UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

DESARROLLO DE UNA RED SOCIAL ORIENATADA A COCINA Y COCTELERÍA

Francisco Méndez López
Tutor: Gonzalo Martínez Muñoz

JULIO 2015

Resumen

En este documento se describen las distintas fases de desarrollo del sistema Cooks: Análisis de requisitos, Diseño, Implementación y Pruebas con el detalle suficiente como para comprender el diseño y arquitecturas elegidos y para realizar pruebas y labores de mantenimiento del software. Este documento cubre el desarrollo de los tres componentes fundamentales del proyecto: el modelo de datos, la lógica de negocio y de acceso a los mismos y la vista o interfaz de usuario.

La aplicación Cooks es una red social orientada a cocina y coctelería. La idea de construir un sistema de este tipo viene motivada por el deseo de crear una aplicación web que permita al usuario disponer de un recetario online en su perfil y consultar recetas de otros usuarios.

El sistema permitirá entre otras cosas crear borradores y publicarlos en el tablón de usuario, seguir la actividad de otros usuarios y realizar búsquedas de recetas publicadas por cualquier usuario.

El documento comenzará introduciendo al lector en el proyecto que se va a realizar. Antes de entrar en las fases de desarrollo del proyecto se realizará un estudio de las distintas tecnologías disponibles para resolver el problema planteado. Tras ello se comenzará con el desarrollo del proyecto, empezando con el análisis detallado de los requisitos software. En la fase de diseño se propone una posible arquitectura a utilizar y se describen sus componentes. Tras haber realizado el diseño se explicarán las tecnologías utilizadas y detalles tenidos en cuenta en la fase de implementación. Por último se explicarán las pruebas realizadas a cada uno de los componentes junto con los resultados obtenidos, finalizando con las conclusiones obtenidas y posibles requisitos futuros.

En la elaboración del documento se ha intentado señalar en la medida de lo posible las distintas opciones sopesadas, las posibilidades disponibles y el razonamiento de la toma de decisiones necesaria en algunos puntos del desarrollo del proyecto.

Palabras clave

Red social, API, Web, cocina, coctelería, MVC, modelo, vista, controlador, receta, recetario, catálogo.

Abstract

In this document will be described all software development phases for Cooks system: Requirements Analysis, Design, Implementation and Testing. Those will be described with enough detail to understand the design and software architecture chosen for this project, as well as making software tests and maintenance jobs. This paper includes the development of all of three-layer architecture selected for this system: data model, controller for accessing data model and user interface.

Cooks application is a cooking and cocktails oriented social network. The idea for making this kind of software is motivated by the inspiration of making a web application where the system dispose to users the possibility of building an online and personal cooking catalog and searching for some recipes from others users.

Cooks will allow to create recipe drafts and publish them into the own user board, following other users activities and doing searches to view some specific recipes.

This document will start introducing the reader to the project with a general description. Before describing development phases a short research will be done to describe current technology context that wraps this project. Then we will start describing software with a thorough requirements catalog analysis. In the design phase a possible architecture will be provided and the description of all of its components. Once we have done all software design tasks we will see the different technologies used to carry out the design chosen and will be provided the most relevant implementation details. The correct functionality of Cooks system has been verified using a variety of tests. The description and outputs of these tests are presented in this document. Finally, the conclusions and possible future lines of work of this work will be presented.

Throughout the document the author has tried to include as far as possible all of the options that have been pondered among all possibilities, as well as the argument for arguing different choices taken along the whole project.

Keywords

Social network, API, Web, cooking, cocktail, cocktail craft, MVC, model, view, controller, recipe, recipe catalog.

Índice de contenidos

Índice de ilustraciones

Índice de tablas

Glosario

AJAX (*Asyncrhonous Javascript And XML)*: tecnología de desarrollo web que permite la realización de peticiones asíncronas desde el lado del cliente para mejorar la interactividad y usabilidad del sistema.

API (*Application Programming Interface)*: constituye un conjunto de funcionalidades que permite a un software abstraerse de ciertas características al utilizar cierta aplicación.

Android: sistema operativo para dispositivos móviles propiedad de Google.

