



**PROYECTO FIN DE
CICLO - DAM**

Gestión integral: Envío de paquetería.

**Fase 4: Seguimiento y control
Rolando Caldas Sánchez**

Índice

Índice de contenido

Identificación de las necesidades del proyecto.....	5
Contextualización.....	5
Justificación.....	6
Otros aspectos.....	7
Diseño del proyecto.....	9
Contenido.....	9
Objetivos y recursos.....	10
Viabilidad económica.....	10
Costes capital humano.....	10
Coste de equipo.....	11
Servidores web.....	12
Presupuesto anual.....	12
Financiación.....	12
Modelo de solución.....	13
Desarrollo API.....	13
Desarrollo WebAPP.....	23
Desarrollo App Móvil.....	26
Ejecución del proyecto.....	28
Contenido.....	28
Tareas dependientes y recursos necesarios.....	28
Permisos y autorizaciones necesarias.....	31
Riesgos.....	31
Revisión de recursos.....	32
Documentación de ejecución.....	32
Instalación y configuración de aplicación API.....	32
Instalación y configuración de WebAPP.....	36
Instalación y configuración de nuestra app móvil.....	40
Desarrollo realizado en API.....	41
Desarrollo realizado en webApp.....	41
Desarrollo realizado en movil App.....	42
Puesta en marcha de aplicación API.....	43
Puesta en marcha de aplicación web.....	44

Puesta en marcha de aplicación web.....	45
Documentación de código.....	45
Código fuente.....	45
Capturas API.....	46
Capturas webApp.....	50
Capturas aplicación móvil.....	53
Manual de usuario.....	54
Seguimiento y control.....	55
Valoración del proyecto.....	55
Incidencias.....	56
Cambios.....	57
Pruebas y soporte.....	58
Pruebas unitarias.....	58
Pruebas de carga.....	61
Pruebas de interfaz.....	64
Acuerdo de servicio.....	67

Gestión integral: Envío de paquetería - DAM

1. Identificación de las necesidades del proyecto

1.1 Contextualización

El proyecto a realizar se enmarca dentro de las necesidades específicas del sector de mensajería urgente. La realización del mismo nace como fruto de la necesidad interna de una empresa de mensajería de optimizar y actualizar la gestión de sus recursos.

Tras observar el funcionamiento operativo de otras empresas del sector como MRW, Seur o Correos se detectaron múltiples mejoras a realizar en el servicio y carencias que los servicios de las empresas de referencia tienen y pueden ser solucionados.

El objetivo que cubre el proyecto a realizar es poder llevar con éxito el paso de una gestión en papel del proceso de mensajería a una gestión digital y en tiempo real. Con esta demanda de mejora, se busca mejorar tanto los tiempos de la gestión como el servicio post-venta ofrecido a los clientes, pudiendo darles información en tiempo real del estado de su encargo.

La empresa analizó la situación y la conveniencia o no de solicitar a una empresa del sector informático la realización del proyecto. Esta opción fue descartada, tomando la decisión estratégica de incorporar un departamento de desarrollo como parte activa de la organización, para la realización de la solución y su mantenimiento futuro.

Para la toma de las decisiones estratégicas se ha realizado un pequeño estudio de mercado intentando localizar las soluciones existentes en el mercado, ya sea para realizar una implantación directa o una mejora sobre un sistema ya existente. Durante la búsqueda de soluciones se tomó en consideración valorar las siguientes soluciones:

- **Paquetería.One**
- **Gestión5 SQL: Paquetería y Logística**
- **Recawin**
- **JR Gestión de paquetería**

Las soluciones analizadas tuvieron que ser descartadas debido a la inconveniencia de ser implementadas. Actualmente, los servicios de mensajería estándar buscan la unificación de tarifas y servicio, de forma que se aplica una tarifa por bulto a sus clientes, en lugar de precios diferentes en función de tamaños y pesos. Siguiendo esta tendencia, la empresa de mensajería tiene unos pesos y tamaños máximos por bulto, de manera que si un paquete excede esos tamaños no puede ser entregada por la empresa. Del mismo modo, el precio por paquete es único, para conseguir ser más eficientes y rápidos en el servicio de atención al cliente y para simplificar el proceso de contratación del servicio.

Esta necesidad choca frontalmente con las soluciones estudiadas, puesto que éstas designan gran parte de los recursos de la solución a la gestión de tarifas.

Otra carencia de las soluciones analizadas es la falta de una solución multiplataforma y de movilidad, que permita a los transportistas informar en tiempo real y de forma clara, el estado de los pedidos. Además, las soluciones se basan en entorno de escritorio y Windows, algo que limita el cambio de software en la empresa de mensajería. Aunque esto podría interpretarse como requisitos de la solución y no como un problema, los problemas existentes con la privacidad de la información en la última versión del sistema operativo Windows, hace que la empresa no quiera una solución que fuere una plataforma concreta.

Además de estas soluciones, se descartó intentar implementar las soluciones de otras empresas líderes del sector como MRW o Seur debido a que la solución de movilidad está supeditada a unos dispositivos concretos, algo que se pretende evitar puesto que se quiere tener la libertad de poder actualizar los dispositivos de forma sencilla.

1.2 Justificación

Tras analizar las posibles soluciones del mercado, se toma la decisión estratégica de desarrollar la solución de manera interna. Esta solución se soporta en los siguientes pilares:

- La solución a realizar tiene que estar pensado y basado en la movilidad.
- La solución debe conformarse por diferentes desarrollos de software que estén comunicados entre sí en tiempo real.
- La solución debe permitir eliminar todo soporte en papel para la realización del servicio.
- La solución no debe realizar labores de facturación, puesto que para ello ya existe un ERP.
- La solución debe utilizar tecnologías libres exclusivamente.

Durante el proceso de análisis del problema y la solución, surgió la necesidad de decidir si el proyecto a realizar quedaría como un desarrollo interno o si la solución debería plantearse como un software a comercializar. La decisión tomada en este sentido fue la de realizar la solución como un desarrollo interno no comercializable en primera instancia. Esta decisión se toma principalmente escuchando al departamento de marketing, quien considera que el software a desarrollar puede ser utilizado para obtener una ventaja sobre la competencia, utilizándola como una herramienta diferenciadora de venta del servicio. A pesar de ello, como la competencia es intensa y las mejoras son continuas, una vez la solución esté implementada y sea un punto diferenciador del servicio, se creará una marca comercial para el software, para poder comercializarlo como producto de software a empresas del sector que trabajen en provincias diferentes a la de la empresa de transporte.

1.3 Otros aspectos

Como el desarrollo va a realizarse dentro de la empresa de transporte, no se creará ninguna empresa para tal fin. Lo que sí se procederá es a crear un departamento de desarrollo conformado por cuatro desarrolladores de diferentes perfiles:

- 1 Jefe de proyecto
- 1 Analista programador
- 1 Programador backend senior
- 1 Programador frontend junior

Para poder formar el departamento, la empresa modificará sus datos de actividad económica incorporando la de “Actividades de consultoría informática”, estando los cuatro desarrolladores contratados bajo el convenio de empresas de consultoría y estudio de mercados y la opinión. Esta decisión se toma debido a que no sólo se realizará el desarrollo de software sino que para poder realizar el desarrollo tendrán que ejercer también como consultores informáticos, especialmente cuando el producto llegue a su fase de comercialización.

Para el desarrollo del proyecto se solicitarán ayudas públicas para proyectos de I+D+i a través del IGAPE (Instituto Gallego de Promoción Económica) y los contratos a los desarrolladores serán indefinidos celebrados bajo la cláusula específica de apoyo a los emprendedores.

Los nuevos desarrolladores pasarán a trabajar en la nave de la empresa de transportes, habilitando para ello una sala en la nave, dotada de ventilación, calefacción, servicio propio y dispensador de agua.

Se comprarán cuatro mesas de trabajo, con sus correspondientes sillas, todas con un plus de ergonomía, especialmente importante debido a la cantidad de horas que los nuevos empleados pasarán en el trabajo.

La empresa comprará nuevos equipos informáticos, uno para cada empleado, con las siguientes características:

- Procesador i7-4790K 4.0Ghz
- Memoria G.Skill Trident X DDR3 2400 PC3-19200 32GB 4x8GB CL10
- Disco duro Samsung 850 Pro SSD Series 1TB
- GeForce GTX 980 4GB DDR3
- Teclado y ratón Microsoft Sculpt Ergonomic Desktop
- Monitor con tecnología Anti-Blue Light y Flicker Free

La motivación de las características del equipo es doble: Por un lado dotar al empleado de los recursos necesarios para hacer funcionar con fluidez las herramientas de desarrollo y, por otro, incorporar soluciones ergonómicas teniendo en cuenta los riesgos laborales.

Para llevar a cabo el desarrollo del proyecto, se cuenta con un guión base sobre el cual poder trabajar:

1. Seleccionar los profesionales encargados de desarrollar la solución.
2. Con el equipo formado, analizar el proyecto con el jefe de proyecto y el analista programador, para realizar los ajustes necesarios.
3. Establecer el cronograma de trabajo, desde la fase de análisis.
4. Realizar el análisis y diseño del proyecto.
5. Iniciar la ejecución de proyecto.
6. Fase de pruebas.
7. Presentación del software: demo.
8. Implementar el software con dos transportistas: Experiencia piloto.
9. Realizar las correcciones necesarias tras el resultado de la experiencia piloto.
10. Implementación de la solución a gran escala.

El presente documento plasmará las fases de diseño, ejecución y seguimiento y control, quedando el resto fuera del alcance del presente proyecto.

2. Diseño del proyecto

2.1 Contenido

Teniendo claro el proyecto que quiere realizarse debemos comprobar si es viable o no desde el punto de vista técnico.

Actualmente los sistemas de información han evolucionado rápidamente hacia una interconexión global. Además, la red móvil tiene una cobertura 3G casi absoluta en las grandes ciudades. Esto hace que, la solución objetivo, pueda ser puesta en funcionamiento.

La existencia de lenguajes de alto nivel y de profesionales con dilatada experiencia en dichos lenguajes hace que desde el punto de vista humano no exista problemas en poder realizar el proyecto. El mayor problema se encontraría en la parte de movilidad del proyecto, pudiendo llegar a ser un problema crítico e insalvable en el supuesto de que la aplicación móvil tuviese que desarrollarse para varias plataformas.

Esto hace que se tenga que analizar con cuidado el desarrollo de la aplicación móvil. Afortunadamente existen varios proyectos de software que pretenden llevar el concepto de multiplataforma a los dispositivos móviles tales como Cordova, PhoneGap o Appcelerator entre otros.

Estas soluciones son una puerta de esperanza que hacen viables múltiples proyectos. En el caso que nos ocupa, se valora utilizar Cordova para poder desarrollar una aplicación híbrida que pueda ser multiplataforma de manera extremadamente sencilla.

Es preciso comprobar si la aplicación puede desarrollarse en HTML5 y con Cordova. Para ello se tienen que cotejar las funcionalidades deseadas. Las funcionalidades que deben poder cubrirse son:

- Escaneo de códigos QR
- Intercambio de datos vía API
- Geolocalización
- Acceso a mapas
- Notificaciones Push

Aunque actualmente sólo se contempla la funcionalidad de escaneo de códigos QR, el resto de funcionalidades deben ser tenidas en cuenta para una implementación futura en caso de éxito, por lo que la aplicación desarrollada hoy tiene que ser capaz de incorporar todas las funcionalidades indicadas.

Cordova posee plugins que permiten realizar todas las funcionalidades indicadas sin problema.

Por lo tanto, se considera viable llevar a cabo el proyecto.

2.2 Objetivos y recursos

Los objetivos del proyecto son sencillos y claros: Se busca aumentar el activo de la

Los objetivos del proyecto son sencillos y claros: Se busca aumentar el activo de la empresa aportando una solución software que permita, a largo plazo, agilizar a nivel provincial, autonómico e incluso nacional la gestión de los envíos de paquetería.

Como el desarrollo de un aplicación de estas características es largo y repleto de opciones y posibilidades, se centran los esfuerzos en crear una solución que sirva para lanzar una experiencia piloto a nivel local.

Respecto a los recursos, en el apartado 1.3 del presente documento ya se detallan los recursos tanto de software como de personal; por lo que en este apartado contemplaremos los recursos software:

Todo el software a utilizar en el proyecto tiene que ser software libre, abierto o gratuito:

- **Sistema operativo:** Kubuntu
- **Sistema operativo servidores:** Debian
- **IDE:** Netbeans (Para la webapp y app móvil) y LiClipse (API)
- **Navegador Web:** Chromium
- **Máquina Virtual:** Vagrant + VirtualBox
- **Diagramas:** Dia
- **Suite ofimática:** LibreOffice 5

2.3 Viabilidad económica

Como el proyecto es para uso interno y aumento del activo de la empresa, no se cuenta con unas limitaciones claras de presupuesto como en el caso de que su objetivo fuese ser comercializado y lograr rentabilidad económica.

Sin embargo, es preciso limitar económicamente el proyecto tanto para no descuadrar las cuentas de la empresa como para evitar el derroche de recursos.

En base a lo especificado en el primer epígrafe del presente documento, podemos realizar el siguiente presupuesto económico, planteando un ciclo de desarrollo de un año.

Costes capital humano

- 1 Jefe de proyecto: 30.000 EUR
- 1 Analista programador: 28.000 EUR
- 1 Programador backend senior: 24.000 EUR
- 1 Programador frontend junior 15.000 EUR

El coste anual en personal ascendería a 97.000 EUR. Este coste se considera

excesivamente elevado y de difícil justificación para un proyecto interno. Se decide revisar a la baja esta partida, eliminando parte del equipo que se desea incorporar.

Debido a que desde la empresa se ve factible crear una experiencia piloto con una funcionalidad limitada y, gradualmente, ir complementando la solución, se decide unificar la figura de Jefe de proyecto y Analista programador, junto con la substitución del programador junior por un becario a través del FEUGA quedando el coste de capital humano de la siguiente manera:

- 1 Jefe de proyecto / AP: 30.000 EUR
- 1 Programador backend senior: 24.000 EUR
- 1 Becario FEUGA: 4.000 EUR

Quedando el coste de capital humano en 58000 EUR. Este coste no incluye las cuotas a la seguridad social, puesto que la empresa se acoge durante el primer año a una subvención sobre la totalidad de las cuotas a la seguridad social, al contratar a desarrolladores actualmente sin empleo.

Coste de equipo

Mi cesta	
	BenQ XL2430T 24" LED 399€ x <input type="text" value="3"/>
	G.Skill Trident X DDR3 2400 PC3-19200 32GB 4x8GB CL10 240€ x <input type="text" value="3"/>
	Gigabyte GeForce GTX 980 Gaming G1 WindForce OC 4GB GDDR5 569€ x <input type="text" value="3"/>
	Intel Core i7-4790K 4.0Ghz Box 316€ x <input type="text" value="3"/>
	Microsoft Sculpt Mobile Keyboard Bluetooth 15.75€ x <input type="text" value="3"/>
	Samsung 850 Pro SSD Series 1TB 470€ x <input type="text" value="3"/>
(18 productos) Total: 6029.25€	

Servidores web

Por último, aunque no hay costes asociados en cuanto a software, como se van a desarrollar dos aplicaciones del lado del servidor (API y WebApp), se considera oportuno contratar dos servidores web en uno de los principales centros de datos europeos por un año.

Se valoran las diferentes opciones y se selecciona el servidor de la gama para PYMES de Ovh (soyoustart) E3-SSD-3 que tiene un coste anual de 480 EUR/año cada uno.

Características servidores:

Procesador	Intel Xeon E3 1245v2
Cores/Threads	4 cores/ 8 threads
Frecuencia	3.4 GHz+
RAM	32GB DDR3
Discos	3x 120 GB SSD
RAID	Soft
Conexión de red	1 Gbps
Ancho de banda	250 Mbps
Tráfico	Ilimitado

Presupuesto anual

- Personal: 58.000 EUR
- Equipos: 5.000 EUR
- Servidores: 480 EUR
- Total: 63.480 EUR

Financiación

A pesar de que el presupuesto es ajustado y aceptable, como el proyecto puede encajar dentro de un proyecto de I+D+i, se solicita una ayuda al desarrollo a modo de subvención a través del IGAPPE por 50.000 EUR

2.4 Modelo de solución

La realización de la solución conlleva tener que valorar y modelizar las diferentes aplicaciones que, en su conjunto, conforman la solución. Por lo tanto, se han modelizado las siguientes aplicaciones:

1. **API:** Se va a generar una aplicación cuya única finalidad es la de servir de modelo o capa de datos al resto de aplicaciones. Para poder realizar este objetivo de forma eficiente, se va a establecer una interfaz de comunicación siguiendo los principios de una API CRUD
2. **WebApp:** Esta aplicación tiene como finalidad proporcionar una interfaz amigable al personal administrativo que les permita gestionar el proceso de envío de paquetería.
3. **App móvil:** Se va a crear una pequeña aplicación móvil que utilizarán los transportistas, de forma que desde su teléfono móvil se comuniquen en tiempo real con la aplicación API, pudiendo acceder a los envíos pendientes e informar de la entrega exitosa, o no.

Al tratarse de tres aplicaciones diferentes, analizaremos cada solución por separado, empezando por la API puesto que es de la que se sirven tanto la WebApp como la App móvil.

