Aplicación web de soporte al Aprendizaje-Servicio

Web application for supporting Service-Learning

UNIVERSIDAD COMPLUTENSE DE MADRID
FACULTAD DE INFORMÁTICA
GRADO EN INGENIERÍA DE SOFTWARE

TRABAJO DE FIN DE GRADO

Autores:

Sergio Arroyo Galán (Grado en Ingeniería de Software)
Yrving David Conde Cubas (Grado en Ingeniería de Software)
Adrián Dorta Yagüe (Grado en Ingeniería de Software)

Directores:

Manuel Montenegro Montes
Simon Pickin

Curso académico 2021-2022

Agradecimientos

Agradecimientos a Simon Pickin y a Manuel Montenegro por todo el apoyo que nos han brindado durante el proyecto. También a nuestras familias y amigos por animarnos y confiar en nosotros más que nosotros mismos. Somos quien somos gracias a todos vosotros.

Índice general

Índice de figuras

Enlace al código fuente del proyecto

https://github.com/ucm-tfg/tfg-aps

https://github.com/ucm-tfg/tfg-aps-configuration

Resumen

Un camino de mil millas comienza con un paso

Benjamin Franklin

El Aprendizaje-Servicio es una práctica académica que combina procesos de aprendizaje con servicios en favor de la comunidad para así, ayudar al alumnado a implicarse en proyectos y actividades de su entorno.

La experiencia de los profesores que han llevado a cabo, o intentado llevar a cabo, iniciativas de tipo ApS en España, muestra que muchos proyectos potenciales no llegan a realizarse por la dificultad en casar la oferta con la demanda, es decir, cuadrar las necesidades didácticas, organizativas, etc. de la institución educativa que quiere prestar un servicio, con las necesidades y disponibilidad de la organización o comunidad que quiere recibir un servicio. Un buen soporte informático podría ser de gran ayuda en la tarea de casar la oferta y la demanda de ApS. Gracias a este soporte se facilitarían la identificación de potenciales partenariados así como la colaboración entre el prestador y el receptor potenciales del servicio en la tarea de refinar una idea inicial y convertirla en un propuesta de proyecto realista que cumple las necesidades de ambas partes.

Durante los últimos cursos académicos se ha ido desarrollando una aplicación web que ofrezca este tipo de soporte a los proyectos ApS. Este TFG tiene como objetivo la extensión de dicha aplicación.

Partiendo del TFG del curso pasado, los objetivos a desarrollar fueron la creación de nuevos formularios como los de listado de ofertas y demandas, los de información detallada para una oferta o demanda concreta y el de creación de partenariados, la implementación de filtros para las ofertas y las demandas, la integración de un chat en la aplicación, la ampliación del modelo de dominio y el de datos.

El presente TFG ha sido desarrollado utilizando tecnologías como Angular, Express.js, JavaScript, Node.js y MySQL.

Palabras clave

- Aprendizaje-Servicio
- ApS
- Demanda de servicio
- Necesidad social
- Oferta de servicio
- Partenariado
- Socio comunitario

Abstract

La historia es un incesante volver a empezar.

Tucídides

Service-Learning is an academic practice that combines learning processes with services in favor of the community in order to help students to get involved in projects and activities in their environment.

The experience of teachers who have carried out, or tried to carry out, SL type initiatives in Spain, shows that many potential projects do not come to fruition due to the difficulty in matching offer with demand, that is, balancing needs educational, organizational, etc. of the educational institution that wants to provide a service, with the needs and availability of the organization or community that wants to receive a service. Good IT support could be of great help in the difficult task of matching SL offer and demand. Thanks to this support, the identification of potential partnerships would be facilitated, as well as the collaboration between the potential provider and receiver of the service in the task of refining an initial idea and turning it into a realistic project proposal that meets the needs of both parties.

During the last academic years, a web application has been developed that offer this type of support to ApS projects. This TFG aims to extend of said application. Starting from the FPD of the last course, the objectives to be developed were the creation of new forms such as those for the list of offers and demands, those for detailed information for a specific supply or demand and the creation of partnerships, the implementation of filters for offers and requests, the integration of a chat in the application, the extension of the domain model and the data model.

Starting from the FDP of the last course, the objectives to develop were the creation of new forms such as those for the lists of offers and demands, those for detailed information for a specific offer or demand and the one for creation of partnerships, the implementation of filters for offers and demands, the integration of a chat in the application, the expansion of the domain model and data model.

This FDP has been developed using technologies such as Angular, Express.js, JavaScript, Node.js and MySQL.

Keywords

- SL
- Service-Learning
- Service demand
- Social need
- Service offer
- Partnership
- Community partner

Capítulo 1

Introducción

El secreto de ir avanzando es empezar

Mark Twain

1.1. Presentación de la propuesta

Para Barbara Jacoby [3], el *Aprendizaje-Servicio* (ApS) es una forma de educación basada en la experiencia, en la que los estudiantes contribuyen en diversas actividades que abordan distintos aspectos como pueden ser los derechos humanos o las necesidades de una comunidad, junto con oportunidades para la reflexión para así lograr los resultados del aprendizaje deseados. Jacoby [3] añade que algunas definiciones de *Aprendizaje-Servicio* establecen claramente la necesidad de que el *ApS* sea parte del currículo académico. Además el *ApS* no es una actividad de voluntariado, ya que, en las actividades de voluntariado el énfasis está en el servicio prestado a la comunidad, y en consecuencia, el aspecto de aprendizaje tiene poco peso. Aps tampoco es lo mismo que prácticas externas, ya que, en estas el énfasis está en el aprendizaje, y en consecuencia, el servicio prestado tiene poco peso.

Los profesores con experiencia en iniciativas de tipo ApS han llegado a la conclusión de que un buena herramienta informática podría ser de gran utilidad a la hora de emparejar las ofertas y las demandas ApS. Gracias a este soporte se facilitarían la identificación de potenciales partenariados, así como la colaboración entre el prestador y el receptor de un

determinado servicio.

El objetivo principal de este proyecto es desarrollar una aplicación web de soporte al ApS, extendiendo y completando un prototipo desarrollado en Node.js en el contexto de un TFG [4] realizado durante el curso 2020-21.

1.2. Objetivos

La aplicación implementada en el contexto de este TFG está compuesta por dos subsistemas claramente diferenciados:

- *Subsistema de ofertas-demandas*: Este subsistema tiene como finalidad gestionar las ofertas de servicio creadas por los profesores y las demandas de servicio creadas por los socios comunitarios. Ambas se intentan emparejar, ya sea de forma manual o automáticamente a través del sistema de *matching* para crear partenariados entre el socio comunitario y los profesores, con el fin último de crear y lanzar proyectos ApS. Este primer subsistema ya existía a la hora de comenzar el presente proyecto, pero tenía bastantes partes por añadir y terminar.

- *Subsistema de proyectos activos*: Este subsistema permite dar soporte a proyectos ApS en marcha, concretamente, soporte para la evaluación de los alumnos. Este subsistema, se creó en relación a este proyecto. En la aplicación no existe esta funcionalidad, ni siquiera la especificación de requisitos.

Partiendo del proyecto anterior, y teniendo en cuenta los subsistemas de los que está compuesta la aplicación, los objetivos establecidos fueron la creación y modificación de diversas vistas, la capacidad de filtrado por varios campos, la creación de partenariados a partir de una oferta y demanda, y la especificación del ciclo de vida de un proyecto ApS y evaluación de los estudiantes. A continuación se puede observar un listado de los objetivos de este TFG:

- *Eliminación de todas las referencias a las iniciativas, debido a que ya no se utilizan en el presente TFG.* En una versión previa de la aplicación realizado en la *UNED* [5], tanto

un profesor como un socio comunitario podían crear iniciativas. Esto fue debido a una mala "comprensión" de la falta de simetría entre los conceptos de oferta de servicio (creadas por profesores) y demandas de servicio (creadas por socios comunitarios). Esta parte debió ser eliminada en la aplicación llevada a cabo en el pasado TFG [4], pero no fue el caso.

- *Limpieza del código fuente referido a MongoDB, sistema de base de datos no utilizado en este proyecto.* En el TFG desarrollado en la UNED [5], se decidió implementar la base de datos usando MongoDB. En la aplicación desarrollada el curso pasado [4] se cambió la base de datos por una MySQL, y se programó los DAO correspondientes, pero todavía quedaba código relacionado con MongoDB, ahora ya eliminado en su totalidad.

- *Implementación de las vistas para el listado de ofertas y demandas.* Se implementaron estas vistas debido a la necesidad de visualizar en la aplicación las ofertas y demandas disponibles.

- *Implementación de filtros para las vistas mencionadas anteriormente.* Se implementaron, en el listado de ofertas, los filtros necesarios por título, por profesor, según los *tags* creados, por área de implementación, por año académico y cuatrimestre objetivo. En relación al listado de demandas, se crearon filtros según el título, la necesidad social a cumplir, el área de implementación y la entidad demandante.

- *Creación de las vistas de información detallada, tanto para oferta como para demanda.* Estas vistas muestran la información de cada oferta y demanda de forma individual, además de dar la posibilidad de aceptar la oferta (si el usuario es un socio comunitario) o respaldar la demanda (si el usuario es un profesor interno).

- *Implementación del formulario para la creación de partenariados a través de una oferta y una demanda.* Para crear un partenariado hay tres posibilidades:

    - La aceptación por parte de un socio comunitario de la oferta realizada por un profesor interno.
    - El respaldo por parte de un profesor de una demanda de servicio creada por un socio comunitario.
    - El emparejamiento automático de una oferta de servicio creada por un profesor y una demanda de servicio creada por un socio comunitario, haciendo uso del algoritmo de *matching* desarrollado en el TFG [4] del año pasado.

- *Implementación de la página personal de cada usuario*. Es un perfil para mostrar la información de cada usuario.

- *Implementación de infraestructura*.

- *Implementación de notificaciones dentro de la aplicación*. Las notificaciones se utilizarían para avisar a los profesores cuando un socio comunitario ha aceptado su oferta de servicio o cuando un profesor decide respaldar una demanda para avisar al socio comunitario correspondiente. Todo esto, con la finalidad de crear un partenariado entre ambos.

- *Corrección de bugs encontrados en el anterior TFG*.

- *Ampliación del modelo de dominio y modelo de datos*. Ampliación del modelo de dominio y de datos del subsistema ofertas-demandas definidos en el anterior TFG [4] para así incluir el modelado del subsistema proyectos-activos.

- *Definición de ciclo de vida de un proyecto*. Definición del proceso de ciclo de vida de un proyecto, desde la creación de una oferta o demanda, con el fin de especificar cómo será la navegación entre las distintas vistas de la aplicación. Se podría haber especificado la parte del proceso soportado por el subsistema ofertas-demandas en la fase de análisis del desarrollo del pasado TFG [4], pero no fue el caso. Se había constatado, durante

el TFG [4] anterior, que en ausencia de una especificación del proceso solían surgir ciertas confusiones al respecto entre los desarrolladores.

- Definición de la evaluación del proyecto. La calificación del proyecto se evalúa en función del servicio que ha sido capaz de proporcionar a los beneficiarios.

1.3. Plan de trabajo

Una vez establecidos los objetivos principales de este TFG, se decidió dividirlo en las fases que se observan a continuación:

- La primera fase consistió en el estudio de la memoria del TFG anterior, además de investigar acerca del Aprendizaje-Servicio para tener una primera aproximación al contexto en el que se desarrollaba el proyecto.

- En la segunda fase, nos adentramos en el código del anterior TFG para familiarizarnos con él. Además, realizamos pruebas manuales en las que encontramos algunos *bugs* que se corrigieron más adelante. En la parte de Diseño, realizamos la ampliación del Modelo de Dominio y el Modelo de Datos para profundizar más acerca de cómo plantear un proyecto ApS.

- La tercera fase consistió en el aprendizaje de Angular, tecnología desconocida para todos nosotros pero indispensable en la implementación de este proyecto y en la creación y modificación de varios formularios. En la parte de Análisis decidimos profundizar más en los procesos de un proyecto mediante diagramas BPMN para representar la creación de un proyecto y el ciclo de vida del mismo.

- En la cuarta fase nos centramos en la parte de infraestructura del proyecto, de esta manera podemos hacer mas robusto y escalable la aplicación en medio y largo plazo.

- Para finalizar nos centramos en el desarrollo de los casos de usos pendientes, asi como limpiar y solucionar *bugs* encontrados a medida que se avanzaba en el proyecto.

1.4. Presentación del resto del documento

A continuación aparece la secuencia de capítulos, así como una breve descripción de su contenido:

- **Capítulo 1: Introducción**. En este capítulo se ha hecho una breve presentación de la propuesta, además de presentar los objetivos propuestos y el plan de trabajo llevado a cabo.
- **Capítulo 2: Introduction**. Traducción del capítulo anterior al idioma inglés.
- **Capítulo 3: Estado del arte**. En esta sección se presentan los antecedentes al presente proyecto.
- **Capítulo 4: Elección de tecnologías**. En este capítulo se comentan las tecnologías utilizadas durante el desarrollo del proyecto y las razones que llevaron a su uso.
- **Capítulo 5: Especificación de requisitos**. En este capítulo se comentan los modelos de dominio y de datos y los diagramas *BPMN* acerca del ciclo de vida de un proyecto.
- **Capítulo 6: Implementación**. En este capítulo se relata el proceso seguido en la creación de los formularios y de la infraestructura.
- **Capítulo 7: Conclusiones y trabajo futuro**. En esta sección se tratan los objetivos completados, los problemas encontrados y el posible trabajo a desarrollar en futuros proyectos.
- **Capítulo 8: Conclusions and future work**. Traducción del capítulo anterior al idioma inglés.
- **Capítulo 9: Contribuciones**. En esta capítulo se relatan las contribuciones al proyecto de cada miembro del equipo.

- **Capítulo 10: Instalación**. En este capítulo se detallan los pasos a seguir para instalar el proyecto.

Capítulo 2

Introduction

Lo que nunca empieza, nunca termina.

Patricio Osorio

2.1. Background

For Barbara Jacoby [3], Service-Learning (SL) is a form of education based on experience, in which students contribute with various activities that address different aspects such as human rights or the needs of a community, along with opportunities for reflection in order to achieve the desired learning outcomes. Jacoby adds that some SL definitions clearly state the need for SL to be part of the academic curriculum. Furthermore, SL is not a volunteer activity; the emphasis is on the service, that is, the needs of a community.

Teachers with experience in SL iniciatives have come to the conclusion that good IT support could be very useful in matching SL offers and demands. Thanks to this support, the identification of potencial partnerships would be facilitated, as well as the collaboration between the provider and the receiver of a given service.

The main objective of this FDP is to develop a web application to support Service-Learning, extending and completing a prototype developed in Node.js in the context of a FDP carried out during the 2020-2021 academic year.

2.2. Objetives

The application implemented in the context of this FDP is made up of two differenciated subsystems:

- *Subsystem of offers-demands*: This subsystem is intended to manage the service offers created by the teachers and the service demands created by the community partners. Both try to match each other, either manually or automatically through the matching system to create partnerships between the community partner and the teachers, with the ultimate goal of creating and launching SL projects. This first subsystem already existed at the time of starting this project, but it had many parts to add and finish.

- *Active projects subsystem*: This subsystem allows support for ongoing SL projects, specifically, support for students assessment. This subsystem was created in relation with this project. This functionality does not exist in the application, not even the requirements specification.

Starting from the last FDP, and considering the subsystems of which the application is made up, the established objectives were the creation and modification of various views, the ability to filter by various fields, the creation of partnerships based on offer and demand, the specification of the life cycle of a SL project. Below you can see a list of the objectives of this FDP:

- *Removal of all references to initiatives, as they are no longer used in the present FDP.* In a previous version of the application made at the UNED [5], both teacher and community partner could create iniciatives. This was due to a poor "understanding" of the lack of simmetry between the concepts of service offer (created by teachers) and service demands (created by community partners). This part should have been eliminated in the application carried out in the last FDP [4], but it was not the case.

- *Cleanup of source code referring to MongoDB, database system not used in this FDP.*

In the FDP developed at UNED [5], it was decided to implement the database using MongoDB. In the application develop last year [4], the database was changed for MySQL, in addition to its corresponding DAO, but there was still code related to MongoDB, now already removed in it whole.

- *Implementation of the views for the list of offers and demands.* These views were implemented due to the need to visualize the available offers and demands.

- *Implementation of filters for the previous views.* In the offers list view, the neccesary filters where implemented by title, by professor, according to the tags created, by area of implementation, by academic year and objective semester. In relation with the demands list, filters were created according to title, the social need to comply, the area of implementation and the requesting entity.

- *Creation of detailed information views, both for supply and demand.* These views show the information of each offer and demand individually, in addition to give the possibility to accept the offer (if the user is a community partner) or support the request (if the user is a professor).

- *Implementation of a form for the creation of partnerships through an offer and a demand.* To create a partnership there are three possibilities:

  - The acceptance by a community partner of the offer made by a professor.
  - Endorsement by a professor of a service demand created by a community partner.
  - Automatic matching of a service offer created by a professor and a service demand created by a community partner, making use of the matching algorithm develop in the last year FDP [4].

- *Implementation of the personal page for each user.* Is a profile to show the information of each user.

- *Implementation of the infrastructure.*

- *Implementation of notifications within the application.* Notifications would be used to notify teachers when a community partner has accepted their offer of service or when a teacher decides to support a demand to notify the community partner. All this with the purpose of being able to create a partnership between both of them.

- *Correction of bugs found in the previous FDP.*

- *Extension of the domain model and data model.* Extension of the domain and data model of the offers-demands subsystem defined in the previous FDP [4] in order to include the modeling of active-projects subsystem.

- *Definition of a project life cycle.* Definition of the life cycle process of a project, from the creation of an offer or demand, in order to specify how will be the navigation between the different views of the application. The part of the proccess supported by the offer-demand subsystem could have been specified in the analysis phase of the development of the past FDP [4], but it was not the case. It had verified, during the previous FDP [4], that in the absence of a specification of the process there used to be some confusion about it among developers.

- *Definition of project evaluation.* The project evaluation is on function of the service it has been able to provide to the beneficiaries.

2.3. Workplan

Once the main objectives of this project were established, it was decided to divide it into the phases which are seen below:

- The first phase consisted of studying the memory of the previous FDP. In addition to investigate about Service-Learning in order to get an idea of what we were facing.

- In the second phase, we delve into the code of the previous FDP to familiarize ourselves with it. In addition, we perform manual tests in which we find some bugs that were fixed later.

- The third phase consisted of learning Angular, an unknown technology for all of us but indispensable in the implementation of this project and in the creation and modification of various forms. In the Analysis part, we decided to deepen more on the processes of a project using BPMN diagrams to represent the creation of a project and its life cycle.

- In the fourth phase, we focus on the infrastructure of the project, in this way we can make the application more robust and scalable in the medium and long term.

- Finally, we focus on the development of pending use cases, as well as clean and fix bugs found as the project progressed.

2.4. Presentation of the rest of the document

Below is the sequence of chapters, as well as a brief description of their content:

- **Chapter 1: Introduction**. In this chapter, a brief presentation of the proposal, in addition to presenting the propose objectives and the work plan carried out.

- **Chapter 2: Introduction**. Translation of the previous chapter into English.

- **Chapter 3: State of the art**. This chapter presents the background of this project.

- **Chapter 4: Choice of technologies**. This chapter discusses the technologies used during the development of the project and the reason that led to its use.

- **Chapter 5: Requirements specifications**. This chapter discusses the data and domain models and *BPMN* diagrams about the life cycle of a project.