CSS (*Cascading Style Sheets)*: lenguaje utilizado para describir los detalles de los elementos que componen una interfaz web de usuario que permite separar el contenido de un documento de su presentación.

HTML (*HyperText Markup Language*): lenguaje estándar de marcado y etiquetado utilizado para la construcción de páginas web.

Java EE (*Java Enterprise Edition)*: plataforma de programación que utiliza el lenguaje Java y permite crear aplicaciones del lado del servidor.

JDBC (*Java DataBase Connectivity)*: conjunto de funcionalidades que permite a programas escritos en Java conectar con bases de datos.

JPA (*Java Persistance API)*: API que permite a un programa Java hacer persistentes en una base de datos operaciones ejecutadas sobre un diseño orientado a objetos.

JSF (*JavaServer Faces)*: tecnología Java que permite crear interfaces web de usuario utilizando el lenguaje de programación Java.

JSP (*JavaServer Pages)*: plataforma Java que permite crear páginas web dinámicas del lado del servidor.

JSON (*Javascript Object Notation)*: estándar que define una forma de representación de estructuras de datos.

Java: lenguaje de programación orientado a objetos actualmente propiedad de Oracle.

MD5 (*Message-Digest Algorithm 5)*: algoritmo utilizado en criptografía que permite representar un flujo de datos en una cadena de bits de longitud fija normalmente de tamaño reducido.

MVC (*Model-View-Controller)*: patrón de diseño que separa los datos utilizados por un software de su lógica de presentación y de la lógica de control de los mismos.

NoSQL: conjunto de paradigmas que permiten afrontar el diseño de una base de datos de una forma distinta a las tradicionales bases de datos relacionales.

PDO (*PHP Data Objects)*: extensión del lenguaje PHP que proporciona un conjunto de funcionalidades de acceso a una base de datos.

PHP (*PHP Hypertext Pre-processor)*: lenguaje orientado al desarrollo de páginas web dinámicas del lado del servidor que permite aplicar distintos paradigmas de programación.

RDBMS (*Relational DataBase Management System)*: sistema que se encarga de gestionar una base de datos relacional que permite al usuario la definición, manipulación, análisis y borrado de los datos, entre otras operaciones.

REST (*Representational State Transer*): arquitectura que permite definir una interfaz propia entre sistemas distribuidos sobre HTTP orientado a la existencia de recursos.

RMI (*Remote Method Invocation*): tecnología similar a RPC que permite la comunicación con un sistema remoto y ejecutar operaciones en éste utilizando el paradigma de orientación a objetos.

RPC (*Remote Procedure Calls*): arquitectura que permite la ejecución de procedimientos remotos en escenarios con arquitecturas cliente-servidor.

XML (*eXtensible Markup Language*): lenguaje de marcado y etiquetado utilizado para almacenar datos y estructuras de datos en un formato estándar.

Cliente-servidor: modelo de arquitectura que separa las tareas de solicitud de recursos de las relacionadas con el proceso de proveer dichos recursos.

Servlet: instancia de un programa escrito en Java que actúa como servidor en una comunicación entre dos aplicaciones.

1. Introducción

1.1. Motivación

En la actualidad está muy extendido el uso de las redes sociales entre un público muy amplio y compuesto por personas de todas las edades. Ocasionalmente las redes sociales están orientadas a determinados subconjuntos de la sociedad, variando entre géneros musicales, deportes o cualquier otro estilo de vida. En este trabajo se pretende construir una red social que atraiga a conjuntos de aficionados y profesionales de la cocina y la coctelería. La cocina y la coctelería constituyen actividades en las que muchas personas se ven implicadas diariamente, ya sea por dedicación profesional o bien por afición.

La cocina ha experimentado un boom en los últimos años. Este auge ha sido seguido y fomentado en parte por los medios de comunicación [1]. En paralelo la coctelería es una actividad que cada vez está más en auge y es practicada en todo tipo de eventos desde un enfoque tanto profesional como amateur. Además en ambos casos, con la difusión y la mayor facilidad de acceso a la Web, y junto con la gran cantidad de información disponible, Internet se convierte en un medio de consulta muy utilizado para visualizar recetas.

