PLANTIUM

Plantium

Memoria de Trabajo Final de Grado

Grado Multimedia

Desarrollo de aplicaciones interactivas

Autor: Javier Searle Riesgo

Consultor: Kenneth Capseta Nieto

Profesor: Kenneth Capseta Nieto

13 de enero de 2017

UOC
Universitat Oberta de Catalunya

Créditos/Copyright

Plantium

Laravel [1]

https://github.com/laravel/laravel

FullCalendar [4] ***(Standard Edition)***

https://fullcalendar.io/license/

DataTables [5]

https://datatables.net/license/mit

Dedicatoria

Dedicada a mi familia y a mi pareja, Silvia,
por alentarme siempre que las fuerzas flaquean.

Abstract

El objeto de este documento es la realización de una memoria de trabajo de fin de carrera para la carrera Grado Multimedia, en el cual se proyecta el desarrollo de una aplicación interactiva: **Plantium**

Plantium es plataforma interactiva cuya intención es facilitar la gestión y planificación de tareas para la realización de cualquier proyecto.

La idea de la creación de una aplicación de gestión de procesos por tareas surge de la necesidad personal de una herramienta que cubra este propósito de forma simple y rápida. La idea de poder visualizar las tareas personales del día a día es el motivación principal que me ha llevado a realizar este proyecto. Una aplicación que de un vistazo muestre los eventos personales, laborales, tareas pendientes y sus prioridades del día a día, a modo de agenda personal, es algo trivial en un principio, pero es en la simplicidad en lo que radica esta herramienta.

También el hecho de poseer un buen *know-how* del multimedia y el desarrollo web en concreto, motiva que el proyecto se plasme en forma de esta aplicación web, de desarrollo asequible, no obstante, pretende ser ambiciosa en cuanto al espectro de materias vistas en la carrera. El conocimiento previo de *frameworks* MCV (Model - Controller - View) al estilo de Laravel, así como el conocimiento de HTML, PHP, Javascript y CSS facilita los medios para su desarrollo.

La aplicación web consiste en una plataforma en la cual habrá que darse de alta como usuario para poder acceder a la gestión de cada proyecto. Un mismo usuario podrá dar de alta distintos proyectos, los cuales tienen una asignadas una serie de tareas/procesos en distintas vistas de líneas de tiempo: vista calendario, vista listado y línea temporal por proyectos.

Palabras clave: Plantium, Memoria, Trabajo de Final de Grado, 2016, aplicación interactiva, planificación de tareas.

Abstract (english version)

The purpose of this document is the realization of an end-of-course project memory for the Multimedia Degree, in which the development of an interactive application is planned: **Plantium**

Plantium is an interactive platform whose intention is to facilitate the management and planning of tasks for the realization of any kind of project.

The idea of creating a process management application by task arises from the personal need for a tool that covers this purpose simply and quickly. The idea of being able to visualize the personal tasks of the day-to-day is the main motivation that has led me to carry out this project. An application that at a glance shows personal events, tasks, pending tasks and their day-to-day priorities, as a personal agenda, is somewhat trivial at first, but it is in the simplicity on which its is based.

Also the fact of possessing a good know-how of multimedia and the web development in particular, motivates that the project is embodied in a web application, of affordable development, nevertheless, it pretends to be ambitious in the spectrum of subjects studied in the degree. Prior knowledge of Laravel style MCV (Model - Controller - View) frameworks as well as knowledge of HTML, PHP, Javascript and CSS facilitates the means for its development.

The web application consists of a platform in which it will be necessary to register as a user to be able to access the management of each project. The same user can register different projects, which have one assigned a series of tasks / processes in different timeline views: calendar view, list view and project timeline.

Key words: Plantium, Memory, Final Grade Work, 2016, interactive application, task planning.

Índice

Figuras y tablas

Índice de figuras y tablas por apartados

1. Prefacio

Este trabajo tiene el objetivo de plasmar las diferentes disciplinas estudiadas durante la carrera en base a los conocimientos adquiridos durante la realización de la misma, en forma de aplicación interactiva.

El desarrollo de esta aplicación contempla múltiples áreas estudiadas en el Grado Multimedia: diseño y lenguaje visual, programación, bases de datos, planificación de proyectos, presupuestos y estimaciones, diseño de interacción, arquitectura de la información, redes y programación backend, usabilidad, etc.
En definitiva, este desarrollo tiene la intención de abarcar un gran espectro de la malla curricular de la carrera.

Como resultado de una meditada reflexión en busca de una posible necesidad tecnológica, surgió la idea de desarrollar una aplicación web cuya función es la planificación de proyectos y la cual incorpora un sistema de seguimiento del proyecto por usuarios, un sistema de notificaciones, recordatorios y tres modos de vista del avance del proyecto mediante modos calendario, linea de tiempo y listado de hitos/eventos.

Figura : Logotipo de Plantium

2. Descripción

2.1 Descripción y justificación del proyecto

Plantium es una aplicación web para la gestión y planificación de proyectos.

En el mercado actual ya existen algunas aplicaciones web que cubren algunos de los servicios comentados. Sin embargo son aplicaciones algo complejas, añaden funcionalidades extra que Plantium no contempla: bancos de conocimiento, repositorios de datos, sistemas complejos de notificaciones, funcionalidades con pronunciadas curvas de aprendizaje, gestión de proyectos de gran magnitud llevados a cabo por un equipo de personas.

La mayoría de las aplicaciones están enfocadas a empresas que gestionan proyectos en equipo, y donde un proyecto implica a distintas personas que hacen distintas tareas y han de entenderse para llegar a tal fin. Estas no son las necesidades que cubre Plantium.

Además muchas veces implica costes extra, por lo que son de pago. Plantium es un producto gratuito.

Plantium es una aplicación sencilla, con una interfaz limpia y clara, en la que se visualiza el estado de cada proyecto de manera inmediata. Sin embargo, es una aplicación pensada para la vida personal, no para la gestión de un gran proyecto en equipo. Es una agenda personal, con sistema de recordatorios, donde poder visualizar los proyectos o tareas personales a corto, medio o largo plazo.

2.2 Características técnicas

Plantium en una aplicación web, es decir, accesible mediante un navegador. Toda información en ella guardada se almacena en un servidor remoto. El código fuente también está emplazado en el mismo servidor, actualmente servido por un servidor Ubuntu con el proveedor DigitalOcean.

Esta arquitectura mixta es de manera provisional, si a medio plazo sufre un incremento sustancial de visitas y número de usuarios, está preparada para una separación de código fuente y base de datos.

No será difícil el cambio de arquitectura a una base de datos mas estable y elástica (Amazon RDS o similar), así como con la ejecución del código fuente en instancias escalables o de procesado elástico (Amazon EC2 o similar).

El core de la aplicación se sostiene en lenguaje PHP bajo el framework MVC (Modelo - vista - controlador) Laravel desarrollado por Taylor Orwell. Laravel es un entorno estable, versátil, escalable, seguro y ampliamente testado con una amplia y dinámica comunidad de usuarios y una documentación detallada y bien actualizada.

La base de datos empleada es MySQL, aunque Laravel permite cambiar de manera relativamente sencilla a cualquier otro motor de base de datos (PostgreSQL, Oracle…).

2.3 Marco de desarrollo

El marco de desarrollo es el siguiente:

Existe un equipo de desarrollo (en principio sólo un programador) que accede al repositorio de código de BitBucket mediante GIT, un sistema de control de versiones.

Al plantearse una funcionalidad nueva o corrección, se procede al desarrollo. Se desarrolla y testa en un servidor local, un entorno de desarrollo para el programador.

Una vez probado y dado el visto bueno a la nueva funcionalidad, se sube al servidor de producción mediante un despliegue mediante GIT haciendo uso de la consola bash del servidor de producción. Se accede al servidor remoto 'Master' y se actualiza el código de la aplicación en producción mediante el comando *pull* de GIT. Si se han instalado componentes nuevos hay que aplicar los comandos oportunos de despliegue de dependencias.

3. Objetivos

El desarrollo de este proyecto pretende alcanzar el cumplimiento de una serie de objetivos que no se focalizan únicamente en que el resultado sea un producto final, útil, práctico y novedoso, sino mas bien conocer, experimentar y aprender todas las implicaciones transversales que conllevan el desarrollo de una aplicación web com métodos ágiles de desarrollo.

3.1 Principales

Objetivos clave:

- Desarrollo de una aplicación web interactiva competitiva, en un plazo viable, con herramientas y recursos limitados.
- Aprender a manejar las herramientas y metodologías que se plantean como medios para lograr un buen producto.
- Poner en marcha metodologías ágiles de desarrollo de software.
- Cumplimiento de los plazos establecidos en la planificación.

3.2 Secundarios

Objetivos adicionales:

- Aprendizaje del desarrollo ágil de un producto.
- Aplicación de un diseño coherente y acorde con los conocimientos adquiridos durante la carrera.
- Empleo de una programación de buena calidad y depurada, valiéndose de buenas prácticas de desarrollo.

4. Marco teórico

En las últimas décadas la sociedad moderna ha cambiado drásticamente, el individuo se ha vuelto más ambicioso en cuanto a como emplear su tiempo. Compaginar el tiempo libre con el trabajo, la familia… es una constante en nuestras vidas. No es ninguna barbaridad afirmar que el recurso más valioso de nuestras vidas es el tiempo, ya que el buen uso del mismo nos facilita emplearlo en tiempo de disfrute.

Una buena organización del tiempo pasa por la optimización de en qué lo empleamos. Nuestras vidas consisten en una secuenciación constante de tareas y quehaceres, y una buena organización de estas es primordial para una buena calidad de vida.

Para ello contamos con muchas herramientas, agendas, relojes, alarmas, aplicaciones móviles… Sin embargo en los últimos años una buena porción de la sociedad pasa mucho tiempo delante de un ordenador por razones laborales u de otras índoles como el ocio o las redes sociales. Durante todo este tiempo, surgen nuevos recordatorios, tareas, recados, planes que muchas veces no apuntamos y quedan en el olvido.

Recuerdo numerosas ocasiones en que personas con las que me he relacionado tanto en el entorno laboral como personal, han echado de menos una herramienta sencilla que cubra esta necesidad tan obvia. Pero es cierto que muchos de estos casos no han querido contar con las grandes plataformas (Google Calendar, iCal de Apple, etc.) que ya prácticamente gobiernan su vida y quieren preservar la estanqueidad de su privacidad frente a estos.

Plantium pretende ser una plataforma independiente, de código abierto, que en un futuro cercano, pueda instalarse en servidor privados, donde la seguridad la configura el interesado, tanto un particular como una organización.

Además, de tener interés a medio plazo, la plataforma podría ser adaptada a App para móviles, mediante una pequeña evolución de la API actual.

Hay muchas otras aplicaciones que cubren las funcionalidades de aporta esta plataforma, sin embargo esta aplicación radica en su sencillez.

5. Contenidos

En este apartado se describen los distintos apartados/contenidos de la herramienta en cuestión. Para comprender mejor cada apartado, y a modo introductorio, se puede describir así la plataforma:

> La aplicación web Plantium es una plataforma en la que el usuario se da de alta y registra sus tareas y proyectos para poder visualizar, crear, editar y eliminar los mismos.

Con esta breve descripción podemos hacernos una idea de los contenidos de que consta la aplicación:

5.1 Secciones de la aplicación web

- **Página de acceso**: donde el usuario introduce su e-mail y contraseña para acceder a la parte privada. También es aquí donde el usuario accede a la sección de recordatorio de la contraseña, en caso de haberla olvidado.

Figura : Página de entrada

- **Página de recuperación de contraseña**: en ella pregunta por el e-mail y solicita al usuario recibir un correo de recuperación.

Figura : recuperación de contraseña

- **Página de registro**: en ella se solicita el Nombre, e-mail y contraseña para acceder a la aplicación.

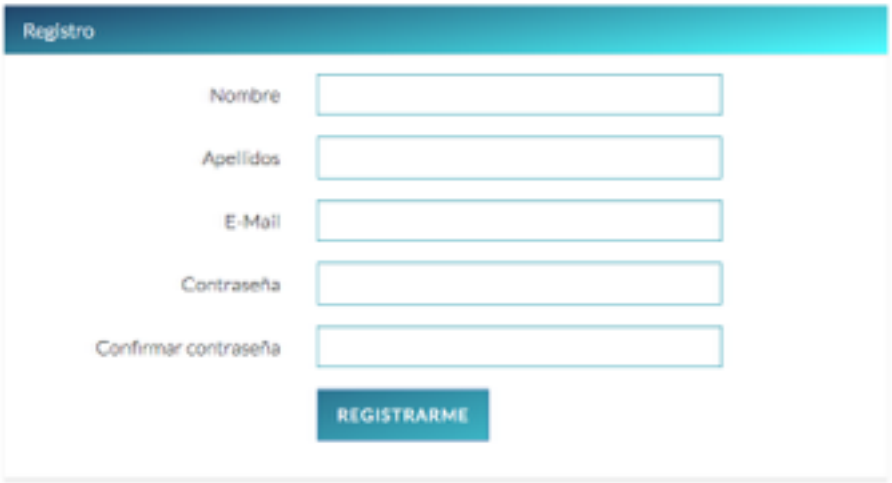


Figura : página de registro

- **Proyectos:**

  - **Vista de calendario:** en esta vista se visualizan los proyectos sobre un calendario, con la hora de comienzo.

