



U N I V E R S I D A D
COMPLUTENSE
M A D R I D

Facultad de Informática

Aplicación web de soporte al
Aprendizaje-Servicio: gestión de ofertas y
demandas de servicio

Web application for supporting Service
Learning: management of service offers
and service requests

Autores

Daniela-Nicoleta Boldureanu (Grado en Ingeniería del Software)
Victoria Gnatiuk Romaniuk (Grado en Ingeniería Informática)
Jesús Sánchez Granado (Grado en Ingeniería Informática)

Directores

Simon Pickin
Manuel Montenegro Montes

Curso 2020/2021

Índice general

Resumen	7
Palabras clave	9
Abstract	11
Keywords	13
1. Introducción	15
1.1. Antecedentes	15
1.2. Objetivos	16
1.3. Plan de trabajo	16
2. Introduction	19
2.1. Background	19
2.2. Objectives	20
2.3. Workplan	20
3. Contexto de la propuesta	23
3.1. Introducción	23
3.2. Elementos que intervienen en un proyecto ApS	23
3.3. Motivación	24
3.4. TFG de partida	25
3.5. Estudio tecnológico	26
4. Tecnologías utilizadas	29
4.1. Node.js	29
4.2. Angular	29
4.3. GitKraken	30
4.4. Pivotal tracker	31
4.5. L ^A T _E X	31
4.6. MySQL	31
4.7. Modelio	32
4.8. MySQL Workbench	32
5. Modelos de dominio, de datos y de relación	33
5.1. Introducción	33
5.2. Modelo de dominio	34
5.3. Modelo de datos	38
5.4. Modelo relacional	38

5.4.1. Separación de datos	40
5.4.2. Usuarios	40
5.4.3. Anuncios de servicio	42
5.4.4. Colaboración	44
5.4.5. Comunicación	46
6. DAO	49
7. Matching entre oferta de servicio y demanda de servicio	51
7.1. Definición del matching	51
7.2. Criterios de matching y antimatching	52
7.2.1. Criterios de <i>matching</i>	52
7.2.2. Criterios de <i>antimatching</i>	53
7.3. Obtención del porcentaje final de <i>matching</i>	54
7.4. Ejemplo de <i>matching</i>	55
8. Creación de formularios	57
8.1. Formulario de registro de usuarios	57
8.2. Formulario para editar los datos de un usuario	59
8.3. Formulario creación demanda de servicio	60
8.4. Formulario creación oferta de servicio	61
8.5. Formulario creación de partenariado profesor	63
9. Conclusiones y trabajo futuro	69
9.1. Introducción	69
9.2. Objetivos cumplidos	69
9.3. Problemas Encontrados	70
9.4. Trabajo Futuro	71
10. Conclusions and future work	73
10.1. Introduction	73
10.2. Objectives completed	73
10.3. Problems Found	74
10.4. Future work	75
11. Contribución	77
11.1. Daniela-Nicoleta Boldureanu	77
11.2. Victoria Gnatiuk Romaniuk	79
11.3. Jesús Sánchez Granado	80

Índice de figuras

2.1. PivotalTracker: tareas	21
4.1. GitKraken: Información de una rama de desarrollo	30
4.2. Pivotal tracker: Estadísticas de historias	31
5.1. Modelo de dominio	36
5.2. Modelo de datos	37
5.3. Diagrama de entidad-relación	39
5.4. Diagrama de entidad-relación - Usuarios	41
5.5. Diagrama de entidad-relación - Anuncios de servicio	43
5.6. Diagrama de entidad-relación - Colaboración	45
5.7. Diagrama de entidad-relación - Comunicación	47
8.1. Formulario de registro	59
8.2. Formulario de creación de demanda	65
8.3. Formulario de creación de ofertas	66
8.4. Formulario de creación de partenariado: parte 1	67
8.5. Formulario de creación de partenariado: parte 2	68

Resumen

Un proyecto ApS es una práctica académica que combina procesos de aprendizaje y servicio a la comunidad para ayudar al alumnado a implicarse en proyectos y actividades de su entorno. De esta manera el alumnado podrá adquirir nuevos conocimientos y progresar en su desarrollo de aprendizaje en la universidad.

Este TFG (*Trabajo de Fin de Grado*) es la continuación de un TFG, *Desarrollo de una comunidad web para el soporte virtual del Aprendizaje-Servicio III*, realizado por David Jiménez Del Rey en la UNED (*Universidad Nacional de Educación a Distancia*). Su TFG fue dirigido por Ángeles Manjarrés y codirigido por Simon Pickin.

Los profesores con experiencia en iniciativas de tipo ApS se dieron cuenta de que un buen soporte informático podría ser de gran ayuda en la difícil 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 una propuesta de proyecto realista que cumple las necesidades de las dos partes.

Partiendo del TFG de David Jiménez Del Rey, los objetivos de nuestro *Trabajo de Fin de Grado* fueron crear un modelo de dominio, un modelo de datos y un modelo relacional para la aplicación, cambiar de una base de datos no relacional, MongoDB, a una base de datos relacional, MySQL, implementar DAOs (*Objetos de Acceso a Datos*) para encapsular el acceso a la base de datos, crear DTOs (*Objetos de Transferencia de Datos*) para el transporte de los datos entre las diferentes capas de la aplicación, implementar un sistema de *matching* entre las ofertas de los profesores y las demandas de los socios comunitarios, adaptación del código de los formularios ya creados en el anterior proyecto, tales como registro y editar perfil de usuarios y la implementación de unos nuevos para la creación de la oferta, la demanda y los partenariados y la corrección del TFG precedente.

Nuestro TFG fue desarrollado usando tecnologías como Angular, Express, JavaScript, Node.js y MySQL, donde la mayoría de estas tecnologías ya se habían usado en el anterior TFG

Palabras clave

- ApS
- aprendizaje-servicio
- propuesta educativa
- servicios a la comunidad
- necesidad social
- *matching*
- oferta
- demanda
- socio comunitario

Abstract

An SL project is an academic practice that combines learning processes and community service to help students get involved in projects and activities in their environment. In this way, students will be able to acquire new knowledge and progress in their learning development at university.

This FDP (*Final Degree Project*) is the continuation of a FDP, *Development of a web community for the virtual support of Service-Learning III*, carried out by David Jiménez Del Rey at the UNED (*Universidad Nacional de Educación a Distancia*). His FDP was directed by Ángeles Manjarrés and co-directed by Simon Pickin.

Teachers with experience in SL initiatives realized that good IT support could be of great help in the difficult task of matching the supply and demand of SL. Thanks to this support, the identification of potential partnerships would be facilitated as well as the collaboration between the provider and the potential recipient 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.

Based on David Jimenez Del Rey's FDP, the objectives of our FDP were to create a domain model, a data model and a relational model for the application, change from a non-relational database, MongoDB, to a relational database, MySQL, implement DAOs (*Data Access Objects*) to encapsulate access to the database, create DTOs (*Data Transfer Objects*) for transport of the data between the different layers of the application, implement a system of matching between the offers of the teachers and the demands of the community partners, adaptation of the code of the forms already created in the previous project, such as registration and edit user profiles and the implementation of new ones for the creation of supply, demand and partnerships and the correction of the previous FDP.

Our FDP was developed using technologies such as Angular, Express, JavaScript, Node.js and MySQL, where most of these technologies had already been used in the previous FDP

Keywords

- SL
- service-learning
- academic methodology
- community services
- social need
- matching
- offer
- demand
- community partner

Capítulo 1

Introducción

El objetivo de este TFG es retomar la idea de creación de una comunidad web de proyectos ApS (*Aprendizaje-Servicio*) empezada a nuestro saber, en el año 2004 por Emmanuel Parmentier en su PFC (*Proyecto Fin de Carrera*). Otros trabajos después siguieron esta trayectoria y el último de ellos fue el TFG de David Jiménez del Rey, este es el proyecto que nosotros continuamos en nuestro TFG. Estos proyecto precedentes se detalla más adelante en el capítulo 3.3. El proyecto objetivo en cuestión consiste en la creación de una plataforma web que permita crear un entorno digital en el que las empresas, principalmente del sector público, y las universidades desarrollen actividades de labor social que ayuden a los alumnos a desarrollar de forma práctica lo aprendido en el aula y a moldearles como ciudadanos éticos y solidarios.

1.1. Antecedentes

Este trabajo parte del TFG de David Jiménez del Rey, que se desarrolló en la UNED (*Universidad Nacional de Educación a Distancia*) bajo la tutela de Ángeles Manjarrés Riesco y Simon Pickin. Todos los requisitos del TFG nos han sido dados por nuestros directores que actuaban como clientes del proyecto. Esto implica que el funcionamiento de todos los elementos que intervienen en el TFG han sido definidos por nuestros directos junto a colaboradores entendidos en la materia. El proyecto ha sido desarrollado con tecnologías como Node.js, Angular, Express y MySQL.

La plataforma web desarrollada en este proyecto tiene como objetivo ayudar a crear, gestionar y evaluar proyectos ApS. Un proyecto ApS es una práctica académica en la que el alumno aplica las habilidades teóricas aprendidas en las clases en el mundo real, ayudando a su comunidad con todo tipo de tareas. La actividad es planteada y gestionada por uno o varios profesores y un socio comunitario, que es una empresa que está interesada en desarrollar estos proyectos. Al acabar la actividad, el alumno es evaluado por los profesores y es motivado a reflexionar sobre los servicios prestados, con el objetivo de fortalecer la solidaridad y la ética del alumno.

El principal problema de los ApS es acordar los proyectos entre el socio comunitario y los profesores, ya que cada uno tiene una idea muy diferente del proyecto. Aunque el socio comunitario y el profesor quisieran desarrollar el mismo proyecto, lo plantean de formas muy diferentes y es por esto por lo que es difícil emparejarlos. Esta fue la principal motivación de los profesores que empezaron el planteamiento de este proyecto en el 2004. En el capítulo 3 se explica con más detalle que es un proyecto ApS.

1.2. Objetivos

Partiendo del trabajo anterior, nuestros objetivos principales en este TFG fueron continuar el proyecto remodelando la base de datos, rediseñando la aplicación y creando un sistema de emparejamiento de los proyectos planteados por un profesor y los planteados por un socio comunitario. A continuación, se listan los objetivos de este TFG.

- Construir unas bases sólidas del proyecto, creando un modelo de dominio que aclare los conceptos implicados en la aplicación y un modelo de datos que enriquezca el modelo de dominio y plasme cómo la aplicación gestiona la información.
- Crear un modelo relacional que muestre la estructura de la base de datos, facilitando su entendimiento y manejo a los futuros desarrolladores del proyecto.
- Crear una base de datos relacional compleja y rica en detalles.
- Implementar cuatro DAOs que realicen la lógica de acceso y gestión de datos, encapsulando el acceso a la base de datos. Crear *transfers* que permiten estructurar y manejar de forma sencilla los datos de la BD.
- Implementar un sistema de *matching* de los proyectos planteados por un profesor y los planteados por un socio comunitario que determina qué porcentaje de encaje tienen.
- Adaptar las páginas de registro y de perfil del usuario al nuevo sistema, e implementar formularios para la creación de ofertas, demandas y partenariados.
- Corregir *bugs* encontrados en el proyecto precedente.

1.3. Plan de trabajo

Establecidos los objetivos anteriores por nuestros directores, lo primero que hicimos fue encontrar una herramienta de gestión de proyectos que nos permitiera organizar el trabajo. Esta herramienta es Pivotal Tracker, ver Figura 2.1. Esta herramienta centrada en la gestión de proyectos de tipo SCRUM nos ha ayudado a crear las tareas, asignarles dificultad, clasificarlas y llevar un control general del trabajo realizado y por realizar. El trabajo realizado en este TFG se puede dividir principalmente en seis grandes fases.

- En la primera fase hemos leído la memoria del TFG de David Jiménez para comprender las raíces del problema a resolver y conocer los detalles del proyecto en el que íbamos a trabajar. En paralelo hemos estado investigando por nuestra cuenta sobre los proyectos ApS y las tecnologías en las que estaba implementado el proyecto, sobre todo Node.js que no conocíamos antes de empezar el TFG. Esta fase ha comprendido desde el día 30 de septiembre hasta el día 6 de noviembre.
- En la segunda fase, hemos explorado el código del anterior TFG para familiarizarnos con él y después hemos realizado pruebas manuales de la solución. Al realizar estas pruebas hemos descubierto algunos *bugs* que posteriormente hemos corregido. Esta fase ha comprendido desde el día 7 de noviembre hasta el día 19 de noviembre.
- La tercera fase ha consistido en el desarrollo de un modelo de dominio y un modelo de datos que ilustran la solución del problema de una forma más detallada y concisa. Por otro lado, hemos estado diseñando la nueva base de datos relacional teniendo en

cuenta la nueva estructura de la aplicación. Esta fase se describe con más detalle en el capítulo 5. Esta fase ha comprendido entre el día 20 de noviembre y el día 26 de febrero.

- La cuarta fase ha comprendido entre el día 27 de febrero y el día 10 de marzo. En esta fase hemos implementado los cuatro DAOs, los *transfers* y hemos adoptado los controladores al nuevo sistema. Esta fase se describe con más detalle en el capítulo 6.
- La quinta fase ha consistido en la creación del sistema de *matching*. Para conocer más detalles sobre esta fase lea el capítulo 7. Esta fase ha comprendido entre el día 11 de marzo y el día 23 de abril.
- La sexta fase ha consistido en el aprendizaje de Angular, tecnología que desconocíamos antes de empezar el TFG y en la creación y adaptación de parte de la interfaz. En concreto la parte de la interfaz afectada se menciona en la sección de objetivos y se explica con más detalle en el capítulo 8. Esta fase ha comprendido entre el día 24 de abril y el día 20 de mayo.

Capítulo 2

Introduction

The objective of this FDP (*Final Degree Project*) is to take up the idea of creating a web community of SL (*Service-Learning*) projects started to our knowledge, in 2004 by Emmanuel Parmentier in his FDP. Other works later followed this path and the last of them was David Jiménez del Rey's FDP, this is the project that we continued in our FDP. These preceding projects are detailed later in the chapter 3.3. The objective of the project consists in the creation of a web platform that allows creating a digital environment in which companies, mainly from the public sector, and universities develop social work activities that help students to develop in a practical way what they have learned in the classroom and to mold them as ethical and caring citizens.

2.1. Background

This work is based on David Jiménez del Rey's FDP, which was developed at the UNED (*Universidad Nacional de Educación a Distancia*) under the tutelage of Ángeles Manjarrés Riesco and Simon Pickin. All the requirements of the FDP have been given to us by our directors who acted as clients of the project. This implies that the functionality of all the elements that intervene in the FDP have been defined by our directors together with collaborators who are knowledgeable in the matter. The project has been developed with technologies such as Node.js, Angular, Express and MySQL.

The web platform developed in this project aims to help create, manage and evaluate SL projects. An SL project is an academic methodology in which the student applies the theoretical skills learned in the classes in the real world, helping their community with all kinds of tasks. The activity is planned and managed by one or more teachers and a community partner, which is a company that is interested in developing these projects. At the end of the activity, the student is evaluated by the teachers and is motivated to reflect on the services provided, with the aim of strengthening the solidarity and ethics of the student.

The main problem of SLs is to agree the projects between the community partner and the teachers since each has a very different idea of the project. Although the community partner and the teacher would like to develop the same project, they pose it in very different ways and that is why it is difficult to pair them. This was the main motivation of the teachers who started planning this project in 2004. Chapter

3 explains in more detail what a SL project is.

2.2. Objectives

Based on the previous work, our main objectives in this FDP were to continue the project by remodeling the database, redesigning the application and creating a matching system of the projects proposed by a teacher and those proposed by a community partner. The objectives of this FDP are listed below.

- Build a solid foundation for the project, creating a domain model that clarifies the concepts involved in the application and a data model that enriches the domain model and reflects how the application manages information.
- Create a relational model that shows the structure of the database, facilitating its understanding and management by future project developers.
- Create a complex and detailed relational database.
- Implement four DAOs that perform the logic of access and data management, encapsulating the access to the database. Create transfers that allow the database data to be structured and managed in a simple way.
- Implement a system of matching of the projects proposed by a teacher and those proposed by a community partner that determines what percentage of fit they have.
- Adapt the registration and user profile pages to the new system, and implement forms for the creation of offers, demands and partnerships.
- Correct bugs found in the previous project.

2.3. Workplan

Having established the above objectives by our directors, the first thing we did was find a project management tool that would allow us to organize the work. This tool is Pivotal Tracker, see Figure 2.1. This tool focused on managing SCRUM-type projects has helped us to create tasks, assign them difficulty, classify them and keep a general control of the work done and to be done. The work carried out in this FDP can be divided mainly into six main phases.

- In the first phase we have read David Jiménez's FDP report to understand the roots of the problem to be solved and to know the details of the project we were going to work on. In parallel, we have been investigating on our own about the SL projects and the technologies in which the project was implemented, especially Node.js that we did not know before starting the FDP. This phase has ran from September 30th to November 6th.
- In the second phase, we have explored the code of the previous FDP to become familiar with it and then we have carried out manual tests of the solution. During these tests we have discovered some bugs that we have subsequently corrected. This phase has ran from November 7th to November 19th.