Sin embargo los medios de consulta existentes no suelen ofrecer al internauta la posibilidad de crear su propio recetario online, por lo que resultaría interesante que una aplicación diera esta posibilidad y que pudiera optimizar las consultas de recetas disponibles en los recetarios de todos los usuarios de la aplicación.

En este trabajo se propone realizar una red social de cocina y coctelería, que denominaremos Cooks, y que constituirá una oportunidad para que todos puedan crear su propio recetario online y compartir sus creaciones y tendencias gastronómicas en el mundo de la cocina y de la coctelería, así como ofrecer la posibilidad a los usuarios de que puedan incorporar recetas de cualquier estilo. La red social se va a enfocar para que sea de tipo abierta –*follow*-, lo que posibilitará que cualquier usuario sea capaz de realizar un seguimiento de la actividad de otro usuario sin el consentimiento de éste último.

1.2. Objetivos

El objetivo principal de este proyecto es crear una red social de cocina y coctelería con las siguientes características:

- Permitir al usuario crear un recetario personal.
- Servir de repositorio de recetas.
- Búsqueda filtrada de recetas.
- Permitir a un usuario seguir la actividad de otros usuarios mediante una relación de seguimiento.
- Interacción de los usuarios con las recetas mediante la valoración de las mismas y la agregación de comentarios.

Este trabajo cubre el desarrollo tanto de la base de datos del software, como de la API de acceso a los datos y la propia interfaz de usuario.

1.3. Visión general del documento

A lo largo de este documento se describirán las diferentes fases de desarrollo del proyecto. La memoria está compuesta de 8 apartados y 2 anexos cuyas descripciones se pueden resumir así:

➔ **Apartado 1: Introducción**

Exposición de los motivos que han llevado a la realización del proyecto, los objetivos y beneficios que se esperan alcanzar con él y una breve perspectiva general del documento.

➔ **Apartado 2: Estado del arte**

Descripción de las distintas tecnologías investigadas, valoradas y estudiadas como posible solución al problema que subyace al proyecto, encuadradas en el marco tecnológico que envuelve al sistema.

➔ **Apartado 3: Análisis de requisitos**

Esta es la primera fase de desarrollo del proyecto: el análisis. Se proporcionará una descripción general del proyecto, funcionalidades y objetivos esperados. Asimismo se hace hincapié en el catálogo de requisitos del software y la indicación de algunos casos de uso de ejemplo.

➔ **Apartado 4: Diseño**

Fase de diseño, donde se dará respuesta a los requerimientos mencionados en la sección 3 y se expondrá la solución de diseño tomada, la arquitectura software adoptada y una descripción de los componentes que la forman.

➔ **Apartado 5: Implementación**

Detalle sobre las distintas soluciones técnicas utilizadas que responden a las directrices ya establecidas en el apartado 4. Para ello se comenzará describiendo las plataformas y tecnologías empleadas, indicación de los paquetes y librerías utilizados finalizando con el detalle de la implementación de cada componente de la arquitectura software.

➔ **Apartado 6: Pruebas y resultados**

Diseño previo a la ejecución de las pruebas a realizar a cada uno de los componentes del sistema, con el fin de verificar que cumplen los requisitos establecidos. Tras ello se proporcionará detalle de los resultados obtenidos.

➔ **Apartado 7: Conclusiones y trabajo futuro**

Planteamiento general del trabajo realizado, utilidad de la experiencia en el proceso formativo académico, dificultades y retos encontrados y reflexión sobre el estado del sistema final junto con posibles mejoras futuras.

➔ **Referencias**

Bibliografía de referencia.

➔ **Anexo A: Descripción de los casos de uso**

Análisis de algunos casos de uso de ejemplo.

➔ **Anexo B: Descripción de la API**

Descripción detallada de las funciones que componen la API del sistema.

2. Estado del arte

2.1. Introducción

En este apartado será expuesto el contexto tecnológico que rodea al proyecto realizado, analizando las distintas posibilidades valoradas para llevar a cabo las tareas implicadas.