Desarrollo API

La aplicación que va a funcionar como API es la única que va a almacenar la información del negocio, por lo que es en esta parte de la solución en la que analizaremos el modelo de datos a través de un diagrama E/R

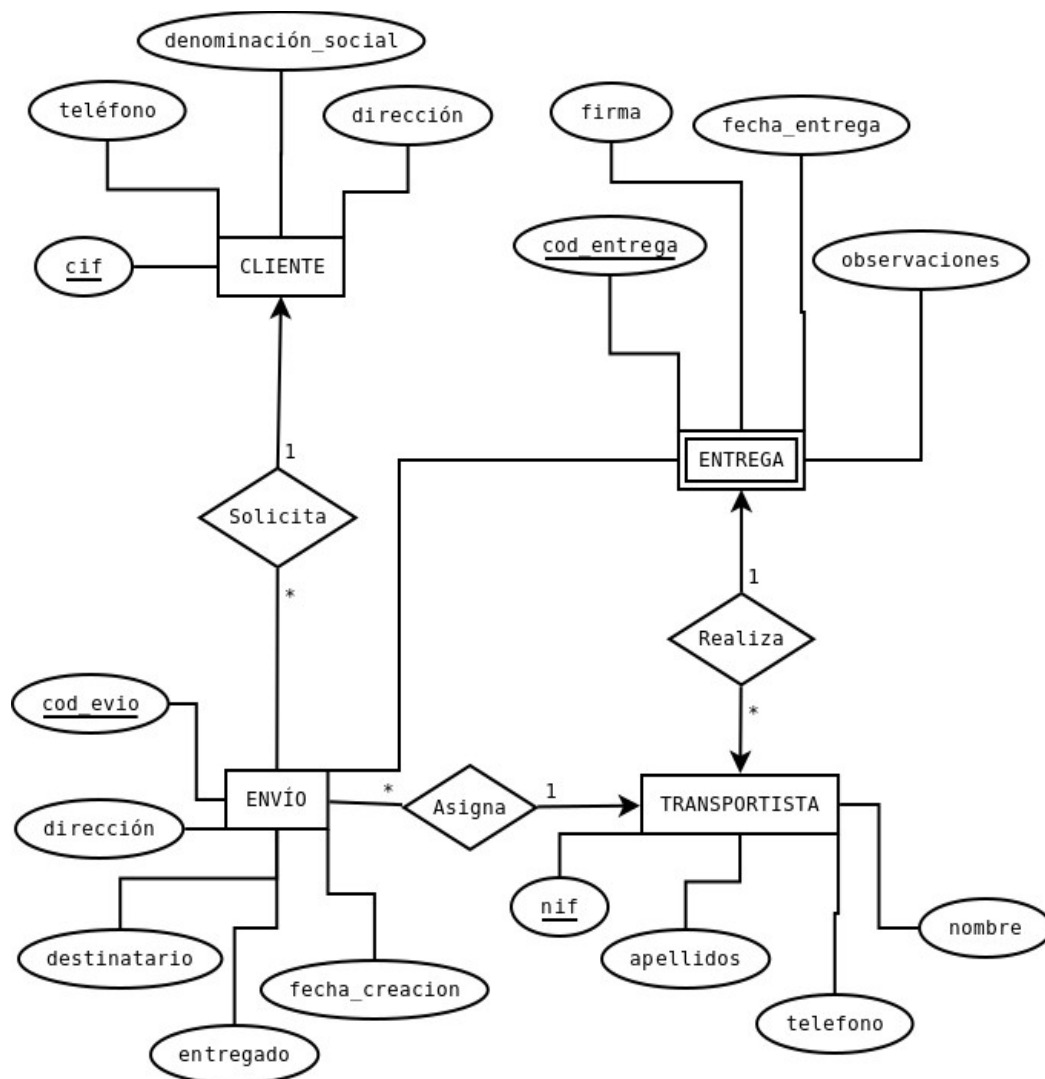
Modelo E/R

Para poder realizar el modelo de E/R se analiza la información con la que trabaja la empresa y se llega a la conclusión de que, para la experiencia piloto, la API debe permitir acceder, crear, editar y modificar la siguiente información:

- **Clientes:** Todo envío es solicitado por una empresa o autónomo, por lo que debe existir una base de datos de clientes que sea gestionable. Para agilizar el proceso se decide utilizar la información mínima necesaria, puesto que recabar más datos en la API sería duplicar información, puesto que los detalles de cada cliente estarán en la ficha gestionada por el ERP que tiene la empresa.
- **Transportistas:** Es necesario tener una base de datos de transportistas, puesto que éstos son los encargados de realizar las entregas. Son personal de la empresa.
- **Envíos:** Los clientes solicitan que se envíe un paquete a un tercero, por lo que los envíos son el eje central de la solución.
- **Entrega:** Es necesario tener constancia de que el envío se entregó correctamente o de si ha existido alguna incidencia. Se entiende entrega por el proceso de ir a entregar el paquete, por lo que puede darse el caso de que un mismo envío tenga dos entregas, la primera podría ser no exitosa (por lo que

indicaría como observaciones, por ejemplo, que el destinatario no estaba disponible) y la segunda sí (por lo que tendría la firma de recepción del destinatario).

Con las entidades y sus relaciones identificadas de forma general, incluimos el modelo E/R inicial con las entidades, sus atributos y las relaciones existentes entre las diferentes entidades:



CLIENTE (cif, denominación_social, dirección, teléfono)

ENVÍO (cod_envio, destinatario, dirección, fecha_creación, cif, nif)

TRANSPORTISTA(nif, nombre, apellidos, telefono)

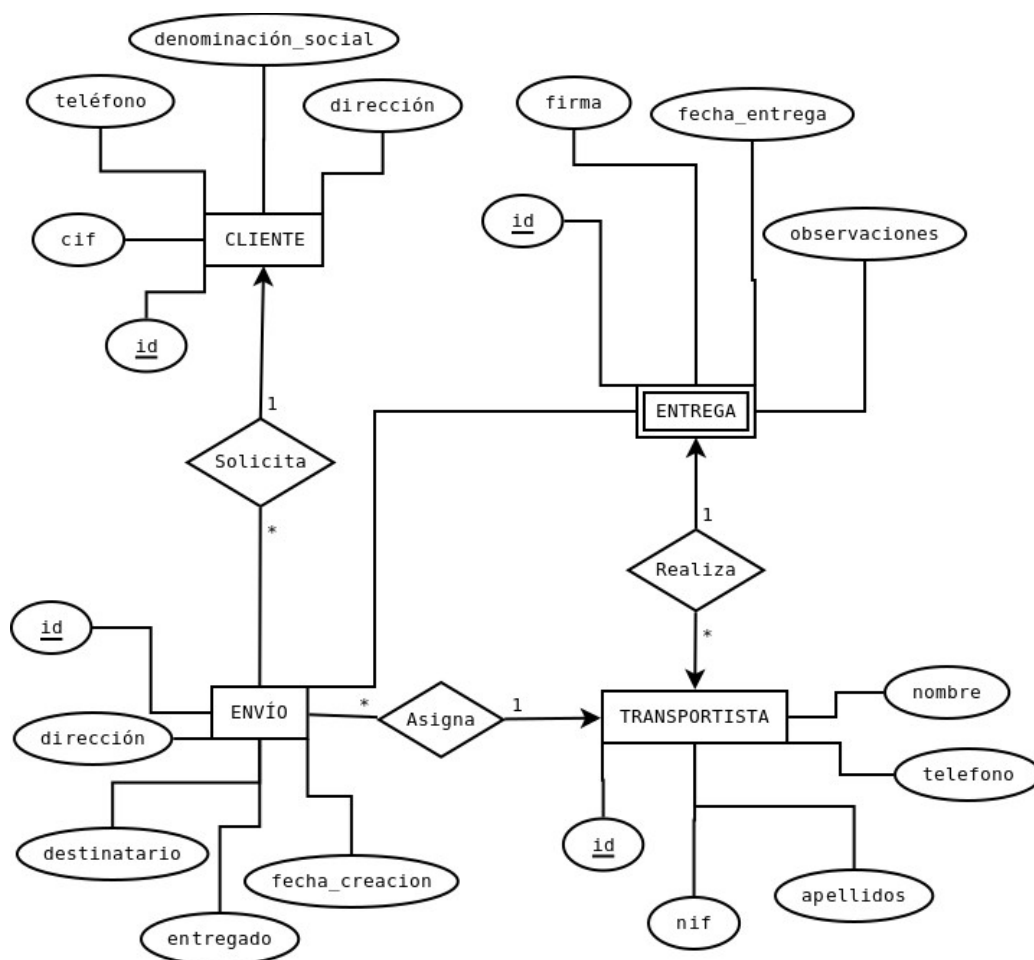
ENTREGA (cod_entrega, cod_envio, firma, fecha_entrega, observaciones, nif)

A pesar de que se considera la solución idónea, este modelo debe ser modificado puesto que se valora una premisa que prevalece sobre la solución resultante del modelo E/R.

Como el tiempo y los recursos son limitados, se tomó como decisión estratégica apoyar todo el desarrollo en diferentes frameworks consolidados que nos permitan avanzar con celeridad. Para realizar la API, se realizó una investigación sobre los diferentes frameworks que ofrecen una solución robusta para una API CRUD. La conclusión fue el framework Django, programado en Python, y en su aplicación “Django REST framework”

Django, de forma nativa, utiliza como SGBD SQLite. Una de las características de este framework es que para cada tabla crea, automáticamente, un campo id auto incremental como clave primaria de la tabla. A pesar de que es posible modificar esta conducta, al hacerlo es necesario encargarse, por código, del valor autoincremental en caso de que la clave primaria deba tener esa condición. Además, las diferentes clases sobre las cuales se asienta el ORM y Django REST Framework asumen la existencia de este campo como clave primaria.

Así pues, se toma la decisión de modificar nuestro modelo E/R para adecuarlo a las premisas que el Django proporciona.



Casos de uso

Con la estructura de la base de datos determinada, se analiza las acciones que desde la API se deben permitir llevar a cabo.

Nuestra aplicación API es una API CRUD, o sea: Create, Read, Update, Delete. Por lo tanto, nuestra API, al menos, debe permitir realizar tales acciones a las diferentes entidades.

Aunque se puede ver en el diagrama de casos de uso, debido a su extensión también los enumeramos a continuación:

- **Entidad cliente:** Obtener clientes, crear cliente, editar cliente, información cliente, borrar cliente.
- **Entidad transportista:** Obtener transportistas, crear transportista, editar transportista, información transportista, borrar transportista.
- **Entidad envío:** Obtener envíos, crear envío, editar envío, información envío, borrar envío.
- **Entidad entrega:** Obtener entregas, crear entrega, editar entrega, información entrega, borrar entrega.

Además de estos casos de uso básicos, para mejorar el rendimiento de las aplicaciones que se conectan a la API se contemplan los siguientes casos de uso:

- **Envíos del Cliente:** Envíos asociados a un cliente concreto.
- **Entregas del envío:** Entregas asociadas a un envío concreto.
- **Envíos asociados al transportista:** Todos los envíos asignados a un transportista.
- **Envíos realizados por el transportista:** Todos los envíos asociados a un transportista que fueron entregados satisfactoriamente.
- **Envíos pendientes del transportista:** Todos los envíos asociados a un transportista que todavía no han podido ser entregados.

En todos los casos de uso, se toma por actor la figura “Conector API” que no es más que el modo de denominar al software que solicita a conexión con la API.

La API está protegida, por lo que sólo con un usuario y contraseña existente en la API se puede consultar, por lo que todos los casos de uso incluyen el caso de uso de “Validar acción”.

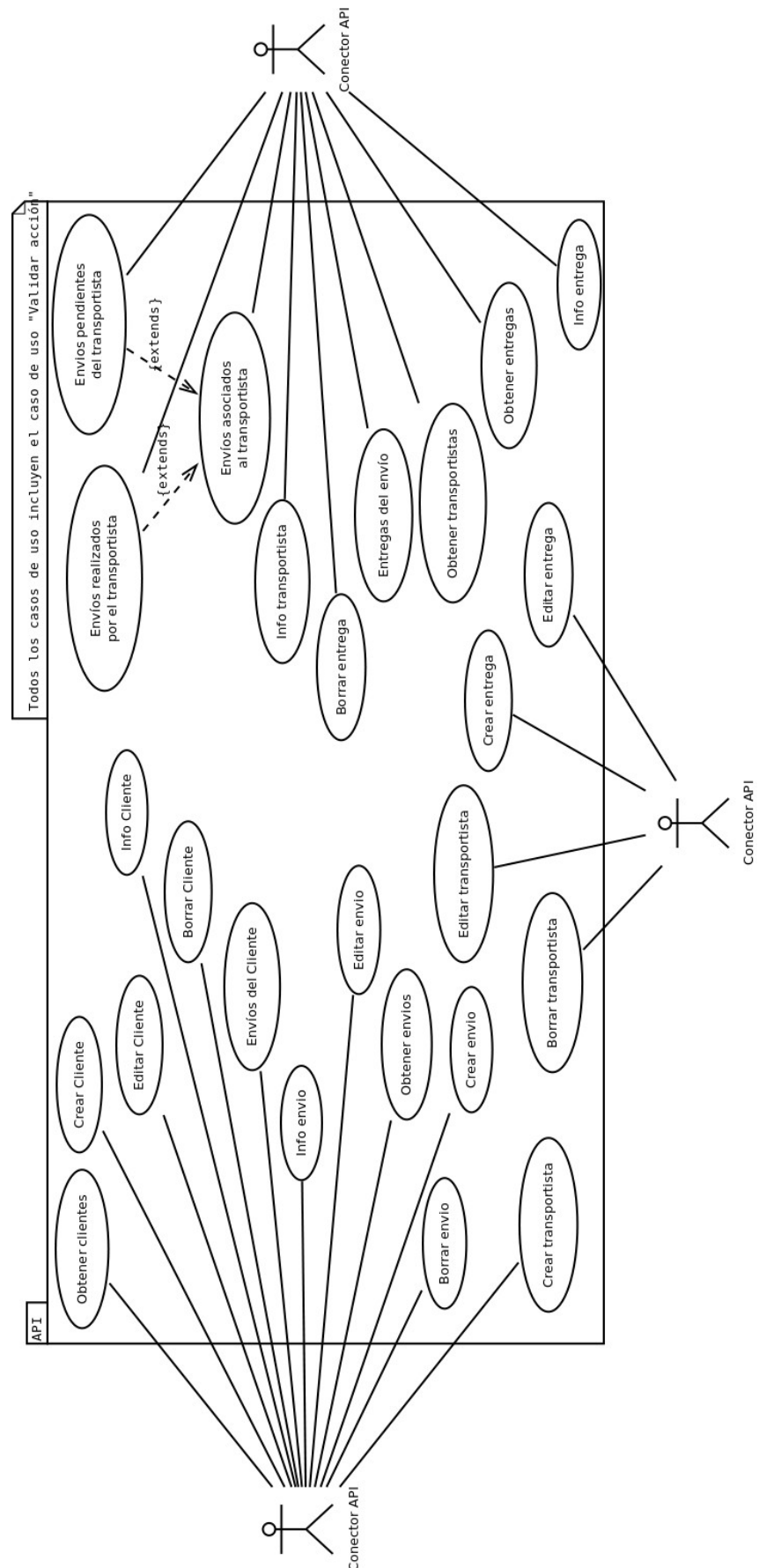


Diagrama UML

Para elaborar el diagrama UML de la API, se estudió tanto Django como su REST Framework para evitar realizar clases y código ya realizado. Django sigue la filosofía DRY (Don't Repeat Yourself). Esto es especialmente importante en el modelo de datos, puesto que Django cuenta con un potente ORM. Gracias a este ORM el código a realizar para crear un modelo de datos se limita a pocas líneas.

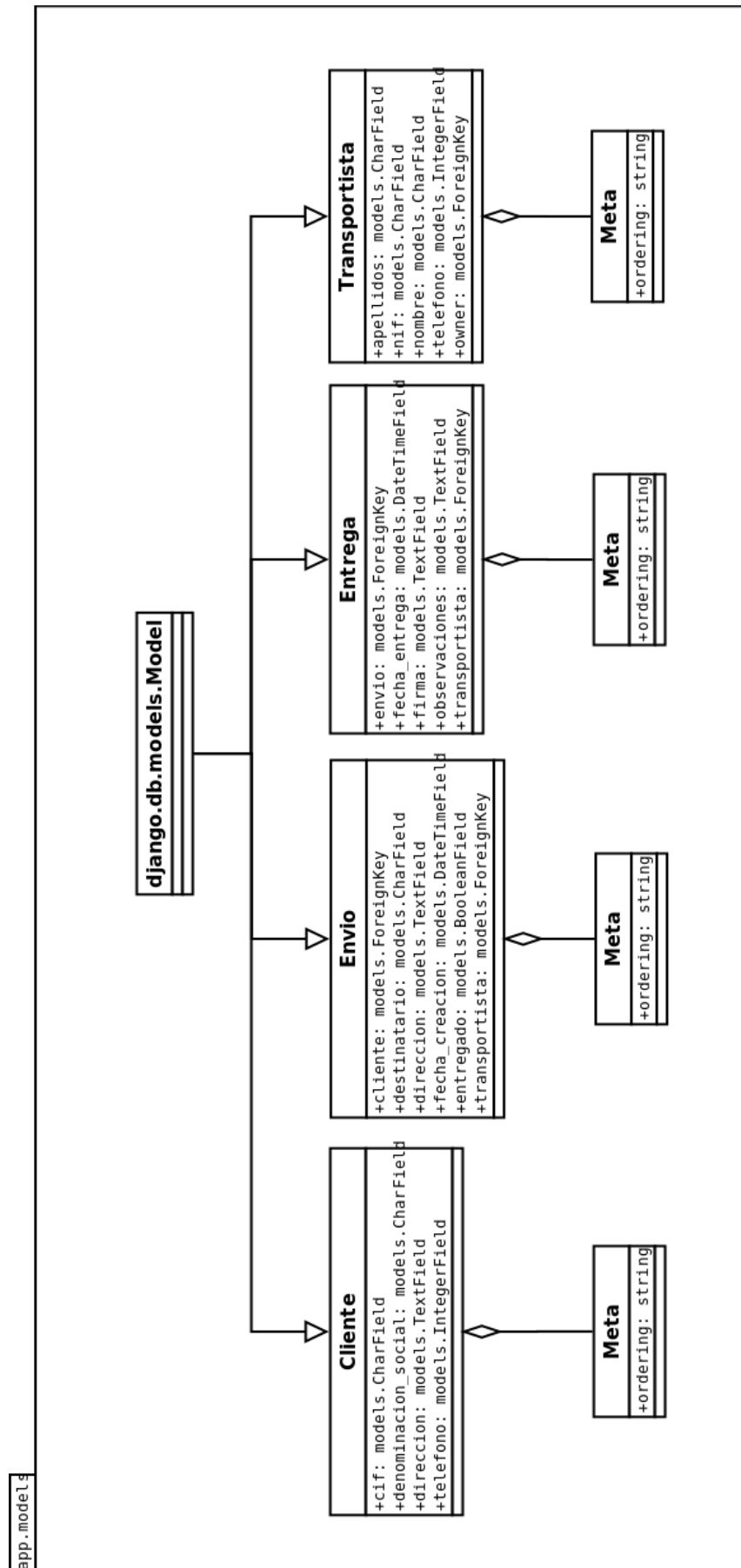
Como en el modelo de datos se almacena el NIF/CIF, es necesario codificar una clase que se encargue de validar el NIF o CIF facilitados.