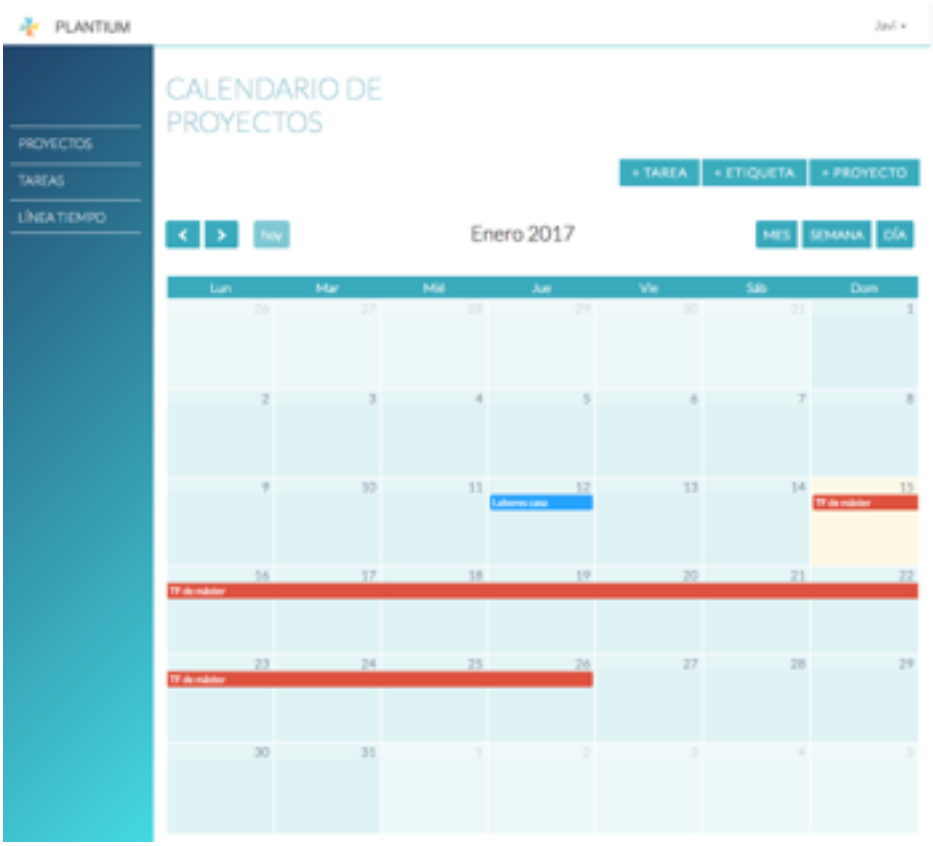


Figura : Vista calendario

  - **Vista listado:** en esta vista se visualizan los proyectos sobre un listado, con mas detalles que en el calendario.

Figura : Vista listado

- **Crear proyecto:** en este apartado se preguntan las características de un proyecto, nombre, descripción, fecha de comienzo y fecha de finalización.

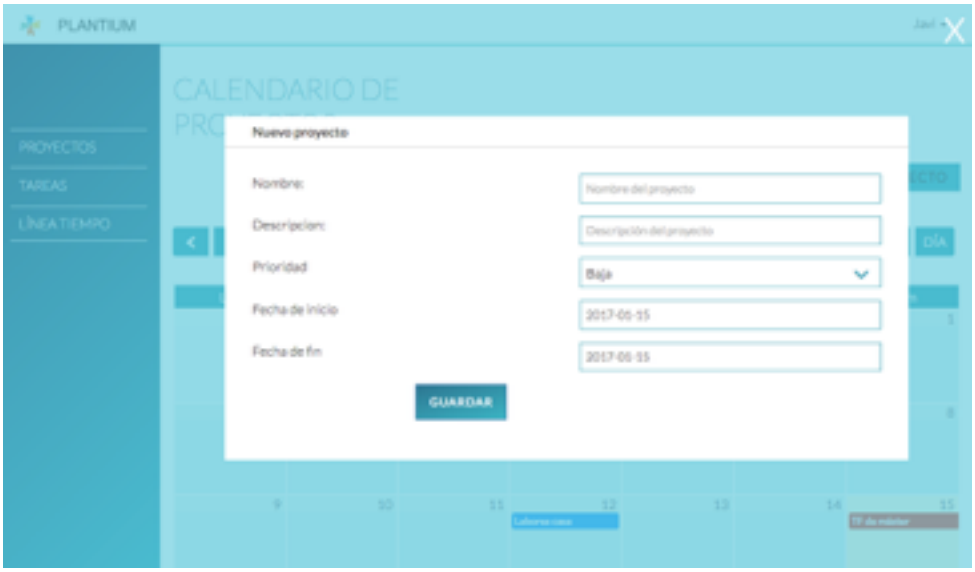


Figura : Pop-up nuevo proyecto

- **Editar proyecto**: aquí se pueden modificar las características de un proyecto, tanto el nombre, descripción, fecha de comienzo y fecha de finalización.

Figura : Edita proyecto

- **Tareas:**

  - **Vista de calendario:** en esta vista se visualizan las tareas sobre un calendario, con la hora de comienzo.

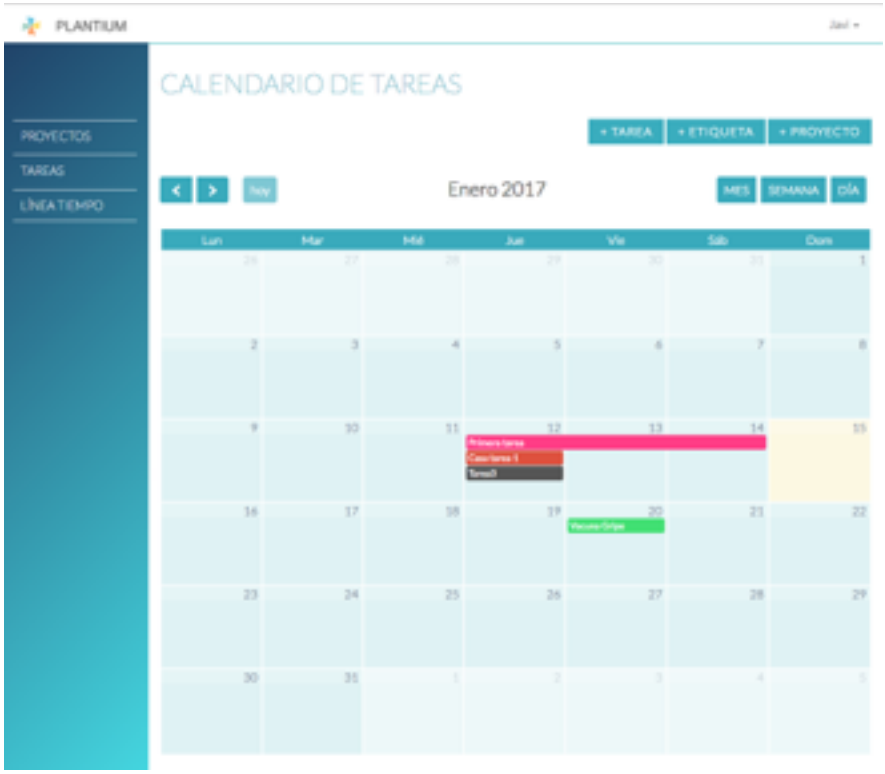


Figura : Vista calendario de tareas

  - **Vista listado:** en esta vista se visualizan las tareas sobre un listado, con mas detalles que en el calendario..

Figura : Vista listado de tareas

- **Crear tarea:** en este apartado se preguntan las características de una tarea, nombre, descripción, fecha de comienzo y fecha de finalización.

Figura : Vista creación de tareas

- **Editar tarea**: aquí se pueden modificar las características de una tarea, tanto el nombre, descripción, fecha de comienzo y fecha de finalización.

Figura : Vista edición de tareas

- Línea de tiempo: Vista de los proyectos y sus tareas asignadas.

Figura : Vista Línea de tiempo por proyectos/tareas

6. Metodología

6.1 Investigación y desarrollo de producto

El desarrollo de aplicaciones web actual hace uso de algunos métodos de desarrollo determinados. Las aplicaciones web suelen emplear un marco de integración continua, es decir, se van desarrollando sobre la marcha. Esta característica es posible debido a la facilidad de interacción con el código en el servidor, a diferencia de las aplicaciones AdHoc de escritorio.

Para un desarrollo eficiente y seguro, es conveniente el desarrollo de código en servidores estancos al de producción. El código de una aplicación web se desarrolla en un servidor de pruebas que puede estar instalado en la máquina del desarrollador o en remoto, y una vez terminada la funcionalidad se testa se depura. Tras esto se despliega al servidor en producción.

Por otro lado, la arquitectura de desarrollo MVC en frameworks modernos como Laravel, implica una serie de procesos de preparación un tanto particulares. Una vez estos son implementados el desarrollo se vuelve muy ágil y productivo.

Servidor de desarrollo y servidor en producción.

El proceso de desarrollo de una aplicación web se ve beneficiado con el empleo dos servidores, uno de desarrollo (en local) y otro de producción en un servidor accesible. Esta arquitectura de desarrollo permite mayor seguridad, y si se hace con las herramientas adecuadas, aporta una agilidad de desarrollo extra.

Se contempla la opción de hacer uso de un tercer servidor intermedio remoto que sirviese de pruebas donde poder realizar una tarea de testeo previo a el despliegue del proyecto a producción.

Al tratarse de un proyecto de pequeñas dimensiones llevado a cabo por un solo desarrollador, se toma la decisión de descartar el servidor de pruebas intermedio, ya que, en este caso no se considera rigurosamente necesario, con el servidor en local nos bastaremos para los test y pruebas.

Seguridad e integridad del código fuente

Para un correcto desarrollo haremos uso de sistema de sistema de control de versiones, en concreto, Git (https://git-scm.com/).

para ello se procede a la creación de un nuevo proyecto en la plataforma Bitbucket (https://bitbucket.org/product), gestionado por la aplicación Git con el entorno visual de SourceTree (https://www.sourcetreeapp.com) para la versión local y con el Git (https://git-scm.com/) por consola de comandos instalado en el servidor de producción.

En esta fase se procede a buscar una necesidad y desarrollar un producto que cubra esa necesidad. Supone una serie de procesos como son: *brainstorming*, investigación, reflexión, decisión y deshechado de ideas. Este proceso es iterativo y cíclico, pero en espiral, hasta la toma de una decisión final.