- **Chapter 6: Implementation**. This chapter describes the proccess followed in the creation of forms and infrastructure.
- **Chapter 7: Conclusions and future work**. This section discusses the completed objetives, the encoutered problems and the possible work to be done in future projects.
- **Chapter 8: Conclusions and future work**. Translation of the previous chapter into English.
- **Chapter 9: Contributions**. This chapter relates the contributions of teach team member to the project.
- **Chapter 10: Instalation**. This chapter details the steps to follow to install the project.

Capítulo 3

Estado del arte / Precedentes

El día precedente enseña el día que sigue

Píndaro

3.1. Antecedentes

El Aprendizaje-Servicio no es una técnica educativa nueva. Surgió en la década de 1960 en Estados Unidos. En los últimos años este tipo de proyectos han ido adquiriendo popularidad. Roser Batlle, fundadora de Hispanic Service Learning Network, define esta pedagogía como aquella que vincula el aprendizaje con la responsabilidad social. Consiste en aprender sirviendo a la sociedad y educándola ya que persigue la moralidad y el desarrollo del alumno, aparte de proporcionar un servicio útil. El Centre Promoter d'Aprenentatge Servei sostiene que el aprendizaje servicio es un programa educativo que combina el aprendizaje y el trabajo comunitario en un proyecto bien definido donde los participantes aprenden trabajando en función de sus necesidades reales del entorno para mejorarlo.

3.2. Motivación

El contenido de esta sección se ha obtenido a partir de las explicaciones proporcionadas por los directores del TFG.

Se ha constatado es que incluso cuando un profesor, o un grupo de profesores, quieren ofrecer un servicio y saben cómo integrarlo en una asignatura, y existe un socio comunitario interesado en recibir un servicio de la naturaleza propuesta no es nada fácil crear un proyecto ApS por las diferencias de perspectiva entre las dos partes. Es decir, no basta con que exista una oferta y una demanda para crear un proyecto. Se definió la noción de partenariado para nuestra aplicación precisamente con el fin de:

- Dejar claro a las dos partes participantes que hace falta bastante trabajo conjunto previo para entender las necesidades de la otra parte y poder definir y sacar adelante un proyecto ApS.

- Proporcionar un contexto en el que el profesor y el socio comunitario pueden trabajar en la definición de uno o más proyectos ApS (Siendo una de las tareas implicadas .ªnalizar la compatibilidad de la oferta y la demanda").

- Dar soporte de carácter informático a la comunicación entre las dos partes. Este soporte orientado a la definición de uno, o varios proyectos ApS

El ApS tiene poco valor como tema de investigación de para profesores de Informática. Es por ello que los intentos de crear una aplicación informática de soporte al ApS sean bajo el contexto de un TFG/PFC. El desarrollo de una herramienta de soporte al ApS tampoco tiene cabida fácil en las convocatorias de proyectos de Inovación Docente de las universidades españolas, que suelen estar orientadas a experiencias con métodos y/o herramientas en asignaturas concretas, y a los que, por otro lado, destinan muy pocos fondos.

Emmanuel Partmentier [6] realizó un PFC de Ingeniería Industrial de la UPM (*Universidad Politécnica de Madrid*), que fue el primer intento de desarrollar una aplicación de soporte de PFC de cooperación. Este proyecto dio como resultado una aplicación basada en el CMS Plone. Pero esta aplicación era un prototipo y no contó con suficientes recursos humanos para su despliegue y administración. Por ello nunca llegó a utilizarse. Ángeles Manjarrés, profesora de la UNED y Simon Pickin dirigieron el TFG del grado de Informá-

tica de la UNED realizado por Sol Lozano Lorenzo [10]. Este TFG retoma la idea de [6]. En este trabajo se desarrolló una aplicación web basada en el CMF Drupal. Esta aplicación constituyó un avance hacia el objetivo, pese a ser básica y tener restricciones técnicas que la limitaban funcionalmente. La iniciativa de Sol Lorenzo Lozano [10] fue retomada con el TFG del grado de Educación Social de la UNED [8] y por dos TFG del grado de Informática de la UNED [7] y [5], estos dos últimos dirigidos por Ángeles Manjarrés y Simon Pickin, codirector del TFG actual.

El TFG [8] fue realizado por Natalia Rodríguez y dirigido por Juan García Gutierrez, profesor de la UNED y se focalizó en especificar una aplicación informática de soporte de proyectos ApS. En [7] , trataron de partir de la aplicación basada en Drupal sin éxito debido a que este CMF resultó ser más cerrado de lo esperado. Por ello se desarrolló otra aplicación partiendo de cero basada en el *stack* MyEAN *(MySQL-Express-Angular-Node)* ,descrita brevemente en [9]. En [5] continuaron con el desarrollo de la aplicación [7] pero se basaron en el *stack* MEAN *(MongoDB-Express-Angular-Node)*. Este cambio fue debido a la familiaridad del desarrollador con estas tecnologías. La aplicación desarrollada está pensada para dar soporte a la comunidad de ApS, pese a que en [7] y en [5] se referencian como ApS Virtual. En el último TFG [4] se puede destacar el cambio de la base de datos a MySQL, ya que se consideró un error del alumno del TFG [5] usar MongoDB en un sistema cuyo modelo de datos subyacente es marcadamente relacional.

3.3. Punto de partida

Anteriormente a este TFG, ya se habian desarrollado otros proyectos dentro de este ámbito como pueden ser el realizado por David Jiménez del Rey [5] o más recientemente, el desarrollado por nuestros compañeros de la facultad Daniela Boldureanu, Victoria Gnatiuk y Jesús Sánchez [4]. El presente TFG se apoya en este último proyecto. En el último proyecto se desarrollaron modelos de dominio y de datos muy completos para el subsistema ofertas-

demandas. Se cambió la base de datos de MongoDB a MySQL y se crearon algunos de los DAO necesarios para la comunicación con la base de datos ya que en el TFG desarrollado antes que el suyo [5] se hacía desde el controlador. También se definió los formularios de creación de oferta y de creación de demanda. Por último, se definió y se implementó una primera versión de un algoritmo de *match* entre ofertas y demandas, dejando los resultados en una tabla nueva en la base de datos. Nosotros nos hemos centrado en las funcionalidades que estaban pendientes como son la creación, modificación, borrado, filtrado y listado de las ofertas, demandas y partenariados, tanto en la parte *frontend* como *backend*. Además, el proyecto tenía dependencias en algunas funcionalidades con una base de datos tipo MongoDB y con un servicio S3 de AWS para guardar las imagenes de los usuarios.

Capítulo 4

Elección de tecnologías

Se ha vuelto terriblemente obvio que nuestra tecnología ha superado nuestra humanidad.

Albert Einstein

En este capítulo, se tratarán las tecnologías usadas en el proyecto explicando brevemente qué son y las razones que nos han llevado a utilizarlas.

4.1. Node.js

Node.js es un entorno asíncrono dirigido por eventos. Está basado en el uso de promesas, es decir, funciones que devolverán un resultado en el futuro. Estas se pueden encadenar unas con otras, recibiendo cada una el resultado de la promesa anterior. Esta es una forma de conseguir concurrencia de una forma distinta a la habitual basada en mecanismos de sincronización, como por ejemplo, candados, monitores, o paso de mensajes.

Node.js posee un modelo de concurrencia asíncrono y no maneja hilos de ejecución de forma explícita. Por todo esto, *Node.js* es una elección adecuada para la creación de aplicaciones escalables.

Ya que en los proyectos previos se ha utilizado esta tecnología, continuamos con el desarrollo sobre ella. Una alternativa que se podría haber empleado para este proyecto debido a la baja complejidad de la parte *backend* es un herramienta *low code* (proporciona

un entorno de desarrollo para crear software a través de una interfaz gráfica de usuario) como Apicurio [1], que hubiera reducido el tiempo de desarrollo de esta parte y la capacidad de ampliar el tiempo dedicado a la parte *frontend*.

4.2. Express.js

Express.js es un *framework* basado en Node.js que permite gestionar el servidor de una forma sencilla. Esta tecnología se usó en el anterior trabajo y se decidió mantenerla debido a que todo el equipo había utilizado este *framework* anteriormente.
Existen diferentes *frameworks* que permiten realizar un trabajo equivalente, como pueden ser KOAjs (usa especificación ES6), SailsJS (centrado mas en la creación de APIs mediante el uso del patrón MVC) o fastify (*framework* muy liviano), aunque cada uno de ellos tiene ventajas a largo plazo, ha de tenerse en cuenta la curva de aprendizaje que conlleva el hecho de utilizar estos *frameworks* desconocidos para nosotros.

4.3. Angular

Angular es un *framework* para crear aplicaciones de una sola página, también conocidas como *SPA* o *Single-Page Application*, sigue un patrón *modelo-vista-controlador* aunque no como el patrón clásico, ya que, utiliza el *two-way data blinding* para sincronizar los datos con la vista. Una aplicación *Angular* está formada por *componentes* y *servicios*. Un *componente* está formado por cuatro archivos: un *.html* que contiene la vista a mostrar al usuario, un *.ts* que contiene la lógica del componente y las propiedades a mostrar en la vista, un *.scss* que contiene los estilos del componente y, por último, un *.spec.ts* que contiene los tests del componente. Los componentes usan *servicios*, que son clases que se encargan de acceder a los datos y entregárselos a los componentes.

Se usó esta tecnología porque venía impuesta en el proyecto, a pesar de que ninguno de los autores del trabajo teníamos experiencia con ella. Aunque *Angular* es un *framework* bastante común en areas empresariales, debido a la estructura de los proyectos, así como

la cantidad de soporte que existe sobre esta tecnología, hay otras alternativas como pueden ser ReactJS o VueJS, las cuales en los últimos años vienen siendo las más populares debido a que en el caso de ReactJS, al ser una tecnología hecha por Facebook está en constante crecimiento, mientras que VueJS tiene una curva de aprendizaje bastante inferior a los otros *frameworks* indicados anteriormente.

4.4. Git

Git es un software de control de versiones diseñado por Linus Torvalds. Su propósito es llevar un registro de los cambios realizados en archivos, además de coordinar el trabajo que varias personas realizan sobre los archivos compartidos en un repositorio de código.

Se decidió utilizar esta tecnología porque es ampliamente usada a nivel mundial y porque la mayoría de los miembros del equipo tenían experiencia usándola.

4.5. GitHub

GitHub es un servicio en la nube, usado para alojar proyectos. Utiliza el sistema de control de versiones *Git*.

Se decidió usar *GitHub* debido a que es la plataforma de colaboración más importante y a que todos nosotros ya habíamos trabajado con ella.

4.6. LaTeX

LaTeX es un sistema de composición de documentos que está formado mayoritariamente por órdenes construidas a partir de comandos de *TeX* —un lenguaje «de bajo nivel».

LaTeX es ampliamente usado en la generación de artículos y libros científicos. Es por esta razón por la que se decidió usar *LaTeX* para la realización de esta memoria.

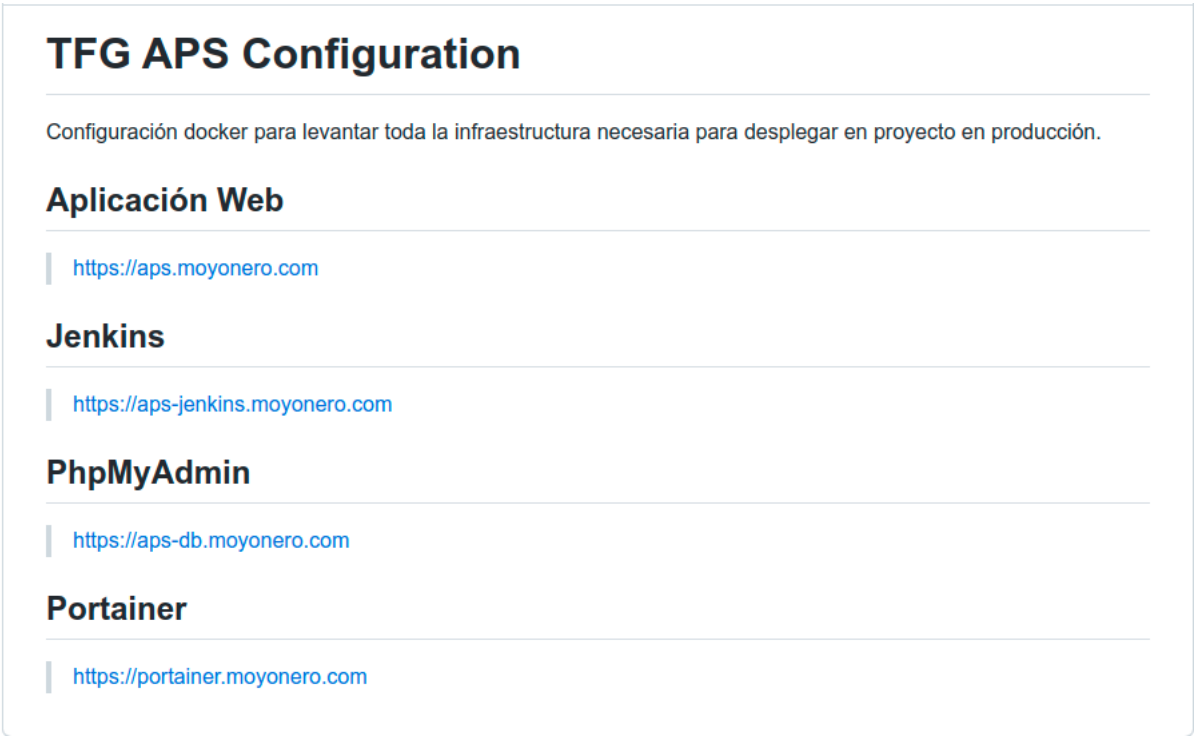


Figura 4.1: *Github: Información del proyecto*

4.7. Docker

Docker es un proyecto de código abierto que permite "empaquetar" una aplicación y todas sus dependencias en contenedores ligeros. Entre sus principales ventajas se encuentran el hecho de que permite un despliegue rápido y que garantiza el aislamiento y la portabilidad de las aplicaciones.

Se apostó por esta tecnología debido a que está ganando fuerza en el ámbito de las tecnologías de la información y a que varios miembros del equipo ya habían trabajado con ella.

4.8. Docker Compose

Compose es una herramienta que permite definir y ejecutar aplicaciones Docker de varios contenedores. Se utiliza un archivo YAML para configurar los servicios de la aplicación.

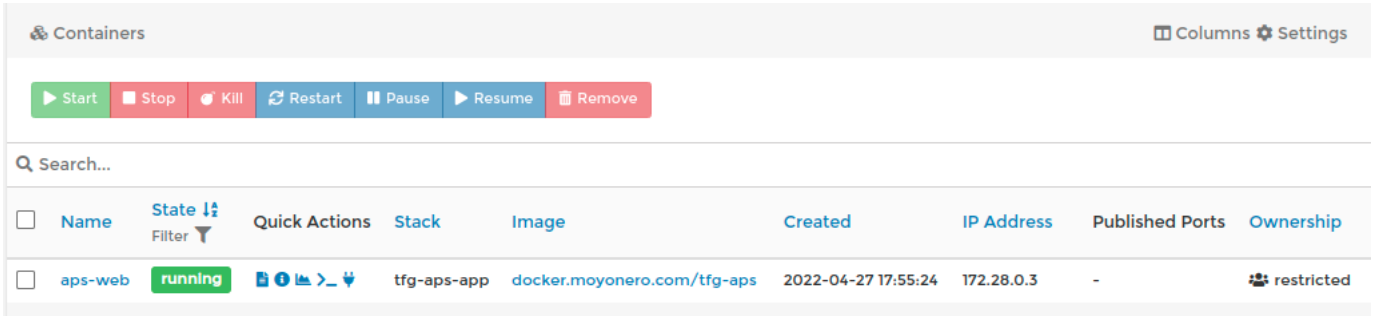


Figura 4.2: *Portainer: Información del portainer del proyecto*

Luego, con un solo comando, se crean e inician todos los servicios.

Se apostó por esta tecnología debido a que varios miembros del equipo ya habían trabajado con ella.

4.9. Portainer

Portainer es una herramienta web de *código abierto* desde la cual se pueden gestionar contenedores *Docker*. Esta herramienta permite administrar contenedores de forma remota o local.

Se decidió usar esta tecnología porque algunos miembros del equipo, en concreto Jonathan y David, tenían experiencia en su uso.

4.10. SonarQube

SonarQube es una plataforma que permite evaluar la calidad del código fuente. Es software libre y usa diversas herramientas de análisis estático de código como *Checkstyle*, *FindBugs* o *PMD* para obtener métricas que pueden ayudar a mejorar la calidad del código. Informa sobre errores potenciales, código duplicado, etc. Se integra con *Maven*, *Ant* y herramientas de integración continua como *Jenkins*.

4.11. phpMyAdmin

phpMyAdmin una herramienta escrita en *PHP* que sirve para manejar la administración de, tanto *MySQL* como *MariaDB* a través de páginas web.

Se usó esta tecnología porque todos los miembros del equipo ya la habían utilizado anteriormente, además de por su facilidad de uso.

4.12. Modelio

Es una herramienta de modelado UML y BPMN de código abierto. Se usó esta herramienta debido a que permite realizar modelos y procesarlos, permitiendo así generar código.

Otras ventajas de esta herramienta son la relativa sencillez de su uso y que al ser código abierto es gratis. En cambio, la mayoría de aplicaciones de modelado tienen una licencia de pago.

4.13. Jenkins

Jenkins es un servidor de automatización de *código abierto* escrito en Java. *Jenkins* ayuda en la automatización de parte del proceso de desarrollo de software mediante *integración continua*.

La *integración continua* consiste en hacer integraciones automáticas de un proyecto lo más a menudo posible para, de esta forma, poder detectar fallos lo antes posible. Se entiende por integración la compilación y la ejecución de pruebas de un proyecto.

Cada despliegue de la aplicación verifica que el código fuente compila satisfactoriamente y obtiene un archivo *Docker* el cuál, si todo es correcto, se desplegará en el servidor correspondiente.

La principal razón que llevó al uso de *Jenkins* es que es una tecnología que está teniendo un gran impacto en los equipos de desarrollo a nivel mundial. Otra razón fue que varios miembros del equipo ya habían trabajado con ella con anterioridad.

4.14. Heimdall Application Dashboard

Heimdall Application Dashboard es, como indica su propio nombre, un panel para mostrar aplicaciones web. Permite añadir *links* asociados con un icono de una manera gráfica. Una vez configurado, crea un panel con un botón para cada *link* añadido en una sola pantalla. Es una gran solución para organizar enlaces a aplicaciones web. En nuestro caso lo hemos utilizado para centralizar el acceso a las diferentes urls relacionadas con nuestra aplicación.

4.15. Overleaf

Overleaf es una herramienta de publicación y redacción colaborativa en linea basada en LaTeX. Brinda la conveniencia de un editor LaTeX fácil de usar con colaboración en tiempo real y la salida totalmente compilada producida automáticamente en segundo plano a medida que se escribe.

Se utilizó esta herramienta debido a su facilidad de uso y aprendizaje y a que no es necesario ningún tipo de instalación.

4.16. Trello

Trello es una aplicación para administrar proyectos mediante una interfaz web. Emplea el sistema japonés *Kanban*, que incorpora tarjetas virtuales y tablones para el registro y coordinación de las actividades.

Se decidió utilizar *Trello* debido a que ya lo habíamos usado en anteriores asignaturas del Grado, además de por su facilidad de uso.