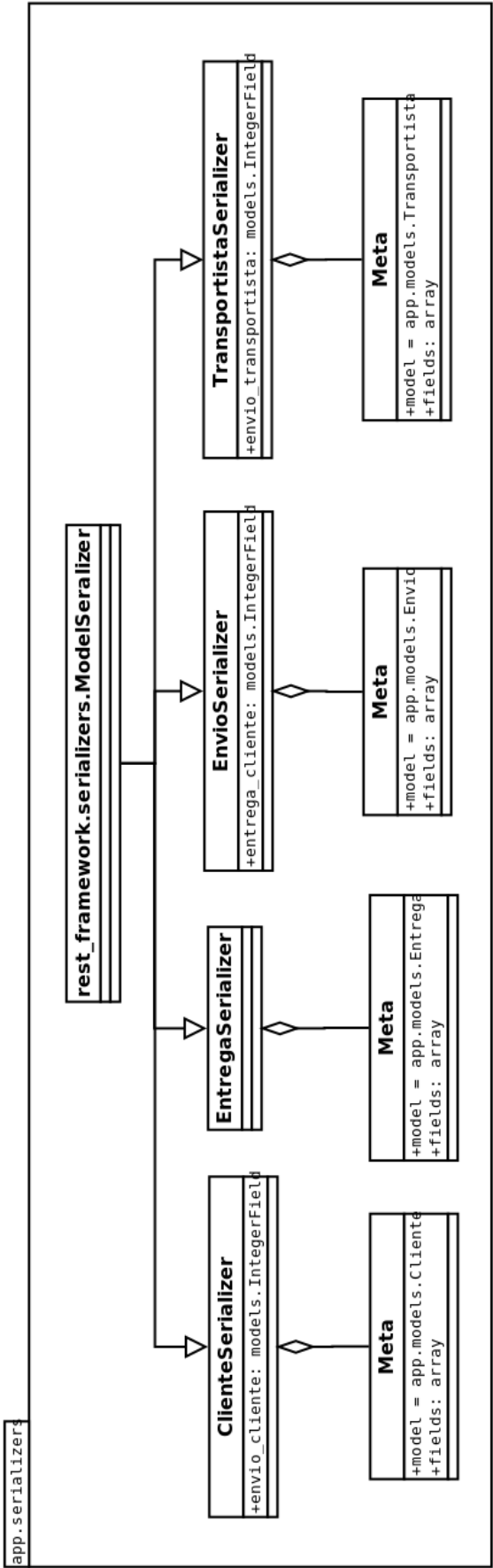
Además del modelo de datos, el diagrama UML tiene que contemplar la capa de vistas y serializers, elementos necesarios para integrar la solución en el Django Rest Framework.

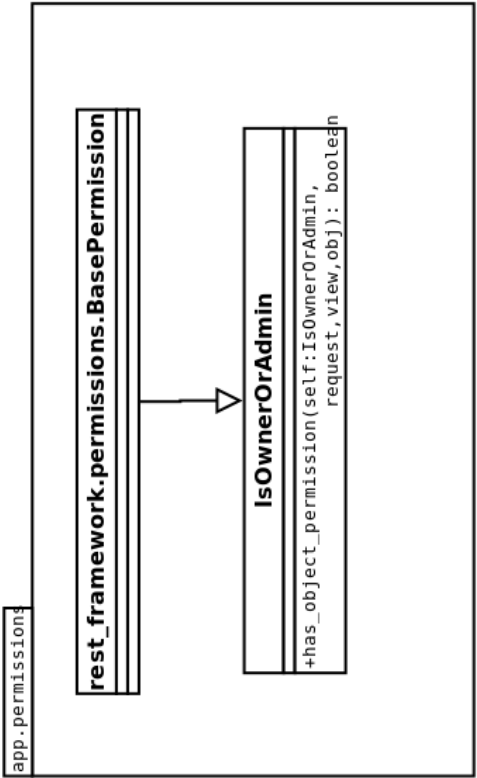
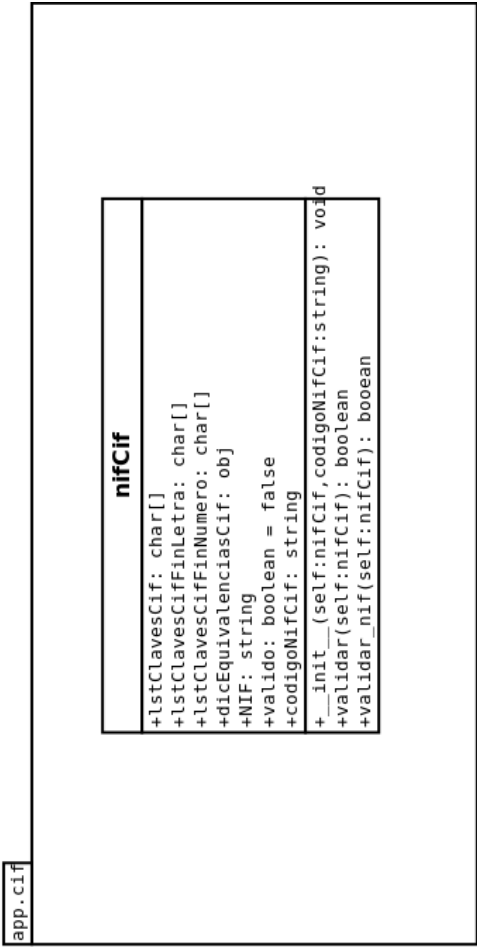
Por último, como las acciones con la API tienen que estar sujetos a una validación de permisos, se debe desarrollar una clase que realice las acción de validación.

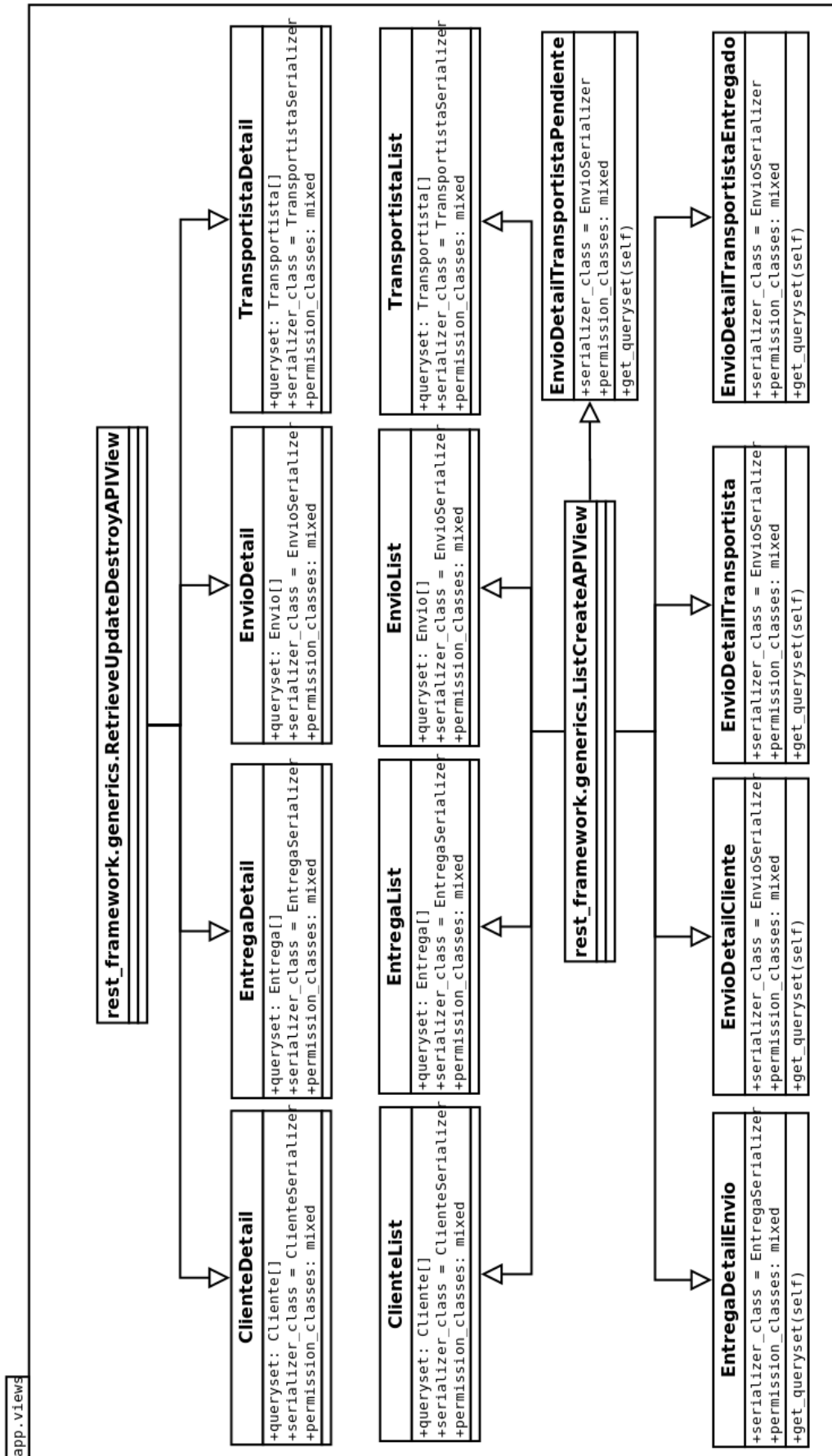
Para facilitar la lectura del diagrama UML, se han agrupado las diferentes clases a desarrollar en función de su funcionalidad:

- app.models: Las clases que conformarían el modelo de datos.
- app.serializers: Las clases que se encargan de serializar/deserializar la información a mostrar/recibir en JSON o XML
- app.views: Las clases de vistas. En Django Rest Framework las vistas no sólo son visualización de datos, sino que también contienen lógica de programación.
- app.permissions: Para validar las acciones.
- app.cif: Para validar nif/cif









Desarrollo WebAPP

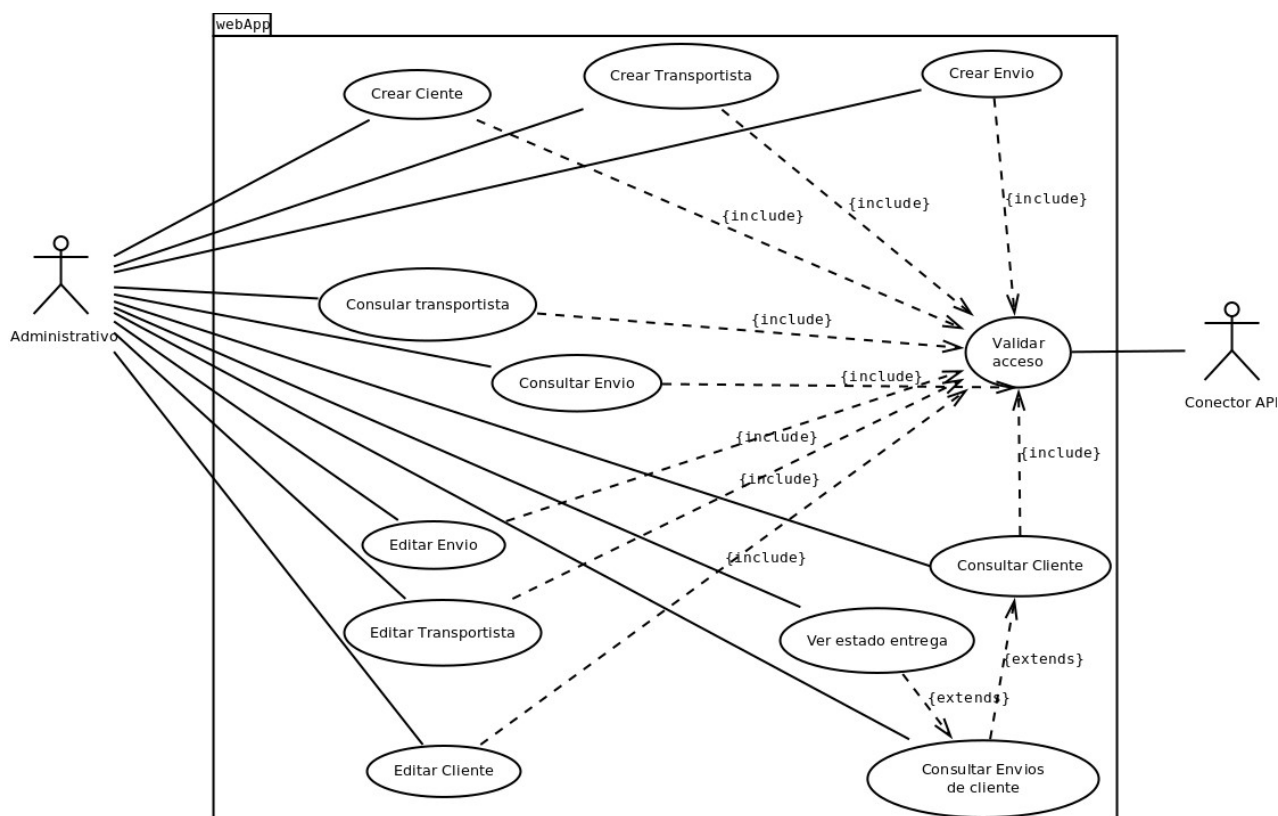
Para desarrollar la aplicación web que se utilizará desde la administración, también se ha buscado entre las diferentes soluciones web para localizar un framework en el que basar el desarrollo.

En este punto se tuvo que decidir si vitaminar la API y dotarla de interfaz web, puesto que Django es un framework para desarrollo web, o bien optar por una solución independiente. Se decidió utilizar una aplicación independiente para mantener de forma independiente las diferentes partes de la solución.

El framework seleccionado para la webapp es Laravel, en su versión 5, un framework de PHP orientado a objetos que se caracteriza por ser de los más punteros para la realización de aplicaciones modernas.

Casos de uso

Al tratarse de la aplicación que sirve de interfaz entre el administrativo y la API, en los casos de uso consideramos a ambos como agentes.

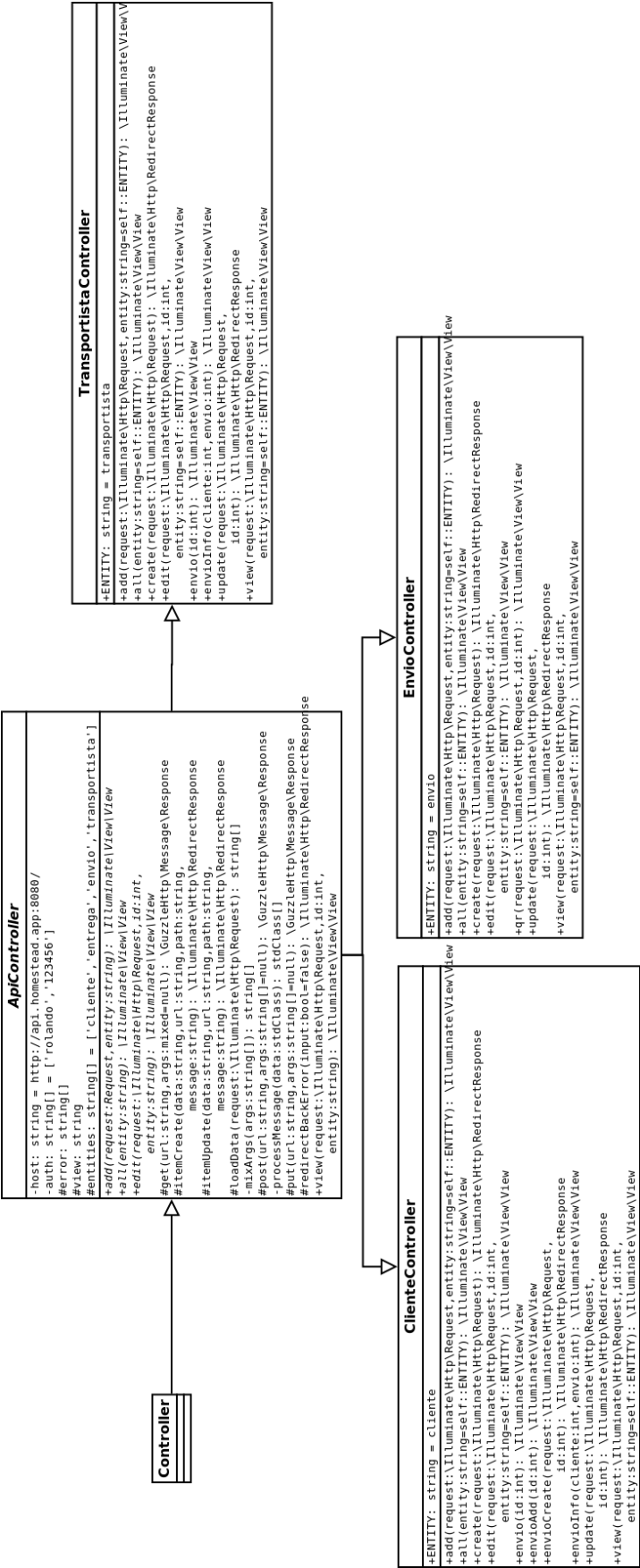


Se puede observar cómo en los casos de uso no se contempla la gestión de las entregas. Esto es así porque se considera responsabilidad del transportista, por lo que sólo él debe poder realizar tareas que alteren las entregas.

Diagrama UML

Laravel es un framework que implementa una visión evolucionada del MVC. A pesar de ello, nuestra solución sólo contempla la creación de clases que actúen a modo de controlador, puesto que las vistas no son clases (utiliza el gestor de plantillas Blade) y el modelo no existe como tal, puesto que tienen que ser conexiones a la API.

Lo que sí se contempla es la creación de una clase abstracta base que tenga la funcionalidad común, puesto que existirá un controlador por entidad gestionada por la aplicación y gran parte de la funcionalidad es compartida (como la conexión con la API).