6.2 Planteamiento de funcionalidades

La plataforma está preparada para realizar las siguientes funcionalidades:

- Alta/registro de un usuario
- Login/acceso de un usuario
- Logout/cierre de sesión de un usuario
- Edición de perfil de un usuario
- Creación de un proyecto
- Creación de una tarea
- Creación de una etiqueta
- Edición de un proyecto
- Edición de una tarea
- Eliminación de una tarea
- Eliminación de una proyecto
- Eliminación de una etiqueta
- Visualización en vista calendario de proyectos
- Visualización en vista listado de proyectos
- Visualización en vista calendario de tareas
- Visualización en vista listado de tareas
- Visualización de tareas por proyectos en vista línea de tiempo
- Búsqueda de proyectos en vista listado
- Búsqueda de tareas en vista listado

6.3 Diseño funcional

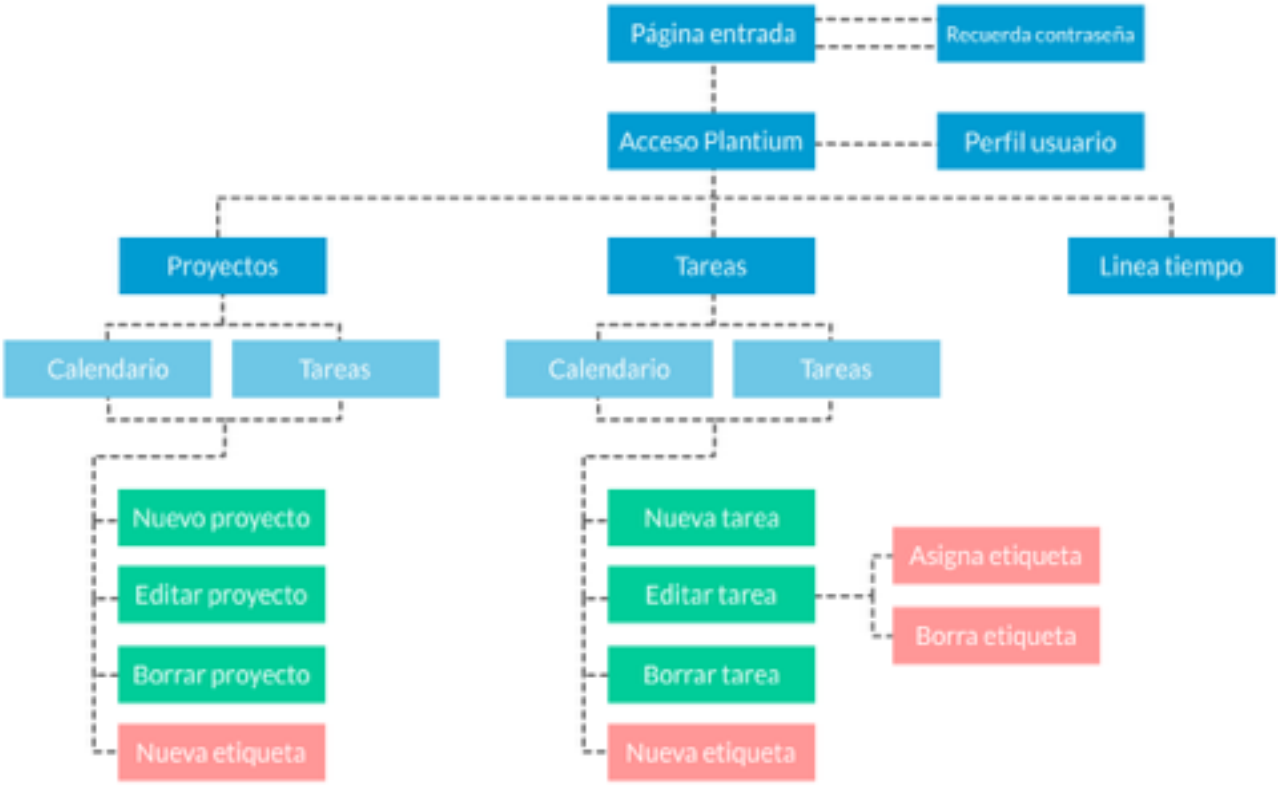


Figura : Diseño funcional

7. Arquitectura de la aplicación

7.1 Arquitectura HTTP

Plantium, al ser una aplicación web, hace uso de la arquitectura cliente-servidor. Como se explica en el apartado de metodología, el código fuente es desarrollado en un servidor en local, y una vez testado se sube al servidor en producción. En principio, y mientras el rendimiento del servidor no se vea comprometido, la base de datos estará instalado en el mismo.

El usuario accede a la aplicación mediante su navegador accediendo al dominio plantium.com Entonces se lanza una petición GET y el servidor DNS re-dirige tal petición a nuestro servidor en producción, que estará instalado en DigitalOcean. Este es el encargado de generar las páginas dinámicamente, haciendo las consultas pertinentes a la base de datos, (que esta instalada en el mismo servidor) y enviando las respuestas al cliente.

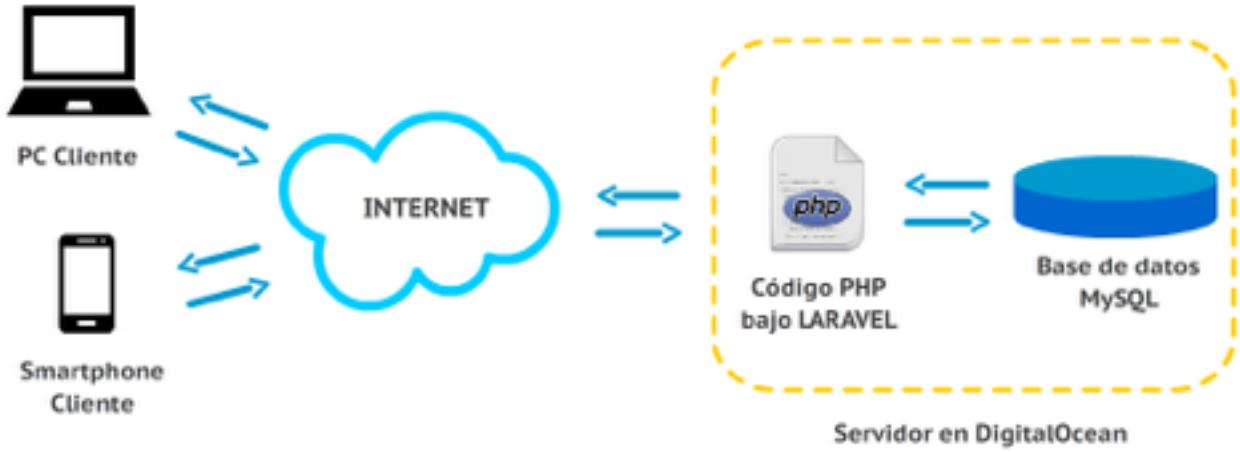


Figura : Esquema arquitectura Plantium

7.2 Estructura de la Base de datos

Usuario:

- id
- name
- surname
- email
- password
- avatar
- configuración (interfaz)
- remember_token
- Timestamps

proyecto:

- id
- id_usuario
- nombre
- descripción
- prioridad
- fecha_ini
- fecha_fin
- timestamps

tarea:

- id
- id_usuario
- id_proyecto
- prioridad
- nombre
- descripción
- fecha_ini
- fecha_fin
- timestamps

etiquetas

- id
- titulo
- color

imagenes

- id
- id_user
- fichero

- nombre

tareas_etiquetas:

- id
- id_tarea
- id_etiqueta

proyectos_etiquetas:

- id
- id_tarea
- id_proyecto

8. Plataforma de desarrollo

El proyecto se ha realizado con el editor de código: SublimeText 3 [5]. El desarrollo en servidor, backend, se ha realizado en lenguaje PHP en su versión 5.6 mediante el *framework* Laravel V5.1 [1]. El desarrollo interactivo ejecutable en cliente se ha desarrollado con Javascript y su framework jQuery [3]. Los estilos en CCS3.
Los frameworks:

8.1 Laravel

Para el lado del servidor: back-end, se ha escogido el framework Laravel.

Laravel es un framework de desarrollo de aplicaciones y servicios web de código abierto. Este framework es muy versátil. Posee una buena documentación y está amparado por una gran comunidad que lo mantiene y evoluciona de manera constante. Laravel está inspirado en otros marcos de desarrollo como Ruby on Rails, Sinatra…

Características principales de Laravel:

- Sistema de rutas API-RESTfull
- Motor de plantillas Blade
- Peticiones Fluent
- Eloquent ORM
- Basado en Composer
- Soporte para el caché
- Soporte para MVC
- Usa componentes de Symfony
- Adopta las especificaciones PSR-212 y PSR-413

8.2 jQuery [3]

Para el desarrollo de Front-end en el lado del cliente se emplea Javascript empleando su afamada librería jQuery. Esta librería de Javascript de código abierto, permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web (https://es.wikipedia.org/wiki/JQuery).

9. Planificación

La elaboración de un proyecto de esta magnitud obliga a dividir cuidadosamente las distintas tareas que implica y a estimar y distribuir en el tiempo cuatrimestre. Por lo tanto la planificación contempla las siguientes fases y sus tiempos de entrega:

28/09/2016	Búsqueda de una idea de aplicación interactiva y desarrollo del producto. Llegados a este punto, habrá una consulta de viabilidad con el tutor-consultor y en función de su *feedback*, se repite todo el proceso o se da por decidida una idea final.
17/10/2016	Desarrollo de la idea y planteamiento de las distintas funcionalidades teniendo en cuenta el factor de la viabilidad y de los *deadlines* o plazos de entrega. Desarrollo del backend de la aplicación: Puesta en marcha del servidor, bases de datos, PHP (bajo framework Laravel).
01/11/2016	Esquematización del funcional la aplicación.
10/11/2016	Creación de cada apartado separado por funcionalidad. Arquitectura de la información, primera fase de diseño de interacción y estudio de la usabilidad. Como resultado: realización de wireframes.
10/12/2016	Elaboración del diseño web final de cada apartado.
10/12/2016	Desarrollo del front-end, maquinación de la aplicación.
18/12/2016	Creación de las APIs y servicios pertinentes. Conexión de back-front.
28/12/2016	Fase de tests y depuración de la aplicación final.
05/01/2017	Documentación.

Tareas constantes, durante la realización del proyecto:

- Replanteamiento de viabilidad y recorte de funcionalidades así como supresión de la planificación de las inviables. Estas tareas se valorará archivarlas e incorporarlas en funcionalidades aplicables a futuro.
- Realización de la memoria:
    - Entrega PEC 1 - 04/10/2016

- Entrega PEC 2 - 02/11/2016
- Entrega PEC 3 - 04/12/2016
- Entrega final - 16/01/2017

10. Proceso de desarrollo

El primer paso ha sido elaborar el funcionamiento de la aplicación mediante un gráfico funcional, se comienza a diseñar las pantallas en forma de bocetos.

Una vez se dan por buenos comienza el proceso de diseño real. Esta parte está condicionada por muchas variables, como el color, las tipografías empleadas, las librerías empleadas para el calendario. Todo ello bien trabajado da lugar a un diseño que pasa a ser maqueado en HTML/CSS.

Paralelamente se avanza en la configuración del framework y el desarrollo backend.
Antes de esta fase conviene recordar el apartado 3.2 Marco de desarrollo.
El proceso se podría explicar así:

Se pone en marcha un servidor en local y otro en producción. Se configuran los frameworks y el servidor para que sean lo mas parecido posible.

El equipo de desarrollo (en principio sólo un programador) que accede al repositorio de código de BitBucket mediante GIT, un sistema de control de versiones.

Al plantearse una funcionalidad nueva o corrección, se procede al desarrollo. Se desarrolla y testa en un servidor local, un entorno de desarrollo para el programador.

Una vez probado y dado el visto bueno a la nueva funcionalidad, se sube al servidor de producción mediante un despliegue mediante GIT haciendo uso de la consola bash del servidor de producción. Se accede al servidor remoto 'Master' y se actualiza el código de la aplicación en producción mediante el comando *pull* de GIT. Si se han instalado componentes nuevos hay que aplicar los comandos oportunos de despliegue de dependencias.

De esta forma va evolucionando paralelamente las funcionalidades de la aplicación y el código, de forma fiable y segura.

11. Diagramas UML

La base de datos de la aplicación se puede dividir en dos partes. Por un lado estaría la propia de la aplicación y por otro está la inherente al framework Laravel.

Tablas de la aplicación :

- Users: el nombre se mantiene en inglés para evitar conflictos con el framework
- Proyectos: los usuarios tienen proyectos asociados
- Tareas: los usuarios tienen tareas asociadas a sus proyectos
- Imágenes: los usuarios poseen imágenes
- Etiquetas: las etiquetas se asocian a tareas y proyectos
- poryectos_etiquetas: tabla pivot de ambas tipologías
- tareass_etiquetas: tabla pivot de ambas tipologías

Tablas del framework:

- Users: el nombre se mantiene en inglés para evitar conflictos con el framework, hay campos retocados
- Passwords reset:tabla de respeto de contraseñas
- Migrations: histórico todas las migraciones ejecutadas