4.17. Compodoc

Compodoc es una herramienta para la generación de documentación en proyectos Angular. No es necesario ningún servidor ni subir código a ninguna web, ya que, la documentación

se genera totalmente *offline*. Una posible alternativa a *Compodoc* es *Dgeni*, aunque esta última es más compleja de utilizar.

Se ha utilizado *Compodoc* para generar la documentación de este proyecto debido a su fácil instalación a través de *npm*.

Capítulo 5

Especificación de requisitos

En software, muy raramente partimos de requisitos con sentido. Incluso teniéndolos, la única medida del éxito que importa es si nuestra solución resuelve la cambiante idea que el cliente tiene de lo que es su problema

Jeff Atwood

Un proyecto ApS puede realizarse en una amplia variedad de campos. Teniendo en cuenta esto necesitamos desarrollar un modelo de aplicación que sea utilizable para cualquier proyecto ApS, indistintamente del campo sobre el que se esté realizando.

5.1. Análisis

En esta sección se describirán los diagramas de creación y ciclo de vida de un proyecto ApS. También se ha realizado un modelo de Dominio para representar el subsistema proyectos-activos.

5.1.1. Diagrama creación proyecto

Se ha realizado un diagrama BPMN para representar los posibles casos que pueden ocurrir para que un proyecto se cree. En este diagrama se pueden distinguir dos tipos de usuarios:

- Profesor: Es el usuario que se va a encargar de crear el partenariado, el que realiza esta acción es asignado automáticamente como Profesor responsable del mismo. Este rol de Profesor responsable puede ser transferido a otro profesor que forme parte del mismo proyecto.

- Socio comunitario: Es la entidad que presenta una demanda de servicio.

Un partenariado es una asociación entre una universidad y una entidad social en la que la demanda presentada por la entidad va a ser respondida por la oferta de la universidad. A una creación de partenariado se puede llegar a partir de tres situaciones:

- Situación 1: Un profesor crea una oferta de servicio, la cual se almacena en el sistema. El sistema tiene programada una operación de *matching* entre ofertas y demandas ya almacenadas en el sistema. Por tanto, si existe una demanda que casa con una oferta o viceversa, se produce un *match* y el profesor es notificado para que cree el partenariado.

- Situación 2: Un profesor busca en el sistema la lista de demandas de servicio, y decide respaldar una demanda de servicio ya existente. Tras esto procede a la creación del partenariado. Figura 5.2.

- Situación 3: Un socio comunitario busca en el sistema la lista de ofertas de servicio y decide aceptar una oferta de servicio, tras esto el profesor es notificado para que cree el partenariado.Figura 5.2.

Tras realizarse la creación del partenariado, el socio comunitario responsable de la demanda de servicio que ha encontrado *match* es notificado de que se ha creado el partenariado con su demanda y este tiene la opción de validarlo para comenzar con las negociaciones, o declinarlo con lo cual el partenariado se cancela. Figura 5.3. La negociación se lleva a cabo entre el profesor responsable, el socio comunitario y posiblemente otros profesores que vayan

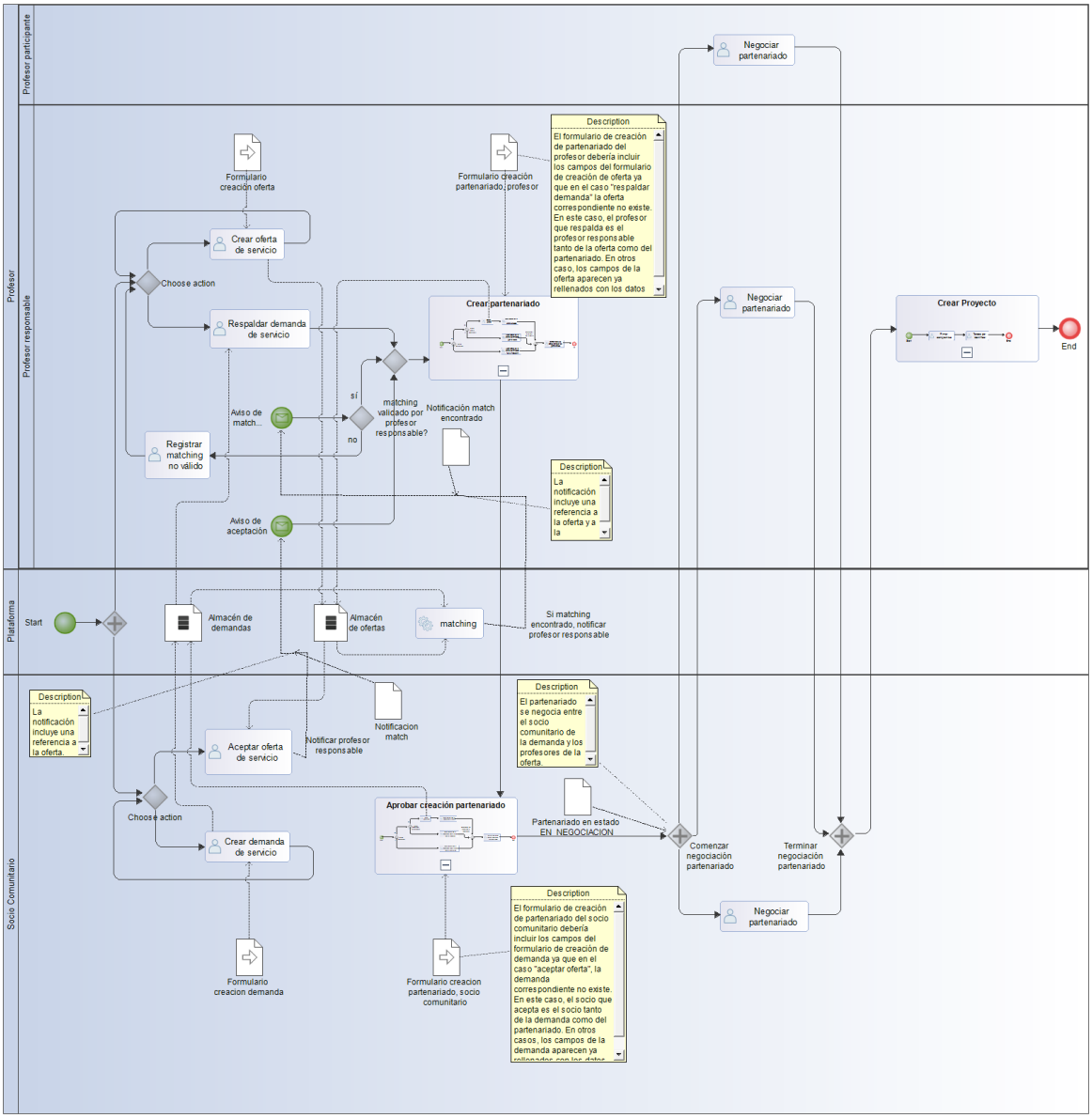


Figura 5.1: *Modelio: Diagrama de posibles casos en la creación de un proyecto*

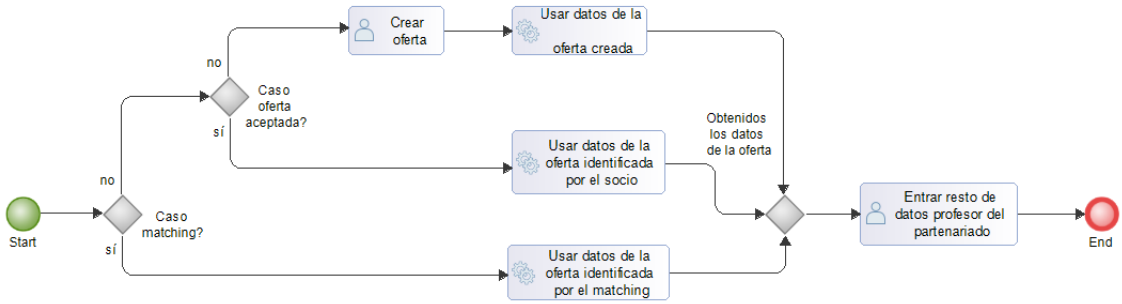


Figura 5.2: *Modelio: Diagrama subproceso creación de un partenariado*

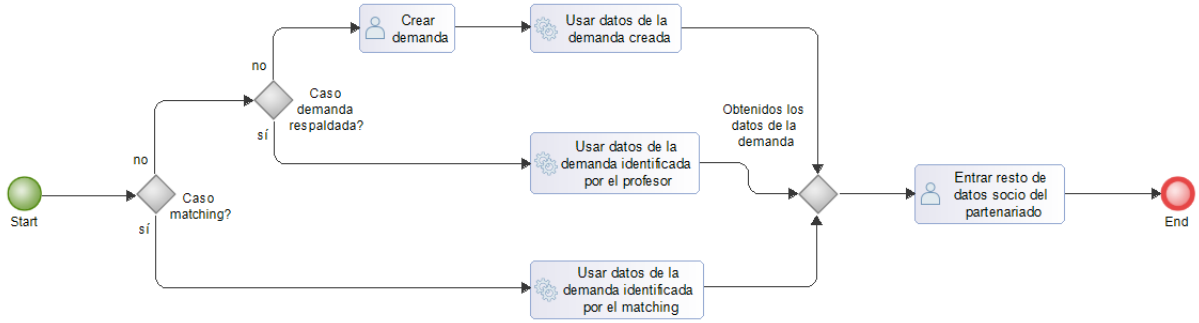


Figura 5.3: *Modelio: Diagrama subproceso de aprobación de la creación de un partenariado*

a participar en el proyecto. Una vez que las negociaciones han finalizado, se establecen los requisitos y se crea un proyecto en estado EN CREACION.

5.1.2. Ciclo de vida de un proyecto

El ciclo de vida (figura 5.4), comienza tras la creación del proyecto a partir de un partenariado, pero todavía no está abierto a alumnos hasta que se defina la rúbrica del proyecto. El Socio Comunitario y los Beneficiarios sugieren criterios de evaluación y el Profesor define la rúbrica, que es la medida con la que se va a evaluar a los alumnos que formen parte del proyecto. La rubríca contiene, entre otras cosas, la información sobre cuáles de los usuarios (de entre los posibles: Socio Comunitario, Beneficiario, Estudiante y Promotor) van a despempeñar un papel en la evaluación de los alumnos, así como la información sobre cómo y

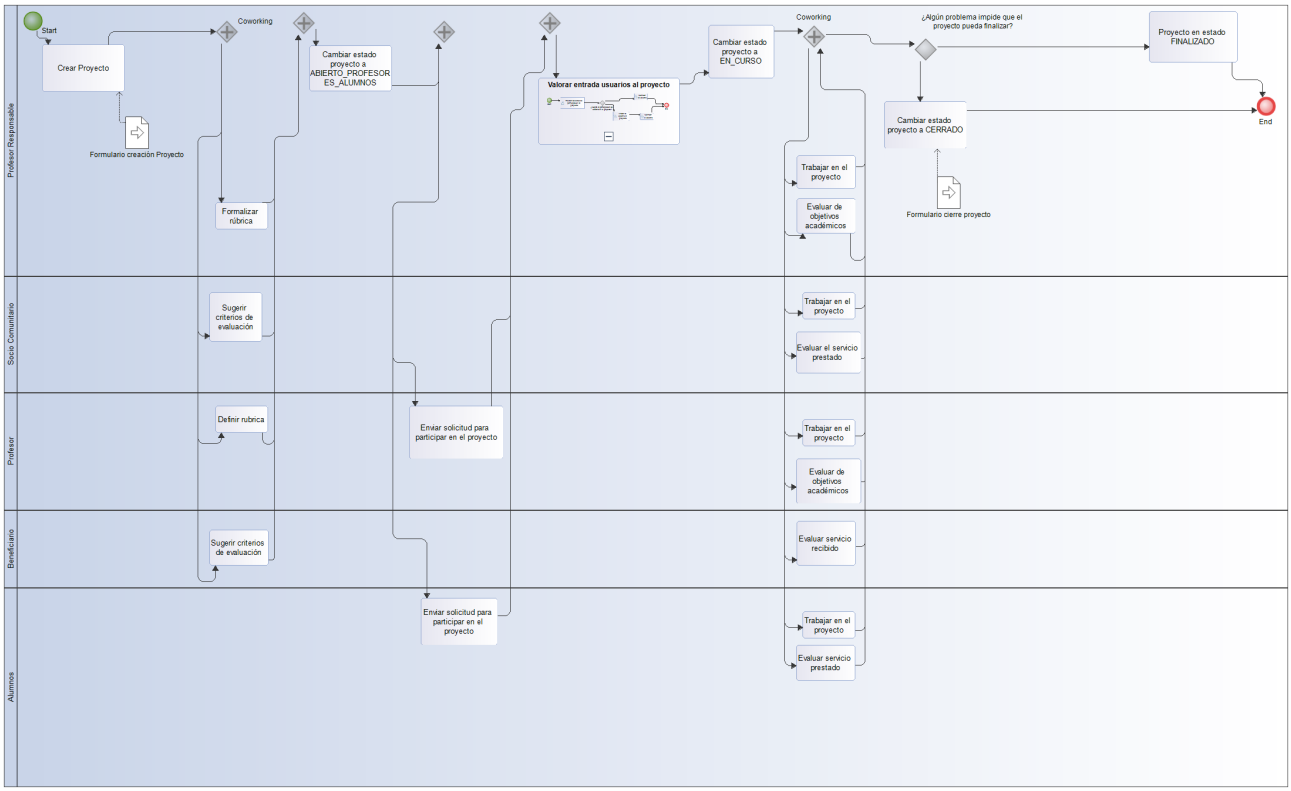


Figura 5.4: *Modelio: Ciclo de vida de un proyecto ApS*

cuándo deben intervenir en esta evaluación.

Tras esto el proyecto pasa al estado ABIERTO_PROFESORES_ALUMNOS y tanto profesores como alumnos pueden participar en el proyecto, pero la decisión de aceptar o rechazar al usuario dentro del proyecto la realiza el profesor responsable, (ver figura 5.1.2). Una vez que el profesor responsable considera que el proyecto está listo para comenzar, el proyecto pasa a estar EN_CURSO. A medida que se trabaja en el proyecto los alumnos son evaluados por los usuarios previamente definidos en la rúbrica.

Si en algún momento hay que cancelar el proyecto por alguna causa, puede cerrarse el proyecto y este pasará a estado CERRADO.

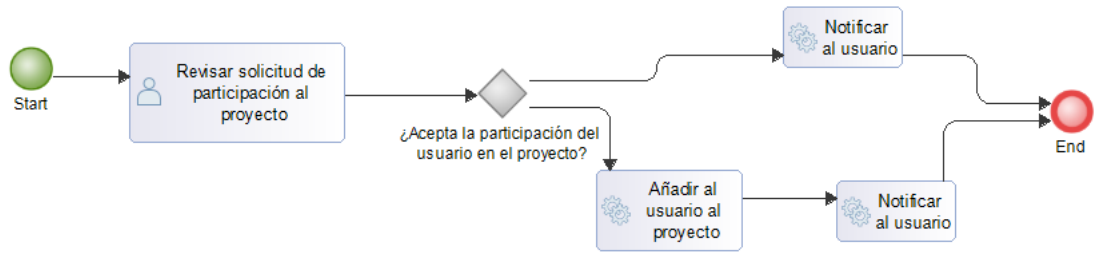


Figura 5.5: *Modelio: Diagrama valoración de entrada de usuarios en el proyecto.*

5.1.3. Modelo de Dominio

El modelo de Dominio del subsistema proyectos-activos comparte algunas entidades del modelo de Dominio del subsistema ofertas-demandas (,figura 5.7,) realizado en el TFG del año pasado [4]. También añade nuevas clases y asociaciones.

Se han considerado las siguientes entidades:

- Objetivo: El proyecto puede marcarse objetivos que pueden formar parte de la rúbrica del proyecto a la hora de realizar la evaluación.
- Evento: En el proyecto se pueden crear eventos para realizar reuniones o marcar fechas importantes. Estos eventos son creados por el profesor responsable, el cual puede invitar a cualquier participante del proyecto.
- Participación: Cada estudiante en el proyecto tiene una participación, que también se tiene en cuenta a la hora de realizar la evaluación del estudiante.
- Evaluación: Se obtiene una calificación a partir de la rúbrica establecida en el proyecto.
- Beneficiario: Un beneficiario es la entidad que va a recibir el servicio proporcionado por el proyecto ApS.