2.2. Servicio de red social

Actualmente existen numerosos y variados tipos de redes sociales. A menudo se distingue entre redes sociales horizontales y verticales [2]. Las redes sociales horizontales se caracterizan por no incluir ninguna temática en particular y están dirigidas a cualquier tipo de usuario con participación libre en la aplicación. Facebook y Twitter son ejemplos de redes sociales de este tipo. Por otro lado tenemos redes sociales verticales, que establecen una temática particular y están dirigidas a usuarios con intereses o aficiones comunes. En este segundo tipo es donde hay mayor variedad de aplicaciones, teniendo desde redes sociales de deportistas[1] hasta redes sociales para profesionales como Linked In.

Aunque el sistema a construir es una red social, existen otras perspectivas que nos permiten afrontar el problema inicial, que no es otro que el de facilitar e implantar una aplicación en la red que albergue recetas de cocina y coctelería.

La primera alternativa a la red social podrían ser los foros. Un foro de Internet se caracteriza por facilitar un entorno para la discusión de diversos temas, donde conceptualmente el peso de la utilidad de la aplicación recae en el entorno que gira alrededor de cada tópico. Los foros típicamente son sitios de consulta, donde los usuarios tan solo pueden crear temas y comentarlos, y donde éstos tienen un rol poco relevante en el objetivo principal de este tipo de aplicación.

La siguiente opción podría ser un simple sistema de compartición de recetas online, donde no existirían relaciones entre los usuarios, por lo que no se podría realizar un seguimiento de la actividad de uno en particular de una manera cómoda y directa.

Sin embargo, si a este último modelo le añadimos las relaciones entre los usuarios, con el fin de que el sistema les muestre la actividad de algunos en particular por una preferencia motivada por intereses comunes en las profesiones abarcadas. De esta manera el usuario podrá obtener un filtrado del contenido publicado de forma directa y dirigida hacia sus intereses.

El usuario entonces pasará a tener un rol más importante y podrá utilizar la aplicación tanto para publicar contenido y crear así su propio catálogo personal de recetas como para realizar consultas.

[1] http://amatteur.com/

Además la importancia que adquiere el rol del usuario abre un abanico de posibilidades, por ejemplo las relacionadas con la creación de un cierto grado de competitividad entre ellos.

Sin embargo queda otra cuestión por resolver y es de qué manera orientar la red social. En términos de teoría de grafos, podríamos distinguir entre redes sociales dirigidas y no dirigidas. Desde la primera perspectiva en una relación entre dos usuarios deben proporcionar los dos su confirmación de amistad. En cambio en el segundo tipo no es necesario el consentimiento de ambos, tan solo una simple declaración de seguimiento por parte del usuario interesado.

Esto genera beneficios en un sistema de este tipo frente a las redes no dirigidas, por ejemplo la posibilidad de visualizar el contenido publicado por un usuario sin su aceptación, por lo que no es necesaria la existencia de relaciones bidireccionales de seguimiento.

2.3. Base de datos

Para el almacén de datos en principio tenemos dos opciones posibles: RDBMS (*Relational Database Management System*, Sistema Gestor de Bases de Datos) y NoSQL.

En primer lugar tenemos NoSQL, que resulta ser un nuevo paradigma para afrontar el almacenamiento de datos que difiere en ciertos aspectos de las clásicas bases de datos relacionales. Este tipo de tecnología permite incrementar el rendimiento total del sistema escalando horizontalmente y agilizando las operaciones con memoria que requieren tratamiento de grandes volúmenes de datos, aunque a costa de un aumento en el riesgo en la consistencia e integridad de los datos [3].

Por otro lado tenemos las bases de datos relacionales, que aunque implican funciones más costosas, aseguran operaciones y datos estructurados e integridad de los datos en todas sus operaciones [4].

Sin embargo las bases de datos relacionales otorgan un beneficio fundamental, que está relacionado la madurez de éstas sobre NoSQL en el ámbito de la gestión de datos, y en el ámbito personal al tener experiencia sobre RDBMS. NoSQL es una tecnología joven en la actualidad en fase experimental y las distintas implementaciones no son aún estándares, por lo que migrar el almacén de datos puede resultar una tarea complicada, donde las bases de datos relacionales no contribuyen a esta situación. NoSQL está orientado a grandes volúmenes de datos y a un tratamiento específico de datos que lo requiera [5].