Esquema de la base de datos

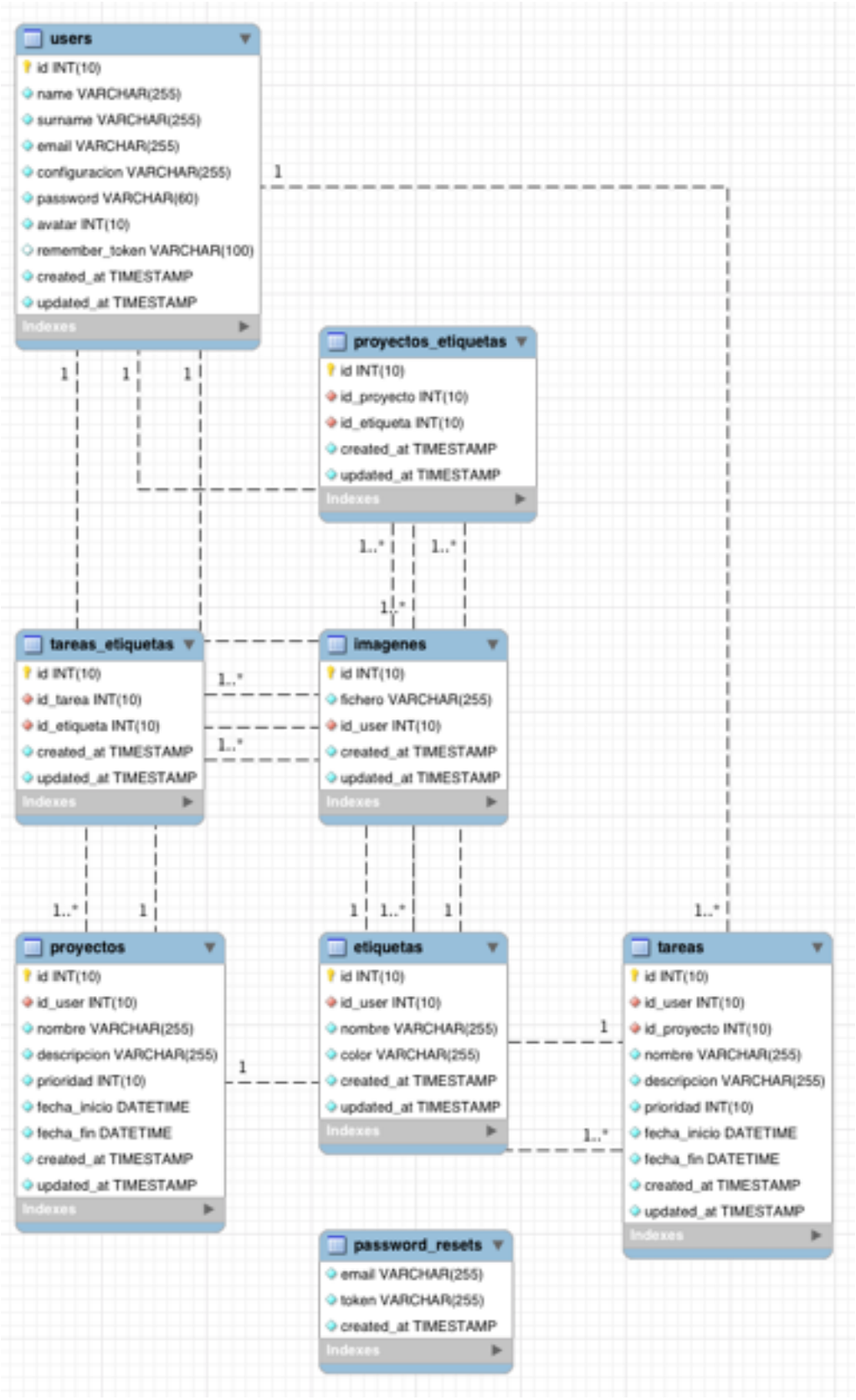


Figura : recuperación de contraseña

12. Prototipos

Durante el proceso de planificación se han creado dos prototipos, un boceto inicial muy esquemático (Lo-Fi) y posteriormente un prototipo algo más elaborado Hi-Fi.

12.1 Lo-Fi

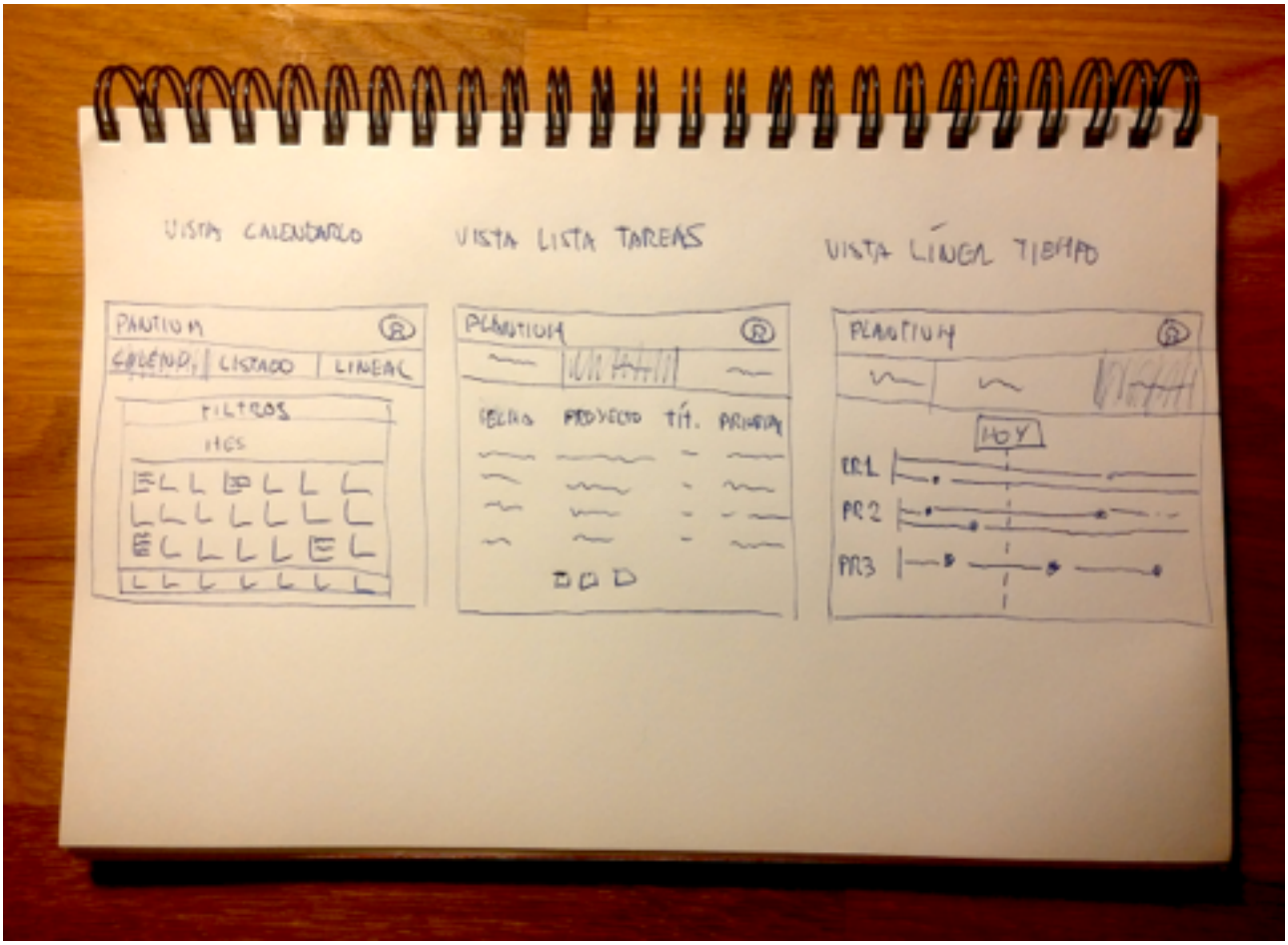


Figura : Wireframe inicial

12.2 Hi-Fi

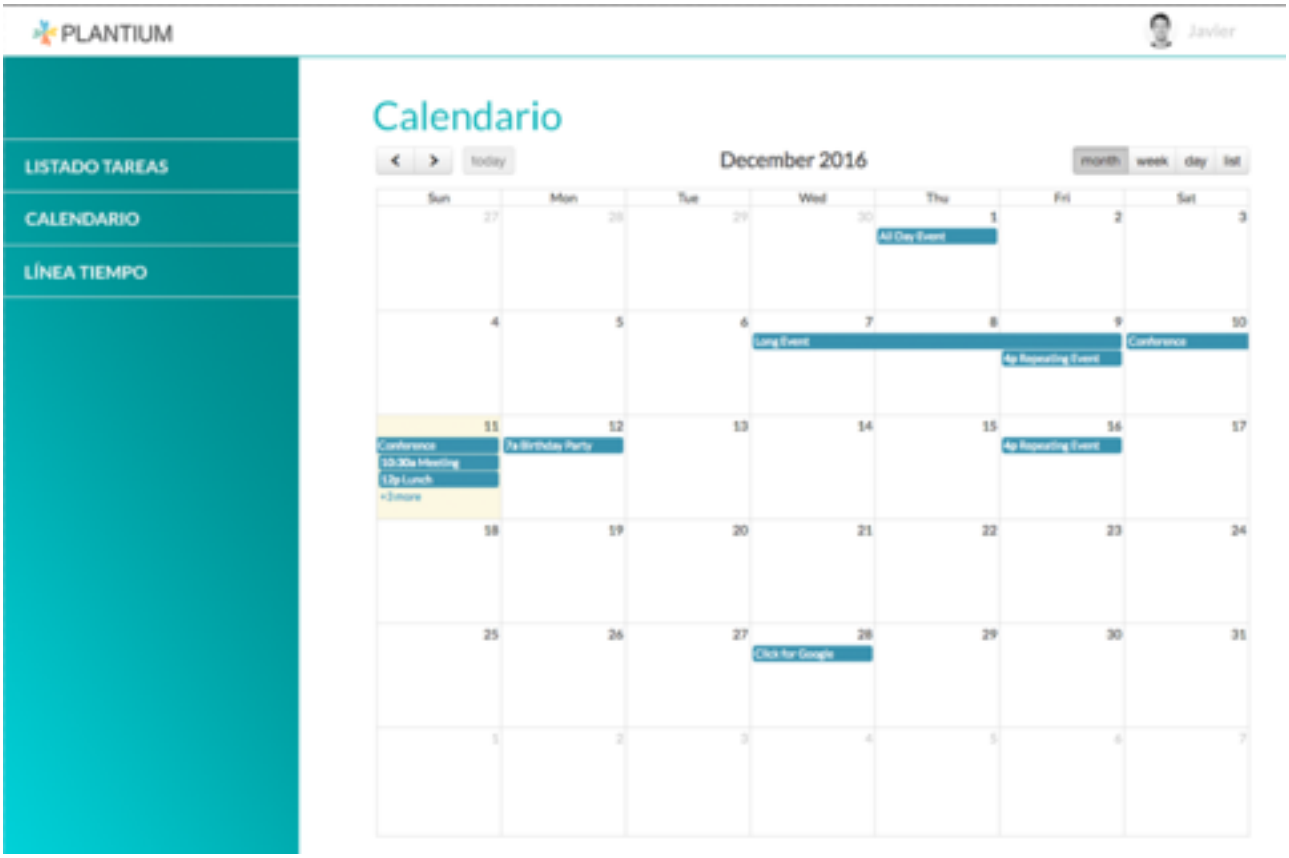


Figura : Prototipo calendario

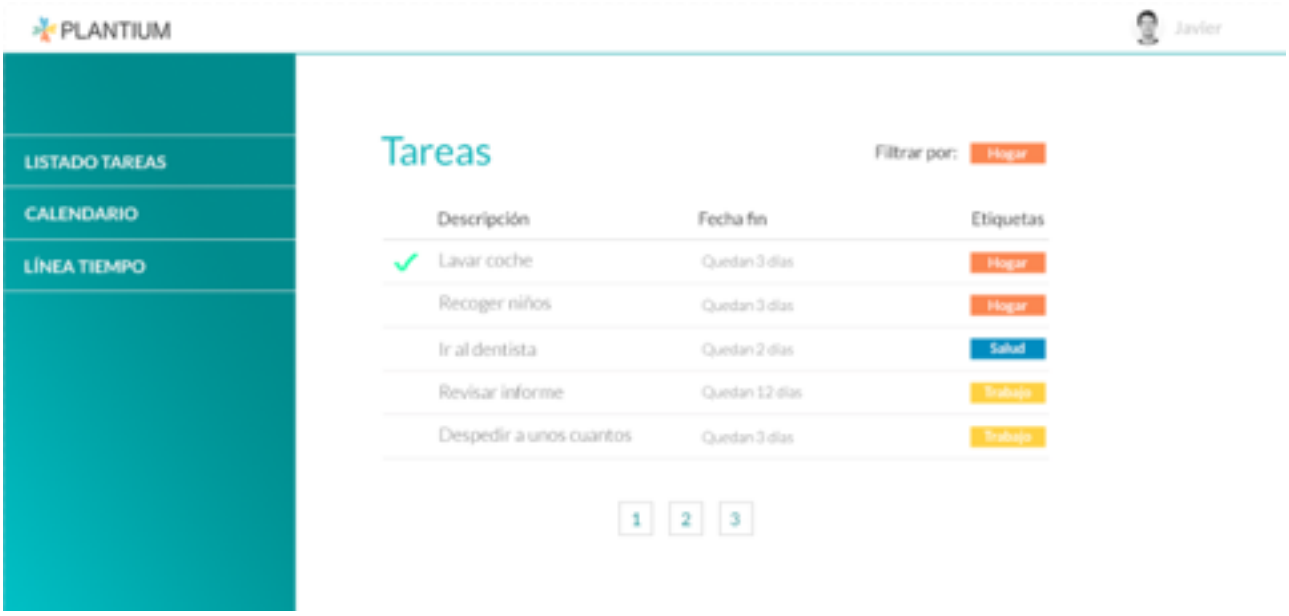


Figura : Prototipo listado tareas

13. Usabilidad/UX

13.1 Criterios de usabilidad

Durante el proceso de diseño de usabilidad y de diseño de interacción, así como en la navegabilidad se ha tenido muy presente que la aplicación Plantium pretende ser intuitiva y poco arriesgada. El diseño a buscado en todo momento que la interfaz pase desapercibida, siguiendo los códigos de la navegación e interacción evidente.