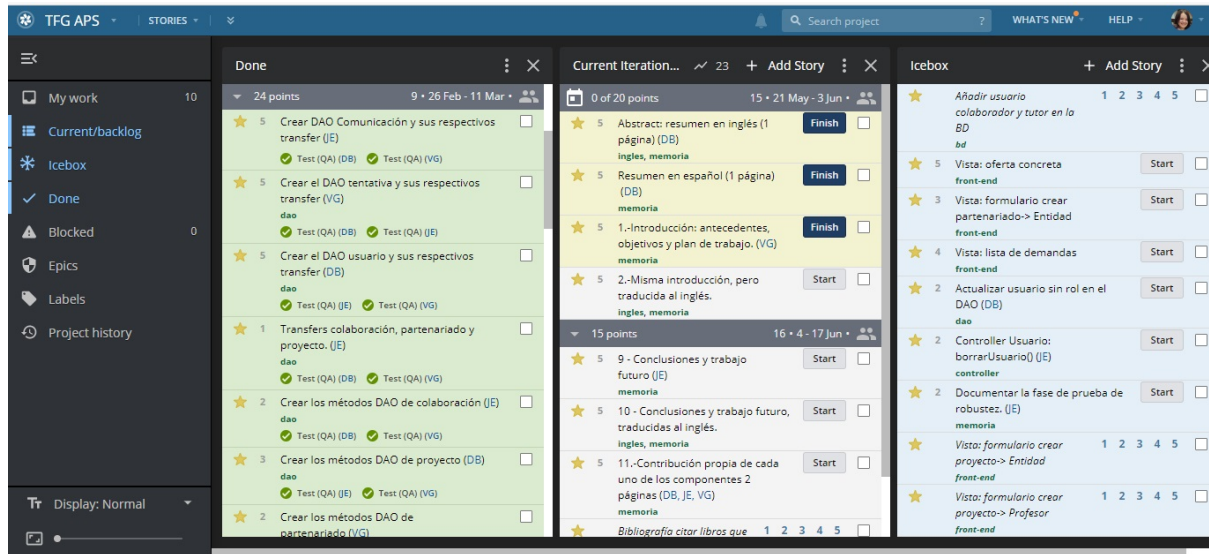


Figura 2.1: PivotalTracker: tareas

- The third phase consisted of developing a domain model and a data model that illustrate the solution of the problem in a more detailed and concise way. On the other hand, we have been designing the new relational database considering the new structure of the application. This phase is described in more detail in chapter 5. This phase has been between November 20th and February 26th.
- The fourth phase ran from February 27th to March 10th. In this phase we have implemented the four DAOs, the transfers and we have adopted the controllers to the new system. This phase is described in more detail in chapter 6.
- The fifth phase consisted of creating the matching system. For more details on this phase, read the chapter 7. This phase has been between March 11th and April 23th.
- The sixth phase consisted of learning Angular, a technology that we were unaware of before starting the FDP, and creating and adapting part of the interface. Specifically, the part of the interface affected is mentioned in the objectives section and explained in more detail in the chapter 8. This phase has been between April 24th and May 20th.

Capítulo 3

Contexto de la propuesta

3.1. Introducción

El ApS es una propuesta educativa que combina aprendizaje y servicios a la comunidad. Los proyectos ApS permiten a los alumnos aprender de una forma más práctica, aplicando sus conocimientos adquiridos en clase mediante la realización de tareas útiles para la comunidad.

Además de dar a los estudiantes la oportunidad de aplicar sus conocimientos en un entorno real, el ApS les impulsa a comprender el funcionamiento de la sociedad y las responsabilidades sociales que estos tienen por formar parte de una sociedad.

Todo proyecto ApS empieza por una iniciativa relacionada con una necesidad social real que implica la ejecución de un servicio para solventarla y tiene como objetivo el aprendizaje y la reflexión del alumno. Para saber más sobre el tema dirijase a la referencia [8].

3.2. Elementos que intervienen en un proyecto ApS

En un proyecto ApS intervienen los siguientes elementos:

- El *alumno* es el individuo que aplica sus conocimientos teóricos en un entorno físico beneficiando a su comunidad. Además de adquirir habilidades prácticas relacionadas con su formación, es importante que se incite al alumno a reflexionar sobre sus actos y el impacto positivo que tienen estos sobre los demás. Esto permite al alumno adquirir compromiso social y desarrollar pensamiento ético, cultivando un ciudadano responsable capaz de mejorar la sociedad de la que forma parte.
- El *socio comunitario* es una empresa pública o de tercer sector que colabora con la institución educativa para resolver un determinado problema social. El socio comunitario suele tener en mente un problema muy concreto, pero no lo suficientemente detallado para la creación de un proyecto educativo. Es por eso que es necesario el partenariado. Es importante que la universidad haga entender al socio comunitario que el ApS no es voluntariado. Por tanto, bajo ningún concepto se puede usar al alumno para la generación de beneficios propios de la empresa o la competencia desleal. El principal objetivo del ApS es formar al alumno introduciéndolo en un entorno real para que este establezca una relación entre lo aprendido en el aula con lo realizado en el proyecto ApS.

- El *profesor* es el individuo que se encarga de guiar al alumno en todo el proceso del proyecto, evaluando sus tareas e incitando al alumno a la reflexión. Además de guiar al alumno en su formación y gestionar el proyecto, ofrece su formación y conocimientos al socio comunitario con la que se colaborará en el proyecto. El profesor se encarga de acordar y organizar los proyectos con el socio comunitario, estableciendo todos los requisitos necesarios para la correcta formación del alumno y el cumplimiento del socio comunitario con los principios del ApS.
- El *partenariado* es una colaboración entre un profesor, o un equipo de profesores, y el socio comunitario. Partiendo de un problema social real y los conocimientos dispuestos por el profesor, el profesor y el socio comunitario determinan las características y particularidades del problema. Una vez definidos los términos y condiciones del futuro proyecto, el profesor abre el proyecto a otros profesores que quieran colaborar en las propuestas y posteriormente el proyecto se abre a los alumnos. Todo partenariado tiene un profesor responsable que pertenece a la universidad en la que se ha desplegado la aplicación.
- El *proyecto* consiste en la ejecución de ciertas tareas realizadas por el alumno que están relacionadas con su formación. Estas tareas permiten al alumno establecer una relación entre lo aprendido en clase y el mundo real. Gracias a estas tareas o servicios, el alumno beneficia a su comunidad, otorgándole una satisfacción personal. El alumno es evaluado de forma continua por el equipo docente. Todo proyecto tiene un profesor responsable que pertenece a la universidad en la que se ha desplegado la aplicación.

3.3. Motivación

La motivación descrita a continuación ha sido extraída de nuestros directores los cuales conocen el histórico del proyecto:

La experiencia de los profesores que han llevado a cabo, o intentado llevar a cabo, iniciativas de tipo ApS 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. El establecimiento de relaciones de partenariado en el ApS requiere tiempo y energía, siendo el análisis de compatibilidad entre la oferta y la demanda de servicios un aspecto clave. Los profesores con experiencia en iniciativas de tipo ApS, tanto de carácter local como en el contexto de proyectos de cooperación internacional para el desarrollo, se dieron cuenta hace tiempo de que un buen soporte informático podría ser de gran ayuda en la difícil 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 una propuesta de proyecto realista que cumple las necesidades de las dos partes.

Los programas de ApS, ya consolidados en el continente americano, tan solo recientemente empiezan a cobrar fuerza en las instituciones educativas europeas, y en particular en las españolas. Los primeros proyectos de ApS se iniciaron en nuestro país apenas hace 15 años. El escaso valor académico concedido hasta ahora a la práctica del ApS fuera del ámbito de los estudios pedagógicos, ha sido motivo de que los intentos de desarrollar la

aplicación informática descrita hayan surgido siempre en el contexto de PFC/TFG. El desarrollo de una tal aplicación requiere una alta dedicación y no tiene valor de investigación tecnológica, de modo que los profesores técnicamente cualificados no tienen disponibilidad para abordarlo.

El PFC de Ingeniería Industrial de la UPM (*Universidad Politécnica de Madrid*) [7] fue el primer intento, según nuestro conocimiento, de desarrollar una aplicación para el soporte de la definición de PFC de cooperación (y del previo establecimiento de un partenariado Universidad-ONG) y dio como resultado una aplicación basada en el CMS (*sistema de gestión de contenidos*) Plone. Sin embargo, su naturaleza de prototipo, junto con la falta de recursos humanos para su despliegue y administración, hizo que nunca llegara a utilizarse. Ángeles Manjarrés, profesora de la UNED, y Simon Pickin, uno de los directores del presente trabajo, retomaron la idea de [7] en el TFG del grado en Informática de la UNED [10], en el que se desarrolló una aplicación web basada en el popular CMF (*framework de gestión de contenidos*) Drupal. El prototipo desarrollado fue bastante básico e incompleto y tenía restricciones técnicas que limitaban su funcionalidad, pero constituyó un pequeño avance hacia el objetivo. Aunque el propósito del PFC [7] y del TFG [10] fue el soporte de la definición de PFC/TFG de cooperación, este objetivo puede verse como un caso particular del soporte de proyectos ApS y la plataforma informática necesaria es prácticamente idéntica en los dos casos.

Se retomó la iniciativa de [10] en el contexto de una colaboración entre la UCM y el grupo de investigación en innovación docente COETIC (“Grupo de Innovación Docente de la UNED para el Desarrollo de la Competencia Ética y Cívica y las metodologías basadas en la Comunidad en la educación superior”) con el TFG del grado en Educación Social de la UNED [9], centrado en la especificación de una aplicación para el soporte del ApS virtual, y los TFG del grado en Informática de la UNED [6] y [2], centrados en el desarrollo de tal aplicación, los dos últimos codirigidos por Ángeles Manjarrés y Simon Pickin. En [6], después de un intento fallido de continuar con el desarrollo en Drupal empezado en [9], debido a que Drupal resultó ser mucho más cerrado de lo esperado, se desarrolló una nueva aplicación desde cero basada en el stack MyEAN (*MySQL-Express-Angular-Node*). Esta aplicación se describe brevemente en [1]. En [2], se continuó el desarrollo de la aplicación de [6] pero esta vez con el *stack* MEAN (*MongoDB-Express-Angular-Node*); la substitución de MySQL por MongoDB no se debía a razones técnicas sino a razones de conveniencia y familiaridad del desarrollador. Aunque los títulos de [6] y [2] hacen referencia al ApS Virtual, la aplicación desarrollada está pensada para dar servicio a toda la comunidad de ApS.

3.4. TFG de partida

El TFG de David Jiménez presentaba una base a partir de la cual hemos partido para desarrollar nuestra parte de este proyecto. Partiendo del TFG anterior que estaba desarrollado en Angular y Node.js, David Jiménez siguió desarrollando la aplicación hasta conseguir un prototipo de la futura aplicación.

David Jiménez implementó las páginas de registro, *login*, perfil, iniciativa y partenariado. También incluyó diferentes perfiles como el *admin*, el socio comunitario, el profesor y el alumno.

Por razones de comodidad y familiarización con la tecnología, en el TFG anterior se construyó la base de datos en MongoDB. Debido a que los datos de la aplicación son claramente estructurados y muy relacionados entre sí, no había razones suficientes para seguir desarrollando la aplicación sobre MongoDB, así que junto con nuestros directores decidimos cambiar la base de datos a SQL.

Debido a las dificultades provocadas por la pandemia del COVID-19, las bases de la aplicación no quedaron del todo definidas en el TFG anterior y por esta razón hemos tenido que redefinirlas creando un modelo de dominio y de datos que representa todos los elementos del problema y cómo se relacionan entre sí. Diseñamos una base de datos nueva más compleja y rica en detalles que servirá como cimientos para nuestros sucesores, ya que esperamos que llegue el día en que esta plataforma sirva a usuarios reales, los cuales ayudaran a que la educación sea más eficaz y enriquecedora.

3.5. Estudio tecnológico

A continuación, se explicará qué tecnologías tienen el potencial para desarrollar este proyecto y cuáles hemos elegido finalmente. Las razones de las decisiones tomadas sobre el uso de estas tecnologías se detallan en la sección 4.

1. Servidor web:

- Express es un *framework* basado en Node.js que permite gestionar el servidor de una forma sencilla. Este *framework* fue utilizado por David Jiménez en el TFG precursor y se ha mantenido.

2. *Back-End*:

- NodeJS es entorno basado en JavaScript muy popular. Este entorno fue usado en el anterior TFG y se ha mantenido.

3. *Front-End*

- Angular es un *framework* utilizado en el *Front-End* del TFG anterior y se ha decidido mantener.

4. Base de datos:

- MongoDB es un sistema de base de datos no estructurado. Este sistema es el que se estaba usando en el proyecto.
- MySQL es un sistema de base de datos relacional. Decidimos utilizar este sistema en nuestro TFG.

5. Software de control de versiones:

- Git es el controlador de versiones más conocido y muy eficaz, así que desde el principio supimos que es el software que íbamos a usar.

6. Repositorio:

- GitHub es un repositorio gratuito que permite almacenar todos los archivos relacionados con un proyecto y mantenerlos de forma colaborativa con otros usuarios. Se ha decidido utilizar GitHub por su integración con Git.

- Google Drive es un contenedor gratuito que permite almacenar cualquier fichero y compartirlo con los demás. En un principio se estudió utilizar para guardar los *Backups* pero se acabó descartando. Al final se ha utilizado para almacenar todo tipo de documentos relacionados con el TFG, excepto el código fuente de la aplicación.

7. Herramientas de organización:

- GitHub Projects es una herramienta que ofrece GitHub que permite crear una organización de proyecto tipo Kanban.
- Trello es una herramienta sencilla estilo Kanban para organizar los proyectos, pero tiene muchas limitaciones en su versión gratuita.
- PivotalTracker es una herramienta de gestión de proyectos basada en Scrum que permite crear *stories*, asignarles un peso en función de lo compleja que sea la *story* y ofrece analíticas que permiten analizar el progreso del proyecto. Hemos decidido utilizar esta herramienta para organizarnos porque es una herramienta completa.

8. Herramientas UML:

- Diagrams.net es una herramienta de diseño de diagramas *online* la cual permite diseñar varios tipos de diagramas entre los cuales se encuentran diagramas de clases, de flujos, de entidades, etc.
- Modelio es una aplicación de escritorio que permite crear modelos UML complejos, indicando los atributos, los métodos y las relaciones que tienen los elementos entre sí. Debido a que es una herramienta completa y que es una herramienta que ya conocíamos, la hemos elegido para la creación de nuestros modelos.

9. Herramientas para el diseño del modelo relacional de la base de datos:

- phpMyAdmin es una herramienta web que permite gestionar una base de datos SQL. Esta herramienta muestra un modelo relacional de la base de datos muy simple.
- MySQL Workbench es una herramienta de gestión de diseño de base de datos visual que permite crear modelos relacionales complejos. Esta es la aplicación que se ha decidido utilizar.

10. Herramientas para la redacción de la memoria:

- Microsoft Word siendo una herramienta popular y muy conocida para la creación de documentos escritos, fue nuestra primera opción para la redacción de la memoria.
- LaTeX es una herramienta que permite crear documentos profesionales con resultados profesionales. Esta es la herramienta que se ha decidido utilizar.

11. Lenguaje para insertar datos en la BD:

- Python se ha utilizado para insertar valores enumerados en la base de datos.

Capítulo 4

Tecnologías utilizadas

A continuación, se hablará sobre las tecnologías utilizadas explicando brevemente que son y los motivos por los que han sido seleccionadas.

4.1. Node.js

Node.js [3] es un entorno de ejecución asíncrono dirigido por eventos. Funciona a base de promesas, es decir, funciones que devolverán un resultado en algún momento del futuro. Las promesas se pueden encadenar una tras otra, recibiendo cada una el resultado de la anterior.

Esta forma de conseguir concurrencia es distinta a la manera más común que es utilizando cierres de exclusión mutua o candados. Los candados funcionan de la siguiente manera: si durante la ejecución de un programa concurrente un elemento es compartido por varios hilos, el resultado dependerá de las operaciones y el orden en que se lleven a cabo las mismas en dicho elemento. Para evitar que dos hilos accedan de manera simultánea a un mismo recurso, hay que “bloquear” ese recurso utilizando candados, llegando a la situación conocida como exclusión mutua, pero cuando hay varios procesos concurrentes puede darse la situación de que el proceso A esté esperando a que el proceso B libere un recurso, y al mismo tiempo B espera que A libere un recurso. Esta situación se conoce como *deadlock* o interbloqueo y es un bloqueo infinito.

Node.js no utiliza cierres de exclusión mutua o candados, por lo que es imposible que el programa alcance el estado de *deadlock*, lo que lo hace bastante adecuado para desarrollar sistemas escalables.

Otro motivo para continuar con esta tecnología es que Daniela ya tenía conocimiento previo de este entorno y es una tecnología que venía impuesta por el proyecto.

4.2. Angular

Angular [4] es un *framework* para la construcción de SPA (*Singl Page Application*) o aplicaciones de página única que utiliza HTML y Typescript. Angular sigue el patrón modelo-vista-controlador, el cual consiste en separar la aplicación en tres partes:

- Modelo: Es la piedra angular del patrón, se encarga de manejar los datos y la lógica de la aplicación.
- Vista: Es la parte que se le muestra al usuario.

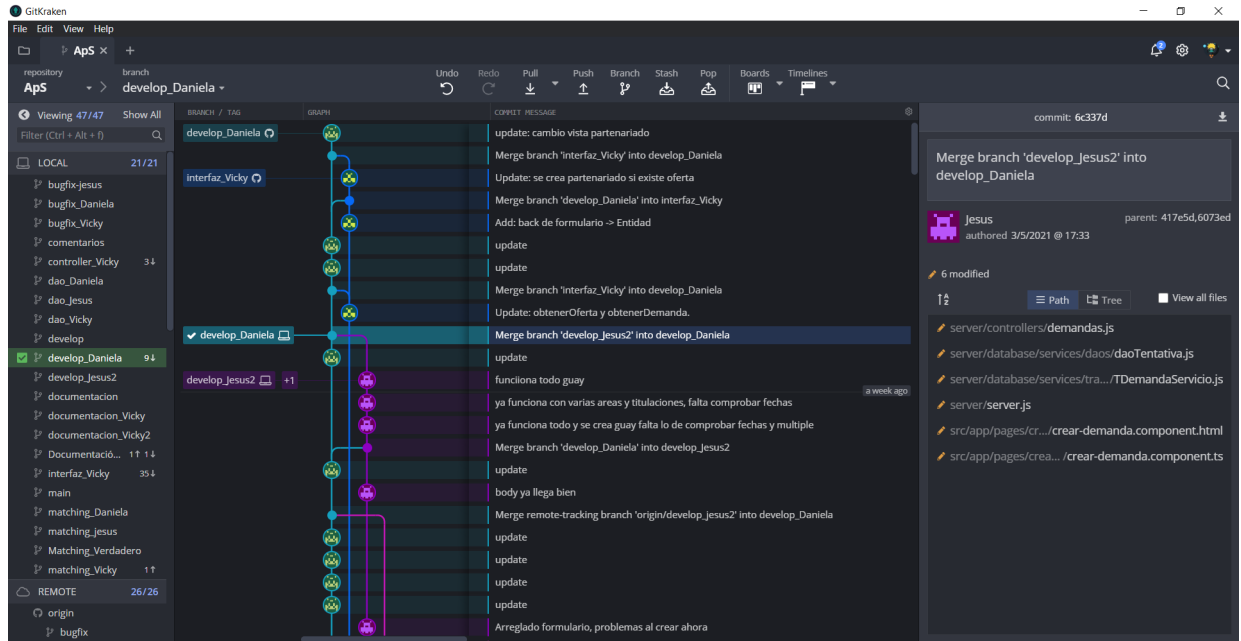


Figura 4.1: GitKraken: Información de una rama de desarrollo

- **Controlador:** Es la parte que se encarga de comunicar a la vista y al modelo. El controlador recibe los eventos de interacción del usuario a través de la vista y se los pasa al modelo, el cual hace las operaciones necesarias y devuelve los resultados al controlador, quien se los pasa a la vista para mostrárselos al usuario.