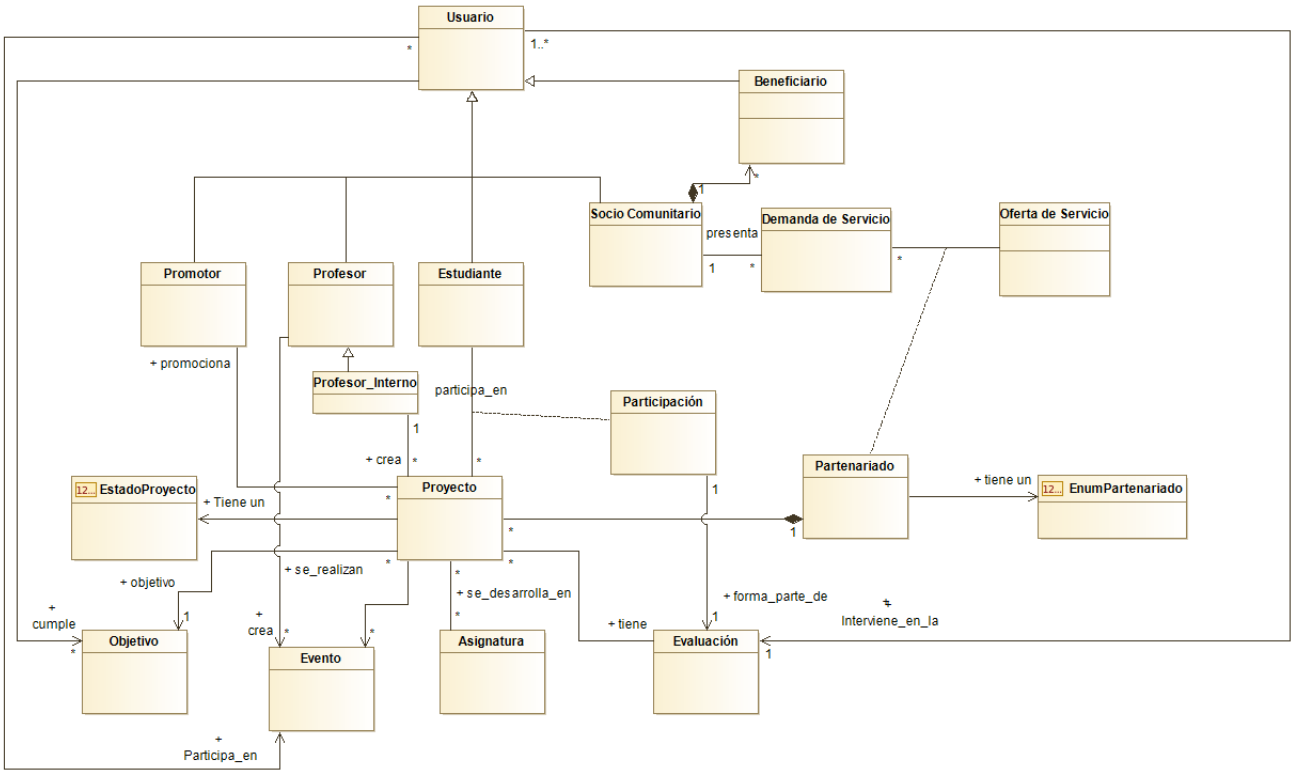


Figura 5.6: *Modelio: Modelo de Dominio*

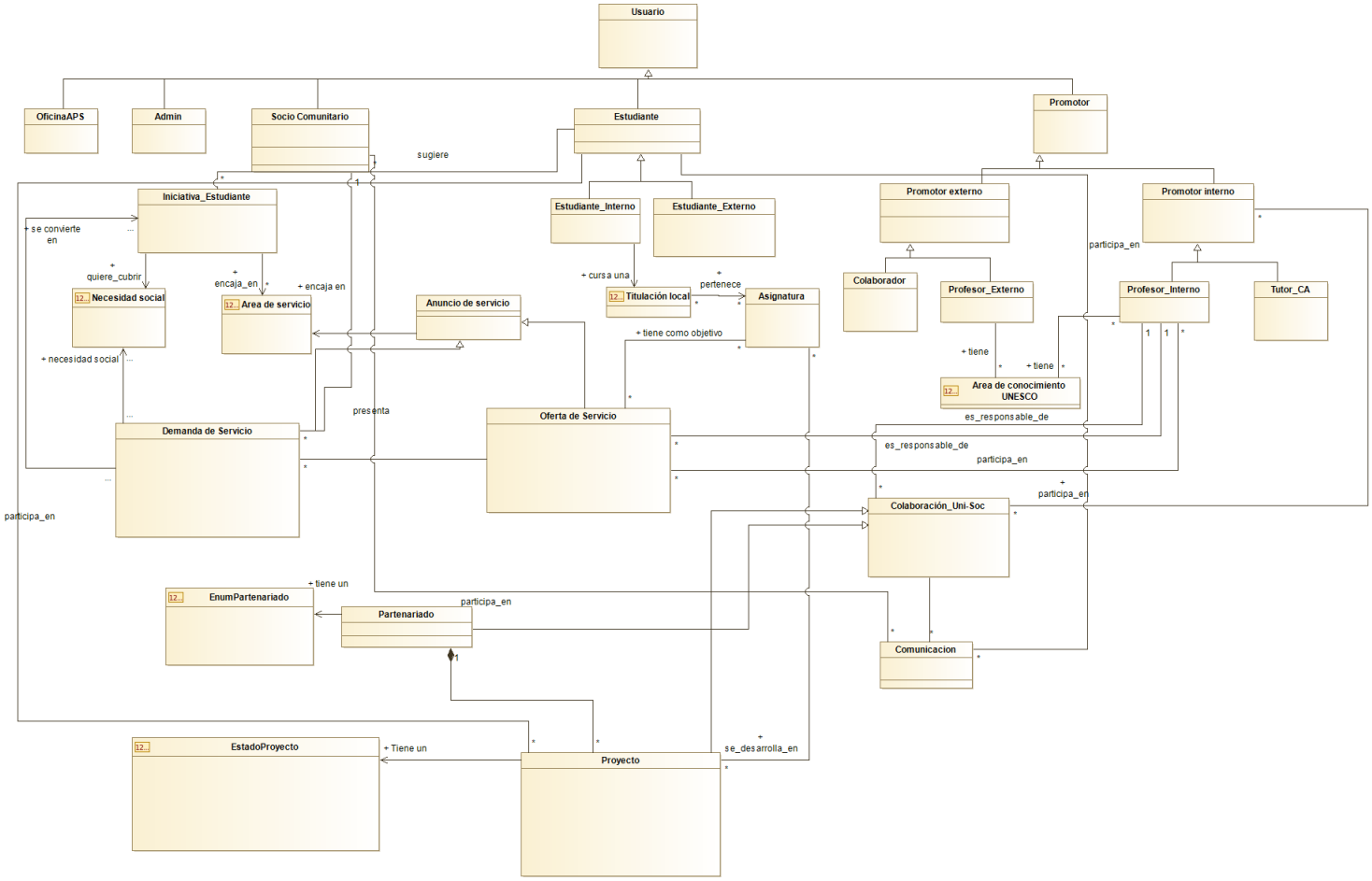


Figura 5.7: *Modelio: Modelo de Dominio del TFG anterior*

5.2. Diseño

En esta sección se detalla el modelo de Datos realizado para el subsistema proyectos-activos partiendo del modelo de Dominio realizado en el apartado anterior.

5.2.1. Modelo de Datos

Al igual que el modelo de Dominio, hemos partido del modelo de Datos del subsitema ofertas-demandas realizado en el TFG del año anterior [4], (figura 5.9) para realizar este modelo de Datos del subsistema proyectos-activos. En este modelo existen clases en común con el subsistema ofertas-demandas. También se añaden nuevas clases y asociaciones. (Figura 5.8). Nuevas entidades:

- Objetivo: Dentro de un objetivo se contemplan los atributos de Título, Descripción y fechacumplimiento.
    - Título: Nombre que se le va a dar al objetivo.
    - Descripción: Datos más detallados acerca del propio objetivo
    - fechacumplimiento: Fecha en la que el objetivo se da por finalizado.
- Evento: Un evento puede tener Nombre, una descripción para dar mas datos acerca del evento y las fechas de inicio y final del mismo.
    - Nombre: Título del evento.
    - Descripción: Datos más detallados acerca del evento.
    - fechainicio: Fecha de comienzo del evento.
    - fechafin: Fecha de finalización del evento.
- Evaluación: Dentro de la entidad evaluación se ha considerado el parámetro calificación.
    - Calificación: Nota asociada al alumno del proyecto.

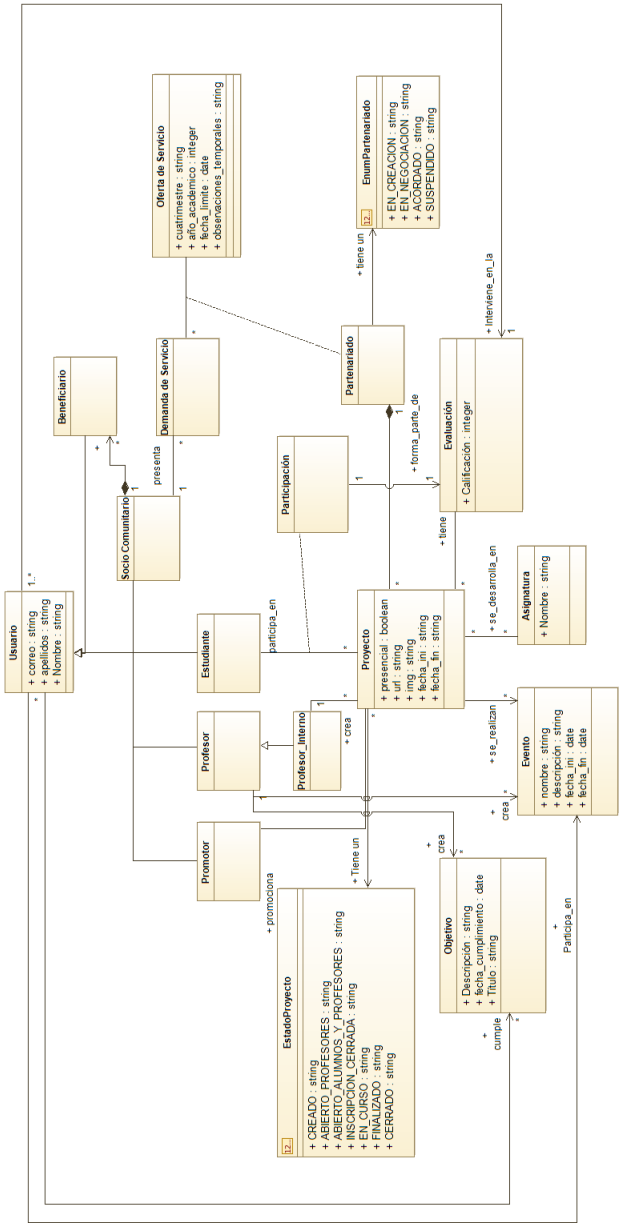


Figura 5.8: *Modelo: Modelo de Datos*

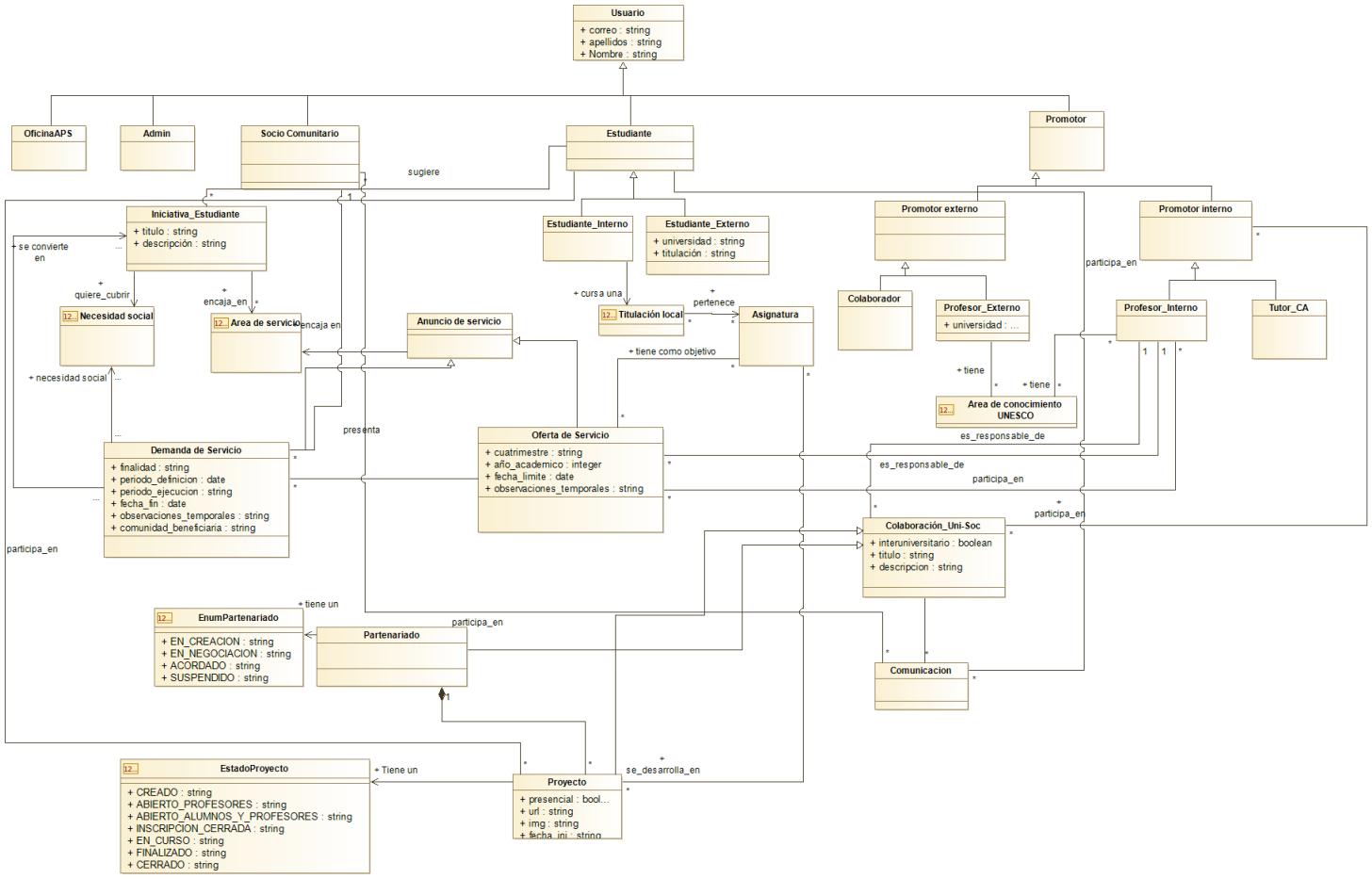


Figura 5.9: *Modelio: Modelo de Datos del TFG anterior*

Capítulo 6

Implementación

Ley de Alzheimer de la programación: si lees un código que escribiste hace más de dos semanas es como si lo vieras por primera vez

Dan Hurvitz

Tras llevar a cabo la limpieza del código fuente relativo al anterior TFG, se pasó a implementar las vistas requeridas en el presente proyecto y que faltaban en el anterior TFG [4], relacionadas con las ofertas, las demandas y los partenariados. Para ello, tuvimos que aprender Angular, *framework* de JavaScript, que requiere un vasto conocimiento para el desarrollo de las vistas, ya que, estas se componen de dos partes: los objetos que se encuentran en el componente para almacenar y gestionar los datos de la vista y la visualización que se encuentra en la plantilla.

La implementación llevada a cabo en el presente proyecto mejora y completa el subsistema de ofertas-demandas. Con respecto al subsistema de proyectos-activos no se pudo implementar nada.

6.1. Página de listado de ofertas

La primera página que implementamos fue la del listado de ofertas, ya que necesitábamos una forma de poder visualizar las ofertas creadas por los profesores y almacenarlas en la aplicación. Al principio se implementó una tabla para mostrar todos los datos relativos a las

ofertas, pero se terminó cambiando, debido a que el formato de la misma no nos terminó de convencer. Se decidió sustituirla por un diseño usando las *cards* de *Bootstrap*. Estas tarjetas están formadas por una cabecera donde aparece el título de la oferta correspondiente, en el cuerpo se encuentra parte de la descripción y en el pie se encuentra un botón para mostrar la información detallada de esa oferta. Además, esta página posee a su izquierda una zona de filtrado, mediante la cual un usuario podrá filtrar las ofertas según su título, el profesor responsable de la misma, los *tags* generados en la oferta, el área de implementación a la que pertenece y, por último, el cuatrimestre y año académico objetivo.

Unas imágenes de esta vista se pueden ver en las figuras 6.1 y 6.2. A continuación, se expondrá lo relacionado con la vista de la información detallada de una oferta.

6.2. Página de información detallada de una oferta

Esta página se creó para poder mostrar toda la información relacionada con una oferta. Como se ha comentado anteriormente, al principio, tanto en la vista del listado de ofertas como en la de demandas, había una tabla que se decidió sustituir por las *cards* de *Bootstrap*. En cada *card* se añadió un botón que redirige a la página que tratamos en este apartado. En esta vista, mostramos el título y la descripción completa de la oferta, el cuatrimestre objetivo y año académico. También aparecen dos fechas más: la de creación de esa oferta concreta y su fecha límite, además de los datos del profesor creador de la oferta y un campo para posibles observaciones a tener en cuenta. Una imagen de esta página puede verse en la figura 6.3.

6.3. Página de listado de demandas

En segundo lugar se implementó la página que muestra las demandas de servicio. En ella se puede observar un listado de todas las demandas de servicio activas en la aplicación Esta vista, al igual que la del listado de ofertas, posee a su izquierda una zona con filtros mediante los cuales el usuario podrá filtrar las demandas según el título de la misma, la

necesidad social que satisface la demanda, el área de servicio en la que se implementa la demanda y, finalmente, según la entidad demandante. Se puede observar una imagen de esta vista en la figura 6.4

6.4. Página de información detallada de una demanda

Esta página se implementó para mostrar toda la información relacionada con una demanda concreta. Esa información está compuesta por el título de la demanda y su descripción detallada, el cuatrimestre y año académico objetivo, la comunidad beneficiaria de la demanda y el objetivo de la misma. Además de estos datos, también se muestran las fechas de inicio y fin para la definición de la demanda y las de ejecución de la misma. Por último, se pueden observar los datos del socio comunitario creador de esa demanda y un campo con posibles observaciones.

6.5. Página de creación de partenariado

Esta vista se implementó para poder crear partenariados a través de una oferta y una demanda.

Cuando un *socio comunitario*, acepta una oferta de servicio, se debe notificar al profesor responsable de esa oferta, para que de esa forma el profesor puede crear el partenariado. Para que esto se produzca, el socio comunitario.....

Al acceder a la vista de información detallada de una determinada oferta de servicio aparece un botón *Crear partenariado* (ver figura 6.5) que lleva a la vista de creación de demandas, como se puede observar en las figuras 6.6, 6.7, 6.8.

6.6. Página personal de cada usuario

Se decidió crear esta página porque se vió la necesidad de crear una vista que aglutinara toda la información de un usuario concreto. Para acceder a esta vista, es necesario pulsar en el botón *Resumen* dentro del desplegable generado al pulsar el nombre de usuario (ver

figura 6.9). En esta vista de resumen, según el tipo de usuario, aparece una determinada información.

- En el caso de ser un *profesor interno* se pueden sugerir ofertas de servicio, además de disponer de la información sobre los partenariados en los que está involucrado y las ofertas de servicio creadas.
- En el caso de ser un *socio comunitario*, aparece la información de sus demandas de servicio.

Las imágenes de esta página se pueden observar en las figuras 6.10, 6.11 y 6.12.

