

**ETSISI
UPM**



PROYECTO FIN DE MÁSTER

Máster en Ingeniería Web

Aplicación Web y Móvil usando Android y AngularJS contra Spring Web Services

Alumno: Javier Combarros Gómez

Tutor: Jesús Bernal Bermúdez

Julio, 2015

Agradecimientos

A mis padres, Esteban y Charo y a mi hermana Belén; a mis amigos de toda la vida, a los que conocí durante en mi Erasmus en Dinamarca y los que hecho durante la universidad, el máster y el trabajo; a los profesores de la UPM que he tenido en el grado y el máster y a los la SDU que tuve en mi Erasmus; a los compañeros del trabajo y todas las personas que me han ayudado a convertirme en un profesional de esto: el desarrollo Software.

Índice

Abstract.....	6
Resumen	7
Introducción	8
Objetivos.....	9
Tecnologías y herramientas.....	10
Tecnologías comunes	11
Herramientas comunes	18
Aplicación servidora - Tecnologías.....	22
Aplicación Servidora - Herramientas	27
Aplicación Web - Tecnologías	31
Aplicación Web - Herramientas	37
Aplicación móvil - Tecnologías.....	42
Metodología	44
Modelado y Requisitos	45
Análisis y Diseño	53
Desarrollo	64
Pruebas.....	70
Lecciones aprendidas y Futuras mejoras	71
Conclusión	74
Bibliografía.....	75
Apéndice	79
Versiones.....	80
Casos de uso detallados	82
Mockups.....	101
Instalación y configuración	110
Capturas de pantalla	115

Abstract

En los últimos años los avances en ingeniería web han permitido que las páginas web que se crean sean más potentes visual y funcionalmente.

Debido a este continuo avance, existen una gran cantidad de tecnologías en el lado de cliente que hacen que cada vez sea más complicado desarrollar una página de manera ordenada. Con la ayuda de AngularJS se busca unir todas esas tecnologías, reducir su complejidad y separar conceptos mediante el patrón MVC para que según aumente el tamaño del código no se haga inmanejable su desarrollo.

En el lado servidor ocurre algo parecido, cada vez las aplicaciones son más grandes y hacen uso de diversas tecnologías lo cual implica que se necesiten de más recursos y por tanto la capacidad de procesamiento se ve afectada. Mediante los Web Services seremos capaces de liberar en cierta medida al servidor, haciendo las páginas más ágiles.

De esta manera y al disponer de Web Services, no estamos atados a una determinada tecnología en el lado cliente: tanto un navegador como un teléfono Android pueden lanzar peticiones contra un API siendo transparentes para el servidor.

Resumen

Durante la elaboración de este proyecto se ha buscado realizar tanto una Aplicación Web como una Móvil para gestionar Notas y Recordatorios usando AngularJS, y Android respectivamente como clientes y que se comuniquen con una tercera aplicación servidora mediante el uso de servicios web por REST usando Spring.

Durante su desarrollo se ha querido analizar la forma en la que dos aplicaciones independientes interactúan para obtener un resultado común: en el caso particular, una aplicación web de gestión de notas y recordatorios. Estas dos aplicaciones se encuentran en ambos extremos de la comunicación web: una como cliente usando AngularJS y otra como servidora ofreciendo una API implementada usando Servicios Web mediante Spring.

Además se ha buscado seguir una arquitectura que permita intercambiar el cliente sin que el servidor se vea afectado. Ese ha sido el motivo por el que se ha desarrollado otra aplicación cliente para dispositivos móviles Android que se comunique de igual manera con el servidor. Esta arquitectura hace uso de los Servicios REST.

Introducción

Tras haber realizado hasta ahora aplicaciones web usando tecnologías tales como JSP, PHP o .NET que principalmente llevan toda la carga de procesamiento en el lado servidor, con el desarrollo de este proyecto se ha buscado balancear dicha carga entre el cliente y el servidor. Por ello mediante AngularJS y Servicios Web conseguimos que el cliente tenga una labor más importante y reconocida, liberando de esta manera al servidor para que pueda recibir un mayor número de peticiones.

Además en muchos casos el uso de estas tecnologías que corren en el servidor, tenían el problema de que la capa de presentación y la capa de negocio estuviera muy ligadas y que fuera complicado reutilizarlas. Al usar AngularJS y tecnologías propias del cliente, tales como HTML, CSS y Javascript, conseguimos que dicha separación de conceptos quede más diferenciada.

Como muestra de ello, se ha querido elaborar una aplicación cliente adicional para dispositivos móviles usando Android y que utilice los servicios sin necesidad de ninguna modificación.

Puesto que la información no se encuentra en posesión del cliente sino que está almacenada en el servidor y viaja continuamente de uno a otro, se ha tenido cuidado en que la comunicación sea lo más segura posible implementando medidas de seguridad tales como control de usuario y autorización basada en tokens.

Con todo ello funcionando junto se ha conseguido elaborar una aplicación que gestione notas y recordatorios. Además gracias al uso tecnologías web y móviles conseguimos una aplicación global que nos permite una gran movilidad al poder usarla en cualquier sitio y dispositivo. Por otro lado, la información ya no se encuentra en nuestro dispositivo físico, sino en la nube lo que nos permite disponer de ella en cualquier lugar. Ello implica un gran esfuerzo en seguridad para que esa información esté disponible siempre que el usuario quiera consultarla y que la misma sea de exclusivo acceso por parte del autor.

Objetivos

Con la elaboración de este proyecto se ha buscado un aprendizaje en diversas tecnologías, herramientas y arquitecturas. También se ha buscado poner en práctica dichos conocimientos como conjunto para que colaboren entre ellos, analizándolos y adecuándolos en base a las necesidades de la aplicación:

- Crear una API usando Web Services con Spring
- Hacer uso de dicha API desde diversos clientes
- Utilizar el framework AngularJS para mostrar la información en un navegador web
- Crear una aplicación Android que muestre la información en un dispositivo móvil
- Configurar un proyecto web servidor usando Maven como gestor de dependencias
- Conseguir abstraerse de la capa de persistencia en el servidor usando un ORM
- Usar del patrón MVC en diferentes contextos y variantes (MVP y MVVM)
- Configurar diversos entornos y proyectos
- Implementar medidas de seguridad sobre el acceso a la API
- Integrar todo lo antes mencionado en una única aplicación
- Crear un modelo arquitectónico siguiendo una metodología
- Definir las vistas de la arquitectura a través de diagramas UML

Tecnologías y herramientas

En la elaboración de las distintas aplicaciones que se han realizado en transcurso de este proyecto, se han necesitado gran variedad de tecnologías y herramientas para llevar a cabo su desarrollo.

Una parte importante del proyecto ha sido estudiar, probar y configurar cada una de estas tecnologías y herramientas además de integrarlas todas juntas para que en de manera conjunta puedan ofrecer la base que con el tiempo ha permitido construir estas aplicaciones.

Puesto que ha habido tres aplicaciones, agruparemos todas estas tecnologías en torno a la aplicación para la cual trabajan. De esta manera la lista de aplicaciones se agrupa entorno a:

- Aplicación servidora
- Aplicación web
- Aplicación móvil

Junto a ellas añadimos un cuarto grupo que recoge aquellas tecnologías y herramientas comunes al desarrollo.

En este apartado describiremos brevemente cada una de las tecnologías y herramientas usadas, así como posibles alternativas.

Las versiones de cada una de las tecnologías y herramientas que han aparecido durante el desarrollo de este proyecto se definen en los apéndices de este documento.

Tecnologías comunes

- Servicios Web. REST

Servicios Web: Es un método de comunicación entre dos dispositivos a través de internet.

En un extremo se encuentra un sistema software que publica en una dirección de internet un conjunto de recursos a cualquier otro sistema que lo necesite siempre y cuando respete la arquitectura de los Servicios Web definida en la interfaz bajo el formato WSDL. El conjunto de servicios ofrecido se conoce como API y su servicio deberá estar siempre activo.

Por lo general la comunicación se realizará sobre el protocolo HTTP. De esta manera otros sistemas se podrán conectar al primer sistema. Enviarán solicitudes usando un protocolo de comunicación (REST o SOAP), las cuales recibirán respuesta por parte del servidor enviado de vuelta los recursos solicitados.

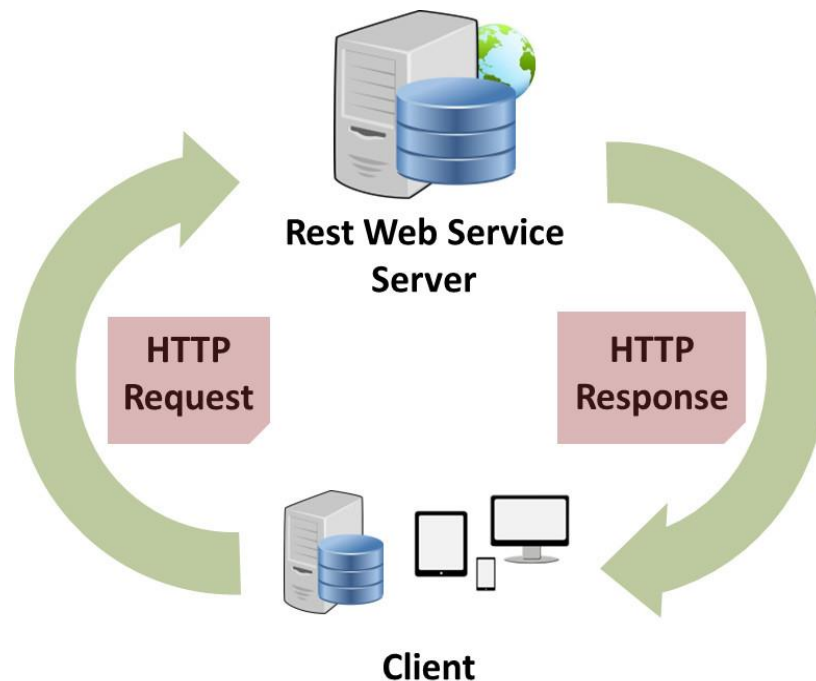
Los lenguajes de programación y plataformas utilizados en las aplicaciones participantes son indiferentes.

Los web services son ofrecidos como pila de protocolos y estándares. Estos son:

- **Servicio de transporte:** capa responsable del transporte de mensajes entre aplicaciones. Los protocolos habituales son HTTP, SMTP y FTP. En esta aplicación al ser una aplicación web lo haremos sobre el protocolo HTTP.
- **Mensajería:** capa encargada de la codificación de los mensajes. Los formatos de mensaje más comunes son XML y JSON. Los protocolos habituales de comunicación REST y SOAP. En esta aplicación se ha apostado por el uso de REST. A continuación se hace una comparación entre ambos.
- **Descripción del servicio:** se utiliza para describir la interfaz pública de un Servicio Web mediante el protocolo WSDL.
- **Descubrimientos de servicios:** se encarga de mantener un registro de los servicios web disponibles en la web, así como su localización y su descripción y hacer así más fácil su consulta. Se utiliza el estándar UDDI.

Los dos protocolos de mensajería mencionados anteriormente son:

- SOAP se caracteriza como paradigma estricto de mensajería. Es el sucesor de otro protocolo previo: el XML-RPC. SOAP solo utiliza mensajes XML, esto implica que sea lento. Puede viajar sobre cualquier protocolo de transporte haciéndolo muy versátil. Es un protocolo sin estado, esto significa que entre dos peticiones consecutivas no se mantiene ningún tipo de información, son independientes, toda la información necesaria tiene que viajar en cada una. SOAP no ofrece datos sino operaciones.
- Como alternativa a SOAP existe REST. REST es un estilo arquitectónico ligero que define unas guías de buena práctica para crear Web Services que sean escalables, mantenibles y que tengan un mejor rendimiento. Frente a las operaciones de SOAP, REST ofrece recursos y para recuperarlos utiliza los métodos GET, POST, PUT, DELETE, los cuales son más adecuados para la comunicación web en navegadores. Además de XML, permite también otros formatos de mensaje como JSON, mucho más ligeros y legibles. Al igual que SOAP es sin estado. En cambio una ventaja sobre SOAP es que las peticiones son cacheables. Al ser más simple no es necesario generar WSDL pues es auto-descriptible.



Como hemos mencionado, en el caso particular de nuestro proyecto hemos hecho uso del protocolo de comunicación REST usando mensajes JSON sobre HTTP. Las principales razones son su simplicidad y el uso de mensajes JSON.

A continuación describiremos el formato de mensajería elegido y sus bondades frente a otros.

- JSON

JSON es un formato ligero para intercambio de datos. Está ligado fuertemente a los objetos Javascript. Esa simplicidad he hecho que se use se generalice en la comunicaciones web tanto para AJAX como para Web Services. Además dicha simplicidad hace que la tarea de parsearlo sea menos costosa. Es ampliamente soportado por gran variedad de lenguajes de programación, tanto aquellos que pertenecen al lado cliente, como al servidor. JSON es muy compacto lo cual hace que su fácil legibilidad sea un aspecto a favor. Pese a que se conoce ampliamente como JSON, es tan solo una parte de un protocolo mayor conocido como LJS.

XML es un lenguaje de marcas auto descriptivo. Su uso se extiende más allá que el del protocolo de comunicación, sirviendo como formato para almacenar datos así como para archivos de configuración de maneja legible. Permite definir una gramática usando DTD o XSD. Además de para Internet, se utiliza para el intercambio estructurado entre plataformas. Es más complejo que JSON, en cambio permite el uso de comentarios.

Aunque en este caso ambos formatos se enfrentan como alternativas en la comunicación mediante Servicios Web, en muchos casos son compatibles dentro de una misma aplicación pues ambos tienen sus ventajas.



En el caso concreto de nuestras aplicaciones hemos apostado por el uso de JSON por su simplicidad y por su facilidad de uso en las aplicaciones cliente. Cabe destacar que en el proyecto también se hace uso de ficheros XML como archivos de configuración, por ejemplo en la aplicación servidora o en el cliente Android.

- MVC y sus variantes MVP y MVVM

Model-View-Controller (MVC) es un patrón de arquitectura de software cuya principal función es separar en componentes la interfaz de la aplicación, de la lógica de negocio y de los datos de la misma. Además sobre su función también recae la gestión de las comunicaciones entre estos componentes así como gestionar los eventos que desencadenarán la actividad de los componentes. Nació en la década de los 80.

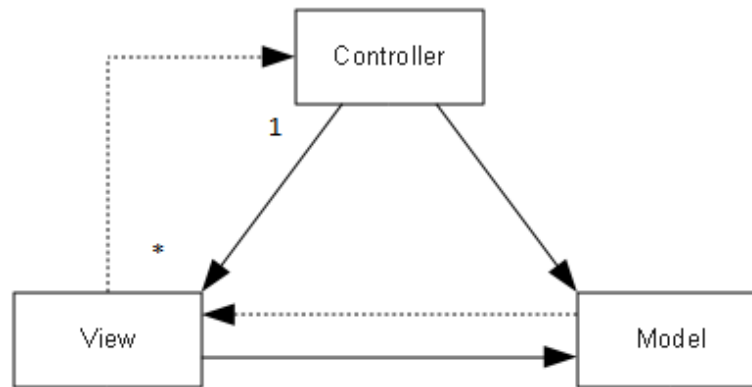
Busca separar conceptos y la reutilización de código mediante tres componentes que son:

- **Modelo:** es la representación de la información del sistema. Gestiona el acceso a esa información.
- **Controlador:** responde a los eventos del usuario e invoca peticiones al modelo. También determina la vista a mostrar (puede tener varias) y se comunica con ella si la información se tiene que presentar de una determinada forma.
- **Vista:** se encarga de la lógica de presentación la información del modelo adecuándola a la visión del usuario permitiéndole interactuar con su interfaz gráfica.

Hay cierto acoplamiento entre la Vista y el Modelo.

La interacción entre el usuario y los distintos componentes del sistema se realiza de la siguiente manera:

- El usuario interactúa con la interfaz gráfica y solicita una acción mediante una petición.
- Esta **acción corresponde a una llamada a un método del controlador**. El controlador gestiona la petición.
- El controlador accede al modelo para modificar cierta información en función de la petición.
- El **controlador delega a la vista la acción de desplegar la interfaz** de usuario reflejando los cambios del modelo. De esta forma la **Vista accede al modelo**. El controlador no pasa datos del modelo a la vista.
- La interfaz de usuario queda activa esperando nuevas acciones.



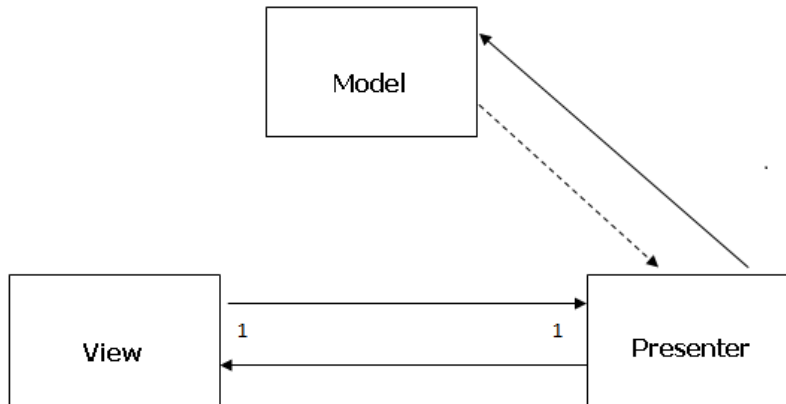
Inicialmente nació asociada a aplicaciones de escritorio, aunque con el tiempo se ha popularizado en aplicaciones web gracias a la aparición de gran variedad de frameworks en distintos lenguajes de programación usando la arquitectura cliente-servidor. Al principio la implementación de dicho patrón se desempeñaba en el lado servidor, aunque más recientemente se ha llevado también al lado cliente.

El patrón MVC se utiliza en .NET con el framework ASP.NET MVC y en Java en framework Spring.

Desde su inicio, MVC ha evolucionado sufrido algunos cambios llevando a modificaciones de la idea inicial llegando al punto de que el MVC clásico no tiene sentido en muchos casos. De esta manera han surgido variantes como MVP o MVVM.

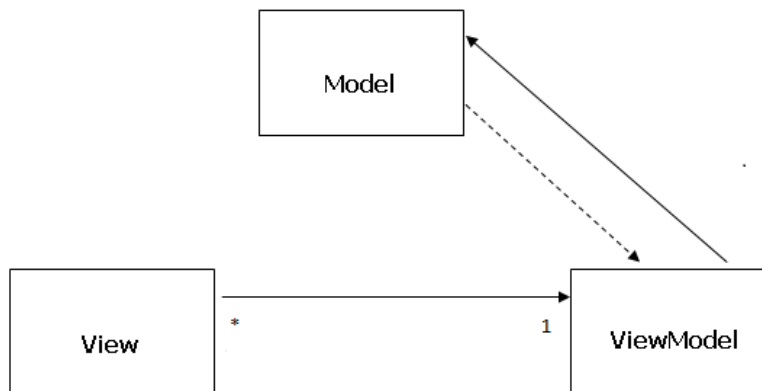
Model-View-Presenter (MVP) es una variación de MVC que apareció en los 90. En este caso el Presentador ejerce de man-in-the-middle entre el Modelo y la Vista. Ahora lo **lógica de presentación recae sobre el Presentador** accediendo al Modelo para recuperar la información. En esta variante la **Vista es mero un interfaz pasivo** que tan solo representa datos. El Modelo sigue siendo la representación de la información del sistema. Las **peticiones se realizan a la Vista** aunque esta tan solo invoca al Presentador que actúa sobre ellas, por tanto hay un Presentador por vista. La Vista se compone de la propia Vista y de una Interfaz.

Se Elimina el acoplamiento entre la Vista y el Modelo.



Muchos frameworks utilizan esta variante, por ejemplo en lenguaje .NET para los Windows Forms y ASP.NET o en Java para AWT y Swing.

Model-View-Viewmodel (MVVM) es otra variación de MVC nacida en 2005 que abstrae el estado (cambio en el modelo) y el comportamiento de la vista. En este caso el Viewmodel es el intermediario y en vez de controlar manualmente los cambios en el modelo, **los cambios se actualizan automáticamente cuando estos suceden**, este concepto se conoce como data-binding. El data-binding es en ambos sentidos, un cambio en el Viewmodel actualiza la Vista y viceversa. El encargado del sincronismo se conoce como Binder, por eso en algunos casos se conoce como Model-View-Binder.



Hay poco acoplamiento entre el Viewmodel y la Vista lo que permite un trabajo en paralelo en ambos componentes así como simplificar los tests. El Viewmodel puede ser utilizado por múltiples vistas.

En algunos casos el Viewmodel no sustituye al Controller, y hace necesario que coexistan ambos. El Controller se encarga de la lógica de negocio y de responder a eventos. En este caso se conoce como MVCVM.

Este tipo de variante es más común en implementaciones del patrón en el lado cliente, como puede ser el caso de Knockout.js o AngularJS.

Herramientas comunes

- Cmder



CMDER es un emulador de la consola de Windows y es una alternativa al CMD que viene por defecto instalada en Windows. Se ha apostado por el uso de esta consola frente al tradicional CMD porque alguna de las tecnologías y herramientas que en este proyecto se usan eran incompatibles con la consola por defecto creando problemas en la ejecución normal de los procesos. Es mucho más intuitiva y fácil que manejar que CMD, además utiliza tanto comandos de Windows como de Linux.

- Git

GIT es un software de control de versiones (VCS). Es muy útil para mantener una aplicación durante su desarrollo, teniendo un rastreo constante de las variaciones que sufren sus ficheros fuente, así como permitiendo el desarrollo conjunto de varias personas.



Permite un desarrollo no lineal mediante la gestión de ramas. Tiene gestión distribuida local para cada uno de los desarrolladores pudiendo interactuar entre ellos. Otra forma de gestión es la centralizada donde existe un único repositorio al que los desarrolladores se conectan.

Hace uso de un esquema colaborativo donde cualquiera puede modificar un archivo al mismo tiempo. En frente está el esquema exclusivo en el que solo un desarrollador puede hacer modificaciones.

Permite el intercambio de información mediante HTTP y SSH. Todas las versiones previas de un archivo se mantienen para facilitar su consulta y analizar sus variaciones.

Existen gran variedad de alternativas que siguen tanto el modelo distribuido como el centralizado. Algunas de ellas son: Subversion, Visual Studio Team Foundation Server, Mercurial o Plastic.

Se ha optado por usar GIT pues es un SCM que usa el modelo distribuido y que es de Código Libre. Además es ampliamente utilizado en proyectos individuales.

- Git extensions y EGit

Como hemos mencionado GIT es una gran herramienta muy útil durante el desarrollo de un proyecto software. Para poder manejarla, por defecto, se hace uso de la línea de comandos o CMD. Esto es un poco engorroso, por ello se ha buscado una alternativa más visual.



GitExtensions es un cliente que hace uso de GIT. Este GUI tiene una presentación más sencilla de comprender que la propia línea de comandos. Simplifica mucho la gestión de commits, ramas y conflictos.

EGit es otro cliente que al igual que GitExtensions hace uso de GIT. Es menor visual que GitExtesions, pero su gran ventaja es que se puede integrar directamente con los entornos de desarrollo (IDEs) mediante plugins, pudiendo gestionar todo desde una única herramienta.

Existen otras aplicaciones alternativas que también sirven de cliente para GIT. Estas pueden ser SourceTree o Tortoise Git. Se ha optado por el uso de GitExtensions y EGit pues es software libre y son bastante sencillas sobre todos para un proyecto pequeño en que solo interviene una persona.

- Github

Github es una plataforma de desarrollo colaborativo o 'forge' que sirve para alojar en la nube proyectos que hacen uso de GIT. Sirve de repositorio principal para un sistema de control de versiones como GIT. El código se almacena de manera pública de tal manera que cualquier persona puede consultarlo. También se puede utilizar su versión privada de pago.



Cada proyecto se almacena de manera individual y este tiene asociado una wiki donde publicar información asociada al proyecto.

Un sistema de gestión de proyectos y errores donde se pueden consultar

gráficas de cómo evoluciona el proyecto, las tareas definidas o los bugs que necesitan ser resueltos. También funciona como una red social simple permitiendo seguir otros proyectos o realizar comentarios sobre los mismos.

Otras alternativas son SourceForge o Bitbucket. Bitbucket sí permite el uso de proyectos privados de manera gratuita y también posee una interfaz simple.

La razón principal del uso de GitHub es que goza de gran popularidad y su manejo es muy simple. En la actualidad tanto grandes empresas como desarrolladores particulares utilizan esta página para alojar los proyectos OpenSource que realizan.

- [Kdiff3](#)

Kdiff3 es un programa para visualizar diferencias entre ficheros y realizar merges. Su principal atractivo es que se puede comparar ficheros visualmente para contrastar los cambios entre ellos. Es muy útil en el desarrollo de aplicaciones para descubrir esas pequeñas diferencias entre archivos de código fuente. Hace un uso de un lenguaje de colores simple para remarcar esas diferencias.



Se integra con GitExtensions y otros clientes de CVS para la resolución de conflictos a la hora de realizar merges entre distintas ramas.

Al igual que el resto de aplicaciones existen algunas alternativas, estas pueden ser Meld, Kompare, Winmerge.

Se ha elegido Kdiff3 pues se integra con GitExtensions, es muy simple su manejo y es gratuita.

- Apache Tomcat



Es un servidor web de código libre con un contenedor de servlets que además implementa diversas especificaciones de JavaEE como JSPs, EL, etc. Tomcat no es servidor de aplicaciones, aunque comúnmente así se cree pues deja fuera

algunas otras características de JavaEE como Enterprise Java Beans o Java Message Service. Se encuentra por encima del Sistema Operativo entre la JVM y la(s) aplicación(es) actuando como middleware. Fue desarrollado por Apache Software Foundation.

Está compuesto por los siguientes componentes:

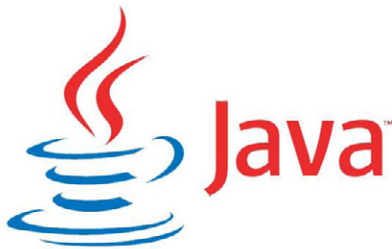
- Catalina: como contenedor de servlets,
- Jasper: como motor de JSPs,
- Coyote: como conector que soporta el protocolo HTTP
- Cluster: como balanceador de carga
- High Availability: para facilitar las actualizaciones de sistema
- Componentes adicionales construidos por usuarios terceros que incluyen varias de las características que deja fuera Tomcat haciendo que pueda ser un servidor de aplicaciones.

Otras alternativas a Tomcat son Jboss, WebGlassfish, WebLogic, WebSphere. Pese a que el resto sí son servidores de aplicaciones, en nuestro caso solo utilizamos contenido estático con lo que Tomcat nos es suficiente. Además la instalación es inmediata.

Aplicación servidora - Tecnologías

- Java

Java es un lenguaje de programación orientado a objetos de propósito general. Está diseñado para que tenga el menor número de dependencias posibles y que pueda ser ejecutado en cualquier plataforma sin necesidad de ser recompilado, para ello el código fuente es compilado en y ejecutado sobre la máquina virtual JVM. Posee un recolector de basura para solventar problemas de fugas de memoria. La sintaxis de este lenguaje viene en mayor medida de otros como C o C++.



Incluye componentes como JDBC para conexión a base de datos, RMI y CORBA para invocación remota en plataformas distribuidas, una API gráfica como Swing o AWT, JNDI para servicios de directorio, Applets para aplicaciones embebidas, JavaFX para aplicaciones RIAs ...

Es uno de los lenguajes de programación más populares, en gran medida gracias a las aplicaciones cliente-servidor. Fue desarrollado por Sun Microsystems y comprada por Oracle. Conocida como Java 2 Standard Edition (J2SE). Su última versión es la 8 lanzada en marzo de 2014.

No debe confundirse con JavaEE. JavaEE es una extensión de Java para entornos empresariales con aplicaciones que corren sobre un servidor. Entre los componentes que incluye se encuentran: Servlets como mecanismo para generar respuestas en aplicaciones web, JSPs como respuesta en peticiones HTTP, JPA para acceso a base de datos vía ORM, JSFs para construir interfaces **gráficas, Enterprise Java Beans, Portlets, WebSockets, DI, EL, ...**

Existen multitud de lenguajes de programación de diferente tipología: alto y bajo nivel, compilados o interpretados, declarativos o imperativos. También pueden ser orientados a objetos, de scripting, procedurales, esotéricos, y muchas otras clasificaciones.