El uso de Angular venía impuesto por el trabajo realizado con anterioridad y, aunque es una tecnología con la que ningún miembro del equipo estaba familiarizado, es cierto que el diseño de aplicación de página única hace mucho más liviana la ejecución de la aplicación por parte del usuario, al no tener unos tiempos de espera tan grandes como los que tendría al cargar de nuevo cada página. Esto hace de Angular una buena elección para un trabajo de esta índole.

4.3. GitKraken

Gitkraken es un cliente de Git con interfaz de usuario la cual se puede conectar a distintas plataformas de Git, haciendo de intermediario entre el usuario y el repositorio de Git, el cual en este caso está alojado en Github. Hemos escogido esta interfaz para nuestro control de versiones porque permite trabajar desde Windows sin necesidad de conocer los comandos de Git. A diferencia de otros clientes de Git, tiene una representación gráfica muy intuitiva que permite ver la distribución de las ramas, los *commits* y su evolución, como se puede observar en la Figura 4.1, y además permite resolver los conflictos generados al mezclar las distintas ramas de desarrollo dentro de la propia aplicación de una manera bastante sencilla. Esto, sumado a la experiencia previa de Victoria con la aplicación, ha hecho que sea seleccionada como herramienta de control de versiones.

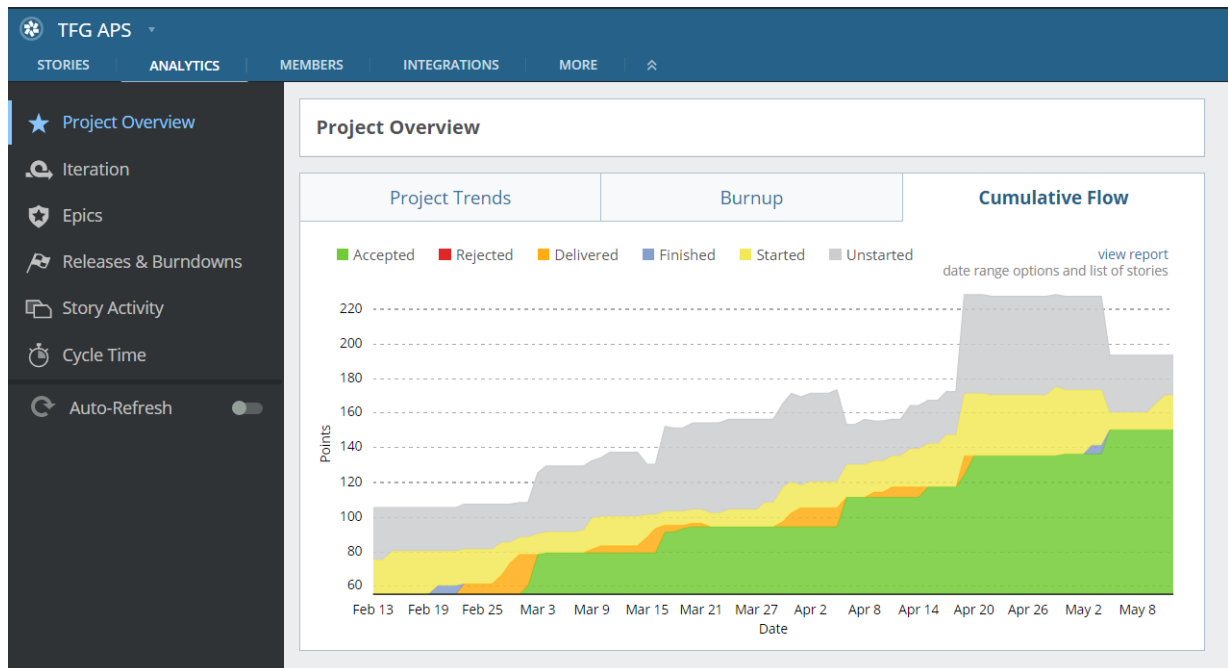


Figura 4.2: Pivotal tracker: Estadísticas de historias

4.4. Pivotal tracker

Pivotal tracker es una herramienta de *product planning* y administración de tareas diseñada para equipos de desarrollo que siguen metodologías de diseño ágiles. Esta herramienta permite crear historias de usuario y asignarles una puntuación del 1 al 5 indicando su dificultad y/o tiempo invertido en dichas tareas. Además, permite cambiar el estado de las tareas (empezado, finalizado, en revisión, etc.) y cualquier cambio en el estado de dichas tareas se informa por correo de manera automática a quien esté involucrado en ella. También permite ver las tareas completadas y rechazadas y generar gráficos indicando el esfuerzo realizado, como el que podemos ver en la figura 4.2. Esta tecnología fue sugerida por Victoria y nos ha facilitado mucho tanto la organización como el seguimiento de nuestros avances.

4.5. L^AT_EX

L^AT_EX es un lenguaje de creación de documentos escritos utilizado comúnmente en el mundo académico, que es una de las principales razones por la que lo hemos escogido para redactar nuestra memoria, a pesar de que ningún integrante del grupo tuviera experiencia previa con ello. A diferencia de otros procesadores de texto, como Microsoft Word o LibreOffice Writer, se escribe el texto plano y se formatea dicho texto con etiquetas.

4.6. MySQL

Aunque nuestro proyecto continúa el trabajo realizado por David Jiménez del Rey, el cual ya contaba con un sistema gestor de bases de datos, dicho sistema era MongoDB. Como los datos que se iban a manejar en la aplicación eran en su mayoría relacionales se tomó la decisión de utilizar MySQL para la base de datos. Dado que todos los componentes

del grupo tenían experiencia previa en bases de datos SQL fue un cambio bien recibido.

4.7. Modelio

Modelio es un entorno de modelado *open-source* el cual permite trabajar con un amplio rango de modelos y diagramas. Dado que ya se contaba con experiencia previa en esta herramienta por parte de todos los miembros del equipo, se ha escogido para realizar los modelos de datos necesarios para la aplicación.

4.8. MySQL Workbench

MySQL Workbench es una herramienta para diseño, desarrollo y administración de bases de datos relacionales. Cuenta con funcionalidades de validación de esquemas y modelos y promueve las mejores prácticas de los estándares de modelado de datos. También promueve los estándares de diseño específicos de MySQL para evitar errores al generar esquemas relacionales o creando bases de datos MySQL. Por estos motivos, junto con su relativa simplicidad, es por lo que se ha elegido esta herramienta para hacer los diagramas de entidad-relación.

Capítulo 5

Modelos de dominio, de datos y de relación

5.1. Introducción

Al empezar con el proyecto, junto con nuestros directores, nos hemos dado cuenta de que la aplicación necesitaba ser definida de una forma más detallada. En TFGs anteriores se entendía que los profesores y las entidades definían los proyectos de la misma forma, pero esto no es así. Las entidades tienen un problema en mente que quieren resolver, pero no saben cómo adaptarlo a un proyecto estudiantil. Los profesores, por su parte, tienen una idea abstracta del problema que quieren resolver. Normalmente tienen ciertos conocimientos académicos los cuales quieren aplicar para mejorar el mundo, pero la necesidad social en cuestión no suele estar muy clara.

En el TFG precedente se creó un elemento llamado *Iniciativa* que almacenaba algunas de las características generales que comparten las propuestas del profesor y el socio comunitario. La *Iniciativa* derivaba en un formulario que se les ofrecía a las entidades y a los profesores, pero las entidades y los profesores no plantean los problemas de la misma manera. Las entidades tienen una idea de proyecto clara pero no saben cómo orientarlo al mundo académico y los profesores tienen una idea de proyecto abstracta y lo plantean con el objetivo de encajar en un plan académico, es por esto por lo que no es apropiado utilizar el mismo formulario para ambos.

Para definir con precisión el funcionamiento correcto y completo de la aplicación, se ha creado un modelo de dominio y un modelo de datos. Estos modelos nos permitirán entender acertadamente el funcionamiento de la aplicación, pero lo que es más importante, permitirán transmitir dicho funcionamiento e idea general a otras personas que trabajarán en este proyecto después de nosotros.

Gracias al modelo de dominio podemos entender qué elementos intervienen y cómo interactúan entre sí, además de las restricciones que se presentan en sus interacciones.

Con el modelo de datos podemos conocer información detallada de cada elemento que interviene en la resolución del problema, así como los atributos que posee.

Como se ha rediseñado la base de datos se ha decidido crear un modelo relacional que muestra todas las tablas de la nueva base de datos y sus relaciones. De esta manera,

podemos representar sus especificaciones técnicas para comprender su estructura y funcionamiento.

5.2. Modelo de dominio

El modelo de dominio es un mapa conceptual de la aplicación que permite a cualquier individuo entender su funcionamiento. El modelo de dominio de esta aplicación se puede observar en la Figura 5.1.

Si empezamos mirando nuestro modelo desde arriba podemos observar que hay una clase madre, llamada *Usuario* y de ella se ramifican otras cinco clases que representan los cinco grupos de usuarios que tiene la aplicación.

- Los estudiantes se dividen en internos y externos. Los internos representan a aquellos estudiantes que pertenecen a la universidad donde está instalada la aplicación y los externos pertenecen a otras universidades.
- El grupo de usuarios promotor se divide en externos e internos por el mismo motivo que los estudiantes.
 - El promotor externo representa al profesor externo, que es aquel profesor que no forma parte de la universidad donde está la aplicación, pero puede participar en un proyecto o partenariado evaluando y guiando a un estudiante externo. El colaborador, por otra parte, es un experto en algún tema en concreto que puede participar en un proyecto o partenariado ofreciendo sus conocimientos o habilidades.
 - El promotor interno puede ser un profesor de la universidad en la que se aloja la aplicación y también puede ser un tutor si pertenece a la UNED, donde existe este rol. El profesor interno puede crear las ofertas de servicio y es el responsable del partenariado y del proyecto.
- El socio comunitario es aquella entidad que hace de socio para la creación del proyecto.
- El admin es el usuario administrador de la aplicación.
- La Oficina APS es un usuario responsable de todos los trámites relacionados con la gestión de los proyectos ApS.

Todo proyecto tiene que definirse en el contexto de un partenariado y este partenariado se puede originar de varias maneras.

Una de ellas es a partir de una oferta de servicio y una demanda de servicio. En este caso el partenariado puede surgir porque una de las dos partes decida aliarse con la otra buscándolo manualmente en la plataforma o porque el sistema de *matching* sugiera la unión de ambas partes. Otra forma es mediante la creación de lo que llamamos partenariados “monoparentales”, es decir, partenariados creados cuando un socio comunitario acepta una oferta creada por un profesor sin haber creado previamente una demanda, o cuando un profesor respalda una demanda creada por un socio comunitario sin haber creado previamente una oferta. En estos casos, la correspondiente oferta (resp. demanda) se creará en el momento de aceptar (resp. respaldar) la oferta (resp. la demanda) y se

utilizará un atributo de la oferta y de la demanda para almacenar la naturaleza “mono-parental” del partenariado.

En general una demanda de servicio es creada por un socio comunitario, pero puede llegar a crearse gracias a un estudiante, el cual previamente ha tenido que crear una iniciativa. Una iniciativa es una propuesta de proyecto de un estudiante, la cual es acogida por un socio comunitario después de haber sido aprobada por la oficina ApS. Cuando el socio comunitario acoge dicha iniciativa, esta se convierte en una demanda de servicio.

Debido a que los proyectos definidos por los profesores y los definidos por las entidades comparten ciertas características, podemos ver en el diagrama que ambos cuelgan de un elemento padre llamado **Anuncio de servicio** que posee las características comunes de ambos. Lo mismo ocurre con **partenariado** y **proyecto** que cuelgan de **Colaboración Uni-Soc**.

Queremos destacar que los atributos de **Necesidad social** y **Area de implementación**, tan importantes para emparejar ofertas y demandas, han sido obtenidos de la página web www.eoslhe.eu/resources/ Los partenariados y los proyectos presentan ciertas restricciones relacionadas con la participación de alumnos y promotores externos.

En los proyectos solo pueden participar estudiantes externos si el responsable del proyecto acepta estudiantes externos. Los profesores externos solo pueden participar en aquellos partenariados y proyectos que los acepten. Además, en el caso de un proyecto, un participante solo puede tener el perfil de profesor externo si también participan alumnos externos de su universidad, cuya participación tendrá que evaluar; en otro caso, tendrá el perfil de colaborador, independientemente del tipo de puesto que ocupa en su universidad.

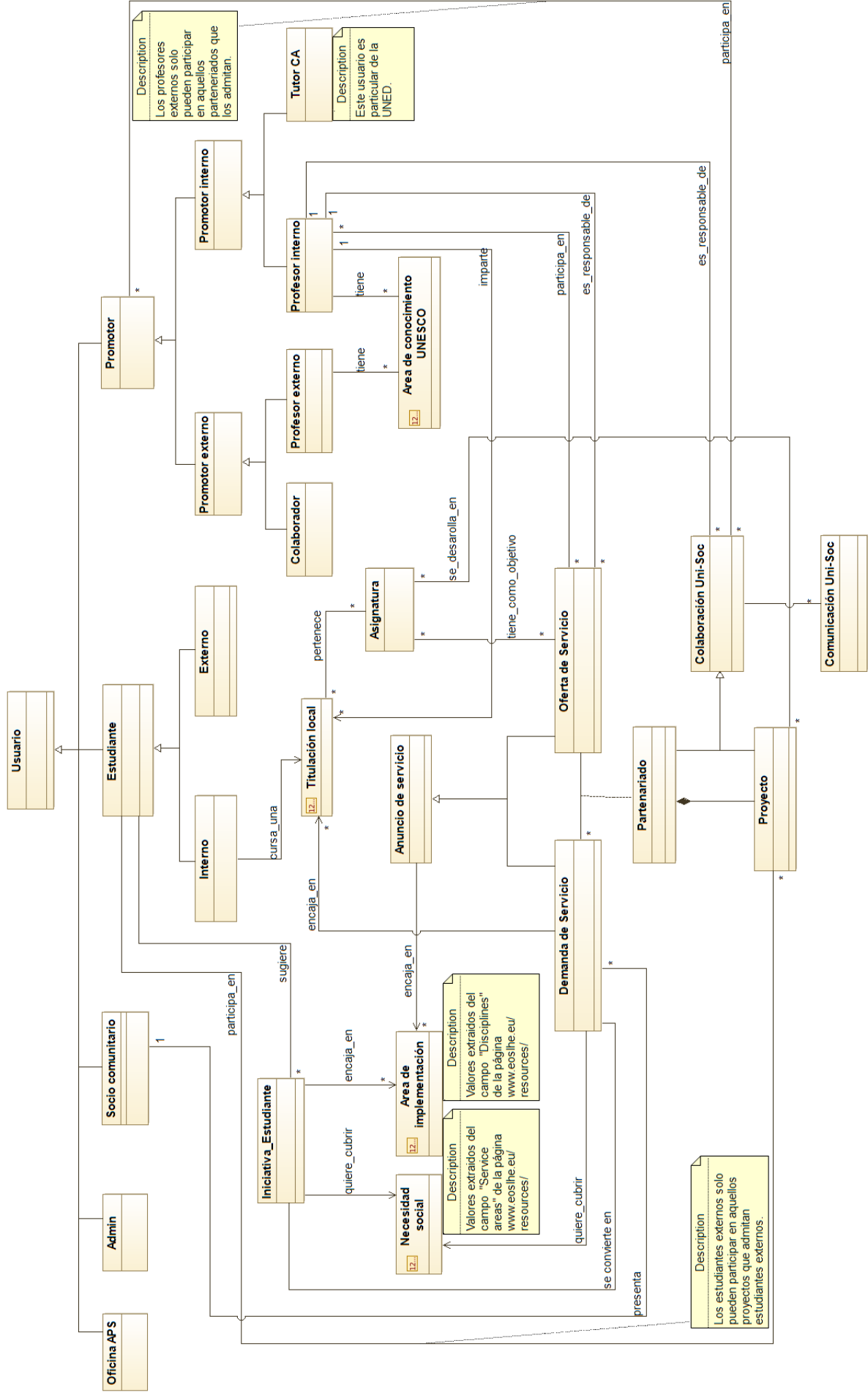


Figura 5.1: Modelo de dominio

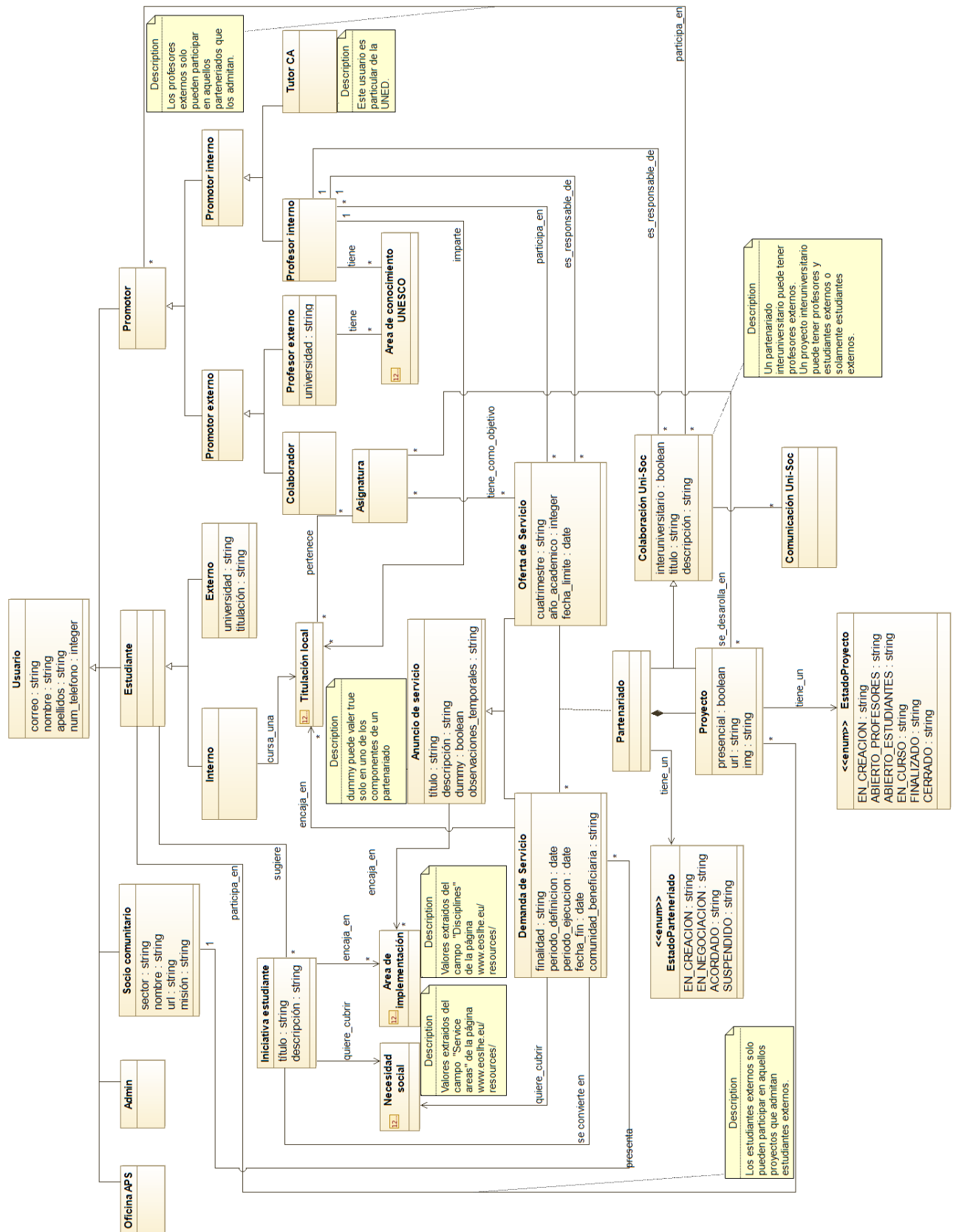


Figura 5.2: Modelo de datos

5.3. Modelo de datos

El modelo de datos describe con más precisión el dominio de la aplicación. Además de algunos atributos importantes como el *interuniversitario*, que indica si el partenariado o el proyecto está abierto a externos, podemos observar los estados que pueden tener el partenariado y el proyecto, representados por dos enumerados. En particular podemos ver que el partenariado puede encontrarse en estados *EN_CREACION*, *EN_NEGOCIACION*, *ACORDADO* y *SUSPENDIDO*. El modelo de datos se puede observar en la Figura 5.2.

