UNIVERSIDAD DE GRANADA

TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

SmartU

Desarrollo de un espacio colaborativo de ideas y proyectos

Autor
Juan José Jiménez García

Tutor
Miguel Gea Megías

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN

—

Granada, 9 de septiembre de 2017

SmartU: Desarrollo de un espacio colaborativo de ideas y proyectos

Juan José Jiménez García

Palabras clave: *multidisciplinar, colaborativo, desarrollo web, redes sociales, software de código abierto*

Resumen

Este documento detalla mi trabajo y participación en el proyecto multidisciplinar llevado a cabo en la Universidad de Granada.

SmartU nace del trabajo en equipo realizado por un grupo de estudiantes y sus tutores, como respuesta a un problema habitual de la vida universitaria, que es el fomento de proyectos de carácter multidisciplinar.

Lo que se pretende con este trabajo es crear una plataforma basada en un formato de **red social** que permita a los estudiantes publicar proyectos e ideas y con ello encontrar a otros estudiantes de diversas disciplinas que quieran unirse para llevar a cabo la idea.

SmartU pretende ser un apoyo y una forma de fomentar más el **trabajo en equipo**, algo que la sociedad actual demanda mucho en sus puestos de trabajo y que no termina de fraguar del todo en las universidades.

Como resultado de este trabajo, también se obtienen una serie de **resultados y conclusiones** sobre cómo ha sido este primer "proceso piloto" de equipo multidisciplinar, que servirá de ayuda para mejorar en el futuro los problemas encontrados.

SmartU: Development of a collaborative space of ideas and projects

Juan José Jiménez García

Keywords: *multidisciplinary, coworking, web development, social networks, open source software*

Abstract

This document explains my work and collaboration on the multidisciplinary project made on the University of Granada.

SmartU is the result of the teamwork carried out by a group of students and their tutors in response to a common problem in the university life, which is the promotion of multidisciplinary projects.

This work aims to create a **social network based** web platform which allows students to publish projects and ideas in order to find another students from other specialties who want to join to carry it out.

SmartU is meant to be a support to promote **work in team**, since it's something highly required by today's jobs but it's not very popular in universities.

As a result from this project, we also got a bunch of **results and conclusions** about how well this first multidisciplinary project was, which will be used to improve in the future the found problems.

Yo, **Juan José Jiménez García**, alumno de la titulación **Grado en Ingeniería Informática** de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación** de la **Universidad de Granada**, con DNI 76655977J, autorizo la ubicación de la siguiente copia de mi Trabajo de Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Juan José Jiménez García

Granada, a 9 de septiembre de 2017

D. **Miguel Gea Megías**, profesor del **Departamento de Lenguajes y Sistemas Informáticos** de la **Universidad de Granada**.

Informa:

Que el presente trabajo, titulado ***SmartU: Desarrollo de un espacio colaborativo de ideas y proyectos***, ha sido realizado bajo su supervisión por **Juan José Jiménez García**, y autoriza la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expide y firma el presente informe en Granada, a 9 de septiembre de 2017.

El tutor:

Miguel Gea Megías

Agradecimientos

Quisiera agradecer a mi familia todo el apoyo que me han brindado no solo durante mi etapa en la universidad, sino también en el colegio y el instituto. Gracias por toda vuestra ayuda y cariño.

Gracias a mis amigos y amigas de la ETSIIT de Granada, los mejores compañeros de trabajo que he conocido y con quienes he pasado momentos geniales.

A mis profesores, tanto a los buenos como a los “no tan buenos”, les agradezco que me hayan ayudado a ser quien soy hoy con su atención y esfuerzo por querer transmitir sus valiosos conocimientos.

Y a mis compañeros del TFG, con quienes he compartido estos últimos meses dando forma a este proyecto, muchas gracias. Especialmente a Emilio, y a mi tutor Miguel.

Índice general

Capítulo 1

Proceso de desarrollo

Este capítulo abarca el desarrollo del producto en si, es decir, la aplicación web de SmartU para la publicación de proyectos y difusión de opiniones e ideas. Siguiendo una estructura similar a la del proceso tradicional de desarrollo del software, el capítulo se divide en **cuatro apartados**, cada uno de ellos dedicado a una de las fases de dicho proceso.

Comenzaremos por una introducción para explicar la forma de trabajo que se ha seguido, y a continuación pasamos al apartado de análisis, en el que definimos qué es lo queremos que sea exactamente SmartU, pasando por el diseño de la aplicación web habiendo conocido ya los actores del sistema y sus requisitos.