Algunos de los lenguajes de propósito general más usado actualmente que pueden ser alternativa a Java son: C, C++, C#, Python, Ruby, Javascript,

Objetive-C, PHP. Se ha elegido Java por ser compilado y orientado a objetos, así como el lenguaje más utilizado actualmente.

- EclipseLink JPA


JPA es una framework que incluye un motor de persistencia que maneja datos relacionales en aplicaciones bajo plataforma Java. Busca no perder las ventajas de la orientación a objetos al interactuar con una base de datos.

 Utiliza el patrón Object-Relational Mapping (ORM) usando objetos simples o POJOs. Junto a JPA se puede hacer uso de JPQL como lenguaje para hacer consultas contra entidades almacenadas en la base de datos.

Existen varias implementaciones de JPA, cada cual amplía sus prestaciones, alguna de ellas son: EclipseLink, Hibernate, TopLink, ObjectDB, etc. La implementación de referencia es Eclipselink la cual es utilizada en este proyecto por el mismo motivo.

- Spring

Spring es un framework para el desarrollo de aplicaciones en Java y funciona como contenedor de beans usando inversión de control o IoC. Se puede utilizar sobre cualquier aplicación Java y, usando algunas extensiones, sobre aplicaciones JavaEE.

 Ha reemplazado el uso de los Enterprise Java Beans. Para ello hace uso el contenedor Inversión de Control el cual proporciona una manera consistente la administración a través de la Reflexión. Es el encargado de gestionar el ciclo de vida (creación, inicialización, configuración, etc.) de los objetos o beans. Para administrar estos objetos hace uso del Inyección de Dependencias (DI) mediante constructores, propiedades o builders.

Ha evolucionado continuamente en el tiempo, llegando a la versión estable actual 4.1.6. Está compuesta por varios módulos:

- Core: es el módulo base. Tiene los contenedores BeanFactory y ApplicationContext.
- Contenedor Inverse of Control: para la administración del ciclo de vida de los beans mediante la inyección de dependencias.
- Modelo Vista Controlador: hace uso servlets y servicios web basados en HTTP.
- Autenticación y autorización: provee características de seguridad usando el sub proyecto Spring Security.
- Programación Orientada a Aspectos: hace uso del paradigma AOP.
- Y otros muchos como Acceso a datos, Gestión de transacciones, Acceso remoto, Mensajes, Testing, etc.

Como alternativa a Spring, puede haber varios frameworks que sean similares a alguno de los módulos de Spring, como por ejemplo: Struts (MVC), JSFs (MVC), JAAS (Security). El único framework que puede hacer frente a Spring es JavaEE que ya describimos anteriormente. Se ha optado por Spring por su gran popularidad y por su uso creciente en grande proyectos.

- MySQL

MySQL es un sistema de gestión de base de datos relacionales (RDBMS) haciendo uso del lenguaje SQL junto algunas directivas propias. Es uno de RDBMS más populares de software libre y actualmente Oracle es su propietaria. Es ampliamente utilizada en aplicaciones web, forma parte de la pila LAMP para el desarrollo de aplicaciones, está incluida en Gestores de Contenidos como Joomla, Drupal o WordPress, etc.



Algunas características de MySQL son su gran variedad de motores de almacenamiento, agrupación de transacciones, es multi hilo, es multiplataforma, tiene conectividad con aplicaciones escritas en gran variedad de lenguajes de programación, posibilidad de procedimientos almacenados, vistas, triggers, cursores, soporta SSL, cacheo de queries, etc.

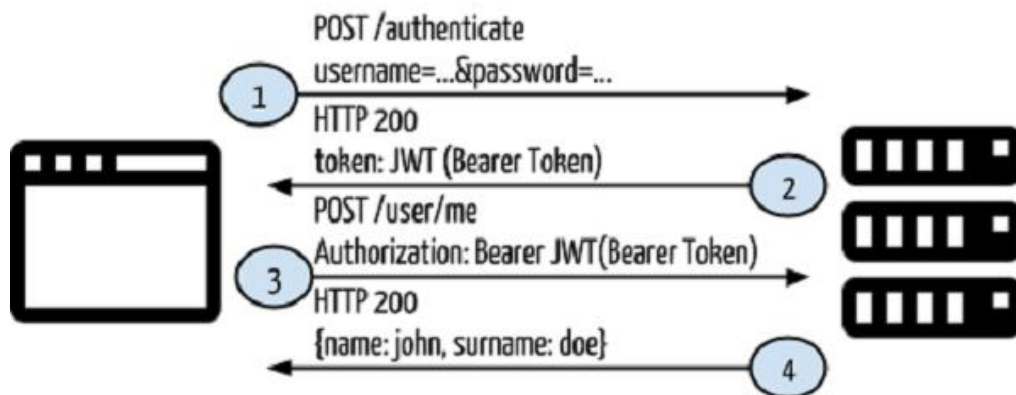
No tiene interfaz gráfica de usuario (GUI), para hacer uso de esta tecnología hay que hacerlo mediante la línea de comandos, aunque existen algunos clientes desarrollados por terceros que ayudan a su manejo de manera gráfica. Más adelante los comentaremos más detenidamente.

Existen otros RDBMS en el mercado como pueden ser Oracle DB, Microsoft SQL Server, IBM DB2, Informix, PostgreSQL. Se ha elegido MySQL por su simplicidad, su popularidad y su fácil instalación y manejo.

- JWT

JSON Web Token es un medio compacto, estandarizado y seguro para la transferencia de peticiones entre dos partes. Usa JSON como formato del token y es utilizado principalmente en la autenticación de un usuario en una aplicación web. No está basado ni en cookies ni en sesiones, sino en tokens que se envían en cada petición. Existen librerías con generadores de tokens para la gran mayoría de lenguajes de programación.

El flujo de control en una aplicación web usando JWT sería el siguiente: El usuario se loguea usando su nombre de usuario y su contraseña y se envía la petición. Se valida el usuario en el servidor, si es válido se crea un token utilizando la información brindada por el servidor y se le envía de vuelta al cliente en la cabecera de la respuesta. El cliente lo almacena en local (no sesiones ni cookies). En cada nueva petición que haga el cliente se enviará además el token en la cabecera, que será validado por el servidor y así podrá acceder a las partes restringidas de la aplicación. El servidor envía de vuelta la información solicitada.



El token se forma de tres partes:

- Cabecera: almacena el tipo de token y el método de encriptación. Va encriptado en base 64.
- Carga útil: contiene la información del usuario y está encriptada en base 64.
- Firma: consiste en combinaciones del encabezado, la carga útil y la clave secreta. Esta clave se almacena en el servidor.

Para hacer seguro este método de autenticación es conveniente securizar también la transferencia de peticiones mediante SSL

JWT es más conveniente para la autenticación de usuarios usando Servicios Web pues en cada petición se envía el token y en cada petición se valida que esta sea correcta. No tiene cookies ni tiene estados por lo que no es vulnerable a ataques CSRF Tampoco es cross-domain. Permite el desacoplamiento entre el cliente y el servidor pues la comunicación sigue el formato JSON pudiéndose hacer peticiones desde cualquier tipo de cliente con la única restricción que use peticiones HTTP.

Aplicación Servidora - Herramientas

- Eclipse

Eclipse es un software que se compone de un conjunto de herramientas que sirven como espacio de trabajo para el desarrollo de código fuente para aplicaciones de cliente enriquecido. Es multiplataforma. Es utilizada como Entorno de Desarrollo Integrado (IDE) para gran variedad de lenguajes, donde Java es el predominante. Otros de los lenguajes que soporta es C, C++, Python, Ruby, PHP, Ada, COBOL, etc.



Posee una gran comunidad de usuarios que amplían continuamente las herramientas que este IDE hace uso. Es desarrollado por la Fundación Eclipse para fomentar el código abierto aunque inicialmente era IBM quien se encargaba de su desarrollo. Existen diferentes soluciones en función de las necesidades del proyecto que incluyen un subconjunto de herramientas concreto aptas para las tecnologías de ese proyecto. La última versión disponible es Eclipse Luna, aunque próximamente (Junio 2015) se liberará la siguiente Eclipse Mars.

Entre las características que este software tiene, destacan:

- La disposición de un editor de texto y analizador sintáctico para los múltiples lenguajes que soporta.
- Un compilador en tiempo real.
- Soporte para pruebas unitarias.
- Integración con varios CVS.
- Soporte de tareas Ant.
- Asistentes para la creación de proyectos.
- Capacidad de refactorización de código.
- Soporta desarrollo para servidores

Además existen gran cantidad de plugins desarrollados, tanto por la Fundación Eclipse como por terceros que amplían la oferta de herramientas que se pueden integrar con Eclipse. Muchos de estos plugins al no venir instalados por defecto hacen más ligero el programa. Algunos de estos programas pueden ser: editores de texto WYSIWYG, frameworks para el diseño de **interfaces gráficas, editores de UML,...**

Existe gran variedad de alternativas, entre ellas destacan otros IDEs como: NetBeans, IntelliJ IDEA, Microsoft Visual Studio o BlueJ. También compiten en este apartado editores de texto potentes como pueden ser: SublimeText, NotePad++ o VIM. La elección de Eclipse se basa en que es muy potente para el lenguaje de programación elegido, Java, es de código libre, y es ampliamente configurable y conocido. Para la realización de este proyecto también se ha utilizado SublimeText como editor para realizar la aplicación cliente. Más adelante lo describiremos en detalle.

- Maven

Maven es una herramienta software utilizada para la creación y gestión de proyectos Java. Es similar a Ant aunque con un modelo de configuración más simple basado en ficheros con formato XML. Para describir el proyecto Maven utiliza el Modelo de Objetos de Proyecto (POM), donde describe las dependencias de los otros módulos que hace uso un proyecto. Además trae un conjunto de herramientas para la compilación y empaquetado del proyecto.

The logo for Maven, featuring the word "maven" in a bold, lowercase, sans-serif font. The letter 'a' is highlighted in orange, while the rest of the letters are black.

Su gran ventaja es que está diseñado para funcionar en red. De esta manera Maven puede descargar dinámicamente las dependencias de un repositorio central mantenido por Maven. También

puede actuar en la otra dirección para subir artefactos al repositorio para que cualquier usuario pueda acceder.

Una característica importante es que busca definir una convención sobre la configuración de proyectos, para ello define una serie de Arquetipos que cualquier proyecto podría seguir y así estandarizar los modelos en cierta medida. El ciclo de vida de un proyecto Maven es: compilar, tests, empaquetado, instalación y despliegue. Actualmente existen dos versiones de Maven, la segunda ha sido diseñada para poder soportar proyectos de mayor tamaño.

Tiene integración con diferentes IDEs como puede ser Eclipse, Netbeans o IntelliJ IDEA.

Algunas alternativas a Maven son: Ant y Gradle. Ant no incluye un gestor de dependencias, necesita hacer uso de Ivy. Se ha usado Maven por su

integración con Eclipse y su simpleza tanto para crear proyectos como para gestionar dependencias en proyectos pequeños. Gradle lo mencionaremos más adelante como parte para configurar la aplicación móvil.

- Advanced REST Client

Es un plugin para el navegador Google Chrome para realizar peticiones REST contra una API definida mediante Web Services a través de una conexión HTTP. Es una herramienta muy útil durante la etapa de desarrollo de una aplicación servidora basada en servicios web pues se pueden realizar pruebas sobre dichos servicios.



Da la posibilidad de crear peticiones a una API desde el propio navegador de manera sencilla. Soporta gran variedad de métodos HTTP como GET, POST, PUT, DELETE, etc así como la definición del cuerpo de la petición. También incluye la definición de cabeceras. Incluye la visualización de la respuesta del servidor en múltiples formatos: plain text, JSON, XML, HTML, etc.

Existen otros plugins similares para otros navegadores, pero se ha optado por este porque Google Chrome es el navegador elegido para dar soporte a la aplicación web del proyecto.

- HeidiSQL



HeidiSQL es un software cliente de código abierto para el sistema de gestión de base de datos MySQL. También sirve para todos forks de MySQL, como son: PostgreSQL, Microsoft SQL Server, MariaDB, etc. Para poder conectarse y gestionar bases de datos el usuario debe loguearse y crear una sesión contra un servidor MySQL ya sea en local o en remoto. Tiene una interfaz gráfica simple, más amigable que usar la línea de comandos de windows, el método definido por defecto por MySQL. Es muy útil para desarrolladores durante la creación de un proyecto, demasiado simple para un administrador de base de datos.

Entre las características que destacan se encuentran:

- Conexión con múltiples sesiones usando TCP/IP, sockets y SSH.
- Gestión de usuarios y privilegios para el acceso a bases de datos.
- Exportación de datos a archivos SQL.
- Gestión de las variables del servidor.
- Gestión de bases de datos.
- Gestión de tablas, vistas, procedures, triggers y eventos.

Existen otras alternativas a HeidiSQL como clientes de MySQL: pueden ser MySQL Workbench, PHPMyAdmin, Navicat, etc. Se ha elegido esta por ser de código abierto, por su fácil instalación y por su simpleza.

Aplicación Web - Tecnologías

- HTML, CSS Javascript y AJAX



HTML es un lenguaje de marcado para crear páginas web. Define una estructura básica para la definición del contenido de una página web mediante etiquetas y es el navegador el encargado de interpretar el código. Hace uso de la referenciación para incluir elementos externos como imágenes, videos, scripts, etc.

Es el estándar de referencia, aunque para hacerlo más eficiente sufre variaciones en sus diferentes versiones lo que hace que los navegadores tengan que actualizarse para mostrar correctamente las páginas. Eso muchas veces es un problema pues distintos navegadores utilizan diferentes variaciones de HTML y no siempre las páginas se visualizan correctamente



CSS u hojas de estilo es un lenguaje para definir la presentación de un documento HTML. Surge con la necesidad de desacoplar la estructura de la presentación, por lo tanto se debe definir en un archivo aparte que será referenciado en el documento HTML. Tiene una sintaxis sencilla que hace uso de propiedades de estilo para definir las reglas que los elementos HTML deben seguir, estas reglas se asignan a los elementos

mediante selectores.

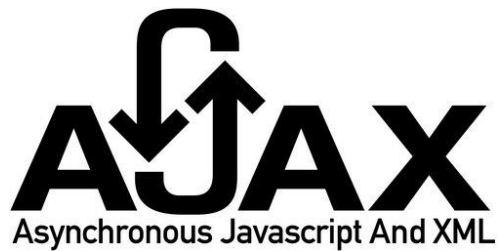
Al igual que HTML es un estándar de referencia. El encargado de definir sus especificaciones es el consorcio W3C. También tiene el problema de compatibilidad con la amplia gama de navegadores.



Javascript es un lenguaje de programación interpretado utilizado mayoritariamente en el lado cliente para el desarrollo de páginas web. Es interpretado por el navegador y permite mejorar la interfaz de usuario haciendo las páginas más dinámicas. Usa una sintaxis similar a C o Java aunque su similitud termina aquí pues son para propósitos diferentes. Su funcionalidad principal es el manejo de los elementos del modelo de objetos del documento (DOM), del navegador así

como interactuar con el CSS. También responde a los eventos generados al interactuar con una página web.

Como pasa con HTML y CSS es el W3C el encargado de definir la especificación y al igual tiene dificultades de que dicha estandarización esté implementada por igual en todos los navegadores.



AJAX no es una tecnología propiamente dicha, es una técnica de desarrollo web para que las aplicaciones cliente se comuniquen asíncronamente con el servidor en segundo plano. De esta manera se pueden recuperar datos en la propia página web sin necesidad de recarga.

HTML, CSS, Javascript y AJAX en conjunto son el medio estándar para definir páginas web modernas con contenido dinámico y visualmente atractivo. No tienen ninguna alternativa directa pues son los únicos lenguajes que los navegadores son capaces de interpretar. Es cierto que con el paso del tiempo se ha hecho insuficiente el uso de estas tecnologías por sí solas, y actualmente existen gran cantidad de frameworks que ayudan al desarrollo de aplicaciones web.

A continuación veremos algunos frameworks que amplían las capacidades de estas tecnologías: Bootstrap como framework visual que hace uso de CSS haciéndolo más potente. jQuery que extiende javascript para modificar el DOM de manera más simple y eficiente. Y AngularJS que usa etiquetas HTML propias y AJAX encapsulado en servicios para recuperar datos del servidor y mostrarlos de manera ordenada siguiendo el patrón MVC.

Como alternativa parcial a HTML, CSS, JS y AJAX están las tecnologías de JSP, PHP y ASP que hacen uso de los lenguajes de programación tradicionales del lado servidor. Estas tecnologías, que se apoyan en las tecnologías cliente aquí vistas, se utilizan para construir vistas con contenido dinámico aunque llevan su carga de procesamiento a la aplicación servidora. El problema principal es que el código de presentación y el de negocio están muy acoplados. Es por ello por lo que se ha apostado por el conjunto de tecnologías cliente y varios frameworks para el desarrollo de la aplicación web.

- AngularJS

AngularJS es un framework del lado cliente de código abierto creado en 2009 y desarrollado por Google. Es utilizado para la creación de aplicaciones de navegador de una sola página (SPA) que sigan el patrón Modelo Vista Controlador (MVC) de manera sencilla pues hace una separación de responsabilidades entre CSS, HTML y Javascript. También simplifica en gran medida la creación de test, para ello se ayuda de otras tecnologías como Jasmine y Karma.



AngularJS define un conjunto de etiquetas personalizadas que son utilizadas por atributos HTML para unir elementos de entrada o salida a variables de un modelo. Esa

unión se conoce como doble data-binding y sirve para mostrar contenido dinámico de manera sincronizada, esto es que cada cambio en el modelo se actualiza automáticamente en la vista, y viceversa. El responsable de unir ambas partes es el scope o viewmodel. De esta manera desasocia la manipulación del DOM de la aplicación.

También se desasocia la aplicación cliente de la aplicación servidora, permitiendo trabajo en paralelo y eliminando cualquier tipo de acoplamiento. Esto es posible a que la información del modelo se recupera principalmente de servicios web ofrecidos por un servidor web en formato JSON. También hace uso de la inyección de dependencias para inyectar diferentes componentes cuando los necesitemos. Así también consigue reusabilidad de componentes.

Usa la programación declarativa para generar interfaces de usuario de la vista y deja la programación imperativa para definir la lógica de negocio del controlador. Hace uso de arquitecturas propias de un servidor como es el patrón MVC liberando así la carga del servidor y haciendo aplicaciones más ligeras.

Algunas de las alternativas a son Ember.js y Backbone.js Backbone fue el primero. Es el más fácil de aprender pues tenemos más control sobre el código aunque esto hace que pierda potencia y que sea más difícil estructurar bien el código lo que conlleva que a medida que crezca sea más probable que se haga

inmanejable. Angular pese a ser más reciente posee ya una amplia comunidad desarrolladores. Es un framework complejo que cambia la forma de desarrollar habitual, pero que una vez que se domina lo hace muy potente haciendo que el desarrollo sea rápido. Ember también es complejo y muy potente, su principal problema es que sufre muchas actualizaciones que cambian bastante el framework y hace que su comunidad no sea tan extensa.

- jQuery



jQuery es un framework Javascript de código abierto que simplifica la manera de interactuar con los elementos del DOM de una página HTML, elementos del navegador, manejar eventos, modificar los estilos y agregar interacciones con el servidor mediante AJAX. Todas sus funcionalidades están descritas en un único fichero. Goza de gran popularidad y actualmente es la librería javascript más utilizada. Su versión más reciente es la 1.11.

Ofrece un conjunto de funcionalidades basadas en Javascript pero reduciendo en gran medida la cantidad de código escrito así como los tiempos de ejecución y el espacio ocupado.

Sus características principales son:

- Selección de elementos del DOM.
- Modificación del árbol del DOM.
- Gestión de eventos.
- Manipulación de las hojas de estilo.
- Creación de efectos y animación.
- Soporte a AJAX.
- Interacción con elementos del navegador (BOM) .
- Compatibilidad con una amplia gama de navegadores y varias de sus versiones.
- Compatibilidad con otros frameworks Javascript.

Existen gran cantidad de frameworks adicionales basados directamente en jQuery haciendo uso de muchas de sus funcionalidades, alguno de ellos son jQueryUI, KendoUI o Bootstrap. El último de ellos es utilizado también en este proyecto y será descrito más adelante.

Algunos de los frameworks alternativos a jQuery son Prototype, MooTools, Ext o Script.aculo.us (Prototype based). Soportan la gran mayoría de las características que jQuery ofrece, pero en algunos casos, con un rendimiento peor, y con la necesidad de escribir más líneas de código. El uso de jQuery está mucho más extendido que el de los demás.

- Bootstrap

Twitter Bootstrap, es un framework de software libre creado para ayudar en el desarrollo del diseño de una aplicación web. Contiene una amplia gama de plantillas para el diseño de múltiples elementos, como son tipografías, formularios, campos y botones, menús de navegación, secciones, y otros elementos HTML y CSS. También usa extensiones Javascript. Es el proyecto más popular en la plataforma GitHub para publicar proyectos colaborativos. Cualquier desarrollador puede aportar mejoras a este framework público.



Como se indica en su nombre fue creado por dos desarrolladores de Twitter, para fomentar la consistencia de los múltiples componentes de su aplicación. Es

compatible con la mayoría de los navegadores (Chrome, Firefox, Opera, Safari y Internet Explorer), así con la amplia gama de dispositivos (ordenadores, smartphones y tablets) gracias al diseño responsive dando la posibilidad a ajustarse dinámicamente a las propiedades del dispositivo.

Bootstrap está creado en módulos que implementan la gran variedad de componentes que lo forman. Para construir los módulos hace uso de LESS, un lenguaje dinámico que puede ser compilado para la creación de hojas de estilo CSS. El framework proporciona un conjunto de hojas de estilo que proveen las definiciones básicas para todos los componentes, de esta manera se consigue uniformidad en el diseño.

Para el ajuste de los elementos se dispone de cuadrícula que permiten un diseño responsive, aunque también permite una configuración tradicional usando el ancho-variable. Ambas técnicas permiten el ajuste automático.

Bootstrap puede ser adaptado por los desarrolladores si se necesita personalizar opciones. También dispone la posibilidad de crear plugins para extender funcionalidades adicionales haciendo uso de jQuery

Existen otras alternativas a Bootstrap como pueden ser Foundation, KickStart, Kickstrap o Skeleton aunque en algunos casos son más simples y en otros más complicados de aprender y manejar.

Aplicación Web - Herramientas

- Google Chrome

Google Chrome es un navegador web creado por Google, fue lanzado en septiembre de 2008 y puede utilizarse en la gran mayoría de los sistemas operativos actuales. Es el navegador más utilizado actualmente pese a ser el último en ser lanzado de los grandes navegadores actuales. Es un software privativo, aunque tiene una versión paralela en software libre llamada Chromium, que sirve de base para Chrome.

Utiliza desde 2013 el motor de renderizado Blink basado en Webkit. Sustituyó a este último por razones de optimización de código fuente y mejora del rendimiento. Otras de las características que cuenta este navegador son:

- Interfaz simple y amigable basado en pestañas.
- Multiproceso para aislamiento de pestañas y asegurar la seguridad y estabilidad.
- Motor independiente para la máquina virtual de Javascript para un alto rendimiento.
- Sustitución de página de inicio por página de sitios más visitados.
- Sugerencias automáticas de búsqueda en Google.
- Integración con otros servicios de Google como traducción y geolocalización.
- Permite la instalación de extensiones, plugins y temas a través de su Web Store.
- Modo incógnito para navegar en total privacidad.
- Uso de listas negras para control de sitios maliciosos.
- Instalación automática de versiones.



Entre sus alternativas en el mercado se encuentran gran variedad navegadores como: Mozilla Firefox, Internet Explorer, Opera o Safari. Hay una lucha reñida entre ellos por posicionarse en el mercado, dando lugar a lo que se conoce como la segunda guerra de los navegadores, actualmente todos con una cuota de mercado considerable donde gana Google Chrome seguido de cerca por Mozilla Firefox.

Pese a que se ha intentado hacer una aplicación multi navegador, se ha optado por Google Chrome por navegador por defecto porque ofrece una característica muy útil durante la etapa de desarrollo del proyecto esto es, deshabilitar la Política de Mismo Origen (SOP). Más adelante explicaremos qué es este concepto, aunque la política no sería deshabilitar SOP sino implementar el mecanismo CORS.

- Sublime Text

Ya ha sido mencionado anteriormente en este documento como alternativa a Eclipse. Sublime Text es un editor de texto y código fuente desarrollado inicialmente como una extensión de Vim aunque actualmente ha evolucionado de forma propia.. Es software propietario aunque su distribución es gratuita.

Soporta gran variedad de lenguajes de programación y de marcado como Java, C++, C#, PHP, Javascript, SQL, XML. Entre otras de las opciones por las que se caracteriza destacan:

- Mini mapa para pre visualización de código.
- Multi selección de términos y búsqueda dinámica.
- Multi cursor para escribir a la vez en varias posiciones.
- Marcado de sintaxis configurable para los lenguajes soportados.
- Función de autocompletado de términos.
- Soporte de snippets y plugins lo que permite una gran extensibilidad.
- Realización de tareas mediante comandos basados en Python.
- Otras características como acceso rápido a archivos, resaltado de **sintaxis, uso de pestañas, configuración de atajos, ...**



Actualmente la versión estable es la Sublime Text 2 aunque se espera que la tercera sea lanzada próximamente. Cabe resaltar una de sus grandes características: el uso de plugins de terceros gestionados por el Package manager, esto permite hacer mucho más completo a este editor haciéndolo customizable para cualquier desarrollador.

Algunas de sus alternativas son: Brackets, Atom, Notepad ++, Ultraedit, Vim. Como hemos mencionado antes, también compite de forma indirecta en esta

categoría los IDEs. Se ha optado por Sublime Text 2 para el desarrollo de la aplicación web por su ligereza, por su extensibilidad, y adecuación a lenguajes de programación del lado cliente.

- Node.js y NPM

Node.js es un entorno de programación principalmente del lado servidor, que también puede ser de lado cliente, tiene una arquitectura orientada a eventos y está basado en el motor javascript V8 de Google. Nacido en 2009 es open-source. Es muy útil para la creación de programas de red o servidores web.

Usa el lenguaje de programación ECMAScript, una estandarización de los lenguajes de scripting como Javascript o Jscript, para definir programas que se ejecutan en el lado servidor. Puede ser utilizado sobre gran variedad de servidores basados en Windows, Mac OS X o Unix.



Node.js se compone de módulos/nodos que ofrecen funcionalidades como conexión de red, gestión entrada y salida y ficheros o programación asíncronas. Alguno de estos nodos básicos son Path, FileSystem, Buffer, Timers o Stream. Además ofrece la posibilidad de utilizar nodos creados por terceros que vienen pre compilados e implementan utilidades para utilizar en aplicaciones web. Para ello incluirlos se hace uso de Node Package Manager (NPM), un programa que facilita la compilación, instalación y actualización de los mismos. Algunos nodos terceros con cierto reconocimiento son Connect o Express.

Node.js trae consigo un nuevo stack de programación llamado MEAN y basado en el lenguaje de programación Javascript. MEAN son las iniciales de MongoDB una base de datos NoSQL, Express, un framework de lado servidor para hacer aplicaciones web, AngularJS framework web de lado cliente y el propio Node.js.

Como alternativa a programas para creación de servidores web están PHP y entornos para Python, Rudy y otros múltiples lenguajes. Alternativas también basadas en Javascript están IO.js o JXcore. En nuestra aplicación es utilizado para montar el servidor web sencillo sobre el que correrá la aplicación web basada en AngularJS.

- Yeoman, Bower y Grunt

Yeoman es una pila de desarrollo del lado cliente compuesto por herramientas y frameworks para ayudar a la rápida construcción y desarrollo de un aplicación web. Es un proyecto de código abierto cuyo código se encuentra publicado en Github. Su principal uso es realizar el scaffolding de aplicaciones web basadas en AngularJS, aunque no ata al uso de esta tecnología, pero ha sido la que ha popularizado a la herramienta. Está escrito en Node.js y utiliza la línea de comando para que el desarrollador pueda realizar todas las tareas necesarias., tales como crear el scaffolding, gestionar dependencias, ejecutar los tests unitarios o arrancar un servidor de desarrollo.