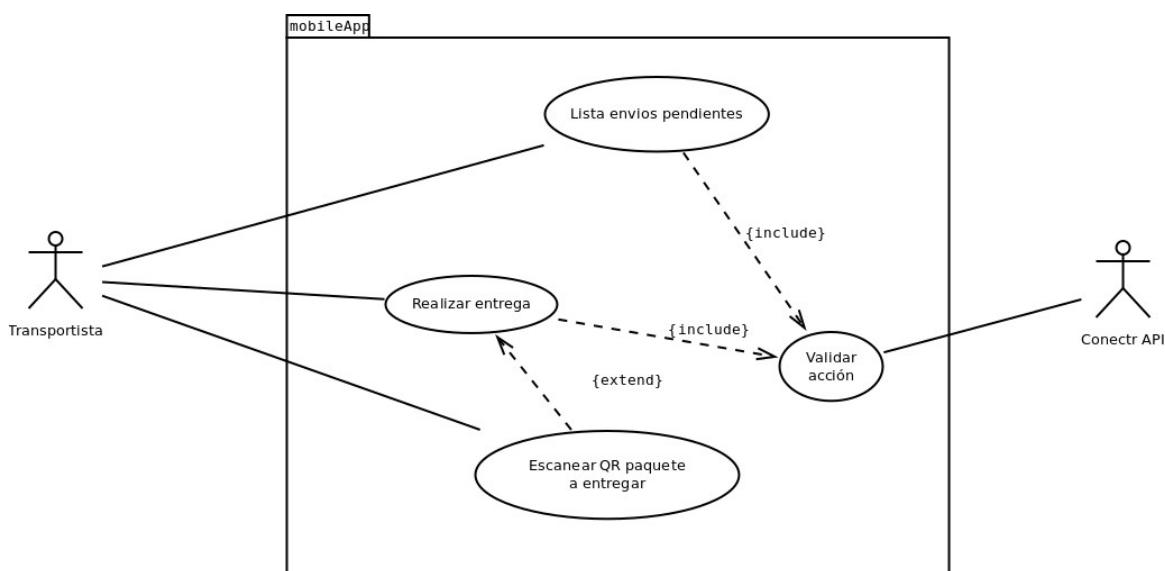
Desarrollo App Móvil

Para desarrollar la aplicación móvil se toma la decisión de realizarla con una solución híbrida en HTML5. Esto permitirá aprovechar el mismo desarrollo tanto para Android como para iPhone, Windows, Blackberry, FirefoxOS o UbuntuPhone.

Para el desarrollo de la aplicación en HTML5 se utilizará el framework AngularJS y para su implementación como aplicación “nativa” será Cordova el escogido.

Casos de uso

Al tratarse de la aplicación que sirve de interfaz entre el administrativo y la API, en los casos de uso consideramos a ambos como agentes.



Como el transportista sólo puede ver los envíos pendientes y alimentar el sistema de entregas, los casos de uso se limitan a los indicados.

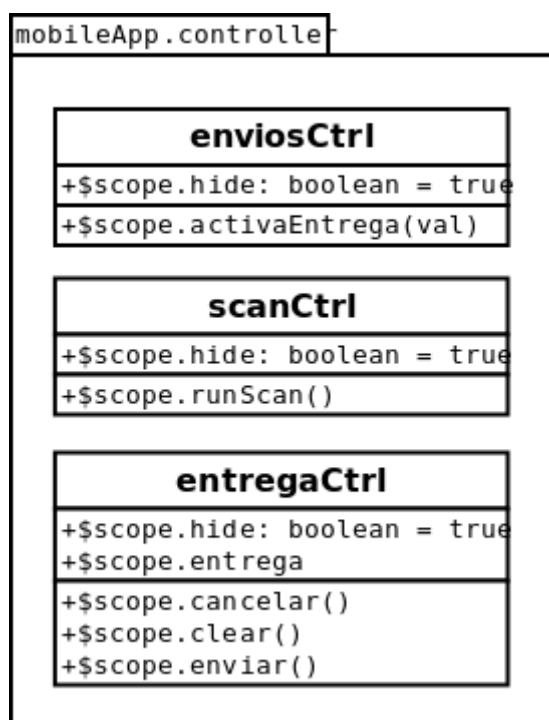
Para facilitar el trabajo al transportista, se incluye como caso de uso el escaneo del código QR que está pegado en cada paquete, de forma que la aplicación le lleve automáticamente a la entrega de dicho envío.

Diagrama UML

Como la aplicación móvil se realiza en HTML5, el lenguaje de programación es JavaScript (AngularJS es un framework de JavaScript) no podemos contar con un diagrama UML al uso.

Aunque en JS existen objetos, no se trata de un lenguaje basado en clases sino que se basa en prototipos.

Lo que se muestra a continuación es el diagrama UML dónde se refleja los controladores de AngularJS que deben ser desarrollados. Las vistas no existen como tales, puesto que es el etiquetado HTML incluido en el index.html



3. Ejecución del proyecto

3.1 Contenido

Tareas dependientes y recursos necesarios

Aunque a estas alturas del desarrollo del proyecto ya se han identificado múltiples tareas e incluso desarrollado un modelo inicial, es pertinente establecer un calendario de acciones y tareas en un diagrama de Gantt, permitiéndonos localizar tareas dependientes y mejorar la identificación de acciones ya realizada de manera informal.

El diagrama de Gantt se adjunta al documento del proyecto y también podrá consultar su versión abreviada (desglose de tareas) en el apartado 3.4; mientras tanto detallaremos las tareas, fases del proyecto.

Análisis estratégico

El análisis estratégico fue realizado al inicio del proyecto, motivo por el cual no procedemos a detallar esta tarea, puesto que ya ha sido realizada.

Esta tarea recayó en los gestores de la empresa de transporte, sin que en ella entrase en juego ningún técnico.

Se trata de la primera tarea y, su finalización, es condición necesaria para poder iniciarse la siguiente tarea.

Análisis técnico

Esta tarea requiere de la finalización del análisis estratégico, puesto que en esta fase del proyecto es necesario la existencia de los requisitos obtenidos en la fase de análisis estratégico junto con la incorporación del personal técnico necesario para llevar a cabo el análisis del proyecto desde un punto de vista técnico.

En este punto del proyecto, el jefe de proyecto y analista programador debe encontrarse ya en plantilla.

Además de llevar a cabo el análisis ya realizado de la viabilidad técnica y las diferentes soluciones posibles, decidiendo tanto el modelo de la solución como las herramientas y recursos a utilizar, formará parte del equipo que lleve a cabo la selección de personal para completar la plantilla con el desarrollador senior y el becario del FEUGA.

Diseño

Con la fase del análisis técnico terminada se iniciaba la fase de diseño. Durante esta fase, el Analista programador realiza el diseño del modelo de la aplicación, apoyándose en la experiencia del programador Senior. El becario se encuentra presente en todas las reuniones de diseño como observador.

Implementación API

Esta fase del desarrollo es importante y crítica. Aunque se trata de una tarea de desarrollo y podría realizarse de forma pareja al resto de fases de desarrollo, especialmente gracias a los diagramas UML; debido a que la aplicación que sirve de API es la aplicación que va a convertirse en la capa de datos tanto de la WebApp como de la App móvil, las tareas de implementación WebApp y App móvil son dependientes de la implementación de la API no pudiendo iniciarse hasta que ésta termine.

Se trata, pues, de la primera fase netamente de codificación.

Esta tarea se divide en varias subtareas, siendo varias de ellas desarrolladas en paralelo aprovechando la existencia de varios desarrolladores. En esta fase los tres trabajadores del proyecto realizarán labores de codificación.

Implementación WebApp y App móvil

Se tratan de dos tareas diferentes, pero las comentamos juntas puesto que su desarrollo se inicia en paralelo, de esta forma el personal más experimentado se encargará del desarrollo de la WebApp, dejando al becario del FEUGA trabajar en el desarrollo de la App móvil supervisado en todo momento por el AP. Esta división de tareas se realiza bajo el criterio de dificultad y situación crítica, al considerarse menos crítico el desarrollo móvil tanto por dificultad técnica como por número de pantallas / opciones.

Pruebas

Durante esta fase, el Analista Programador y el desarrollador senior definen y desarrollan los diferentes test de prueba. El lanzamiento de estos test se asignan al becario del FEUGA.

Experiencia piloto

Parte final del proyecto, en la cual se selecciona a un transportista y un administrativo para realizar la experiencia piloto, de manera que se pueda comprobar bajo el terreno la utilidad y correcto funcionamiento de la aplicación.

Gestión Integral Envío Paquetería**Start Date: June 1, 2015****Rolando Caldas Sánchez**

Task	Start Date	End Date	Duration (days)	Percent Complete
Date:				
1.0 Análisis estratégico	2015-06-01	2015-07-10	40	100.00%
1.1 Establecer objetivos	2015-06-01	2015-06-05	5	100.00%
1.2 Identificar necesidades básicas	2015-06-08	2015-06-19	12	100.00%
1.3 Estudiar viabilidad proyecto	2015-06-22	2015-07-10	19	100.00%
2.0 Análisis técnico	2015-07-13	2015-08-21	40	100.00%
2.1 Determinar modelo de la solución	2015-07-13	2015-07-24	12	100.00%
2.2 Analizar herramientas y recursos	2015-07-27	2015-08-21	26	100.00%
3.0 Diseño	2015-08-24	2015-10-09	47	100.00%
3.1 Creación de diagramas	2015-08-24	2015-09-22	30	100.00%
3.2 Validación de diagramas	2015-09-07	2015-10-09	33	100.00%
3.3 Correcciones diagramas	2015-09-14	2015-10-09	26	100.00%
4.0 Implementación API	2015-10-12	2015-11-06	26	100.00%
4.1 Creación de entorno virtual Python	2015-10-12	2015-10-13	2	100.00%
4.2 Instalacion Django + REST Framework	2015-10-14	2015-10-14	1	100.00%
4.3 Configuración framework	2015-10-15	2015-10-15	1	100.00%
4.4 Creación paquete models	2015-10-19	2015-10-27	9	100.00%
4.5 Creación paquete serializers	2015-10-22	2015-10-27	6	100.00%
4.6 Creación paquetes vistas	2015-10-22	2015-10-27	6	100.00%
4.7 Pruebas bajo entorno web	2015-10-28	2015-11-03	7	100.00%
4.8 Implementación salida JSON y XML	2015-11-03	2015-11-06	4	100.00%
4.9 Protección acceso API por user:pass	2015-11-03	2015-11-06	4	100.00%
4.0 Implementación WebApp	2015-11-09	2015-11-27	19	100.00%
4.1 Creación de entorno virtual Vagrant	2015-11-09	2015-11-10	2	100.00%
4.2 Configuración Laravel	2015-11-11	2015-11-11	1	100.00%
4.3 Configuración enrutado URLs	2015-11-12	2015-11-12	1	100.00%
4.4 Creación controladores	2015-11-13	2015-11-17	5	100.00%
4.5 Creación vistas de los controladores	2015-11-18	2015-11-20	3	100.00%
4.6 Comprobación funcionalidades	2015-11-23	2015-11-27	5	100.00%
4.7 Pruebas bajo entorno web	2015-11-25	2015-11-27	3	100.00%
5.0 Implementación App móvil	2015-11-09	2015-12-03	25	85.00%
5.1 Instalación Cordova y DVA	2015-11-09	2015-11-09	1	100.00%
5.2 Creación aplicación móvil Cordova	2015-11-10	2015-11-10	1	100.00%
5.3 Desarrollo JS Angular	2015-11-11	2015-11-17	7	100.00%
5.4 Comprobación funcionalidades	2015-11-18	2015-11-18	1	100.00%
5.5 Pruebas bajo entorno web	2015-11-19	2015-11-20	2	100.00%
5.6 Pruebas bajo entorno móvil	2015-11-23	2015-11-27	5	100.00%
5.7 Correcciones	2015-12-01	2015-12-03	3	0.00%
5.0 Pruebas	2015-11-23	2015-12-11	19	38.46%
5.1 Definición batería de pruebas formales	2015-11-23	2015-11-27	5	100.00%
5.2 Lanzamiento tests	2015-12-01	2015-12-03	3	0.00%
5.7 Correcciones	2015-12-07	2015-12-11	5	0.00%
6.0 Experiencia piloto	2015-12-14	2016-02-06	55	0.00%
6.1 Selecccion equipo para piloto	2015-12-14	2015-12-18	5	0.00%
6.2 Despliegue experiencia piloto	2016-01-08	2016-02-06	30	0.00%

Permisos y autorizaciones necesarias

Protección de datos

Como la aplicación almacena información de clientes y transportistas hay que tener en cuenta la Ley Orgánica de Protección de Datos antes de que la aplicación empiece a alimentarse con contenido.

Como la información de los clientes se limita a empresas o autónomos, el desarrollo no se ve afectado por la LOPD en este punto. Hay que tener en cuenta que en una de las últimas reformas legislativas, se incorporó la aclaración de que aunque la información de un autónomo son datos personales, si éstos son necesarios/utilizados en el marco de su actividad económica, no se encuentran supeditados a la LOPD.

Sin embargo, los datos de los transportistas sí se consideran personales, puesto que son empleados por cuenta ajena. Del mismo modo, no se puede garantizar que los destinatarios de los envíos sean siempre empresas o autónomos.

Por lo tanto, el equipo de desarrolladores firmarán el correspondiente anexo contractual con las cláusulas de confidencialidad, tratamiento y protección de datos de acuerdo a la normativa para los datos de carácter personal de protección media.

Los transportistas, también firmarán el consentimiento para el tratamiento de los datos y de la incorporación de los mismos al fichero exclusivo para la aplicación, que será dado de alta en la agencia de protección de datos por parte del responsable de los datos de la empresa.

Otros permisos o autorizaciones

No se requiere ningún tipo de autorización, permiso o licencia a mayores. A pesar de ello y con la finalidad de proteger la propiedad intelectual del desarrollo, todo el personal involucrado en el desarrollo del proyecto deberá firmar un contrato en el que se ceden todos los derechos a la empresa, salvo el derecho moral de autoría.

3.2 Riesgos

Como todo trabajo, el desarrollo del proyecto no está libre de riesgos. Los desarrolladores, tras la firma del contrato, tendrán que asistir a un curso formativo en las instalaciones de la empresa, impartido por una empresa certificada, de prevención de riesgos laborales.

Además de este curso de formación, los trabajadores recibirán el manual de buenas prácticas de la empresa en el cual se especifican normas básicas de convivencia tales como los retrasos en la incorporación al puesto de trabajo, necesidad de dejar el escritorio limpio al finalizar la jornada laboral, etc.

Para finalizar este punto, el personal del proyecto realizará su trabajo en puestos de trabajo individuales. Estos puestos de trabajo se encontrarán ubicados en una nave

correctamente equipada para el desarrollo de software, con amplia ventilación, aire acondicionado y aseos.

Un punto importante en prevención de riesgos sería la situación de los servidores web. Como se contratan en un centro de datos, la empresa no tiene que preocuparse por garantizar una temperatura óptima, conectividad, etc.

3.3 Revisión de recursos

No es preciso realizar una revisión de recursos, puesto que ya han sido revisados con anterioridad (Tarea 2), junto con una correcta temporización de tareas (Diagrama de Gantt)

3.4 Documentación de ejecución

En este punto se explicará el desarrollo realizado, incorporando información sobre cómo se procedió a instalar y configurar el entorno. También se adjuntará el código fuente del desarrollo realizado. La documentación de código se encuentra dentro del propio código fuente.

Instalación y configuración de aplicación API

La API correrá, tanto en desarrollo como en producción, bajo un entorno Linux. Como se va a desarrollar en Python, no es necesario instalar Python puesto que ya se incluye en toda instalación de Linux.

Por motivos de seguridad, la aplicación se desarrolla y ejecuta en un entorno virtual. Python es un lenguaje que proporciona, de manera nativa, un entorno virtual a modo de jaula, de esta manera pueden cohabitar múltiples versiones y configuraciones de Python sin que entren en conflicto.

Dentro del directorio “proyecto” se creó la carpeta env, en la cual se instaló el entorno virtual con el siguiente comando de consola:

```
$ virtualenv env
```

Si no se encuentra el programa virtualenv se debe instalar por apt:

```
# apt-get install python-virtualenv
```

Para instalar django, debemos hacerlo desde dentro del entorno virtual, algo que conseguimos activando el entorno:

```
$ source env/bin/activate
```

Tras ello, en la terminal que lanzamos el comando source nos encontramos bajo el nuevo ambiente, por lo que deberemos seguir en ella, puesto que una nueva terminal no se ejecutaría en nuestro entorno virtual seguro.

Para instalar django, lo hacemos a través de la herramienta pip. También se instalarán las aplicaciones de Django: Django Rest Framework y Django Cors Headers (para evitar el error cross-origin en las peticiones AJAX que se realizan desde AngularJS en la aplicación móvil):

```
$ pip install django
$ pip install djangorestframework
$ pip install django-cors-headers
```

Una vez instalado, en la carpeta de “proyecto” se creó el proyecto django API, con el siguiente comando:

```
$ django-admin.py startproject api
```

Dentro de proyecto se creó la carpeta api, al mismo nivel que la carpeta env. Dentro de api tenemos una instancia de django sobre la cual trabajaremos.