Además utilizar esta tecnología requiere un estudio previo de la misma y de la implementación elegida, así como la librería de que disponga el aplicativo que vaya a utilizarla y que conlleva al exceso de tiempo en el trabajo actual.

2.4. Interfaz de Programación de Aplicaciones

Para la implementación de la funcionalidad del servidor existen variadas tecnologías disponibles:

En primer lugar tenemos la plataforma Java, que en Java EE (*Java Enterprise Edition*) ofrece los *servlets*, que constituyen instancias de servidores cuyo único objetivo sería en este caso interaccionar con el modelo de datos mediante la API JDBC, que junto con la tecnología JPA permite realizar un sofisticado acceso y manipulación de la base de datos

La otra posibilidad valorada ha sido la utilización de PHP como lenguaje de servidor, que tiene generación directa de código HTML y es un lenguaje multiparadigma[2], por lo que no obliga a usar la orientación a objetos y tiene leves pinceladas de programación funcional que resultan interesantes.

2.5. Interfaz gráfica de usuario

Para la interfaz gráfica de usuario de nuevo se pueden sopesar los beneficios aportados por Java. En este caso podemos incorporar el uso de páginas dinámicas de servidor JSP (*JavaServer Pages*) junto con el *framework* JSF (*JavaServer Faces*), que ofrecen la posibilidad de crear páginas HTML mediante código Java, con etiquetas personalizadas y reutilizando plantillas.

La siguiente opción tenida en cuenta ha sido una aplicación para dispositivos móviles con la plataforma Android, que junto con una interfaz web puede extender el radio de alcance de la aplicación.

La siguiente opción a tener en cuenta es PHP, que permite generar de manera dinámica y directa código HTML mediante un lenguaje multiparadigma y ampliamente utilizado en el desarrollo web.

2.6. Middleware

En [6] se nos proporciona la siguiente definición para Middleware:

"Middleware […] es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, o paquetes de programas, redes, hardware y/o sistemas operativos."

En nuestro sistema el Middleware se incorporará para facilitar la comunicación entre la API y la interfaz gráfica de usuario. En la actualidad existen variados estándares y protocolos que nos permiten resolver este problema de distintas maneras adoptando arquitecturas diferentes:

[2] Lenguaje multiparadigma: lenguaje de propósito general que abarca diferentes paradigmas de programación.

- **REST**: la arquitectura REST (*Representational State Transfer*) sugiere una solución a adoptar para los sistemas web distribuidos que utilicen el protocolo HTTP, donde la importancia recae sobre la representación de las estructuras de datos frente a otras soluciones [7].

- **RPC**: con la tecnología RPC (*Remote Procedure Calls*) podemos permitirnos implantar middleware orientado a la ejecución de procedimientos remotos utilizando HTTP y otros estándares, como XML o JSON, encapsulados en este protocolo, mediante la existencia de un *client stub* y un *server skeleton* que posibilitan la ejecución y comunicación punto a punto entre el cliente y el servidor [8].

- **RMI**: otra posibilidad es la inclusión de RMI (*Remote Method Invocation*), que de manera conceptualmente similar a RPC, permite ejecutar procedimientos remotos con un paradigma subyacente orientado a objetos, por lo que permite la ejecución de instanciar objetos y cambiar su estado desde el propio cliente [9].

3. Análisis de requisitos

3.1. Descripción general

3.1.1. Propósito del sistema

La aplicación Cooks constituirá una red social orientada a cocina y coctelería que servirá a los usuarios para crear un catálogo personal de recetas, consultable por cualquier otro usuario y que forme parte de un recetario universal en la red. Para ello los usuarios serán capaces de crear recetas y compartirlas a través de su perfil con el resto de usuarios. Para permitir estas características, la red social Cooks será de tipo seguimiento –*follow*- donde los usuarios se conectan a través de una relación de "seguimiento" unidireccional que permite que un usuario pueda declararse como seguidor de otros usuarios siempre sin necesidad del consentimiento de estos últimos. Esta relación de seguimiento permite al usuario seguidor obtener una realimentación de la actividad del usuario al que sigue.