YEOMAN

Hace uso de Generators para crear la estructura básica del proyecto usando librerías externas como HTML5 Boilerplate, jQuery, Normalize.css. Durante la construcción del proyecto también se pueden añadir otras librerías adicionales como Bootstrap o Require.js. Como ya se ha mencionado, también se pueden utilizar Generators más avanzados para el desarrollo de aplicaciones de lado cliente basadas en la arquitectura MVC como AngularJS o Backbone. Cualquiera puede diseñar Generators para hacer plantillas para cualquier tipo de proyecto.

Aparte de ser usado en la construcción del proyecto, también aparece durante la etapa de desarrollo pues ayuda a la creación de nuevas funcionalidades mediante generación de código mediante otros Generators, así como ejecutar tests unitarios, comprobar problemas de código, o proveer un servidor de desarrollo. También participa en la etapa final del proyecto pues proporciona herramientas para minimizar código u optimización del mismo previa al despliegue.

Se ha dicho que es una pila de desarrollo que incluye herramientas para la construcción rápida de aplicaciones web. En el desarrollo de este proyecto se han usado algunas, estas son:

- Grunt: es un gestor de tareas para la automatización de las mismas en procesos como el de minimización, compilación, ejecución de tests o arrancar el servidor. Ahorra mucho tiempo en la realización de tareas repetitivas, pues únicamente hay que configurar el gruntfile. Tiene gran cantidad de plugins que son publicados y que cualquiera puede usar y que hacen uso de tecnologías como Less, Sass, Require.js, Stylus o JSHint.



- Bower: es un gestor de paquete para aplicaciones web. Es similar a NPM usado en Node.js solo que más orientado a aplicaciones cliente. Es muy útil para buscar, instalar y gestionar las dependencias en un proyecto hecho con Yeoman pues se hace de manera automática y el desarrollador no tiene que manipular nada. Tiene un archivo de configuración llamado bower.json.



Existen otras herramientas que ayudan a Yeoman a construir aplicaciones web, como Gulp una alternativa a Grunt para la automatización de tareas, Karma un entorno para configurar tests unitarios, o Jarmine como framework Javascript para definir esos tests.

Como alternativa a Yeoman existen Lineman.js o Slush. Yeoman es mucho más popular y se integra con múltiples herramientas adicionales como las que hemos descrito. Se ha utilizado en este proyecto por su facilidad de utilización.

Aplicación móvil - Tecnologías

- Android



Android es un sistema operativo (OS) basado en Linux diseñado para dispositivos móviles como smartphones y tablets aunque más recientemente su uso se ha extendido a smartwatches, televisiones o automóviles. Es desarrollado por Google y apoyado por la Open Handset Alliance, un consorcio de compañías para desarrollar estándares abiertos para dispositivos móviles. Android fue presentado en 2007 y lanzado en 2008.

Entre las características que destacan a este OS están:

- Adaptabilidad de la pantalla a múltiples tamaños y resoluciones.
- Uso de una base de datos liviana como SQLite para almacenamiento local.
- Navegador web similar al usado en OS de escritorio.
- Basado en Java aunque con una máquina virtual propia, Dalvik.
- Soporte para multimedia y streaming.
- Es multitarea
- Soporte para hardware adicional (cámaras, pantallas táctiles, GPS, ...).
- Entornos de desarrollo específicos como Eclipse o Android Studio.

Las aplicaciones Android usan el Android Development Kit (SDK) y Java para su desarrollo haciendo uso de la API de Android. EL SDK, incluye un conjunto de herramientas de desarrollo, que incluyen un debuggeador, librerías, emuladores, etc.

Como entorno de desarrollo se puede utilizar una versión de Eclipse ideada para Android o Android Studio desarrollado por IntelliJ IDEA. También se pueden utilizar herramientas como Apache Cordova para migrar aplicaciones desarrolladas con tecnologías web a formato móvil. Tanto si se desarrolla de manera nativa como si se utiliza Cordova, se obtiene una aplicación instalable en formato APK (Android Package). Todas las aplicaciones creadas se publican en un Market llamado Google Play, aunque también puede ser instaladas directamente desde la web de terceros.

Durante el desarrollo de las aplicaciones estas pueden probarse directamente sobre terminales móviles, muy útil si hacen uso del hardware propio del dispositivo, o mediante emuladores, algo más lento y engorrosos, pero que si necesitan hacer uso de dicho hardware, puede simularlo.

Existe una gran comunidad de desarrolladores que crean aplicaciones basadas en Android, lo que permite que el OS sea muy potente. Están apareciendo recientemente varios frameworks y librerías que facilitan el desarrollo. Además Android periódicamente va mejorando el sistema para facilitar a los desarrolladores su tarea.

Como alternativas a Android, en cuanto a sistema operativo móvil existen dos grandes competidores: IOS de Apple y Windows Phone de Microsoft. Los sistemas operativos van fuertemente ligados al Hardware del dispositivo por tanto los terminales soportan un único OS. Se ha optado por el desarrollo de la aplicación móvil usando Android por su gran comunidad de desarrolladores, así como el gran número de dispositivos y usuarios que lo utilizan.

Para el desarrollo de este proyecto se ha utilizado Eclipse for Android developers, una versión de Eclipse que ofrece la SDK así como un conjunto de herramientas de desarrollo Android (ADT) y emuladores del OS.

Metodología

La forma de trabajo seguida durante la creación de este proyecto responde a una metodología pesada, donde se ha hecho hincapié en un proceso basado en casos de uso, con desarrollo iterativo e incremental y centrado en una arquitectura. Esto es que tras definir un conjunto de casos de uso, en cada iteración se ha optado por realizar un pequeño subconjunto de ellos.

Cabe destacar que como una parte importante en el proceso ha sido el estudio y análisis de las técnicas y tecnologías concretas que se querían utilizar y ha tenido gran peso durante la etapa de Análisis y Diseño de la arquitectura a seguir.

Para la metodología a seguir se ha optado por basarse en el desarrollo software que Rational Unified Process (RUP) propone así como seguir las disciplinas que describe. Por el tipo de proyecto y su tamaño la metodología no ha sido seguida al pie de la letra y algunas de las etapas/disciplinas se han unificado como puede ser Modelado y Requisitos o Análisis y Diseño. Otras etapas han tenido un peso casi nulo, como puede ser la de Gestión al tratarse de un proyecto pequeño y de un único miembro.

También se ha reducido en el alcance del proyecto la realización de pruebas pues se quería prestar mayor atención a la arquitectura y al desarrollo usando tecnologías concretas. Por la corta duración del proyecto, la realización de pruebas unitarias y de integración apenas ha tenido importancia más allá de probar que el conjunto de funcionalidades unidas respondían correctamente.

A continuación se va a describir las disciplinas por las que se ha pasado para posteriormente describir en detalle las actividades realizadas en cada:

- Modelado y Requisitos
- Análisis y Diseño
- Implementación
- Pruebas

Como disciplinas de soporte se ha optado por agrupar todas en una sola

- Entorno y Configuración

Modelado y Requisitos

Las necesidades que se plantea resolver el desarrollo de este proyecto son:

Poder disponer de una herramienta para la gestión de notas y recordatorios que sean de acceso exclusivo el autor de las mismas. Se necesita que dichos elementos sean agrupados en función de las necesidades del autor. Se desea que dicha herramienta permita la consulta de las mismas desde cualquier lugar sin importar su ubicación.

Una nota es una anotación realizada por un autor con un título y una descripción.

Un recordatorio es una anotación realizada por un autor con un título y una fecha de completitud.

En la gestión de las notas y recordatorios se desea poder crearlos, editarlos, consultarlos y borrarlos.

Para poder agrupar las notas y recordatorios se desea la posibilidad de crear Categorías donde podamos incluir y excluir estos elementos.

Para el acceso privado y exclusivo del autor (usuario de la herramienta) se desea disponer de un sistema de autenticación.

De esta manera podemos sacar un conjunto de requisitos que se agrupan de la siguiente forma:

Requisitos funcionales

- Gestión de notas
 - crear nota
 - ver todas las notas
 - consultar nota
 - editar nota
 - borrar nota

- Gestión de recordatorios
 - crear recordatorio
 - ver todas las recordatorios
 - consultar recordatorio
 - editar recordatorio
 - borrar recordatorio

- Gestión de categorías
 - crear categoría
 - editar categoría
 - añadir nota a categoría
 - añadir recordatorio a categoría
 - eliminar nota de categoría
 - eliminar recordatorio de categoría
 - consultar notas de una categoría
 - consultar recordatorios de una categoría
 - eliminar categoría

- Gestión de usuario
 - registrar de usuario
 - logueo del usuario
 - consulta de los datos del usuario
 - deslogueo del usuario
 - borrado del usuario

Requisitos no funcionales

- Móvilidad de la herramienta
- Seguridad en el acceso a la herramienta

Aunque todas las funcionalidades se describen al inicio del proyecto se va a definir dos fases, en las que se van a agrupar las funcionalidades. Cada una de las fases debe acabar con una herramienta operativa:

1º fase

- Gestión de notas
- Gestión de recordatorios
- Gestión de usuarios
- Seguridad en el acceso a la herramienta
- Móvilidad de la herramienta (consulta de información)

2º fase

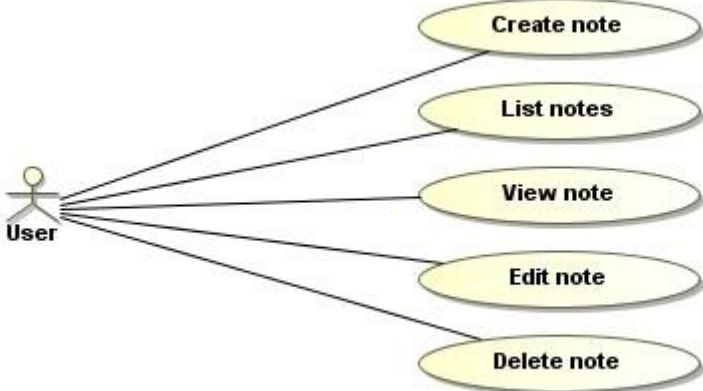
- Gestión de categorías
- Móvilidad de la herramienta (creación y edición de la información)

Una vez que tenemos definido el modelo del dominio y los requisitos del usuario se va a pasar a definir los casos de uso que marcarán el desarrollo de la aplicación.

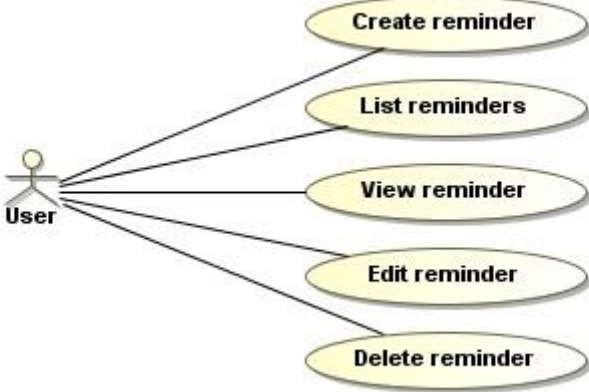
En esta etapa se van a definir los casos de uso de todas las fases del proyecto. Se hace de esta manera para disponer de todas las funcionalidades bien especificadas en la etapa de Análisis y Diseño.

Para la definición de los casos de uso nos basamos en el conjunto de requisitos funcionales. Se hace uso de la herramienta MagicDraw para crear los diagramas de caso de uso. Para que sea más fácil su manejo los hemos agrupado en funcionalidades. Estos son:

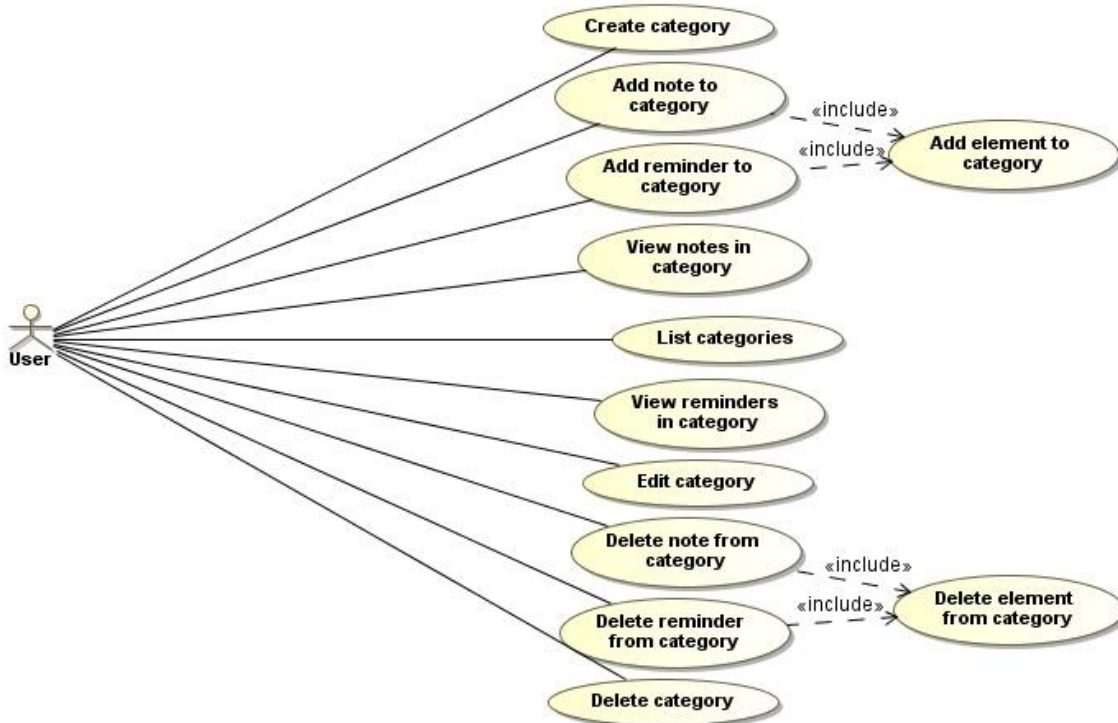
Gestión de notas:



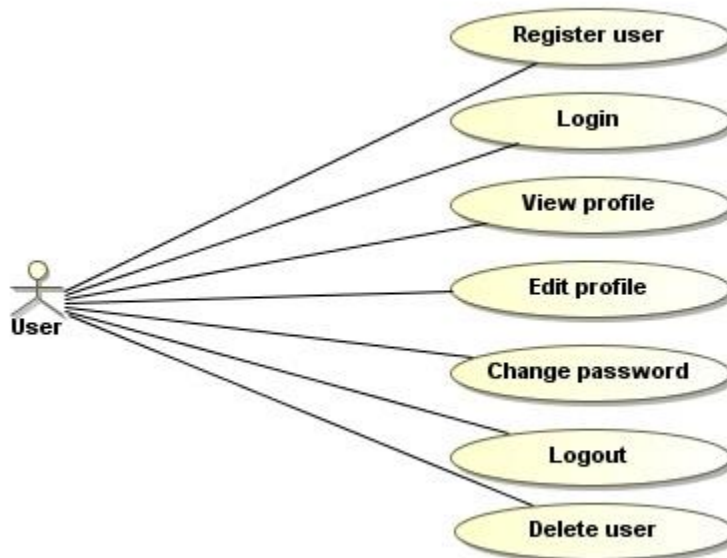
Gestión de recordatorios:



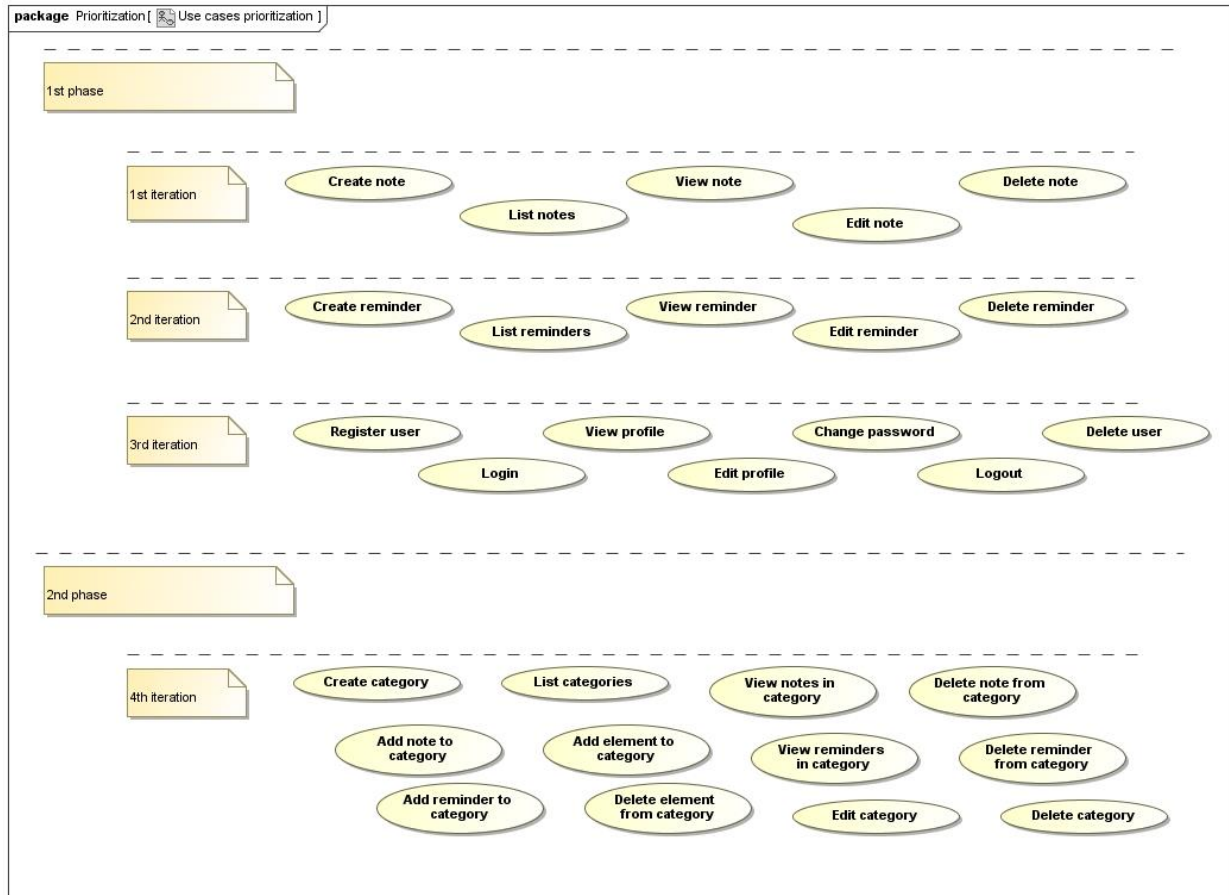
Gestión de categorías



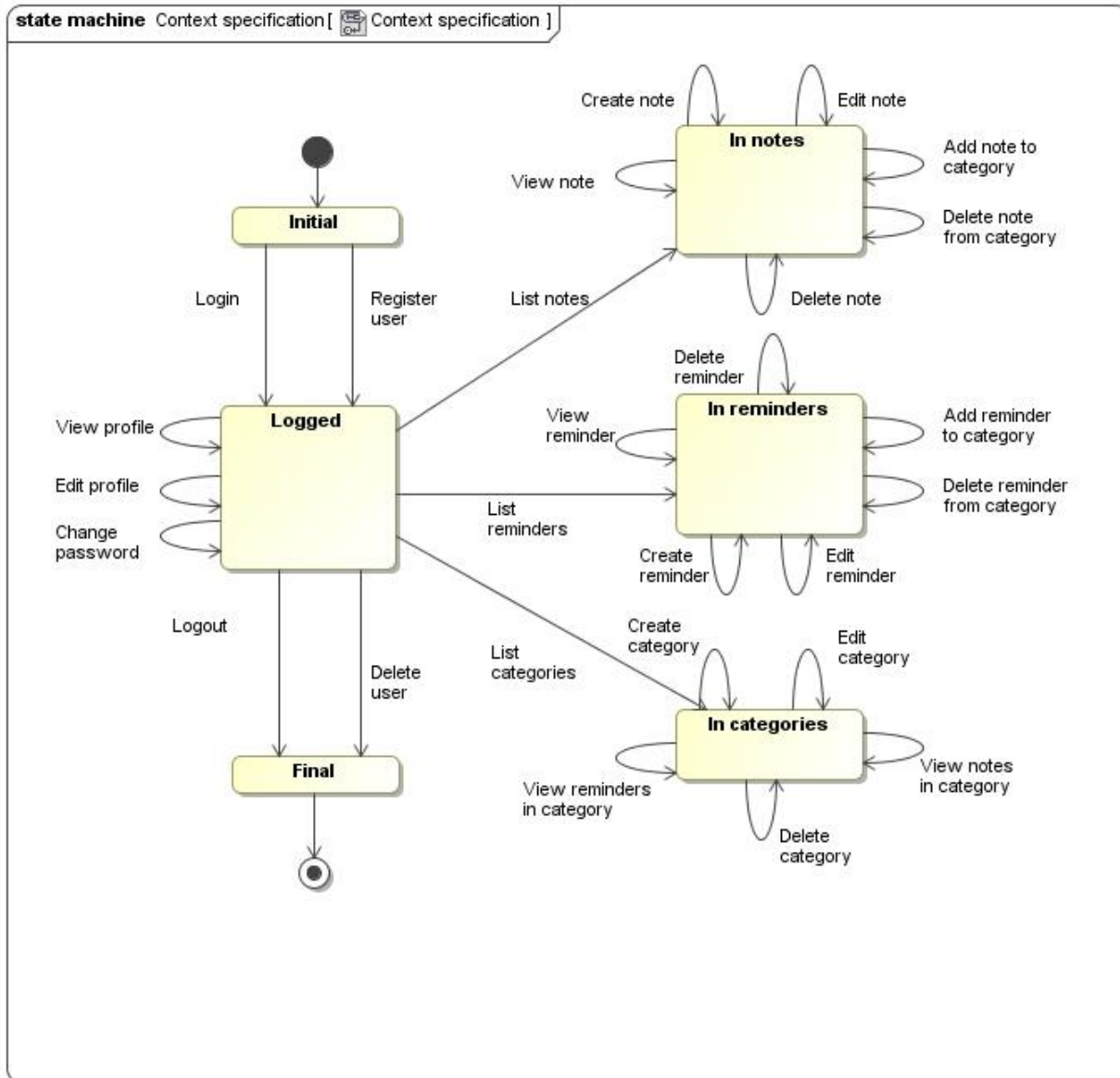
Gestión de usuarios:



Una vez hemos definido todos los casos de uso que nuestra aplicación tiene que llevar a cabo vamos a priorizarlos en función de las fases y en función a la iteración a la que pertenecen. Cuanto más arriba se encuentren más importantes son, si se encuentran en el mismo nivel significan que son de igual importancia.



A continuación se va a poner en conjunto los casos de uso que hemos localizado, para ello haremos un diagrama del contexto general de la aplicación donde situaremos todos los casos de uso que hemos descrito hasta ahora. Se hace uso de un diagrama de estados.



Al final de este documento, en el apéndice, se encuentran detallados cada caso de uso particular haciendo uso de diagramas de estados.

El siguiente paso es definir unos prototipos de la aplicación. Para ello se hace uso de los Mockups desarrollados con la herramienta web moqups.com

Algunos de los casos de uso no tiene una vista propia, ya que no requiere de interfaz gráfica para representar, un caso concreto es el de Logout. En otros casos varios casos de uso comparten vista para, como puede ser Editar perfil y Cambiar Contraseña.

Los mockups también se han definido en el apéndice de este documento.

Con esto queda terminada la etapa de Modelado y Requisitos. El conjunto de diagramas de caso de uso y de estado generado en esta etapa sirven de entrada para la siguiente fase: Análisis y Diseño.

Análisis y Diseño

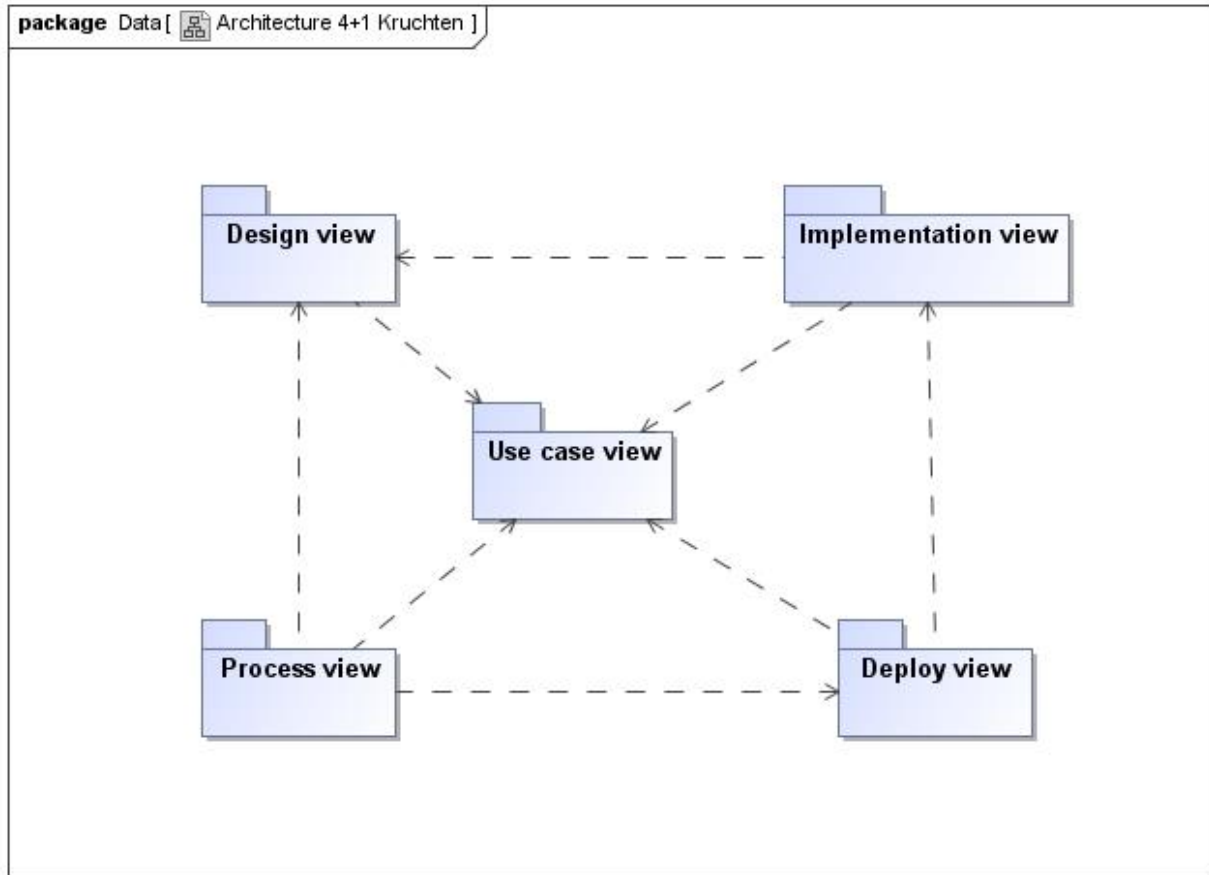
Una vez tenemos definidos los casos de uso de la aplicación, vamos a definir la arquitectura que nuestra aplicación va a tener. Para definir la arquitectura nos vamos a ayudar de los requisitos funcionales descritos en los casos de uso, y en los requisitos no funcionales.

Aunque en la etapa de requisitos anotamos y evaluamos todos los casos de uso definidos para el desarrollo la aplicación en sus ambas fases, a partir de ahora vamos a centrarnos principalmente en la primera fase aunque incluiremos también algunos conceptos de la segunda fase para obtener una base sólida. De esta manera obtendremos todas las entidades deducidas de la etapa del modelado, pero solo levantaremos la arquitectura en torno a las que incumben en la primera fase.

Al tratarse de una aplicación pequeña las etapas de Análisis y Diseño las vamos a hacer en una única etapa. Por el tamaño de la aplicación podemos hacer el diseño a la par que la comprensión del sistema.