Lo primero que se tiene que hacer es crear la aplicación, que llamaremos app:

```
$ cd api
$ python manage.py startapp app
```

Con esto tenemos creada dentro de nuestra instancia de django, la aplicación app

Lo siguiente que se tiene que hacer es configurar nuestro proyecto de django para que utilice nuestra aplicación (app) además de las aplicaciones de rest_framework y corsheaders. Para ello se edita el fichero api/settings.py alterando INSTALLED_APPS para dejarlo de la siguiente manera:

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'corsheaders',
    'app',
)
```

Para terminar de editar el fichero settings.py, necesitamos activar como Middleware corsheaders y permitir la conexión desde cualquier origen:

```

MIDDLEWARE_CLASSES = (
    'django.contrib.sessions.middleware.SessionMiddleware',
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'django.middleware.security.SecurityMiddleware',
)

CORS_ORIGIN_ALLOW_ALL = True

```

Con nuestra aplicación correctamente configurada, necesitamos establecer en `api/urls.py` los patrones para el enrutado de las URLs, ver fichero `proyecto/api/urls.py`

El siguiente paso es crear los modelos de datos, esto es importante en django puesto que no se crean las tablas de la base de datos directamente, sino que el ORM de Django se encarga de ello en función de las clases creadas en el fichero `models.py` dentro de las aplicaciones existentes (`proyecto/api/*/`)

Una vez creados los modelos, desde la misma terminal ejecutamos en la raíz de nuestra aplicación `api`:

```

$ python manage.py makemigrations app
$ python manage.py migrate

```

Si en el futuro se realizasen cambios en el modelo de datos, se deben volver a lanzar estos comandos. El ORM de django es lo suficientemente potente como para modificar satisfactoriamente la estructura de tablas sin problemas.

Continuando con la configuración, lo siguiente que tenemos que hacer es crear un superusuario, que se utilizará para conectarse a la API, esto lo hacemos también por consola:

```

$ python manage.py createsuperuser

```

El usuario creado fue “rolando” con contraseña “123456”. Por supuesto, en un entorno real la contraseña deberá tener, al menos, 24 caracteres y deberá contener mayúsculas, minúsculas, números y caracteres especiales.

El entorno ya estaría listo para empezar a desarrollar la API. Si en cualquier momento se quieren realizar pruebas sobre, por ejemplo, el uso de los modelos de datos, django nos proporciona una shell ejecutando el comando:

```
$ python manage.py shell
```

Una vez el desarrollo está avanzado, para probarlo en entorno web no se instalará ningún servidor estilo Apache o Ngnix, sino que se utilizará el propio servidor que django incluye lanzando el comando:

```
$ python manage.py runserver
```

En nuestro caso, como queremos lanzar el servidor bajo una IP y puerto concreto, se creó al mismo nivel de manage.py el fichero runserver:

```
#!/bin/bash
```

```
# Rolando
```

```
exec ./manage.py runserver 192.168.0.17:8080
```

Para que funcione correctamente, a este fichero se le dan permisos de ejecución:

```
$ chmod +x runserver
```

Además, como se va a probar via web accediendo a la API por el “dominio” api.homestead.app se edita el fichero /etc/hosts tanto del equipo dónde se encuentra la API instalada como el de todos los desarrolladores:

```
$ vim /etc/hosts
```

```
192.168.0.17    api.homestead.app
```

Para que al acceder a api.homestead.app:8080 se acceda realmente a 192.168.0.17:8080

Una vez levantad el servicio, la propia terminal muestra el log de peticiones como podemos ver a continuación:

```

api: python
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
(env)rolando@rolando-Lenovo-Z50-70:~/proyecto/api$ ./runserver
Performing system checks...

System check identified no issues (0 silenced).
November 27, 2015 - 15:37:07
Django version 1.8.6, using settings 'api.settings'
Starting development server at http://192.168.1.104:8080/
Quit the server with CONTROL-C.
[27/Nov/2015 15:37:14] "GET /cliente.json HTTP/1.1" 200 1444
[27/Nov/2015 15:37:16] "GET /cliente/4.json HTTP/1.1" 200 132
[27/Nov/2015 15:37:20] "GET /envio/cliente/4.json HTTP/1.1" 200 2
[27/Nov/2015 15:37:20] "GET /cliente/4.json HTTP/1.1" 200 132
[27/Nov/2015 15:37:24] "GET /transportista.json HTTP/1.1" 200 712
[27/Nov/2015 15:37:26] "GET /transportista/6.json HTTP/1.1" 200 116
[27/Nov/2015 15:37:28] "GET /envio/transportista/6.json HTTP/1.1" 200 2
[27/Nov/2015 15:37:28] "GET /transportista/6.json HTTP/1.1" 200 116
  
```

Instalación y configuración de WebAPP

Preparación del entorno

La aplicación web, al igual que la API, correrá bajo entorno Linux. Como seleccionamos Laravel como framework de desarrollo, estamos hablando del lenguaje de programación PHP en su versión 5, junto con la necesidad de instalar un servidor web como Apache o nginx, MySQL como SGBD, etc etc.

Sin embargo, Laravel proporciona a los desarrolladores un entorno virtual con todo preparado para utilizar el framework. Para ello Laravel ha desarrollado la herramienta Homestead, que se apoya en Vagrant, un software para crear y lanzar rápidamente entornos de desarrollo. Vagrant, a su vez, es una capa que se encuentra por encima de VirtualBox o VMWare.

Por lo tanto, utilizaremos Homestead para desarrollar y probar la aplicación webApp. Lo primero que necesitaremos es instalar VirtualBox y vagrant, algo que puede realizarse fácilmente desde apt-get o la web oficial del proyecto.

```
# apt-get install virtualbox-5.0
# apt-get install vagrant
```

Con vagrant instalado, necesitaremos descargar el Homestead Vagrant Box:

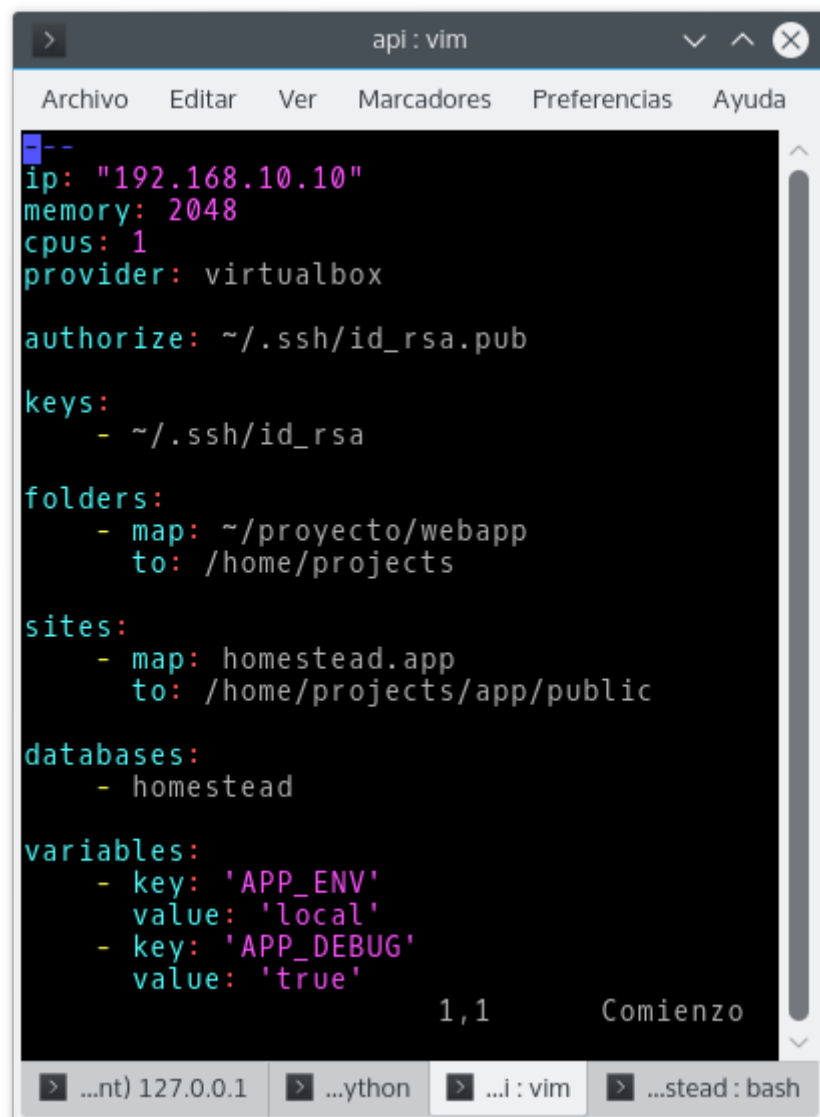
```
$ cd $HOME/proyecto
$ vagrant box add laravel/homestead
```

El siguiente paso es configurar Homestead para poder vincular nuestro directorio de trabajo con la máquina virtual, porque en el caso de Homestead, se realiza un mapeo de directorios asociando una ruta del servidor con una ruta dentro de la máquina virtual, además de un mapeo de puertos.

```
$ vim /home/rolando/.homestead/Homestead.yaml
```

Lo que haremos será:

- indicar la IP de la MV (192.168.10.10)
- la memoria RAM disponible (2048)
- el número de núcleos que puede utilizar (1)
- la llave SSH utilizada para poder conectarnos a la MV via SSH
- el mapeo de directorios del equipo (map) a la MV (to)
- el mapeo de sites, indicando el dominio asociado a la MV (homestead.app) y la ruta dentro de la MV que leerá como si un VirtualHost de apache se tratase.
- Bases de datos, etc.



The screenshot shows a terminal window titled 'api : vim'. The menu bar includes 'Archivo', 'Editar', 'Ver', 'Marcadores', 'Preferencias', and 'Ayuda'. The main content area displays a configuration file with the following structure:

```
--
ip: "192.168.10.10"
memory: 2048
cpus: 1
provider: virtualbox

authorize: ~/.ssh/id_rsa.pub

keys:
- ~/.ssh/id_rsa

folders:
- map: ~/proyecto/webapp
  to: /home/projects

sites:
- map: homestead.app
  to: /home/projects/app/public

databases:
- homestead

variables:
- key: 'APP_ENV'
  value: 'local'
- key: 'APP_DEBUG'
  value: 'true'
```

The status bar at the bottom shows '1,1' and 'Comienzo'. The bottom-most bar contains four tabs: '...nt) 127.0.0.1', '...ython', '...i : vim' (active), and '...stead : bash'.

El siguiente paso es generar la clave SSH que llamaremos id_rsa

```
$ ssh-keygen -t rsa -C "rolando.caldas@gmail.com"
```

Ya podemos lanzar la máquina virtual:

```
$ vagrant up
```

Y conectarnos a ella por SSH:

```
$ vagrant ssh
```

Listos para instalar Laravel!

Instalación de Laravel 5.1

Para instalar laravel, conectados por SSH a Homestead, descargamos su instalador de manera global a través de composer:

```
$ composer global require "laravel/installer=~1.1"
```

Ya podemos instalar una instancia de laravel

```
$ cd /home/projects  
$ mkdir webapp  
$ cd webapp  
$ laravel new app  
$ cd app
```

Ya tenemos el framework instalado y listo para iniciar a desarrollar la webapp. Antes de terminar debemos pensar si tenemos más tareas de configuración que realizar.

Lo primero, es recordar que necesitamos poder generar códigos QR, puesto que deben ser generados por la webapp para poder imprimir y pegar en los paquetes. Utilizaremos para ello el programa "simple-qrcode" que agregamos al proyecto vía composer:

```
$ composer require simplesoftwareio/simple-qrcode
```

Con la dependencia de paquete agregada a nuestro proyecto, lo actualizamos:

```
$ composer update
```

Lo último que nos falta instalar la tabla de usuarios y crear el user de prueba con el que acceder a nuestra webApp. Laravel ya proporciona un completo sistema de autenticación, por lo que no tenemos que hacer casi nada para tener listo un completo sistema de usuarios.

Lo primero que necesitamos hacer es crear un fichero PHP:

```
$ vim /home/projects/app/database/seeds/UserTableSeeder.php
```

Este fichero será el encargado de vaciar la tabla de usuarios y agregar nuestro user de prueba:

```
<?php

use Illuminate\Database\Seeder;
use App\User as User;

class UserTableSeeder extends Seeder {

    public function run() {
        User::truncate();

        User::create( [
            'email' => 'rolando.caldas@gmail.com' ,
            'password' => Hash::make( '123456' ) ,
            'name' => 'rolando' ,
        ] );
    }
}
```

Además, en el mismo directorio se encuentra el fichero DatabaseSeeder.php que tendremos que editar y descomentar esta línea:

```
// $this->call(UserTableSeeder::class);
```

De este modo, cuando hagamos el seed, se creará el usuario rolando con la contraseña 123456

Para crear la tabla de usuario y que se inserte nuestro user de prueba debemos ejecutar por consola:

```
$ php artisan migrate
$ php artisan optimize
$ php artisan db:seed
```

Por supuesto, todos los comandos se ejecutan dentro de nuestro homestead y en la raíz de nuestro proyecto.

Ya está el entorno listo para empezar a desarrollar la aplicación. El desarrollo se puede hacer desde el equipo principal con nuestro IDE habitual.

Instalación y configuración de nuestra app móvil

Aunque se trata de una aplicación móvil, su desarrollo se realiza desde un ordenador. Como se optó por una solución híbrida con Cordova, deberemos instalar tanto Cordova como el SDK de Android, la plataforma para la cual se desarrollará la aplicación.

En <https://rolandocaldas.com/android/instalar-cordova-en-ubuntu> se encuentran las instrucciones de instalación de forma detallada, aunque a continuación dejamos de forma resumida los comandos necesarios:

```
$ sudo apt-get update
$ sudo apt-get install nodejs npm
$ sudo ln -s /usr/bin/nodejs /usr/bin/node
$ sudo npm install -g cordova
$ cd /home/rolando/proyecto
$ cordova create mobileapp com.rolandocaldas.mobileApp mobileApp
```

Ya tenemos el esqueleto de la aplicación generada; sin embargo, necesitamos descargar el SDK de Google, extraerla (por ej en /home/rolando/android-sdk) y alterar el PATH:

```
$ export PATH=${PATH}:/home/rolando/android-sdk-linux/tools/
```

Agregamos la plataforma android a nuestra app:

```
$ cd mobileapp
$ cordova platform add android
```

Y el plugin para poder leer códigos QR:

```
$ cordova plugin add phonegap-plugin-barcodescanner
```

Ya podemos crear nuestro proyecto en Netbeans, para posteriormente instalar vía Bower los paquetes necesarios para nuestra app, que son:

- angular: El framework de Javascript AngularJS
- bootstrap: El framework de CSS3 para Responsive Design
- font-awesome: Tipografías de iconos
- jquery: Es una dependencia de bootstrap y será instalado automáticamente gracias a las dependencias de bower

Para compilar nuestra aplicación y obtener el APK sólo es necesario lanzar el comando:

```
$ cordova build
```


Desarrollo realizado en API

En Django, es necesario indicar explícitamente las direcciones HTTP que es capaz de procesar, de forma que cualquier patrón de URL que no esté contemplado generará un error 404. Además, cada enrutado se asocia con una vista a cargar.

Se puede consultar el fichero de urls en `api/api/urls.py`

Dentro de `api/app` se desarrollaron los siguientes ficheros:

- `models.py` Fichero con el modelo de los datos (nuestro diagrama E/R), contiene una clase por cada tabla, indicando sus campos y restricciones.
- `permissions.py` Fichero donde se crea la clase utilizada para informar si el usuario conectado a la API tiene permisos o no para realizar las acciones CRUD.
- `cif.py` Es el fichero que contiene la clase que coteja que los NIF y CIF introducidos sean válidos (restricción incluida en los modelos)
- `serializers.py` Es el fichero que contiene una clase por cada vista, para serializar/deserializar la información en JSON o XML
- `views.py` Es el fichero que genera la interfaz de comunicación

Desarrollo realizado en webApp

Laravel proporciona un sistema sencillo y rápido para desarrollo. Al igual que Django, existe un fichero de enrutado, en este caso se trata del fichero de PHP `webapp/app/app/Http/routes.php`

Además del `routes.php` debemos crear los controladores y las vistas. No creamos los modelos puesto que utiliza la API como modelo.

Los controladores se crean dentro de `webapp/app/app/Http/Controllers` con las especificaciones indicadas en los UML

Los controladores se complementan con las vistas que, organizadas por carpetas (1 carpeta = 1 entidad) se ubican en `webapp/app/resources/views`

Desarrollo realizado en movil App

La aplicación movil presenta un index.html con el etiquetado HTML que funciona como vistas de la aplicación gracias a los atributos ng-* específicos de AngularJS. Es en este index.html en el cual se indica qué porción de código HTML es gestionado por qué controlador de JS creado en AngularJS.

Como todo documento HTML, carga los ficheros CSS y JS necesarios, los ficheros de JS se explican y detallan a continuación:

- cordova.js Este fichero se incluye en el documento, pero sólo está disponible cuando se compila la app puesto que es la interfaz que proporciona cordova para comunicar nuestro JS con el hardware del dispositivo móvil.
- js/write.js Librería para crear un lienzo o canvas sobre el cual el destinatario de un envío firma la entrega. Esta firma se almacena como imagen en la API.
- js/config.js Es el fichero que incluye el objeto de configuración de la app. Aquí se indica la URL de la API, el usuario y password para conectar y el id del transportista. Cada aplicación tendrá un id de transportista diferente, de forma que cada dispositivo móvil tendrá la misma app, pero con este fichero config.js diferente.
- bower_components/angular/angular.js El framework de desarrollo AngularJS
- js/index.js Nuestro código JS encargado de gestionar las pantallas de la aplicación y realizar las correspondientes llamadas a la API

Durante el desarrollo, en el momento de compilar al APK se obtuvo el siguiente error:

```
java.io.IOException: Cannot run program "/home/rolando/android-sdk/build-tools/23.0.2/aapt": error=2, No such file or directory
```

El problema nace de estar desarrollando en un equipo de 64bits y, tras investigar el problema, se llegó a la conclusión que era preciso instalar la versión de 32 bits de varias librerías:

```
sudo apt-get install gcc-multilib lib32z1 lib32stdc++6
```

Tras esta instalación de software se pudo obtener el APK sin problema.

Puesta en marcha de aplicación API

Para poner en marcha en un entorno real la aplicación, es necesario instalar en la máquina destino el entorno, no pudiendo trasladarse tal cual el directorio env que contiene el entorno virtual de python.

En <http://api.demosonline.org:8080/> está corriendo el entorno de pruebas de la aplicación, lista para su experiencia piloto.

Para poder montar esta aplicación, en un servidor bajo Debian, se realizaron los siguientes pasos:

Se instala el entorno virtual

```
$ cd $HOME
$ mkdir proyecto && cd proyecto
$ sudo apt-get install python-virtualenv
$ virtualenv env
$ source env/bin/activate
```

Se django y las demás apps

```
$ pip install django
$ pip install django-rest-framework
$ pip install django-cors-headers
$ django-admin.py startproject api
$ cd api
$ python manage.py startapp app
```

Se copian los ficheros de nuestra API