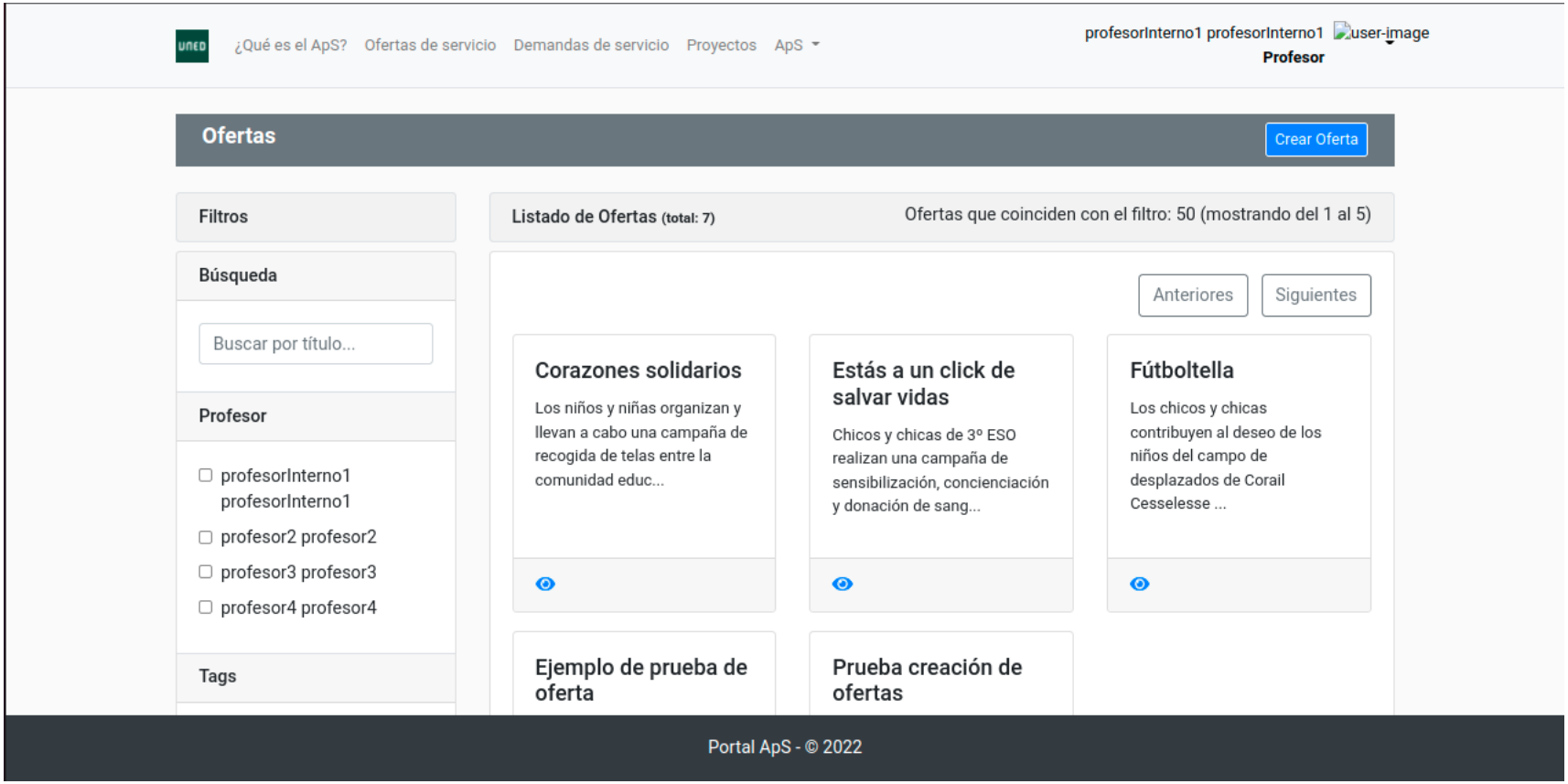


Figura 6.1: *Página listado de ofertas*

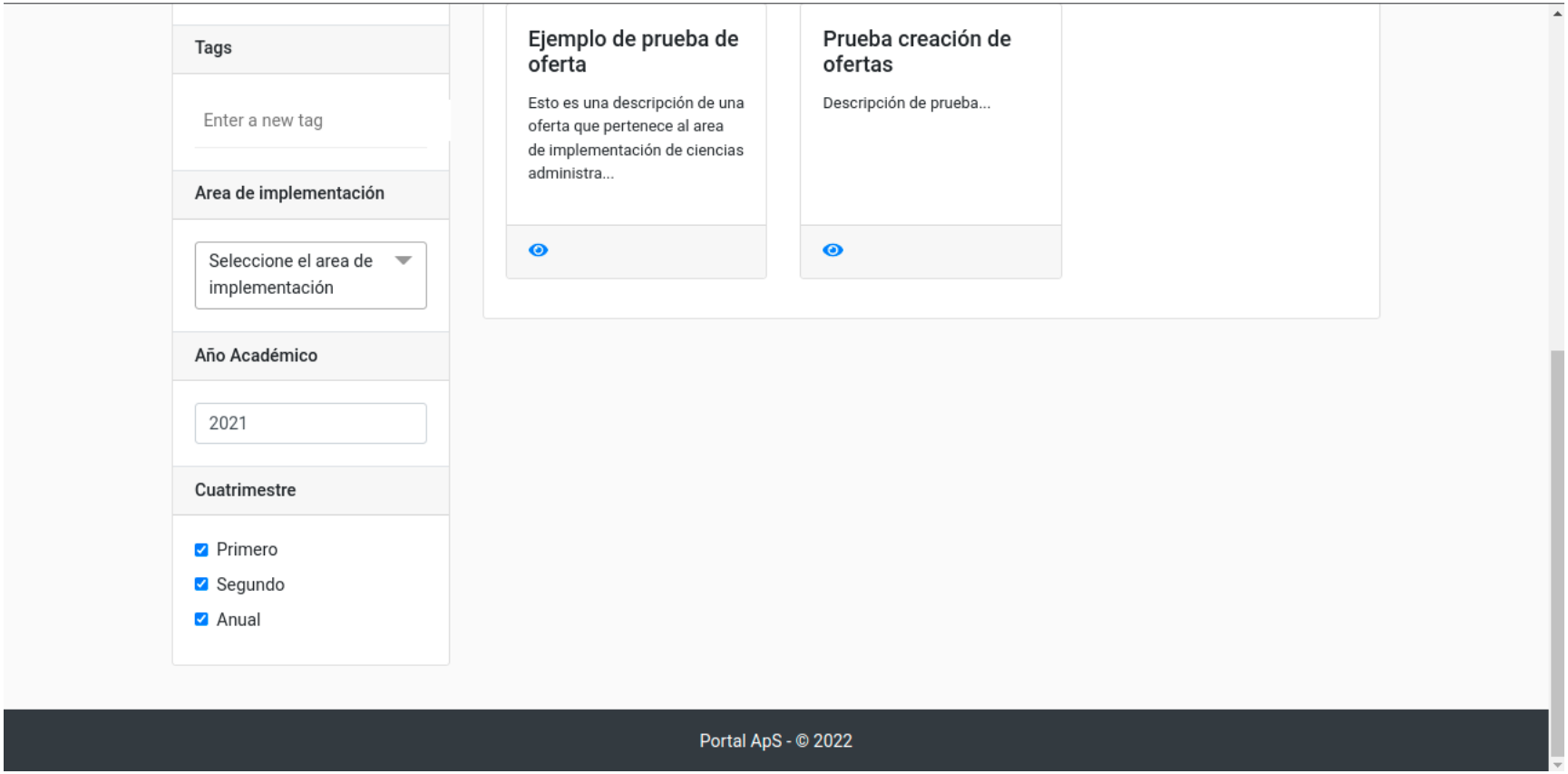


Figura 6.2: *Página listado de ofertas (continuación)*

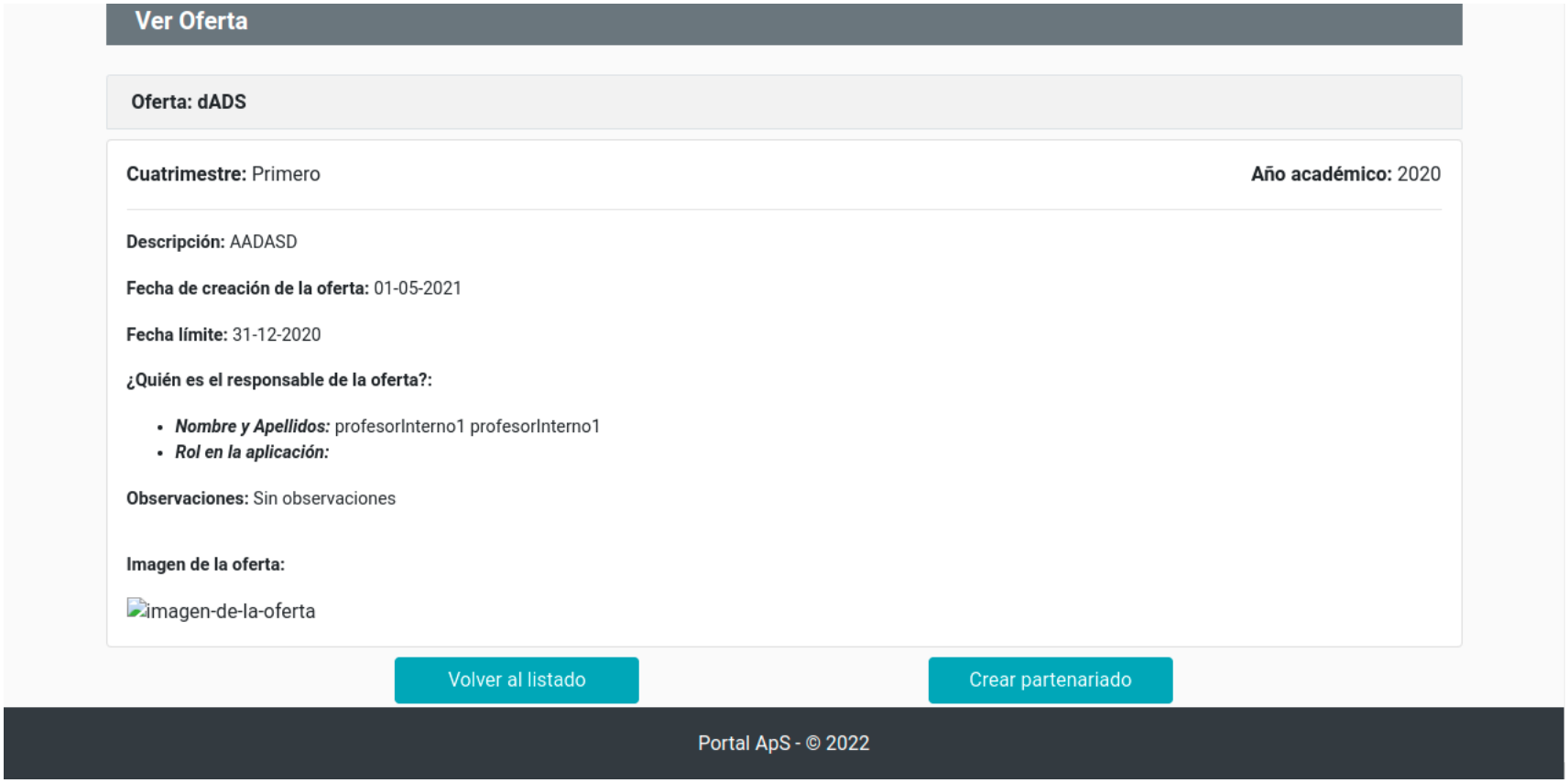


Figura 6.3: *Página información detallada de oferta*

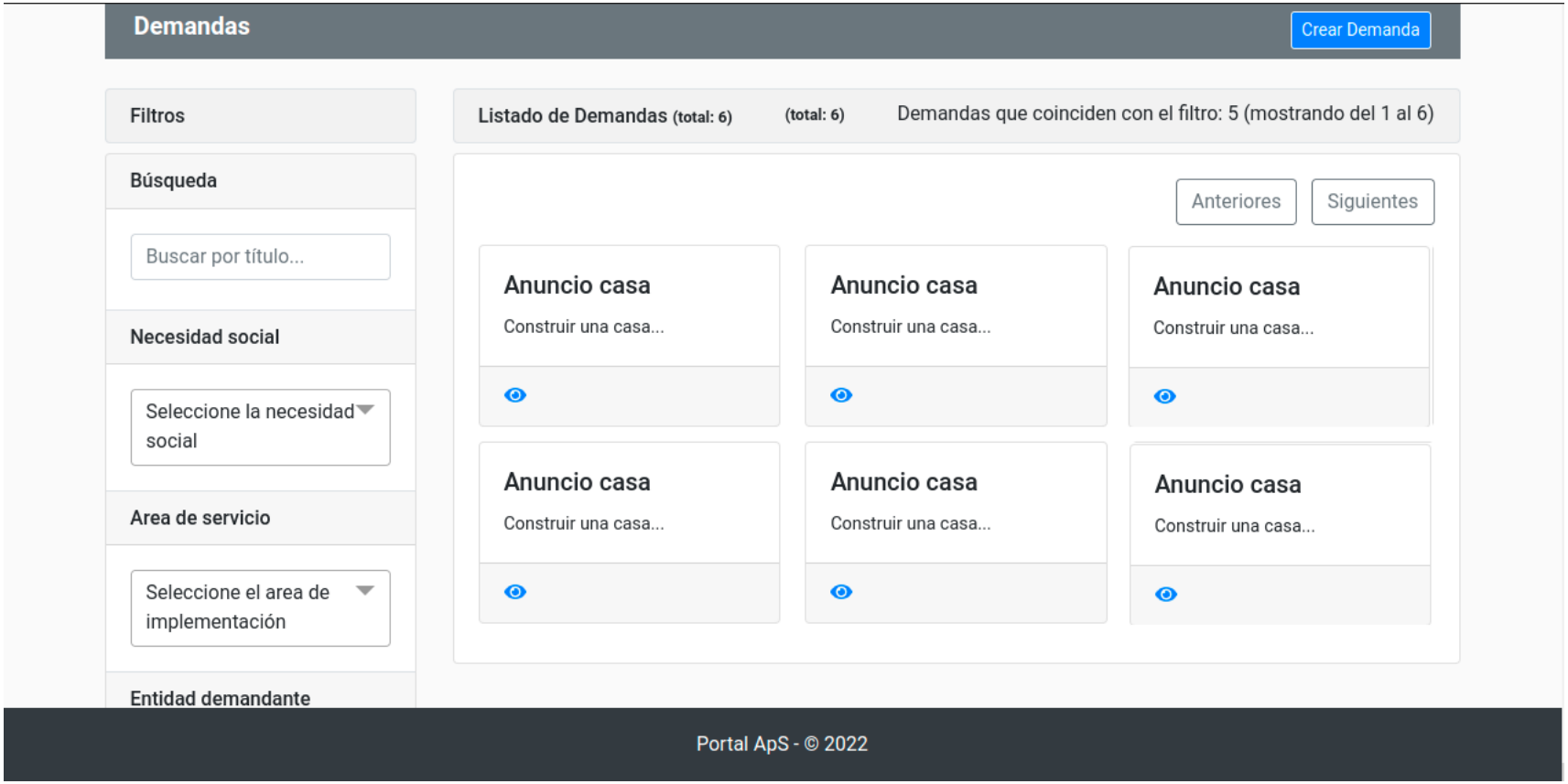


Figura 6.4: *Página listado de demandas*

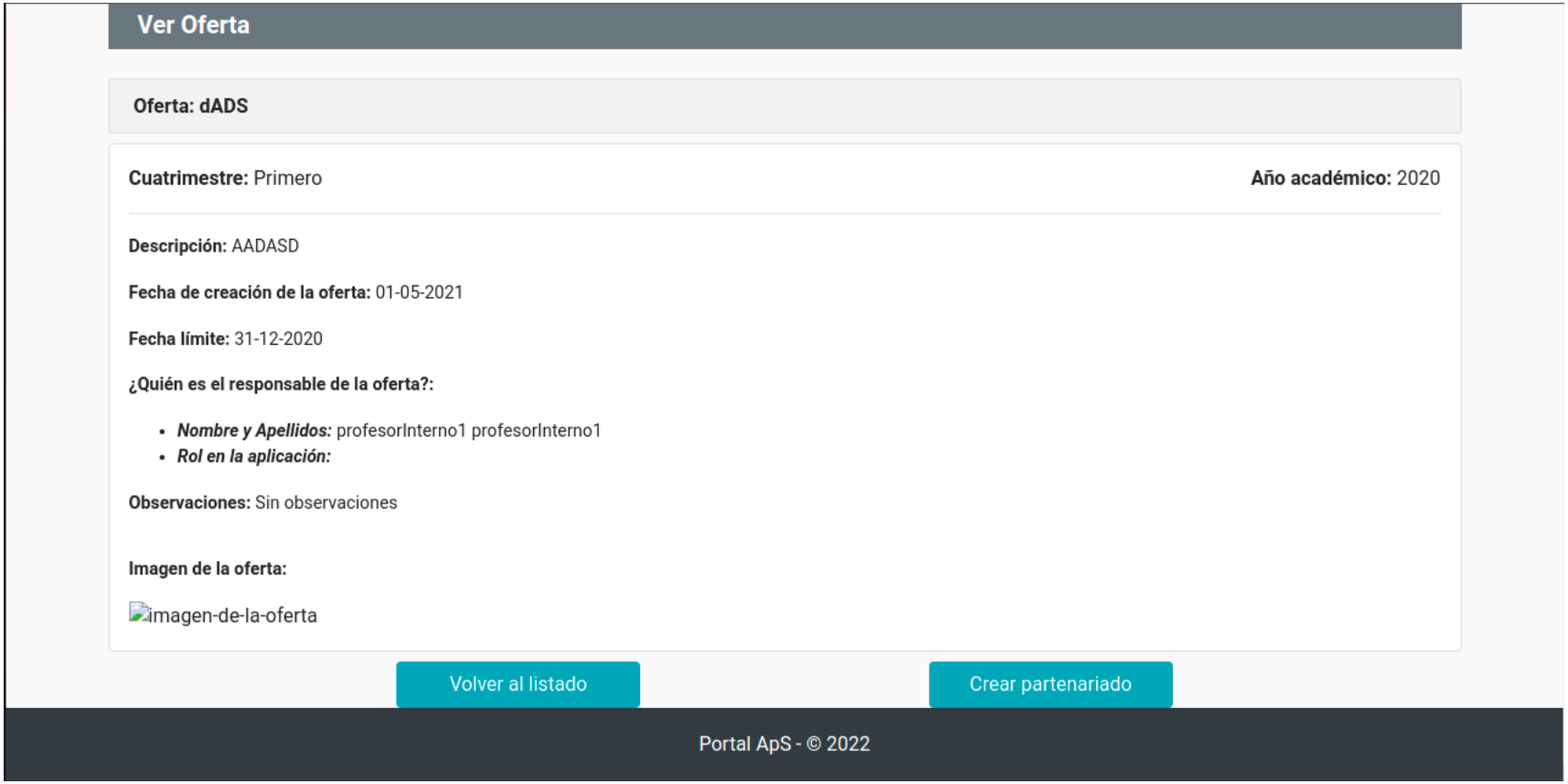


Figura 6.5: *Página creación de partenariado parte 1*

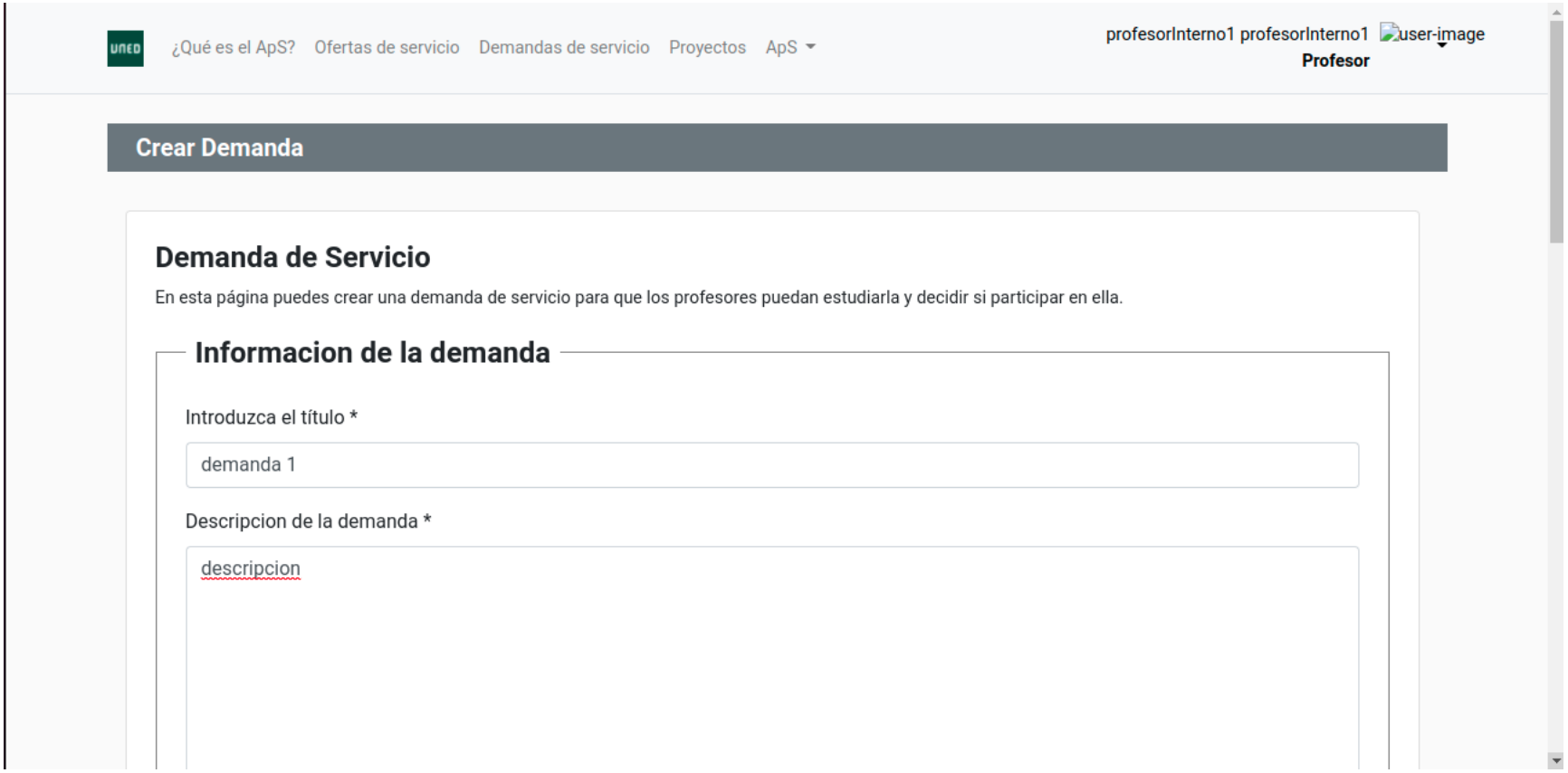


Figura 6.6: *Página creación de partenariado parte 2*

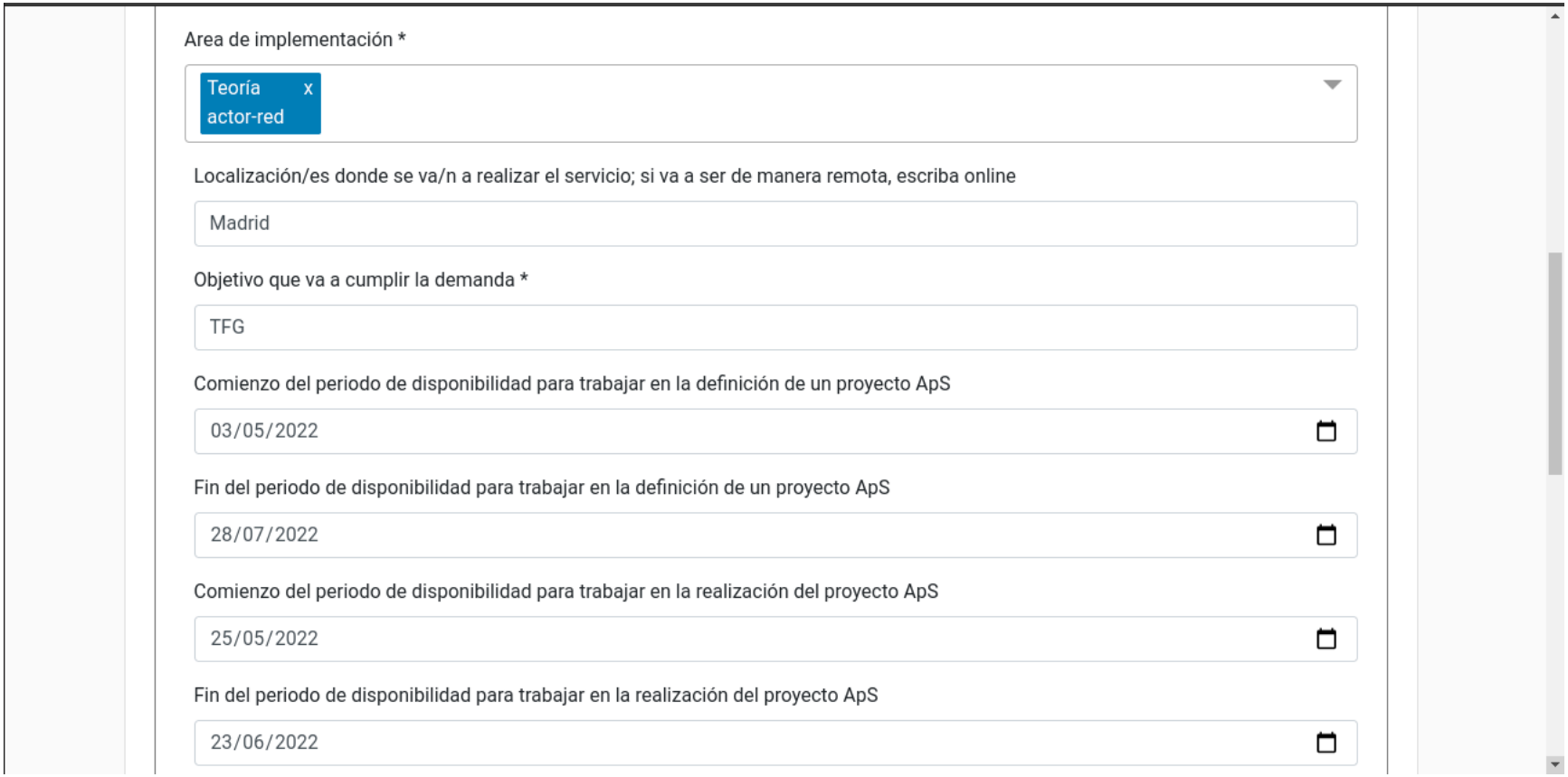


Figura 6.7: *Página creación de partenariado parte 3*

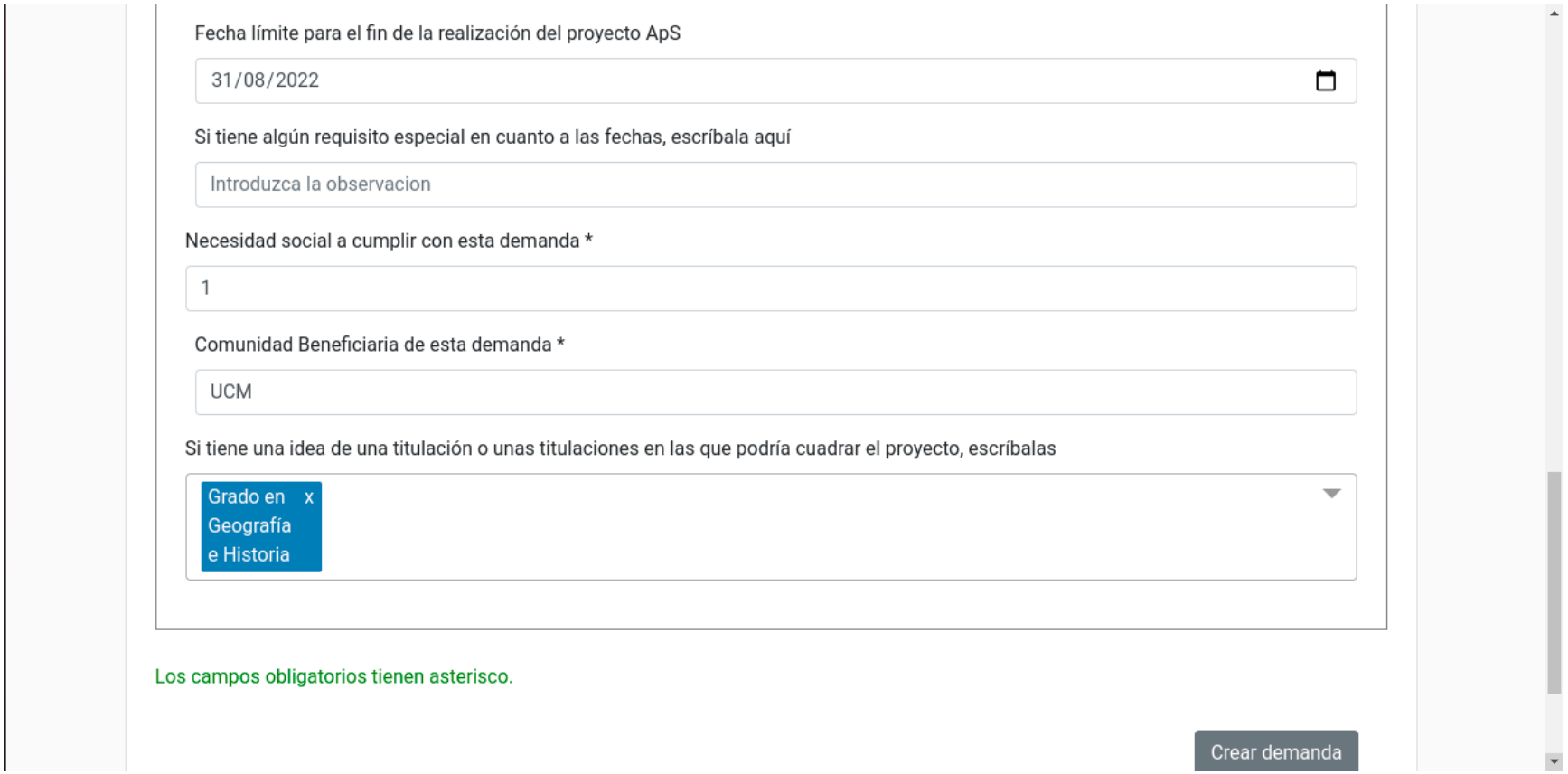


Figura 6.8: *Página creación de partenariado parte 4*

Figura 6.9: *Página personal de usuario parte 1*

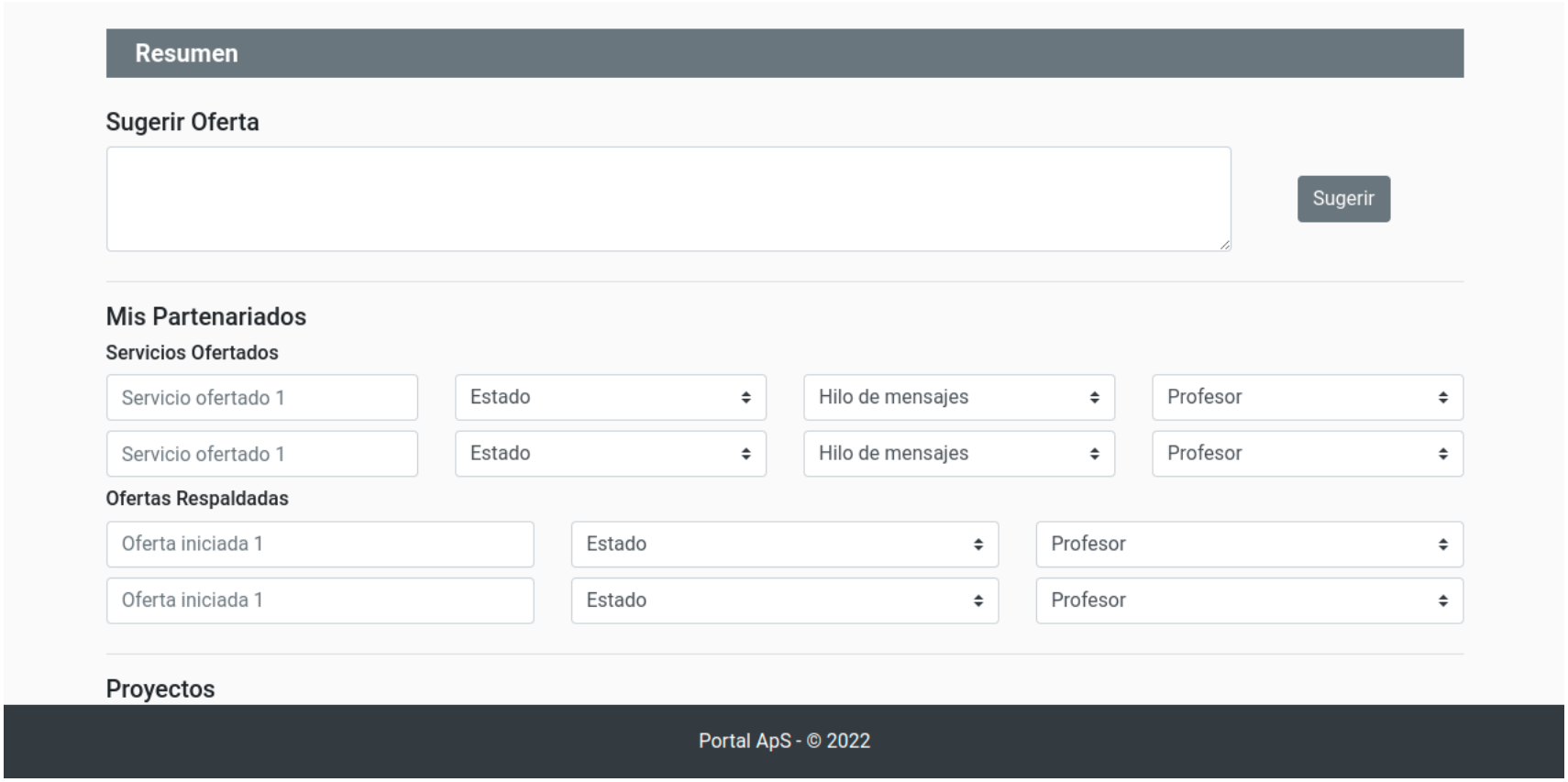


Figura 6.10: *Página personal de usuario parte 2*

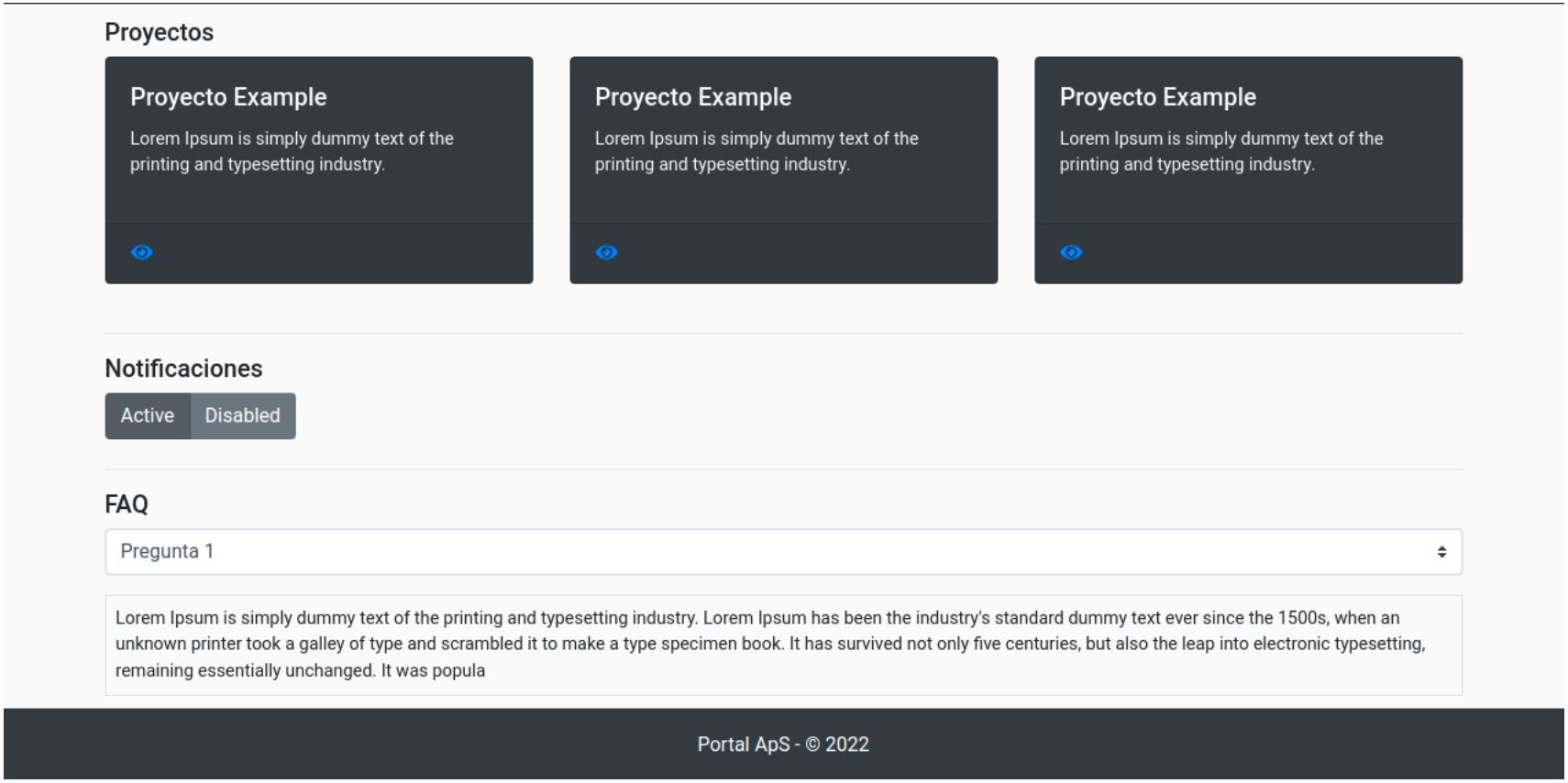


Figura 6.11: *Página personal de usuario parte 3*

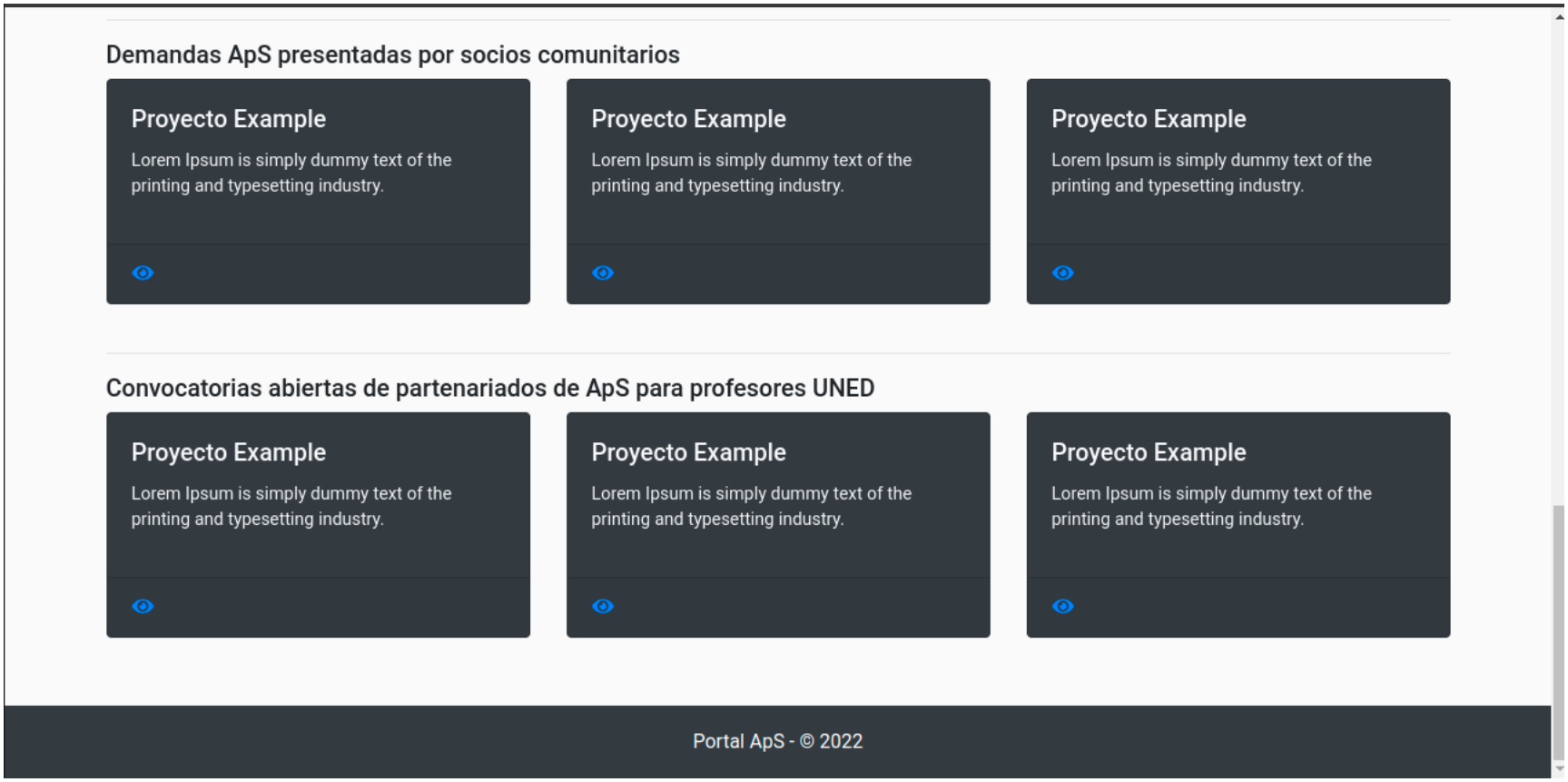


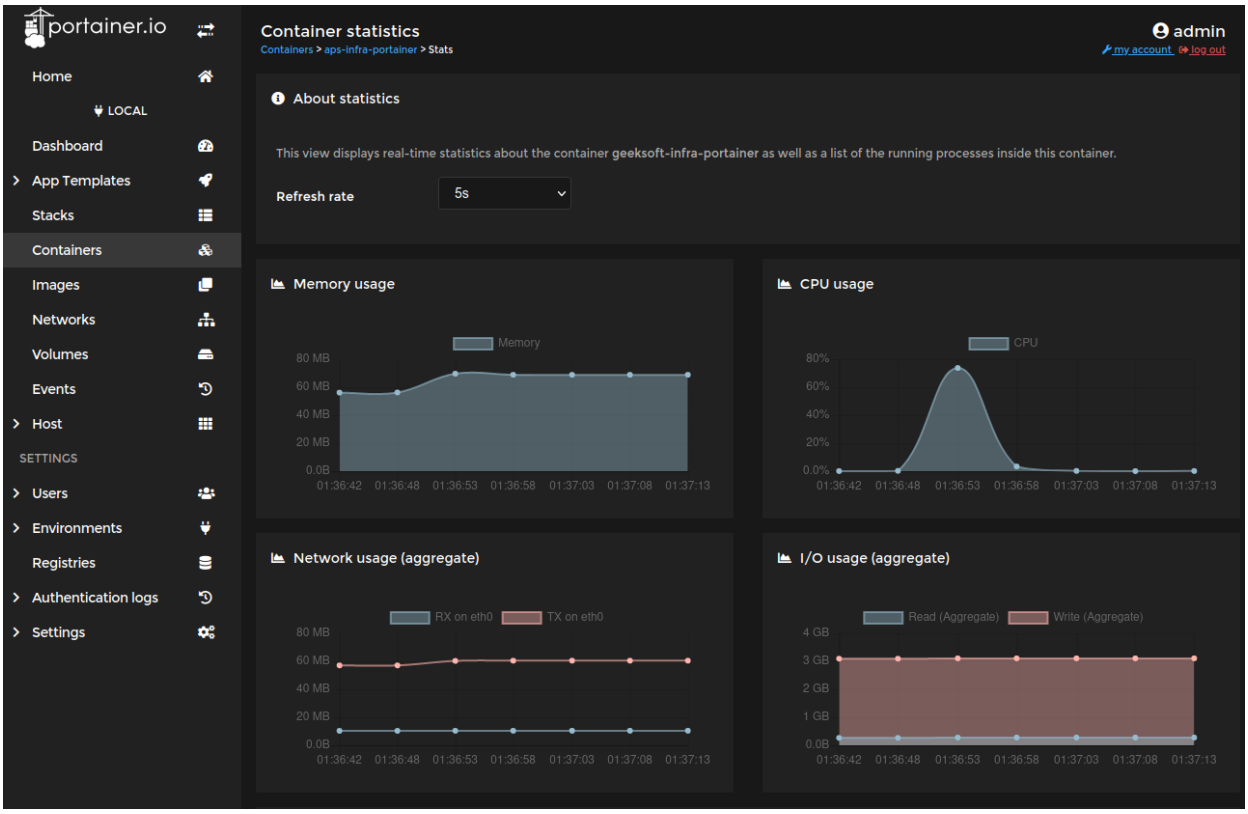
Figura 6.12: *Página personal de usuario parte 4*

6.7. Creación de la infraestructura

Decidimos montar una infraestructura basada en contenedores de *Docker*. Está decisión se debió a la experiencia del equipo usando esta tecnología. Esta infraestructura abarcaría desde la base de datos, hasta una instalación de SonarQube para monitorizar la calidad del código. Los detalles de esta infraestructura se describirán en los siguientes puntos.

6.7.1. Portainer

Hemos utilizado esta herramienta tanto para la monitorización de los diferentes contenedores de *Docker*, como para algunos despliegues puntuales dentro de la propia interfaz gráfica que nos proporciona esta herramienta. De esta manera hemos podido realizar desde pruebas manuales con otro tipo de base de datos, hasta relanzar las instancias que han sufrido algún tipo de percance.



Panel de estadísticas (Portainer)

6.7.2. MariaDB

El proyecto utiliza utiliza una base de datos MariaDB por defecto. Esto puede modificarse ya que utiliza internamente el paquete node Knex, el cual permite que la base de datos destino de los DAO sea independiente a las consultas utilizadas en el desarrollo. Cabe resaltar que entre las posibilidades están PostgreSQL, MySQL, CockroachDB, MSSQL, SQLite3 y Oracle.

6.7.3. PHPmyadmin

Hemos incorporado este gestor de base de datos debido a que el equipo tenía una mayor destreza en el uso de esta aplicación. En el transcurso del desarrollo encontramos una alternativa bastante interesante llamada DBeaver Cloud, la cual proporcionaba una interfaz bastante amigable y la capacidad de dar roles a los usuarios que accedan mediante esta plataforma. Pese a esto, finalmente, se continuó con el uso de *PHPmyadmin*.

6.7.4. Jenkins

Para la integración continúa, nos decantamos por Jenkins debido a las posibilidades que nos brinda. Aunque en el flujo actual del despliegue no se está aprovechando al cien por cien, consideramos que tenerlo ya añadido como parte de la infraestructura, aporta una mayor robustez en la aplicación y permitirá mejorar notablemente la calidad del software, así como evitar los despliegues de versiones que no cumplan con los tests respectivos.

6.7.5. Sonarqube

Está herramienta nos ha permitido el análisis de la calidad del código a medida que íbamos avanzando en el proyecto. Con la configuración actual era bastante sencillo comprobar que no teníamos una mayor duplicidad de código a la encontrada cuando se inició el proyecto de esta manera. Hemos intentado mantener una calidad de código legible y reutilizable.

Capítulo 7

Conclusiones y trabajo futuro

Los finales no son algo malo. Simplemente significan
que algo más está por comenzar.
C. JoyBell C.

En esta sección se comentará la situación actual del proyecto y sus objetivos, además del posible trabajo futuro a realizar.

7.1. Objetivos cumplidos

A continuación se repasarán los objetivos de este trabajo y su completitud:

- *Eliminación de todas las referencias a las iniciativas, debido a que ya no se utilizan en el presente TFG.* Se han eliminado todas las dependencias que se tenían con el modelo iniciativas. En algunos casos no bastaba con eliminarlas, sino que se tuvo que adaptar el modelo ofertas.

- *Limpieza del código fuente referido a MongoDB, sistema de base de datos no utilizado en este proyecto.* En versiones anteriores del proyecto se añadió esta base de datos no relacional en la aplicación. Se han eliminado estas referencias y se ha corregido redirigiendo esta data en la base de datos correcta.

- *Implementación de las vistas para el listado de ofertas y demandas.* Se completaron satisfactoriamente los listados de las ofertas y demandas, tanto en la parte *front-end* como la corrección en *back-end* de las peticiones.

- *Implementación de filtros para las vistas mencionadas anteriormente.* Los filtros por cada uno de los atributos han sido añadidos, incluso se añadió la capacidad de filtrado mediante *tags* los cuales se generan de forma automatica, extrayéndose de las descripciones de las ofertas y las demandas durante su creación.

- *Creación de las vistas de información detallada, tanto para oferta como para demanda.* Se completaron estas vistas y se reutilizaron para la edición de estas mismas entidades.

- *Implementación del formulario para la creación de partenariados a través de una oferta y una demanda.* Se completó el desarrollo del formulario para la creación de los partenariados dependiendo de una ofertas y una demanda, los cuales si no existen se crearán dinamicamente.

- *Implementación infraestructura.* Se ha montado toda la infraestructura (entorno para gestionar y operar el software), lo cuál permitirá tener una mejor calidad de codigo y hacer despliegues más controlados en producción.

- *Corrección de bugs encontrados en el anterior TFG.* Una vez que nos adentramos en el código, nos percatamos que había algunos puntos que no estaban del todo claros en los desarrollos previos, lo cual nos llevo a solventar los problemas que nos ibamos encontrando a medida que avanzaba el desarrollo.

- *Ampliación del modelo de dominio y modelo de datos.* Hemos tenido que realizar algunas modificaciones en las tablas, ya que, como indicamos anteriormente, algunos campos hacían referencia a la base de datos MongoDB o a campos inexistentes en el sistema.

- *Definición de ciclo de vida de un proyecto.* Se han realizado dos diagramas BPMN de un ciclo de vida de un proyecto desde su creación hasta que este se cierra o termina, aparte de los diagramas BPMN de los subprocesos.