El sistema por tanto albergará todas las recetas publicadas por los usuarios, por lo que también servirá como sitio de consultas de recetas de cocina y coctelería.

3.1.2. Ámbito del sistema

El sistema solo contemplará la creación de recetas de cocina y coctelería y una búsqueda filtrada de recetas y de usuarios, por lo que queda fuera del alcance de la aplicación cualquier otro ámbito relacionado con la gastronomía más allá del entorno de la cocina.

Desde una primera perspectiva general sobre nuestro sistema, se pueden describir todas aquellas acciones que los usuarios podrán llevar a cabo de la siguiente manera:

- Realizar un seguimiento de la actividad de otros usuarios.
- Crear recetas y compartirlas en el tablón personal de usuario.
- Valoración de recetas, junto con la posibilidad de comentarlas y añadir sugerencias.
- Búsqueda y filtrado de recetas y usuarios.
- Disponer de una lista de borradores de usuario, que constituirán recetas que aún no estarán listas para ser publicadas.

Los beneficios obtenidos son numerosos, entre los que se pueden señalar los siguientes:

- Conectar a los usuarios a través de una estructura de red social, no solo para favorecer la creación de lazos de amistad sino para que cada usuario pueda recibir una retroalimentación del sistema, mostrando contenidos de interés publicado por otros usuarios.

- Seguimiento de la actividad de los usuarios, lo que suscita una retroalimentación por parte de la aplicación al usuario con el fin de informarle del contenido publicado por sus "amigos".

- Almacén de consulta, donde los usuarios podrán realizar búsquedas sofisticadas dirigidas a recetas específicas.

- Diversidad y novedad en la cultura gastronómica. Un sistema de este tipo favorecerá a la importación y exportación de conocimientos muy diversos y variados en el ámbito de la cocina, algo relevante que está presente en cualquier cultura del mundo. Por otro lado también contribuirá a que las personas prueben con nuevas experiencias en su entorno gastronómico particular.

- Posibilidad de darse a conocer a cualquier aficionado o experto en los terrenos de la cocina y la coctelería, lo que puede tener como consecuencias el alcance de ciertos objetivos personales a nivel de usuario.

- Visualizar contenido publicado y compartido por profesionales, tanto a nivel personal como a nivel de empresa u organización.

- Oportunidad de crear un recetario de cocina en formato digital, alojado en Internet y consultable mediante el acceso a la Web.

- Objeto de investigación. El sistema puede utilizar y contener datos que sean objeto de investigación, por ejemplo los relacionados con los gustos de un usuario implícitos tanto en las conexiones del grafo que define la propia red social como en sus propias valoraciones, o las recomendaciones que realizará el sistema centradas en los gustos y la actividad de cada usuario, entre otros.

3.2. Descomposición en subsistemas

La funcionalidad total de la aplicación ha sido descompuesta en 2 subsistemas distintos atendiendo a los distintos conjuntos funcionales que se han diferenciado. Los dos subsistemas aquí expuestos no resultan ser independientes entre sí, sin embargo nos permiten agrupar funcionalidades que tienen objetivos comunes, a saber: Usuario y Publicación. A continuación se describirán en rasgos generales cada uno de estos subsistemas:

Usuario

Este primer subsistema contiene toda aquella funcionalidad relacionada directamente con el usuario del sistema, y agrupará aquellos requisitos que definan restricciones que afectan de manera directa al propio usuario, tales como los distintos apartados de que dispondrá en la aplicación. Este primer conjunto de requerimientos reunirá todos aquellos aspectos generales relativos a la información del perfil del usuario en el sistema, así como todas aquellas acciones que el usuario podrá llevar a cabo que queden más allá de la publicación de contenidos, como puede ser la acción de seguir o dejar de seguir a otro usuario, registro en la aplicación, obtener la información del perfil de otros usuarios o la búsqueda de usuarios y recetas.