- El partenariado se crea en el estado *EN_CREACION* o bien cuando un profesor interno respalda a una demanda presentada previamente por un socio comunitario o acepta un *match* de una oferta y demanda propuesto por la aplicación, o bien cuando un socio comunitario acepta una oferta presentada previamente por un profesor interno. Cuando la otra parte da su asentimiento a la creación del partenariado, su estado pasa a ser *EN_NEGOCIACION*. Cuando el profesor y el socio comunitario terminan de establecer los términos y condiciones del partenariado y ambos están de acuerdo con dichos terminos, el partenariado pasará a estar *ACORDADO*. Si ocurre cualquier discrepancia durante la fase de *EN_CREACION*, *EN_NEGOCIACION* o *ACORDADO* el partenariado puede pasar al estado de *SUSPENDIDO*.
- El proyecto toma los estados *EN_CREACION*, *ABIERTO_PROFESORES*, *ABIERTO_ESTUDIANTES*, *EN_CURSO*, *FINALIZADO* y *CERRADO*. El profesor interno responsable de un partenariado en estado *ACORDADO* puede pedir que se cree un proyecto derivado de este partenariado. El estado inicial de un proyecto es *EN_CREACION*. Cuando el socio comunitario da su asentimiento a la creación del proyecto, su estado pasa a ser *ABIERTO_PROFESORES*. Una vez terminada la definición del proyecto se abre a los alumnos y es allí cuando el proyecto pasa al estado de *ABIERTO_ESTUDIANTES*. Si el proyecto finaliza correctamente, pasará al estado de *FINALIZADO*. Si sucede cualquier imprevisto durante las 4 fases anteriores el proyecto puede pasar al estado *CERRADO*.

5.4. Modelo relacional

Debido a la complejidad de la nueva base de datos compuesta por 46 tablas relacionales, se ha decidido crear un diagrama que sirva como mapa en la gestión de la base de datos. Este modelo relacional se divide en cuatro secciones claramente diferenciadas.

- La sección de los usuarios que contiene tablas relacionadas con información de los usuarios. La sección del **Anuncio de servicio** que contiene todas las tablas que representan información de las ofertas de servicio, las demandas de servicio y las iniciativas.
- La sección de **Colaboración** contiene la información relacionada con los partenariados y los proyectos.
- La sección de Comunicación que contiene las tablas de *mail*, mensaje, *upload* y *newsletter*, estas fueron creadas por David Jiménez en los documentos de MongoDB cuya estructura se ha mantenido intacta. En la Figura 5.3 se puede observar el modelo relacional y sus cuatro secciones separadas por colores.



Figura 5.3: Diagrama de entidad-relación

5.4.1. Separación de datos

Una cosa importante para destacar es que, en un entorno real, la aplicación cogería parte de los datos de los usuarios de la base de datos de la universidad en la que se despliegue la aplicación. Es por esto por lo que los datos de los usuarios se han separado en dos grupos; internos y externos. Los usuarios internos son aquellos que pertenecen a la universidad en la que se ha desplegado la aplicación y por ello utilizan SSO (Single Sign-On) de la universidad para acceder a la aplicación. Esta separación de datos se ha hecho con el propósito de facilitar la transición entre la base de datos del prototipo y la base de datos de la aplicación real.

En concreto, los datos que son afectados por esta separación son el correo electrónico, el nombre, los apellidos y la contraseña que deberían alojarse en la base de datos de la universidad.

5.4.2. Usuarios

Esta sección es la más compleja debido a la separación que hay que realizar de los usuarios externos e internos. Ver Figura 5.4.

Un usuario interno es aquel profesor, tutor, estudiante, administrador o representante de la oficina ApS que forma parte de la universidad en la que se despliega la plataforma y por tanto tiene sus datos personales dentro de ella. Debido a que, en un despliegue real tendrían parte de los datos que necesitamos para la aplicación en el sistema interno de la universidad, hay que tratarlos de manera diferente a los usuarios externos, que son aquellos que no pertenecen a la universidad. Los colaboradores y directores que aquí mencionamos se pueden ver representados en el modelo de datos y de dominio, pero no se observan aquí porque no nos dio tiempo a integrarlos en la base de datos.

Debido a esta separación, todos los usuarios comparten una tabla común, llamada usuario, que contiene datos exclusivos de la cuenta de la plataforma. Después tenemos tablas que contienen datos particulares de cada tipo de usuario. Estas son las tablas del *socio comunitario*, estudiante interno, estudiante externo, *admin*, profesor interno, profesor externo y oficina ApS.

Cada uno de estos usuarios poseen una tabla que almacena sus datos personales, haciendo diferenciación entre internos y externos. La tabla de *datos_personales_internos* es una tabla creada para la simulación de la aplicación. Una vez la aplicación sea desplegada en un entorno real, esta tabla será eliminada y los datos personales se obtendrán haciendo consultas a la base de datos de la universidad.

Por otra parte, podemos observar tablas secundarias que representan características de los usuarios como, por ejemplo, la universidad del profesor y estudiante interno, las áreas de conocimiento UNESCO de los profesores y las titulaciones que imparten los profesores o cursan los alumnos.

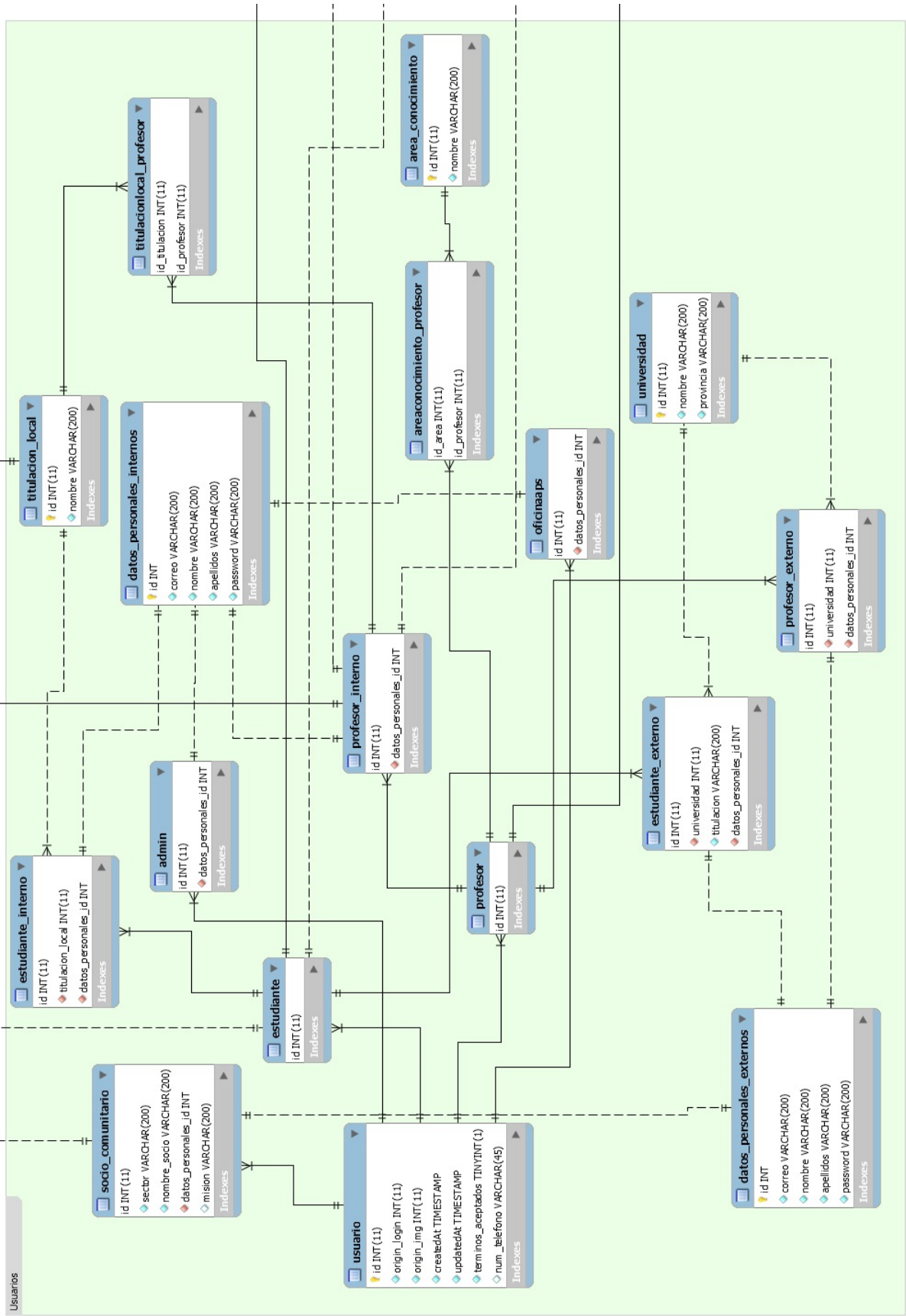


Figura 5.4: Diagrama de entidad-relación - Usuarios

5.4.3. Anuncios de servicio

En este conjunto de tablas podemos encontrar las pertenecientes a la demanda de servicio, la oferta de servicio y la iniciativa. Ver Figura 5.5.

- La iniciativa es una propuesta de proyecto realizada por un estudiante. Esta propuesta debe ser validada por la oficina ApS, que estudiará si la propuesta es viable para ser adoptada por un socio comunitario. Una vez validada la iniciativa, puede ser adoptada por un socio comunitario que desee realizar el proyecto.
- La demanda de servicio es creada por un socio comunitario y define una necesidad específica que quiere cubrir. Esta necesidad social es representada por los elementos de un enumerado alojado en la tabla de necesidad_social. Esta demanda
- La oferta de servicio es creada por un profesor interno y suele tener menos detalles que la demanda porque suele ser una propuesta más genérica.
Cuando una oferta y una demanda son procesadas por el sistema de *matching* se crea una entrada en la tabla *matching* almacenando los *ids* de ambos elementos y el porcentaje de emparejamiento que tienen.
Tanto demanda de servicio como oferta de servicio están conectadas a mensajes y *uploads* porque estos permiten la comunicación con las personas interesadas en las propuestas.

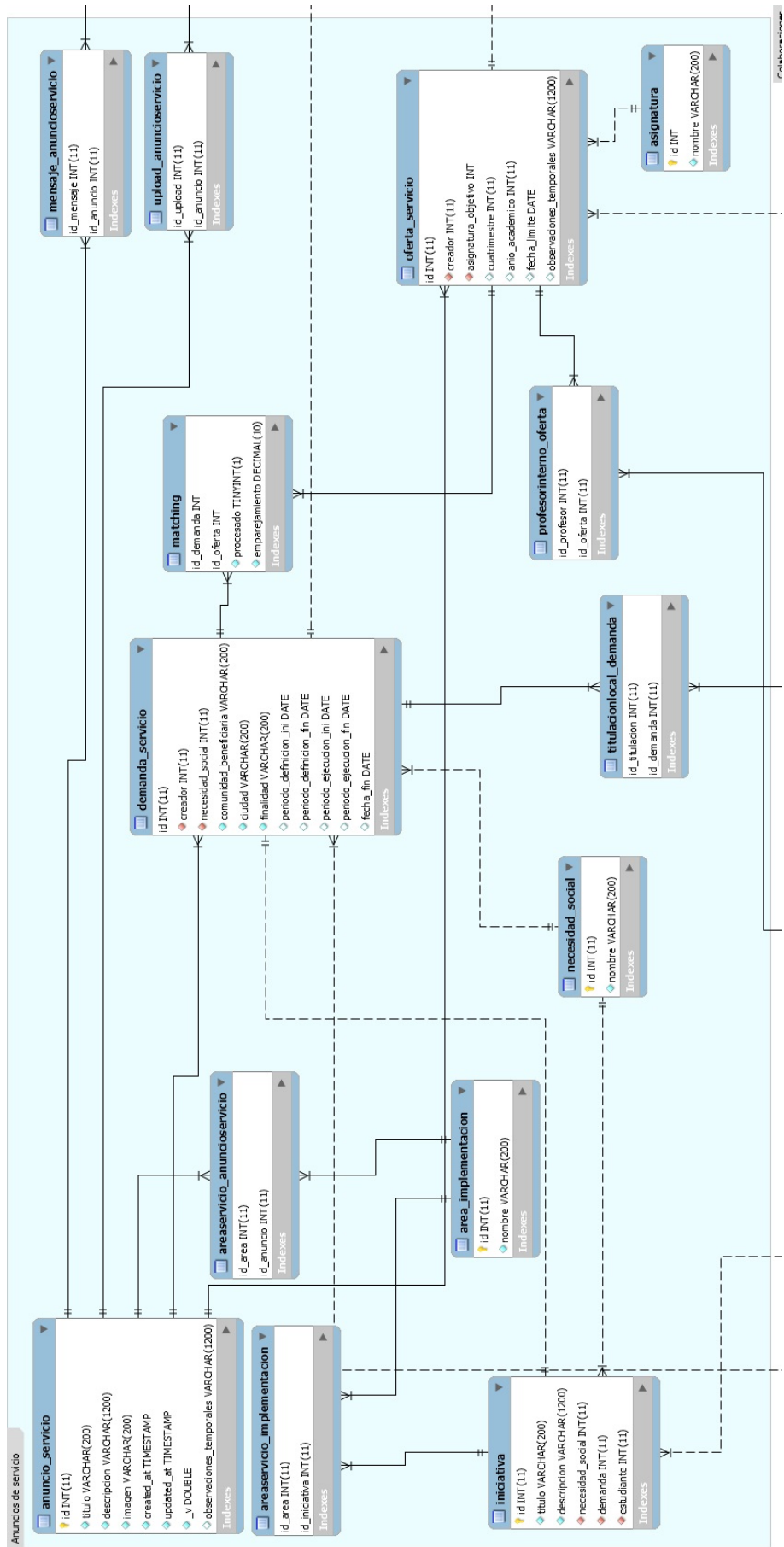


Figura 5.5: Diagrama de entidad-relación - Anuncios de servicio

5.4.4. Colaboración

El partenariado es el segundo paso en la creación de un proyecto. Esta tabla contiene los *ids* de la demanda y la oferta que la componen.

Un proyecto ApS no puede existir sin un partenariado previo y es por eso por lo que tiene un identificador del partenariado a partir del cual se creó. El proyecto posee estudiantes y por ello tiene una conexión con los mismos.

Tanto proyecto como partenariado necesitan un sistema de comunicación y es por eso por lo que tienen tablas intermedias que los conectan a mensaje y *uploads*. Ver Figura 5.6.