Finalmente pasamos a la implementación, donde se detalla de forma más técnica cómo se ha creado este sistema de información web, y terminamos hablando de la fase de pruebas. Se incluye un apartado adicional de herramientas utilizadas para el desarrollo.

1.1. Introducción

Al comenzar el desarrollo del proyecto, se pudo apreciar que había una alta posibilidad de sufrir cambios a lo largo del proceso debido a que el alcance del mismo no estaba muy delimitado. Es por ello que se optó por seguir una metodología principalmente ágil, la cual permitía mayor flexibilidad y facilidad para adaptar posibles cambios.

En el desarrollo ágil es importante la comunicación entre todos los miembros del equipo y que éstos, de alguna forma, participen activamente durante el desarrollo. No todos los miembros de este equipo multidisciplinar trabajan en el ámbito de la Ingeniería Informática, pero sabíamos que podían colaborar en las primeras fases aportando ideas y posibles requisitos.

Sin embargo, sí que era importante que hubiera comunicación y consenso entre los miembros del área de Informática. Con mi compañero Emilio se realizó una puesta en común para intentar unificar las plataformas que cada uno teníamos que desarrollar.

Siguiendo algunos de los principios del desarrollo ágil, se ha evitado realizar documentación innecesaria, y se ha optado por ir a lo primordial y esencial, sin que ello repercuta en la calidad del software. Hay que tener en cuenta que este desarrollo pretende ser continuado y mejorado en el tiempo, por lo que una documentación base correcta es más importante que rellenar páginas y páginas innecesarias.

Por último, añadiremos que se ha preferido optar por establecer una lista de requisitos del sistema en lugar de emplear la técnica de las **historias de usuario**, ya que no son de obligado cumplimiento el desarrollarlas, y se ha de tener en cuenta el hecho de que el desarrollo será continuado por otras personas en el futuro.

1.2. Análisis

La fase de análisis se distribuye en **dos grandes bloques**:

- El primer bloque comprende el análisis en equipo del problema y la extracción de unos primeros requisitos y usuarios base sobre los que basaremos nuestras decisiones futuras a la hora de desarrollar el sistema. Aquí se realizarán diversas técnicas de generación de ideas y prototipado para acotar y definir mejor el sistema.
- El segundo bloque es más habitual del proceso de desarrollo de software, ya que se va a dedicar a recopilar y mostrar los requisitos del sistema ya definidos correctamente, tras haber terminado el análisis del primer bloque.

1.2.1. Generación de ideas y extracción de requisitos

El equipo de trabajo multidisciplinar se reunió varias veces para realizar sesiones de generación de ideas, tales como el *Brainstorming*, que llevó a cabo mi compañero Emilio, o el proceso creativo de *Design Thinking* llevado a cabo por mi compañero Juan y por mí.

Las reuniones tuvieron lugar en diferentes puntos de encuentro de todos los campus de la Universidad de Granada, siendo el más habitual el de

la Facultad de Ciencias Económicas y Empresariales, debido a la disponibilidad de mejores salas de reunión y trabajo en equipo.

Más adelante se obtuvo más información gracias a la presentación del proyecto multidisciplinar que se hizo en la Facultad de Ciencias de la Actividad Física y del Deporte, realizada por los compañeros Javier y Juan. Allí se pudo enseñar el proyecto a potenciales usuarios (*stakeholders*) y recibimos opiniones muy interesantes de posibles características que les gustaría ver para que fuese una aplicación web más completa y funcional.

Usuarios del sistema

Tras las reuniones de generación de ideas, se llegó a la siguiente conclusión respecto a **quiénes son los usuarios potenciales del sistema**. Es importante poder definir a los posibles futuros usuarios de nuestra aplicación para así poder enfocar mejor los requisitos y el diseño de nuestra aplicación web. Nuestro equipo en si consta de algunos de ellos: son los estudiantes y profesores, o en general, la comunidad educativa.

- Los **estudiantes** son potenciales usuarios registrados de nuestra aplicación web. Podemos concretar más e irnos al segmento de estudiantes de último año de grado o máster que necesitan una idea para realizar un proyecto de final de grado/máster, pero también están aquellos estudiantes que cuentan con una idea para un proyecto y necesitan a otros estudiantes con los que conformar un equipo.
  Este grupo de usuarios es el que se prevee que sería el que utilizase el sistema con mayor frecuencia.