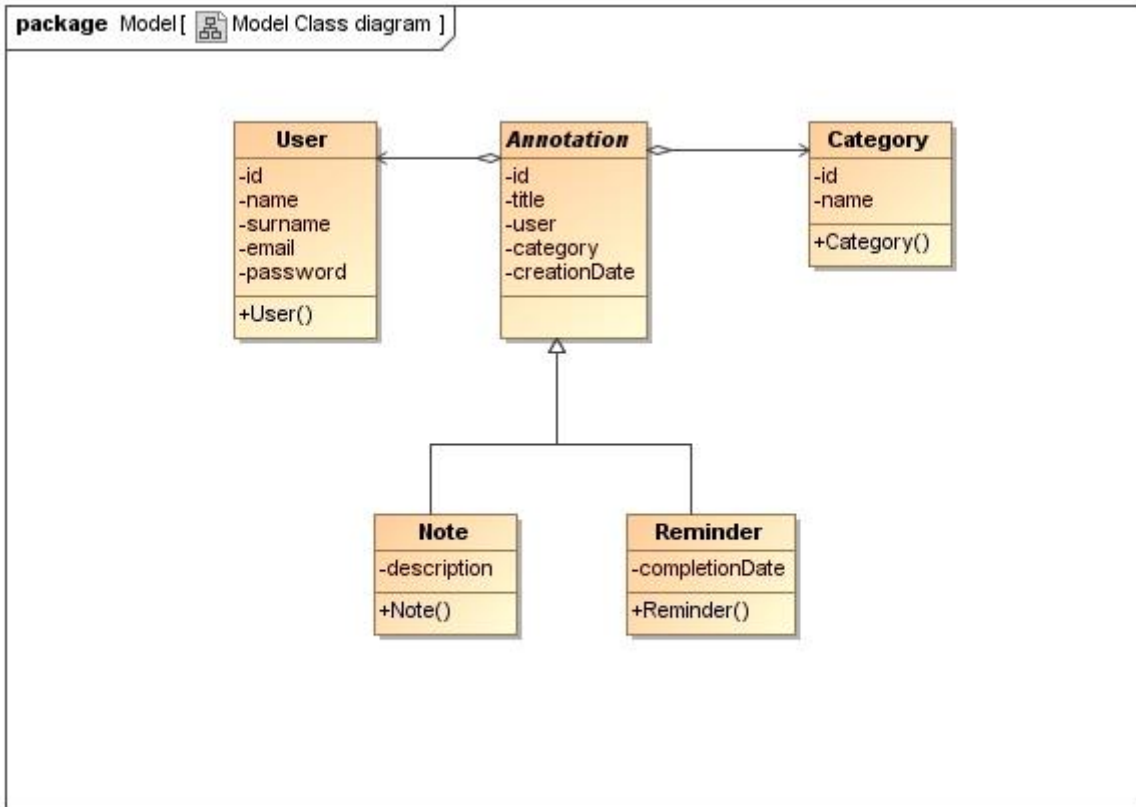
Vamos a describir el diseño de nuestro sistema mediante la definición de su arquitectura. Para definir la arquitectura nos vamos a ayudar del modelo de 4+1 vistas de Kruchten y para definir el modelo (o abstracción del sistema global) nos vamos a apoyar en 5 vistas que proyectan una perspectiva del modelo. Estas vistas son:

- Vista de casos de uso: usada por el usuario final. Ya definida.
- Vista de diseño: usada por arquitectos y analistas.
- Vista de implementación: usada por desarrolladores y manager.
- Vista de despliegue: usada por el ingeniero de sistemas.
- Vista de procesos: Usada por el integrador. No tiene incumbencia en este proyecto.



Una vez que ya hemos definido los casos de uso en la primera etapa de Modelado y Requisitos, en esta etapa vamos a centrarnos con el resto de las vista de la arquitectura que dependen de la vista de casos de uso: estas son las vistas de diseño, implementación, despliegue y proceso.

Empezamos con la vista de diseño. Lo primero que vamos a hacer obtener una visión general de todo nuestro sistema donde definiremos las clases que formarán la aplicación, para ello nos basaremos en el modelo del dominio (dos fases). De esta manera obtenemos el diagrama de clases del modelo de nuestra aplicación que nos ayudará a comprender su funcionamiento y relaciones:



Ahora que comprendemos el sistema, vamos a empezar a introducir conceptos más técnicos. Anteriormente hicimos un estudio de tecnologías, y ahora es donde entran en acción.

Antes de nada vamos a destacar algunas de esas técnicas, protocolos, patrones arquitectónicos que ya hemos definido pero que conviene conocer en detalle para el correcto diseño de la aplicación. Durante la definición del proyecto se ha decidido apostar por estas tecnologías, por ello, en cada decisión que tomemos hay que tenerlas en cuenta. También en este punto debemos tener en cuenta los requisitos no funcionales descritos durante la captura de requisitos. Así tenemos:

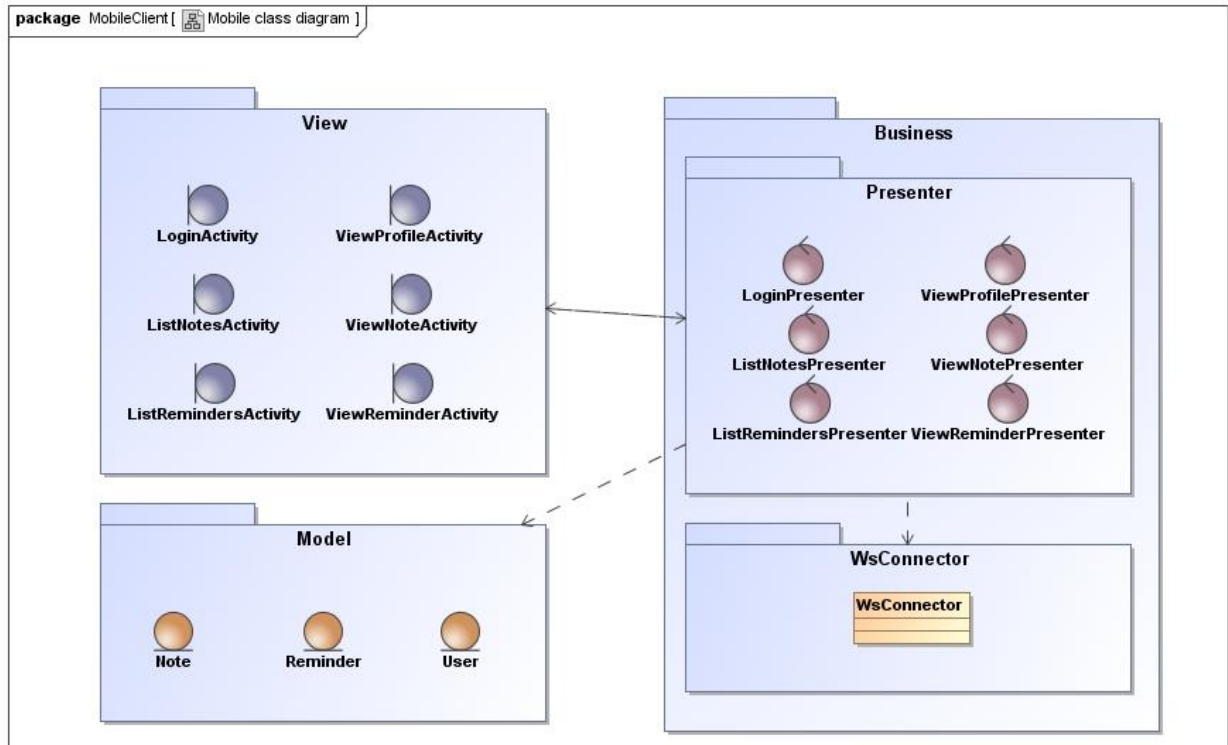
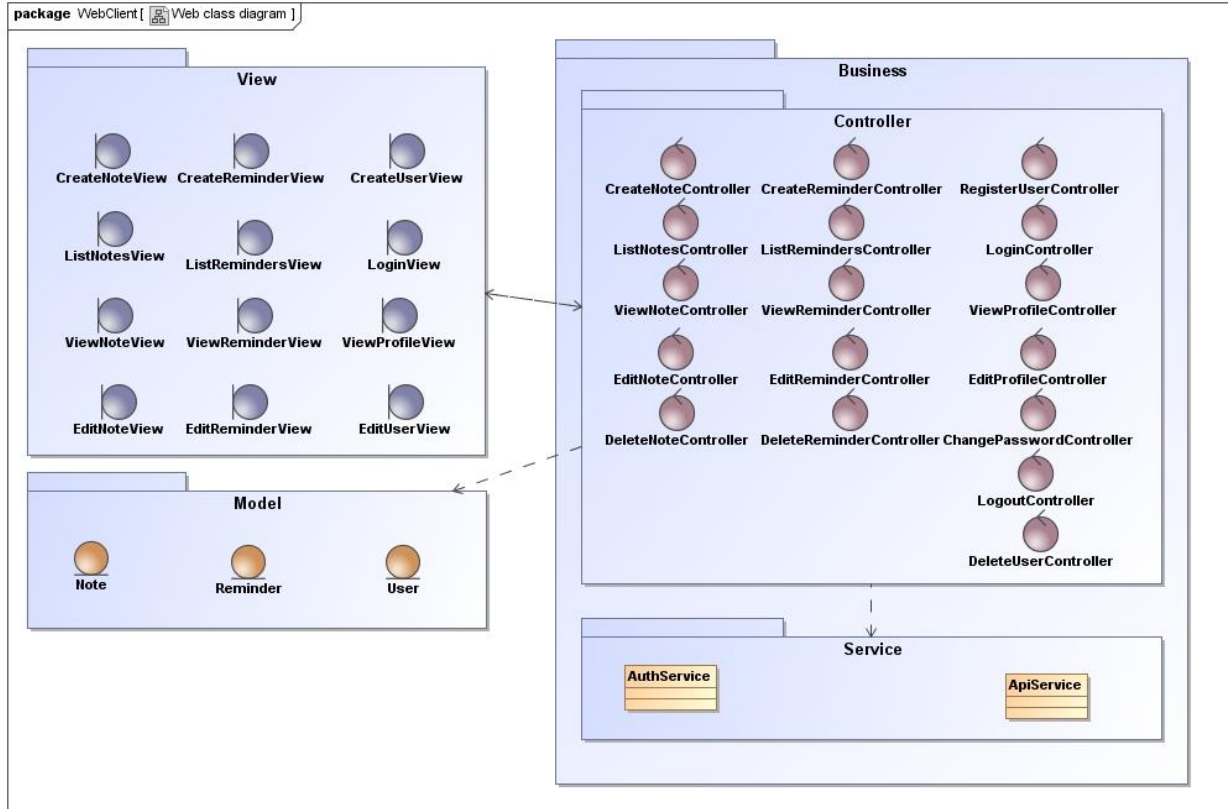
- AngularJS y Android
- MVC y sus variantes MVVM y MVP
- Servicios web. REST y JSON
- Autenticación basada en token. JWT
- Mapeo relacional de objetos. JPA

Vamos a analizarlas y adecuarlas a los requisitos de la aplicación. A medida que las descubramos iremos definiendo pequeñas parte de nuestro sistema y lo representaremos con diagramas de clases.

El primer requisito no funcional afirma que la aplicación debe ser móvil, por ello se plantea un diseño orientado a navegador, que sea ligero para que cualquier dispositivo, tanto sobremesa, como móvil sea capaz de soportarlo. También debe tener una interfaz responsive que se adapte al tamaño y resolución de la pantalla. Además, como complemento, se quiere realizar una aplicación nativa Android, que abarca un gran porcentaje del mercado de smartphones. Aquí entra en acción tecnologías web y móviles como AngularJS, Android.

Tanto la aplicación web como la móvil se encargan de gran parte de la lógica de la aplicación dejando al servidor únicamente la tarea de almacenar y ofrecer los datos.

Para estructurar adecuadamente la lógica en los dispositivos clientes hacemos uso del patrón arquitectónico MVC, de esta manera conseguimos separar los conceptos de Vista, Modelo y Controlador. Realmente ninguna de las dos aplicaciones hace uso directamente de dicho patrón, sino que implementan dos de sus variantes. La aplicación móvil hace uso MVP donde P es Presentador y la aplicación utiliza la variación MVVM donde VM es Viewmodel. Ambas implementaciones difieren de su predecesora en que hay menor dependencia entre la Vista el Modelo. En ellas, tanto el Presentador como el Viewmodel se encargan de la lógica. La diferencia entre ambas es que MVVM permite más de una Vista por Viewmodel, en cambio MVP asigna una única Vista a cada Presentador.



Para permitir esa movilidad, y al disponer de múltiples dispositivos, necesitamos que la información se mantenga actualizada en todo momento. Para ello se opta por una arquitectura donde la información no se almacene en el cliente, sino que esté en otra entidad aparte, un servidor. También responde a medidas de seguridad tales como disponibilidad, pues en caso de fallo del dispositivo, la información no se pierda.

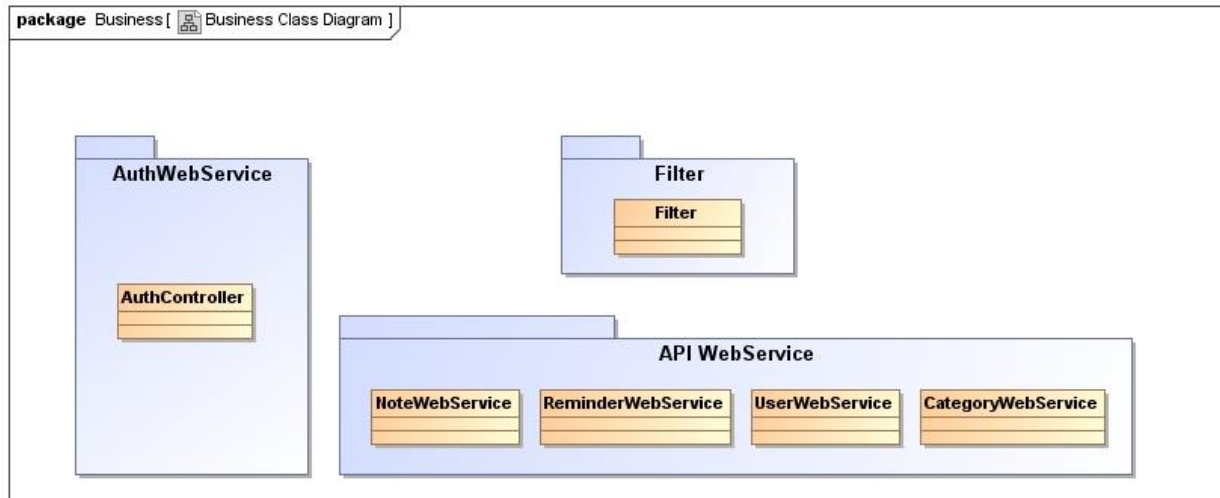
Ahora necesitamos que haya una comunicación continua entre las aplicaciones cliente y la aplicación servidora. Para reutilizar la funcionalidad en ambos clientes optamos por una comunicación basada en servicios web haciendo uso de REST y JSON. Los servicios web publican una serie de rutas a las que los clientes lanzan peticiones y responden en forma de respuesta JSON con los recursos solicitados. Esto permite que la aplicación web sea más ligera en la carga y procesamiento de las vistas, así como la comunicación con el servidor.

Otro requisito no funcional era que fuera el autor el único que pudiese visualizar sus anotaciones. De esta manera hay que implementar medidas de seguridad tales como autenticación. Para ello creamos un sistema de logueo de usuario. Pero esto no es suficiente y hay que garantizar que el usuario registrado es el único que accede a su información privada. Para ello se definen algunas rutas de los servicios web como privadas, y solo si un usuario está registrado el cliente puede lanzar peticiones sobre estas rutas para así ver su información. Para implementar este mecanismo hacemos uso de la autenticación basada en Token usando el protocolo JWT.

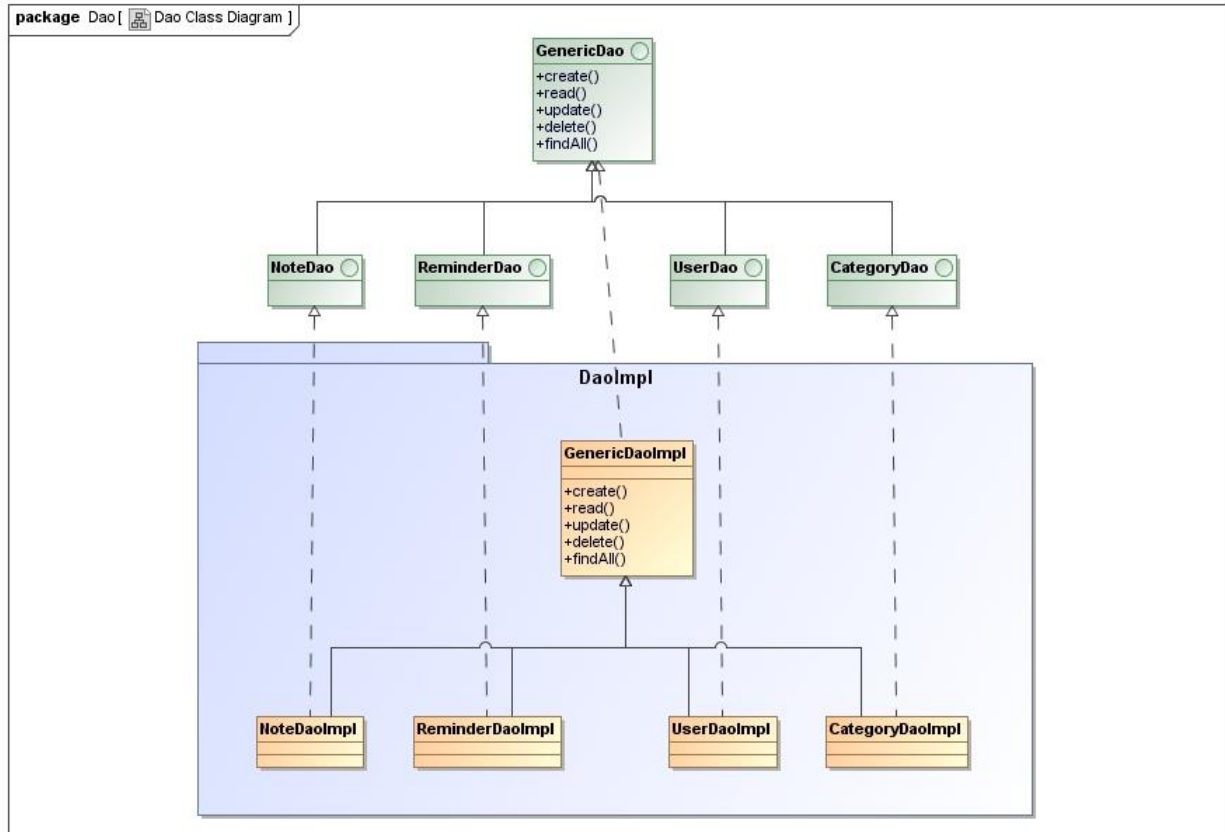
De esta forma hemos conseguido un acceso seguro, pero no la comunicación entre el cliente y el servidor. Para securizar la comunicación hacemos que el protocolo HTTP vaya sobre SSL, obteniendo HTTPS. De esta forma impedimos que terceras personas puedan ver los datos de logueo y realizar un ataque de robo de sesión. También impedimos que tenga acceso a la información confidencial a través de un ataque Man-in-the-middle.

Hemos dicho que en el servidor dispone de rutas privadas y públicas. En las públicas los servicios web ofrecen recursos que no hace falta estar registrado en la aplicación para consultarlos. En cambio las privadas devuelven los recursos que pertenecen al autor exclusivamente. Para montar este sistema hacemos uso de los Servlets de Spring y de Filters de manera que se define un servlet público, y otro privado. Las peticiones que se lanzan sobre el privado, pasan antes por un filtro que comprueba que si el usuario está

logueado en la aplicación. El servlet público contiene los servicios web de autenticación. En cambio el servlet privado contiene los servicios web que devuelven información sensible del paciente, por ello necesitan de un filtro de acceso.

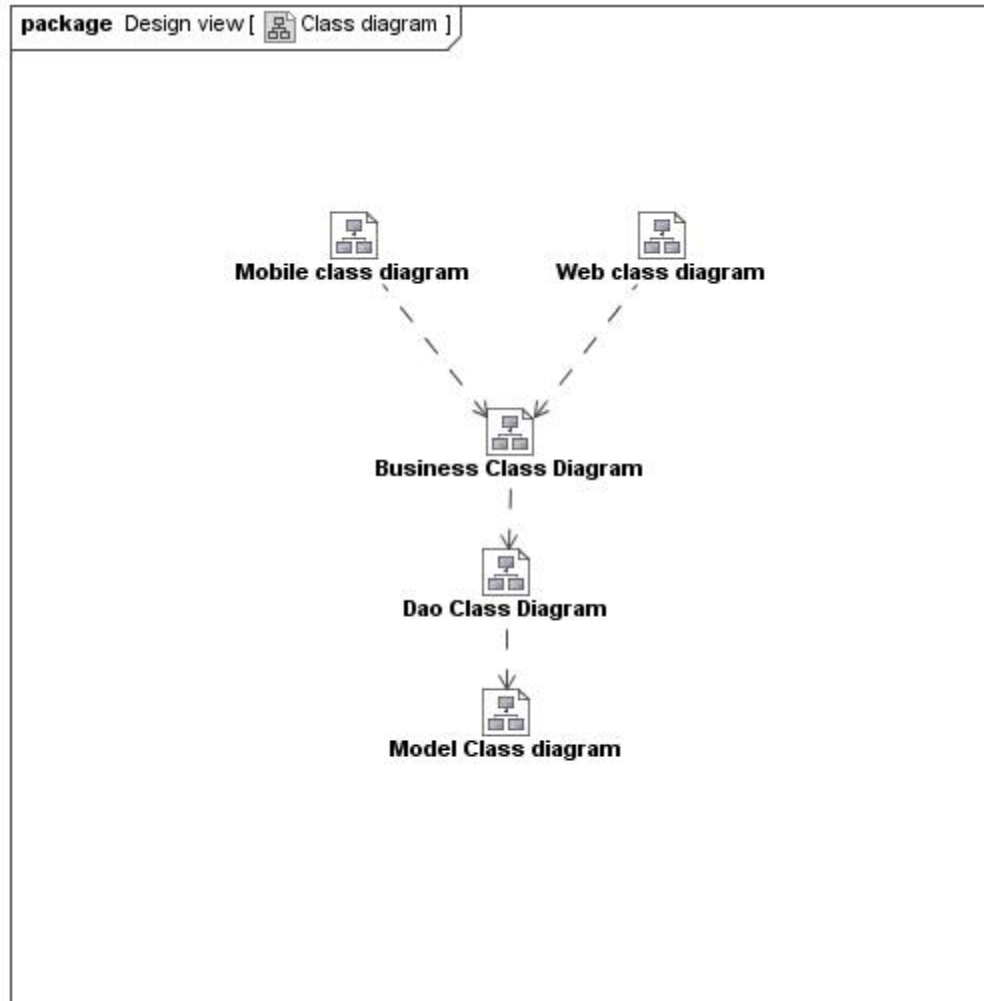


Toda la información que los clientes piden/envían al servidor vía servicios web, se debe consultar/almacenar de una base de datos. Para que la aplicación acceda a la BD se hace uso de una técnica conocida como Object Relational Mapping (ORM), definida por la especificación JPA e implementada por Eclipselink. JPA hace uso de patrones DAO para acceder a la información del modelo del sistema.

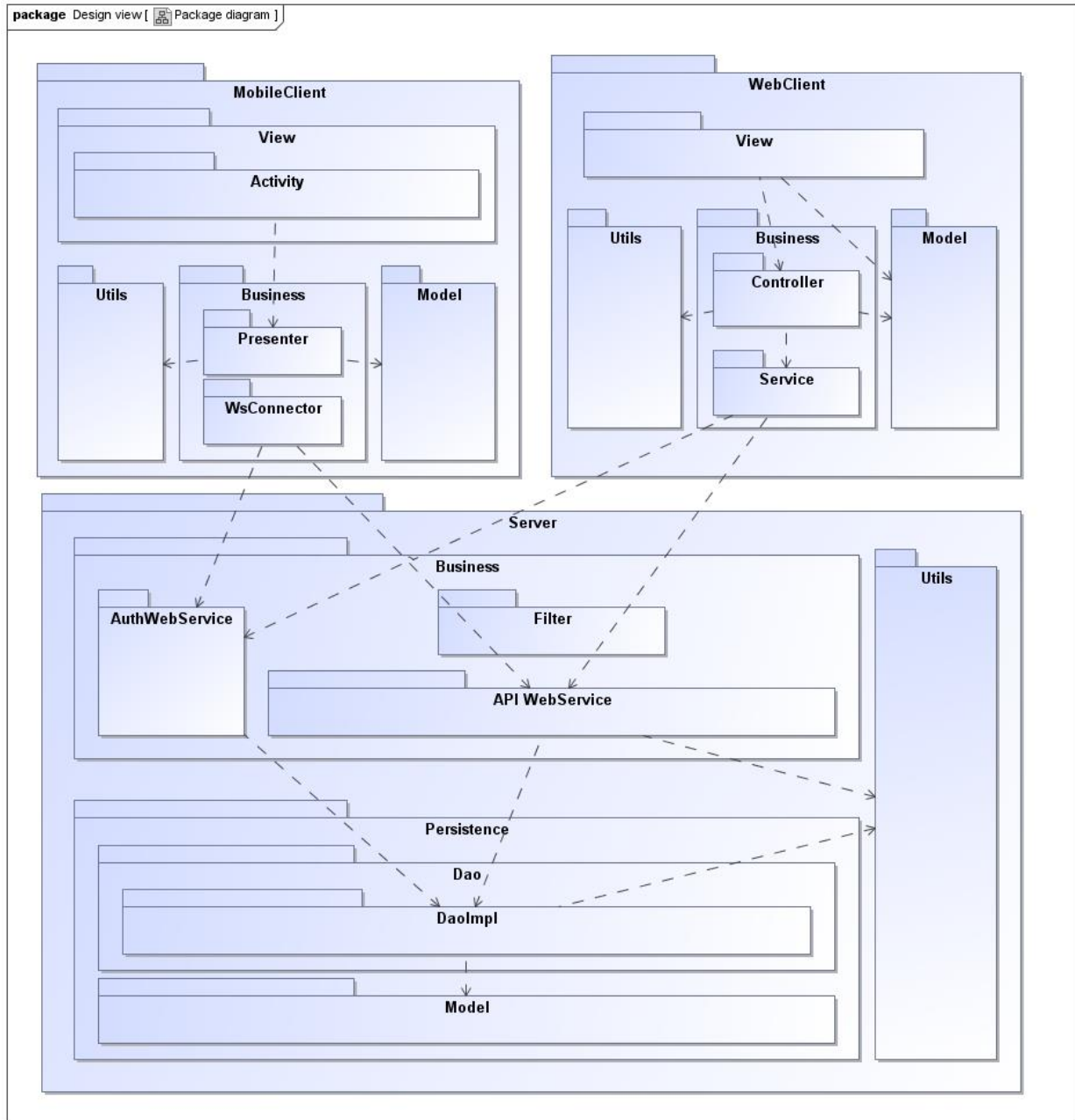


De esta manera he terminado de definir la arquitectura de la aplicación, empezando por las capas superiores y acabando por las inferiores. Esta arquitectura nos permite además el desarrollo en paralelo de las tres aplicaciones, pues las tres usan de interfaz con las demás los servicios web.

