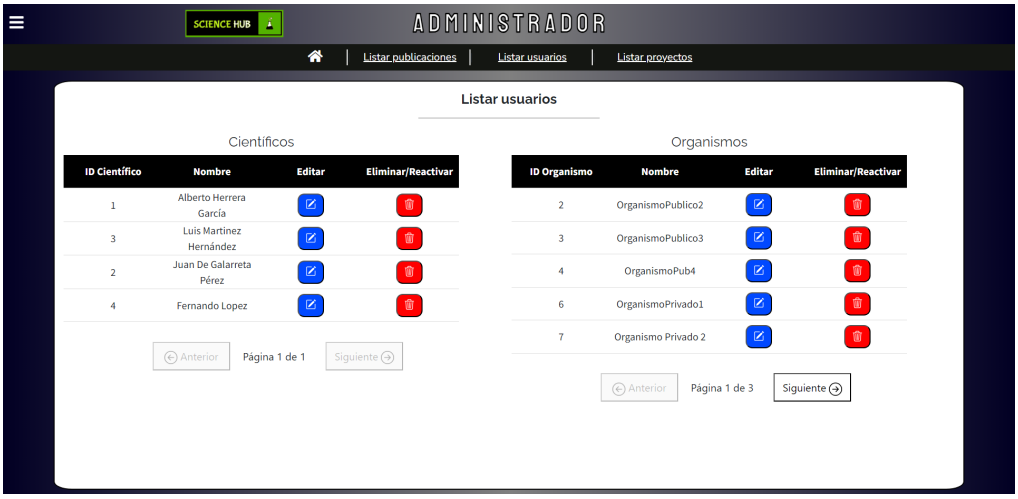

Figura 30: Listar usuarios

5.6 Editar usuario

Para editar un usuario, el administrador debe dirigirse a “Listar usuarios” y hacer click en el botón azul de “Editar” del usuario que desea editar.

Al pulsar el botón, la aplicación le redirigirá a la página “Editar usuario” donde se mostrará un formulario con la información del usuario a editar. Si el usuario es un científico se mostrará el formulario presente en la Figura 31 mientras que, si el usuario es un organismo, se mostrará el formulario que se encuentra en la Figura 32.

Una vez que el administrador haya editado los campos del usuario, debe hacer click en el botón de “Editar” situado en la parte inferior del formulario.