- Otro segmento de usuarios es los **habitantes de la ciudad**. La diferencia radica en que por lo general, es un colectivo no asociado a la universidad y desconocedor de lo que ocurre dentro de ésta. Por ello, no son considerados potenciales usuarios que se van a registrar en el sistema, pero si entrarán y consultarán las ideas que se han publicado.
  Pueden ser participantes activos de un proyecto que se esté creando si éste es un proyecto que le interese o afecte a su entorno (barrio, movilidad, economía, etc).

- Por último, tenemos el grupo de **usuarios empresarios**. Estos son aquellas personas que, o bien tienen una idea, o encuentran interesante una idea que han visto publicada en la aplicación. Pueden ponerse en contacto con un equipo de un proyecto y ayudarles con financiación o promoción de su idea, entre otras posibilidades.

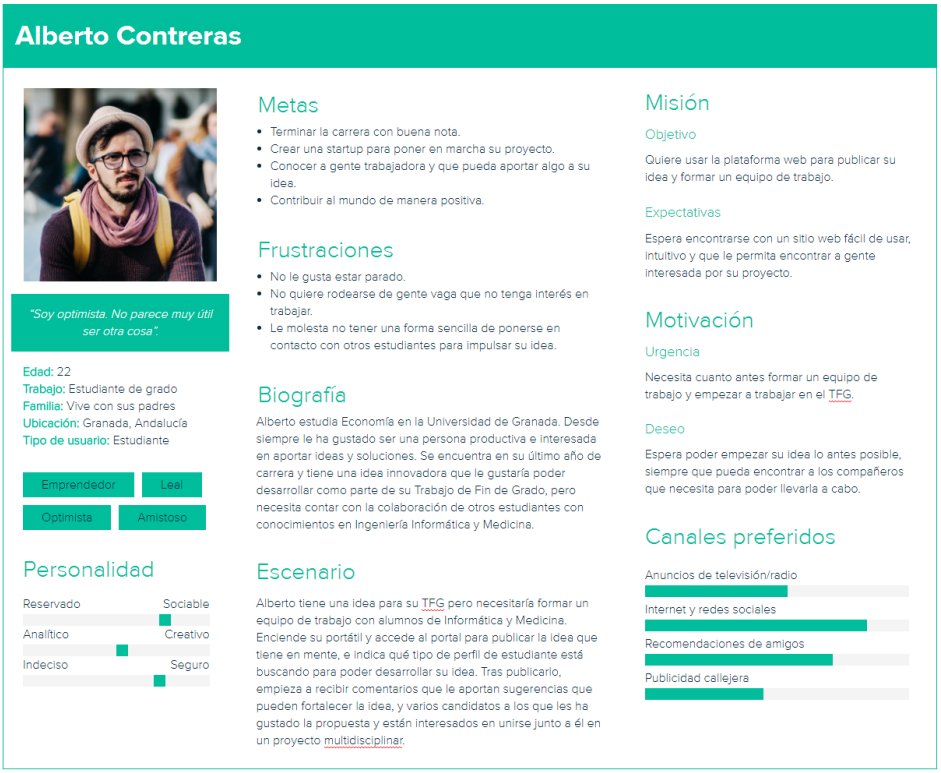

Figura 1.1: *User Persona* de un estudiante

Para representar de una forma más visual a estos usuarios, se han creado una serie de bocetos de usuario (o *User Persona*), que esquematizan y ayudan a definir de un vistazo el tipo de perfil de usuario que hemos definido. Podemos verlos en las figuras 1.1, 1.2 y 1.3.

Como podemos ver, el conjunto de potenciales usuarios intenta abarcar a todo el conjunto de la población, por lo que una buena implantación del mismo podría **reportar enormes beneficios** para todos ellos, ya sea encontrando un proyecto en el que trabajar, como llevar a cabo una idea que mejore la vida en general de los ciudadanos.

1.2.2. Requisitos del sistema

En base a todas las reuniones e información recabada de las mismas y de posibles stakeholders, se ha confeccionado la siguiente lista de requisitos del sistema.

Figura 1.2: *User Persona* de un ciudadano

Figura 1.3: *User Persona* de un empresario

Requisitos de datos

Se quiere implementar un sistema de información basado en web que sea capaz de almacenar la información relativa a proyectos para la universidad, permitiendo la asignación de información más detallada como área de conocimiento del proyecto, descripción y avances del mismo, así como la información perteneciente a los usuarios que se registren en la plataforma, sus comentarios e información de perfil.