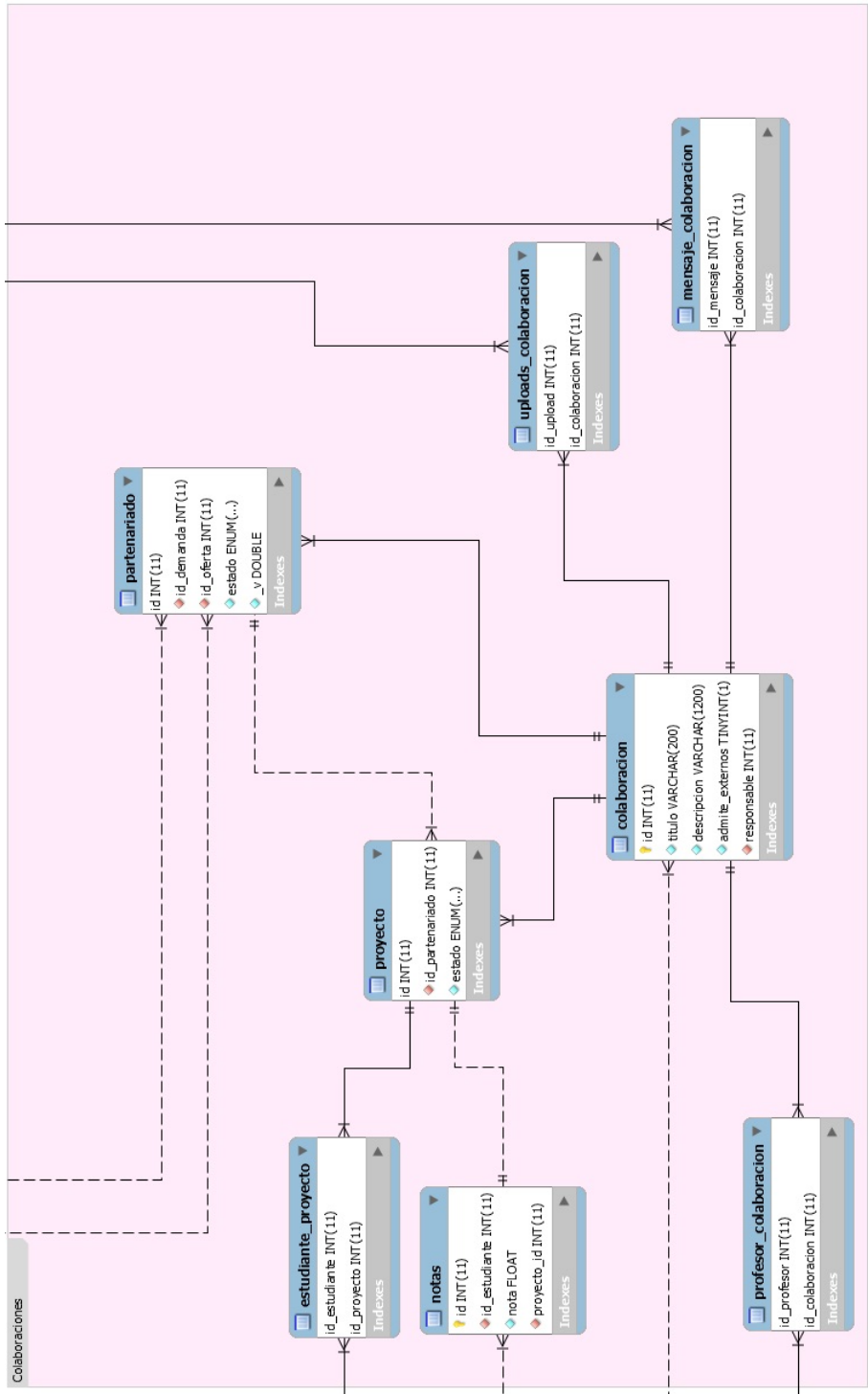


Figura 5.6: Diagrama de entidad-relación - Colaboración

5.4.5. Comunicación

Las funcionalidades de estas tablas no han sido del todo definidas, ya que en un principio se pensó que podrían comunicar al equipo docente con los socios comunitarios en los partenariados y los proyectos, pero también se podrían usar para la comunicación con los creadores de las ofertas y las demandas. Debido a que el funcionamiento no ha sido estudiado aun y que no hemos trabajado con estas tablas en nuestro TFG, se ha mantenido la misma estructura que había definido David Jiménez en sus colecciones de MongoDB adaptándola al modelo relacional. Ver Figura 5.7.

- La tabla de mensajes conecta con las ofertas de servicio, las demandas de servicio, los partenariados y los proyectos porque todos estos necesitan de los mensajes para poder comunicarse.
- La tabla *upload* almacena la información de los ficheros e imágenes subidos tanto en ofertas de servicio, como demandas de servicio, como partenariados y proyectos.
- La tabla *mail* y *newsletter* no han sido conectadas con ninguna otra tabla porque no se han tenido en cuenta para el desarrollo de este TFG, pero representan los correos electrónicos internos de la aplicación y las noticias periódicas enviadas a los usuarios.

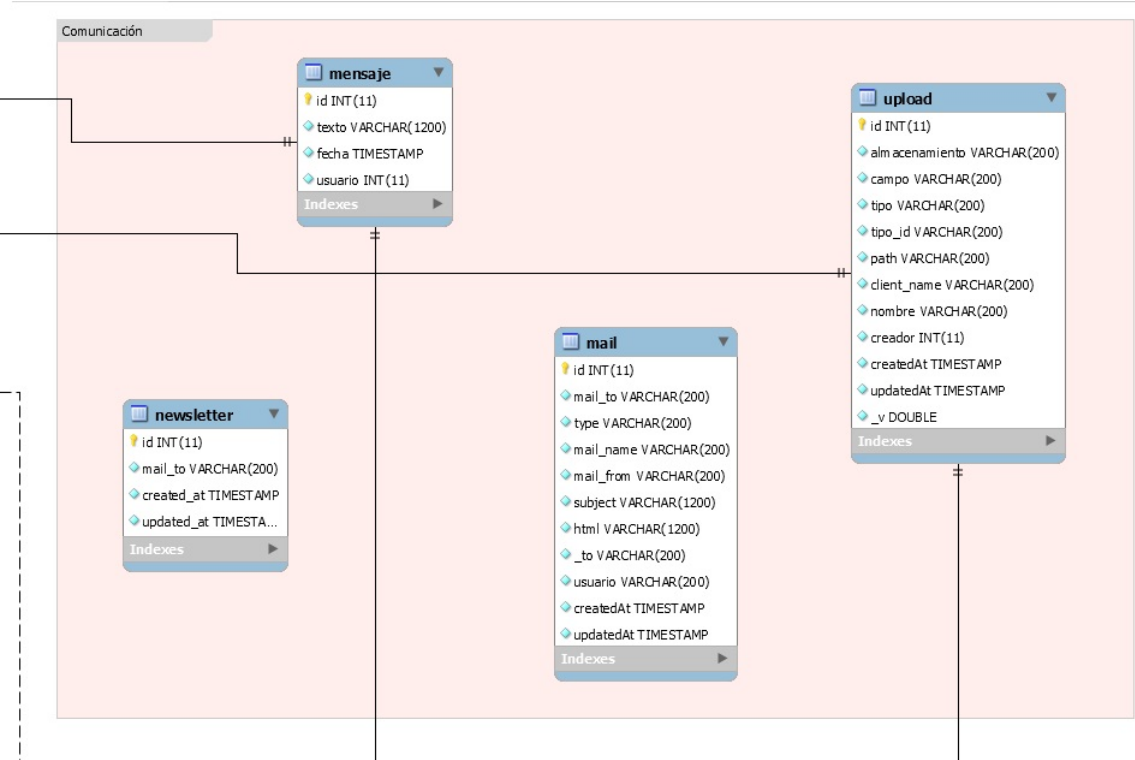


Figura 5.7: Diagrama de entidad-relación - Comunicación

Capítulo 6

DAO

Tras cambiar la base de datos de MongoDB por una relacional, también era necesario reimplementar la lógica de accesos a la base de datos, por lo que se crearon cuatro DAOs que se encargarían de las operaciones de cada una de las cuatro áreas definidas en el modelo de entidad-relación.

El DAO es un patrón de diseño el trata de proporcionar una interfaz para la comunicación con una base de datos u otro sistema de persistencia de datos. Esta interfaz se encarga de llevar a cabo las operaciones CRUD, es decir creación, lectura, actualización y eliminación de datos y además asegura la independencia entre la lógica de la aplicación y la capa de negocio.

Aunque no eran estrictamente necesarios dado que en Javascript no hace falta declarar el tipo de los objetos, se decidió crear objetos *transfer* para así tener más documentados los campos de cada tipo de objeto. Un *transfer* o *Data Transfer Object* es un objeto cuya única función es guardar la información de cierto objeto y permitir su acceso y manipulación. De esta forma, si hay algún problema, este se detectará cuanto antes y evitará que la aplicación falle repentinamente en fases más avanzadas de la ejecución.

Los objetos *transfer* contienen simplemente los atributos deseados de cada tipo de objeto además de las funciones *get* y *set* para poder acceder y actualizar la información de dichos atributos. En conjunto con los DAO, los *transfer* ayudan aún más a la separación de capas de negocio y lógica.

Los cuatro DAO que se crearon a partir del diagrama entidad-relación son:

- **DAOColaboracion:** Se encarga de manejar toda la información relacionada con los proyectos y los partenariados, desde sus participantes, ya sean profesores o estudiantes hasta los mensajes y archivos asociados a estos proyectos o partenariados. Este DAO se llama así porque tiene como piedra angular la clase **Colaboración**. Esta clase fue creada para hacer de padre de las clases **Partenariado** y **Proyecto** y así evitar la repetición de métodos y atributos similares. Utiliza los *transfer* **TColaboracion**, **TPartenariado** y **TProyecto**.
- **DAOComunicacion:** Se encarga de manejar toda la información relacionada con todas las formas de comunicación disponibles, desde los mensajes y los *uploads* que se pueden intercambiar durante las distintas fases de un partenariado o proyecto hasta los *emails* o las *newsletter* a las que se pueden suscribir los usuarios. Por lo tanto utiliza los *transfer* **TUpload**, **TMensajes**, **TMail** y **TNewsletter**
- **DAOTentativa:** Se encarga de manejar toda la información relacionada con ofertas y demandas y sus relaciones con la titulación local ofrecida por la universidad, las

áreas de servicio y las necesidades sociales que pudiera tener la demanda. Al igual que antes, se creó una clase padre llamada **Anuncio** para evitar la repetición de atributos en las clases **Oferta** y **Demanda** y en sus derivadas. Este DAO también se encarga de las iniciativas, que son propuestas de proyecto realizadas por un estudiante a la espera de que se le dé el visto bueno, y de los mensajes y *uploads* que pudieran tener tanto la oferta como la demanda. Para poder llevar a cabo esta función, este DAO utiliza los *transfer* **TIniciativa**, **TOfertaServicio**, **TAnuncioServicio** y **TDemandaServicio**.

- **DAOUsuario**: Se encarga de manejar los datos pertenecientes a las distintas clases de usuario, que son: profesor interno, profesor externo, estudiante interno, estudiante externo, *admin*, socio comunitario y oficina ApS. Además de estas clases, también interactúa con los respectivos padres de cada una de ellas y con las titulaciones locales, áreas de conocimiento y universidades que son necesarias para completar los atributos de los profesores. Para ello utiliza los *transfer* **TAdmin**, **TEntidad** (que deberá cambiarse en un futuro por **TSocioComunitario**), **TUsuario**, **TProfesor**, **TOficinaAPS**, **TEstudiente**, **TProfesorExterno**, **TProfesorInterno**, **TEstudienteInterno** y **TEstudienteExterno**.

Se ha intentado que los DAO tengan todas las funcionalidades necesarias para que la aplicación pudiera seguir funcionando tras sufrir cambios sin necesidad de actualizar los DAO con frecuencia, pero resulta imposible saber qué nuevas funcionalidades puede adquirir la aplicación o qué cambios podría sufrir el modelo de datos así que, aunque cuenta con bastantes funcionalidades, será necesario actualizarlo sobre la marcha si en un futuro la aplicación sufre cambios.

Capítulo 7

Matching entre oferta de servicio y demanda de servicio

7.1. Definición del matching

Para nuestro TFG hemos implementado un algoritmo de *matching* para las ofertas y las demandas, de manera que, cogiendo una oferta y una demanda de la base de datos, verificamos si tienen suficientes similitudes como para poder establecer un partenariado a partir de ellas. Partiendo de unas especificaciones del algoritmo que se estableció entre nosotros y los directores del TFG, las hemos aplicado para poder obtener la información relacionada representada por un porcentaje, con el cual se decidirá el resultado final. Cuantos más datos haya relacionados entre una oferta y una demanda, mayor porcentaje sacará nuestro algoritmo. El objetivo del algoritmo de *matching* es ayudar a los profesores a encontrar más fácilmente demandas relacionadas a sus propuestas y a las entidades a obtener ofertas acordes a sus solicitudes.

Nuestro *matching* no es tan sencillo como buscar un piso. Por ejemplo, cuando se busca un piso en función de lo que quieres, (tres habitaciones, un baño y una zona en concreto) te dan todos los pisos que cumplen esas características, en cambio nuestro *matching* no es así, porque las ofertas son muy genéricas, las demandas son específicas y los campos no son simétricos.

Definimos el *matching* según nuestro algoritmo, como el proceso que consiste en identificar los datos que se ajustan a unos criterios de coincidencia los cuales se van a enumerar a continuación en los siguientes párrafos. De este modo, si se encuentran suficientes puntos de similitud entre los datos recopilados, estos son considerados para sacar un porcentaje de coincidencia, donde si este es menor que el valor mínimo establecido no se considerará *matching*. En el prototipo que hemos construido, el valor mínimo que hemos establecido para considerar la existencia de un *matching* es 50 % pero, al igual que los pesos de cada factor, es configurable.

7.2. Criterios de matching y antimatching

7.2.1. Criterios de *matching*

Hemos definido unos criterios en base a los datos proporcionados por los usuarios en las ofertas y demandas de servicio según los cuales se resolverá el *matching*:

- Hacer coincidir las descripciones tanto de la oferta como de la demanda de servicio mediante NLP (*Procesamiento del Lenguaje Natural*). Puede encontrar más información sobre el NLP en la siguiente referencia [5]. Para ello hemos tenido que buscar las palabras vacías del idioma español y almacenarlas en un fichero, para así poder procesarlas para obtener el resultado deseado. El procedimiento consiste en lo siguiente: dadas las dos descripciones, las guardamos en dos estructuras simples de datos, quitamos las palabras comunes (aquellas que sean iguales a las del fichero) y nos quedamos con las que puedan coincidir en las dos descripciones. Cada una de estas se guarda en una estructura simple de datos. Al tenerlas, empezamos a procesar las palabras resultantes de las dos descripciones, distinguiendo las mayúsculas, minúsculas, tildes y caracteres especiales, donde obtenemos el número de palabras que coinciden de las descripciones. Para poder obtener el porcentaje de coincidencias, dividimos el número de palabras coincidentes con el número de palabras de la descripción más breve. Dicho porcentaje se tendrá en cuenta para poder calcular el valor final de *matching*.
- Encontrar la similitud entre las áreas de servicio tanto de la oferta como de la demanda. Se dispone de setenta y ocho áreas de servicio en la base de datos para poder realizar esta comprobación. Para ello se comparan todas las áreas de servicio de ambas, y se devuelve el número de coincidencias. Cuanto mayor dicho número, mayor probabilidad de que se produzca el *matching*.
- Obtener las coincidencias entre las titulaciones que el socio comunitario ha propuesto como candidatos en los que cuadrar un proyecto ApS a la hora de introducir la demanda (si es que ha introducido algunas) con las titulaciones en las que imparten docencia los profesores asociados a la oferta. Se dispone de ciento nueve titulaciones locales en la base de datos para poder realizar esta comprobación. Tanto la oferta como la demanda pueden tener una o varias titulaciones, y en función de la cantidad de titulaciones de la demanda, se calcula el resultado, el cual será un porcentaje obtenido a partir de la división del número total de titulaciones que producen coincidencias entre el número total de las titulaciones de la demanda. Este campo es opcional, y en el caso de que no esté relleno no se va a proceder dicha comprobación.
- Obtener las coincidencias en las restricciones temporales de la oferta de servicio y de la demanda de servicio para las que hemos aplicado NLP. Una vez obtenidas las dos restricciones temporales, tanto de la oferta como de la demanda, procedemos a aplicar el algoritmo para emparejar las palabras que coinciden en ambos lados y obtener un porcentaje. Para ello una vez más se quitan las palabras comunes de las descripciones y se guardan las palabras no comunes para cada una de las restricciones temporales en una estructura de datos. Tras esto, se procesan cada una de las estructuras, resultando el número de restricciones temporales que coinciden. El porcentaje de coincidencias se obtiene mediante la división del número de palabras coincidentes entre el valor (restricciones temporales) que tiene el menor número de palabras.

- Relacionar el área de implementación de la demanda y las titulaciones en las que imparten docencia los profesores que participan en la oferta. Para ello tenemos asignados al menos una titulación a cada área de implementación. Estas relaciones están almacenadas en una tabla de la base de datos. De esta manera se podrá sacar la relación directa o indirecta entre estos dos valores para así poder calcular un porcentaje válido para el resultado final de nuestro algoritmo de *matching*. Por ejemplo el área de implementación **Computer_science** se relacionaría con las titulaciones *ingeniería de Computadores*, *ingeniería Informática*, *ingeniería del Software*, *Telecomunicación*, etc. Para el cálculo del porcentaje se usa el mismo procedimiento que en los demás criterios: contamos el número de coincidencias y lo dividimos entre la cantidad total de las áreas de servicio.
- Encontrar la similitud entre el área de implementación de la demanda y las áreas de conocimiento UNESCO de los profesores que participan en la oferta. Para ello tenemos asignados al menos un área de conocimiento a cada área de implementación, estas relaciones están almacenadas en la tabla **matching_areas** de la base de datos. Se dispone de ciento noventa áreas de conocimiento en la base de datos para poder realizar esta comprobación. Para ello tenemos en cuenta las áreas de conocimiento con las cuales están relacionadas las áreas de servicio de la demanda y de la oferta como el principal valor en el cálculo del porcentaje. Para encontrar las posibles coincidencias contamos el número de ellas y lo dividimos entre las áreas de servicio de la demanda.

Para un futuro proyecto, se debería implementar la obtención de las coincidencias entre las titulaciones que el socio comunitario ha propuesto como posibles y las titulaciones a las que pertenecen la/s asignatura/s identificadas en la oferta como asignatura/s objetivo.

7.2.2. Criterios de *antimatching*

También hemos definido unos criterios de *antimatching* para encontrar posibles incompatibilidades entre los datos proporcionados.

Para poder realizar el *antimatching* nos hemos centrado en los valores temporales, en el caso de que tengan valor puesto, tanto de la demanda como de la oferta. Para ello hemos partido de si el comienzo del periodo de disponibilidad para trabajar en la definición de un proyecto ApS definido por la demanda no es posterior a la fecha límite para el fin de la realización del proyecto definida por la oferta. En el caso de que esté fuera del plazo se considera *antimatching* y se descartará la posibilidad de una negociación entre dicha demanda y oferta.

En el caso de que los plazos estén en el periodo aceptado, se procede a verificar si alguno de los siguientes se cumple:

- Si el fin del periodo de disponibilidad para trabajar en la realización del proyecto ApS es menor que el inicio del año académico definido por la oferta
- Si el fin del periodo de disponibilidad para trabajar en la realización del proyecto ApS definido por la demanda es menor que la fecha de inicio del cuatrimestre del año académico de la oferta

- Si el inicio del periodo de disponibilidad para trabajar en la realización del proyecto ApS es mayor que el fin del año académico definido por la oferta
- Si el inicio del periodo de disponibilidad para trabajar en la realización del proyecto ApS es mayor que la fecha de finalización del cuatrimestre del año académico definida por la oferta.

Si ninguno no coincide, se considera *antimatching* y en caso contrario se continúa con la comprobación de los siguientes valores temporales.

Otro criterio de *antimatching* es mirar si la fecha límite para el fin de la realización del proyecto ApS definido por la demanda es menor que la fecha de inicio del cuatrimestre en el año académico elegido de la oferta. De modo, si no se corresponden correctamente, se considera incompatibilidad y se descarta la negociación.