7.2. Problemas encontrados

A continuación, se describirán los problemas y dificultades encontradas durante la realización del proyecto.

El principal problema ha sido la complejidad de trabajar con Angular debido a la poca experiencia del equipo con esta tecnología. El equipo empezó a familiarizarse con esta tecnología antes del inicio del curso pero, aún así, al ser una tecnología compleja, surgieron contratiempos durante el desarrollo del proyecto. A pesar de esta dificultad, se ha conseguido aprender a usar esa tecnología. Se ha llegado a la conclusión de que Angular es una gran elección a la hora de implementar *Single-page Applications*.

Otro problema encontrado fue el hallazgo de código referido a MongoDB que se debía eliminar, debido a que no se utiliza este sistema de base de datos en el proyecto, lo que hizo que se retrasarán las tareas requeridas en este proyecto.

Por otra parte, otro problema surgido a mitad del proyecto fue el abandono de una compañera del proyecto.

7.3. Trabajo futuro

A pesar de que se han completado la mayoría de los objetivos propuestos para este proyecto, a continuación se listan posibles mejoras a realizar de cara a un futuro:

- *Implementación de proyectos.* Esta funcionalidad estaba bloqueada hasta terminar la creación de partenariados. En el trabajo futuro, al tener lista la creación de partenariados, se podrá comenzar con este desarrollo.

- *Creación de tests unitarios, integración y aceptación.* La parte de creación de tests es fundamental para tener una aplicación robusta, por lo que toda la infraestructura creada alrededor de la aplicación ayudará a que estos tests se aprovechen tanto en la integración continua, como cuando se hagan las comprobaciones de calidad de software mediante Sonarqube.

- *Implementación de notificaciones dentro de la aplicación.* En nuestra investigación, para encontrar una alternativa que cumpla con los requerimientos solicitados, consideramos que *Rocketchat* permite cubrir esta funcionalidad, al ser una solución ya desarrollada, agregándole que es de código abierto. El trabajo consistirá en la configuración de la integración mediante *webhooks* (un *webhook* es un sistema de sincronización e intercambio de datos entre aplicaciones. Básicamente, los *webhooks* son retrollamadas HTTP de cliente). En RocketChat, hay dos tipos de integraciones las entrantes y las salientes. En las primeras se hace una llamada a una url interna de RocketChat, se ejecuta un script. El segundo caso, se produce cuando se realiza un petición externa a RocketChat.

- *Página personal para cada usuario.* Esta funcionalidad no se pudo terminar, debido a que necesitábamos la implementación de la funcionalidad relativa a los proyectos. Aunque la parte *frontend* se encuentra medianamente desarrollada para cada uno de los roles de la aplicación, el desarrollo del *backend* se encuentra pendiente.

- *Evaluación del proyecto.* La evaluación de un proyecto ApS es una parte muy importante del mismo. En esta evaluación se valora el proyecto como producto final o servicio proporcionado. Al finalizar la realización del proyecto todos los miembros del mismo, (o los seleccionados por el profesor responsable), realizan una evaluación del proyecto a partir de una rúbrica que se definió en el momento de la creación del proyecto.

- *Rúbrica del proyecto.* La rúbrica del proyecto se establece en el punto de creación del proyecto entre todos los integrantes del proyecto en ese momento (profesores, socio

comunitario y/o beneficiarios). Esta rúbrica determina los objetivos del proyecto. Lo más primordial al tener en cuenta los criterios de esta rúbrica es saber qué demanda social es la que se quiere cubrir y qué necesidades tiene esta.

Capítulo 8

Conclusions and future work

Soñar con el futuro es mucho mejor que lamentarse por el pasado.

Toba Beta

This section will discuss the current status of the project and its objectives, as well as the possible future work to be done.

8.1. Objectives completed

- *Elimination of all references to initiatives, because they are no longer used in this FDP.* All the dependencies that existed with the initiatives model have been eliminated, in some cases it was not enough to be eliminated, it had to be adapted to the offers model.

- *Cleanup of the source code referring to MongoDB, a database system not used in this project.* In previous versions of the project, this non-relational database was added to the application, these references have been removed and it has been corrected by redirecting this data to the correct database.

- *Implementation of the views for the list of offers and demands.* The lists of offers and demands were satisfactorily completed, both in the frontend and the correction of the requests in the backend.

- *Implementation of filters for the views mentioned above.* The filters for each of the attributes have been added, including a filter for tags which are automatically generated.

- *Creation of detailed information views, both for offer and demand.* These views were completed and reused for editing these same entities.

- *Implementation of the form for the creation of partnerships through an offer and a demand.* The development of the form for the creation of a partnership depending on an offer and a demand, which if they do not exist will be created dinamically, was completed successfully.

- *Infraestructure implementation.* All the infrastructure for the project has been set up, which will allow for better code quality and more controlled deployments in production.

- *Correction of bugs found in the previous FDP.* Once we get into the code we realized that there were some points that were not entirely clear in previous developments, which led us to solve the problems that we were finding as development progressed.

- *Extension of the domain model and data model.* We have had to make some modifications to the tables, since, as indicated above, some fields they referenced the MongoDB database or non-existent fields in the system.

- *Definition of the life cycle of a project.* Two BPMN diagrams of the life cycle of a project from its creation until it is closed or terminated, apart from the BPMN diagrams of the subprocesses have been made.

8.2. Problems found

Next, the problems and difficulties encountered during the project will be described.

The main problem has been the complexity of working with Angular due to the lack of team's experience with this technology. The team began to familiarize themselves with this