RD1. Proyecto: Un proyecto necesita almacenar la siguiente información:

Nombre: Cadena de caracteres.

Descripción: Cadena de caracteres.

Página web: Cadena de caracteres.

Localización: Cadena de caracteres.

Coordenadas: Cadena de caracteres.

Fecha de creación: Fecha de alta del proyecto en el sistema.

Fecha de eliminación: Fecha estimada de finalización del proyecto.

Fecha de actualización: Fecha de última actualización del proyecto.

Propietario: Identificador del usuario registrado que creó el proyecto.

RD2. Redes sociales: Los proyectos y los usuarios pueden definir sus propias redes sociales para darse a conocer, y necesitan la siguiente información:

Nombre: Cadena de caracteres.

URL: Cadena de caracteres.

ID de usuario: Identificador del usuario.

ID del proyecto: Identificador del proyecto.

Una red social petenece a un usuario o a un proyecto, se han unificado para evitar redundancia.

RD3. Área del proyecto: Los proyectos tienen una o varias áreas, y se necesita la siguiente información para poder guardarlas.

Nombre: Cadena de caracteres.

Descripción: Cadena de caracteres.

RD4. Buena Idea: Una buena idea se asemeja a un “Me gusta” de redes como Facebook o Twitter. Necesita la siguiente información:

ID del proyecto: Identificador del proyecto.

ID del usuario: Identificador del usuario.

RD5. Especialidad: La especialidad es algo a lo que un usuario se dedica o que tiene experiencia en ello. Sirve para encontrar a gente de un determinado campo que necesita un proyecto. Se necesita la siguiente información:

Nombre: Cadena de caracteres.

Descripción: Cadena de caracteres.

RD6. Experiencia en especialidad: Partiendo del requisito de dato anterior, un usuario cuenta con un cierto nivel de experiencia en una especialidad, lo cual requiere almacenar la siguiente información:

Experiencia: Cadena de caracteres.

ID de la especialidad: Identificador de la especialidad.

ID del usuario: Identificador del usuario.

RD7. Comentario: Los usuarios registrados pueden dejar comentarios en sus proyectos o en los de otros usuarios. Se necesita la siguiente información:

Contenido: Cadena de caracteres.

Fecha de creación: Fecha insertada cuando se creó el comentario.

ID del usuario: Identificador del usuario que hace el comentario.

ID del proyecto: Identificador del proyecto donde se comenta.

RD8. Avance: Un proyecto puede tener un conjunto de avances que muestre a los usuarios el progreso que se está logrando con el proyecto. Necesita la siguiente información:

Nombre: Cadena de caracteres.

Descripción: Cadena de caracteres.

Fecha de creación: Fecha de publicación del avance

Imagen destacada: Cadena de caracteres del nombre de un archivo de imagen.

ID del proyecto: Identificador del proyecto al que va asociado.

RD9. Vacante: Una vacante representa una disponibilidad de un puesto, en una determinada disciplina, como integrante en un proyecto. Se necesita la siguiente información:

Experiencia: Una cadena de caracteres.

Especialidad: Un identificador de una especialidad concreta.

RD10. Hashtag: Uno o varios proyectos pueden tener etiquetas conocidas como *hastags*, y se necesita guardar para ello:

Nombre: Una cadena de caracteres.

RD11. Usuario: Un usuario necesita almacenar la siguiente información:

Nombre: Cadena de caracteres.

Apellidos: Cadena de caracteres.

Email: Cadena de caracteres.

Contraseña: Cadena de caracteres.

Biografía: Cadena de caracteres.

Página web: Cadena de caracteres.

Localización: Cadena de caracteres.

Puntos: Número entero.

CIF: Cadena de caracteres.

Admin: Booleano.

Verificado: Booleano.

Avatar: Cadena de caracteres del nombre de un archivo de imagen.

RD12. Solicitud colaboración: Los usuarios pueden solicitar colaborar en un proyecto que tenga vacantes disponibles. Se necesita almacenar la siguiente información:

Fecha: Fecha de creación de la solicitud.

Descripción: Una cadena de caracteres.

Usuario solicitante: Identificador del usuario que solicita colaborar.

Proyecto solicitado: Identificador del proyecto en el que se desea colaborar.

RD13. Colaborador: Un usuario puede ser colaborador de un proyecto en una determinada especialidad. Se necesita almacenar:

ID del usuario: Identificador del usuario colaborador.