Las librerías empleadas para la visualización de las tareas y eventos han sido modificadas y adaptadas a un diseño más coherente, añadiendo interacciones tipo *roll-overs* buscando la evidencia.

Se ha hecho uso de la librería Bootstrap, que también ha sido personalizada en cuanto a márgenes y otras características. Esta librería posee la virtud de la sencillez y está ampliamente universalizada.

13.2 Apartados

Como se puede apreciar en el apartado de diseño funcional, la plataforma no es compleja:

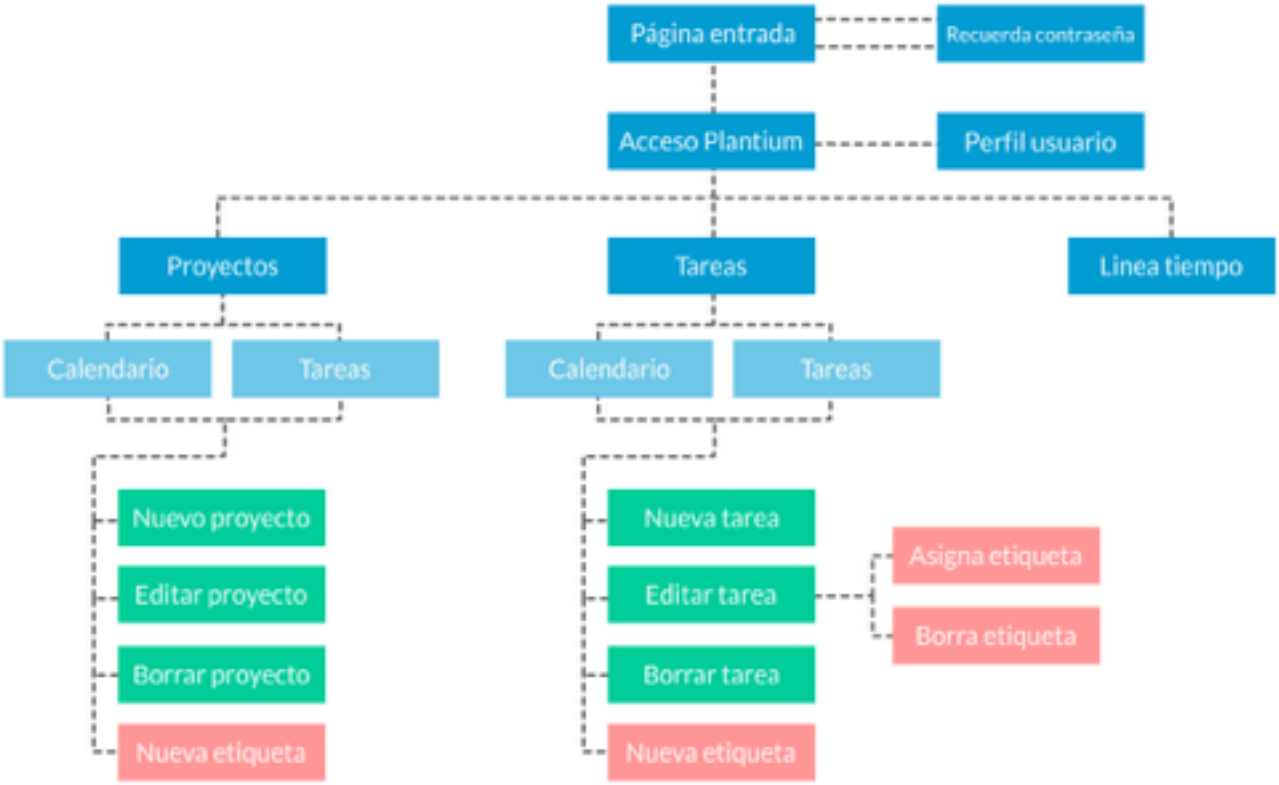


Figura : Apartados

13.3 Navegación

Para el máximo aprovechamiento de el espacio de la pantalla se ha procurado dar prioridad al calendario y al listado, ya que contienen mucha información en poco espacio.
Por esta razón se decide hacer uso del un menú lateral a la izquierda como eje navegación principal, con algún elemento desplegable:

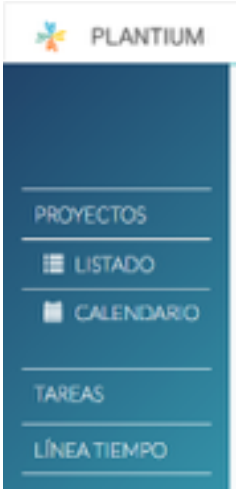


Figura : Menú lateral izquierdo

Además también se hace uso de un pequeño menú superior desplegable, donde poder acceder al perfil y a la acción de cierre de sesión:

Figura : Menú superior

Acciones en tabla de tareas / proyectos:

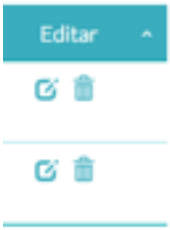


Figura : Menús contextuales

Menú contextual superior:

Figura : Menús contextuales

14. Seguridad

La seguridad que ha de conllevar una plataforma de estas características ha de ser exhaustiva y minuciosa. El tipo de arquitectura que emplea supone mucha exposición a ataques de múltiples naturalezas. Además de los los riesgos que previene el empleo del framework Laravel, se aplica por activa una segunda capa de seguridad que cubre gran parte del espectro de riesgos más conocidos. Los principales son los siguientes:

- Filtración de información.
- Ataque a la base de datos.
- Cross-Site Scripting (XSS).
- Cross-Site Request Forgery (CSRF).
- Inyección de código MySQL en formularios.
- Suplantación de identidad, por peticiones a la API.
- Ataques tipo Clickjacking

Para ello se toman las siguientes medidas preventivas:

- Validación de formularios por backend, la validación front-end previene únicamente peticiones innecesarias.
- Base de datos separable del servidor de ejecución de código.
- Prevención CSRF mediante token (csrf_field() de Laravel), tanto para la recepción de formularios como para la recepción de peticiones por AJAX mediante el empleo de *token* autenticados por sesión de usuario.
- API de escucha con autenticación por token.
- Securización redundante en controladores mediante comprobación de autenticación para peticiones CRUD ("Creación, Lectura, Edición y Borrado") (https://es.wikipedia.org/wiki/CRUD).
- Configuración X-FRAME-OPTIONS.

Consejos importantes para instalación del código en servidor propio:

- Instalación de certificado de seguridad en servidor (HTTPS)
- Separación de servidor de base de datos con respecto al servidor de ejecución de los archivos fuentes.
- Prevención de ataques Clickjacking mediante X-FRAME-OPTIONS en Apache, y correspondientes en otros servidores tipo Nginx, …
- Back-ups seguros y periódicos.
- Empleo de SSH con contraseñas cifradas, obligando al empleo de contraseñas livianas.
- Configuración de acceso a servidor de BBDD mediante única IP sólo accesible desde el servidor de código.

15. Tests

Durante todo el proceso de desarrollo se hacen pruebas y testeos puntuales de cada funcionalidad.

- Usuarios

  El usuario de pruebas principal, como suele ser en estos casos ha sido el programador. Por supuesto esta mala práctica se debe a la habitual falta de recursos y tiempo, sin embargo se han empleado puntualmente otros dos actores secundarios. Un perfil técnico también programador, y otra persona con perfil poco técnico

- Usabilidad

  Se han practicado pruebas puntuales con sujetos externos que alertado de graves problemas de usabilidad e interacción. Se ha tomado feedback y subsanado los errores. La usabilidad ha sido una de las áreas más trabajadas y que más jornadas ha conllevado para alcanzar un nivel satisfactorio en cuanto a depuración de fallos.

- Seguridad

  También se realizan distintas pruebas de acceso a servicios a los servicios abiertos, tanto por peticiones GET como por POST, PUT ... mediante el empleo de la aplicación POSTMAN. Se recoge feedback constante y se procede a la depuración:

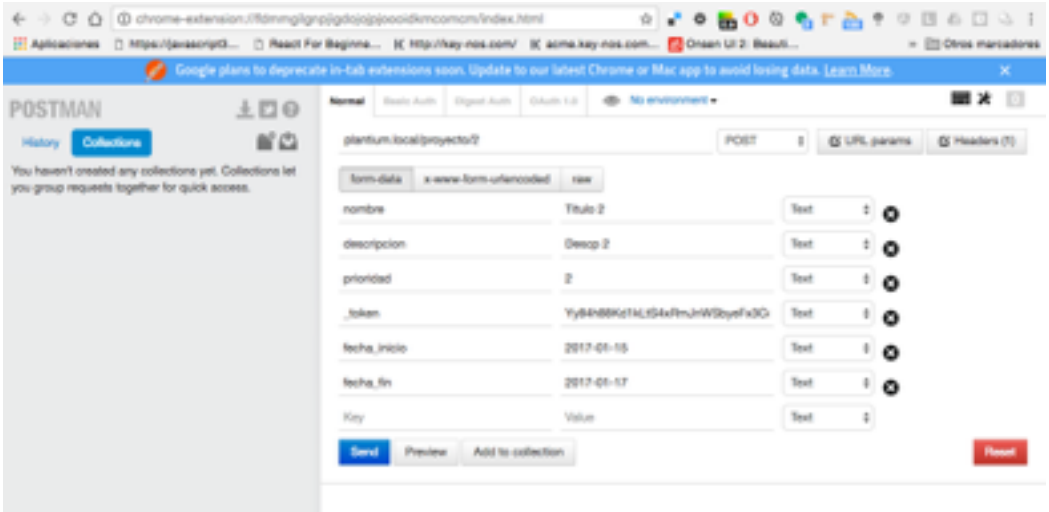


Figura : Pantallazo empleo de Postman para lanzar peticiones HTTP

16. Versiones de la aplicación

La situación actual de la aplicación es algo prematura para comenzar a comercializarse. Puede calificarse como versión beta, ya que faltaría por depurar ajustar y ampliar

17. Instrucciones de uso

Plantium posee unas funcionalidades muy claras e intuitivas. Pese a ello conviene aclarar algunos puntos que pueden llevar a confusión.

IMPORTANTE:
El registro en la aplicación no tiene un acceso directo en la página de entrada de manera intencionada. Ya que la versión es beta y está accesible a todo el mundo en su versión en producción, por esta razón no se pretende facilitar el acceso a la misma.

Para registrarse hay que acceder a la siguiente dirección en local:

http://plantium.local/auth/register

y en producción en el servidor actual:

http://www.big-beef.com/auth/register

A partir de este sencillo registro se dará acceso a la aplicación.

18. Bugs

Todos los bugs se han ido resolviendo sobre la marcha con sus correspondientes depuraciones. Se han realizado varias pruebas y test y no se ha localizado ningún error.

19. Proyección a futuro

Toda plataforma o aplicación web debe sostenerse, mejorarse y cuidarse para mantenerse viva. Plantium no es una excepción.

Al comenzar el proyecto las funcionalidades planteadas eran algo más ambiciosas de lo que es la situación actual del desarrollo. Pero la falta de tiempo principalmente ha ido configurando esta planificación sobre la marcha y durante el proceso se decide sacrificar alguna de estas funcionalidades a cambio de mejorar otras.

Algunas de las funcionalidades sacrificadas, que podrían reemprenderse serían las siguientes:

- Añadir estado a tareas y proyectos: “en proceso”, “finalizado”, “pendiente”
- Posibilitar la inclusión de imágenes, asociadas a un proyecto, una tarea, …
- Eventos arrastrabas desde el calendario
- Mejorar la visualización de “Línea tiempo”

20. Marketing y Ventas

La aplicación Plantium se ha desarrollado sin grandes pretensiones comerciales, sino más bien como experimento de aprendizaje de desarrollo de aplicaciones ágiles, con la intención de perfeccionar el desarrollo mediante la experiencia.

21. Conclusiones

El objetivo inicial de la elaboración de este proyecto era adquirir conocimientos y experiencias en la realización de una aplicación que abarcase el mayor espectro posible de las materias de la carrera . Puedo concluir sin duda que el objetivo se ha cumplido con creces.