7.3. Obtención del porcentaje final de *matching*

Una vez obtenidos todos los porcentajes de los criterios de *matching* y los resultados del *antimatching* procedemos a combinar todos los criterios en un único porcentaje. Para ello multiplicamos cada uno de los porcentajes anteriormente mencionados por los pesos que le corresponden a cada uno definidos en el fichero `configuracion.txt` y son sumados para obtener el valor de compatibilidad entre la oferta y la demanda. El fichero `configuracion.txt` almacena en cada línea datos como `pesoFechas = 0.3`, `pesoTitulaciones = 0.3`, `pesoAreaServicio = 0.2`...

Si el valor obtenido es mayor que 0.5, valor que es configurable, se considerará un match definitivo y se almacenará en la base de datos en una tabla que contendrá el porcentaje final, el identificador de la oferta, el identificador de la demanda y un atributo booleano, *procesado*, que se pone a true indicando si paso por el proceso de verificación del match. En el caso de que una demanda y una oferta no pasaron por el algoritmo de *matching*, este atributo *procesado* tendrá el valor a falso. Si una oferta y una demanda pasaron por el proceso de verificación de *matching* tendrán guardado en la tabla el porcentaje de *matching*, que puede ser superior, inferior o igual a 0.5. En este último caso el atributo *procesado* se pone a true. La tabla de *matching* de la base de datos contendrá la información sobre los posibles match y no match de entre las demandas y ofertas procesadas, de esta manera la aplicación del algoritmo de *matching* se ejecuta una única vez por cada pareja oferta-demanda.

A partir de un *matching* de una oferta de servicio y una demanda se crea un partenariado en el caso de si el profesor quiere. Para ello primero, el profesor responsable de la oferta acepta el match y rellena el formulario que tiene algunos campos rellenos automáticamente a partir de información contenida en la oferta o en la demanda, lo que conlleva la creación de un partenariado en estado `EN_CREACION` y el envío de una notificación a la socio comunitario. Después, el socio comunitario podría aceptar el match,

rellenando un segundo formulario, teniendo algunos campos rellenos automáticamente a partir de información contenida en la oferta o en la demanda, lo que provocaría que el partenariado pasará al estado `EN_NEGOCIACION`.

7.4. Ejemplo de *matching*

A continuación, se muestra un ejemplo válido de *matching* con una oferta y una demanda dada, con un porcentaje de *matching* mayor del 50 %. Se expondrá la aplicación de cada uno de los criterios de *matching* y *antimatching*, y cómo se llegó al resultado final del algoritmo, de modo que se irá paso a paso por cada etapa del algoritmo que hemos creado.

Dada una oferta con los datos más significativos para el *matching*:

- Descripción: Proyecto de investigación en biología y tecnología
- Restricciones temporales: Me interesa que se empiece en septiembre
- Área de implementación: Biología, Tecnología digital
- Área conocimiento: Biología celular
- Titulaciones: Grado en Ciencias Ambientales
- cuatrimestre objetivo: Primer cuatrimestre
- Fecha límite para terminar la definición del proyecto: 2021/12/22

Y una demanda con los datos:

- Titulaciones: Grado en Ciencias Ambientales
- Descripción: Proyecto de investigación en biología
- Restricciones temporales: En septiembre 2021
- Área servicio: Biología
- Comienzo del periodo de disponibilidad para trabajar en la definición del proyecto: 2021/09/04
- Fin del periodo de disponibilidad para trabajar en la definición del proyecto: 2021/09/07
- Inicio del periodo de disponibilidad para trabajar en la realización del proyecto: 2021/09/14
- Fin del periodo de disponibilidad para trabajar en la realización del proyecto: 2021/12/01
- Fecha límite para el fin de la realización del proyecto: 2021/12/04

Se empieza comparando las fechas para detectar un posible *antimatching*. En el ejemplo las fechas cuadran, así que multiplicamos el peso de este atributo que es 0.3. Después de esto se comprueban las titulaciones del profesor que hace la oferta, con las que podría tener la demanda. Como coinciden totalmente se multiplica el peso de este atributo por 0.3 y de momento hay un 0.6 de compatibilidad entre la oferta y la demanda. Tras esto se comprueban las áreas de implementación junto con las áreas de conocimiento de los profesores y se multiplica el porcentaje de coincidencias por el peso de este atributo, que es 0.2. Y actualmente hay una compatibilidad de 0.66. Tras esto empezamos a buscar las coincidencias mediante NLP entre las dos descripciones, quitamos las palabras comunes de ambas y nos quedamos con las no comunes. La descripción de la oferta se queda en “Proyecto,investigacion,biologia,tecnologia” y la de la demanda se queda en “proyecto,investigacion,biologia”. Sacamos el porcentaje resultante entre el número de coincidencias que es tres y la longitud total de la descripción con menos valores que es tres, por lo que se consigue el máximo de coincidencias entre las dos descripciones. Se saca el porcentaje de similitud entre 0 y 100, en este caso es 0.60 y esto lo multiplicamos por 0.1 con lo que la compatibilidad actual sería 0.73. El mismo procedimiento se aplica también para las restricciones temporales, donde el número de coincidencias resultante es uno y el total de valores de la menor de las restricciones temporales es dos, por lo que el resultado tras ello es 0.33 de coincidencias, que se multiplica por 0.1 con lo que la compatibilidad actual sería 0.7366. La compatibilidad actual es de 0.74 y dado que es mayor que 0.5, se hace el *matching*

Como se puede observar, tras el número elevado de coincidencias, se produce el *matching* entre la oferta y la demanda y por lo consecuente se inserta la relación en la tabla *matching*.

Capítulo 8

Creación de formularios

Una vez adaptada la base de datos de no relacional a relacional y creados las correspondientes operaciones CRUD de las tablas, hemos pasado a la creación o a la adaptación de los formularios con los nuevos datos que le corresponde a cada uno de ellos. Para ello, tuvimos que aprender y experimentar con Angular, *framework* de Javascript, que requiere un vasto conocimiento para el desarrollo de los formularios y de los archivos para el correcto funcionamiento de estos. En un futuro proyecto debería añadirse a la izquierda de los campos una opción para obtener más información sobre cada uno de ellos.

8.1. Formulario de registro de usuarios

El primer formulario que tuvimos que tratar fue el registro de usuarios, siendo este el punto de inicio para los posteriores formularios a crear.

En la aplicación de la que partimos ya venía un registro implementado con sus correspondientes campos, pero al probar la aplicación habíamos encontrado algunos errores que tuvimos que corregir. Uno de los errores que encontramos era que el formulario permitía introducir una contraseña no robusta de cualquier longitud y sin ninguna restricción. De manera que la hemos hecho más robusta con una longitud mínima de 8 caracteres, mínimo una mayúscula, mínimo una minúscula y mínimo un carácter especial. Por otro lado, no se verificaba si el campo *email* era correcto de modo que tuvimos que crear las validaciones correspondientes por si no contenía el @ o el . .

En el formulario de David Jiménez existían dos campos para la contraseña: *contraseña* y *repetir contraseña*, donde ambos campos deben contener el mismo contenido para que se pueda permitir la validación del formulario y la creación del usuario. En cambio, si las contraseñas no coincidían lo aceptaba y se procedía al proceso de registro. Por eso tuvimos que añadir la validación y los mensajes de error correspondientes para que no se permita el registro si los dos campos no son iguales.

Una vez arreglados los errores, añadidas las mejoras y las nuevas validaciones del registro tuvimos que añadir los campos nuevos correspondientes a cada tipo de usuario de la base de datos.

En el formulario el *email* del usuario sirve como atributo único, de manera que, si un usuario intenta registrarse con un email que ya está en uso no se permitirá registrarse, llegando un mensaje de la aplicación de que el email ya está en uso por otro usuario.

En el formulario de registro se permite la elección de cuatro tipos de usuarios externos: profesor externo, estudiante externo, entidad y socios comunitarios. No están implementados el registro de los perfiles de colaboradores y Tutores de Centros Asociados de la UNED, donde deberían implementarse en un proyecto futuro.

Para el socio comunitario aparte de los campos ya existentes en el formulario se añadió un nuevo campo *nombre socio comunitario* que representa el nombre del socio comunitario principal que creará el formulario.

En el caso del estudiante externo no se han añadido nuevos campos en el formulario, pero sí el cambio de funcionalidad del campo *universidad*. Antes, en el campo *universidad* se permitía introducir un texto, pero se convirtió en un menú desplegable que permite una única selección por el estudiante de entre ochenta y tres universidades a elegir de España. No se permite la validación del registro si no se selecciona alguna universidad de la lista. En un futuro se debería añadir un campo de texto *universidad extranjera* que sea mutuamente exclusivo con el menú desplegable para que puedan participar profesores y alumnos de universidades extranjeras. Cuando un usuario va a elegir una universidad del menú desplegable, se bloqueará el campo de texto *universidad extranjera*, y si el usuario escribirá algo en el campo de texto, se bloqueará el menú desplegable.

Para el formulario del profesor externo se añadió un campo nuevo y se cambiaron las funcionalidades de algunos otros campos. El campo *universidad* tenía el mismo problema que en el caso anterior por lo que se produjo el mismo cambio. Además, tuvimos que introducir un nuevo campo *Área/s de conocimiento UNESCO*, que representa las áreas de conocimiento UNESCO que tiene el profesor externo a registrar. Este campo es un menú desplegable que permite la selección múltiple de entre las ciento noventa áreas disponibles que guardamos en la base de datos en la tabla `area conocimiento`. Se puede seleccionar entre una o varias áreas de conocimiento y en el caso de que se haya cometido un error al seleccionar un área se permite desmarcar la elección, pero no se permite la validación del registro en el caso de que no se haya seleccionado ningún área de conocimiento de la lista.

Para que la aplicación pueda tener seguridad, en un futuro debería haber algún tipo de Captcha para que se eliminen a los `bots`. Además, debería existir en el formulario de registro un campo de texto de tamaño para la autodescripción del registrante, donde la Oficina ApS será la encargada de dar el visto bueno a esta petición de registro después de hacer las comprobaciones pertinentes. La Oficina ApS tendría la posibilidad de seleccionar las peticiones de registro de los socios comunitarios pendientes y tratarlas al entrar en la aplicación. Otra opción a la alternativa de la intervención humana sería permitir el uso del SSO de Google, de Microsoft, de Facebook, etc., pero como no se puede exigir a que los usuarios que quieran participar tener una cuenta en Google, Facebook, etc., esta opción no encaja con una aplicación destinada al sector público y al tercer sector.

Aunque la figura 8.1 habla de tener cuenta en la UNED, en un futuro esto será aplicable a cuentas universitarias pertenecientes a la universidad en la que esta o la que este desplegada la aplicación.

Inicio · Qué es ApS · Cómo participar · Proyectos · Sobre ApS UNED · Registro · Login

Formulario de registro

Selección de perfil de usuario

En esta página puedes registrarte dentro de la aplicación Portal ApS.

Si deseas utilizar tu cuenta de la UNED, no es necesario que te registres, entra en la pantalla [Login](#) y utiliza el SSO de la UNED.

Si deseas utilizar tu cuenta de la Google como Estudiante, no es necesario que te registres, entra en la pantalla [Login](#) y utiliza el SSO de la Google.

Información de usuario

Perfil de Usuario (Estudiantes, Profesores, Entidades)

Profesor

Correo electrónico (se usará como login) *

Introduzca su Correo

Contraseña *

Repetir Contraseña *

Introduzca su Contraseña

Repita su Contraseña

Información personal

Nombre *

Apellidos *

Introduzca su Nombre

Introduzca su(s) Apellido(s)

Información de perfil

Universidad *

Introduzca su universidad

Area/s de conocimiento UNESCO *

Selecciona la/s area/s de conocimiento

Facultad / Escuela *

Introduzca su facultad

☐ Acepto las [Condiciones de Uso](#)

Regístrate

Figura 8.1: Formulario de registro

8.2. Formulario para editar los datos de un usuario

En el TFG de David Jiménez ya existía un formulario para la edición de los datos de un usuario con sus correspondientes campos, pero al cambiar el tipo de base de datos y al introducir nuevos datos en algunos de los tipos de usuarios tuvimos que adaptar este formulario.

También añadimos las validaciones para los campos *email*, *contraseña* y *repetir contraseña* como en el anterior formulario. Una vez añadidas las validaciones se introdujeron los campos necesarios para el formulario en función del tipo de usuario.

Para el socio comunitario aparte de los campos ya existentes en el formulario, se añadió el campo *nombre socio comunitario* que representa el nombre del socio comunitario principal que creará el formulario. Dicho campo viene con el valor ya rellenado, y no tiene la

posibilidad de cambiar su valor.

Para el estudiante externo no se han añadido nuevos campos en el formulario, pero sí el cambio de funcionalidad del campo *universidad*, que se convirtió en un desplegable que permite una única selección por el estudiante. No se permite la validación de registro si no se selecciona alguna universidad de la lista. El campo viene con el valor ya relleno, teniendo la posibilidad de cambiar su valor.

En el caso del profesor externo el campo *universidad* paso a ser un campo que no se puede cambiar su valor debido a que, si cambia de universidad inevitablemente cambiará también de dirección de mail corporativo. Además, tuvimos que introducir un nuevo campo *Área/s de conocimiento UNESCO* que representa las áreas de conocimiento de un profesor. Se puede seleccionar al menos un área de conocimiento y el valor viene ya relleno con los valores del usuario ya creado.

8.3. Formulario creación demanda de servicio

Para poder crear una demanda de servicio en la base de datos tuvimos que crear el formulario desde cero con sus correspondientes archivos puesto que en el anterior TFG no existía. Para ello se tuvo que crear su correspondiente modelo con los campos necesarios para la creación de la demanda: identificador, título, descripción, imagen, localización del servicio, objetivo, área de implementación, comienzo del periodo de disponibilidad para trabajar en la definición, fin del periodo de disponibilidad para trabajar en la definición, fin del periodo de disponibilidad para trabajar en la realización, comienzo del periodo de disponibilidad para trabajar en la realización, fecha límite para el fin de la realización, restricciones temporales, necesidad social, titulación local, creador, comunidad beneficiaria, fechas de creación y actualización.

El creador es el socio comunitario que está conectada en la aplicación y la cual accede a la creación de la demanda de servicio. Para el formulario de la creación de la demanda de servicio se han creado las validaciones correspondientes para los campos a completar, de manera que no se pueda permitir la creación de la demanda si alguno de ellos no es correcto y se muestran los mensajes correspondientes de error en este último caso.

En función de cada campo se permiten distintos valores con relación a la base de datos:

- Los campos *título, descripción, imagen, Localización/es donde se va/n a realizar el servicio, objetivo, necesidad social, restricciones temporales, comunidad beneficiaria* permiten introducir texto. El campo *necesidad social* es un requerimiento común de una sociedad respecto a los medios necesarios y útiles para su desarrollo y existencia.
- Los campos son de tipo fecha:
 - *Comienzo del periodo de disponibilidad para trabajar en la definición de un proyecto ApS*
 - *Fin del periodo de disponibilidad para trabajar en la definición de un proyecto ApS*

- *Comienzo del periodo de disponibilidad para trabajar en la realización del proyecto ApS*
- *Fin del periodo de disponibilidad para trabajar en la realización del proyecto ApS*
- *Fecha límite para el fin de la realización del proyecto ApS*
- El campo *Área/s de implementación* es un menú desplegable que permite selección múltiple entre un total de setenta y ocho áreas de servicios disponibles en la base de datos en la tabla `area servicio`. En el caso de que se haya seleccionado algún área de implementación por error, se puede descartar la selección. Hay que seleccionar por lo menos una opción para que se pueda validar el campo correctamente.

Además, hemos comprobado las relaciones de coherencia entre las fechas:

- La fecha límite para el fin de la realización del proyecto Aps puede ser inferior a la fecha finalización del periodo de disponibilidad para trabajar en la realización del proyecto si el socio comunitario necesita los resultados del proyecto antes de cierta fecha para tener disponibilidad en seguir colaborando en tareas de evaluación de los estudiantes hasta una fecha posterior. También puede ser mayor si el socio comunitario tiene disponibilidad para trabajar en la realización del proyecto hasta cierta fecha, pero los profesores y alumnos seguirán trabajando en ello después de esta fecha y el socio comunitario necesita los resultados antes de una fecha posterior.
- La fecha de finalización del periodo de disponibilidad en la definición de un proyecto Aps debe ser mayor que la fecha de comienzo del periodo de disponibilidad en la definición.
- La fecha finalización del periodo de disponibilidad para trabajar en la realización del proyecto ApS es mayor que la fecha de comienzo del periodo de disponibilidad para trabajar en la realización de este, y esta última mayor que la fecha actual de creación de la demanda
- La fecha límite para el fin de la realización del proyecto Aps es mayor que la fecha actual.
- La fecha límite para el fin de la realización del proyecto Aps es mayor que las fechas de finalización del periodo de disponibilidad en la definición y de comienzo del periodo de disponibilidad para trabajar en la realización del proyecto ApS.

Si alguna de estas comprobaciones no es correcta, no se valida el formulario y se muestra el mensaje correspondiente a esta. Una vez completados o seleccionados los campos a completar se comprueba el formulario, y en el caso de que estén todos los campos correctos se valida el formulario y se crea la oferta de servicio insertándose en la base de datos. En caso contrario, se avisa al usuario que los campos no están completados adecuadamente para que este proceda a su corrección.

8.4. Formulario creación oferta de servicio

Para poder crear una oferta de servicio es necesario que crear el formulario de creación de una oferta de servicio. En el anterior TFG no existía el formulario, así que tuvimos que

crearlo desde cero con sus correspondientes archivos para el correcto funcionamiento en angular.

Para el formulario de la creación de la oferta de servicio se han creado las validaciones correspondientes para los campos a completar de manera que no se pueda permitir la creación de la oferta si alguno de ellos no está correcto y los mensajes correspondientes de error en este último caso

En función del campo se permiten los siguientes tipos de datos:

- En los campos *título, descripción, asignatura/s objetivo, restricciones temporales* se permite introducir texto.
- En el campo *año académico* se permite un valor de tipo numérico que represente un año válido.
- El campo *fecha límite* permite un valor de tipo fecha. Este campo es la fecha límite para terminar la definición del proyecto.
- El campo *cuatrimestre objetivo* es un menú desplegable que permite una sola elección entre primer cuatrimestre, segundo cuatrimestre y anual. Si se selecciona un cuatrimestre sin seleccionar año académico, significa que el proyecto se desarrollara ese mismo cuatrimestre todos los años que dure el proyecto.
- El campo *Área/s de implementación* es un menú desplegable que permite selección múltiple. En el caso de que se haya seleccionado algún área de implementación por error se puede descartar la selección. Hay que seleccionar por lo menos una opción para que se pueda validar el campo correctamente.

Hemos comprobado los siguientes campos para poder validar el formulario correctamente:

- El campo *año académico* debe ser un año que no sea inferior al actual año.
- Si se especifica un año académico, la fecha límite para terminar la definición del proyecto y el año académico deben ser mayores que la fecha actual y menores que la fecha de comienzo del cuatrimestre especificado.

Consideramos como asignaturas TFM y TFG. Además, tuvimos que crear su correspondiente modelo con los campos necesarios para la creación de la oferta: identificador, título, descripción, imagen, fechas de creación y actualización, cuatrimestre objetivo, año académico, fecha límite, restricciones temporales, creador, área de implementación, asignatura (o asignaturas) objetivo y (o profesores) profesores. El creador es el profesor interno que está conectado en la aplicación y el cual accede a la creación de la oferta de servicio.

8.5. Formulario creación de partenariado profesor

Una vez creados los formularios de la oferta de servicio y la demanda de servicio, hemos procedido al desarrollo del formulario para la creación del partenariado por parte de un profesor. El formulario para la creación del partenariado no existía en el anterior TFG, así que se procedió a su creación desde cero. Para ello, tuvimos que crear los archivos y el modelo necesarios para el formulario. El formulario aparece con unos campos ya rellenos, algunos de los cuales son editables.

En la creación del formulario de partenariado por parte del profesor, los datos de la demanda de servicio del socio comunitario beneficiario vienen ya completados y sin posibilidad de editarlos. Cuando un profesor interno respalda una demanda de servicio sin haber definido previamente una oferta de servicio, el formulario no viene con los datos de la oferta. Si previamente la oferta de servicio existiera, el formulario vendría con los datos ya rellenos.

Hay dos maneras de crear un partenariado por parte de un profesor:

- Un profesor interno comunica su intención de respaldar una demanda de servicio que ha visto anunciada (y que ha sido creada previamente por un socio comunitario). El profesor rellena un formulario (no viene con datos de la oferta porque no existe) y se crea un partenariado en estado `EN_CREACION`. A continuación, el socio comunitario rellena un formulario y el partenariado pasa al estado `EN_NEGOCIACION`
- Un profesor interno comunica su intención de respaldar una demanda de servicio que le ha sido presentada como un match de una oferta de servicio de la que es responsable. El profesor rellena un formulario (puede venir con datos tanto de la oferta como de la demanda) y se crea un partenariado en estado `EN_CREACION`. A continuación, el socio comunitario rellena un formulario y el partenariado pasa al estado `EN_NEGOCIACION`

El formulario está dividido en tres partes para la distinción entre los datos generales del partenariado que serán la combinación de los datos en común o los datos más relevantes del formulario, los datos de la oferta y los datos de la demanda. También hemos creado validaciones para todos los campos del formulario, para comprobar que tienen los datos validos en el formato apropiado para cada campo.

En el formulario, el *título* y la *descripción* son una combinación de los *títulos* y *descripciones* de la oferta y la demanda. Estos campos son editables. Los datos de la demanda de servicio vienen rellenos, pero no son editables: las *áreas de implementación*, el *socio comunitario* de la demanda, la *localización de desarrollo* del partenariado, la *finalidad*, la *comunidad beneficiaria*, las *restricciones temporales*, *asignatura/s objetivo*, *titulaciones tentativas*, *cuatrimestre* y *año académico*. Aparece como *profesor responsable* el profesor de la oferta, siendo un campo editable que se da a elegir entre una lista de todos los profesores internos de la base de datos.

El campo *equipo de profesores* es una lista que da la posibilidad de selección múltiple y viene ya rellena con los valores de la oferta de servicio ya creada, en el caso de que exista la oferta de servicio. El profesor que procede a la creación del formulario elige si se aceptan personas externas. El *área de implementación* de la oferta no es un campo

editable. Si algún dato no es completado adecuadamente, se mostrarán los mensajes para que informe al usuario de los campos a cambiar o completar. Si se valida el formulario, el estado pasa a `EN_CREACION`. Si se rechaza pasa al estado `SUSPENDIDO`.

Crear Demanda

Demanda de Servicio
En esta página puedes crear una demanda de servicio para que los profesores puedan estudiarla y decidir si participar en ella.

Información de la demanda

Introduzca el título *

Descripción de la demanda *

Escribe una descripción sobre la demanda

Area de implementación *

Selecciona el area de servicio

Ciudad en la que se va a realizar el servicio, si va a ser de manera remota, escriba online

Objetivo que va a cumplir la demanda *

Comienzo del periodo de disponibilidad para trabajar en la definición de un proyecto Ap5

Fin del periodo de disponibilidad para trabajar en la definición de un proyecto Ap5

Comienzo del periodo de disponibilidad para trabajar en la realización del proyecto Ap5

Fin del periodo de disponibilidad para trabajar en la realización del proyecto Ap5

Fecha limite para el fin de la realización del proyecto Ap5

Si tiene algún requisito especial en cuanto a las fechas, escríbala aquí

Necesidad social a cumplir con esta demanda *

Comunidad Beneficiaria de esta demanda *

Si tiene una idea de una titulación o unas titulaciones en las que podría cuadrar el proyecto, escríbalas

Crear demanda

Figura 8.2: Formulario de creación de demanda

Crear Oferta

Título *

Título corto y descriptivo

Area de implementación *

Selecciona el area de servicio

Descripción *

Escribe una descripción sobre la oferta

Asignatura objetivo (Tenga en cuenta que también se consideran asignaturas un TFG o un TFM) *

Asignatura objetivo

Fecha limite para terminar la definición del proyecto

dd/mm/aaaa

Cuatrimestre Objetivo

Año académico

Año académico

Si tiene alguna petición especial en cuanto a las fechas, escribala aqui

Escribe cualquier observación temporal que consideres adecuada

Info

Una vez creados los datos básicos de la oferta, se podrán subir archivos e imágenes a la misma, así como editar sus datos básicos.

Crear oferta

Figura 8.3: Formulario de creación de ofertas

Crear partenariato

Información general

Título

Anuncio casa | Anuncio 9

Descripción

Construir una casa | Prueba 9

Externo ☐

Información demanda

Entidad

entidad1

Si tiene algún requisito especial en cuanto a las fechas de la demanda, escribala aquí

NADA

Ciudad en la que se va a realizar el servicio

Madrid

Necesidad social

No aplicar

Objetivo que va a cumplir la demanda

Construir una casa

Comunidad beneficiaria

0

Area de implementación

Arqueología, Letras, Educación, Servicio de aprendizaje, Investigar, General, Geografía, Historia

Titulaciones locales

Grado en Geografía e Historia, Grado en Historia del Arte

Comienzo del periodo de disponibilidad para trabajar en la definición de un proyecto ApS

04/06/2020

Fin del periodo de disponibilidad para trabajar en la definición de un proyecto ApS

04/07/2020

Comienzo del periodo de disponibilidad para trabajar en la realización de un proyecto ApS

05/07/2020

Fin del periodo de disponibilidad para trabajar en la realización de un proyecto ApS

11/08/2021

Fecha límite para el fin de la realización del proyecto ApS

01/08/2021

Información oferta

Responsable del partenariato

profesorInterno1 profesorInterno1

Si tiene algún requisito especial en cuanto a las fechas de la oferta, escribala aquí

quero un proyecto en enero a medianos

Figura 8.4: Formulario de creación de partenariato: parte 1

04/06/2020

Fin del periodo de disponibilidad para trabajar en la definición de un proyecto ApS

04/07/2020

Comienzo del periodo de disponibilidad para trabajar en la realización de un proyecto ApS

05/07/2020

Fin del periodo de disponibilidad para trabajar en la realización de un proyecto ApS

11/03/2021

Fecha límite para el fin de la realización del proyecto ApS

01/08/2021

Información oferta

Responsable del partenariado

profesorInterno1 profesorInterno1

Si tiene algún requisito especial en cuanto a las fechas de la oferta, escríbala aquí

quiero un proyecto en enero a medianos

Fecha límite para terminar la definición del proyecto

2021-11-04

Equipo de profesores

profesorInterno x

termo1

profesorInterno

termo1

Año academico

2020

Cuatrimestre objetivo

1

Asignatura objetivo (Tenga en cuenta que también se consideran asignaturas un TFG o un TFM)

Ampliación de Matemáticas, Matemáticas

Area de implementación

Arqueología, Letras, Educación, Servicio de aprendizaje, Investigar, General, Geografía, Historia

Crear

Portal ApS - © 2021

Figura 8.5: Formulario de creación de partenariado: parte 2

Capítulo 9

Conclusiones y trabajo futuro

9.1. Introducción

En esta última sección se hablará sobre el estado actual del proyecto y sus objetivos, así como de las mejoras que se podrían realizar sobre el mismo en un futuro.

Dado que este TFG es una continuación del proyecto de David Jiménez del Rey, se comenzará exponiendo las principales diferencias del proyecto actual con este último.

- La base de datos: además de haber cambiado la base de datos de MongoDB a una base relacional, se han realizado varios cambios en el modelo como queda explicado en la sección 5.4.
- La capa de acceso a datos: siguiendo el cambio en la base de datos, se han creado DTO y DAO para llevar a cabo la lógica de accesos a la BD como queda explicado en la sección 6.
- Implantación de un sistema de *matching*: el proyecto del año pasado trataba las ofertas y las demandas como simétricas, es decir, teniendo los mismos atributos, pero esto no es así, por lo que se ha diseñado un sistema de *matching* como queda explicado en la sección 7.
- Cambios en los formularios: debido a los cambios registrados en el modelo de datos, ha sido necesario crear algunos formularios o adaptar otros ya existentes. Esto queda mejor explicado en la sección 8.

9.2. Objetivos cumplidos

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

- *Construir unas bases sólidas del proyecto, creando un modelo de dominio que aclara los conceptos implicados en la aplicación y un modelo de datos que enriquece el modelo de dominio y plasma como la aplicación gestiona la información.*
Este objetivo ha sido cumplido y los diagramas de dominio y de datos se pueden ver en las Figuras 5.1 y 5.2, respectivamente.
- *Crear un modelo relacional que muestre la estructura de la base de datos, facilitando su entendimiento y manejo a los futuros desarrolladores del proyecto.*

Este objetivo también ha sido cumplido y el diagrama resultante se puede observar en la Figura 5.3

- *Crear una base de datos relacional compleja y rica en detalles.*
Este objetivo ha sido completado en su mayoría, aunque debido a la terminología empleada es posible que haya que renombrar alguna tabla. Esto se explicará con más detalle en la sección Trabajo Futuro.
- *Implementar cuatro DAO que realicen la lógica de acceso y gestión de datos, encapsulando el acceso a la base de datos. Crear transfers que permitan estructurar y manejar de forma sencilla los datos de la BD.*
Este objetivo también ha sido completado, aunque, como ya se ha dicho con anterioridad, es prácticamente imposible crear un DAO “perfecto” y es posible que se deban modificar en un futuro si se agregan funcionalidades nuevas a la aplicación.
- *Implementar un sistema de matching de los proyectos planteados por un profesor y los planteados por un socio comunitario que determina que porcentaje de encaje tienen.*
Este objetivo también ha sido completado y el sistema de *matching* queda explicado en la sección 7.
- *Adaptar las páginas de registro y perfil del usuario al nuevo sistema, e implementar formularios para la creación de ofertas, demandas y partenariados.*
Este objetivo no ha sido totalmente completado. Aunque las páginas de registro y perfil de usuario han sido actualizadas, y se han creado formularios para la creación de demandas y ofertas, en el caso de los partenariados, se plantean tres formas para poder crear un partenariado: *match* entre una oferta y una demanda ya creadas, un profesor decide respaldar una demanda, y un socio comunitario decide respaldar una oferta. Cada una de estas formas requeriría dos formularios, uno para el profesor y otro para el socio comunitario. Se ha hecho el formulario de *match* para el profesor y se ha empezado el desarrollo de la parte de *Back-End* para el formulario de la parte del socio comunitario. Esto se debe a la complejidad de trabajar con Angular para un equipo sin experiencia previa y a la falta de tiempo.
- *Corregir bugs encontrados en el proyecto precedente.*
Este objetivo se da como completado dado que la gran mayoría de bugs encontrados fueron arreglados al principio del proyecto, y aunque quedaban algunos, al modificar los formularios y tener que hacer cambios en las interfaces, los bugs ya serán corregidos en la reimplementación de estos formularios.

9.3. Problemas Encontrados

A continuación, se describirán las dificultades y problemas encontrados durante la realización de este proyecto.

La principal dificultad ha residido en la complejidad de trabajar con Node.js y con Angular, y la relativa poca experiencia previa del equipo con estas tecnologías. El equipo comenzó a familiarizarse con las tecnologías antes de comenzar el curso, pero aun así eran tecnologías complejas para quien no había trabajado con ellas previamente. Además, al dedicarse la mayoría del tiempo a la especificación y diseño de la base de datos y al diseño e implementación de las funcionalidades de *Back-End*, cuando se cambió a trabajar en el

Front-End, en el cual es donde se usa Angular, hubo que volver a “aprender” a trabajar con Angular.

A pesar de las dificultades aquí comentadas, se ha aprendido que Angular y Node.js son tecnologías muy acertadas para el diseño de aplicaciones web, tanto por el formato SPA que ofrece Angular como por la concurrencia alcanzada con Node.js.

9.4. Trabajo Futuro

Aunque se hayan completado la mayoría de los objetivos que tenía este proyecto, la aplicación aún no está lista para su despliegue y uso público, pero con otro año más de trabajo debería estar lista. A continuación, se enumeran algunas mejoras para el proyecto con este fin:

- Adaptar la interfaz y terminar los formularios: Como se ha explicado anteriormente al repasar los objetivos, todavía quedan formularios por hacer para cubrir todos los casos en los que se puede crear un partenariado, los cuales serían: el formulario de *match* del socio comunitario, el formulario para que un profesor pueda respaldar una demanda, el formulario para que el socio comunitario aceptara la propuesta de dicho profesor, el formulario para que un socio comunitario pudiera respaldar una oferta y el formulario para que el profesor aceptara esta oferta.
Debido a los cambios que ha sufrido la base de datos y a la implementación de los DAO, algunas vistas de la aplicación han dejado de ser funcionales. Será necesario cambiar los controladores para que en lugar de utilizar la lógica de acceso a base de datos de MongoDB utilicen la lógica proporcionada por el nuevo *Back-End*.
- Implementar un sistema de mensajería: Dado que la aplicación la van a poder usar tanto profesores como estudiantes y socios comunitarios y la comunicación será bastante importante, tanto en los periodos de definición del proyecto como cuando se empiece a trabajar en él, consideramos necesario que se implemente un sistema de mensajería o chat para hacer lo más eficiente posible la comunicación entre los usuarios. Incluso se podría implementar un sistema de foros para que los usuarios compartieran temas relacionados con el ApS o incluso para que los estudiantes pudieran ayudarse unos a otros en cuanto a temas algo más generales.
- Agregar parámetros opcionales en los formularios: Actualmente, algunos de los parámetros que ha de introducir el usuario en los formularios para crear ofertas y demandas podrían dejarse como opcionales. Esto podría interferir con el sistema de *matching*, pero dado que se permite cambiar el peso que tendrá cada atributo a la hora de calcular el porcentaje de *match*, una solución sencilla sería poner el peso de un atributo a cero si se detecta que dicho atributo está vacío.
Otra mejora sería hacer que el año de las fechas de la oferta fuera opcional. En caso de que el profesor no introdujera año, eso significaría que la oferta es vigente de manera cíclica todos los años en los periodos de tiempo que tenga establecidos.

Capítulo 10

Conclusions and future work

10.1. Introduction

This last section will be about the actual state of the project and its objectives, as well as the improvements that could be made in this project in the future.

Since this project is a continuation of David Jimenez del Rey's project the main differences between the two projects are going to be explained.

- The database: besides changing the database from MongoDB to a relational database, various changes have been made in the model as explained in section 5.4.
- The data access layer: following the changes made in the database, DTO and DAO have been created to handle the database access logic as explained on section 6.
- Implementation of a matching system: last year's project treated offers and demands as if they were symmetrical, that is, as if they had the same attributes, which wasn't the case, so a matching system was designed as explained in section 7.
- Changes in the forms: because of the changes that the data model underwent, new forms have been needed and old ones have been modified. This is better explained in section 8.

10.2. Objectives completed

Next, the objectives of this project and its completion will be reviewed:

- *Build a solid foundation for the project, creating a domain model that clarifies the concepts involved in the application and a data model that enriches the domain model and reflects how the application manages information.*

This objective has been met and the data and domain diagrams can be seen on Figures 5.1 and 5.2, respectively.

- *Create a relational model that shows the structure of the database, facilitating its understanding and management by future project developers.*

This objective has also been met and the resulting diagram can be seen on Figure 5.3

- *Create a complex and detailed relational database.*
This objective has been mostly completed, but, due to the terminology used some tables on the database may need to be renamed. This will be explained in more detail in the future work section.
- *Implement four DAOs that perform the logic of access and data management, encapsulating the access to the database. Create transfers that allow the database data to be structured and managed in a simple way.*
This objective has also been met, even though, as it has been said before, it's impossible to create a "perfect" DAO and they may be modified in the future if new functionalities are added to the application.
- *Implement a system of matching of the projects proposed by a teacher and those proposed by a community partner that determines what percentage of fit they have.*
This objective has also been met and the matching system is explained in chapter 7.
- *Adapt the registration and user profile pages to the new system, and implement forms for the creation of offers, demands and partnerships.*
This objective hasn't been totally completed. Even though registration and user profile pages have been actualized, and new forms have been made for the creation of demands and offers, in the partnership case, three ways of creating a partnership exist: match between an offer and a demand created beforehand, a teacher decides to support a demand, and a community partner decides to support an offer. Each of these ways needed two forms, one for the teacher and another for the community partner. The form for the teacher in the match way has been made and the development of the Back-End for the community partner form has begun. This is due to the complexity of working with Angular for a team without previous experience in said technology and due to the lack of time.
- *Correct bugs found in the previous project.* This objective is given as completed on account of most of the bugs being fixed in the early stage of the project and, even though some of them remained, since the interfaces and forms will be modified, the rest of the bugs will be fixed in the reimplementación of said parts.