technology before the start of the course but still setbacks arose during the development of the project. Despite this difficulty, it has been possible to learn to use this technology. it has arrived to the conclusion, that Angular is a great choice when implementing Single-page Applications.

Another problem encountered was the discovery of code referring to MongoDB that was due delete because this database system is not used in the project, which did that the tasks required in this project will be delayed.

On the other hand, another problem that arose halfway through the project was it abandonment by a team member.

8.3. Future work

Although more of the objectives proposed to this project has been completed, possible improvements to be made in the future are listed below:

- *Implementation of projects.* This functionality was blocked until the end of the creation of partnerships. In the future work, when having the part of partnerships ready this development can be started.

- *Creation of unit, integration and acceptance tests.* The test creation part to have a robust application is essential, so the entire infrastructure created around the application will help these tests take advantage both in the continuous integration, such as when performing software quality checks through Sonarqube.

- *Implementation of notifications within the application.* In our investigation for find an alternative that meets the requested requirements, we consider that Rocketchat allows to cover this functionality and being an already developed solution, adding to this that it is open source, the work will consist of configuring the integration via *webhooks* (a *webhook* is a system of synchronization and data exchange between applications. Basically, *webhooks* are HTTP callbacks from the client). In RocketChat,

there are two types of integrations: inbounds and outbounds. In the first ones, a call is made to an internal RocketChat url, a script is executed. The second case occurs when an external request is made to RocketChat.

- *Implementation of the summary information form for each user*. This functionality could not be finished because we needed the implementation of projects, although the frontend part is moderately developed for each of the application roles, the backend development is pending.

- *Project evaluation*. The evaluation of an ApS project is a very important part of it. In this evaluation, the project is valued as a final product or service provided. At the end of the project, all its members (or those selected by the responsible teacher), carry out an evaluation of the project from a rubric that was defined at the time of project creation.

- *Project rubric*. The project rubric is set at the point of project creation between all the members of the project at that moment (teachers, community partners and/or beneficiaries). This rubric determines the objectives of the project. The most important thing to keep in mind of this rubric is what social demand is the one that you want to cover and what needs it has.

Capítulo 9

Contribuciones

La vida exige a todo individuo una contribución
y depende del individuo descubrir en qué consiste.
Viktor Frankl

A continuación, se detallan las contribuciones al presente proyecto de cada uno de los componentes del grupo ordenados alfabéticamente.

9.1. Sergio Arroyo Galán

La primera fase del proyecto fue una fase de investigación acerca del *Aprendizaje Servicio* para saber a qué teníamos que hacer frente.

En la segunda fase se realizaron pruebas manuales para comprender el funcionamiento de la aplicación. Estas pruebas se realizaron de forma conjunta entre los miembros del equipo. Durante estas pruebas se descubrieron algunos *bugs* que se corrigieron más tarde. Además de esto, se creó el repositorio de Github que alojaría los avances realizados en este proyecto.

En la tercera fase, nos encargamos del aprendizaje de *Angular*, tecnología impuesta para el desarrollo del proyecto, y dividir las tareas a implementar entre los miembros del equipo. La primera de ellas realizada por Sergio fue la creación del formulario para el listado de ofer-

tas. Este formulario era necesario desde un principio, ya que, necesitábamos alguna forma de visualizar las ofertas disponibles en la aplicación. Para ello, Sergio creó el componente *ofertas*, al hacer esto se crearon los ficheros *ofertas.component.html*, *ofertas.component.scss*, *ofertas.component.spec.ts* y *ofertas.component.ts*. Al hacer esto, también se tuvo que modificar el fichero de rutas, para añadir la nueva ruta. Para poder acceder a las ofertas de la base de datos, Sergio modificó el servicio *OfertaService* para crear los métodos de acceso a la base de datos al igual que el fichero *ofertas.js* en la parte *back-end*. En la vista del listado de ofertas, Sergio también se encargó del filtrado por *área de servicio* para lo que creó una lista desplegable con todas las áreas de servicio disponibles. Esto, al igual que antes, llevó a crear su correspondiente método en el fichero *ofertas.component.ts* haciendo uso del servicio mencionado anteriormente añadiendo el correspondiente método y ha modificar el *back-end*, creando los métodos correspondientes en el *DAO*.

Como se ha comentado anteriormente en la memoria, esta vista al principio, contenía una tabla con la información de las ofertas pero se decidió cambiar a las *cards* de *Bootstrap*, Sergio se encargó de ello, creándolas directamente en la vista. Finalmente, se decidió convertir las *cards* en un componente propiamente dicho para mejorar la reusabilidad, de eso se encargó David.

Después de esto, Sergio pasó a implementar el listado de demandas, creando el componente *demandas*, esto llevo a la creación de los ficheros *demandas.component.ts*, *demandas.component.html*, *demandas.component.scss* y *demandas.component.ts*. En esta vista, se implementaron los filtros por *necesidad social* y *área de implementación*, para ello se siguieron los pasos realizados en los filtros de las ofertas.

Al finalizar los listados, se decidieron crear los formularios para mostrar la información detallada de las ofertas y las demandas. Las *cards* que se habían creado anteriormente contenían el *título* de cada oferta/demanda y la *descripción* correspondiente. Se decidió mostrar solo un fragmento de la *descripción* para así mostrar la totalidad de la misma acompañada del resto de datos en un formulario por separado. Es por esta razón, que se

crearon los formularios para mostrar la información detallada de cada oferta y demanda, para acceder a los mismos se añadió un botón en el pie de la *card*.