La orientación que he dado al proyecto no iba encaminada a la creación de una aplicación culmen de grandes ambiciones. No quiero decir con esto en que desprecie el valor de la misma, sino que la mayor parte de los esfuerzos se centraban en el perfeccionamiento de técnicas de desarrollo ágil, de poder llevar a cabo un proyecto desde cero, con recursos limitados y con resultado satisfactorio. La experiencia de realizar el proceso completo me ha aportado mucha experiencia y nuevos conocimientos.

Honestamente, las sensaciones finales son algo confrontadas en cuanto a una extraña mezcla de la sensación de satisfacción con el producto final frente a la vez de haber podido dar un poco más. Sin embargo esta última sensación se merma cuando reflexiono en que los recursos temporales eran muy limitados y en que el esfuerzo ha cundido no solo en forma de aplicación web, sino también en experiencia y conocimiento.

Anexo 1. Entregables del proyecto

Archivo .zip con todos los archivos necesarios para instalar la aplicación en un servidor que soporte Laravel 5.1.

Para la instalación se recomienda seguir las indicaciones de Laravel:

https://laravel.com/docs/5.1/

Nota: Los archivos de Node no están incluidos ni desplegados debido a que ocupan un espacio desproporcionado (>200MB). Hay que tener en cuenta que una vez instalado se deben desplegar las dependencias tanto a nivel back como a nivel front con los siguientes comandos:

Para el despliegue de los componentes de Laravel/Sinfony:

```
composer update
```

Para el despliegue de los componentes de Javascript/CSS mediante Elixir:

```
npm install
```

Esta acción desplegará los componentes necesarios nombrados en el fichero gulpfile.json

Para más información, seguir las instrucciones de la documentación de Laravel:

https://laravel.com/docs/5.1/elixir

Anexo 2. Código fuente (extractos)

En adelante se plasman muestras de código fuente ejecutadas en servidor, cliente así como un Dump de la base de datos.

Código fuente en servidor

Ejemplo de un controlador:
TareasController.php

```
<?php

namespace Plantium\Http\Controllers;


use Illuminate\Http\Request;
use Plantium\Http\Controllers\Controller;
use Illuminate\Support\Facades\Auth;


class TareasController extends Controller {
  public function tareasJSON(Request $request)
  {
    $tareas = Auth::user()->tareas()->get();
    $tareas_completas = [];
    foreach ($tareas as $num => $tarea) {
      $tareas_completas[ $num ] = $tarea;
      $tareas_completas[ $num ]['proyecto'] = $tarea->proyecto;
      $tareas_completas[ $num ]['etiquetas'] = $tarea->etiquetas;
    }
    return $tareas_completas;
    // $data['espacio'] = $espacio;
    // return view('rutas.index', $data);
  }
  public function tareasCalendario(Request $request)
  {
    $tareas = Auth::user()->tareas()->get();
    return view('calendario');
  }
```

```
public function tareasListado(Request $request)
{
   $tareas = Auth::user()->tareas()->get();
   return view('listado');
}
public function tareasLineal(Request $request)
{
   $tareas = Auth::user()->tareas()->get();
   return view('lineal');
}
public function store(Request $request, $id = 0){
   $this->validate($request, [
      'nombre' => 'required',
      'descripcion' => 'required',
      'prioridad' => 'required',
      'fecha_inicio' => 'required',
      'fecha_fin' => 'required',
   ]);
   if ($id == 0) {
      $id_proyecto = $id;
   } else {
      $id_proyecto = $id;
   }
   $nombre = $request->input( 'nombre' );
   $descripcion = $request->input( 'descripcion' );
   $prioridad = $request->input( 'prioridad' );
   $fecha_inicio = $request->input( 'fecha_inicio' );
   $fecha_fin = $request->input( 'fecha_fin' );
   $id_user = Auth::user()->id;
   \Plantium\Tarea::create([
      'id_user' => $id_user,
      'id_proyecto' => $id_proyecto,
      'nombre' => $nombre,
      'descripcion' => $descripcion,
      'prioridad' => $prioridad,
      'fecha_inicio' => $fecha_inicio,
      'fecha_fin' => $fecha_fin,
   ]);

   return view('calendario');
```

```
}
public function edit (Request $request, $id_tarea){
  $tarea = \Plantium\Tarea::find( $id_tarea );
  $data['tarea'] = $tarea;
  return view('tareas.edit', $data);
}

public function update(Request $request, $id = 0){
  $this->validate($request, [
    'nombre' => 'required',
    'descripcion' => 'required',
    'prioridad' => 'required',
    'fecha_inicio' => 'required',
    'fecha_fin' => 'required',
  ]);
  // if ($id == 0) {
  //    $id_proyecto = $id;
  // } else {
  //    $id_proyecto = $id;
  // }
  $tarea = \Plantium\Tarea::find( $id );

  $nombre = $request->input( 'nombre' );
  $descripcion = $request->input( 'descripcion' );
  $prioridad = $request->input( 'prioridad' );
  $fecha_inicio = $request->input( 'fecha_inicio' );
  $fecha_fin = $request->input( 'fecha_fin' );
  $id_user = Auth::user()->id;
  $tarea->update([
    'id_user' => $id_user,
    'id_proyecto' => 1,
    'nombre' => $nombre,
    'descripcion' => $descripcion,
    'prioridad' => $prioridad,
    'fecha_inicio' => $fecha_inicio,
    'fecha_fin' => $fecha_fin,
  ]);

  return view('calendario');
```

```
  }

}
```

Código fuente en cliente

Javascript (jQuery[3]):

```
$('#bt_crea_tarea').click(function(event) {
                                        event.preventDefault();
                                        window.location = '/{{ Request::segment(1) }}/
{{ Request::segment(2) }}/tareas/create?fecha_inicio=' + moment( dayDelta._d ).format("YYYY-MM-DD");
                                });
                                $('#bt_crea_evento').click(function(event) {
                                        event.preventDefault();
                                        window.location = '/{{ Request::segment(1) }}/
{{ Request::segment(2) }}/eventos/create?fecha_inicio=' + moment( dayDelta._d ).format("YYYY-MM-DD");
                                });
                                $('#bt_crea_recordatiorio').click(function(event) {
                                        event.preventDefault();
                                        $('.botonera-flotante-calendario').hide();

                                        //Creación del evento
                                        $('.wrapper-popup-calendario h1').text("Nuevo evento");
                                  //transformo el formulario de edición en formulario de creación
                                        $('#evento #metodo').remove();
                                        $('#evento').attr('action', '/{{ Request::segment(1) }}/
{{ Request::segment(2) }}/eventos');
                                $('#titulo').val('');
                                $('#descripcion').val('');
                                        $('.wrapper-popup-calendario').addClass('open');
                                        $('.wrapper-popup-calendario').fadeIn(200);
                                        $('header, section, footer').addClass('blur');
                                $('#fecha_inicio').val(dayDelta.format("YYYY-MM-DD"));
                                $('#hora_inicio').val(dayDelta.format("hh:mm"));
```

```
                $('#fecha_fin').val(dayDelta.add(1, 'hours').format("YYYY-MM-DD"));
                $('#hora_fin').val(dayDelta.add(1, 'hours').format("HH:mm"));

                    });
```

Ejemplo de CSS ejecutado en servidor:

…

```
.navbar{
 border-bottom: 1px solid #00a2a6;
 background-color: #fff;
}

.navbar-brand img.anagrama{
   height: 25px;
   width: auto;
   float: left;
   padding: 0 10px;
}
.navbar-brand img.txt{
   height: 15px;
   width: auto;
   float: left;
   padding: 0 10px;
   margin-top: 5px;
}
.menu-izq{
 background: -webkit-linear-gradient(135deg,#00b1b3,#1d5086);
 background: linear-gradient(-45deg,#00b1b3,#1d5086);
 background: -webkit-linear-gradient(135deg,#55fdff,#214368);
 background: linear-gradient(-45deg,#55fdff,#214368);
 height: 100%;
 position: absolute;
 top: 50px;
}

.menu-izq ul {
 margin-top: 100px;
}
.menu-izq ul li{
```

```
 border-top: 1px solid #fff;
 transition: 0.3s;
 -ms-transition: 0.3s;
 -moz-transition: 0.3s;
 -webkit-transition: 0.3s;
}
.menu-izq ul li a{
 display: block;
 color: #fff;
 text-transform: uppercase;
 padding: 10px;
 font-weight: 200;

 transition: 0.3s;
 -ms-transition: 0.3s;
 -moz-transition: 0.3s;
 -webkit-transition: 0.3s;
}
...
```