Vemos en conjunto los 4 grandes componentes arquitectónicos que forman la aplicación y que hemos presentado en este apartado. Se une un quinto componente que es el que define el modelo y que presentamos anteriormente:



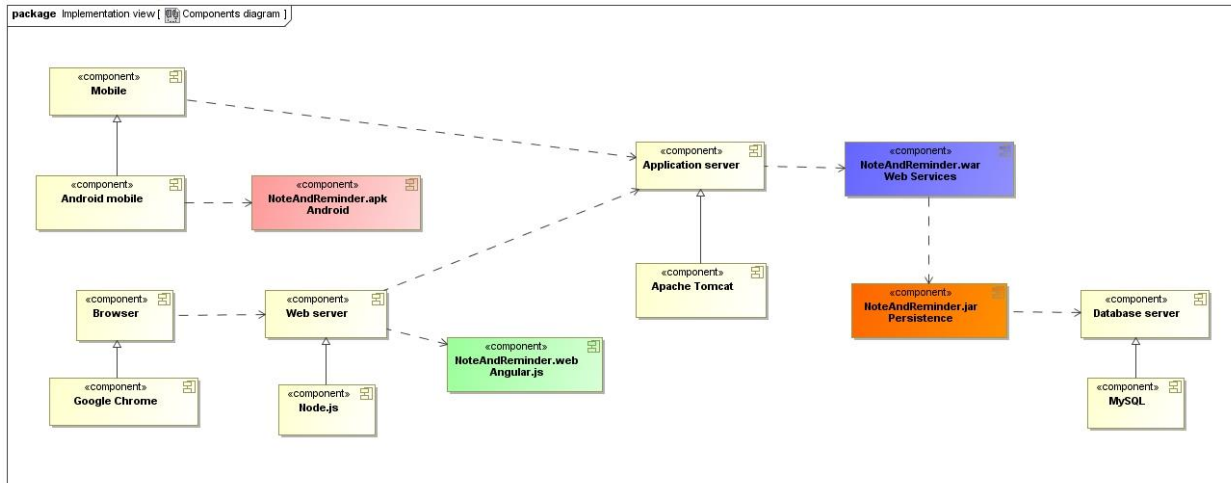
Con todos estos diagramas conocemos las clases que a priori formarán nuestras aplicaciones. Sirve de base para la etapa de desarrollo, aunque es posible que sufran algunas modificaciones más adelante. Veamos ahora esos mismas clases pero desde un punto de abstracción por encima mediante un diagrama de paquetes. En este caso veremos un mapa de la aplicación entera.



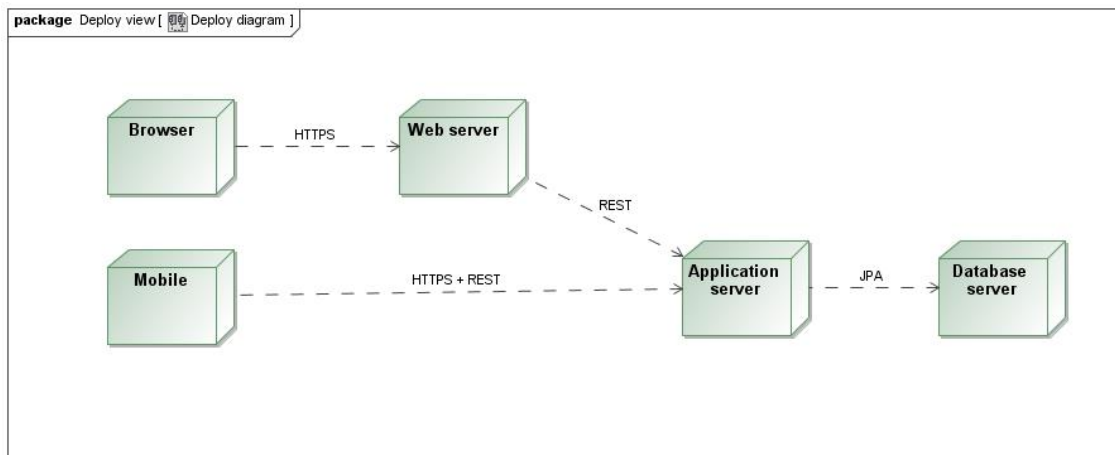
El siguiente paso es definir la vista de Implementación de la arquitectura definida por el modelo de 4+1 vistas de Kruchten.

Ahora entran en acción otra serie de tecnologías que también definimos al inicio de este documento. Son las aplicaciones sobre las que correrán nuestros componentes. Incluyen el navegador web, el sistema operativo móvil, el servidor web, el servidor de aplicaciones y el servidor de base de datos.

Como ya hemos mencionado, hemos definido 4 componentes a lo largo de esta sección. En el 4º incluiremos el modelo, pues trabajan de la mano en la persistencia de datos. En el siguiente diagrama de componentes vemos cómo se alojan los mismos en las distintas tecnologías.



A continuación describimos la vista de Despliegue de nuestra arquitectura. De esta manera, dando un paso más podemos abstraernos del todo y visualizar los nodos que forman nuestra aplicación. Estos nodos corresponden con las aplicaciones donde se alojan nuestros componentes. Para ello se ha creado el siguiente diagrama de despliegue. En él se pueden apreciar las tecnologías de comunicación entre nodos.



Con esto hemos terminado la etapa de Análisis y Diseño. Los diagramas que hemos definidos sirven como entrada para la siguiente etapa: Desarrollo.

Desarrollo

Una vez se tiene toda la documentación necesaria recopilada en las etapas previas, como una definición de las posibles tecnologías y herramientas, y la definición de la arquitectura a través en la obtención requisitos del usuario y los diagramas de la etapa de Análisis y Diseño se pasa a la etapa de Desarrollo donde se harán realidad nuestras aplicaciones.

Como ya hemos mencionado varias veces este proyecto implementa tres aplicaciones distintas, por lo tanto la etapa de desarrollo está dividida en tres fases donde se crea cada una de esas aplicaciones. El orden seguido ha sido el siguiente: primero se ha desarrollado la aplicación servidora, a continuación la aplicación web y por último la aplicación móvil. Aunque se han desarrollado en serie una tras otra, hay algunos puntos en los que se ha habido desarrollo en paralelo de dos de estas aplicaciones por necesidad de una de ellas de modificar las demás.

Como se ha mencionado en la etapa de Análisis y Diseño, únicamente se va a implementar los requisitos definidos en la primera fase del proyecto, por tanto se excluye la gestión de categorías y la aplicación avanzada Android.

Para ello vamos a describir las tareas realizadas en este proyecto. Al final entraremos en detalle de algunos de los problemas y soluciones implementadas en esta etapa de desarrollo que han resultado interesantes y que por ello les prestaremos mayor atención.

El código fuente de las aplicaciones se encuentra alojado en Github y se puede acceder a él libremente. La cuenta es <https://github.com/jcombarros>. Además de en este documento se puede hacer un seguimiento de las tareas realizadas mediante un seguimiento de los commits de GIT que aparecen en Github. Los proyectos se encuentra en:

- Aplicación servidora:
<https://github.com/jcombarros/NoteAndReminderApi>
- Aplicación web:
<https://github.com/jcombarros/NoteAndReminderWeb>
- Aplicación móvil:
<https://github.com/jcombarros/NoteAndReminderAndroid>

Todas las tareas de configuración de entornos y aplicaciones se definirán en los apéndices de este documento.

Las tareas realizadas durante el desarrollo de la aplicación servidora han sido:

- Creación del proyecto haciendo uso de los arquetipos de Maven.
- Configuración de la aplicación servidora en Eclipse incluyendo todas las tecnologías utilizadas en ella.
- Comprobación de que todas las tecnologías funcionan correctamente en conjunto.
- Crear paquete de modelo con las entidades en él definidas.
- Añadir anotaciones de JPA al modelo.
- Comprobar la creación de tablas en la base de datos.
- Definición de script para poblar la base de datos con información por defecto.
- Definir capa de DAOs y sus implementaciones.
- Añadir las implementaciones de los DAO y sus métodos CRUD.
- Definir paquete de Utils con las funcionalidades de JSON
- Añadir anotaciones JSON para evitar bucles.
- Crear servlet para los servicios de las API.
- Definir URL de la API.
- Crear controladores para los servicios web de la API: Notes, Reminders, Users and Categories.
- Definir los métodos CRUD en los controladores.
- Añadir dependencias a través de Spring.
- Probar los servicios web desde el navegador.
- Añadir JWT y definir la entidad Token.
- Crear servlet para los servicios de Autenticación.
- Definir URL para la Autenticación.
- Crear controlador para los servicios web de Autenticación.
- Definir los métodos del controlador de Autenticación: Register, Login y Logout.
- Crear filtro para el servlet de la API para comprobar que el usuario está logueado a través de la información de las Cabeceras.
- Crear métodos adicionales en los DAO de la API: getByUser.
- Crear métodos adicionales para los controladores de la API.
- Probar los servicios web desde el navegador.

Las tareas realizadas durante el desarrollo de la aplicación web han sido:

- Creación del proyecto haciendo uso de Yeoman para crear su scaffolding.
- Configuración de la aplicación web y de todas las tecnologías utilizadas en ella.
- Comprobación de que todas las tecnologías funcionan correctamente en conjunto.
- Comprobar la funcionalidad básica de la web al crearla con Yeoman.
- Crear Ruta, Vista y Controlador usando Generator de Yeoman para la primera funcionalidad.
- Definir la vista usando Bootstrap.
- Definir el controlador añadiendo los objetos y métodos al Viewmodel.
- Editar vista para recuperar objetos del Viewmodel y llamar a los métodos desde los botones.
- Crear la factoría que almacenará los métodos para llamar a los Servicios REST de nuestra API. Usamos Generator de Yeoman.
- Añadir la URL para llamar a la API.
- Añadir a la factoría los métodos del CRUD a través del paquete de AngularJS \$http.
- Inyectar dependencia de la factoría de la API en el Controlador.
- Editar el Controlador para llamar a los servicios web a través de los métodos de la factoría.
- Crear archivo con los Utils de la aplicación.
- Seguir definiendo Rutas, Vistas y Controladores con nuevas funcionalidades de los casos de uso de la API. Hacer uso de los métodos de la factoría para llamar a los servicios web.
- Comprobar que la aplicación web funciona correctamente usando los servicios web la aplicación servidora.
- Definir Layout de la aplicación web así como la página de Home.
- Definir los botones de Back en todas las vistas de la aplicación.
- Añadir restricciones en las rutas así como definir vista para Error 404.
- Crear Rutas, Vistas y Controladores para funcionalidades de Autenticación a través de los Generators d AngularJS.
- Definir las Vistas usando Bootstrap.
- Definir los Controladores añadiendo los objetos y métodos al Viewmodel.
- Editar las Vistas para recuperar objetos del Viewmodel y llamar a los métodos desde los botones.

- Crear la factoría que almacenará los métodos para llamar a los Servicios REST de Autenticación. Usamos Generator de Yeoman.
- Añadir URL para llamar a la Autenticación.
- Añadir los métodos de Autenticación través del paquete de AngularJS \$http.
- Definir Viewmodel global con contiene la información de usuario logueado.
- Inyectar dependencia de la factoría de Autenticación a los Controladores.
- Editar los Controladores para llamar a los servicios web a través de los métodos de la factoría.
- Añadir el envío de Cabeceras en los métodos de la factoría de la API para comprobar la correcta autenticación.
- Comprobar que la aplicación web funciona correctamente usando los servicios web la aplicación servidora.
- Comprobar los circuitos que realizará un usuario para usar la aplicación: Logueo / Consulta y Modificaciones / Deslogueo.

Las tareas realizadas durante el desarrollo de la aplicación móvil han sido:

- Creación del proyecto haciendo uso del Asistente de Eclipse.
- Configuración de la aplicación móvil.
- Comprobación de la aplicación funciona correctamente.
- Creación de página Home (Activity y Layout).
- Edición del Layout de la página Home.
- Definir Android Application custom con contendrá la información de usuario logueado.
- Creación de página Login (Activity y Layout).
- Definición del Layout de la página de Login.
- Creación de la Conexión con el servidor mediante AsyncTask para hacer uso de los servicios web.
- Definir los métodos GET y POST de la conexión.
- Añadir funcionalidad a la Activity de Login haciendo uso de la Conexión.
- Añadir permisos de Internet a la aplicación.
- Añadir Cabeceras a los métodos de la Conexión.
- Refactorizar paquete de Conexión.
- Importar Modelo de la aplicación de servidora (Notas, Recordatorios, Usuarios y Categorías).

- Creación de las páginas Listar notas y Listar recordatorios (Activities y Layouts).
- Edición del Layouts de las páginas Listar notas y Listar recordatorios.
- Implementar Adapter y Item-Layout para mostrar las Listas en los Layouts
- Añadir funcionalidad a las Activities de Listar notas y Listar recordatorios haciendo uso de la Conexión.
- Añadir botones en Layout de la página Home a las funcionalidades de Listar y Login.
- Creación de las páginas Ver nota y Ver recordatorio (Activities y Layouts).
- Edición del Layouts de las páginas Ver nota y Ver recordatorio.
- Añadir funcionalidad a las Activities de Ver nota y Ver recordatorio haciendo uso de la Conexión.
- Modificar la Conexión para mostrar fallos o errores de conexión.
- Añadir enlaces en Layouts de las páginas de Listar a las funcionalidades de Ver nota y ver recordatorio.
- Creado Menú para la Activity de la página Home.
- Definición de los ítems del menú para las funcionalidades de Logout y Ver perfil.
- Creación de página Ver perfil (Activity y Layout).
- Definición del Layout de la página de Ver perfil.
- Añadir funcionalidad a la Activity de Ver perfil haciendo uso de la Conexión.
- Crear funcionalidad de Logout (No tiene Vista).
- Refactorizar la aplicación entera para hacer uso del patrón MVP.

Durante el proceso de desarrollo se ha ido realizando un proceso continuo de refactorización para conseguir dejar el código con la mejor presencia posible, esto es que tenga una buena estructura y una buena legibilidad.

En el apéndice de este proyecto se puede ver un conjunto de capturas de pantalla con las ventanas implementadas, tanto en la aplicación web como en la móvil.

Una vez que se han terminado de implementar todas las funcionalidades descritas en los casos de uso, se ha llegado al punto donde se obtiene como resultado tres aplicaciones totalmente funcionales que responden a las necesidades que se describieron al inicio durante la toma de requisitos. El

siguiente paso es probar de manera conjunta todas las funcionalidades para descubrir pequeños detalles que se hayan podido pasar durante la etapa de desarrollo. Por ello damos fin a esta etapa y comienza la siguiente: las Pruebas.

Pruebas

Una vez ha concluido la etapa de desarrollo, entra en acción la etapa de Pruebas, en este proyecto cobra de mayor importancia pues no se han desarrollado pruebas unitarias durante el desarrollo que permita sacar a la luz errores. Por ello hay que realizar un intensivo plan de pruebas de integración que compruebe de manera conjunta las funcionalidades implementadas.

Para comprobar el plan de pruebas se toma de referencia los casos de uso detallados y descritos mediante diagramas de estado en la etapa de Requisitos y Modelado. Además de probar el circuito correcto de cada caso de uso, también debe probarse todas sus variantes para cerciorarnos de que todo funciona como es debido. Para obtener una visión de los casos de prueba se puede ver en el apéndice los diagramas de estado.

Tras concluir con el plan de pruebas se ha comprobado que no han salido errores que afecten a los circuitos que los usuarios de la aplicación realizarán para manejar la aplicación de manera normal.

De esta manera se finaliza con la etapa de Pruebas, de manera que hemos obtenido una aplicación totalmente funcional y acorde a las necesidades que se buscaba cumplir al inicio del proyecto.

Por ello tan solo quería realizar el despliegue de la aplicación para tener en producción la aplicación desarrollada. De esta manera se pasaría a una siguiente etapa de mantenimiento pero que ya se escapa al alcance de este proyecto.

También hay que recordar que pese a que se analizaron todas las necesidades del proyecto, solo se han implementado las de la primera fase. Por tanto a partir de aquí se debería volver atrás para empezar con el desarrollo de esta segunda fase. Al igual que la etapa de Mantenimiento, la realización de la segunda fase se escapa al alcance del proyecto.

Lecciones aprendidas y Futuras mejoras

Como se ha mencionado a final de la etapa de Pruebas, el desarrollo del proyecto ha quedado finalizado en cuanto en tanto a lo acordado. No por ello significa que el proyecto haya terminado.

Aunque no se haga en la mayoría de los casos es bueno y conveniente realizar una última etapa de análisis para cerciorarnos de aquellas cosas que se han hecho bien y las que conviene revisar para futuros proyectos.

Algunas de estas lecciones aprendidas son las siguientes:

- Realizar una buena planificación desde el comienzo del proyecto para poder valorar acorde avanza este que se están siguiendo las metas establecidas en los tiempos acordados. Aunque en este proyecto no ha habido una planificación exhaustiva debido a su pequeño tamaño y que es de un solo miembro, sí se marcaron unas pautas temporales para llevar a cabo cada uno de los pasos necesarios. Estos eran: el estudio de tecnologías, la definición y análisis de los requisitos, el desarrollo de cada una de las aplicaciones y la realización de la memoria. Cabe destacar que los tiempos se han seguido correctamente y se han finalizado todos ellos en tiempo.
- Realizar una etapa de análisis que realmente sea útil durante la etapa de desarrollo para intentar anticiparnos a los posibles problemas que luego puedan surgir. En este caso sí se ha realizado de manera correcta la etapa de análisis y diseño. Se ha decidido apostar por invertir tiempo en ella y luego durante el desarrollo se han visto los beneficios de haberla realizado detenida y detalladamente.
- Uno de los grandes aliados durante la etapa de desarrollo es disponer de una herramienta como GIT para el control del código fuente. Aunque no estemos atados a GIT si es conveniente el uso de este tipo de herramientas, no solo como backup frente a errores o pérdidas de código, sino también como herramienta para revisar cómo se han desarrollado ciertas etapas pasadas y que pueden ser útiles en el futuro. Todas las aplicaciones del proyecto se han desarrollado haciendo uso de GIT.

- Otra de las grandes consideraciones que hay que tener en cuenta en la etapa de desarrollo es un interés en realizar el código de manera correcta para ello hay que hacer un esfuerzo en revisarlo continuamente para descubrir dónde se puede mejorar este. Por ello se ha querido hacer hincapié en el proyecto en una continua refactorización del código para intentar dejar un código lo más limpio posible, así cuando en futuras etapas se desee revisar el código este sea legible y esté bien estructurado.
- Otra de las etapas por las que todo proyecto debe pasar es por un estudio detallado de las posibles tecnologías que las aplicaciones deberán hacer uso. Esto permite que luego durante el diseño de la arquitectura, las decisiones no sean muy complicadas ni demasiado comprometidas. El estudio detallado implica un interés activo no solo al inicio del proyecto sino durante la el día a día del desarrollador para conocer en todo momento las nuevas tecnologías que salen a la luz en el mercado.
- Además de conocer las tecnologías existentes, también hay que tener un conocimiento al menos básico de las tecnologías seleccionadas para el desarrollo del proyecto. Un estudio previo detallado y la realización de pequeños ejercicios que saquen a la luz las fortalezas de las tecnologías permite que durante la etapa de desarrollo se conozca la manera en que se quiere llevar a cabo el desarrollo de la aplicación. Esto permite que podamos anticiparnos a futuros problemas de codificación, que en muchos casos se resolverán de manera inadecuada.
- En proyectos que agrupan una gran cantidad de tecnologías es necesario dedicar un tiempo suficiente a la integración de todas tecnologías para comprobar que todas ellas funcionan de manera correcta al operar conjuntamente. Es importante conocer las versiones de cada tecnología para prevenir posibles conflictos con otras.
- Cabe destacar la importancia que tienen las pruebas unitarias durante el desarrollo de un proyecto. Aunque era un proyecto pequeño y de poca complejidad, se debería haber hecho uso de esta herramienta que facilita el desarrollo y permite descubrir fallos de manera temprana. Una de las lecciones a aprender es no importa el tiempo que dure el proyecto, pero es necesario dedicar un tiempo suficiente a desarrollar

estas pruebas, que a la larga son beneficiosas. En proyectos complejos y que los requisitos son cambiantes, marcarán la diferencia entre el éxito y el fracaso del proyecto.

Una vez hemos conocido qué se puede mejorar en la realización de un proyecto, también vamos a proponer algunas mejoras en las aplicaciones para hacerlas más potentes y más atractivas para los usuarios. Además de implementar la segunda fase donde desarrollaremos la gestión de categorías y realizaremos una aplicación móvil totalmente funcional, otras posibles mejoras son:

- Aviso mediante correo electrónico de que se acerca la fecha de completitud de un recordatorio.
- Incluir notificaciones en la aplicación Android que avisen de que un recordatorio está próximo.
- Realizar un plugin para los navegadores que igualmente permita la notificación de un recordatorio.
- Poder compartir notas con otros usuarios de manera privada, o definir notas públicas que cualquiera pueda ver.
- Permitir introducir imágenes en la descripción de las notas.
- Además de categorías definir etiquetas para marcar cada nota o recordatorio y que puedan ser buscadas de manera más sencilla.

Conclusión

Como se ha comentado al inicio de este documento, con la elaboración de este proyecto se ha buscado un aprendizaje en diversas tecnologías, herramientas y arquitecturas. Además, poner en práctica dichos conocimientos como conjunto que colaboren entre ellos, analizándolos y adecuándolos en base a las necesidades de la aplicación NoteAndReminder.

A lo largo y ancho de este documento se habla de proyecto, no de aplicación. Se ha querido unir en un único ente un conjunto de tecnologías actuales como son los servicios REST, las implementaciones del patrón MVC, una arquitectura cliente-servidor con varios clientes usando AngularJS y Android.

Aunque parte de importante de este proyecto se lo ha llevado el desarrollo de las aplicaciones también se ha querido hacer con mucho cuidado una implementación de su arquitectura. Esto no va atado a aplicaciones webs o móviles, esto va de ingeniería software y de cómo debe realizarse cualquier proyecto que quiera llegar a buen puerto.

Se ha querido poner en común todo lo aprendido a lo largo del máster, desde el uso de patrones, el desarrollo de un proyecto siguiendo una metodología, tecnologías del lado cliente y del lado servidor, tecnologías móviles y seguridad. Se ha querido demostrar que todo ello junto y bien mezclado da lugar a proyecto completo, totalmente funcional y bien desarrollado.

Bibliografía

API REST usando JPA y Spring

Java y tecnologías de backend

- Material de la asignatura de JEE

Crear proyecto usando Maven

- Video de configuración de la asignatura de JEE
<https://www.youtube.com/watch?v=hJyeSbjMLZk&feature=youtu.be>
- <https://hop2croft.wordpress.com/2011/04/25/creacion-de-un-proyecto-maven/>
- <https://spring.io/guides/gs/maven/>

Configuración de JPA, Servlets e inyección de dependencias

- Material de la asignatura de JEE

Servlets y Sevicios REST con Spring

- <https://spring.io/guides/gs/rest-service/>
- <http://www.cursoangularjs.es/doku.php>

Filters de Spring

- <http://learningviacode.blogspot.com.es/2013/12/delegatingfilterproxy-and-spring.html>

JWT

- <http://jwt.io/>
- <https://github.com/jwtk/jjwt>
- <http://code.tutsplus.com/es/tutorials/token-based-authentication-with-angularjs-nodejs--cms-22543>
- <https://thinkster.io/angularjs-jwt-auth/>

Autenticación

- <https://auth0.com/docs/quickstart>
- <https://spring.io/guides/tutorials/spring-security-and-angular-js/>

Aplicación web usando AngularJS

Video de AngularJS

- AngularJS Fundamentals In 60-ish Minutes:
<https://www.youtube.com/watch?v=i9MHigUZKEM>

Libros de AngularJS

- AngularJS (O'Reilly) - Brad Green & Shyam Seshadri
- <https://thinkster.io/a-better-way-to-learn-angularjs/>

Tutoriales de AngularJS

- <https://docs.angularjs.org/tutorial/>
- https://code.angularjs.org/1.3.15/docs/tutorial/step_00

AngularJS API

- <https://docs.angularjs.org/api>

Instalar Yeoman, Bower y Grunt

- <http://yeoman.io/codelab/setup.html>
- <http://juristr.com/blog/2014/08/node-grunt-yeoman-bower/>

Generators de Yeoman para AngularJS

- <https://github.com/yeoman/generator-angular>

Spring Rest para AngularJS

- <https://spring.io/guides/gs/consuming-rest-angularjs/>
- <http://www.toptal.com/angular-js/a-step-by-step-guide-to-your-first-angularjs-app>

Consumir Servicios web desde AngularJS

- <http://www.pedroalonso.net/blog/2014/04/06/angularjs-resources-for-net-developers/>

Login usando AngularJS

- <http://code.tutsplus.com/es/tutorials/creating-a-web-app-from-scratch-using-angularjs-and-firebase--cms-22391>

Aplicación móvil usando Android

Libro Android

- **Learning Android (O'Reilly)** - Marko Gargenta

API de Android

- <http://developer.android.com/guide/index.html>

Configuración de proyecto Android

- Material de la asignatura de DASM

MVP en Aplicaciones Android

- <http://fernandocejas.com/2014/09/03/architecting-android-the-clean-way/>
- <http://www.limecreativelabs.com/mvp-android/>
- <http://antonioleiva.com/mvp-android/>

Consumir Servicios REST desde Android

- Material de la asignatura de DASM
- <http://gpmess.com/blog/2014/05/28/volley-usando-webservices-en-android-de-manera-sencilla/>
- <https://spring.io/guides/gs/consuming-rest-android/>

Usar ListViews en Android

- <https://amatellanes.wordpress.com/2013/04/14/ejemplo-de-listview-en-android/>

Menús para Android

- <http://developer.android.com/guide/topics/ui/menus.html>

Conceptos interesantes

- <http://sauceio.com/index.php/2014/07/angularjs-data-models-http-vs-resource-vs-restangular/>
- <http://tylermcginnis.com/angularjs-factory-vs-service-vs-provider/>
- <http://www.html5rocks.com/en/tutorials/es6/promises/>

Solución a problemas

- <http://www.w3schools.com/>
- <http://stackoverflow.com/>

Apéndice

Versiones

Como hemos visto anteriormente, se han utilizado una gran cantidad de tecnologías y herramientas para desarrollar las tres aplicaciones que componen este proyecto. En este apartado vamos a definir las versiones utilizadas, en la mayoría de los casos se ha optado por la última versión estable del producto que existía en el momento de la definición del proyecto. Estas son:

Tecnologías

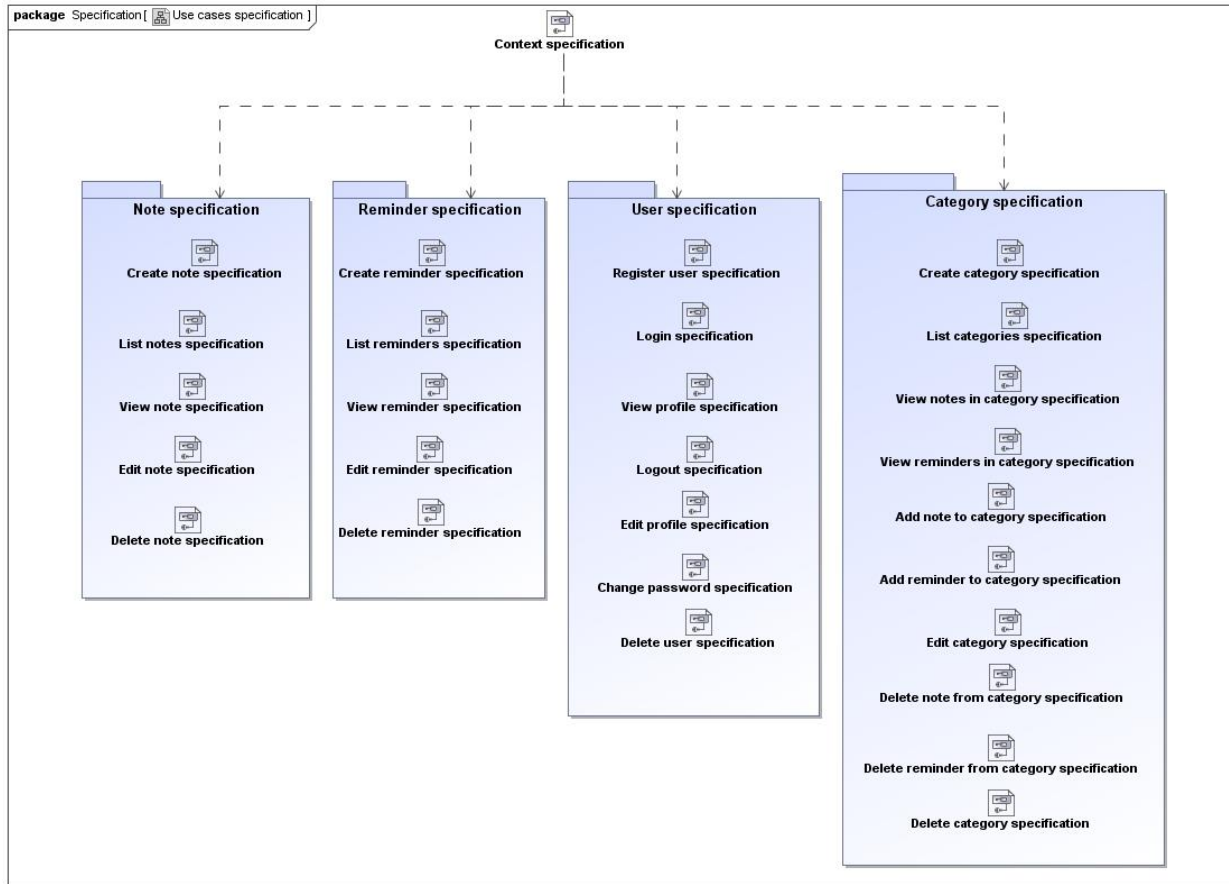
- Web Services - REST: es un estándar de W3C
- JSON: es un estándar de RFC 7156 y ECMA-404
- MVC: variantes MVP y MVVM (no estandarizado)
- Java: JDK v1.8.0_20
- EclipseLink JPA: v2.5
- Spring: v4.2.6
- MySQL: v5.6.24
- JWT: es un estándar de W3C CSS3
- HTML: es un estándar de ISO/IEC 15445 y W3C HTML5
- CSS: es un estándar de W3C CSS3
- JS: es un estándar de ECMAScript 6
- AJAX: es un estándar de W3C
- AngularJS: v1.3.15
- jQuery: v2.1.4
- Bootstrap: v3.3.4
- Android: Kitkat v4.2.2 (API 19)