```
$ cp [ruta fuentes]/api/api/settings.py ./api/
$ cp [ruta fuentes]/api/api/urls.py ./api/
$ cd app
$ cp -R [ruta fuentes]/api/app/* ./
```

Eliminamos los ficheros de migrations para poder crear la BD

```
$ cd migrations
$ rm 0*
$ rm __init__.pyc
```

Se lanza el migrate

```
$ cd ..
$ cd ..
$ python manage.py makemigrations app
$ python manage.py migrate
```

Se crea el supersusuario

```
$ python manage.py createsuperuser
```

Se inicia el servidor web

```
$ vim runserver
```

Este es el contenido a pegar en runserver:

```
exec ./manage.py runserver api.demosonline.org:8080
```

```
$ chmod +x runserver
```

```
$ ./runserver &
```

```
[1] 25126
```

```
$ Performing system checks...
```

```
System check identified no issues (0 silenced).
```

```
November 28, 2015 - 17:37:30
```

```
Django version 1.9c1, using settings 'api.settings'
```

```
Starting development server at http://api.demosonline.org:8080/
```

```
Quit the server with CONTROL-C.
```

Y ya tenemos nuestra API funcionando en el entorno de pruebas.

Puesta en marcha de aplicación web

Para activar la aplicación web, se instala en un servidor Debian Jessie. Hay que tener en cuenta que Laravel 5.1 tiene como requisito que la versión de PHP instalada sea, al menos, la 5.5.9 y las siguientes extensiones:

```
PHP >= 5.5.9 OpenSSL PHP Extension Mbstring PHP Extension Tokenizer PHP Extension
```

Partimos de un servidor con Debian Jessie y un servidor web completo instalado con Apache2 PHP5 y MySQL5, con el PHP corriendo en FastCGI

Sólo necesitamos copiar el directorio app de webapp para colocarlo dentro del directorio asociado al VirtualHost <http://app.demosonline.org/>

En el VirtualHost de Apache se indica que el document_root es el directorio public dentro de app.

Además, es necesario realizar cambios de configuración. En app/config/app.php se cambian los siguientes valores:

```
'url' => 'http://app.demosonline.org'
```

Además, dentro del directorio app lanzamos artisan para generar una nueva clave de aplicación:

```
$ php artisan key:generate
```

Se crea la base de datos a utilizar y se lanza el migrate de artisan:

```
$ php artisan migrate  
$ php artisan optimize  
$ php artisan db:seed
```

Puesta en marcha de aplicación web

La aplicación móvil se encuentra compilada y lista para instalar. Se trata de una compilación en modo debug, no válido para subir al Play Store, pero totalmente válido para la experiencia piloto.

Está compilado para ser utilizado en el dispositivo del transportista 1, que corresponderá al elegido para la experiencia piloto.

Para poder instalar el APK en el dispositivo Android, es necesario que éste tenga activada la opción que permite instalar aplicaciones desde fuentes desconocidas.

Documentación de código

Todo el código fuente se encuentra documentado siguiendo los estándares marcados inicialmente por java-doc tanto en la aplicación móvil como en la web.

En el caso de la API, debido al alto nivel del código, no se consideró oportuno realizar una documentación al uso, puesto que la propia sintaxis del código de lo suficientemente clara.

Código fuente

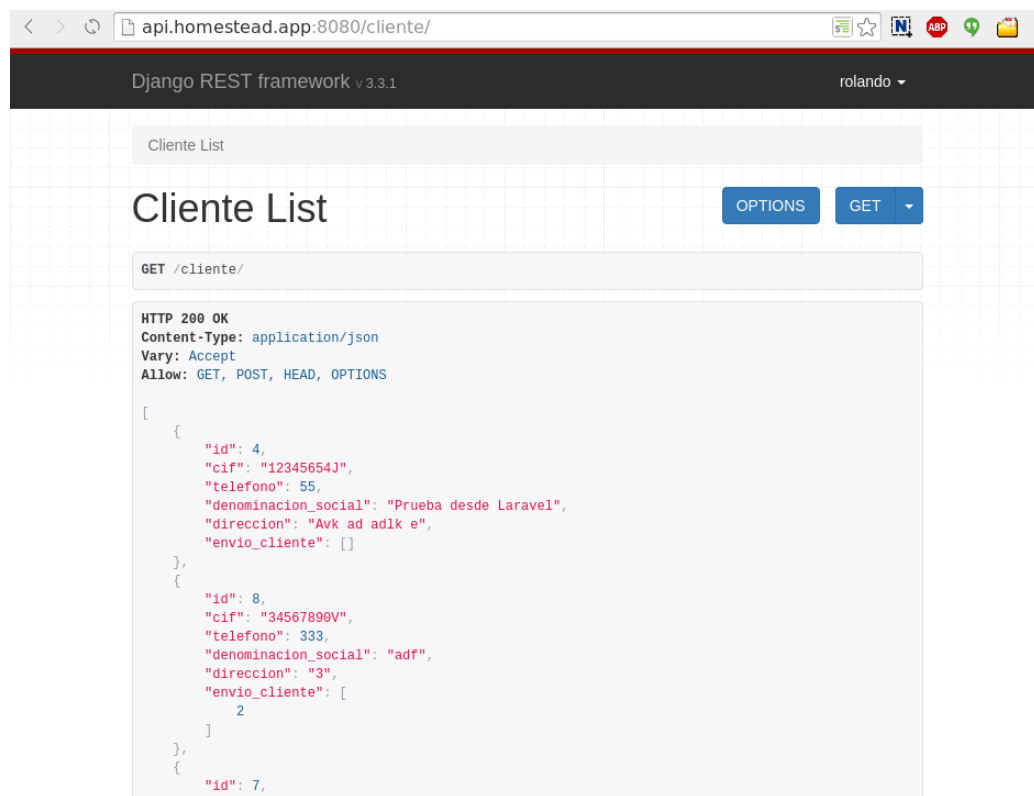
El código fuente se adjunta en el directorio src situado junto al presente documento. Además, se encuentra disponible en github:

<https://github.com/rolando-caldas/dam-project-2015>

Capturas API

Una característica de Django Rest Framework es proporcionar un interfaz HTML y JSON. A continuación mostramos alguna captura de la implementación:

Listado de clientes



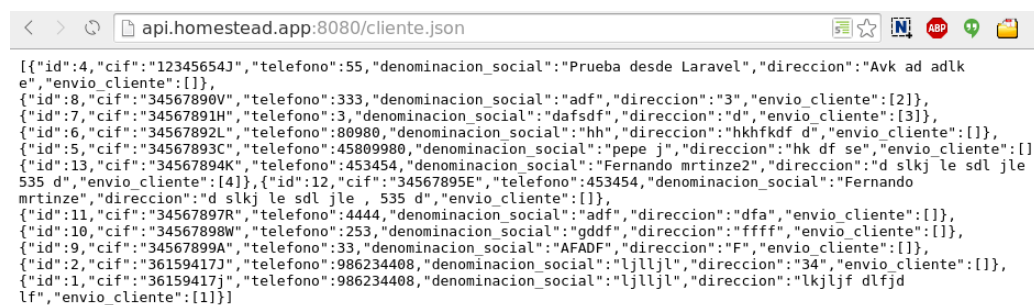
Cliente List

OPTIONS GET

GET /cliente/

HTTP 200 OK
Content-Type: application/json
Vary: Accept
Allow: GET, POST, HEAD, OPTIONS

```
[
  {
    "id": 4,
    "cif": "12345654J",
    "telefono": 55,
    "denominacion_social": "Prueba desde Laravel",
    "direccion": "Avk ad adlk e",
    "envio_cliente": []
  },
  {
    "id": 8,
    "cif": "34567890V",
    "telefono": 333,
    "denominacion_social": "adf",
    "direccion": "3",
    "envio_cliente": [
      2
    ]
  },
  {
    "id": 7,
    "cif": "34567891H",
    "telefono": 3,
    "denominacion_social": "dafsdf",
    "direccion": "d",
    "envio_cliente": [3]
  },
  {
    "id": 6,
    "cif": "34567892L",
    "telefono": 80980,
    "denominacion_social": "hh",
    "direccion": "hkhfkdf d",
    "envio_cliente": []
  },
  {
    "id": 5,
    "cif": "34567893C",
    "telefono": 45809980,
    "denominacion_social": "pepe j",
    "direccion": "hk df se",
    "envio_cliente": []
  },
  {
    "id": 13,
    "cif": "34567894K",
    "telefono": 453454,
    "denominacion_social": "Fernando mrtinze2",
    "direccion": "d slkj le sdl jle , 535 d",
    "envio_cliente": [4]
  },
  {
    "id": 12,
    "cif": "34567895E",
    "telefono": 453454,
    "denominacion_social": "Fernando mrtinze",
    "direccion": "d slkj le sdl jle , 535 d",
    "envio_cliente": []
  },
  {
    "id": 11,
    "cif": "34567897R",
    "telefono": 4444,
    "denominacion_social": "adf",
    "direccion": "dfa",
    "envio_cliente": []
  },
  {
    "id": 10,
    "cif": "34567898W",
    "telefono": 253,
    "denominacion_social": "gddf",
    "direccion": "ffff",
    "envio_cliente": []
  },
  {
    "id": 9,
    "cif": "34567899A",
    "telefono": 33,
    "denominacion_social": "AFADF",
    "direccion": "F",
    "envio_cliente": []
  },
  {
    "id": 2,
    "cif": "36159417J",
    "telefono": 986234408,
    "denominacion_social": "ljlljl",
    "direccion": "34",
    "envio_cliente": []
  },
  {
    "id": 1,
    "cif": "36159417J",
    "telefono": 986234408,
    "denominacion_social": "ljlljl",
    "direccion": "lkjljf dlfdj lf",
    "envio_cliente": [1]
  }
]
```



```
[{"id":4,"cif":"12345654J","telefono":55,"denominacion_social":"Prueba desde Laravel","direccion":"Avk ad adlk e","envio_cliente":[]}, {"id":8,"cif":"34567890V","telefono":333,"denominacion_social":"adf","direccion":"3","envio_cliente":[2]}, {"id":7,"cif":"34567891H","telefono":3,"denominacion_social":"dafsdf","direccion":"d","envio_cliente":[3]}, {"id":6,"cif":"34567892L","telefono":80980,"denominacion_social":"hh","direccion":"hkhfkdf d","envio_cliente":[]}, {"id":5,"cif":"34567893C","telefono":45809980,"denominacion_social":"pepe j","direccion":"hk df se","envio_cliente":[]}, {"id":13,"cif":"34567894K","telefono":453454,"denominacion_social":"Fernando mrtinze2","direccion":"d slkj le sdl jle , 535 d","envio_cliente":[4]}, {"id":12,"cif":"34567895E","telefono":453454,"denominacion_social":"Fernando mrtinze","direccion":"d slkj le sdl jle , 535 d","envio_cliente":[]}, {"id":11,"cif":"34567897R","telefono":4444,"denominacion_social":"adf","direccion":"dfa","envio_cliente":[]}, {"id":10,"cif":"34567898W","telefono":253,"denominacion_social":"gddf","direccion":"ffff","envio_cliente":[]}, {"id":9,"cif":"34567899A","telefono":33,"denominacion_social":"AFADF","direccion":"F","envio_cliente":[]}, {"id":2,"cif":"36159417J","telefono":986234408,"denominacion_social":"ljlljl","direccion":"34","envio_cliente":[]}, {"id":1,"cif":"36159417J","telefono":986234408,"denominacion_social":"ljlljl","direccion":"lkjljf dlfdj lf","envio_cliente":[1]}]
```

Pantalla detalle cliente

api.homestead.app:8080/cliente/1

Django REST framework v 3.3.1 rolando

Cliente List / Cliente Detail

Cliente Detail

DELETE OPTIONS GET

GET /cliente/1

HTTP 200 OK
Content-Type: application/json
Vary: Accept
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS

```
{
  "id": 1,
  "cif": "36159417j",
  "telefono": 986234408,
  "denominacion_social": "ljljl",
  "direccion": "lkjljf dlfjd lf",
  "envio_cliente": [
    1
  ]
}
```

api.homestead.app:8080/cliente/1.json

```
{
  "id": 1, "cif": "36159417j", "telefono": 986234408, "denominacion_social": "ljljl", "direccion": "lkjljf dlfjd lf", "envio_cliente": [1]}

```

Transportistas

api.homestead.app:8080/transportista/

Django REST framework v 3.3.1 rolando

Transportista List

Transportista List

OPTIONS GET

GET /transportista/

HTTP 200 OK
Content-Type: application/json
Vary: Accept
Allow: GET, POST, HEAD, OPTIONS

```
[
  {
    "id": 6,
    "apellidos": "22ddfd",
    "nif": "34567890L",
    "nombre": "aaa",
    "telefono": 3333,
    "envio_transportista": [],
    "owner": null
  },
  {
    "id": 3,
    "apellidos": "22ddfd",
    "nif": "34567890L",
    "nombre": "aaa",
    "telefono": 3333,
    "envio_transportista": [],
    "owner": null
  }
]
```

api.homestead.app:8080/transportista.json

```
[{"id":6,"apellidos":"22ddfd","nif":"34567890L","nombre":"aaa","telefono":3333,"envio_transportista":[],"owner":null}, {"id":3,"apellidos":"d","nif":"34567891H","nombre":"jh","telefono":0,"envio_transportista":[4],"owner":null}, {"id":4,"apellidos":"h","nif":"34567892L","nombre":"jh","telefono":0,"envio_transportista":[3],"owner":null}, {"id":2,"apellidos":"dfalj","nif":"34567893C","nombre":"jl","telefono":9090,"envio_transportista":[],"owner":null}, {"id":5,"apellidos":"dadsf2","nif":"34567894K","nombre":"fdsgsdf","telefono":445454,"envio_transportista": [],"owner":null},{ "id":1,"apellidos":"Caldas SÁnchez","nif":"57432844G","nombre":"Rolando","telefono":986234408,"envio_transportista":[1,2],"owner":null}]

```

Detalle transportista

api.homestead.app:8080/transportista/3

Django REST framework v3.3.1 rolando

Transportista List / Transportista Detail

Transportista Detail

DELETE OPTIONS GET

GET /transportista/3

HTTP 200 OK
Content-Type: application/json
Vary: Accept
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS

```
{
  "id": 3,
  "apellidos": "d",
  "nif": "34567891H",
  "nombre": "jh",
  "telefono": 0,
  "envio_transportista": [
    4
  ],
  "owner": null
}
```

{"id":3,"apellidos":"d","nif":"34567891H","nombre":"jh","telefono":0,"envio_transportista":[4],"owner":null}

Pantalla envios

api.homestead.app:8080/envio/

Django REST framework v3.3.1 rolando

Envio List

OPTIONS GET

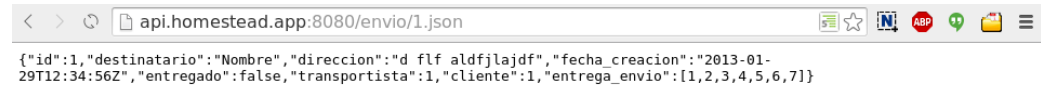
GET /envio/

HTTP 200 OK
Content-Type: application/json
Vary: Accept
Allow: GET, POST, HEAD, OPTIONS

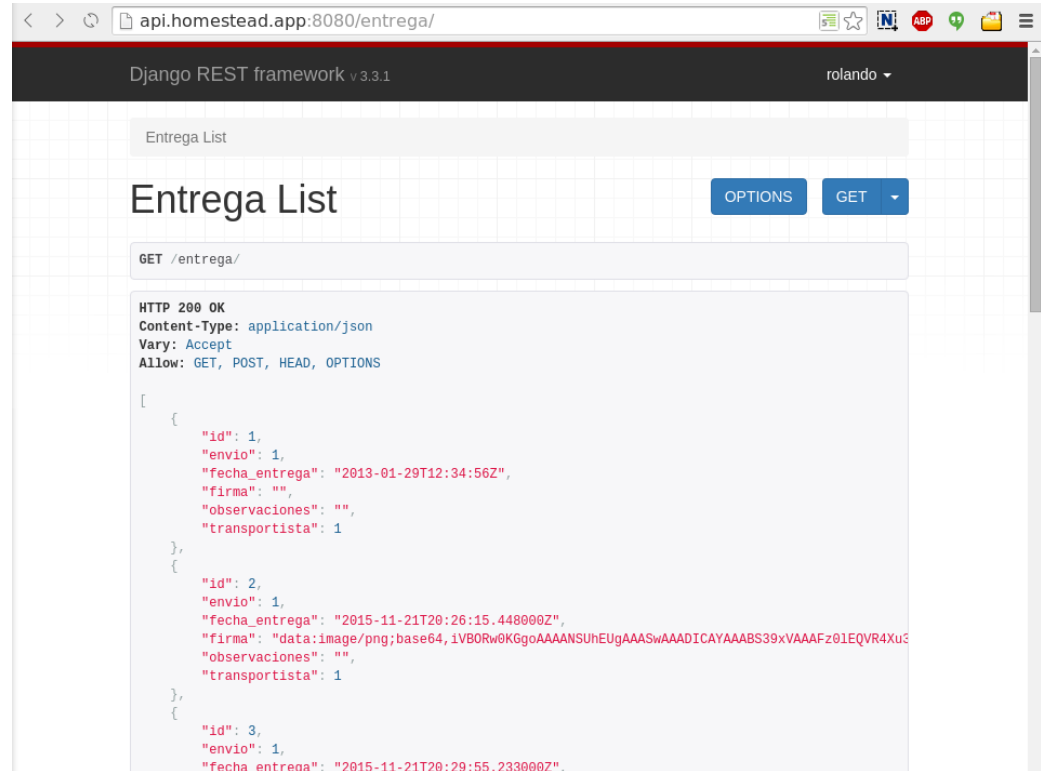
```
[
  {
    "id": 1,
    "destinatario": "Nombre",
    "direccion": "d flf aldfjlajdf",
    "fecha_creacion": "2013-01-29T12:34:56Z",
    "entregado": false,
    "transportista": 1,
    "cliente": 1,
    "entrega_envio": [
      1,
      2,
      3,
      4,
      5,
      6,
      7
    ]
  },
  {
    "id": 2,

```


Pantalla detalle envio



Pantalla entregas

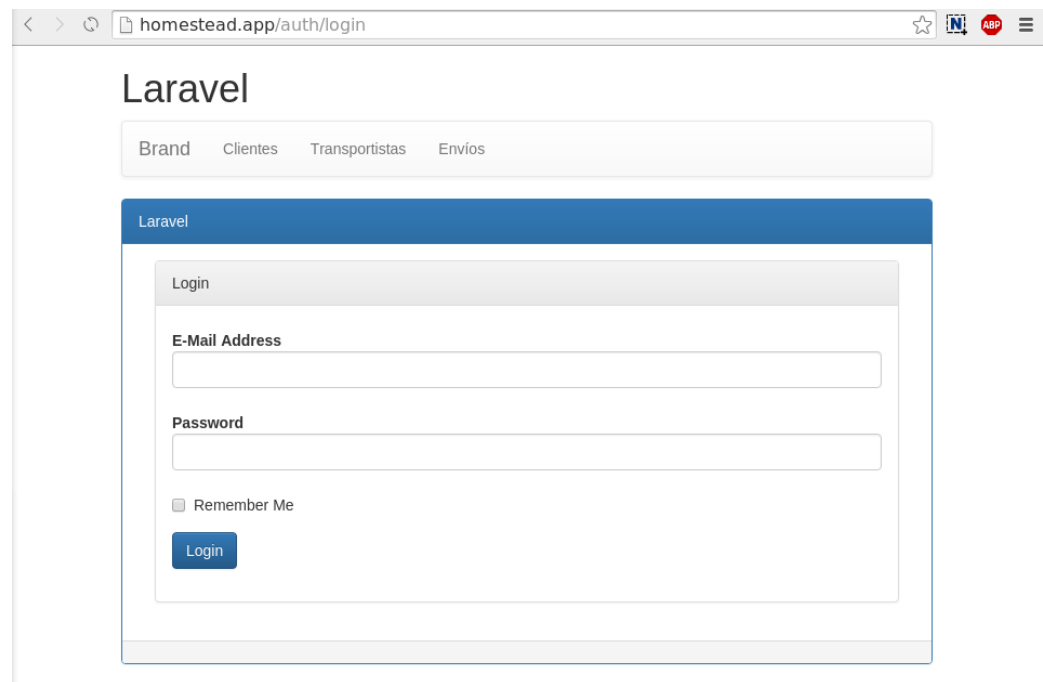


Pantalla entrega detalle



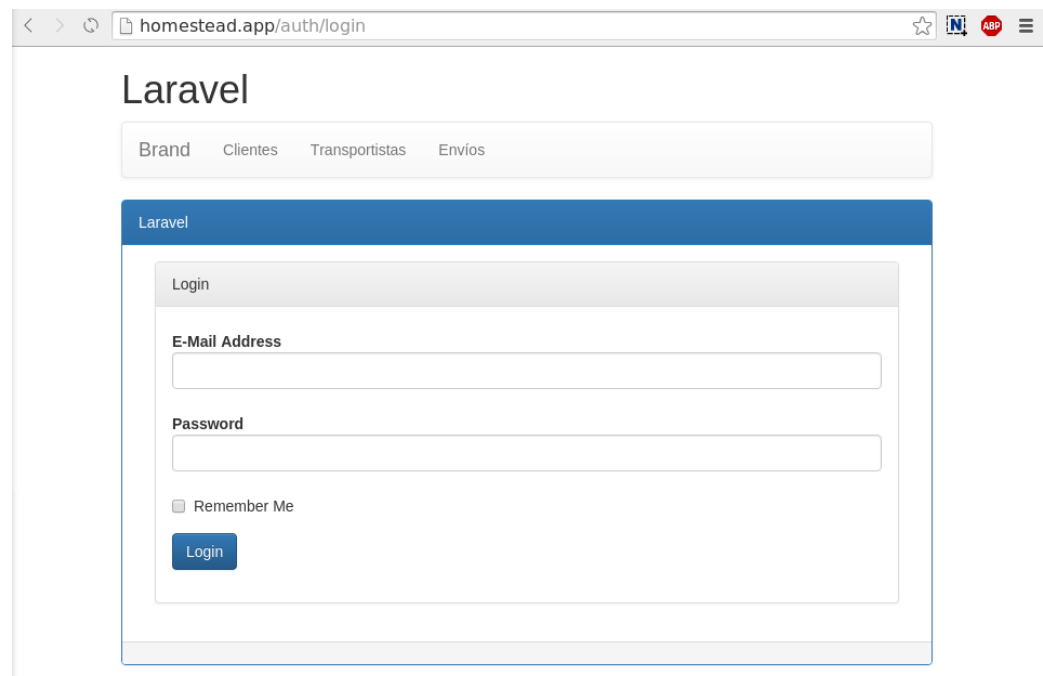
Capturas webApp

Login



A screenshot of a web browser displaying the Laravel login page. The browser's address bar shows the URL `homestead.app/auth/login`. The page has a header with the word "Laravel" and a navigation bar with links: "Brand", "Clientes", "Transportistas", and "Envíos". Below the navigation bar is a blue header with the word "Laravel". The main content area is a white box with a "Login" title. It contains two input fields: "E-Mail Address" and "Password". Below these fields is a checkbox labeled "Remember Me" and a blue "Login" button.

Transportistas



A screenshot of a web browser displaying the Laravel login page. The browser's address bar shows the URL `homestead.app/auth/login`. The page has a header with the word "Laravel" and a navigation bar with links: "Brand", "Clientes", "Transportistas", and "Envíos". Below the navigation bar is a blue header with the word "Laravel". The main content area is a white box with a "Login" title. It contains two input fields: "E-Mail Address" and "Password". Below these fields is a checkbox labeled "Remember Me" and a blue "Login" button.

homestead.app/transportista/view/6

Ficha transportista

Brand Clientes Transportistas Envíos

Información sobre 34567800L

Info Editar Envíos

NIF 34567800L

Apellidos 22ddfd

Nombre aaa

Teléfono 3333

Volver atrás

homestead.app/transportista/edit/1

Ficha transportista

Brand Clientes Transportistas Envíos

Transportista 57432844G

Info Editar Envíos

NIF 57432844G

Apellidos Caldas Sánchez

Nombre Rolando

Teléfono 986234408

Guardar

homestead.app/transportista/envio/1

Envíos Transportista

Brand Clientes Transportistas Envíos

Envíos del transportista 57432844G

Info Editar Envíos

Fecha	Destinatario	Dirección	Acciones
2013-01-29T12:34:56Z	Nombre	d fff aldfjlajdf	QVer
2015-11-21T09:56:55.950549Z	jkhkh	adfadf	QVer

Nuevo transportista

homestead.app/transportista/envio/1/info/1

Envíos Transportista

Brand Clientes Transportistas Envíos

Envíos del transportista 57432844G

Info Editar Envíos

Fecha	Destinatario	Dirección
2013-01-29T12:34:56Z	Nombre	d fff aldfjlajdf

Actividad del envío

Fecha	Observaciones	Recepción
2013-01-29T12:34:56Z		N/A
2015-11-21T20:26:15.448000Z		
2015-11-21T20:29:55.233000Z		

Envíos

homestead.app/envio/

Envíos

Brand Clientes Transportistas Envíos

Listado de envíos

ID	Fecha	Destinatario	Dirección	Acciones
1	2013-01-29T12:34:56Z	Nombre	d fff aldfjlajdf	Editar QVer Cliente Transportista
2	2015-11-21T09:56:55.950549Z	jkhkh	adfadf	Editar QVer Cliente Transportista
3	2015-11-21T10:10:26.916506Z	hkhk	h kh kg	Editar QVer Cliente Transportista
4	2015-11-21T11:22:27.590189Z	hkjhkj	hkhks	Editar QVer Cliente Transportista

[Nuevo envío](#)

homestead.app/envio/view/1

Ficha envío

Brand Clientes Transportistas Envíos

Información sobre #1

Info Etiqueta Editar Cliente Transportista

Destinatario Nombre

Dirección d fif aldfjlajdf

Fecha 2013-01-29T12:34:56Z

homestead.app/envio/qr/1

Ficha envío

Brand Clientes Transportistas Envíos

Información sobre #1

Info Etiqueta Editar Cliente Transportista



Nombre
d fif aldfjlajdf

Capturas aplicación móvil

Transporte APP

Destinatario

Manuel Jiménez

Calle Julián Estévez 32,
bajo b

Ramón Martínez

Avenida Gran Vía 124,
bajo a

Transporte APP

☐ Paquete entregado

Observaciones

No había nadie en casa

Firma

cada Casa caza

q w e r t y u i o p

a s d f g h j k l ñ

z x c v b n m

7123 , . ←

Transporte APP

☒ Paquete entregado

Observaciones

Ok

Firma

Manual de usuario

Se crean dos manuales de uso, uno para la aplicación web y otro para la móvil. Se adjuntan con el presente documento.

- Manual webApp: PI3_RolandoCaldasSanchez-ManualUsuarioWebApp.pdf
- Manual app móvil: PI3_RolandoCaldasSanchez-ManualUsuarioAppMobile.pdf

4. Seguimiento y control

4.1 Valoración del proyecto

Con el proyecto inmerso en la fase de pruebas y listo para arrancar la experiencia piloto, se procede a valorar el proyecto en base a lo esperado y lo existente.

Que el desarrollo sea propio de la empresa y no se trate de un desarrollo contratado a modo de servicio hace que bajo el prisma del precio la inversión haya sido positiva, puesto que el proceso de implementación de un sistema como el desarrollado es altamente costoso, no sólo en la puesta en marcha sino también una vez esté funcionando, debido al mantenimiento y mejoras (costes variables) como a los costes fijos derivados de la infraestructura que el proveedor nos facturaría.

Otro de los aspectos positivos del proyecto ha sido la rapidez en el desarrollo, puesto que tener un equipo de profesionales dedicados exclusivamente al proyecto es una ventaja poco habitual debido a que las empresas de software suelen abordar varios proyectos al mismo tiempo.

Entrando ya algo más en la valoración técnica del proyecto, se destaca positivamente el uso de las últimas tecnologías. Las tres aplicaciones que conforman el proyecto se apoyan en frameworks de reconocido prestigio y muy enfocados en el desarrollo moderno, especialmente en el caso del framework Laravel, que no sólo obliga a utilizar las últimas versiones del lenguaje PHP sino que también es el primero realizado en PHP que aborda, de forma seria, la necesidad de evolucionar el tradicional patrón MVC.

La decisión de desacoplar lo que sería el modelo o capa de datos y dotarla de una interfaz de API CRUD también se valora positivamente, puesto que aumenta su rendimiento y disponibilidad, especialmente gracias a haberse desarrollado en Django y su REST Framework, desarrollados en Python, un lenguaje que presenta un mejor rendimiento que otras alternativas a la hora de soportar gran cantidad de accesos, gracias también a que Django tiene su propio servidor web ultra-ligero.

La valoración de la aplicación móvil es la que más controversia genera. El hecho de haber optado por una solución híbrida la descarta como solución óptima tanto por tiempos de respuesta como por tecnología. A pesar de ello, se considera la solución adecuada, puesto que la realidad de poder generar la versión para iOS en cuestión de minutos la hace extremadamente idónea. Apostando por la solución híbrida se ha conseguido generar una aplicación móvil persistente en el tiempo sin que el proceso de cambio de plataforma móvil suponga un freno o un trauma a la empresa.

Por lo tanto, de cara a justificar el acierto del proyecto debemos basarnos en el bajo coste del proyecto (si lo valoramos a largo plazo), la rapidez en el desarrollo como la velocidad a la hora de servir el contenido, debido en parte por el uso de las últimas tecnologías y versiones de software, junto con una interfaz orientada a la eficiencia en tiempos de carga. Por supuesto, que la solución, aunque mejorable, aporta una modernización vital en la gestión del trabajo, abriendo una puerta a innumerables

mejoras que incidan directamente sobre el rendimiento de la empresa; algo que sin una solución como la planteada, sería totalmente imposible.

Al tratarse de un proyecto interno, no es preciso elaborar ningún acuerdo de servicio o garantizar niveles de servicio o resolución de errores, puesto que el equipo de desarrollo está en la empresa para ello.

Sin embargo, cabe recalcar que la cobertura del servicio sería total durante la jornada laboral, de modo que tanto en los momentos en los que los clientes pueden solicitar envíos como los transportistas realizar entregas, el equipo de desarrollo estará siempre disponible para mantener el servicio activo y poder resolver los posibles problemas que surjan.

Por último, para poder mejorar y evaluar correctamente el servicio. El jefe de proyecto tendrá reuniones periódicas (mínimo una semanal) con el personal que forme parte de la experiencia piloto y se elaborarán informes de situación para que durante la experiencia piloto se detecten los puntos débiles de la solución y puedan ser solucionados. Durante la primera reunión, instruirá a los usuarios sobre el funcionamiento de la aplicación, siguiendo de forma presencial con cada usuario los diferentes pasos contemplados en el manual de uso. Los propios miembros de la experiencia piloto, posteriormente, serán los que formen al resto de personal.

Una vez la solución pase a ser utilizada por todo el personal de la empresa, se les enviará una encuesta de calidad al mes de estar trabajando con la solución.

4.2 Incidencias

La gestión de las incidencias es una cuestión importante y siempre sensible. Mientras dure la experiencia piloto, el jefe de proyecto será el encargado de gestionar las incidencias como si éstas fuesen incidencias del desarrollo del proyecto; de este modo los miembros de la experiencia piloto tendrán contacto directo con desarrollo.

Esta situación cambiará radicalmente cuando se entre en producción. Una vez la solución se implante de forma masiva, se creará un departamento destinado a la gestión de incidencias. Los usuarios de la aplicación (administrativos y transportistas) tendrán contacto telefónico con el personal de incidencias, quienes a su vez tendrán acceso a la herramienta Gitlab (<https://gitlab.com/>).

Desde Gitlab se crearán las incidencias (issues) dentro del proyecto correspondiente (api, webapp, app mobile). Cada incidencia se etiquetará en base a su naturaleza (error, mejora, información) y a su gravedad (normal, grave, crítica). Antes de crear una incidencia, el departamento de soporte comprobará si ya existe registrada en el sistema y si está resuelta o no, para aportar la solución. En caso de no existir, procederá a introducirla en el sistema, con las etiquetas correspondientes.

El jefe de proyecto tendrá la responsabilidad de revisar Gitlab para ver las incidencias abiertas, marcarlas como confirmadas o no y, de ser confirmadas, asignarlas a un miembro de desarrollo, quien será el encargado de solventar la incidencia.

Una vez la incidencia esté solucionada, el issue se cerrará, recibiendo soporte un aviso para trasladar la nueva situación de la incidencia al reporter.

Las soluciones que se apliquen desde desarrollo, puesto que ya estaríamos hablando de un proyecto en producción, no se implantarían directamente, sino que el desarrollador incorporaría la solución a la aplicación mediante un pull-request, siendo el jefe de proyecto el encargado de revisar ese pull-request y aceptarlo como solución o no.

Cuando una incidencia sea crítica, el usuario afectado por la misma deberá continuar con su trabajo de forma externa a la aplicación, utilizando el sistema tradicional del papel y será el departamento de desarrollo quien trasladará esa información al sistema. Para que esta situación ocurra, desde soporte debe indicarse al usuario que se trata de una incidencia crítica.

También se contempla incorporar en los servidores un sistema de monitoreo (Monit) que compruebe cada minuto que todos los servicios estén levantados, procediendo a levantar cualquier servicio que estuviese caído y a enviar una alerta por email a desarrollo informando de la situación. Además de comprobar servicios, también comprobará la huella md5 de los principales ejecutables, para garantizar su integridad.

4.3 Cambios

Aunque la solución está realizada con las últimas versiones de las tecnologías utilizadas, dada la naturaleza de las mismas éstas viven en un proceso de evolución continua y sus cambios pueden afectar a nuestra solución.

Todo el software utilizado sigue la nomenclatura tradicional de versiones en tres niveles (x.x.x). Cualquier actualización dentro de la rama estable será aplicada en los servidores, por lo que nuevas versiones de PHP, MySQL, etc serán aplicadas directamente a través del gestor de paquetes apt-get

En el caso de que, por ejemplo, aparezca una nueva versión de Django o Laravel, no se realizarán actualizaciones, salvo que éstas sean por motivos de seguridad. De forma anual, se valorará la posibilidad de actualizar las versiones de los frameworks.

Si por motivos de errores o ampliación de recursos las aplicaciones deben sufrir una migración de servidor, el equipo de desarrollo será el encargado de realizar dicha migración desde el inicio, esto significa que se encargarán de la instalación y puesta en marcha de los nuevos equipos, utilizando la misma configuración que en los activos.