ID del proyecto: Identificador del proyecto donde colabora.

ID de especialidad: Identificador de la especialidad del colaborador.

RD14. Intereses: Los usuarios pueden marcar intereses en determinadas áreas. Para ello se necesita guardar:

ID del usuario: Identificador del usuario que marca un interés.

ID del área: Identificador del área donde el usuario marca un interés.

RD15. Status: Un ususario puede tener un status en base a su participación en la aplicación web, por ejemplo publicando proyectos o bien dando "Buena Idea" a otros. Se requiere la siguiente información:

Nombre: Cadena de caracteres. Es el nombre que recibe el status.

Puntos: Número entero que representa los puntos que se han de alcanzar para llegar a dicho status.

RD16. Seguidor: Un usuario puede seguir a otro para estar al tanto de sus proyectos y de lo que hace en la aplicación web. Se necesita almacenar lo siguiente:

ID del usuario seguido: Identificador del usuario al que se sigue.

ID del usuario seguidor: Identificador del usuario que está siguiendo.

Restricciones semánticas

RS1. Un usuario no registrado no puede participar en la aplicación web salvo para consultar la información.

RS2. Un usuario no puede tener el mismo email o nombre de usuario que otro registrado anteriormente.

RS3. Un proyecto solo puede tener un usuario propietario.

RS4. Los usuarios solo pueden modificar o eliminar proyectos de los que son propietarios.

RS5. Un usuario puede solicitar unirse al proyecto si hay vacantes y si no es ya propietario o colaborador.

RS6. Un usuario no puede seguirse a sí mismo.

RS7. Los usuarios con correo corporativo reconocido serán registrados automáticamente como usuarios "verificados".

Requisitos funcionales

Requisitos funcionales de inserción

RF1.

RF2.

RF3.

RF4.

RF5.

RF6.

RF7.

RF8.

RF9.

RF10.

Requisitos funcionales de consulta

RF1.

RF2.

RF3.

RF4.

RF5.

RF6.

RF7.

RF8.

RF9.

RF10.

Requisitos funcionales de eliminación

RF1.

RF2.

RF3.

RF4.

RF5.

RF6.

RF7.

RF8.

RF9.

RF10.

Requisitos no funcionales

RNF1. La aplicación web debe estar finalizada para septiembre de 2017.

RNF2. La aplicación web debe tener un diseño adaptable a todo tipo de tamaños de pantalla de dispositivo.

RNF3. La aplicación web debe funcionar correctamente en la mayoría de navegadores de Internet más utilizados y en sus últimas versiones disponibles.

RNF4. El código de la aplicación debe estar correctamente comentado para facilitar la tarea de mejorarlo a los futuros desarrolladores que continúen el proyecto.

1.3. Diseño

1.4. Implementación

Tras el proceso de análisis de nuestro sistema y extracción de los requisitos que necesitaría nuestra aplicación web (*expuestos en anteriores apartados de este capítulo*), en este voy a detallar los principales aspectos de la implementación de la misma, incidiendo en aspectos importantes e inherentes al proceso de desarrollo de un sistema de información web.

La codificación de la aplicación web de SmartU venía condicionada por la necesidad de un desarrollo que fuese agilizado e iterativo, intentando completar funcionalidades específicas en cada *"paso"* que se diese a lo largo de este proceso. Esto me llevó a tomar una serie de decisiones respecto al lenguage de programación que utilizaría, así como el conjunto de herramientas que me ayudarían con esta tarea, pero siempre teniendo como máxima un desarrollo correcto y bien realizado.

1.4.1. Laravel

¿Por qué utilizar un *"framework"* de desarrollo web como Laravel? La elección del mismo no fue fácil, y llevó algo de tiempo debido al análisis de pros y contras que podría encontrarme a lo largo del proceso de desarrollo. Existen numerosos lenguages de programación como Python, JavaScript, Java, etc, todos con diferentes soluciones de desarrollo web. En el caso de Laravel, se basa en el lenguage PHP, que es ya bien conocido en el mundo de la programación web, y a lo largo de los años ha ido mejorando su potencial con nuevas versiones que incorporaban características que si estaban

presentes en otros lenguages.

Laravel es un framework de desarrollo web basado en PHP creado por Taylor Otwell en el año 2011. Fue creado con el intento de proporcionar una alternativa más avanzada a **CodeIgniter**. En muchos lugares de Internet, Laravel es conocido por darle un muy necesitado lavado de cara a PHP, y hacerlo de nuevo más atractivo a la vista de los desarrolladores.