Herramientas:

- Cmdr: v1.1.4
- Git: v1.9.5
- Git extensions: v2.48.05
- EGit: v3.4.1 (Eclipse for Web developers) y v2.2.0 (Eclipse for Android developers)
- Github: versión web
- Kdiff3: v0.9.98
- Apache Tomcat: v8.0.21
- Eclipse for Web developers: Luna 4.4.1
- Maven: 4.0.0
- Advanced REST Client: v3.1.9
- HeidiSQL: v9.2
- Google Chrome: v43.0
- Sublime Text 2: 2.0.2
- Node.js: v0.12.2
- NPM v2.7.4
- Yeoman: v1.4.6
- Bower: v1.4.1
- Grunt: 0.1.13
- Eclipse for Android developers: Juno 23.0.2

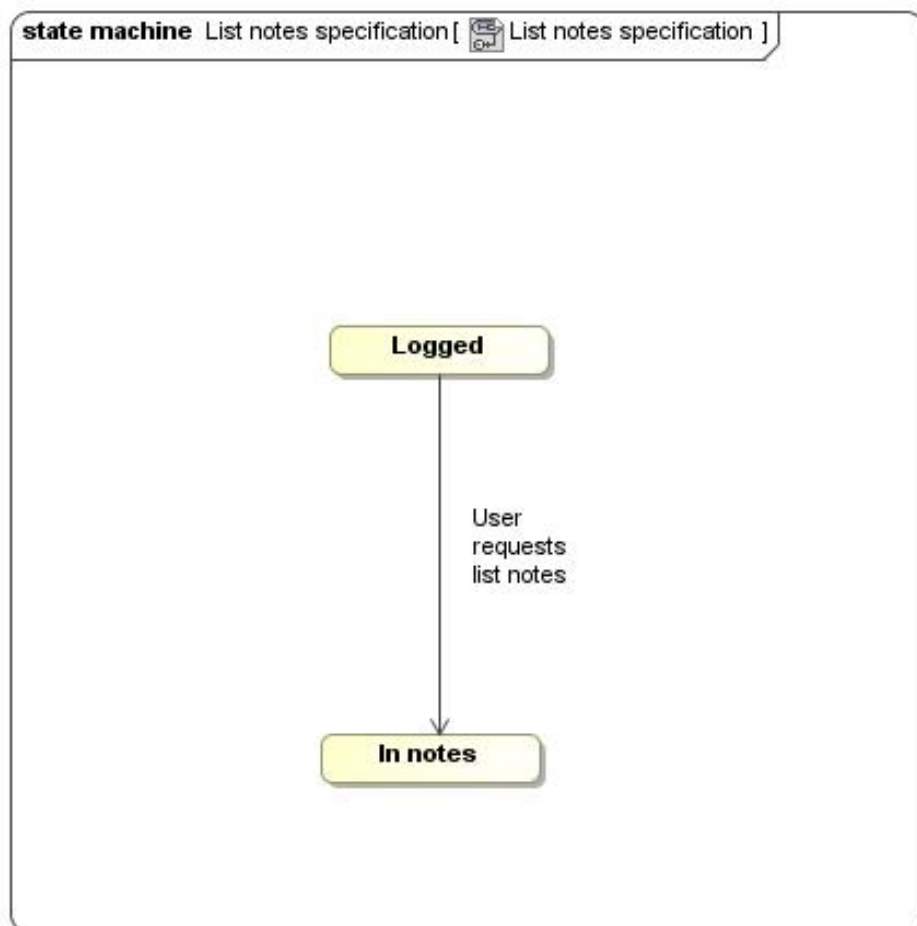
Casos de uso detallados

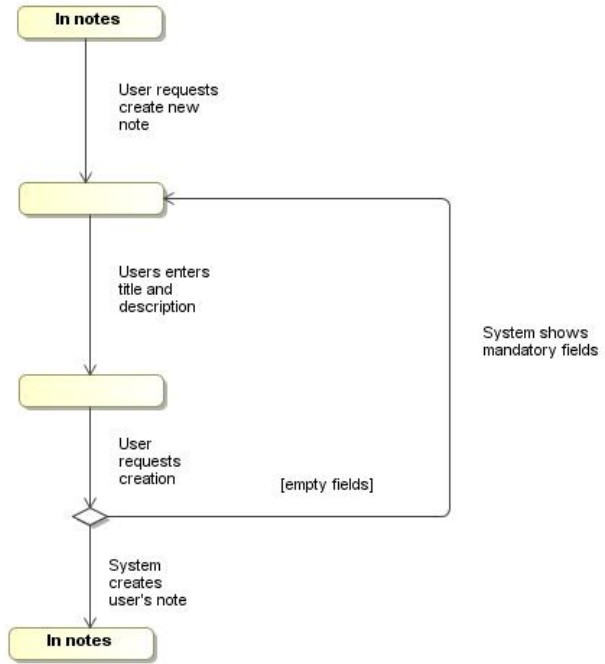
En este apartado se definen todos los casos de uso detallados mediante su diagrama de estado donde se muestra su circuito principal (de arriba a abajo) y todos los circuitos alternativos posibles.



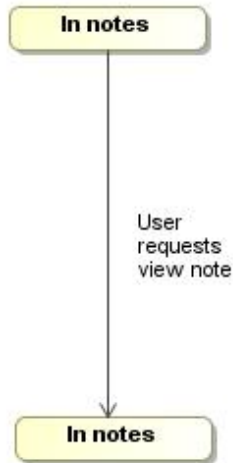
Solo vamos a incluir los casos de uso detallados de Listar, Crear, Ver, Editar y Borrar para la Gestión de Notas porque para Gestión de Recordatorios es igual. También incluiremos todos los de Gestión de Categorías y Gestión de Usuarios. Si se desean ver todos en detalle se puede ir al proyecto de MagicDraw. Los casos de uso detallados son:

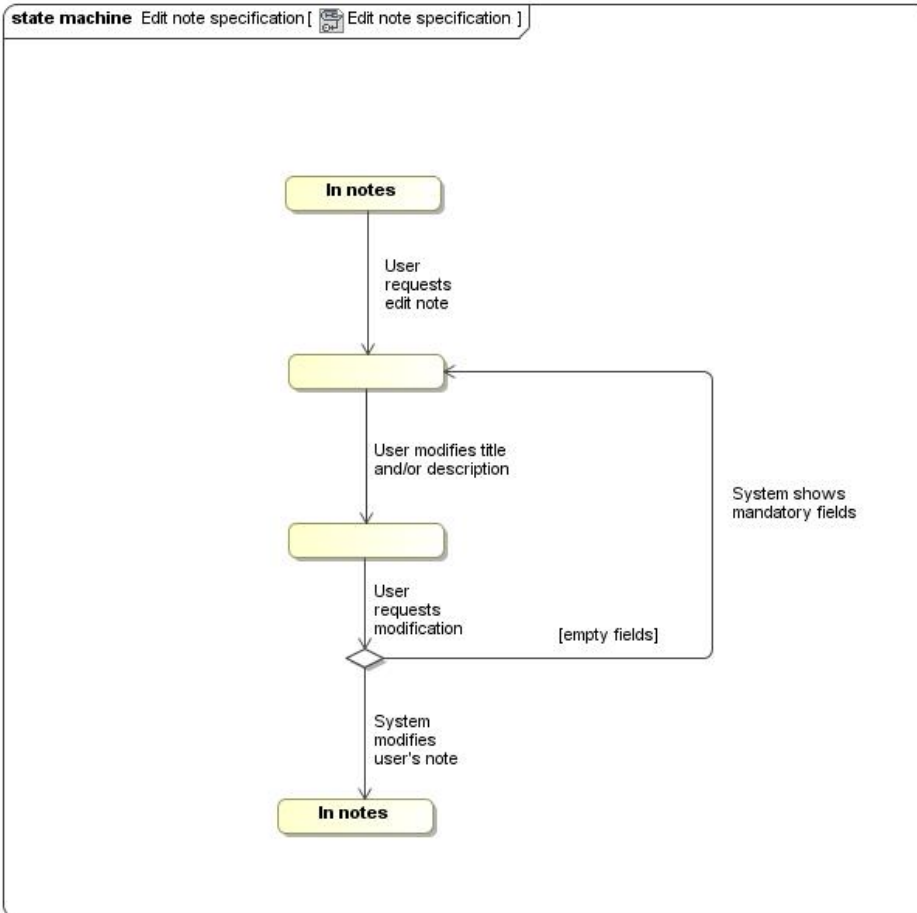
Gestión de Notas (igual para Gestión de Recordatorios):

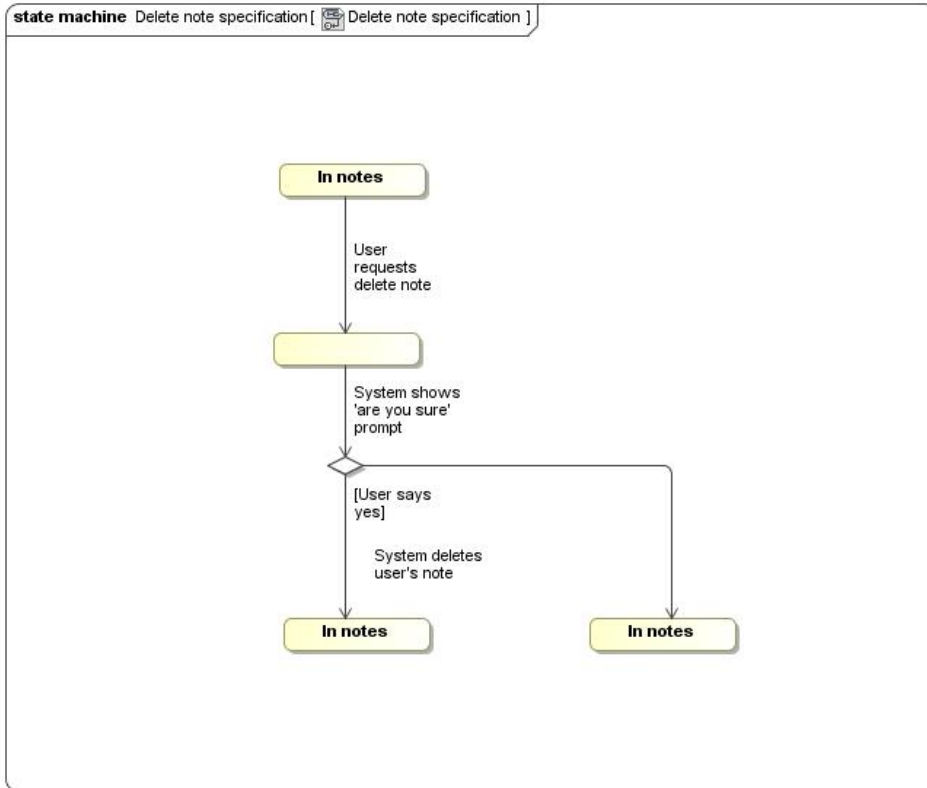




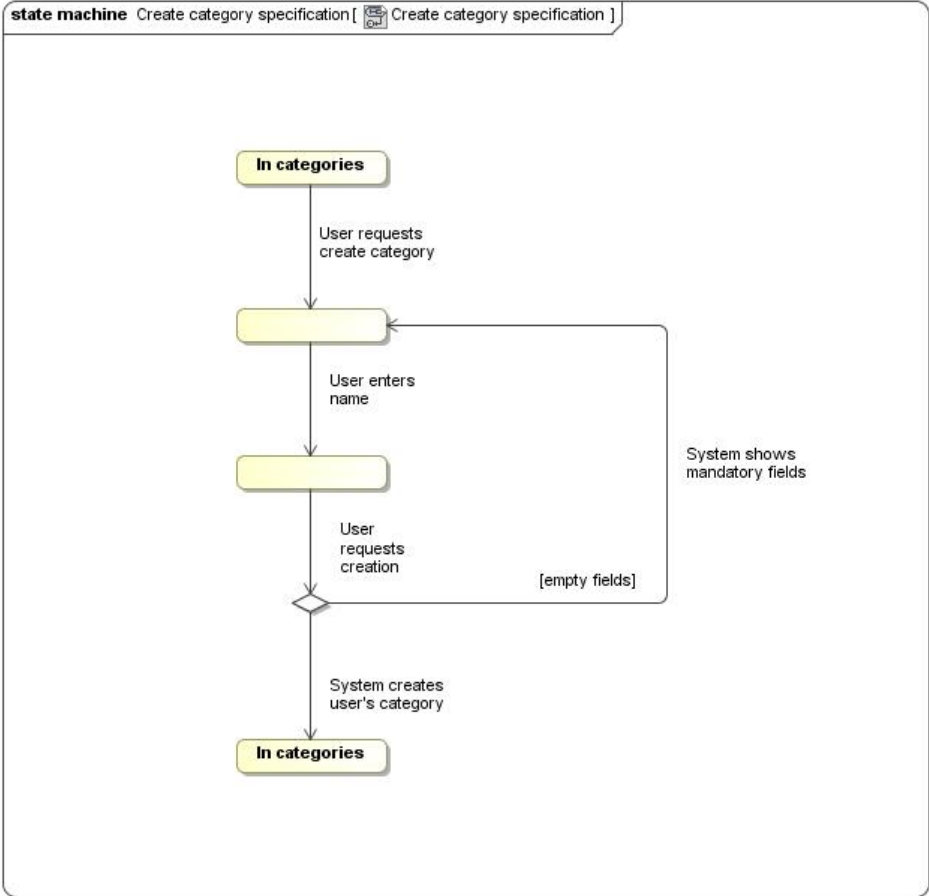
state machine View note specification [View note specification]




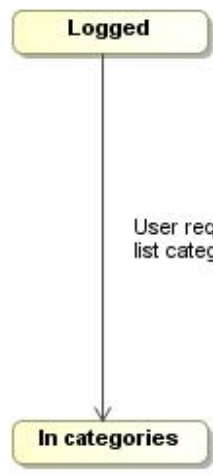





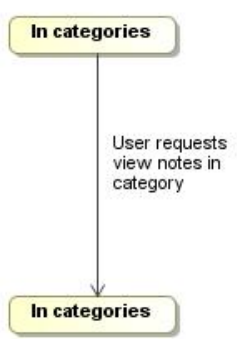
Gestión de categorías:



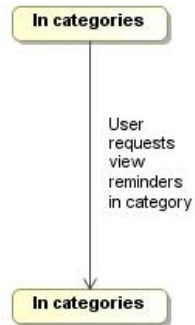
state machine List categories specification [ List categories specification]



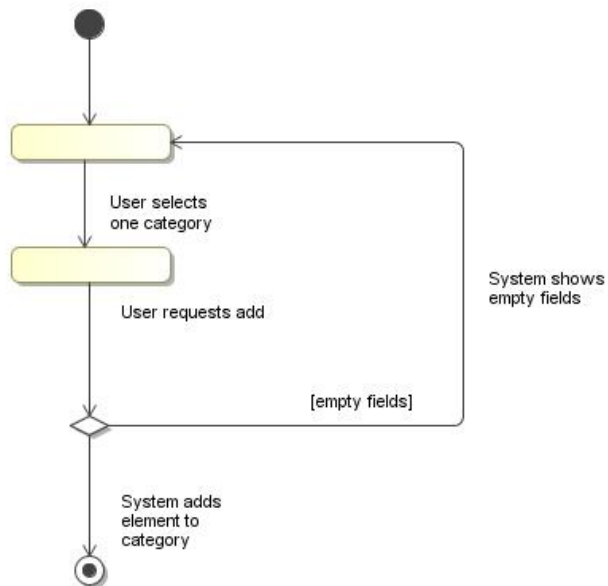
state machine View notes in category specification [ View notes in category specification]



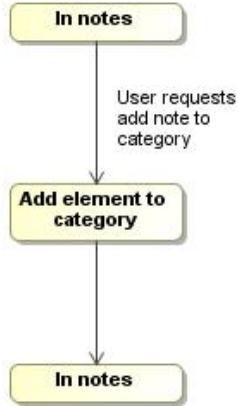
state machine View reminders in category specification [View reminders in category specification]



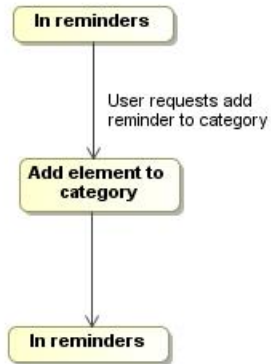
state machine Add element to category specification [Add element to category specification]



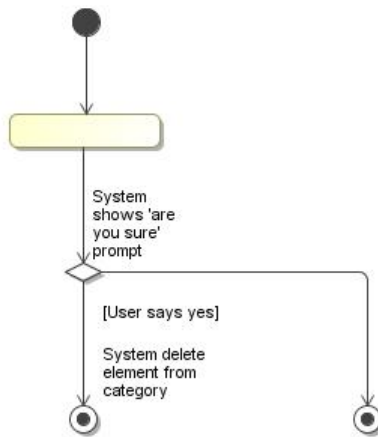
state machine Add note to category specification [Add note to category specification]



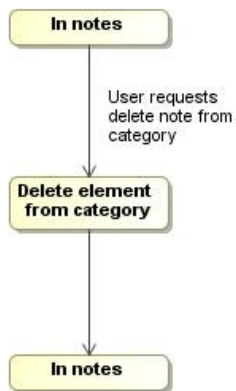
state machine Add reminder to category specification [Add reminder to category specification]



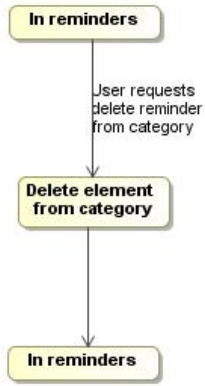
state machine Delete element from category specification [Delete element from category specification]



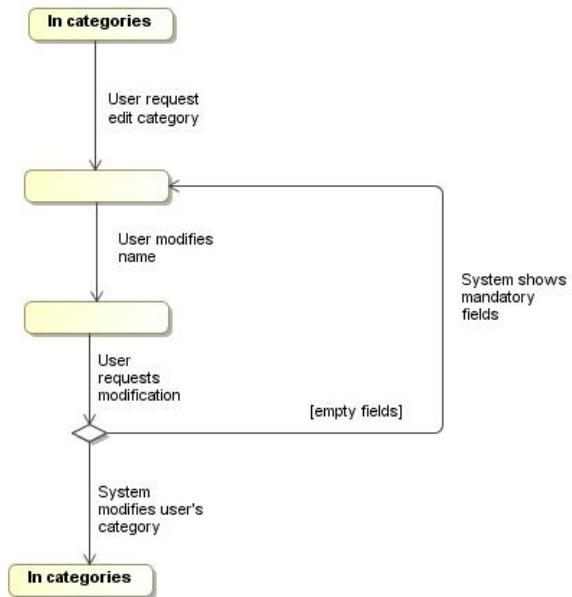
state machine Delete note from category specification [Delete note from category specification]

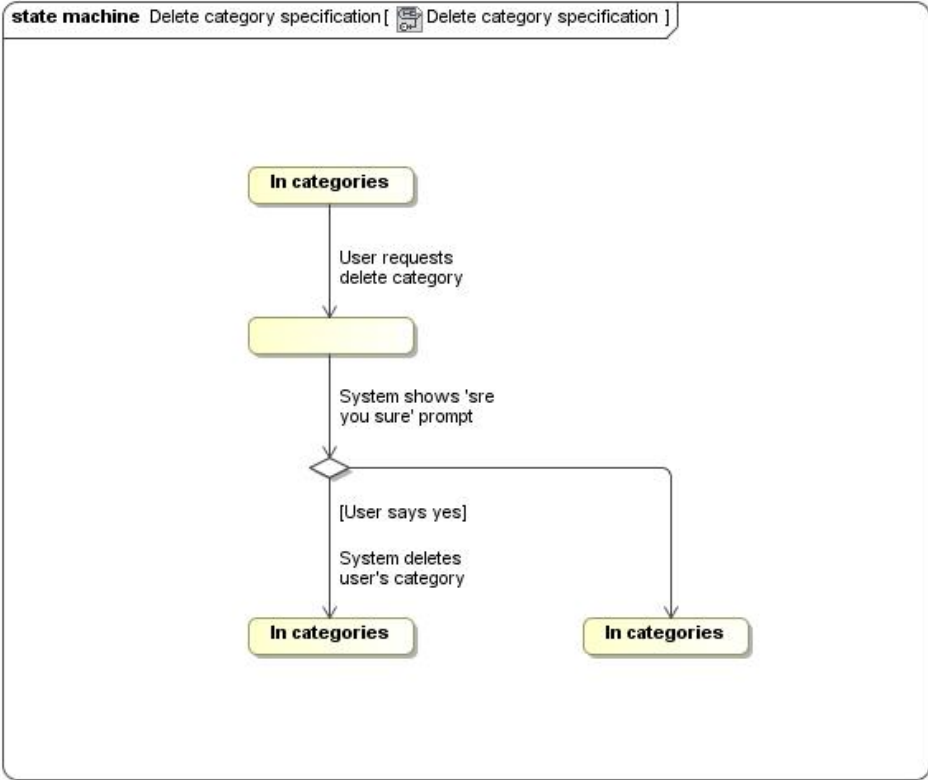


state machine Delete reminder from category specification [Delete reminder from category specification]

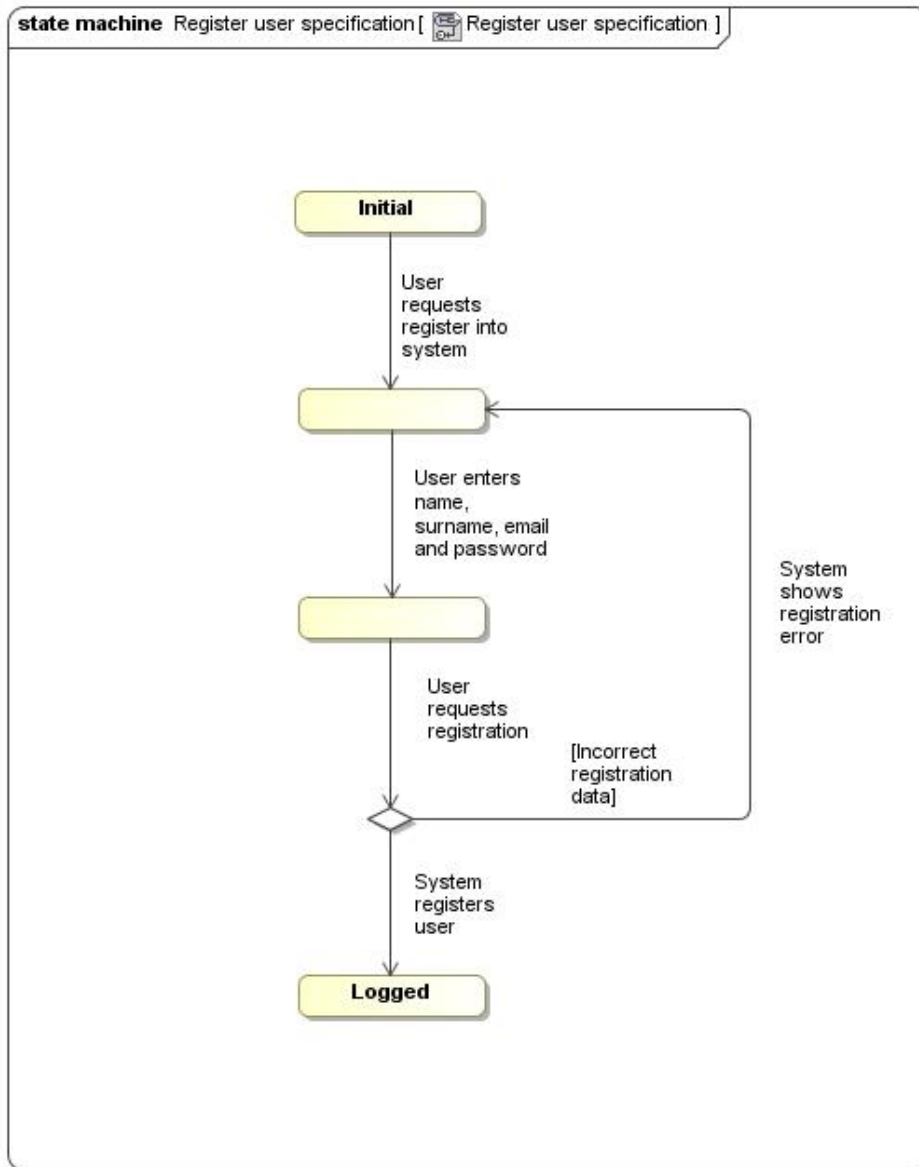


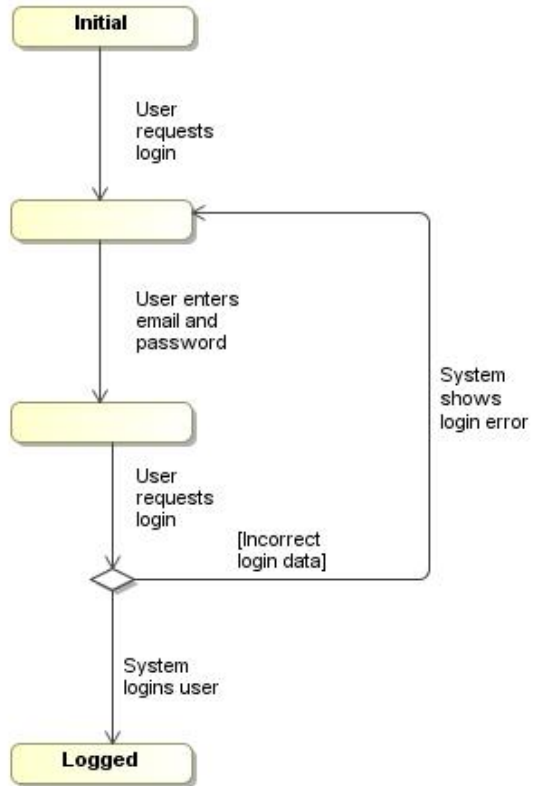
state machine Edit category specification [Edit category specification]





Gestión de usuarios:

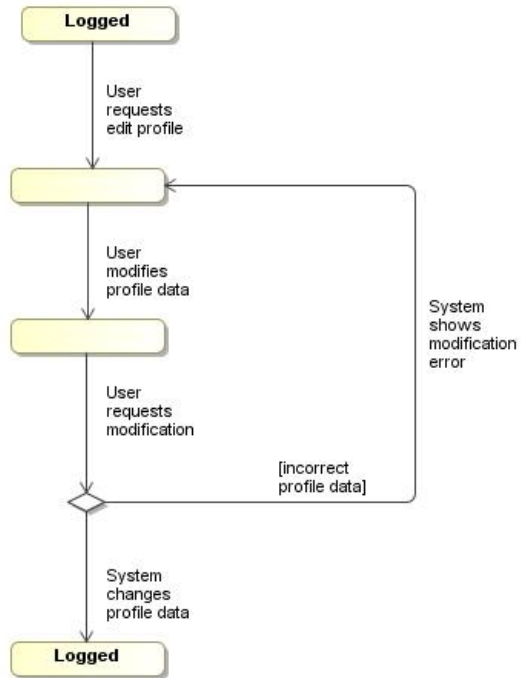


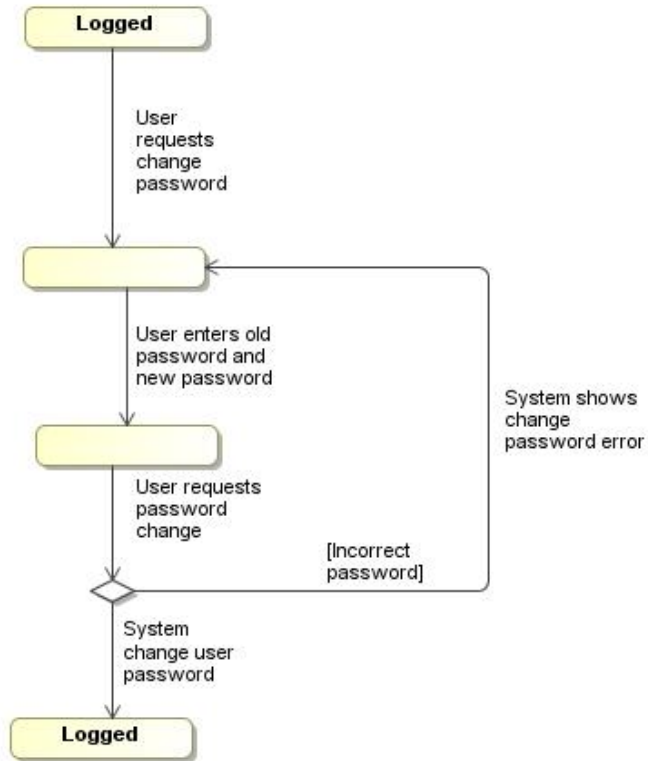


state machine View profile specification [View profile specification]

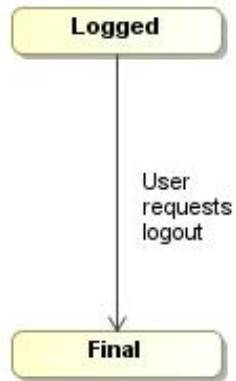


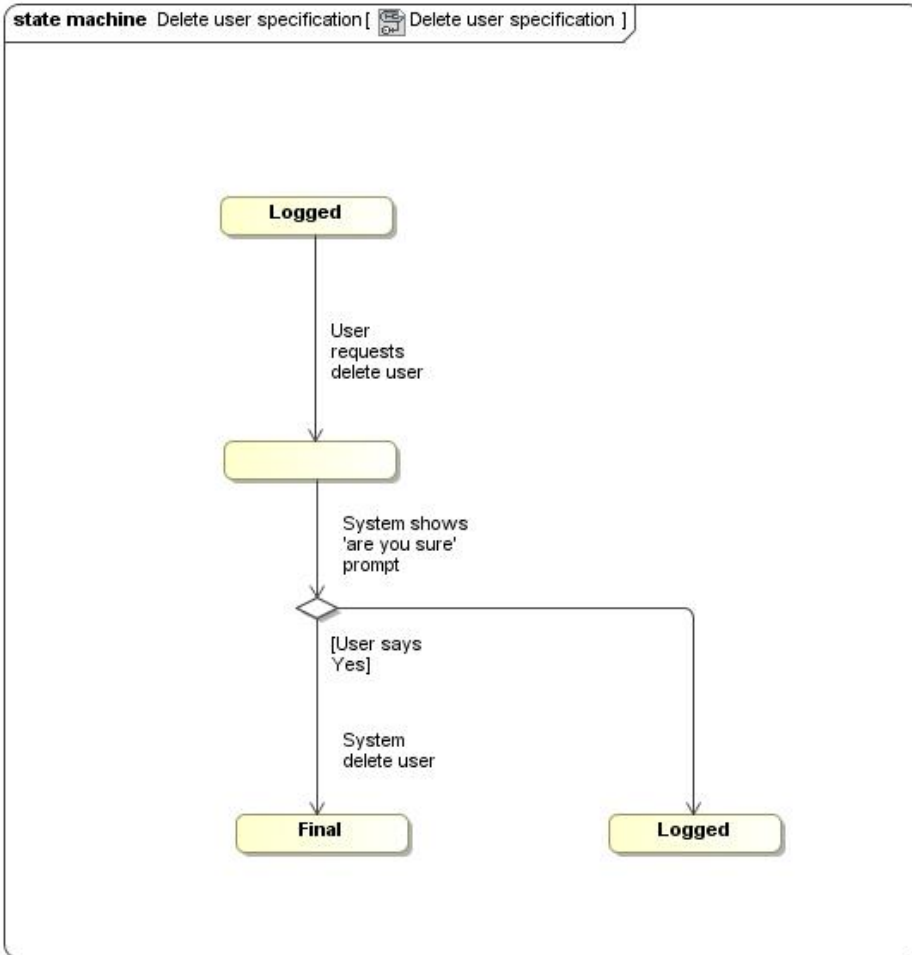
state machine Edit profile specification [Edit profile specification]





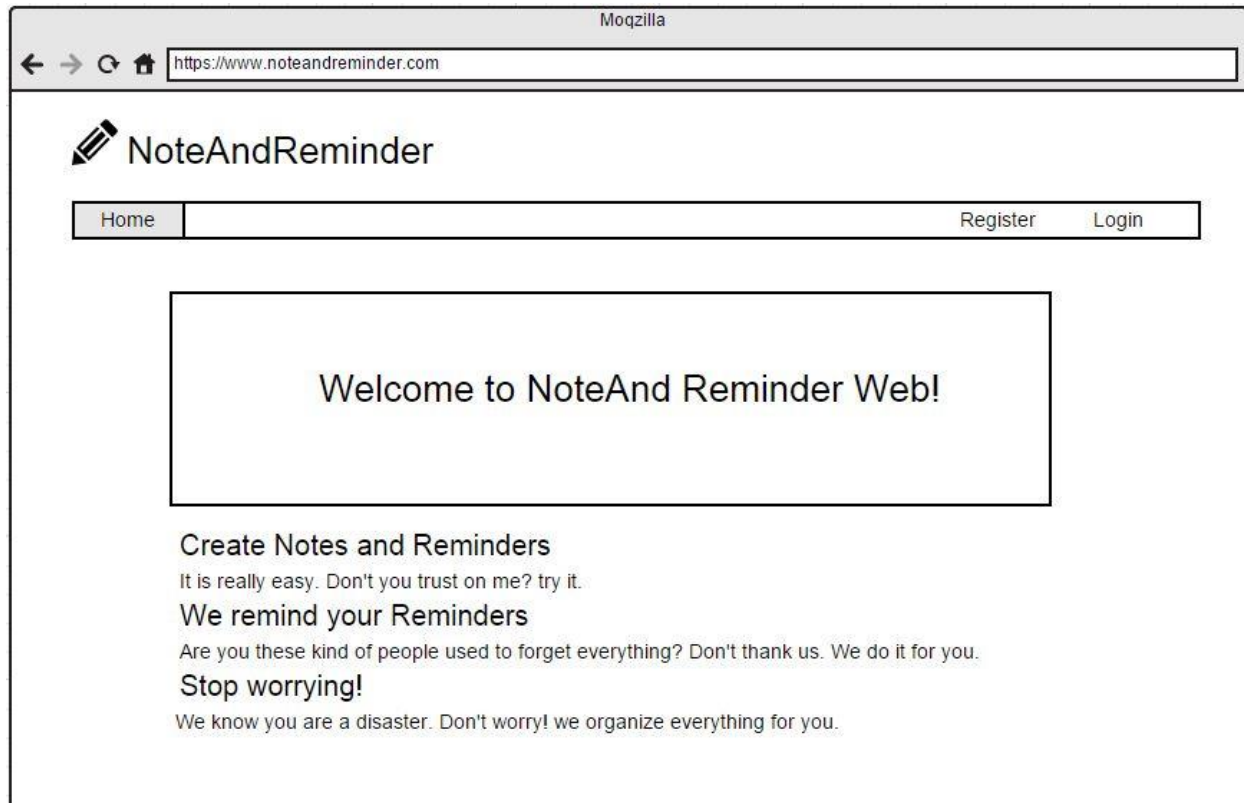
state machine Logout specification [Logout specification]






Mockups

En este apartado se describe los mockups/bocetos realizados con la herramienta web mockups.com. Solo se corresponden con la aplicación web, pues la aplicación móvil es muy sencilla y se ha optado por obviar este paso. Los bocetos están ordenados siguiendo el circuito de operación lógica de un usuario medio de la aplicación:



Moqzilla

← → ↻ <https://www.noteandreminder.com>

 NoteAndReminder

Home Register Login

Register

Name:

Surname:

Email:


Password:

Password (again):

Register Back

Moqzilla

← → ↻ <https://www.noteandreminder.com>

 NoteAndReminder

Home Register Login

Login


Email:

Password:

Login Back

Moqzilla

← → ↻ 🏠 <https://www.noteandreminder.com>

 **NoteAndReminder**

Home | My notes | My reminders | My categories | My profile | Logout

Welcome to NoteAnd Reminder Web!


Create Notes and Reminders
It is really easy. Don't you trust on me? try it.

We remind your Reminders
Are you these kind of people used to forget everything? Don't thank us. We do it for you.

Stop worrying!
We know you are a disaster. Don't worry! we organize everything for you.

Moqzilla

← → ↻ 🏠 <https://www.noteandreminder.com>

 **NoteAndReminder**

Home | My notes | My reminders | My categories | My profile | Logout

My profile [Edit profile](#)

Name:
MyName

Surname:
MySurname


Email:
myemail@email.com

Password:

[Back](#)

Moqzilla

← → ↻ 🏠 <https://www.noteandreminder.com>

 **NoteAndReminder**

Home | **My notes** | My reminders | My categories | My profile | Logout


List of notes

[New note](#)

▼ Title	▼ Creation date	▼ Category
Shopping list	10/04/2015 18:00	Food
Cake recipe	12/04/2015 09:08	Food
Exercises	15/04/2015 11:35	School
Music	15/04/2015 20:24	
Exams	16/04/2015 12:00	School

Moqzilla

← → ↻ 🏠 <https://www.noteandreminder.com>

 **NoteAndReminder**

Home | **My notes** | My reminders | My categories | My profile | Logout


Create a note

Title:

Description:

[Save](#) [Back to list](#)

Moqzilla
 https://www.noteandreminder.com

 NoteAndReminder

Home My notes My reminders My categories My profile Logout

Details of the note


Add category Edit note

Title:
Cake recipe

Description:
 Ingredients: package cake mix, chocolate, 1 cup sour cream, 1 cup vegetable oil, 4 eggs
 1 Preheat oven to 350 degrees F (175 degrees C).
 2 In a large bowl, mix together the cake and pudding mixes, sour cream, oil, beaten eggs and water. Stir in the chocolate chips and pour batter into a well greased 12 cup bundt pan.
 3 Bake for 50 to 55 minutes, or until top is springy to the touch and a wooden toothpick inserted comes out clean. Cool cake thoroughly in pan at least an hour and a half before inverting onto a plate if desired, dust the cake with powdered sugar.

Back to list

Moqzilla
 https://www.noteandreminder.com

 NoteAndReminder

Home My notes My reminders My categories My profile Logout

Edit a note

Delete note


Title:

Description:

Edit Back to details

Moqzilla

← → ↻ 🏠 <https://www.noteandreminder.com>

 NoteAndReminder

Home My notes My reminders My categories My profile Logout


List of reminders

Title: Completion date:

▼ Title	▼ Is completed?	▼ Category		
Mike transfer	true	Bank [X]	edit	delete
Dentist	true	Health [X]	edit	delete
Music	true	Add	edit	delete

Moqzilla

← → ↻ 🏠 <https://www.noteandreminder.com>

 NoteAndReminder

Home My notes My reminders My categories My profile Logout


List of reminders

Title: Completion date:

▼ Title	▼ Is completed?	▼ Category		
Mike transfer	true	Bank [X]	edit	delete
Dentist	true	Health [X]	edit	delete
Creation date: 12/04/2015 11:33, Completion date: 15/04/2015 15:00				
Music	true	Add	edit	delete

Moqzilla

← → ↻ 🏠 <https://www.noteandreminder.com>

 NoteAndReminder

Home My notes My reminders My categories My profile Logout


List of categories

Title:

▼ Title	▼ Has notes?	▼ Has reminders?	▼	▼
Food	true (view)	true (view)	edit	delete
Bank	false (view)	true (view)	edit	delete
Health	false (view)	true (view)	edit	delete
School	true (view)	false (view)	edit	delete

Moqzilla

← → ↻ 🏠 <https://www.noteandreminder.com>

 NoteAndReminder

Home My notes My reminders My categories My profile Logout

List of categories

Title:

▼ Title	▼ Has notes?	▼ Has reminders?	▼	▼
Food	true (view)	true (view)	edit	delete
Notes: Shopping list Cake recipe				
Bank	false (view)	true (view)	edit	delete
Health	false	true		

Moqzilla

https://www.noteandreminder.com

NoteAndReminder

Home My notes My reminders My categories My profile Logout

Details of the note

Add category Edit note

Title: **Cake**

Description: **Shopping list:** Tomatoes 10 u Meat 200 gr Jam 150 gr ...

Ingredients: **Categories:** Food

1 Pre
2 In a
water
3 Ba
come
plate if desired, dust the cake with powdered sugar.

Add Cancel

Back to list

Moqzilla

https://www.noteandreminder.com

NoteAndReminder

Home My notes My reminders My categories My profile Logout

Edit a note

Delete note

Title: **Shop**

Description: **Shopping list:** Tomatoes 10 u Meat 200 gr Jam 150 gr ...

Tomato
Meat
Jam
Chee
Milk
Coffe

Are you sure you want to delete this element?

Yes Cancel

Edit Back to details

Moqzila
https://www.noteandreminder.com

NoteAndReminder

Home My notes My reminders My categories My profile Logout

Details of the note

[Delete category](#) [Edit note](#)

Title: **Cake**

Description: **Shopping list:** Tomatoes 10 u Meat 200 gr Jam 150 gr ...

Categories: Food

Ingredients:

- 1 Pre
- 2 In a water
- 3 Ba come

plate if desired, dust the cake with powdered sugar.

[Delete](#) [Cancel](#)

[Back to list](#)

Moqzila
https://www.noteandreminder.com

NoteAndReminder

Home My notes My reminders My categories My profile Logout

Edit profile

[Change password](#) [Delete profile](#)

Name:

Surname:

Email:

Old password:

New password:

New password:

[Back](#)

Instalación y configuración

Se ha decidido realizar este apartado porque la instalación, creación y configuración de los distintos proyectos no ha sido trivial. Por ello describiremos todas las tareas que se han hecho previamente a poder empezar con el desarrollo de las aplicaciones:

Aplicación servidora:

Instalar de Eclipse for Web Developers.

Comprobar la JDK de Java.

Disponer de Maven integrado en Eclipse.

Disponer de Git integrado en Eclipse.

Instalar Apache Tomcat para ejecutar la aplicación.

Instalar MySQL y crear base de datos y esquema.

Instalar HeidiSQL como cliente de MySQL.

Instalar Google Chrome.

Instalar plugin de Chrome para lanzar peticiones REST a los servicios web.

Crear y configurar el proyecto en Eclipse.

- Crear proyecto basado en arquetipo web de Maven siguiendo el asistente.
- Comprobar correcta estructura de carpetas creada por Maven.
- Definir servidor web Tomcat sobre el que deberá ejecutarse.
- Incluir en el POM de Maven todas las dependencias necesarias, estas son:
 - spring-web
 - spring-web-mvc
 - javaee-api
 - mysql-connector-java
 - eclipselink
 - log4j-api
 - log4j-core
 - junit
 - gson
 - jackson-annotations
 - jackson-databind
 - jjwt

- Crear archivo de configuración de JPA persistence.xml y configurar acceso a MySQL.
- Crear archivo de configuración de dependencias de Spring applicationContext.xml
- Crear los Servlets oportunos y definir los paquetes que estos deben controlar.
- Definir en el web.xml los Servlets definidos y sus rutas así como el Filter.
- Comprobar que todo opera correctamente de manera conjunta.
- Definir en el archivo.gitignore los archivos y carpetas a excluir.
- Hacer primer commit del proyecto en GIT.
- Crear proyecto en Github y asociarlo al proyecto de Eclipse.
- Levantar MySQL y la aplicación en Tomcat y probar todo desde Plugin REST.

Aplicación web:

Instalar Sublime Text 2 como editor de código.

Instalar Git Extensions para control de versiones.

Instalar Cmder para creación y configuración de proyecto.

Crear y configurar el proyecto haciendo uso de Cmder.

- Comprobar correcta instalación de Git
- Instalar Node.js y NPM
- Instalar Yeoman, Bower y Grunt
- Instalar Yeoman Generators
- Definir Paths en las variables del sistema
- Crear carpeta que contendrá el proyecto
- Crear proyecto Usando Asistente de Yeoman
- Durante la creación seleccionar las dependencias:
 - Incluir AngularJS (incluye jQuery)
 - Incluir Módulos por defecto de AngularJS
 - Incluir Bootstrap
 - Excluir Sass
- Instalar Karma y Jasmine.
- Comprobar correcta estructura de carpetas creada por Yeoman.
- Comprobar las clases creadas por defecto.
- Comprobar que todo opera correctamente de manera conjunta.
- Definir en el archivo.gitignore los archivos y carpetas a excluir.
- Hacer primer commit del proyecto en GIT.
- Crear proyecto en Github y asociarlo al proyecto.
- Arrancar Google Chrome en modo que deshabilite SOP.
- Arrancar la aplicación usando Grunt y probar desde Chrome.

Aplicación móvil:

Instalar de Eclipse for Android Developers.

Comprobar la SDK de Android.

Comprobar las herramientas del ADT.

Configurar el emulador de Android.

Instalar la versión 4.4.2 de Android (API 19).

Crear y configurar el proyecto de Eclipse.

- Crear proyecto Android siguiendo el asistente de Eclipse
- Durante la creación:
 - Definir icono de la aplicación.
 - Crear Activity y layout principal de la aplicación.
- Comprobar correcta estructura de carpetas creada por Eclipse.
- Comprobar las clases creadas por defecto.
- Incluir dependencias necesarias:
 - gson
- Comprobar que todo opera correctamente de manera conjunta.
- Definir en el archivo.gitignore los archivos y carpetas a excluir.
- Hacer primer commit del proyecto en GIT.
- Crear proyecto en Github y asociarlo al proyecto.
- Arrancar el emulador y ejecutar la aplicación.

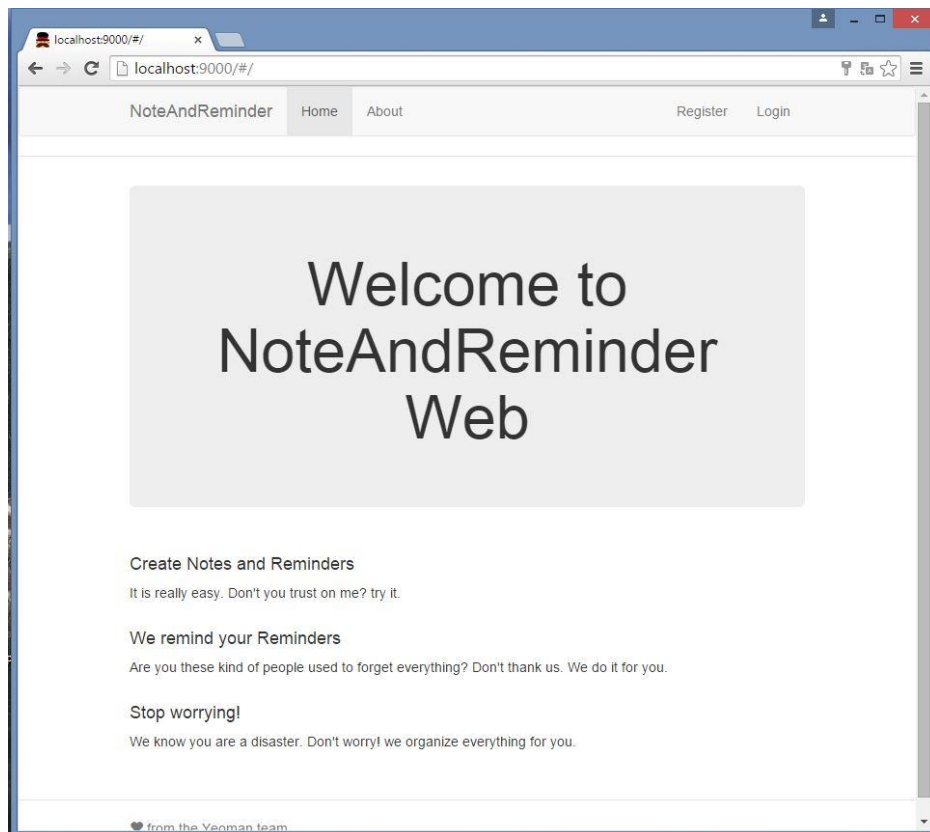
Levantar las aplicaciones:

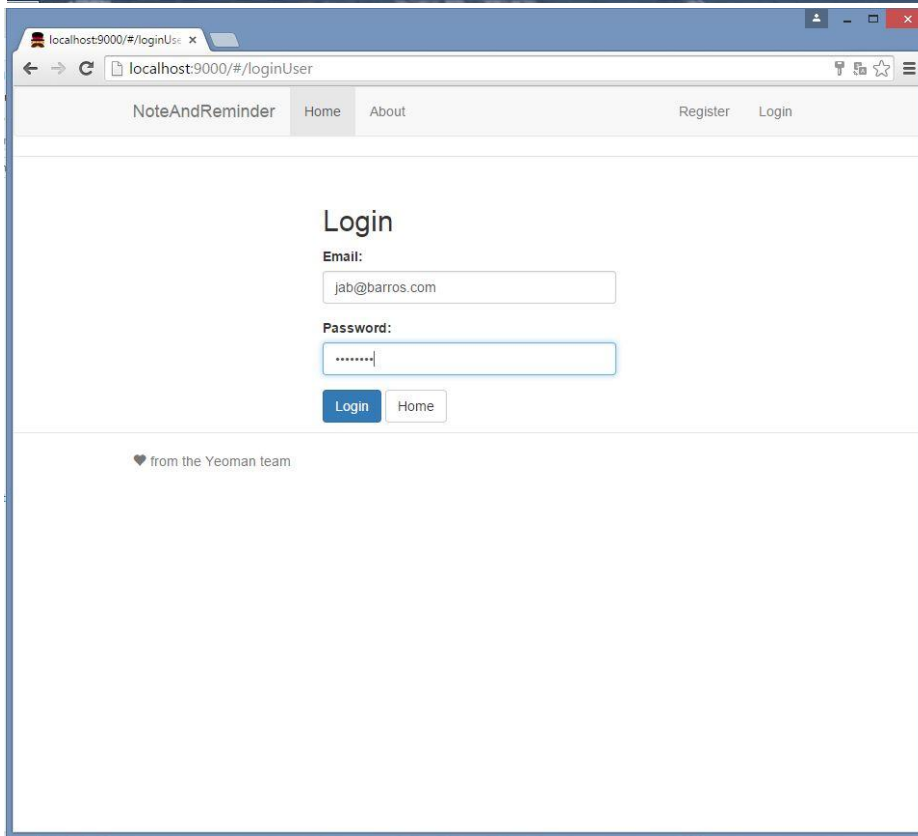
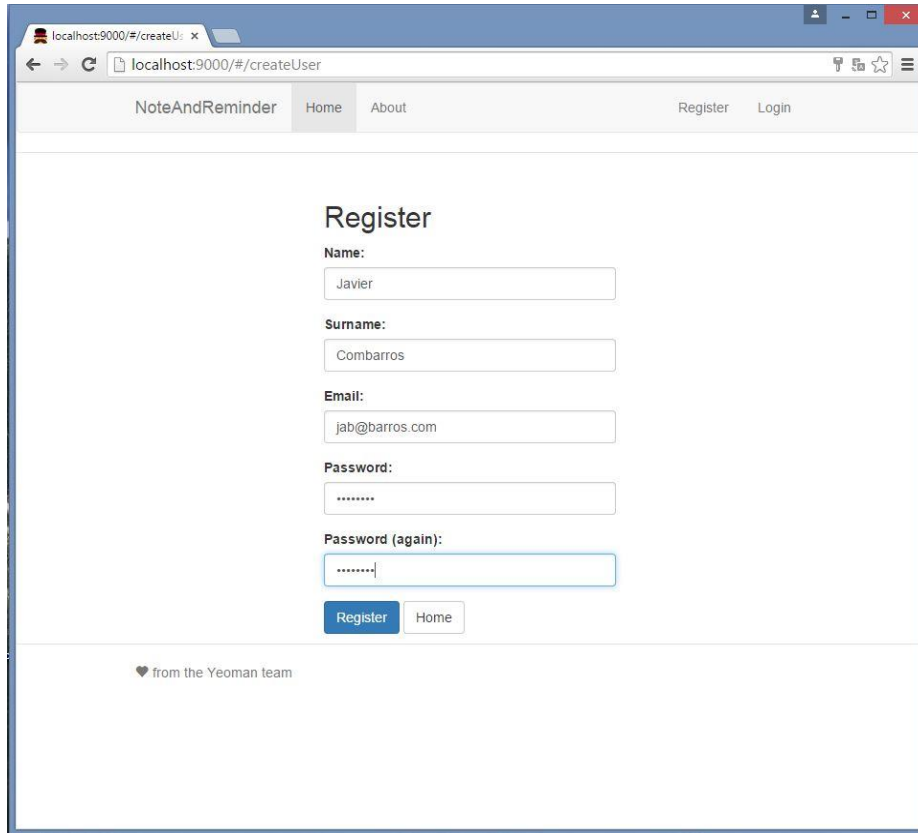
- Arrancar MySQL desde Cmder:
 - *cd C:\Users\Javier\Documents\Master MIW\PFM\mysql-5.6.24\bin && mysqld --console*
- Arrancar servidor Tomcat desplegando la aplicación servidora desde Eclipse
- Arrancar servidor Grunt desplegando la aplicación web desde Cmder
 - *cd C:\Users\Javier\Documents\Master MIW\PFM\Apache Tomcat 8.0.21\webapps\NoteAndReminderWeb && grunt server*
- Google Chrome en modo Disable SOP desde Cmder:
 - *cd C:\Program Files (x86)\Google\Chrome\Application && chrome.exe --user-data-dir="C:/Chrome dev session" --disable-web-security*
- Arrancar Emulador levantando Android y desplegando la aplicación móvil desde Eclipse