Publicación

Una publicación consistirá en la compartición de una receta con el resto de usuarios. En el subsistema Publicación se pretenden asociar todas aquellas operaciones que podrá realizar el usuario relacionadas con la publicación de contenidos. En este subsistema residirá una gran parte de la funcionalidad total del sistema mayor, por lo que podemos deducir que será el más grande y complejo de los dos expuestos.

En términos generales este subsistema definirá los componentes de una publicación y el proceso de creación de la misma, e incluirá funciones para publicar recetas, crear borradores o cualquier forma de interacción con una publicación por parte de un usuario, como por ejemplo la valoración de la misma o la agregación de comentarios.

3.3. Catálogo de requisitos

3.3.1. Requisitos funcionales

En esta sección se describirán los requisitos funcionales educidos a partir de la idea inicial del sistema y del proceso de *brainstorming* realizado. Los requisitos funcionales describen funcionalidades de que debe disponer el sistema con el nivel de detalle suficiente para poder asociar cada uno de ellos en la mayoría de veces a una con una única función directa del sistema.

Requisitos de Usuario

Los requisitos identificados para el subsistema Usuario son los siguientes:

[RF1-U1]

Perfil de usuario

Para poder utilizar la funcionalidad del sistema e identificarse en el mismo de manera unívoca, el usuario debe crearse un perfil de usuario, que será público al resto de usuarios –a excepción de aquella información con carácter estrictamente privado- y con el cual podrá iniciar y cerrar sesión en la aplicación.

[RF-U2]

Información del perfil

La información del usuario contenida en su perfil será aquella determinada por los siguientes campos:

- Nick de usuario (obligatorio), único para cada usuario.
- Nombre de usuario (obligatorio).
- Contraseña (obligatorio).
- Correo electrónico (obligatorio).
- País.
- Edad.

[RF-U3]

Modificación del perfil

El sistema debe permitir al usuario la modificación de los siguientes elementos de su perfil:

- Nombre de usuario.
- Contraseña.
- Edad.
- País.

[RF-U4]

Inicio de sesión

Para ingresar en el sistema el usuario deberá proporcionar la siguiente información de su perfil: nick/correo electrónico y contraseña.

[RF-U5]

Catálogo personal

Dentro del perfil de usuario el sistema debe proporcionar al usuario un catálogo personal, similar a un apartado de "favoritos[3]", donde podrá añadir aquellas publicaciones que le hayan gustado de otros usuarios o de entre sus propias publicaciones.

[RF-U6]

Tablón de usuario

El tablón será el componente principal del perfil del usuario y en él se reflejarán todas sus publicaciones ordenadas por fecha.

[RF-U7]

Lista de borradores

El sistema deberá ofrecer acceso a la lista de borradores de cada usuario, donde aparecerán aquellas publicaciones que aún no haya finalizado. El usuario podrá eliminar cualquier elemento de dicha lista o editarlo en cualquier momento. En el requisito de Publicación P9 se proporciona detalle sobre los borradores.

[RF8-U8]

Seguir/Dejar de seguir

El sistema debe ofrecer a cada usuario la posibilidad de seguir o dejar de seguir la actividad de cualquier otro usuario.

[RF9-U9]

Lista de seguidores/siguiendo

Dentro del perfil de usuario el sistema mostrará la lista de usuarios que le siguen –"seguidores"- y la lista de usuarios a los que sigue –"siguiendo"-.

[3] El apartado de "favoritos" suele ser un subconjunto del contenido de una red social que es seleccionado específicamente por el usuario.

[RF10-U10]

Lista principal

La lista principal constituirá otro de los componentes importantes del sistema y será distinta para cada usuario. La lista principal será una sucesión de las últimas publicaciones realizadas por aquellos usuarios a los que sigue el usuario actual, ordenadas por fecha.

[RF11-U11]

Pantalla principal

En la pantalla principal del usuario, que será cargada tras iniciar sesión, el sistema deberá mostrar al usuario su lista principal además de accesos a los distintos componentes del perfil y al resto de funcionalidades.