Sergio se encargó de la creación de ambos formularios para ello creó los respectivos componentes *ofertas-ver* y *demandas-ver*.

Finalmente, Sergio también se encargó de eliminar las referencias a las *iniciativas*, ya que, estas ya no se iban a utilizar en este proyecto.

9.2. Yrving David Conde Cubas

La primera fase del proyecto fue una fase de investigación de este tipo de proyectos *Aprendizaje Servicio* para conocer a nivel usuario que es lo que se necesitaba y como afrontarlo a medida que se avanzaba en el desarrollo.

En la segunda fase nos centramos en realizar pruebas manuales y en lograr que el proyecto se ejecutará correctamente, ya que no disponiamos de una guía que nos proporcionará las versiones necesarias para el despliegue en local, asi como los pasos a seguir para poder continuar el desarrollo. A medida que ibamos avanzando en este aspecto, encontramos algunos bugs los cuales se solucionaron en ese momento o se creó la tarea en *Trello* para solucionarlo en un futuro.

En la tercera fase, nos encargamos del aprendizaje de *Angular* ya que no teníamos experiencia con esta tecnología. Luego de que David hiciera diferentes cursos y leyera la documentación oficial de *Angular*, se centró en la parte de la creación de los tags en ofertas. Esta funcionalidad en un primer momento se iba a realizar mediante un *input*, en el cual los usuarios debían llenar manualmente esta información pero nos percatamos que esto era demasiado engorroso, por lo que David investigo que alternativas teniamos al respecto y encontró un paquete llamado *keyword-extractor* que permite extraer palabras clave de un texto en especifico, por lo que la integró en nuestra solución, de esta manera al usuario solo le basta con añadir los campos con información relevante (descripción,título) y la propia aplicación genera estos tags en vista y posteriormente se guardan en nuevas tablas creadas

para este proposito. Esta nueva funcionalidad ocasionó un cambio en la estructura de la base de datos ya que se tuvo que crear nuevas tablas (tags, oferta_tags, demanda_tags) y crear las *constraints* necesarías para mantener una congruencia en la información recibida. Luego de ello, se centró en la funcionalidad de los filtros, aunque parte del filtrado estaba desarrollado no estaba terminado, ya que se pasaba un *json* con la información de algunos de los filtros, pero estos nos funcionaban correctamente por dos motivos sea debido a que el los desplegables desde donde se tomaban los datos no estaban terminados o que existian campos que no existian en la base de datos. David se centró en limpiar y corregir estos problemas, añadiendo el filtro por tags que era una de las nuevas funcionalidades solicitadas.
Para finalizar David se centró en la creación de resumenes por roles aúnque no termino todo el desarrollo están creados los componentes necesarios para su correcto funcionamiento, también se encargó de quitar las dependencias que existían con una base de datos MongoDB para la foto de perfil de los usuarios, esto lo solucionó añadiendo un nuevo campo en la tabla *persona*(nombre del archivo) y modificando la ubicación donde se guarda esta imagen, este cambió ocasionó modificar tanto la parte fronted como backend, ya que el frontal hacía reference a un bucket AWS S3.

9.3. Adrian Dorta Yagüe

En la primera fase del proyecto Adrián ha estado investigando acerca del ApS partiendo de la memoria del TFG del año anterior [4].

En la segunda fase del proyecto Adrián ha realizado los modelos de dominio y de datos para representar el subsistema proyectos-activos. El modelo de Dominio realizado comparte algunas entidades del modelo de Dominio del TFG del año pasado [4] debido a que también eran relevantes en el subsistema proyectos-activos. Para realizar el modelo de datos Adrián ha buscado referencias en aplicaciones como *Microsoft Teams* o *Everdo* para definir las nuevas entidades que aparecen en el modelo de dominio, *Evento* y *Objetivo*. El modelo de

dominio resultante es la figura 5.6. El modelo de Datos resultante es la figura 5.8.

En la tercera fase Adrián se ha encargado de realizar los diagramas BPMN de creación de proyectos. Para ello representó las tres situaciones que se pueden dar a la hora de crearse un proyecto, figura 5.1, y también representó los subprocesos creación de partenariado (figura 5.2) y aprobación de creación de partenariado(figura 5.3). Para realizar el diagrama 5.1 tuvo que informarse acerca del estándar BPMN sobre cómo representar tareas de coworking en el mismo. Tras buscar información se añadieron dos tareas en distintas swimlanes indicando en la puerta paralela que se trata de una acción de coworking [2]

Se van a dar situaciones de tareas de coworking en distintos momentos de vida del proyecto, por ejemplo al principio del proyecto cuando este está en estado EN CREACION los profesores pertenecientes al proyecto, el socio comunitario y/o los beneficiarios tienen que trabajar juntos en buscar que tipo de servicio necesitan y como se puede aprovechar al máximo el proyecto ApS. Otro caso es en el momento en el que se está trabajando en el proyecto ya que, la evaluación del alumno se va realizando a medida que se trabaja en el proyecto, como si de una evaluación continua se tratase. Estos dos casos pueden observar en el diagrama de ciclo de vida de un proyecto ApS. Tras realizar el diagrama de creación de proyectos Adrián también realizó el diagrama del ciclo de vida de un proyecto ApS en un diagrama BPMN. Para realizar el diagrama se han comprobado las fases del proyecto definidas en el TFG del año pasado [4]. El modelo del ciclo de vida resultante es la figura 5.4 en el cual se define un proyecto desde que está en estado EN_CREACION hasta que pasa a estado CERRADO o FINALIZADO. El subproceso es la figura 5.1.2 y representa la valoración de entrada de los usuarios dentro del proyecto por parte del profesor una vez envían la solicitud de paricipación. Durante la realización del diagrama de ciclo de vida de un proyecto ApS se incluyó la entidad *Beneficiario* como una composición de Socio Comunitario y siendo una subclase de usuario ya que un Socio comunitario puede que no sea el destinatario del servicio y solamente represente una entidad social. Se actualizó los diagramas de Datos y de Dominio con esta nueva entidad.

Tras haber realizado los diagramas por falta de tiempo Adrián que comenzó a buscar información acerca de LaTeXpara redactar la memoria, por facilidad de trabajar en común decidí utilizar la herramienta Overleaf para comenzar el proyecto de la memoria para familiarizarme con el entorno y con el lenguaje.

Capítulo 10

Instalación

En este apartado se detallará la instalación de la aplicación tanto en un entorno local para el desarrollo de las nuevas funcionalidades, asi como en un entorno cloud para el despligue en producción.
En la siguiente lista se detallará las versiones de las tecnologías utilizadas en el proyecto:

- Node **v14.18.2**
- Npm **6.14.15**
- Angular CLI **13.0.4**
- Visual Studio Code **1.67.2**

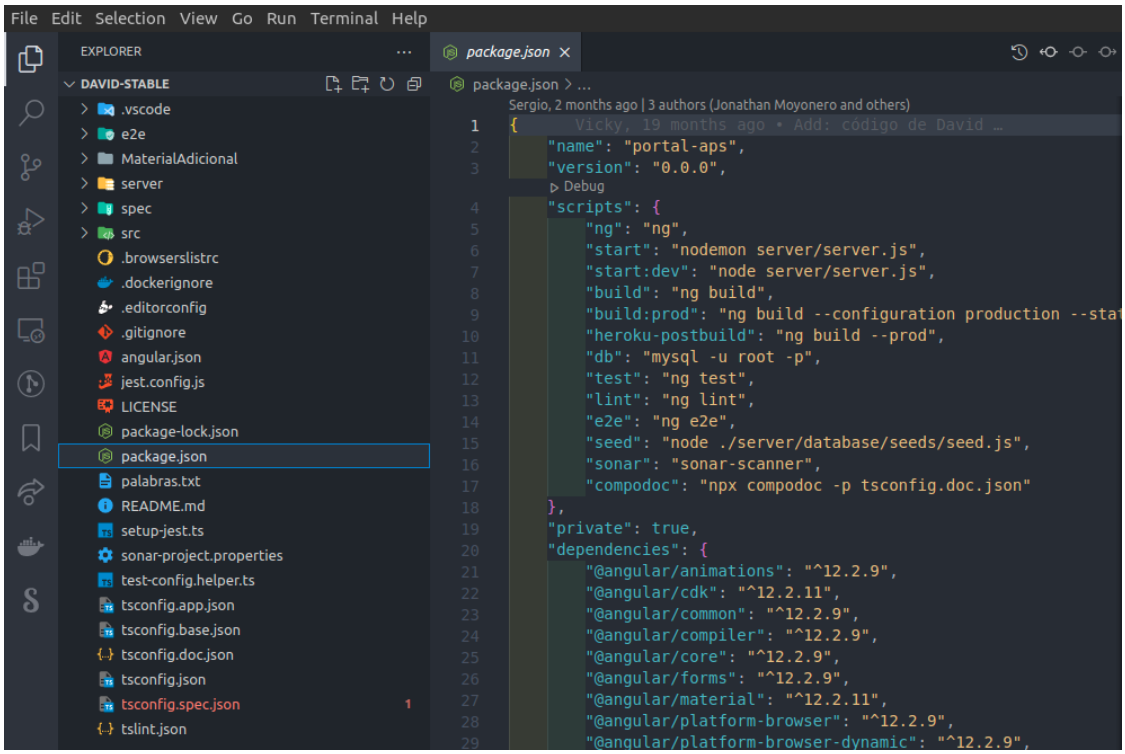
10.1. Local (Desarrollo)

Los pasos a seguir para el despliegue en local son los siguientes:

1. **Instalar cliente git**, la forma de instalación variará dependiendo del sistema operativo instalado en el equipo. En la pagina oficial se encuentran las guías detalladas de cada una de las casos **https://git-scm.com**.
2. **Instalar docker y docker-compose**, la forma de instalación variará dependiendo del sistema operativo instalado en el equipo. En la pagina oficial se encuentran las guías detalladas de cada una de las casos **https://https://docs.docker.com/engine/install**.

3. **Descargar los repositorios en local**, los repositorios son https://github.com/ucm-tfg/tfg-aps-configuration (archivos de configuración) y https://github.com/ucm-tfg/tfg-aps (core de la aplicación), esto lo realizaremos con el comando **git clone**.

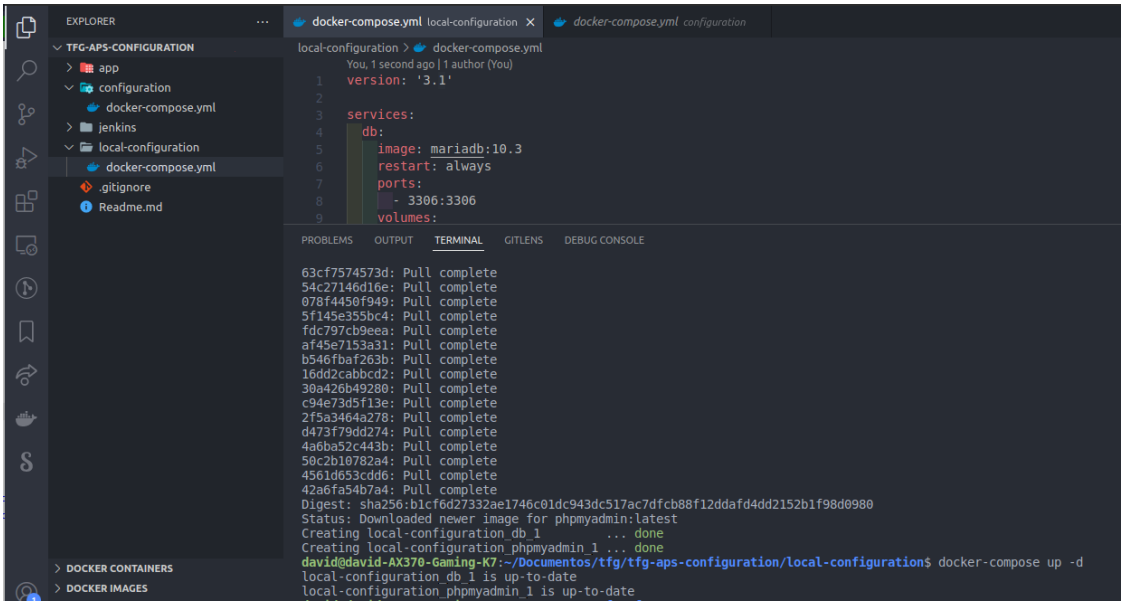
4. **Abrimos las carpetas en un IDE**, en este caso yo he usado Visual Studio Code (VSCode) ya que me siento mas cómodo con el, en cualquier caso se puede usar cualquier otro IDE.



Pantalla principal VSCode

5. **Arrancamos la base de datos**, esto se podría realizar manualmente en local mediante la instalación de xampp, phpmyadmin u otras alternativas. Pero en nuestro caso tenemos creado un archivo docker-compose el cuál ejecutará estos dockers en segundo plano evitando tener aplicaicones externas directamente instaladas en nuestro equipo. Para ello basta con ir al repositorio tfg-aps-configuration que ya tenemos clonado en local y ejecutar el siguiente comando **docker-compose up -d** dentro de la carpeta

local-configuration. Con el comando **docker ps** podemos comprobar que estan corriendo correctamente.



Ejecutando MariaDB mediante docker-compose

6. **Configuramos las credenciales**, en la raiz del proyecto debemos crear un archivo llamado **.env**, con los siguientes parametros (los valores que se observan son los por defecto) :

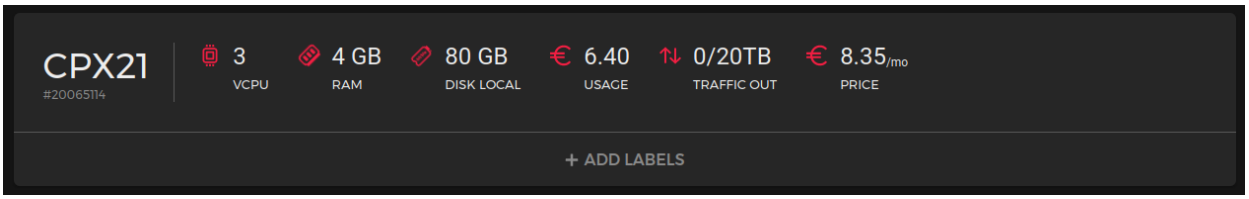
   - JWT_SECRET=1234, //JSON Web Token
   - PORT=8080, //Puerto servidor backend
   - MYSQL_IP=localhost, //Url de la base de datos
   - MYSQL_PORT=3306, //Puerto de la base de datos
   - MYSQL_USER=aps, //Usuario de la base de datos
   - MYSQL_PASSWORD=aps, //Contraseña de la base de datos
   - MYSQL_DATABASE=aps //Nombre de la base de datos

7. **Ejecutamos la parte backend**, utilizando el comando **npm run start**. Esto ocasionará que el **servidor backend se ejecute en el puerto 8080 por defecto**.

8. **Ejecutamos la parte fronted**, utilizando el comando **ng serve**. Con este paso finalizamos la parte del despliegue en local, la parte del **frontal se ejecuta en el puerto 4200**.

10.2. Cloud (Producción)

Para el despliegue a producción, parte de los pasos serán similares a desarrollarlo en local pero nuestro docker-compose, asi como la infraestructura que rodea la aplicación es bastante mas amplia. En este caso usaremos un servidor Debian 11 y con las prestaciones que se observa en la siguiente imagen.



Servidor Hetzner

A continuación se muestra el procedimiento a seguir:

1. **Instalar cliente git**, podemos instalar git en debian 11 con el siguiente comando **apt-get install git**.

2. **Instalar docker y docker-compose**, en la pagina oficial encontramos la guía detallada, adjunto el link **https://docs.docker.com/engine/install/debian**.

3. **Descargamos el repositorio https://github.com/ucm-tfg/tfg-aps-configuration**, esto lo realizaremos con el comando **git clone**.

4. **Ejecutamos el archivo docker-compose**, una vez tengamos descargado el repositorio podemos ir a el y dentro de la carpeta **configuration** ejecutar el comando **docker-compose up -d**. Utilizando el comando **docker ps** podremos ver que todo esta corriendo correctamente.

5. **Modificación de dominio (Opcional)**, este servidor cloud te brinda una ip estática la cual nos permite acceder directamente mediante los puertos a los dockers desplegados, si tenemos un dominio nuestro podemos redirigirlo a este servidor y mediante traefik relacionar cada uno de los subdominios con su respectivo puerto(esto se realiza desde el archivo docker-compose que ejecutamos anteriormente).

6. **Servicios desplegados**

   - **MariaDB**, la base de datos no tiene asociada una url externa debido a que el uso exclusivo es mediante la red interna de los dockers.
   - **PhpMyAdmin**, esta asociada a la url **https://aps-phpmyadmin.moyonero.com**
   - **Jenkins**, esta asociada a la url **https://aps-jenkins.moyonero.com**
   - **Heimdall**, esta asociada a la url **https://aps-dashboard.moyonero.com**
   - **Sonarqube**, esta asociada a la url **https://aps-sonar.moyonero.com**
   - **App**, esta asociada a la url **https://aps.moyonero.com**

   Todas las credenciales para estos servicios se encuentran dentro del archivo docker-compose anteriormente mencionado.

Bibliografía

[1] APIcurio. https://www.apicur.io/.

[2] Coworking Stack Overflow.

[3] Barbara Jacoby. *Service-learning essentials: Questions, answers, and lessons learned* John Wiley Sons. 2014.

[4] Daniela-Nicoleta Boldureanu Victoria Gnatiuk Romaniuk y Jesús Sánchez Granado. Aplicación web de soporte al aprendizaje-servicio: gestión de ofertas y demandas de servicio. *Directores: Simon Pickin UCM, Manuel Montenegro Montes UCM. Titulación: Grado en Ingeniería Informática y Grado en Ingenería de Software. 2021.*

[5] David Jiménez del Rey. Desarrollo de una comunidad web para el soporte virtual del aprendizaje-servicio 3. *Directores: Ángeles Manjarrés Riesco UNED, Simon Pickin UCM. Titulación: Grado en Ingeniería Informática de la UNED. Octubre 2020.*

[6] Emmanuel Parmentier. Diseño e implementación de un servicio de conexión de la oferta y la demanda de proyectos de fin de carrera de cooperación. *Director: Carlos Mataix UPM. Titulación: Ingeniería Industrial de la UPM 2004.*

[7] Hector Alonso Meana. Aplicación de soporte a una comunidad educativa interesada en el aps virtual. *Directores:Ángeles Manjarrés Riesco UNED, Simon Pickin UCM. Titulación: Grado en Ingeniería Informática de la UNED. Junio 2018.*

[8] Natalia Rodríguez. Virtu@l-aps. *Director: Juan García Gutierrez, UNED. Titulación: Grado en Educación Social. 2017.*

[9] Natalia Rodríguez-Fernández Angeles Manjarrés Riesco Simon James Pickin y Héctor Alonso Meana. Virtu@l-aps: Soporte tecnológico para el aprendizaje-servicio virtual. *Revista Iberoamericana de Educación a Distancia, Vol. 23, Núm. 1*, 2020.

[10] Sol Lozano Serrano. Desarrollo de una comunidad virtual de profesores en el Ámbito de la educación para el desarrollo humano y sostenible. *Directores:Ángeles Manjarrés Riesco UNED, Simon Pickin UCM. Titulación: Grado en Ingeniería Informática de la UNED. Sept. 2015.*