Laravel agiliza los procesos en casi todos los apartados existentes del desarrollo de una aplicación web. Proporciona un conjunto de herramientas para gestionar la autenticación de usuarios, bases de datos rutas, entre otros elementos que a continuación se detallan:

- Está basado en el conocido patrón de diseño MVC **(Modelo-Vista-Controlador)**, lo cual permite separar la lógica de nuestra aplicación de la interfaz de usuario, usando como intermediario un controlador que mueva los datos entre modelos y vistas.

- Cuenta con soporte para **Composer**, lo cual nos permite gestionar de forma sencilla las dependencias de nuestro proyecto, por ejemplo para realizar el despliegue.

- Al estar basado en PHP, contamos con el extenso conjunto de funciones que ya nos proporciona el lenguage. Para los programadores actuales de PHP, empezar a usar Laravel **no supone un mayor esfuerzo**, ya que usa todos los elementos y sintaxis del mismo.

- Cuenta con un motor de plantillas, **Blade**, que permite construir la interfaz web de forma dinámica, recibiendo datos como parámetros y utilizarlos para mostrar la información al usuario.

- Laravel dispone de una **extensa comunidad** de desarrolladores que proporcionan consejos y soluciones a los diversos problemas que puedan surgir a la hora de desarrollar con este framework. En este sentido es fácil encontrar la respuesta a cualquier duda, lo cual hace que el aprendizaje sea mucho más sencillo.

La razón por la que elegí este framework era por la sencillez con la que resolvía muchos de los aspectos técnicos que esta aplicación web necesita, además de que se basa en un lenguage de programación que conozco. Siempre es importante estar al tanto de las novedades en los lenguages de programación, pero cuando el tiempo es una restricción muy importante, se hace necesario usar soluciones basadas en algo que ya se conoce.

1.4.2. Estructura del proyecto

El proyecto se divide en una serie de carpetas generadas automáticamente al crear un nuevo proyecto de Laravel. El código logra una mayor legibilidad y mantenibilidad al separar en diferentes directorios la arquitectura de nuestra aplicación web. En la figura 1.4 podemos ver la estructura completa de ficheros y carpetas, destacando las siguientes:

App En esta carpeta encontramos toda la **lógica** de nuestra aplicación. Alberga todos los controladores y modelos de datos, así como otros conjuntos de clases que el framework utiliza para poder funcionar. También podemos destacar otros elementos como los ***Requests*** o los ***Middlewares***, pero hablaré de ellos más adelante.

Config La carpeta de configuración de nuestro proyecto. En ella se encuentran muchas variables que permiten ajustar la base de datos que vamos a usar, el idioma por defecto de la aplicación, la zona horaria, entre otros muchos aspectos.

Database En esta carpeta alojamos todo lo relativo a la base de datos. Principalmente tenemos las **migraciones** y los ***seeders***, y permiten establecer las tablas y columnas que necesitaremos. Hablaremos también de estos aspectos más adelante.

Public La carpeta public es el **punto de inicio** de nuestra aplicación web. Es el único sitio que es visible para los usuarios, o dicho de otra manera, la única carpeta que el servidor web puede "ver".

Resources Esta carpeta contiene todo lo relacionado con la vista. Incluye carpetas para las diferentes **plantillas** de la aplicación, ficheros de **lenguage**, así como archivos de **hojas de estilo** y scripts. Más adelante hablaremos de este punto.

Routes Esta carpeta contiene las rutas de nuestra aplicación. Permite definir URLs que el usuario puede introducir en su navegador, y cada una dará lugar a una acción diferente, como puede ser mostrar la página de inicio o subir un nuevo proyecto al servidor. Las rutas hacen un uso completo de los verbos de HTTP como **GET**, **POST**, **DELETE**, **UPDATE**, etc.