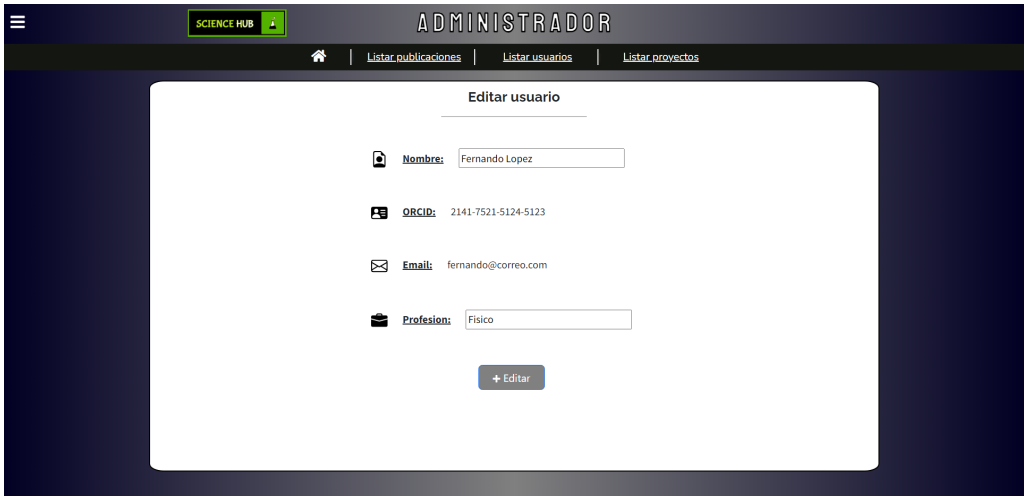

Figura 31: Editar científico

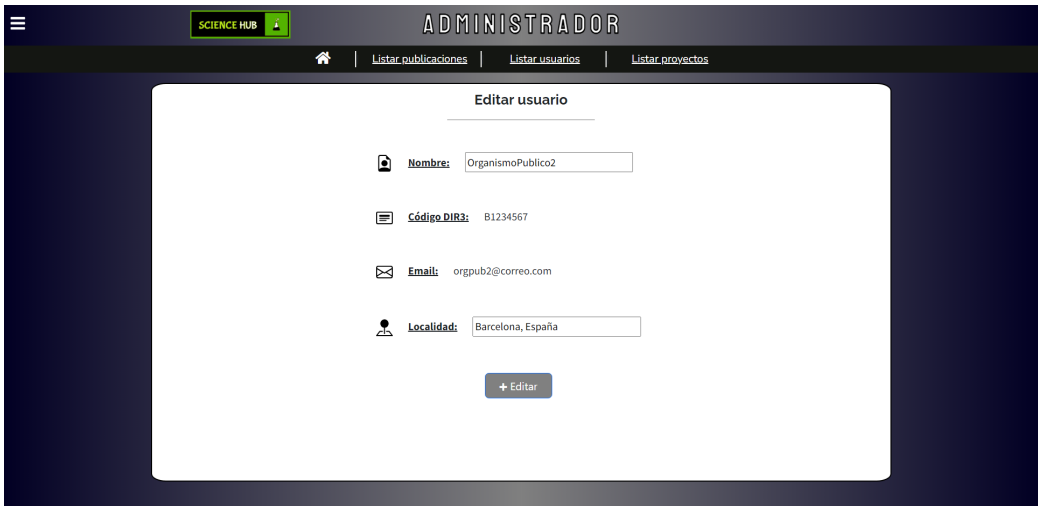

Figura 32: Editar organismo

5.7 Eliminar usuario

Para eliminar un usuario, el administrador debe dirigirse a “Listar usuarios” y hacer click en el botón rojo de “Eliminar” del usuario que desea eliminar.

Si el usuario que se desea eliminar es un científico, aparecerá el mensaje emergente de la Figura 33. Sin embargo, si el usuario que se desea eliminar es un organismo, el mensaje

emergente será el que se puede apreciar en la Figura 34. Para eliminar cualquiera de ellos, debe hacer click en el botón “Eliminar”.

Figura 33: Eliminar científico

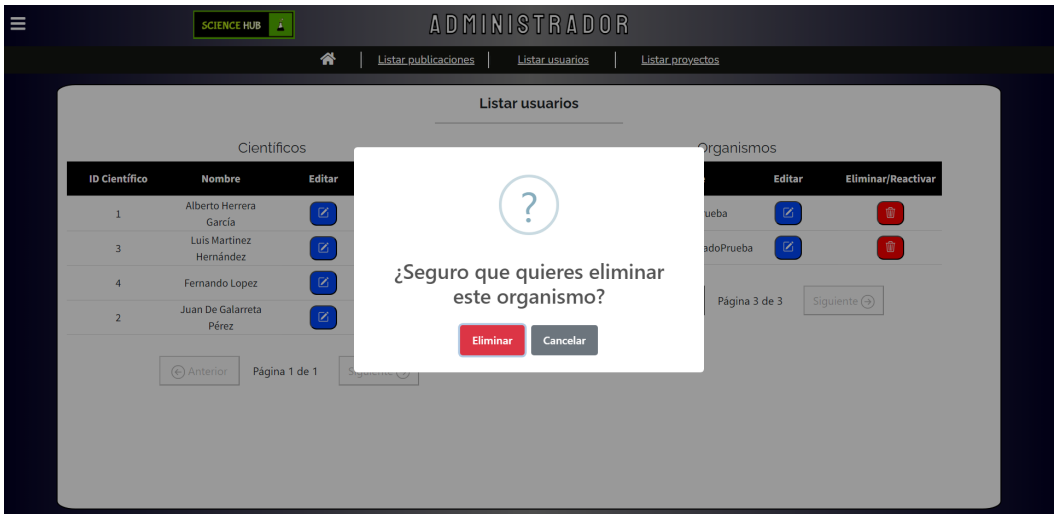

Figura 34: Eliminar organismo

5.8 Reactivar usuario

Para reactivar un usuario, el administrador debe dirigirse a “Listar usuarios” y hacer click en el botón verde de “Reactivar” del usuario dado de baja que desea reactivar.