10.3. Problems Found

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

The main difficulty has been the complexity of working with Node.js and Angular, and the team's relative lack of previous experience with these technologies. The team started working with these technologies before the scholar year started, but even then, the technologies were very complex. Furthermore, since most of the time was spent on the design and development of the database and the Back-End functionalities, when the team started working in Front-End, which is where Angular is used, they had to "relearn" Angular. Despite the difficulties, it has been learned that Angular and Node.js are very successful technologies for the design of web applications, both by the SPA pattern offered by Angular as the concurrency achieved with Node.js.

10.4. Future work

Even though most of the objectives of this project have been completed, the application isn't ready yet for its deployment and public use, but with another year of work, it should be. Next, some improvements for the project are enumerated:

- Adapting the interface and finishing the forms: As it has been explained before while reviewing the objectives, there are still forms to do to cover all the cases in which a partnership can be created: the match form for the community partner, the form needed for a teacher to support a demand, the form needed for a community partner to accept the proposal of said teacher, the form needed for a community partner to support an offer and the form needed for the teacher to accept the proposal of said community partner.

Due to the changes in the database and the DAO implementation, some pages are no longer functional. It will be necessary to change the controllers in order to use the new database access logic instead of the old one, which was prepared for MongoDB.

- Implement a messaging system: Since the application is going to be used by teachers, students and community partners and communication will be key, both in the definition of a project and in the realization, we deem necessary the implementation of a messaging or chatting system in order to make the communication between users as efficient as possible. A forum system could even be implemented so the users could share information related to ApS or even for students to be able to help each other in more general topics

- Add optional parameters in the forms: Now, some of the parameters that the user needs to introduce in the forms so he can create demands or offers could be optional. This might interfere with the matching system, but since the weight of the attributes can be changed, one simple solution could be setting the weight of an empty attribute to 0.

Another improvement would be letting the year set in the offer's dates be optional. In case that the year was left empty, this would mean that the offer is available every year in the periods of time set.

Capítulo 11

Contribución

A continuación, se detallan las contribuciones de cada uno de los componentes del grupo ordenados por orden alfabético.

11.1. Daniela-Nicoleta Boldureanu

La primera fase consistió en investigar, tarea que fue realizada por todos los miembros del TFG.

En la segunda fase del proyecto Daniela se ha encargado de arreglar *bugs* del trabajo anterior, tales como:

- Hacer que las contraseñas sean robustas y que el usuario no pueda registrarse con una contraseña que tenga menos de ocho dígitos, que no contenga ninguna mayúscula, minúscula o carácter especial. Además, arregló el mensaje de error de la contraseña defectuosa porque saltaba siempre cuando otro campo del formulario estaba mal.
- Comprobar que los campos *contraseña* y *repetir contraseña* del registro y editar perfil usuario sean iguales y mostrar el correspondiente mensaje al validar ambos formularios.
- Si se introducía un correo incorrecto no saltaba ningún mensaje de error.
- Corregir faltas de ortografía o palabras repetidas en algunas páginas de la aplicación.
- En el perfil del socio comunitario en la página para editar una iniciativa, proyecto o partenariado al subir un fichero o una foto, si luego se borraba y se intentaba subir de nuevo el mismo fichero/foto, no se subía.
- En el perfil del estudiante en la página de editar perfil si se subía una foto, se borraba y luego se volvía a subir la misma imagen, no se subía.

En la tercera fase del TFG Daniela se ha encargado de empezar el modelo de dominio y el modelo de datos usando Modelio. En paralelo ha creado las tablas **estudiante**, **estudiante_externo**, **estudiante_interno**, **usuario**, **oficina ApS**, **Admin**, **entidad**, **profesor-colaboración**, **profesor** y **profesor_interno** en la base de datos. También unifico todas las tablas de la base de datos una vez creadas. El proceso de unificar consistió en hacer que todos los nombres estén en el mismo formato, que todos los atributos tengan el mismo cotejamiento, que atributos del mismo tipo tengan la misma longitud y que los enumerados tengan el mismo formato. Con crear las tablas nos referimos a crear

la estructura, definiendo los nombres de los campos y los tipos de datos, y creando restricciones entre las tablas.

En la cuarta fase Daniela se ha encargado de desarrollar el DAO llamado *Usuario* que implementa el acceso a la base de datos del grupo de tablas llamado *Usuario* que se puede ver en la Figura 5.4. La creación de este DAO ha supuesto la necesidad de la creación de diez *transfers* que alojan los datos del elemento Usuario, Estudiante, Estudiante Externo, Estudiante Interno, Profesor, Profesor Externo, Profesor Interno, Entidad (socio comunitario), Oficina ApS y *Admin*. Los DAOs han sido implementados usando Knex.js, una librería que facilita las consultas a la base de datos. En el DAO *Usuario* se crearon las funciones CRUD de Usuario, Estudiante, Estudiante Externo, Estudiante Interno, Profesor, Profesor Externo, Profesor Interno, Entidad (socio comunitario), Oficina ApS y *Admin*. Todos están relacionados uno con el otro. También se creó el método para obtener todos los datos de los profesores internos en función de una estructura de datos que recibe como parámetro y que contiene los identificadores de unos profesores internos.

Después de finalizar la implementación del DAO *Usuario* Daniela desarrolló los métodos CRUD del elemento *Proyecto* perteneciente al DAO *Colaboración*. Este DAO implementa las tablas pertenecientes al grupo *Colaboración* que se puede ver en esta Figura 5.6. Además, fue necesario crear un transfer *Notas* y desarrollar los métodos CRUD del elemento *Nota*.

Daniela implementó en el DAO *Usuarios* cuatro métodos para obtener las titulaciones locales de un profesor interno en función de su identificador, obtener todas las universidades, obtener todas las titulaciones locales y obtener todas las áreas de implementación de la base de datos. Estos cuatro métodos han servido para el registro de usuario, la creación de la oferta, la creación de la demanda y del partenariado por el profesor.

Además, ha creado los siguientes scripts necesarios para insertar los valores de los enumerados con Python:

- Recoger las universidades de un Excel, procesar los valores y añadirlos en la tabla universidad de la base de datos.
- Recoger las titulaciones locales de un Excel, procesar los valores y añadirlos en la tabla `titulación_local` de la base de datos.
- Recoger las áreas de conocimiento de un Excel, procesar los valores y añadirlos en la tabla `área_conocimiento` de la base de datos.
- Recoger las áreas de servicio de un Excel, procesar los valores y añadirlos en la tabla `área_servicio` de la base de datos.

En la quinta fase Daniela se ha encargado de adaptar el código de obtener los datos de la página home y de crear los correspondientes métodos en el DAO *Colaboración*: obtener el número de proyectos, el número de partenariados y el número de iniciativas existentes en la base de datos.

También se ha encargado de crear los métodos del *matching* para obtener las coincidencias en las restricciones temporales de la oferta de servicio y de la demanda de servicio y de obtener las coincidencias de las descripciones tanto de la oferta como de la demanda de servicio mediante NLP. Para realizar este procesamiento adecuadamente fue necesario investigar. Para ello tuvo que buscar la lista de todas las palabras vacías del idioma español.

Una vez acabada la investigación y encontradas las palabras vacías, desarrollo el algoritmo.

En la quinta fase Daniela se ha encargado de adaptar, añadir nuevos campos y crear los mensajes correspondientes para los formularios de registro y editar usuarios. Adapto la aplicación tanto de *Front-End* como de *Back-End* en la parte de usuarios y home. Este proceso fue complejo dado que requería mucho conocimiento tanto en Angular como en Node.js y fue un paso importante para la creación de los formularios de la oferta, de la demanda y del partenariado por parte del profesor. También ha desarrollado la parte de *Front-End* del formulario para crear el partenariado por parte del profesor.

11.2. Victoria Gnatiuk Romaniuk

La primera fase era una fase de investigación así que en esta fase todos hemos investigado y no hay ninguna tarea individual que destacar.

En la segunda fase del proyecto Victoria se ha encargado de arreglar algunos *bugs* del trabajo anterior. Los *bugs* eran fáciles de arreglar, uno de ellos era que en el formulario de registro en los campos correo, nombre y apellidos se pedía al usuario que introdujera lo mismo, que era el nombre. Otra cosa que no era un *bug* sino más bien una mejora, era que al editar el perfil de usuario se pedía volver a aceptar las condiciones de uso, cosa que veíamos innecesaria.

En la tercera fase del TFG Victoria se ha encargado de continuar el modelo de dominio y el modelo de datos empezado previamente por Daniela. En paralelo ha creado las tablas **profesor_externo**, **iniciativa**, **anuncio_servicio**, **demanda_servicio**, **oferta_servicio**, **partenariado**, **proyecto**, **colaboracion**, **newsletter** y **entidad_demanda** en la base de datos. También ha creado las tablas que alojan los valores de los enumerados como la tabla **area_implementación**, **necesidad_social**, **area_conocimiento**, **universidad**, **titulacion_local** y las tablas intermedias que permiten conectar las tablas de los enumerados con las tablas que las referencian. Con crear las tablas nos referimos a crear la estructura, definiendo los nombres de los campos y los tipos de datos, y creando restricciones entre las tablas.

Una vez todas las tablas fueron creadas usando MySQL Workbench 8.0 CE, Victoria creó el modelo relacional a partir del fichero SQL. Después hubo que distribuir todas las tablas en los cuatro grupos explicados en el capítulo 5.3 para que todas las tablas fueran debidamente visibles.

En la fase cuatro Victoria se ha encargado de desarrollar el DAO llamado *Tentativa* que implementa el acceso a la base de datos del grupo de tablas llamado *Anuncios de servicio* que se puede ver en la Figura 5.5. La creación de este DAO ha supuesto la necesidad de la creación de tres *transfers* que alojan los datos del elemento Anuncio de servicio, Demanda de servicio y Oferta de servicio. Los daos han sido implementados usando Knex.js, una librería que facilita las consultas a la base de datos. En el DAO *Tentativa* se crearon las funciones CRUD, 3 de cada tipo concretamente. Dos de ellas eran las de la oferta y la demanda y la tercera, la cual siempre se llama desde las otras dos, es la del anuncio de servicio. También se creó el método para obtener todas las iniciativas, todas las demandas y todas las ofertas. Además de estos métodos, se han creado otros necesarios para la creación de los formularios de crear oferta y demanda.

Después de finalizar la implementación del DAO *Tentativa* Victoria desarrolló en los méto-

dos CRUD del elemento *Partenariado* perteneciente al DAO *Colaboración*. Este DAO implementa las tablas pertenecientes al grupo *Colaboración* que se puede ver en esta Figura 5.6. Después Victoria implementó en el DAO *Usuarios* dos métodos para obtener cualquier tipo de usuario con el id y con el correo del usuario. Estos métodos eran necesarios para la adaptación de la página de registro y *login* al nuevo sistema. Junto con Jesús, Victoria adaptó el *login* al nuevo sistema.

En la fase cinco Victoria se ha encargado de crear el método que determina la similitud que tiene los campos **area_implementación**, perteneciente a la *Oferta de servicio* y **area_conocimiento**, perteneciente a la *Demanda de servicio*. Para esto se ha tenido que crear un Excel donde los valores de los enumerados de **area_implementación** se han colocado en la primera fila y los valores de los enumerados de **area_conocimiento** se han colocado en la primera columna. Posteriormente se han colocado X para marcar la similitud entre los valores, esta similitud ha sido determinada por Victoria. Después de crear el Excel se ha creado un *script* con Python que leía este Excel e insertaba los emparejamientos en una tabla de la base de datos, de la cual se leerá posteriormente para determinar si estos dos enumerados pueden ser emparejados. A continuación, se creó el método que compraba estos enumerados. Se hizo este mismo proceso para determinar el emparejamiento entre **area_implementación**, perteneciente a la *Oferta de servicio* y **titulacion_local**, perteneciente a la *Demanda de servicio*.

Además de estos métodos para determinar el *matching*, Victoria implementó otros dos. Uno de ellos determinaba si las dos **areas_implementación**, pertenecientes a Oferta de servicio y Demanda de servicio coinciden. El segundo método determinaba si coincidían las titulaciones de la *Demanda de servicio* y las titulaciones que imparte el profesor que ha creado la *Oferta de servicio*.

En la fase cinco Victoria se ha encargado de implementar el formulario para crear la oferta de servicio y también ha desarrollado la parte *Back-End* del formulario para crear el partenariado por parte del profesor y por parte del socio comunitario. Juntamente con todo esto Victoria ha ido actualizando los modelos de dominio y de datos a medida que iban sufriendo cambios a causa de los cambios sugeridos por los directores del TFG. También ha mantenido actualizado el diagrama relacional a medida que la base de datos iba sufriendo cambios.

11.3. Jesús Sánchez Granado

La primera fase del proyecto era principalmente una fase de investigación, así que en esta fase todos hemos leído la memoria de David Jiménez del Rey y se ha aprovechado para preparar el entorno de trabajo y crear el repositorio de Github al cual se irán subiendo los avances en el proyecto. Además, de manera conjunta se realizaron pruebas de robustez en la aplicación web con el objetivo de intentar encontrar *bugs* o posibles mejoras.

En la segunda fase del proyecto Jesús ha arreglado algunos *bugs*. Uno de ellos era un mal redireccionamiento de un enlace y otro, más que un *bug* era una mejora, pues al editar una iniciativa siempre pedía aceptar los términos y condiciones, cosa que se consideró innecesaria.

Algunos de estos *bugs* se han dejado por hacer pues al tener que realizar cambios en la interfaz, los *bugs* ya serían arreglados al reimplementar las vistas pertinentes.

En la tercera fase del proyecto, Jesús se ha encargado de crear la estructura de las tablas **profesorinterno_oferta**, **estudiante_iniciativa**, **estudiante_proyecto**, **upload**, **upload-anuncioservicio**, **mail**, **uploads-colaboracion**, **mensaje**, **mensaje-anuncioservicio** y **mensaje-colaboracion**. En dicha estructura quedaban especificados los nombres y tipos de sus campos, sus restricciones, las claves foráneas y quedaban definidas también sus relaciones, siempre y cuando fueran con tablas de este grupo ya nombrado.

Una vez se crearon las tablas restantes y se unificó el formato de las mismas, Jesús realizó las conexiones entre las tablas que aún no habían sido relacionadas. Esto se debe a que, para repartir la creación de tablas, estas se dividieron en grupos, y había algunas que se relacionaban con tablas de otros grupos, por lo que no se podían relacionar hasta tener la base de datos completa.

En la cuarta fase del proyecto Jesús se ha encargado de desarrollar el DAO llamado **Comunicacion**, el cual implementa el acceso a la base de datos para el grupo de tablas que se puede ver en la Figura 5.7. Para poder crear este DAO, antes ha necesitado crear los *transfer* **TUpload**, **TMensajes**, **TMail** y **TNewsletter**. Estos *transfer* se encargan, respectivamente, de almacenar los datos de los elementos *upload*, *mensaje*, *mail* y *newsletter* además de permitir modificarlos. Este DAO contiene las funciones CRUD, es decir, las necesarias para crear, leer, actualizar y eliminar elementos en las respectivas tablas de la base de datos, en total unas 22 funciones.

Esto se debe a que tanto los mensajes como los *uploads* pueden pertenecer o a una colaboración o a un anuncio de servicio, por lo que cada uno tiene dos funciones distintas para su creación, una para cada caso. Además, se han hecho también funciones para obtener todos los mensajes y todos los *uploads* de una determinada colaboración o de un determinado anuncio de servicio.

Terminado el DAO en cuestión, Jesús creó los *transfers* **TColaboracion**, **TProyecto** y **TPartenariado**, los cuales eran necesarios para la creación del DAO **Colaboracion**. Además, en dicho DAO hizo las funciones CRUD del *transfer* **TColaboracion**. Tras esto, junto con Victoria, ambos adaptaron la funcionalidad de *login* al sistema actual.

En la quinta fase del proyecto Jesús ha creado el método que se encarga de comprobar si el periodo de disponibilidad de la demanda coincidía con el cuatrimestre o cuatrimestres especificados por el profesor en la oferta de servicio. También se cubrieron los casos en los que se consideraba que había un “*antimatching*”, situaciones en las que, aunque otros atributos coincidieran, el *match* no se llevaría a cabo. Por ejemplo, si los periodos elegidos para la definición del proyecto escogidos por el profesor y por el socio comunitario no fueran compatibles, aunque todos los demás atributos lo fueran, no habría *match*.

Tras esto, y una vez Daniela y Victoria terminaron sus respectivos métodos, Jesús hizo la función “maestra” que agrupaba todos los métodos de *matching* hechos anteriormente, después de moodificarlos ligeramente para que fuera más fácil hacer los cálculos de los resultados, teniendo en cuenta los respectivos pesos de los atributos. Una vez se ha calculado el porcentaje de *matching*, si este es mayor que el 50 %, se guarda el resultado en la base de datos con el fin de notificar a los usuarios pertinentes.

Esta función además permite configurar los pesos de los atributos para así hacerla más versátil a la hora de priorizar qué están buscando los usuarios, de momento esto se ha hecho creando un fichero llamado **configuracion.txt** para que la función lea los valores en él descritos y los asigne.

En la sexta fase del proyecto Jesús ha implementado el formulario para la creación de una

demanda de servicio. Esto, además de la creación de un fichero *crear-demanda.component.html* y un fichero *crear-demanda.component.scss* para mostrar la vista al usuario y de un fichero *crear-demanda.component.spec.ts* y un fichero *crear-demanda.component.ts* para encargarse de la lógica asociada al formulario, ha supuesto además cambios en los *controllers* y en el fichero de rutas. Además de esto, ha cambiado algunos campos en los formularios para hacerlos más legibles para el usuario.

Bibliografía

- [1] Natalia Rodriguez-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.*
- [2] 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.*
- [3] David Herron. *Node.js Web Development: Server-side development with Node 10 made easy, 4th Edition.* Packt Publishing Ltd, 2018.
- [4] Greg Lim. *Beginning Angular 2 with Typescript.* CreateSpace Independent Publishing Platform, 2017.
- [5] M.A. Martí, M.A.M. Antonín, J.L. Boix, and J. Llisterri. *Tratamiento del lenguaje natural: tecnología de la lengua oral y escrita.* Colección UB. Universitat de Barcelona, 2002.
- [6] Héctor 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.*
- [7] 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.*
- [8] M.Á.S. Rego, A.S. Losada, and M.L. Moledo. *Aprendizaje-servicio y misión cívica de la universidad: Una propuesta de desarrollo.* Ediciones Octaedro, 2015.
- [9] Natalia Rodriguez. Virtu@l-aps. *Director: Juan García Gutierrez, UNED. Titulación: Grado en Educación Social. 2017.*
- [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: Ingeniería Informática de la UNED. Sept. 2015.*