1.4.3. Middlewares

Laravel hace un extenso uso de los *Middlewares*. Podemos definirlos como software intermediario que proporciona una capa de abstracción entre una aplicación y otra. En el caso de Laravel, nos proporciona por defecto

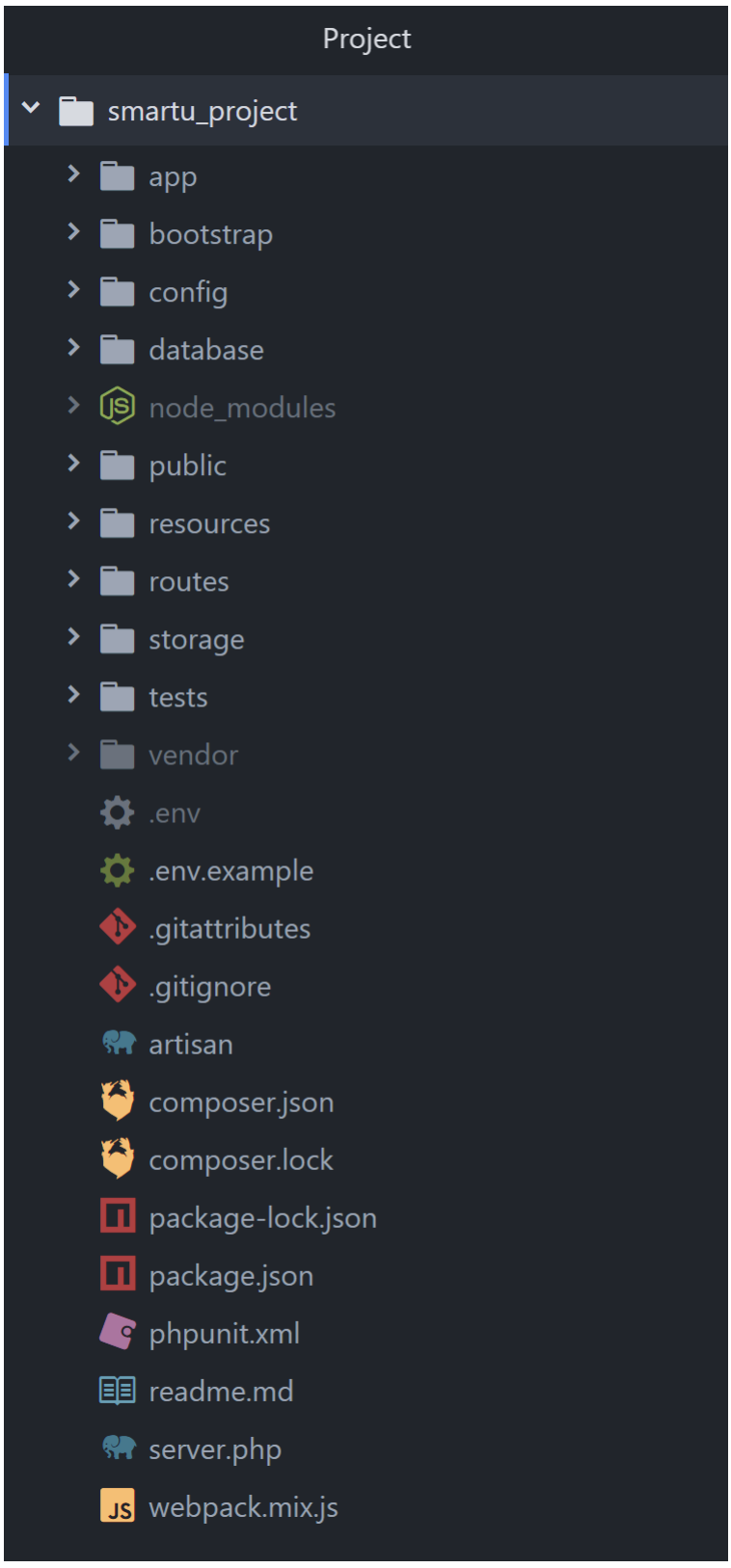

Figura 1.4: Estructura de carpetas y ficheros del proyecto

una serie de middlewares que realizan varias tareas sin que nosotros tengamos que intervenir en ellas, liberando al programador de carga de trabajo.

Uno de los principales middleware que proporciona este framework es el de autenticación. Permite, por ejemplo, que un usuario autentificado en el sistema pueda realizar ciertas tareas y evita que aquellos que no lo están las puedan hacer, en cuyo caso los redigire a otra dirección que si tengan permitida. Otro middleware importante es el de la verificación del token CSRF, pero esto lo hablaremos con más detalle en el apartado de seguridad.

Middleware de localización

Con el objetivo de lograr que la aplicación web tuviera soporte **multilenguaje**, se ha hecho uso de un middleware personalizado que captura los parámetros de la URL y detecta si existe un código de idioma correcto, y en caso de no tenerlo, le añade el del idioma por defecto de la aplicación. Como se puede comprobar, los middlewares nos permiten delegar ciertas tareas para no tener que preocuparnos de nada y poder continuar el desarrollo.