La escalabilidad y mejoras en la solución son siempre factibles debido a que se han utilizado frameworks de desarrollo pensados para ser escalables. Ante posibles problemas de conectividad, se analizará la fuente del problema, optando inicialmente por la mejora de la RAM y discos en los equipos. De no ser suficiente, especialmente en el servidor de la API, se valoraría montar la API en varios servidores y realizar un balanceo de carga por IP. Sin embargo, se considera una situación poco probable debido al número limitado de trabajadores existentes en la empresa.

Respecto a las mejoras, como un proyecto a largo plazo, el software se irá mejorando con las funcionalidades previstas y con las posibles mejoras que vengan solicitadas por los administrativos y/o transportistas en base a la experiencia que adquieran en el uso diario de la aplicación. Se contempla la existencia de una actualización anual con mejoras incorporadas, pudiendo llegar a dos actualizaciones con mejoras como máximo. El resto de actualizaciones serán habitualmente transparentes al usuario de la webapp (el usuario móvil siempre se enterará) y su finalidad será a aplicación de correcciones en base a las incidencias.

4.4 Pruebas y soporte

Pruebas unitarias

Las pruebas unitarias buscan comprobar que nuestro código funcione correctamente. En el proyecto, la zona más crítica es la aplicación API, puesto que es la aplicación que realmente maneja la información, relegando al resto de aplicaciones a ser meros transmisores de los datos.

La API está desarrollada por un fuerte sistema de herencia y tanto Django como su REST Framework tienen desarrolladas sus propias pruebas unitarias y su buen funcionamiento está garantizado.

Debido a que el margen de error en la API es limitado y que para hacer las pruebas unitarias de software deberíamos aprender el framework existente en python para ello, se decide emular las pruebas unitarias directamente haciendo las peticiones a la API directamente. En un futuro se plante incorporar test unitarios, cuando el tiempo del proyecto permita aprender dicha tecnología. La misma situación ocurre con Laravel y phpunit.

Test API

Acción	Login	Datos envío	Resultado
/cliente.json	No	N/A	Error
/cliente.json	Si	N/A	Listado OK
/cliente.json	Si	POST con todos los datos de nuevo cliente	OK
/cliente.json	Si	POST con el campo de CIF erróneo	Error
/cliente.json	Si	POST con campos obligatorios sin incluir	Error
/transportista.json	No	N/A	Error
/transportista.json	Si	N/A	Listado OK
/transportista.json	Si	POST con todos los datos de nuevo transportista	OK
/transportista.json	Si	POST con el campo de NIF erróneo	Error

Acción	Login	Datos envío	Resultado
/transportista.json	Si	POST con campos obligatorios sin incluir	Error
/envio.json	No	N/A	Error
/envio.json	Si	N/A	Listado OK
/envio.json	Si	POST con todos los datos de nuevo envío	OK
/envio.json	Si	POST con identificador de transportista erróneo	Error
/envio.json	Si	POST con identificador de cliente erróneo	Error
/envio.json	Si	POST con formato de fecha erróneo	Error
/entrega.json	No	N/A	Error
/entrega.json	Si	N/A	Listado OK
/entrega.json	Si	POST con todos los datos de nueva entrega	OK
/entrega.json	Si	POST con identificador de envío erróneo	Error
/entrega.json	Si	POST con formado de fecha erróneo	Error
/cliente/1.json	No	N/A	Error
/cliente/1.json	Si	GET	Info cliente
/cliente/1.json	Si	POST para modificar cliente	Error
/cliente/1.json	Si	PUT con datos correctos para modificar cliente	OK
/cliente/[fakeld].json	Si	Datos y protocolo correctos	Error
/envio/1.json	No	N/A	Error
/envio/1.json	Si	GET	Info cliente
/envio/1.json	Si	POST para modificar envío	Error
/envio/1.json	Si	PUT con datos correctos para modificar envío	OK
/envio/[fakeld].json	Si	Datos y protocolo correctos	Error
/envio/cliente/1.json	Si		OK
/envio/transportista/1.json	Si		OK

Se han realizado más pruebas no contempladas en la tabla, con las que se cubrieron las diferentes acciones que la API soporta.

Test WebApp

Acción	Login	Datos envío	Resultado
/cliente	No	N/A	Redirección a la pantalla e login
/auth/login	No	Datos erróneos	Vuelta a login mostrano mensaje error
/auth/login	No	Datos correctos	OK
/cliente	Si		OK
/cliente/add	Si		Formulario
/cliente/add	Si	POST correctos	OK y ficha cliente
/cliente/add	Si	POST erróneo	Error y formulario

Se han realizado más pruebas no contempladas en la tabla, con las que se cubrieron las diferentes acciones que la APP soporta.

Test Mobile App

También se han realizados pruebas en la aplicación móvil. Durante las pruebas se detectaron errores en la aplicación que mostramos en la siguiente tabla

Acción	Datos envío	Resultado
Enviar entrega finalizada	Datos correctos	Información de la entrega enviada y almacenada, pero el envío no se marca como enviado
Escanear código QR		El código se escanea y detecta correctamente, pero no se accede a la pantalla de entrega
Vuelta a lista envíos	Datos correctos	Los envíos no se vuelven a cargar, se muestran siempre los cargados en la primera vez

De los errores detectados se corrigieron los errores de “Enviar entrega finalizada” y “Vuelta a lista envíos” quedado, a día de hoy, pendiente de solucionar el problema con el escaneo del código QR

Pruebas de carga

Para realizar los test de estrés se ha utilizado la herramienta de apache “ab” que permite lanzar múltiples peticiones simultáneas retornando un informe sobre el tiempo total.

Se han realizado las pruebas con 100 500 y 5000 peticiones simultáneas, correspondiéndose con el nivel de carga habitual, el de fechas especiales (navidades, etc) y una simulación muy por encima de la situación real.

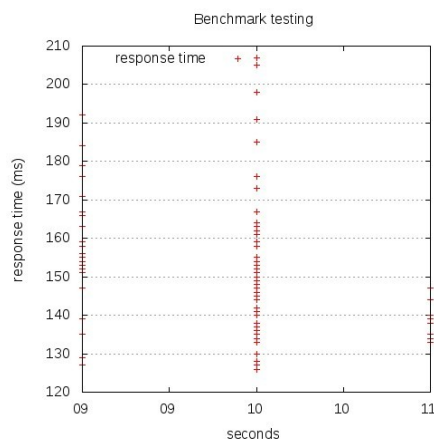
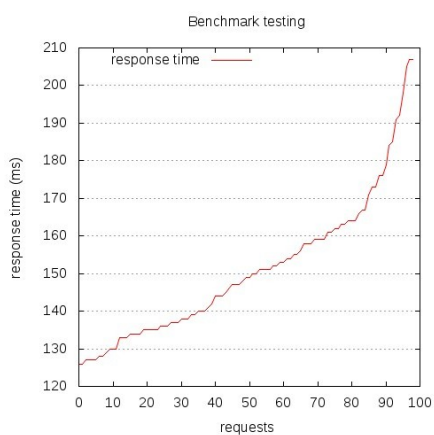
Pruebas API

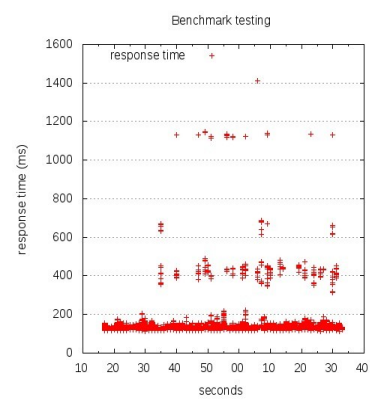
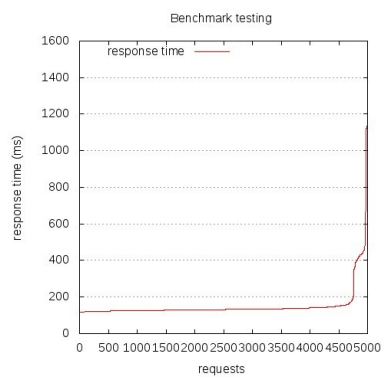
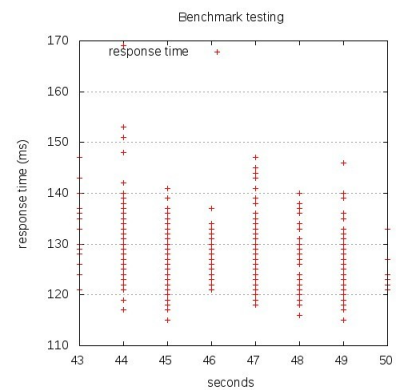
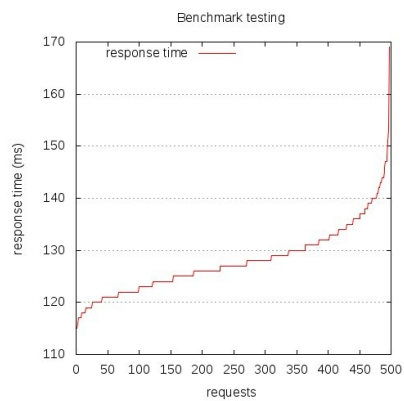
Peticiones	Total segundos	Peticiones / sg	Tiempo / Pet [ms]
100	1.596	62.66	15.960
500	6.467	77.31	12.934
5000	75.619	66.12	15.124

Pruebas WebApp

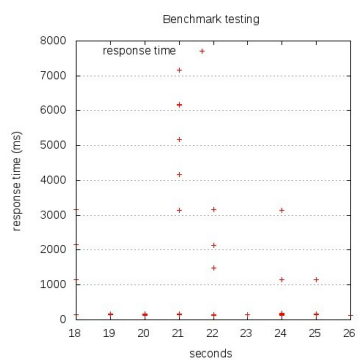
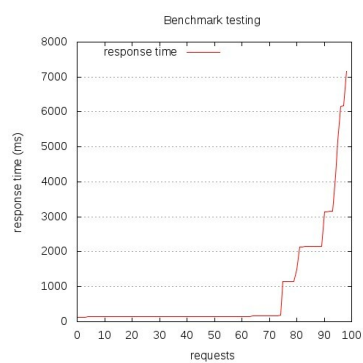
Peticiones	Total segundos	Peticiones / sg	Tiempo / Pet [ms]
100	9.324	10.72	93.241
500	51.877	9.64	103.753
5000	293.626	13.62	73.406

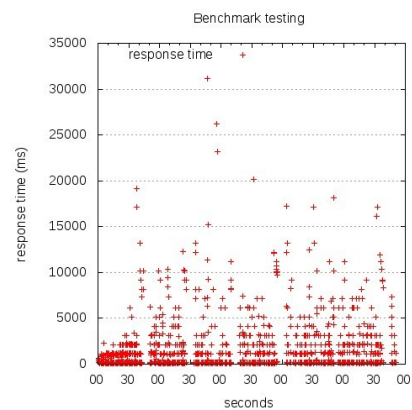
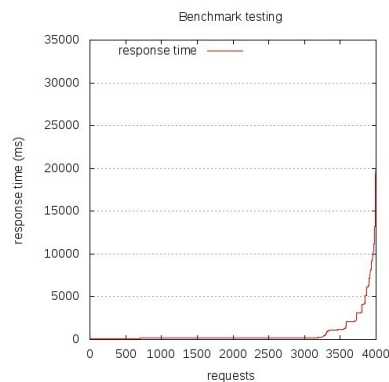
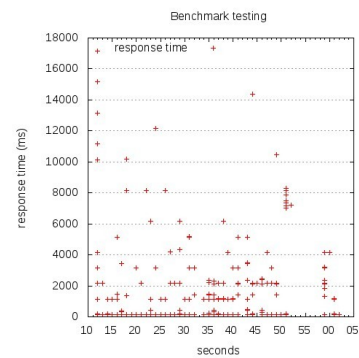
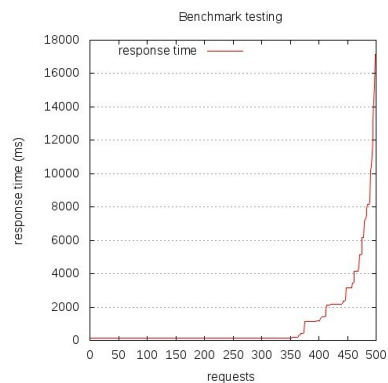
Gráficas Pruebas API





Gráficas Pruebas WebApp





Conclusiones test de carga

Mirando los datos extraemos como conclusión algo que ya habíamos supuesto, por lo que se confirma nuestra supuesto.

La aplicación API en Django tiene una gran capacidad de respuesta y, además, hay muy poca dispersión en los tiempos de respuesta entre las peticiones. Esto conlleva que nuestra API es estable entre las peticiones.

No podemos decir lo mismo de nuestra webapp, cuyo tiempo de respuesta es muy superior al de la API y, lo que es más problemático, existe mucha dispersión entre las peticiones.

La decisión de optar por Python y Django para crear la API CRUD y que ésta sirva de modelo para nuestras aplicaciones es todo un éxito puesto que tiene capacidad más que suficiente de atender las peticiones tanto de la webapp como de la aplicación móvil, de haber optado por la solución API REST proporcionada por Laravel, estaríamos ante un grave problema de rendimiento futuro.

Podríamos pensar que, por lo tanto, la decisión de utilizar Laravel para llevar a cabo el desarrollo de la aplicación web ha sido un error, por los mismos motivos que convierten la decisión de la API en un acierto. Sin embargo, la aplicación web no va a soportar un gran número de peticiones simultáneas, puesto que la propia naturaleza de la aplicación hace que el usuario pase varios segundos en cada pantalla, no presentándose, realmente, un posible problema de rendimiento. Además, las pruebas se han realizado con un servidor Apache2, algo que no afecta a la API porque corre con su propio servidor web. Apache2 es un servidor web clásico, pero se caracteriza por tener un importante cuello de botella bajo estrés. Si en algún momento hubiese problemas de rendimiento en la webApp, cambiando Apache2 por Nginx se solucionará.

Pruebas de interfaz

Se decide hacer pruebas de interfaz de la aplicación web, tanto desde el punto de vista técnico como de usuario. Para las pruebas de interfaz de usuario, se ha facilitado a una muestra del personal, datos de acceso para la webapp y la URL. Junto con estos datos se le facilitó una pequeña encuesta anónima para obtener sus impresiones. A continuación mostramos alguna de las respuestas



Pregunta	Usuario 1	Usuario 2	Usuario 3
¿Ha sido difícil acceder? 0: muy fácil – 5: muy difícil	0	0	0
¿Qué secciones tiene el sitio?	Clientes, transportistas, envíos	Clientes, transportistas, envíos	Clientes, transportistas, envíos
¿Los iconos te han servido de apoyo visual? 0: para nada – 5: vitales	2	2	4
¿La navegabilidad es sencilla? 0: imposible – 5: muy fácil	4	4	5
Qué valoras positivamente	La ficha de cliente y poder acceder a toda su información rápidamente	Poder ver todas las actividades de un envío	La sección de clientes
Qué valoras negativamente	Dónde me desconecto?	No hay opción de desconectar	La sección de envíos me confunde, porque puedo acceder a lo mismo desde clientes y transportistas

Respecto a las pruebas desde el punto de vista técnico, se ha cuidado los estándares web HTML5, se ha colocado un etiquetado semántico y se ha creado pensando en los entornos de escritorio, al no ser una aplicación para utilizar desde el móvil, aunque se han aplicado conceptos básicos de responsive design.

También se ha pasado el test de google PageSpeed, obteniendo una puntuación para escritorio de 83 sobre 100

PageSpeed Insights

http://app.demosonline.org/auth/login ANALIZAR


 Móvil  Ordenador

83 / 100 Resumen de sugerencias

Elementos que debes corregir:
 Eliminar el JavaScript que bloquea la visualización y el CSS del contenido de la mitad superior de la página
[Mostrar cómo corregirlo](#)

Elementos que puedes plantearte corregir:
 Reducir el tiempo de respuesta del servidor
[Mostrar cómo corregirlo](#)

8 reglas aprobadas
[Mostrar detalles](#)



Los elementos de mejora proporcionados por Google se basan en minificar el código JS y CSS además de sugerir que el JS se incluya de forma no bloqueante.

Aunque estas recomendaciones son útiles, no se contempla incorporarlas actualmente porque la relación esfuerzo/beneficio para una aplicación de esta naturaleza no se considera lo suficientemente potente.

Para ilustrar otras pruebas de interfaz pasadas correctamente utilizares alguna imagen de ejemplo:

Cientes

Brand Cientes Transportistas Envios

El menú superior está siempre presente y lleva a las secciones principales

los listados incluyen las filas con diferente fondo para las pares e impares

Teléfono	Envios	Acciones
54321676T Albañiles del Noroeste	986000002 0	Editar QVer Envios Nuevo envio
54321677R Mamparas Fernández	986000001 0	Editar QVer Envios Nuevo envio
54321678W Reparaciones Antonio C.B.	986000000 2	Editar QVer Envios Nuevo envio

[Nuevo cliente](#)

Ubicación consante para la acción principal de la pantalla: uniformidad

Proyecto DAM 2015 - © Rolando Caldas

Cientes

Brand Clientes Transportistas Envíos

Todas las acciones se ubican como botones. Los iconos utilizados tienen un sentido y se utilizan los mismos en todo momento

Cif	Nombre	Telefono	Envios	Acciones
54321676T	Albañiles del Noroeste	986000002	0	Editar QVer Envios Nuevo envio
54321677R	Mamparas Fernández	986000001	0	Editar QVer Envios Nuevo envio
54321678W	Reparaciones Antonio C.B.	986000000	2	Editar QVer Envios Nuevo envio

[Nuevo cliente](#)

Proyecto DAM 2015 - © Rolando Caldas Sánchez

Nuevo cliente

Brand Clientes Transportistas Envíos

cif: NIF / CIF not valid

Todos los mensajes de sistema se muestran entre el menú principal y el contenido: Siempre visibles

TransApp

Los errores se muestran con fondo rojo y las acciones exitosas con fondo verde

la X permite cerrar el mensaje

D.Social Fadsfd

Teléfono 3

Dirección f

Si un formulario retorna error, los campos se cubren solos con la información enviada.

[Crear cliente](#)

Proyecto DAM 2015 - © Rolando Caldas Sánchez

Ficha cliente

Brand Clientes Transportistas Envíos

Información sobre 54321678W

Info [Editar](#) [Nuevo Envío](#) [Envíos](#)

CIF 54321678W

D.Social Reparaciones Antonio C.B.

Teléfono 986000000

Dirección Avda Europa 23, 34213

Los iconos se mantienen por toda la app

cuando una sección tiene varias pantallas, se agrupan siempre por pestañas.

[Nuevo cliente](#)

Proyecto DAM 2015 - © Rolando Caldas Sánchez

Acuerdo de servicio

Al tratarse de un servicio interno, no existe un contrato como tal; sin embargo el departamento de desarrollo adquiere un compromiso con la empresa por el cual tienen que prestar soporte técnico (con la intermediación de soporte) durante la jornada laboral del personal, especialmente la de los transportistas, inclusive las épocas de alta disponibilidad en las cuales los transportistas tienen un horario ampliado. En esas épocas, al menos un miembro de desarrollo deberá estar de guardia.

También se adquiere el compromiso de dar contestación a las incidencias en menos de 24 horas, lo que no incluye que en 24 horas deban estar solucionadas las incidencias. En el caso de incidencias críticas, desarrollo deberá dar solución en menos de 72 horas. Una incidencia crítica es marcada como tal por el jefe de proyecto o la figura que lo sustituya en sus periodos estivales.

Otro compromiso adquirido es la obligatoriedad de mantener los servidores actualizados y aplicar medidas preventivas ante posibles problemas de carga.