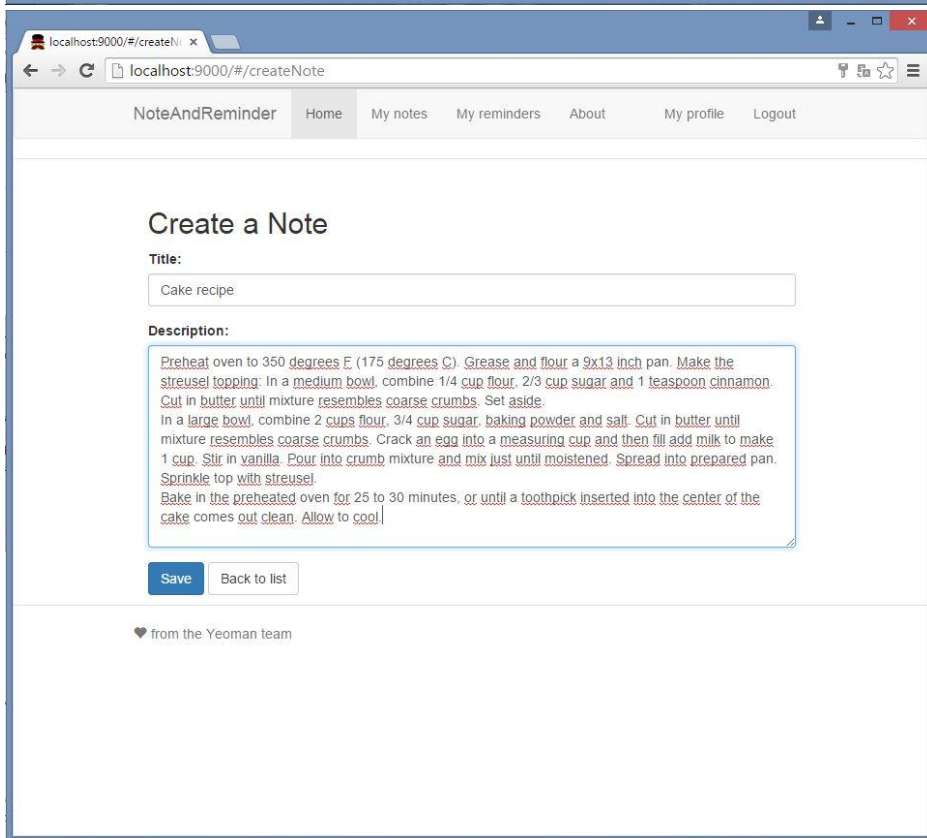
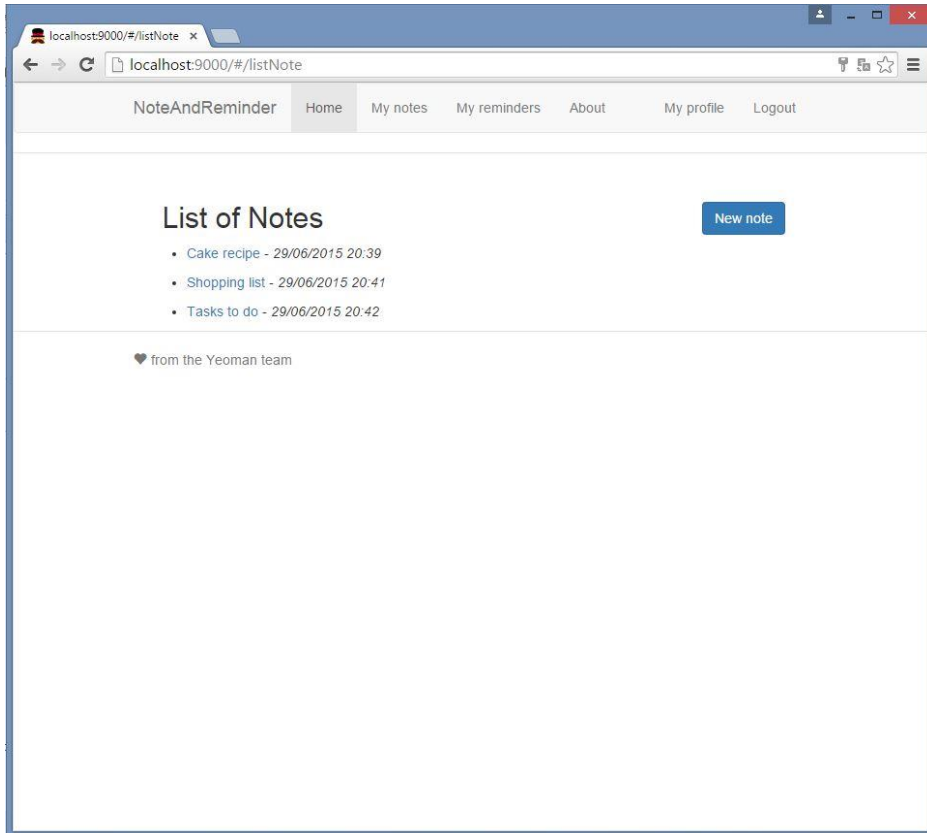
Capturas de pantalla

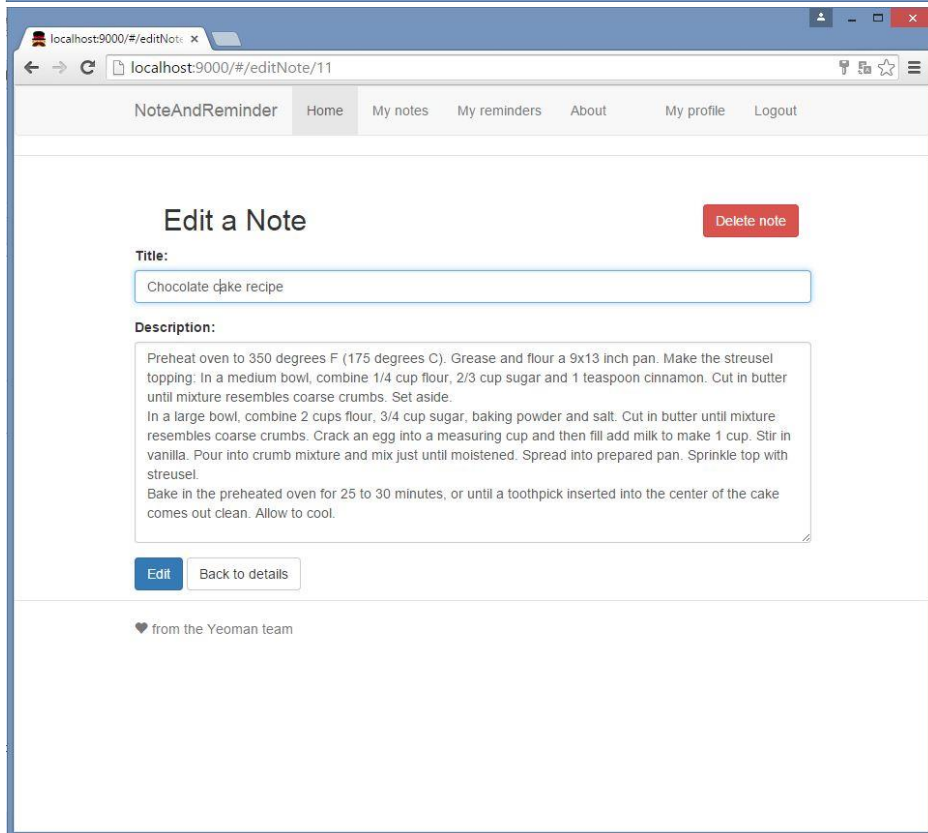
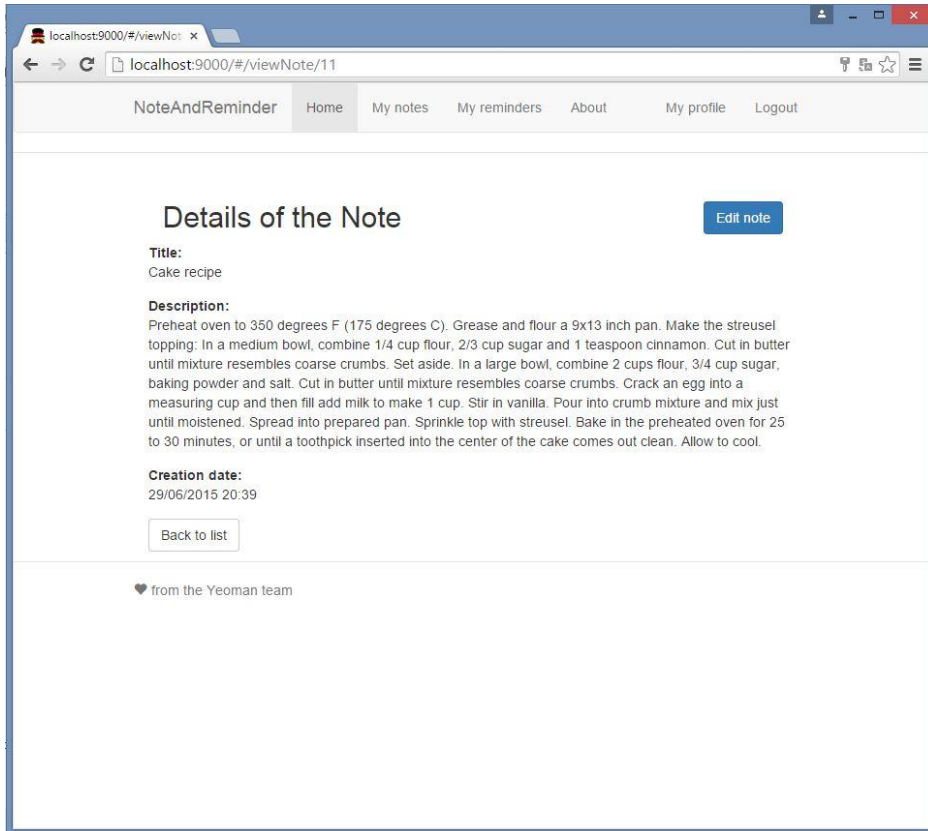
En este apartado mostraremos cómo ha quedado la aplicación finalmente. Veremos un conjunto de capturas de pantalla tanto de la aplicación web como de la aplicación móvil con datos reales. Las imágenes han sido sacadas de las aplicaciones una vez finalizado su desarrollo y tras pasar las pruebas de integración. Como se ha comentado a lo largo del proyecto únicamente se corresponden con la fase 1, esto se ha excluido la gestión de categorías y la aplicación móvil avanzada.

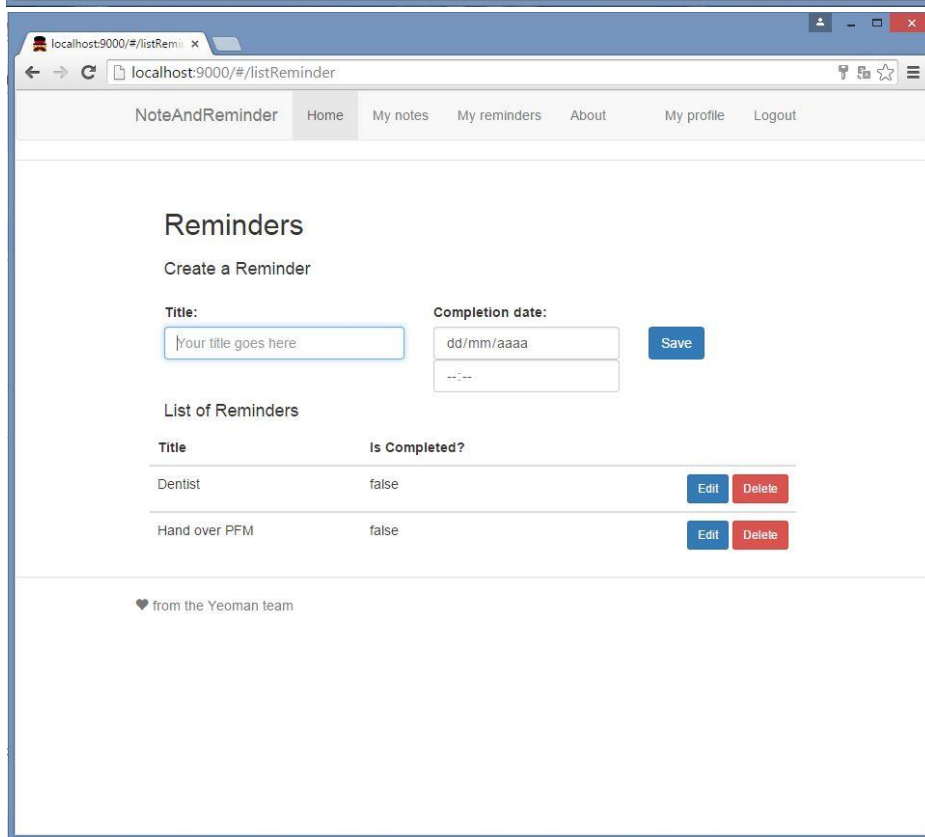
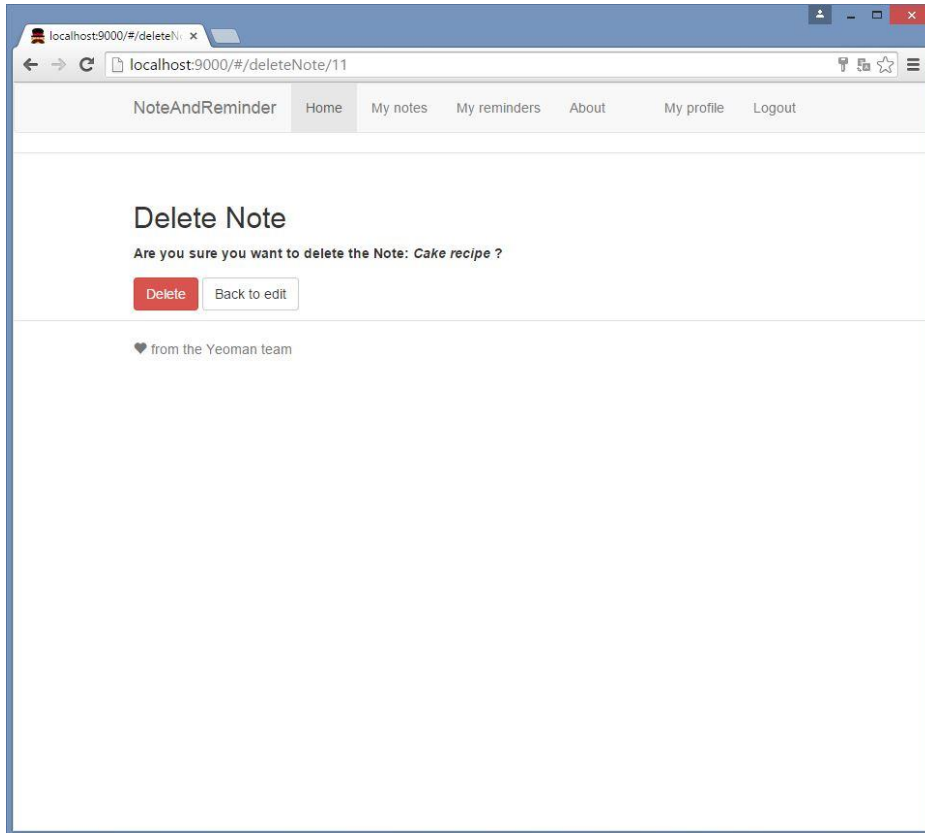
Aplicación web:

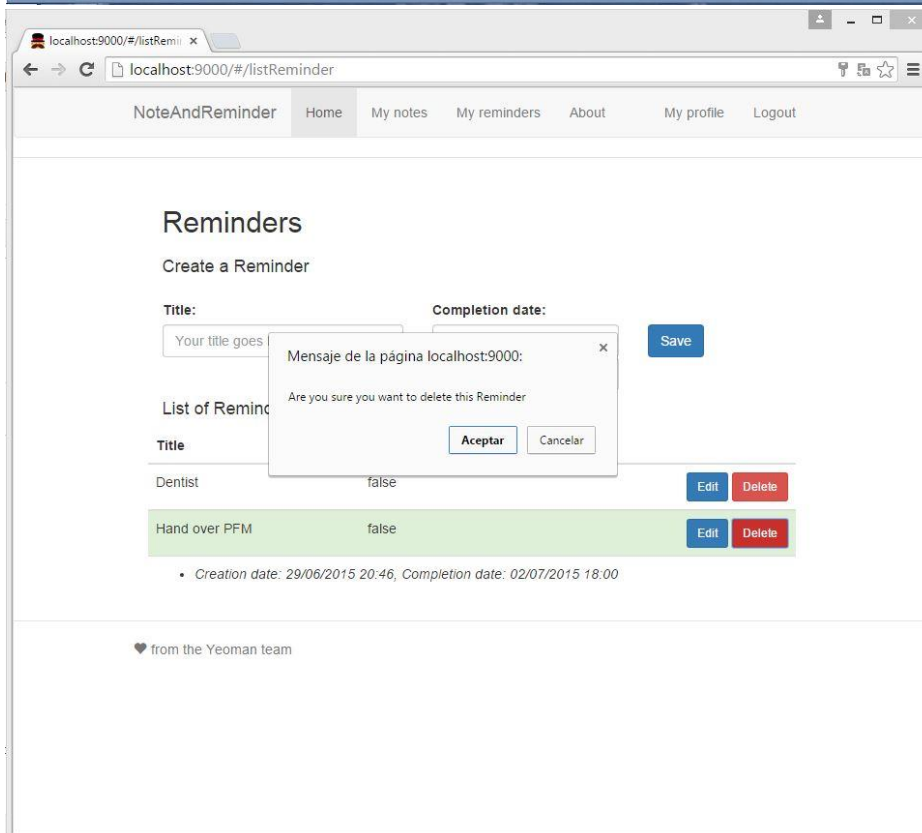
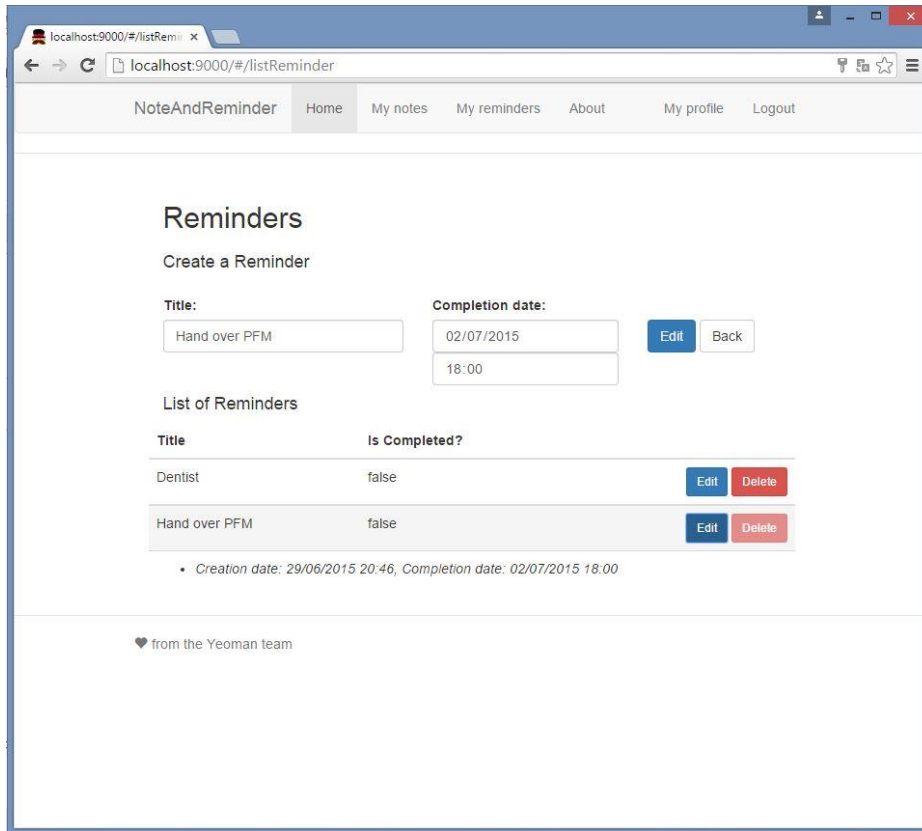


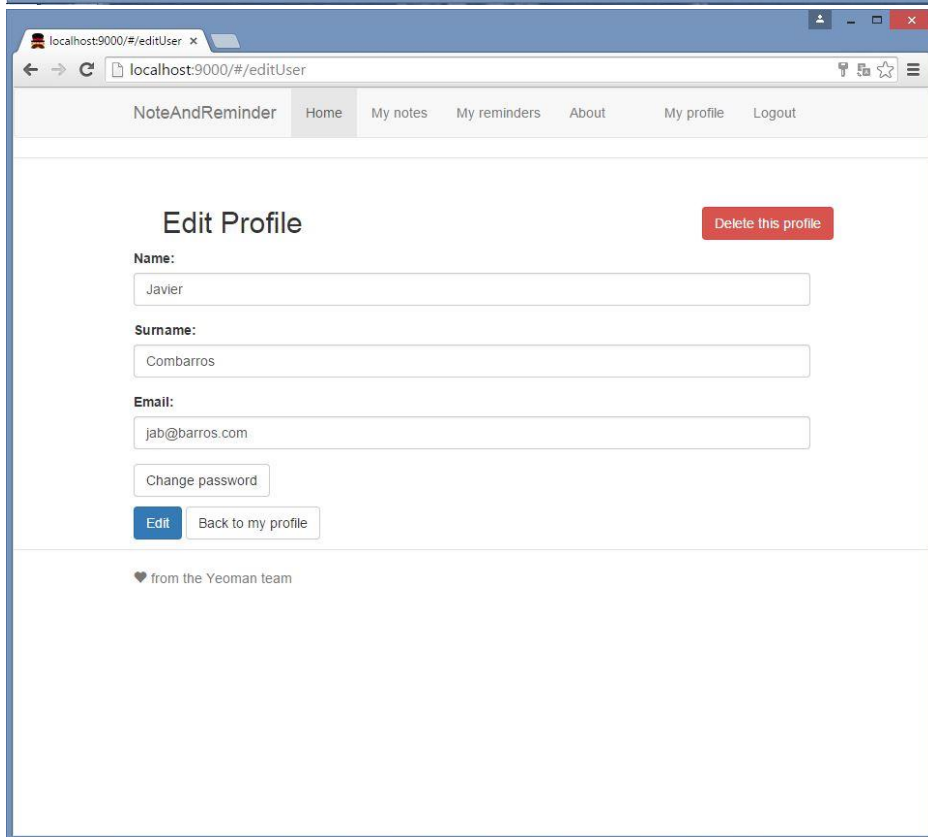
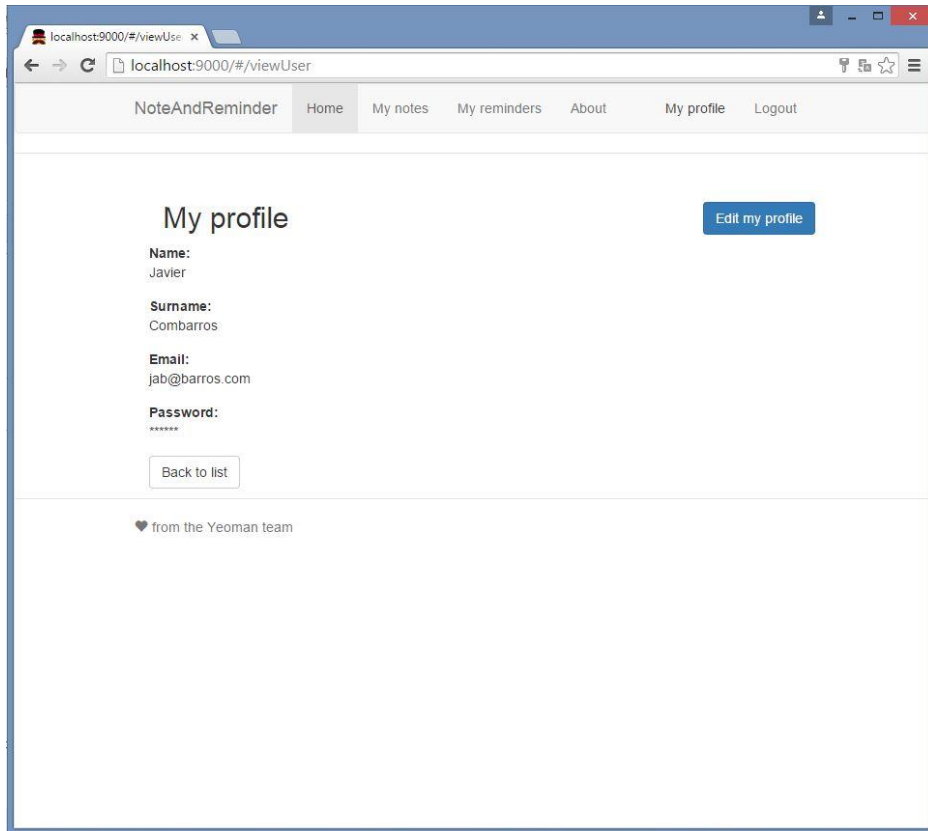


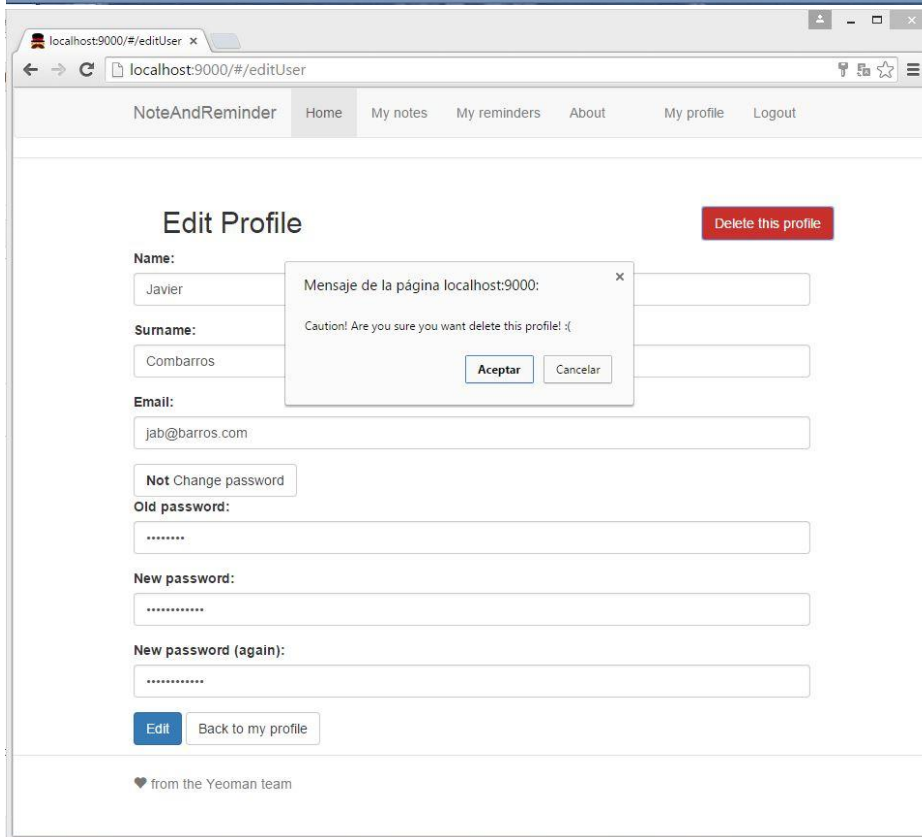
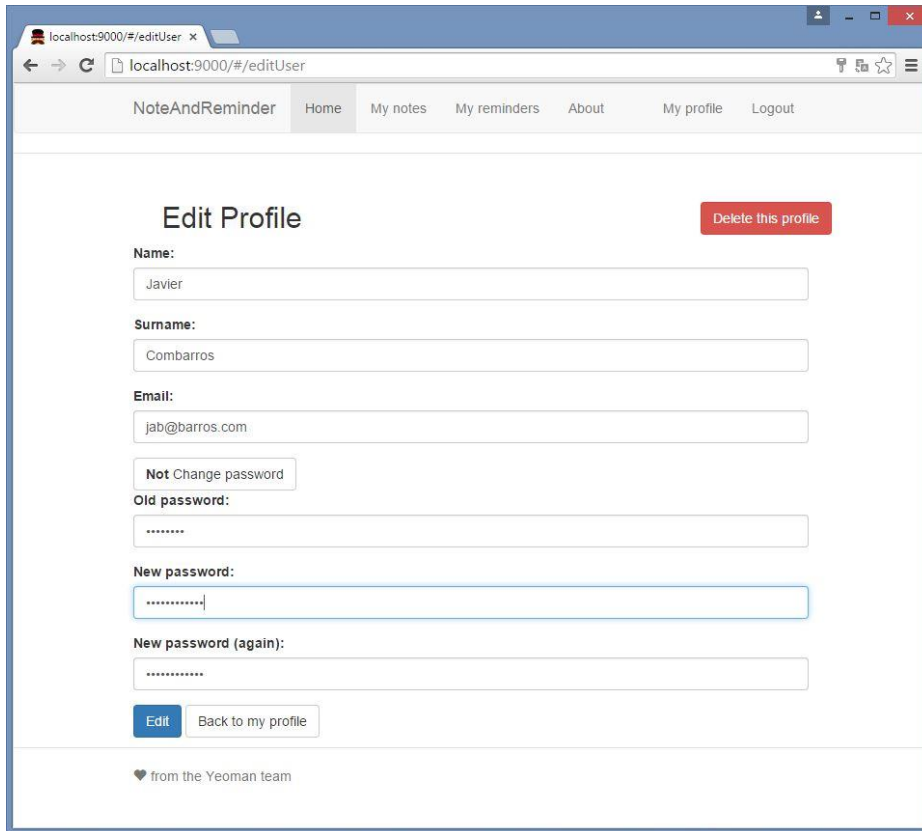












Aplicación móvil:

