UNIVERSIDAD COMPLUTENSE MADRID

Facultad de Ingeniería Informática

Aplicación web de soporte al Aprendizaje-Servicio

Autores

Daniela-Nicoleta Boldureanu (Grado en Ingeniería del Software)

Victoria Gnatiuk Romaniuk (Grado en Ingeniería Informática)

Jesús Sánchez Granado (Grado en Ingeniería Informática

Tutores

Simon Pickin

Manuel Montenegro Montes

Índice

Índice de figuras

1. Contexto de la propuesta

1.1. Introducción

El aprendizaje-servicio es una propuesta educativa que combina aprendizaje y servicios a la comunidad. Estos proyectos 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 darles la oportunidad de aplicar sus conocimientos en un entorno real, les impulsa a comprender el funcionamiento de la sociedad y las responsabilidades sociales que estos tienen por forma parte de una sociedad.

Todo proyecto ApS empieza por una iniciativa 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.

1.2. Elementos que intervienen en un proyecto ApS

- 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 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 a la entidad. El profesor se encarga de acordar y organizar los proyectos con la entidad, estableciendo todos los requisitos necesarios para la correcta formación del alumno y el cumplimiento de la entidad con los principios del ApS.

- La entidad es una empresa pública o privada que colabora con la institución educativa para resolver un determinado problema social. La entidad suele tener en mente un problema muy concreto, pero no lo suficientemente detallado para la creación de un proyecto real. Es por eso que es necesario el partenariado. Es importante que la universidad haga entender a la entidad 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 introduciendo lo en un entorno real para que este establezca una relación entre lo aprendido en el aula con lo realizado en el proyecto ApS.

- El partenariado es una colaboración entre el profesor y la entidad. Partiendo de un problema social real y los conocimientos dispuestos por el profesor, el profesor y la entidad realizan reuniones para determinar las características y particularidades del problema. Una vez definidos los términos y condiciones del futuro proyecto, la entidad y el profesor abren el proyecto a los alumnos.
- 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 estar tareas o servicios, el alumno beneficia a su comunidad otorgándole, una satisfacción personal. El alumno es evaluado de forma continua por el profesor.

1.3. Motivación

El principal problema de los proyectos ApS es que son difíciles de definir y aún más difíciles de acordar.

La entidad suele tener un problema concreto en mente, pero no lo suficientemente especifico ni definido como para crear un proyecto. El profesor por su parte tiene una serie de conocimientos que quiere ofrecer para la resolución de un problema genérico. Este inconveniente anima a Simon Pickin y otros profesores de la UNED a desarrollar una solución.

Nuestro TFG es la cuarta parte de un proyecto destinado a ser una plataforma real para ayudar a universidades y entidades a establecer contacto y definir proyectos ApS en pro de una educación más práctica e inclusiva a la sociedad.

La parte previa a la nuestra fue desarrollada por David Jiménez el año pasado. Él creó el esqueleto del aplicativo sobre el que nosotros hemos desarrollado nuestro TFG.

1.4. El TFG de David

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

Él implementó las páginas, de registro, *login*, perfil, iniciativa y partenariado. También incluyó diferentes perfiles como el *admin*, la entidad, el profesor y el alumno.

Por razones de comodidad y familiarización con la tecnología, cambió la base de datos que estaba construida en SQL a Mondo DB, esto no gustó mucho a nuestros tutores los cuales nos pidieron que la volviéramos a cambiar a SQL.

El problema del proyecto de David residía en que él, no tuvo mucha comunicación con los profesores durante la realización de su TFG debido a problemas de salud ocasionados por el COVID-19. Esto derivó en que no entendiera bien los requisitos deseados para el

proyecto y que tuviera una confusión sobre los conceptos y el objetivo que se intentaba conseguir con la plataforma ApS. Por ello nosotros hemos tenido que redefinir las bases del aplicativo creando un modelo de dominio y de datos que representa todos los elementos del problema y como se relacionan entre si. 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.

1.5. Nuestro objetivo

A continuación se explican los objetivos que hemos tenido en este TFG.

- Construir unas bases solidas del proyecto, creando un modelo de dominio y un modelo de datos que ilustran el objetivo y el funcionamiento de la plataforma.
- Creando una base de datos relacional compleja y rica en detalles pasando de 7 documentos a 46 tablas relacionales.
- Crear un modelo relacional que muestre la estructura de la base de datos facilitando su entendimiento y manejo a los futuros desarrolladores del proyecto.
- Implementar 4 daos que realicen la lógica de acceso y gestión de datos, encapsulando el acceso a la base de datos.
- Creación de *transfers* que permiten estructurar y manejar de forma sencilla los datos de la BD.
- Implementar un sistema de casa de ofertas y demandas que determina que porcentaje tienen de encajar una oferta creada por un profesor y una demanda creada por una entidad.
- Adaptación del registro y el perfil del usuario al nuevo sistema.
- Implementación de formularios para la creación de ofertas, demandas y partenariados.
- Corrección de *bugs* del proyecto de David.

1.6. Estudio tecnológico

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

1. Servidor web:
    - ExpressJS es un *framework* basado en NodeJS que permite gestionar el servidor de una forma sencilla. Este *framework* fue utilizado por David y se ha mantenido.
2. Backend:
    - NodeJS es entorno basado en JavaScript muy popular. Este entorno fue usado por David y se ha mantenido.

3. Frontend
    - Angular es un *framework* utilizado en el *Frontend* que David utilizaba ya en el proyecto y se ha decidido mantener.
4. Base de datos:
    - MongoDB: es un sistema de base de datos estructurado. Este sistema es el que se estaba usando en el proyecto.
    - SQL: es un sistema de base de datos relacional muy potente. Decidimos utilizar este sistema en nuestro TFG.
5. Software de control de versiones:
    - GIT es el controlador de versiones más conocido y 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 ficheros menos el código.
7. Herramientas de organización:
    - GitHub Projects es una herramienta que ofrece GitHub que permite crear una organización de proyecto tipo Kanban.
    - Es una herramienta sencilla estilo Kanban para organizar los proyectos pero tiene muchas limitaciones de pago.
    - 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 *online* de dibujo sencilla que empezamos a usar para la creación de los modelos de datos y de dominio.
    - 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 conociamos, la hemos elegido para la creación de nuestros modelos.
9. Herramienta para el modelo relacional:
    - 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.

2. Tecnologías utilizadas

- Node.js: 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 si es que necesitamos los datos producidos por la anterior promesa. Si durante la ejecución de un programa no hay nada que hacer, node.js se "dormirá". Dado que node no utiliza candados es imposible que se bloqueen los procesos, lo que lo hace bastante adecuado para desarrollar sistemas escalables. Además Daniela ya tenía conocimiento previo de este entorno y es una tecnología que venía impuesta por el proyecto.

- Angular: Es un *framework* para la construcción de aplicaciones de página única(SPA a partir de ahora) que utiliza HTML y Typescript. Angular sigue el patrón modelo-vista-controlador, el cual consiste en separar la aplicación en tres partes:

  - El modelo: Es la piedra angular del patrón, se encarga de manejar los datos y la lógica de la aplicación.
  - La vista: Es la parte que se le muestra al usuario.
  - El controlador: Es la parte que se encarga de comunicar a la vista y al modelo, el controlador recibe el input del usuario a través de la vista y se lo pasa al controlador el cual hace las operaciones necesarias y se lo devuelve al controlador, quien se lo pasa a la vista para mostrárselo al usuario.

  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 liviano 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, lo que la hace una buena elección para un trabajo de esta índole.

- Gitkraken: Es una herramienta de control de versiones 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 herramienta para nuestro control de versiones porque permite trabajar desde Windows sin necesidad de saberse los comandos, a diferencia de otras herramientas de

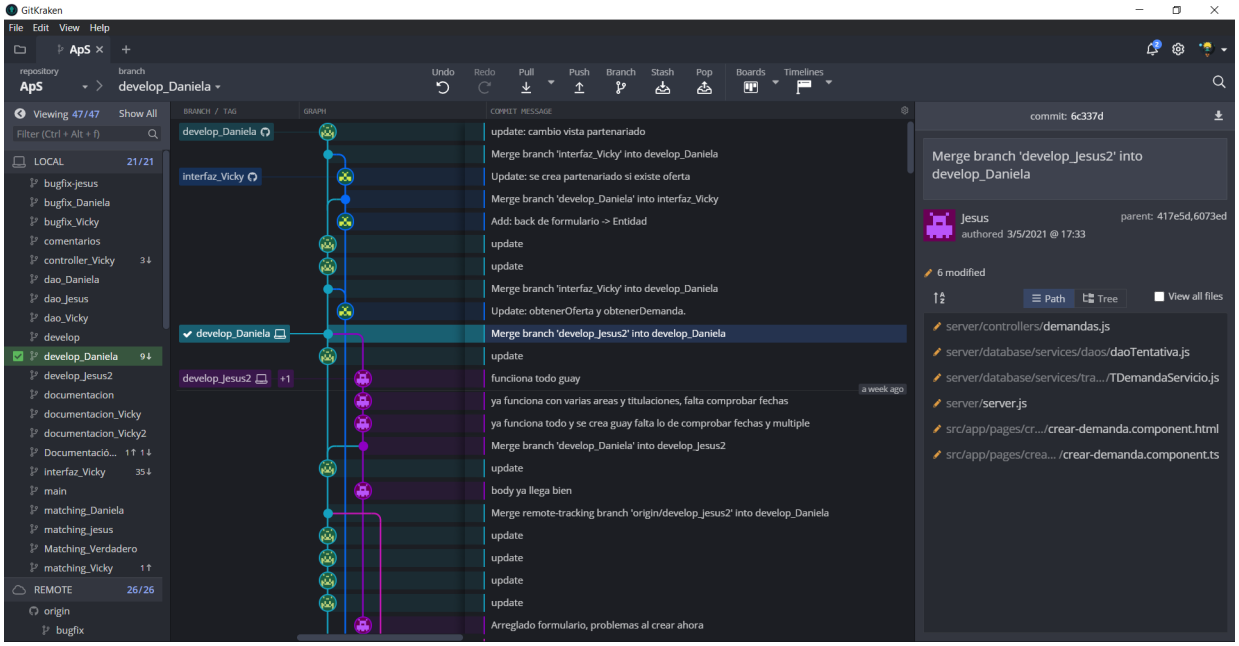

Figura 1: GitKraken

control de versiones tiene una representación gráfica muy intuitiva que permite ver la distribución de las ramas, los commits y su evolución, y además permite resolver los conflictos generados al hacer merge 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.

- 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...) 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. Esta tecnología fue sugerida por Victoria y nos ha facilitado mucho tanto la organización como el seguimiento de nuestros avances.

- LaTex: Es un lenguaje de maquetado 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.

- MySQL: Aunque nuestro proyecto continúa el trabajo realizado por David Jiménez del Rey y ya contaba con un sistema gestor de bases de datos, dicho sistema era MongoDB y 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.

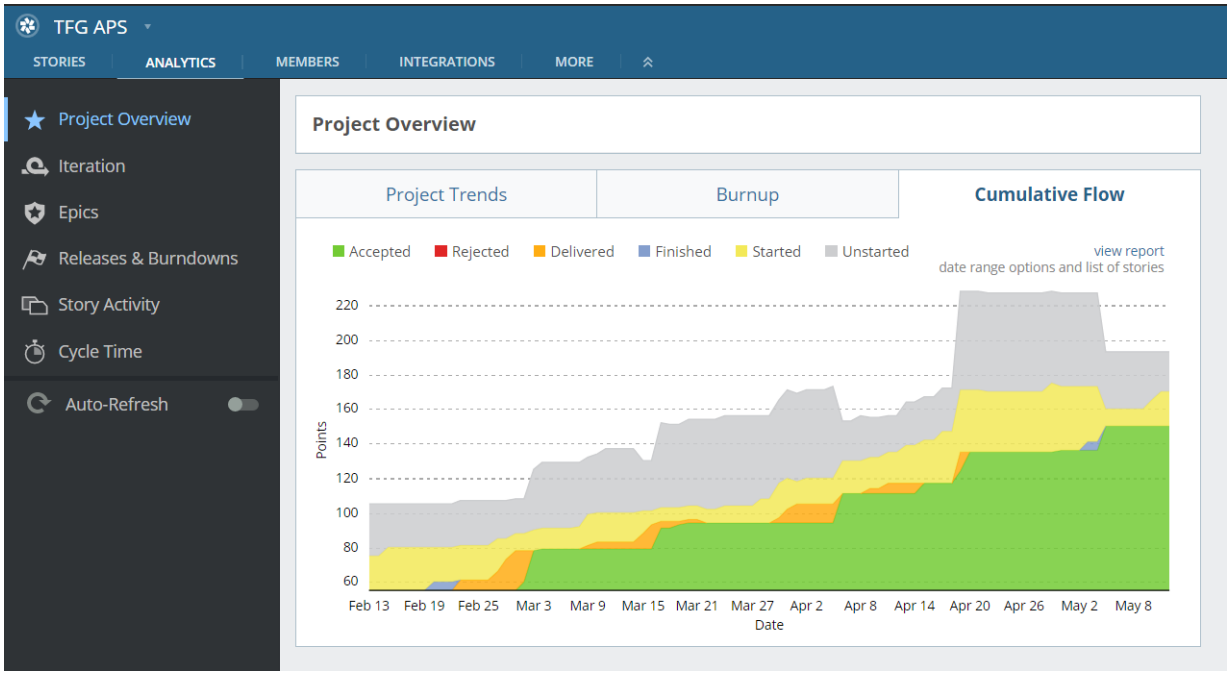

Figura 2: GitKraken

- 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.
- MySQL Workbench: Es una herramienta para diseño, desarrollo y administración de bases de datos relacionales. Cuenta con funcionalidades de validación de esquemas y modelos 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-relacion
- Diagrams.net: Es una herramienta de diseño de diagramas de varios tipos entre los cuales se encuentran diagramas de clases, de flujos, de entidades... Es una herramienta muy poderosa pero dado que ningún componente del grupo tenía experiencia previa con ella y para lo único que se necesitaba era para hacer el diagrama de entidad-relación, se descartó el uso de esta aplicación en pos de otra más simple.

3. Modelos (nombre provisional)

3.1. Introducción

Al empezar con el proyecto, junto con nuestros tutores, nos hemos dado cuenta de que la aplicación necesitaba ser definida de una forma más profesional y robusta. David entendió que los profesores y las entidades definían los proyectos de la misma forma, pero esto no es así.

Las entidades no conocen todos los detalles del problema en cuestión que quieren resolver, porque no suelen dedicarle el suficiente tiempo a la especificación.

Los profesores por su parte tienen una idea muy vaga 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.

David creó un elemento llamado Iniciativa que almacenaba algunas de las características generales que comparten las dos propuestas y derivaba en un formulario que se les ofrecía a las entidades y a los profesores. Pero debido a que las entidades y los profesores no plantean los problemas de la misma manera, no es apropiado presentarles el mismo formulario.

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 del aplicativo, 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 que 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.

Como se ha rediseñado la base de datos, no solo porque ha cambiado su tipo, que ahora es relacional, si no también porque los conceptos no estaban claros en el anterior TFG. 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.

3.2. Modelo de dominio

El modelo de dominio es un mapa conceptual del aplicativo que permite a cualquier individuo entender su funcionamiento.

Si empezamos mirando nuestro modelo desde arriba podemos observar que hay una clase padre llamada usuario y de ella se ramifican otras 5 clases que representan los 5 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 UNED 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.
En promotor externo representaríamos al profesor externo, que es aquel profesor que no forma parte de la UNED 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.

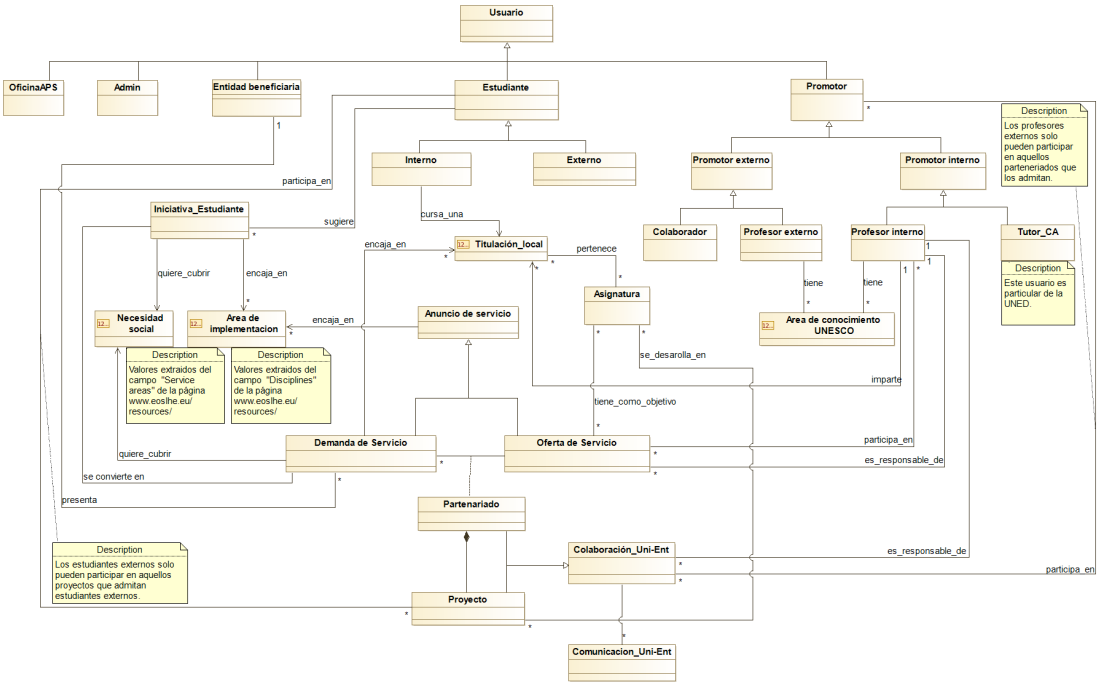

Figura 3: Modelo de dominio

El promotor interno puede ser un profesor o un tutor de la UNED. Este profesor es el que más cargos de responsabilidad tiene, él es el que puede crear las ofertas de servicio y es el responsable de los partenariados y los proyectos.

Todo proyecto tiene que empezar siendo un partenariado y este partenariado se origina en la unión de una oferta creada por un profesor, y una demanda creada por una entidad. La creación de la oferta y la demanda se realiza mediante unos formularios en la web.
Una demanda se puede llegar a crear gracias a un estudiante, este previamente ha tenido que crear una iniciativa. Una iniciativa es una propuesta de proyecto de un estudiante, la cual es acogida posteriormente por una entidad. Cuando la entidad 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-Ent. Queremos destacar, que los atributos de Necesidad social y Area de implementación, tan importantes para caza de 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, ademas el proyecto tendrá que admitir estudiantes de la misma universidad que el profesor para que puedan ser evaluados por este.

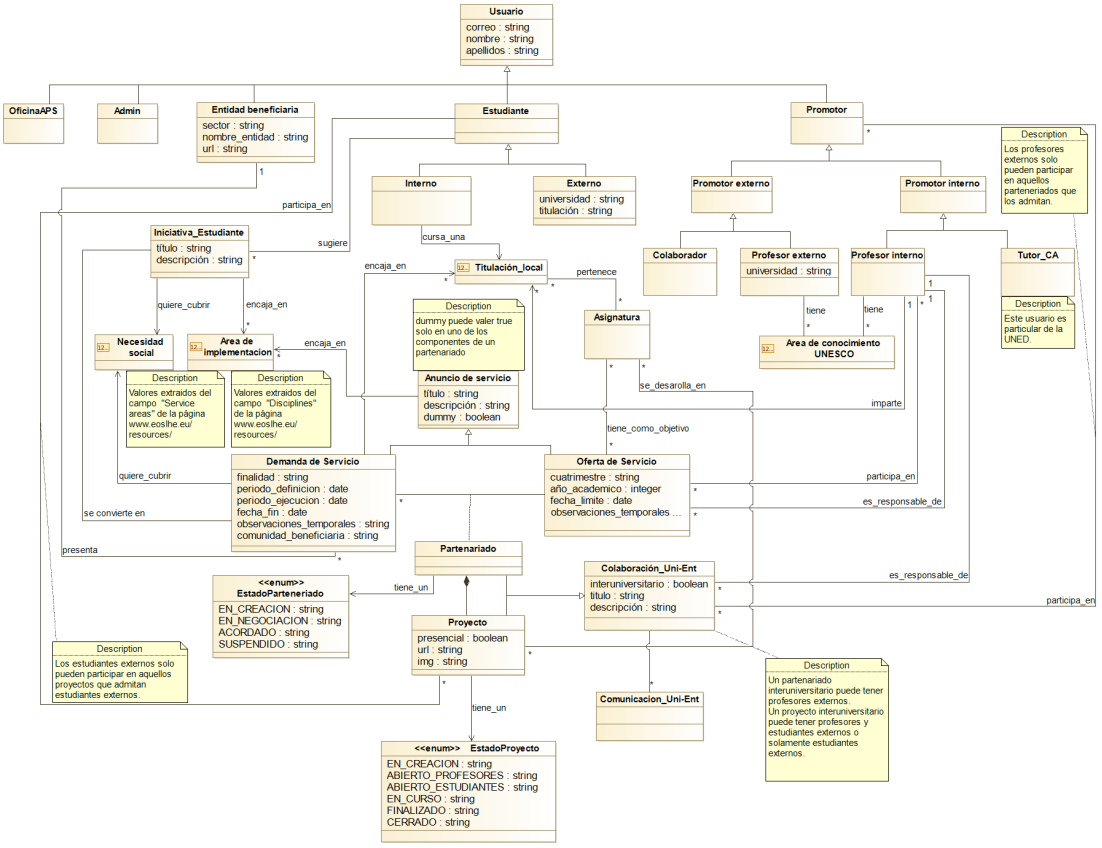

Figura 4: Modelo de datos

3.3. 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 partenariado toma el estado de EN_CREACIÓN cuando el profesor se pone en contacto con la entidad, si la entidad contesta y acepta aliarse con el profesor el estado del partenariado pasará a EN_NEGOCIACION. Cuando el profesor y la entidad terminan de establecer los términos y condiciones del partenariado y ambos están de acuerdo el partenariado pasá 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.

En el caso del proyecto tenemos los estados EN_CREACION, ABIERTO_PROFESORES, ABIERTO_ESTUDIANTES, EN_CURSO, FINALIZADO y CERRADO. Cuando el profesor decide seguir con el proyecto debe rellenar un formulario, al rellenar dicho formulario el estado del proyecto pasa a estar EN_CREACION. La entidad recibe dicha solicitud de continuación y debe rellenar otro formulario, una vez rellenado y enviado el proyecto pasa al estado de ABIERTO_PROFESORES. Una vez que el proyecto ha sido definido 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.

3.4. 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 4 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 la oferta de servicio, la demanda de servicio y la iniciativa. La sección de Colaboración contiene la información relacionada con el partenariado y el proyecto. Y la última es la sección que tiene menos complejidad, la sección de Comunicación que contiene las tablas de *mail*, mensaje, *upload* y *newsletter*, estas creadas por David que se han mantenido intactas.

3.4.1. Usuarios

Esta sección es la más compleja debido a la separación que hay que realizar de los usuarios externos e internos.
Un usuario interno es aquel profesor, tutor, colaborador, estudiante 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 estos tienen parte de los

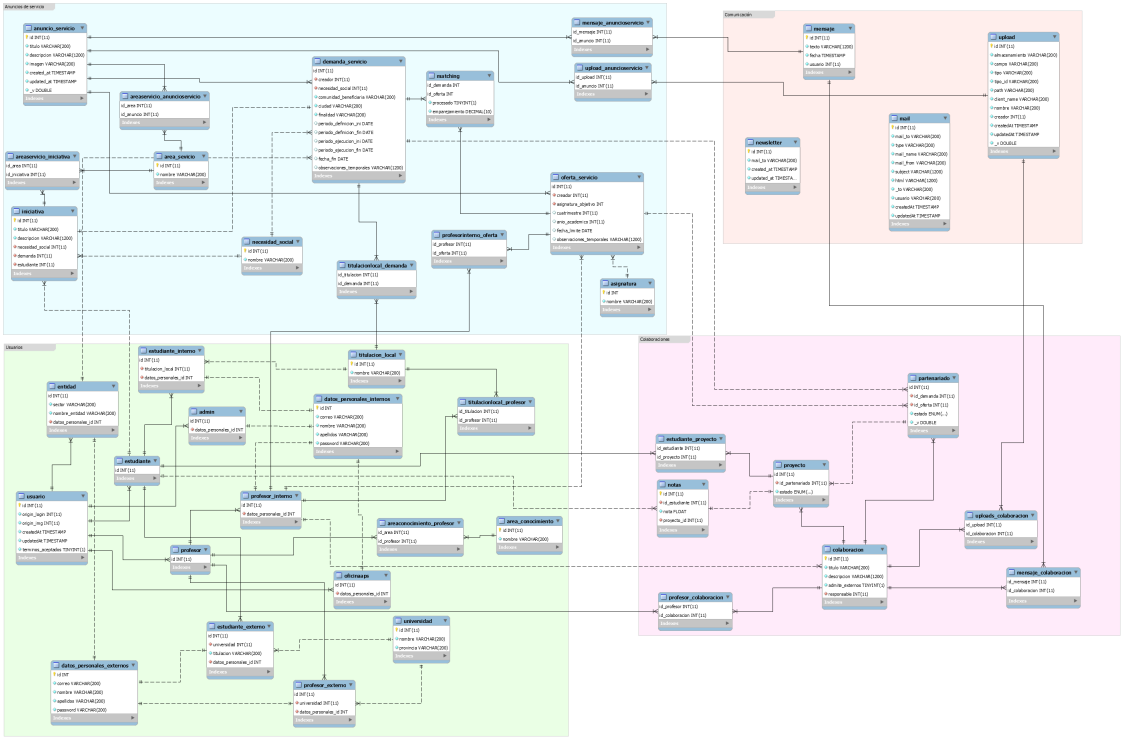

Figura 5: Diagrama de entidad relación

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 tutores 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 de entidad, 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.
Después podemos observar tablas secundarias que representan características de los usuarios como es, 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.

3.4.2. Anuncios de servicio

En este conjunto de tablas podemos encontrar las pertenecientes a la demanda de servicio, la oferta de servicio y la iniciativa.
La iniciativa es una propuesta de proyecto realizada por un estudiante. Esta propuesta

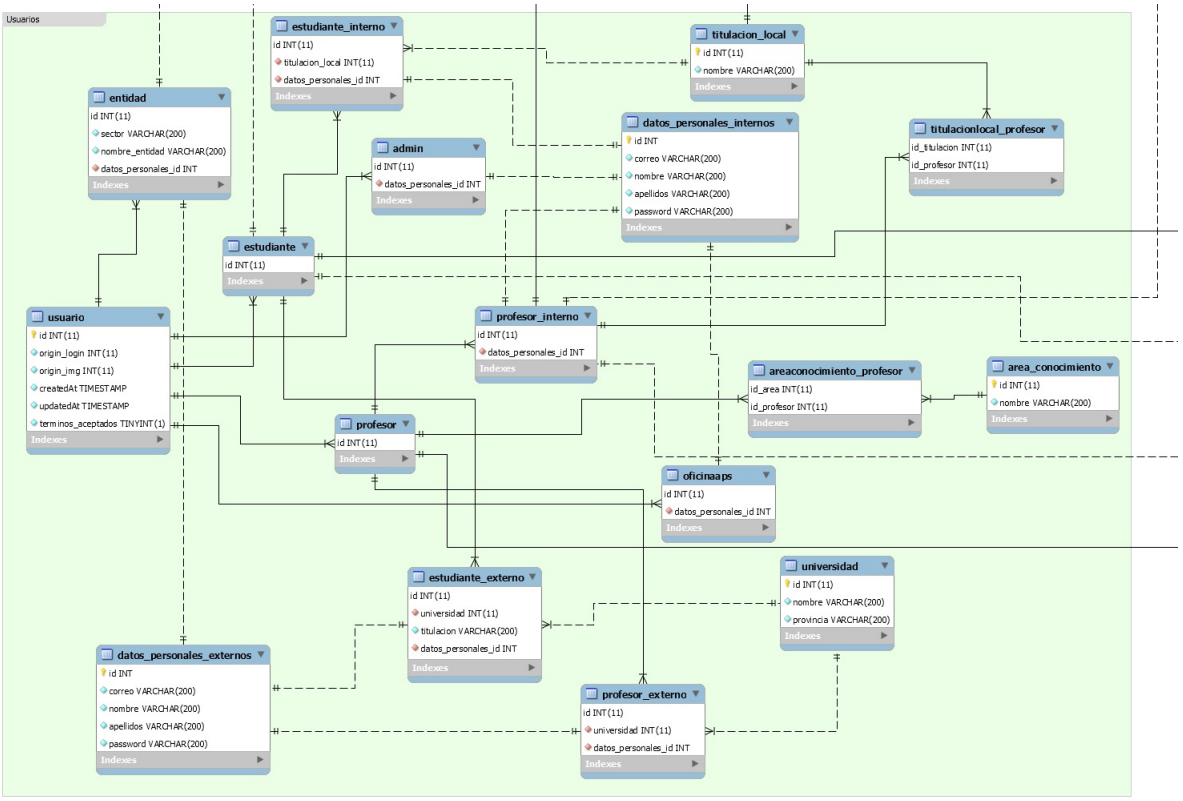

Figura 6: Diagrama de entidad relación: Usuarios

debe ser validada por un administrador y posteriormente adoptada por una entidad que desee realizar el proyecto.

La demanda de servicio es creada por una entidad y define una necesidad especifica que quiere cubrir. Esta necesidad social es representada por un enumerado alojado en la tabla de necesidad_social. Estos enumerados han sido obtenidos del campo Service areas hallado en la página www.eoslhe.eu/resources/ . Los enumerados alojados en la tabla area_servicio también se obtuvieron de esta página, del campo llamado Disciplines.

La oferta de servicio es creada por un profesor interno y tiene menos detalles que la demanda porque es 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.

3.4.3. Colaboración

El partenariado es el segundo paso en la creación de un proyecto, esta tabla contiene las *ids* de la demanda y la oferta que la componen.
El proyecto no puede existir sin un partenariado y es por eso por lo que tiene un id del partenariado que lo creó. El proyecto posee estudiantes y es por eso por lo que tiene una

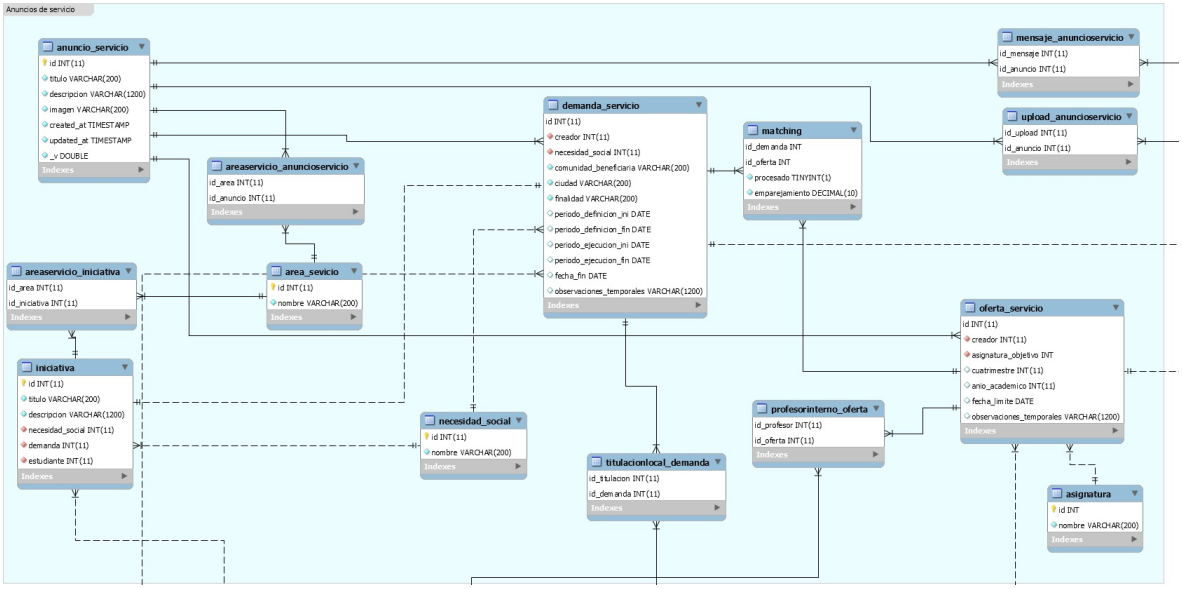

Figura 7: Diagrama de entidad relación: Anuncios de servicio

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*.

3.4.4. Comunicación

Esta parte se ha mantenido tal y como la había creado David, solo se ha adaptado al nuevo sistema.
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 nada 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.

4. DAO

Tras cambiar la base de datos de mongo por una de SQL, también era necesario hacer la lógica de accesos a la base de datos, por lo que se crearon 4 *Data Access Object* (DAO a partir de ahora) que se encargarían de las operaciones de cada una de las 4 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

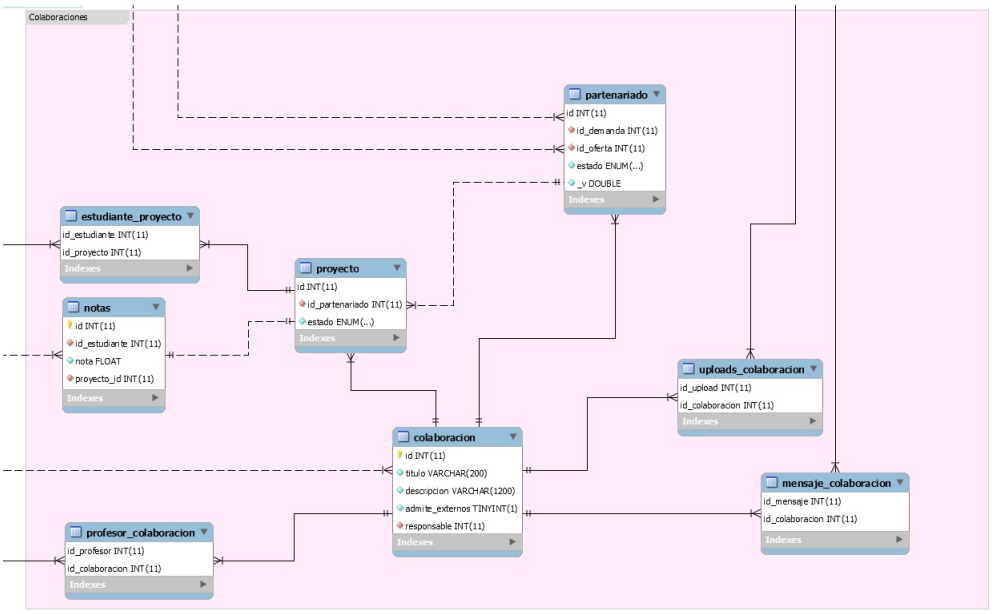

Figura 8: Diagrama de entidad relación: Colaboración

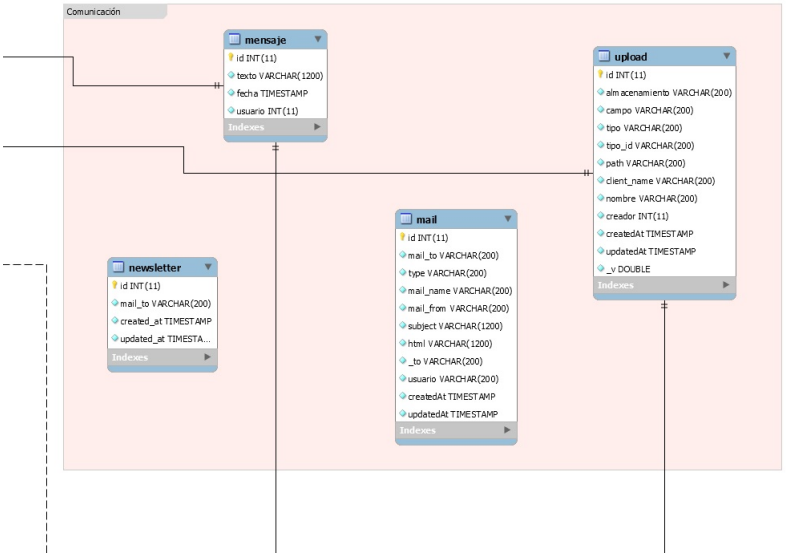

Figura 9: Diagrama de entidad relación: Comunicación

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 más avanzada su 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-relacion son:

- DAOColaboracion: se encarga de manejar toda la información relacionada con los proyectos y los partenariados, desde sus participantes, ya sean profesores o alumnos hasta los mensajes y archivos asociados a estos proyectos o partenariados. Este DAO se llama así porque tiene con 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: trata 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 alumno 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, 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, entidad 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, TUsuario, TProfesor, TOficinaAPS, TEstudiante, TProfesorExterno, TProfesorInterno, TEstudianteInterno y TEstudianteExterno

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 que 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

Referencias