Base de datos

MySQL generado por la migración de Laravel:

```
CREATE TABLE `etiquetas` (
 `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
 `id_rel` int(10) unsigned NOT NULL,
 `tipo_rel` varchar(2) COLLATE utf8_unicode_ci NOT NULL,
 `nombre` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
 `color` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
 `created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
 `updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
 PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;



# Volcado de tabla imagenes
# ------------------------------------------------------------
```

```
CREATE TABLE `imagenes` (
 `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
 `fichero` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
 `id_user` int(10) unsigned NOT NULL,
 `created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
 `updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
 PRIMARY KEY (`id`),
 KEY `imagenes_id_index` (`id`),
 KEY `imagenes_id_user_index` (`id_user`),
 CONSTRAINT `imagenes_id_user_foreign` FOREIGN KEY (`id_user`) REFERENCES `users` (`id`) ON
DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;



# Volcado de tabla migrations
# ------------------------------------------------------------

CREATE TABLE `migrations` (
 `migration` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
 `batch` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;



# Volcado de tabla password_resets
# ------------------------------------------------------------

CREATE TABLE `password_resets` (
 `email` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
 `token` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
 `created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
 KEY `password_resets_email_index` (`email`),
 KEY `password_resets_token_index` (`token`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;



# Volcado de tabla proyectos
```

```
# ------------------------------------------------------------

CREATE TABLE `proyectos` (
 `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
 `id_user` int(10) unsigned NOT NULL,
 `nombre` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
 `descripcion` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
 `prioridad` int(10) unsigned NOT NULL,
 `fecha_inicio` datetime NOT NULL,
 `fecha_fin` datetime NOT NULL,
 `created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
 `updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
 PRIMARY KEY (`id`),
 KEY `proyectos_id_user_foreign` (`id_user`),
 CONSTRAINT `proyectos_id_user_foreign` FOREIGN KEY (`id_user`) REFERENCES `users` (`id`) ON
DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;



# Volcado de tabla tareas
# ------------------------------------------------------------

CREATE TABLE `tareas` (
 `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
 `id_user` int(10) unsigned NOT NULL,
 `id_proyecto` int(10) unsigned NOT NULL,
 `nombre` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
 `descripcion` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
 `prioridad` int(10) unsigned NOT NULL,
 `fecha_inicio` datetime NOT NULL,
 `fecha_fin` datetime NOT NULL,
 `created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
 `updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
 PRIMARY KEY (`id`),
 KEY `tareas_id_user_foreign` (`id_user`),
 KEY `tareas_id_proyecto_foreign` (`id_proyecto`),
 CONSTRAINT `tareas_id_proyecto_foreign` FOREIGN KEY (`id_proyecto`) REFERENCES `proyectos`
(`id`) ON DELETE CASCADE,
```

```
 CONSTRAINT `tareas_id_user_foreign` FOREIGN KEY (`id_user`) REFERENCES `users` (`id`) ON
DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;



# Volcado de tabla users
# ------------------------------------------------------------

CREATE TABLE `users` (
 `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
 `name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
 `surname` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
 `email` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
 `configuracion` varchar(255) COLLATE utf8_unicode_ci NOT NULL DEFAULT '{}',
 `password` varchar(60) COLLATE utf8_unicode_ci NOT NULL,
 `avatar` int(10) unsigned NOT NULL DEFAULT '1',
 `remember_token` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
 `created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
 `updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
 PRIMARY KEY (`id`),
 UNIQUE KEY `users_email_unique` (`email`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

Anexo 3. APIs y librerías / código externo utilizado

Composer - gestor de dependencias para PHP

Laravel [1][2]- Framework de PHP

Bestmomo/scafold - Paquete que implementa serrvicios y vistas para autenticación

Barryvdh/laravel-debugbar - Paquete que instala una barra para hacer depuración

Uxweb/sweet-alert - paquete que implementa un cómodo sistema de alertas

Vinkla/hashids - paquete que hashea los IDs para otorgar algo de seguridad y ocultación en el código

Mews/captcha - paquete que implementa el sistema captcha para como capa de seguridad

Gulp - gestor de dependencias para Javascript

jQuery[3] - Frameworjk de Javascript

DataTables[5] - Librería JS que implementa tablas enriquecidas

Moment.js - Librería JS que incluye varios aspectos relativas a las horas

Bootstrap - Librería basada en JS/CSS

FullCalendar[4] **-** Librería de jQuery que muestra un calendario, en el que mostrar eventos.

Anexo 4. Capturas de pantalla

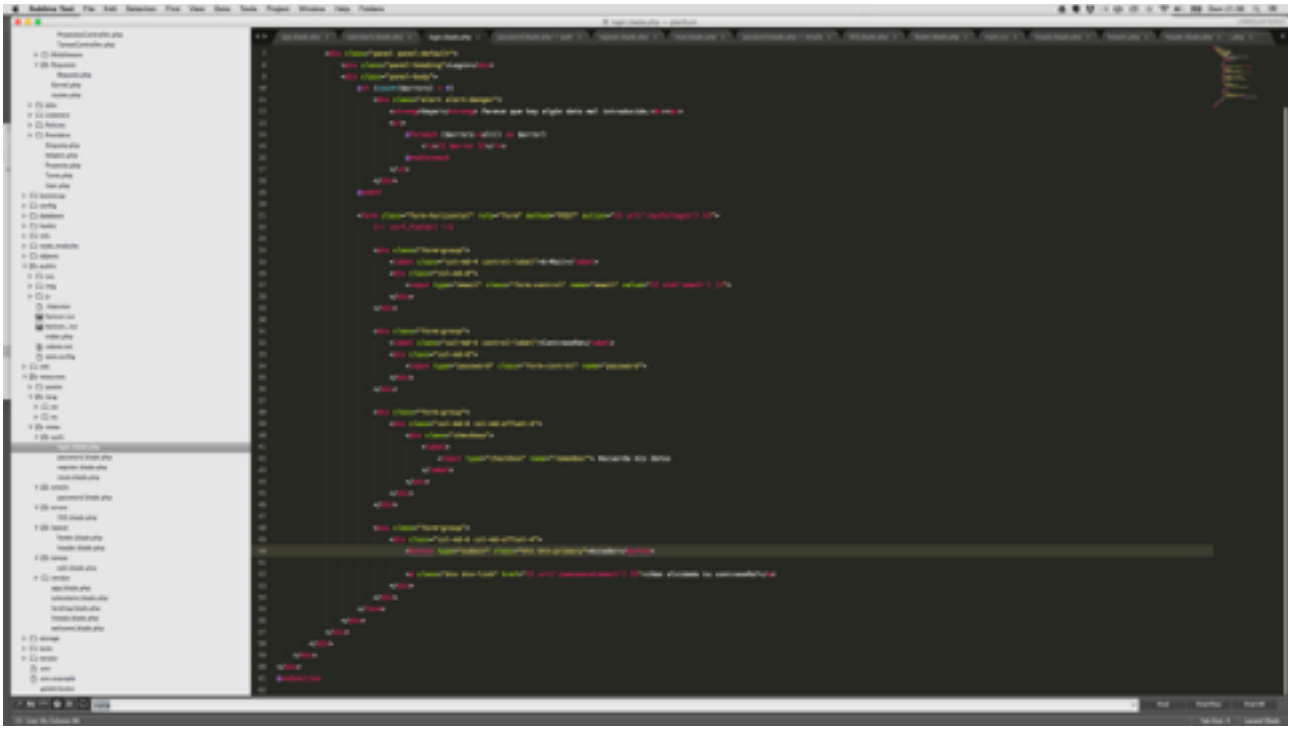


Figura : Captura durante desarrollo de Blade

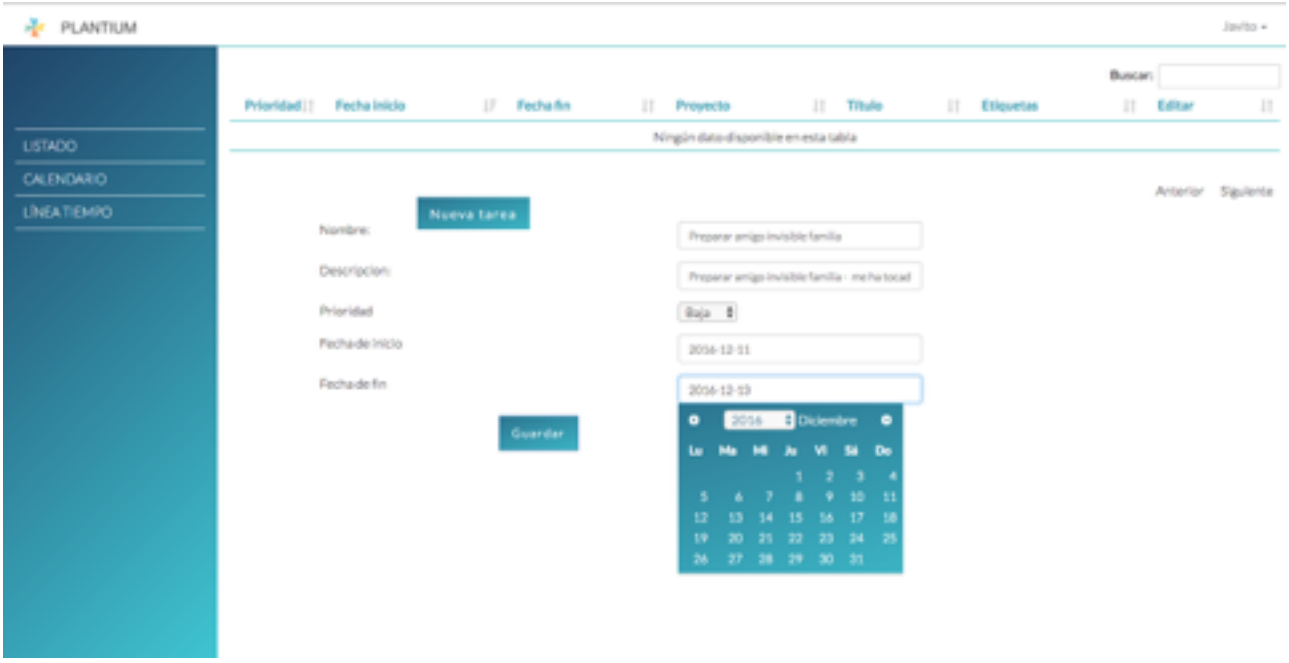


Figura : Captura durante desarrollo de apartado creación de tareas

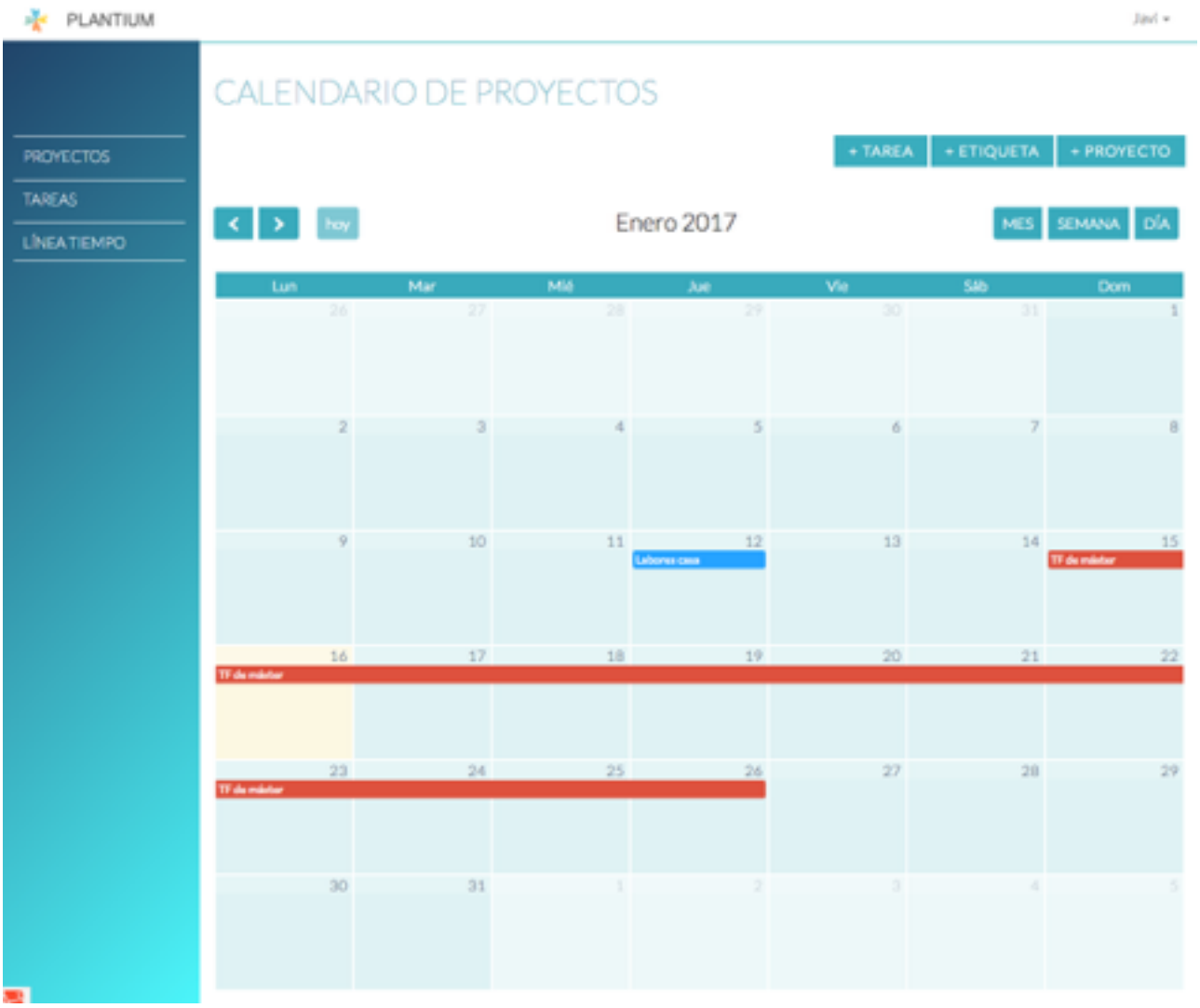
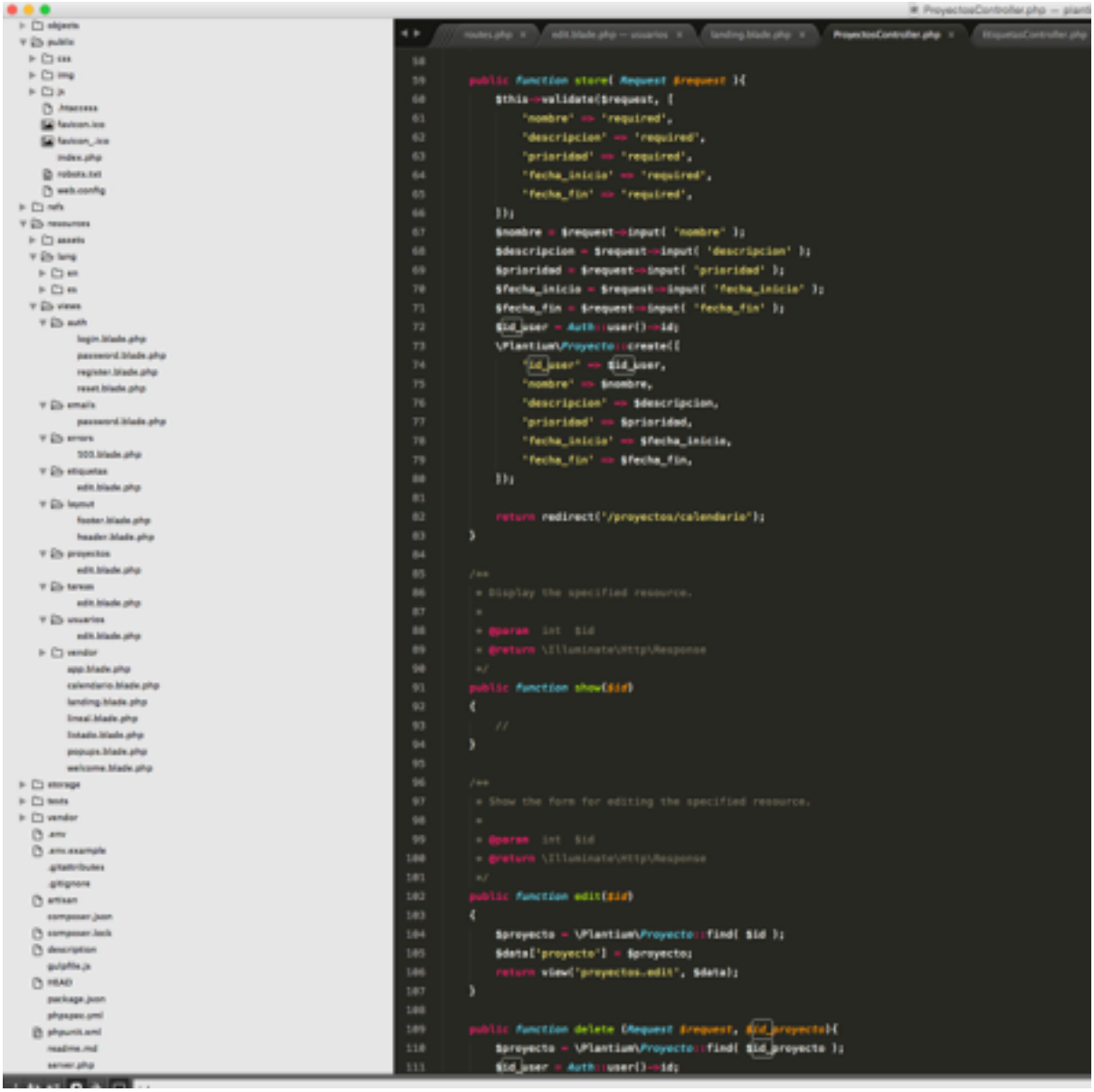


Figura : Captura durante desarrollo de apartado calendario



```
    public function store( Request $request ){
        $this->validate($request, [
            'nombre' => 'required',
            'descripcion' => 'required',
            'prioridad' => 'required',
            'fecha_inicio' => 'required',
            'fecha_fin' => 'required',
        ]);
        $nombre = $request->input( 'nombre' );
        $descripcion = $request->input( 'descripcion' );
        $prioridad = $request->input( 'prioridad' );
        $fecha_inicio = $request->input( 'fecha_inicio' );
        $fecha_fin = $request->input( 'fecha_fin' );
        $id_user = Auth::user()->id;
        \Plantium\Proyecto::create([
            'id_user' => $id_user,
            'nombre' => $nombre,
            'descripcion' => $descripcion,
            'prioridad' => $prioridad,
            'fecha_inicio' => $fecha_inicio,
            'fecha_fin' => $fecha_fin,
        ]);

        return redirect('/proyectos/calendario');
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        //
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        $proyecto = \Plantium\Proyecto::find( $id );
        $data['proyecto'] = $proyecto;
        return view('proyectos.edit', $data);
    }

    public function delete (Request $request, $id_proyecto){
        $proyecto = \Plantium\Proyecto::find( $id_proyecto );
        $id_user = Auth::user()->id;
```

Figura : Captura durante desarrollo de Controlador de proyectos

Anexo 5. Libro de estilo

5.1 Naming

El nombre Plantium es una *latinización* de la palabra 'planta'. Esta palabra acompañada de su anagrama está ligada a las plantas y árboles. Un árbol morfológicamente tiene una estructura por la que a partir de sus raíces y por medio de su tronco y ramas, termina por dar frutos. Es en sí un símbolo de un proceso

Además el árbol y toda la terminología que acompaña está muy ligada a la elaboración de procesos ya que metafóricamente su icono se presta a representar dicha acción:

Raíz (inicio)
Tronco (medio principal)
Ramas (caminos de procesos)
Nudos (nodos, bifurcaciones)
Frutos (objetivos)

Es por esto que una planta podría simbolizar un proceso para obtener un fruto por medio de un proceso.

5.2 Logotipos y anagramas.

La primera versión del logotipo de la aplicación es la siguiente:

Figura : Logotipo provisional de Plantium

El anagrama pretende asemejarse a un árbol de procesos, asemejándose a la estructura de un árbol. Sin embargo el resultado no fue muy acertado, estéticamente hablando.

Tras varias vueltas el logotipo final mantiene una estructura de ramas y nodos, y se le da una composición más estable así como se escoge una paleta de colores más viva. También se altera la tipografía por otra sans-serif mas fina y con esquinados curvos: Hiragino Maru Gothic ProN. El resultado es más armonioso, más moderno y menos aburrido:

Figura : Logotipo final

5.3 Paleta de colores.

Los colores empleados en la interfaz son los siguientes:

Gradiente con el siguiente espectro:

#55fdff **#214368**

En algunos elementos como pastillas de fondo se aplica una inclinación de -45°

Figura : Recorte calendario picker con fondo degradado -45ª

Para elementos como backgroud de botones se emplea la siguiente gama:

#3eacbd
#81c7d3
#dff2f4
#edf8f9

5.4 Tipografía:

Las tipografías empleadas son las siguientes:

- Tipografía Lato: Para toda la plataforma

(https://fonts.google.com/specimen/Lato)

- Hiragino Maru Gothic ProN: Para el logotipo

Figura : Logotipo con fuente Hiragino Maru

Anexo 6. Puesta en marcha del Framework

Puesta en marcha del servidor local

El servidor Apache, la base de datos, y el código fuente para el desarrollo irá montado bajo una maquina Macbook PRO Para el desarrollo de la aplicación se emplea un sistema de desarrollo en local. Para ello previamente configuraremos el servidor local:

Fichero HOSTS para configurar el dominio local: **plantium.local**
Para ello accedemos al fichero hosts del Mac correspondiente y añadiremos el dominio:
sudo nano /private/etc/hosts

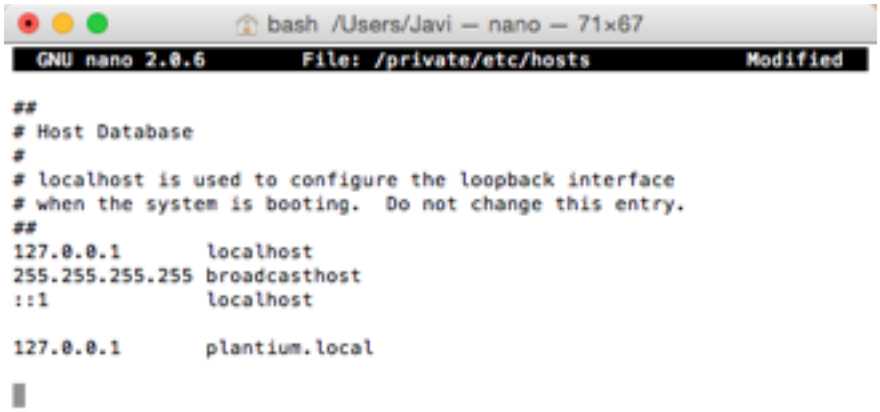


Figura : Captura alterando fichero hosts en local

De este modo, podremos acceder a la aplicación en local directamente desde el navegador accediendo al dominio “plantium.local” y diferenciarlo del dominio en producción.

Además de esto debemos preparar al servidor Apache para hacer lo mismo. Contamos con un servidor Apache con el módulo de PHP habilitado y una base de datos MySQL, todo ello instalado por medio de la aplicación XAMPP (https://www.apachefriends.org) instalado en la máquina local.

Incluímos el siguiente código de configuración en el Apache para completar la configuración.
Archivo: etc/extra/httpd-vhosts.conf (también podemos realizar este cambio en el fichero: httpd.conf)

```
<VirtualHost plantium.local:80>
  ServerAdmin admin@example.local
  DocumentRoot "/Applications/XAMPP/xamppfiles/htdocs/plantium/public"
  ServerName plantium.local
```

ServerAlias *.plantium.local
</VirtualHost>

Se abre una ventana de consola y comienzo la configuración del framework en local:
Instalación de Composer como gestor de dependencias para proyectos php.

Instalación de Laravel desde Composer.

composer global require “laravel/installer”

Creación de un nuevo proyecto:

composer create-project laravel/laravel plantium “5.1.*”

Figura : Captura instalación Laravel por consola

Creo una base de datos desde el gestor Sequel PRO con el nombre: **plantium**
Accedo al archivo oculto de configuración de entorno: .env e introduzco la configuración local:

APP_ENV=local
APP_DEBUG=true
APP_KEY=iasiudhias87ahdi37twi3i6FIUD65FDU
APP_URL=http://localhost

DB_CONNECTION=mysql
DB_HOST=localhost
DB_PORT=3306
DB_DATABASE=plantium
DB_USERNAME=root
DB_PASSWORD=

CACHE_DRIVER=file
SESSION_DRIVER=file
QUEUE_DRIVER=sync

MAIL_DRIVER=smtp
MAIL_HOST=smtp.sendgrid.net
MAIL_PORT=587
MAIL_USERNAME=plantium
MAIL_PASSWORD=plantium_nfdusdf6gv24
MAIL_ENCRYPTION=null

Soluciono errores de permisos por consola:

rm -rf vendor
chmod -R 777 storage
composer install

También experimento complicaciones no documentadas, como un error http 500 sin explicación alguna que resuelvo cambiando los permisos de laravel log:

chmod 777 storage/logs/laravel.log

Creo la clave de la App, para generar entropía:

php artisan key:generate

Instalo Gulp, como gestor de dependencias de Javascript. Gulp se encarga de varias tareas, como aunar todas las dependencias en un mismo JSON y "compilarlo" en uno sólo. También podrá minimizar el javascript en un sólo archivo que recoja todas las dependencias. Lo instalo mediante **npm** que tengo previamente instalado:

npm install gulp

Gulp da un error:

```
module.js:340
throw err;
^
Error: Cannot find module 'laravel-elixir'
  at Function.Module._resolveFilename (module.js:338:15)
            ...
```

Lo resuelvo cambiando el package.json por:

```
{
 "private": true,
 "devDependencies": {
  "laravel-elixir": "*"
 }
}
```

El proyecto ya está en marcha:

Figura : Captura instalación Laravel

Para la gestión de usuarios usaremos el paquete de Laravel:

bestmomo/scafold

Incluyo las dependencias para Composer en composer.json

```
{
  "name": "laravel/laravel",
  "description": "The Laravel Framework.",
  "keywords": ["framework", "laravel"],
  "license": "MIT",
  "type": "project",
  "require": {
    "php": ">=5.5.9",
    "laravel/framework": "5.1.*",
    "illuminate/html": "~5.0",
    "bestmomo/scafold": "dev-master",
    "barryvdh/laravel-debugbar": "^2.0",
    "uxweb/sweet-alert": "~1.1",
    "vinkla/hashids": "^2.3",
    "mews/captcha": "^2.1"
  },
```

```
  "require-dev": {
    "fzaninotto/faker": "~1.4",
    "mockery/mockery": "0.9.*",
    "phpunit/phpunit": "~4.0",
    "phpspec/phpspec": "~2.1"
  },
...
```

Tras esto ejecuto los comandos para traer las librerías y desplagarlas:

```
php artisan vendor:publish
composer update
```

Activo el modo de desarrollo en config/app.php

```
  'debug' => env('APP_DEBUG', true),
```

En el directorio del proyecto:

```
git init
git remote add origin https://bigbeef@bitbucket.org/bigbeef/plantium.git
git push -u origin --all # pushes up the repo and its refs for the first time
git push origin --tags # pushes up any tags
```

Anexo 7. Glosario/Índice analítico

Framework de desarrollo - marco de trabajo, conjunto de herramientas, aplicaciones, librerías empleadas para el desarrollo de software

Anexo 9. Bibliografía

[1] Página framework Laravel . Último acceso: 01/12/2016. :
https://laravel.com/

[2] Página instalación Laravel . Último acceso: 01/12/2016. :
https://laravel.com/docs/5.1/

[3] Página jQuery . Último acceso: 12/11/2016. :
https://jquery.com/

[4] Página FullCalendar . Último acceso: 15/11/2016. :
https://fullcalendar.io/

[5] Página DataTables . Último acceso: 29/12/2016. :
https://datatables.net/

[6] Página SublimeText . Último acceso: 25/112/2016. :
https://www.sublimetext.com/

Anexo 10. Vita

Mi nombre es Javier Searle y vivo en un pueblo de la comunidad de Madrid. Nací en 1981, por lo que en el momento de escribir esta memoria tengo 35 años. Comencé esta carrera hace mas de 15 años, cuando aun se llamaba Graduado Multimedia, y mas o menos desde esa fecha he compaginado el estudio con mi trabajo como desarrollador web. También me he tomado algunos años de pausa para estudiar otra carrera: Ingeniería Técnica de Informática de gestión en la Universidad de Valladolid, pero tras dos años volví al Graduado y al trabajo a tiempo a jornada completa.

He trabajado cerca de 10 años por cuenta ajena, siempre en el ámbito multimedia: desarrollo web, animación, programación de Rich media, diseño web, diseño de interfaz… Y desde hace ya 5 años trabajo por mi cuenta con proyectos de toda clase: desarrollo de plataformas web a medida, Wordpress, aplicaciones móviles para Android, desarrollo de aplicaciones musicales, desarrollo de redes sociales desde cero…

Hasta la fecha procuro compaginar la vida social, los viajes, el deporte y la música, con el trabajo/estudio que tantas horas me mantiene sentado delante de una pantalla y que me obliga a llevar una vida algo más sedentaria de lo que desearía.