1.4.4. Patrones de diseño

El uso de patrones de diseño es muy recomendable a la hora de desarrollar software. Laravel, por su funcionamiento interno, logra aplicar el patrón MVC mencionado anteriormente, pero no es el único que encontramos. Existen más patrones de diseño que se aplican para determinadas características, como por ejemplo:

Builder El patrón Builder permite separar la creación de un objeto complejo de su representación, con el fin de que el mismo proceso de construcción pueda usarse para crear diferentes representaciones. En Laravel podemos tomar como ejemplo la clase *AuthManager*, que hereda de *Manager*, siendo esta última la clase Builder que se encarga de construir los componentes seguros necesarios para almacenar informaciónde autentificación, ya sea en una variable de sesión o en una cookie.

Factoría Este patrón permite definir una interfaz común de creación de objetos, pero deja a las subclases que sean ellas las que decidan qué objeto es el que quieren instanciar. Laravel hace un extenso uso de este patrón, por ejemplo, a la hora de crear conjuntos de reglas de validación de datos (usados habitualmente para comprobar los que se reciben a través de un formulario).

Estrategia El patrón Estrategia o también conocido como *Policy*, define una familia de algoritmos y los encapsula cada uno, haciendo que estos puedan ser reutilizados en cualquier momento.

Fachada El patrón Fachada proporciona una interfaz unificada a un conjunto de interfaces de un subsistema. La fachada define una interfaz de alto nivel que hace que el subsistema sea más sencillo de usar. Laravel está construido con el uso de este patrón en casi todas partes, agrupando funcionalidades complejas en unas más simples de usar.

1.5. Pruebas

Para realizar la fase de pruebas de este proyecto, se ha seguido el proceso habitual de una **metodología ágil**, es decir, probar que las funcionalidades de la aplicación web funcionan correctamente en todos los posibles casos de uso. En concreto, se ha seguido el siguiente proceso:

- El desarrollo se ha realizado implementando en primer lugar las **funcionalidades *core*** o ensenciales de la aplicación, es decir, la gestión de los usuarios y los proyectos, así como la gestión de los diferentes idiomas que pueda haber disponibles.
  Se ha garantizado mediante pruebas que en este punto es posible crear usuarios y proyectos, y los usuarios invitados (no registrados) sólo pueden hacer operaciones de consulta.
  También se ha garantizado que el cambio de idioma de la aplicación se hace correctamente.

- Tras la funcionalidad esencial, se han implementado las **pequeñas funcionalidades** que no presentan un fuerte acoplamiento con otras y que, por ende, no dependen del funcionamiento de éstas, de modo que se puede garantizar más rápidamente su correcto funcionamiento.
  Entre las funcionalidades que se consideran “auto-contenidas” encontramos los comentarios de un proyecto, los avances, los “Buena Idea”, las áreas y las especialidades.

- Por último, una vez que se ha realizado todo lo necesario para probar la funcionalidad esencial y las pequeñas funcionalidades autocontenidas, se procedió a la implementación de las **funcionalidades más complejas** que requerían de la interacción entre otros componentes de la aplicación. Cada funcionalidad se iba probando en todos los posibles casos de uso.
  Aquí se incluye toda la funcionalidad relativa a las vacantes y las solicitudes para incorporarse como integrantes de un proyecto, y las notificaciones.

Debido a los problemas de tiempo que he tenido para realizar este proyecto, no he podido realizar pruebas más exahustivas de la implementación, o incluso pruebas unitarias. Para futuras ampliaciones de este proyecto (ya que se espera una continuación del mismo), sería apropiado realizar una colección de pruebas que se pudieran ejecutar de forma automatizada antes de proceder a incrementar el conjunto de funcionalidades de la aplicación web.

1.6. Herramientas de desarrollo

Para el desarrollo de este proyecto he utilizado el siguiente conjunto de herramientas y programas de desarrollo:

- Laravel 5.5 con la versión 7 del lenguaje PHP.
- Servidor de pruebas XAMPP con instalación de Apache y MariaDB.
- Editor de texto Atom 1.19.
- Draw.io (diagramas del proyecto).
- Windows 10 Pro Versión 1703.

El proyecto es fácilmente portable a otros entornos de desarrollo siguiendo las instrucciones del anexo de intalación incluido en esta documentación.

Bibliografía