Si el usuario que se desea reactivar es un científico, aparecerá el mensaje emergente de la Figura 35. Sin embargo, si el usuario que se desea reactivar es un organismo, el mensaje emergente será el que se puede apreciar en la Figura 36.

Para reactivar cualquiera de ellos, debe hacer click en el botón “Reactivar”.

Figura 35: Reactivar científico

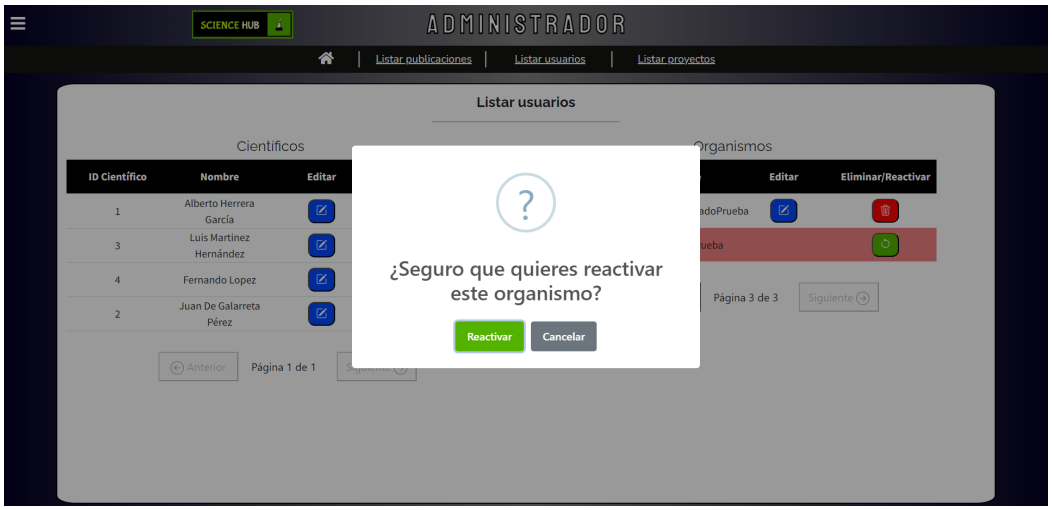

Figura 36: Reactivar organismo

5.9 Listar todos proyectos

Para listar todos los proyectos de los organismos simplemente se tiene que acceder a la sección.

Si hay proyectos creados, se cargarán y se mostrarán en forma de tabla automáticamente, como se puede apreciar en la Figura 37.

Figura 37: Listar todos proyectos

5.10 Editar proyecto

Para editar un proyecto, el administrador debe dirigirse a “Listar proyectos” y hacer click en el botón azul de “Editar” del proyecto que desea editar. Al pulsar el botón, la aplicación le redirigirá a la página “Editar proyecto” donde se mostrará un formulario con la información del proyecto a editar. (Ver Figura 38).

Una vez que el organismo haya editado los campos del proyecto, debe hacer click en el botón de “Editar” situado en la parte inferior del formulario.

Figura 38: Editar proyecto

5.11 Eliminar proyecto

Para eliminar un proyecto, el administrador debe dirigirse a “Listar proyectos” y hacer click en el botón rojo de “Eliminar” del proyecto que desea eliminar.

En la Figura 39 se muestra el mensaje emergente que aparecerá preguntando si desea eliminar el proyecto. Para eliminarlo, debe hacer click en el botón “Eliminar”.

Figura 39: Eliminar proyecto

5.12 Reactivar proyecto

Para reactivar un proyecto, el administrador debe dirigirse a “Listar proyectos” y hacer click en el botón verde de “Reactivar” del proyecto dado de baja que desea reactivar.

En la Figura 40 se muestra el mensaje emergente que aparecerá preguntando si desea reactivar el proyecto. Para reactivarlo, debe hacer click en el botón “Reactivar”.

Figura 40: Reactivar proyecto