[RF12-U12]

Perfil público

Cualquier usuario del sistema deberá poder acceder a la parte pública de cualquier otro usuario, sin necesidad de ninguna relación de seguimiento entre los mismos. Las componentes públicas del perfil serán:

- Tablón de usuario.
- Información de usuario:
  - Nick de usuario.
  - Nombre de usuario.
  - País.
  - Edad.
- Catálogo personal.
- Listas de seguidores/siguiendo.

[RF13-U13]

Búsqueda de usuarios

El sistema deberá ofrecer al usuario la posibilidad de realizar búsqueda de usuarios, tanto en las listas de seguidores/siguiendo como en general. El usuario deberá poder filtrar por:

- Nombre de usuario.
- Nick de usuario.
- País.

[RF14-U14]

Búsqueda de recetas

Como ya se ha explicado anteriormente, uno de los objetivos principales del sistema es constituir un recetario universal, por lo que la búsqueda de recetas se convertirá en algo fundamental de la aplicación. El usuario deberá poder escoger buscar recetas en su Catálogo Personal, en su Tablón y en general filtrando por:

- Nombre de la receta.
- Ingredientes.
- Etiquetas.
- Valoración promedio.
- Autor de la receta.

[RF15-U15]

Top recetas

El sistema deberá ofrecer al usuario acceso a un apartado de top recetas, donde mostrará las publicaciones mejor valoradas publicadas hasta la fecha.

Requisitos de Publicación

A continuación se exponen los requisitos educidos para el subsistema Publicación:

[RF16-P1]

Tipos de publicación

El sistema debe poder diferenciar entre publicaciones de plato y publicaciones de cóctel y ofrecer al usuario la posibilidad de realizar publicaciones de ambos tipos.

[RF18-P2]

Componentes

Se pretende que el sistema pueda incluir en una publicación toda aquella información necesaria para comprender y ejecutar una receta, así como ofrecer a los demás usuarios un medio de interacción con el autor de la misma. Se ha considerado que los componentes necesarios para satisfacerlo son los siguientes:

- Nombre de la receta
- Ingredientes utilizados y su cantidad asociada, si existiera
- Pasos numerados y ordenados que indiquen procedimiento de preparación.
- Comentarios
- Sugerencias
- Etiquetas
- Valoración promedio
- Número de valoraciones

[RF19-P3]

Publicación en detalle

El sistema deberá ofrecer una pantalla donde se pueda visualizar única y exclusivamente la receta deseada en detalle.

[RF20-P4]

Valoración de publicación

El sistema debe dar opción al usuario de valorar una única vez una publicación, asignando un número entre 1 y 5. Una vez realizada la valoración, el usuario no podrá revertir el voto y el sistema deberá actualizar debidamente la valoración promedio de la receta, así como también lo hará con la valoración media del usuario autor de la receta. Además el sistema no debe permitir que un usuario sea capaz de valorar sus propias publicaciones.

[RF21-P5]

Comentarios/Sugerencias

La aplicación deberá permitir a los usuarios añadir comentarios y sugerencias a cualquier publicación. Aunque ambos estén incluidos en el mismo requisito, a efectos prácticos resultan ser instancias de objetos distintos, por lo que se requiere que los comentarios se traten y manipulen por separado de las sugerencias y que por tanto aparezcan en apartados distintos dentro de la publicación a la que pertenecen.

Además en este caso el sistema sí permitirá a los usuarios añadir comentarios o sugerencias a sus propias publicaciones.

[RF22-P6]

Incorporación al catálogo

El sistema debe permitir que cualquier usuario pueda añadir y eliminar cualquier receta a su catálogo personal. Esta posibilidad debe ser accesible desde la propia publicación –añadiendo la receta- o desde su catálogo personal –eliminando la receta-.

[RF23-P7]

Componentes públicos

Todos los componentes de una publicación expuestos en el requisito RF18 deberán ser públicos y accesibles para el resto de usuarios. Además dichos componentes deberán aparecer en zonas claramente diferenciadas unas de otras, aunque todos pertenezcan a la misma receta. Se desea también que los componentes de una receta no formen parte de texto libre de la receta, sino que tengan su propio apartado dentro de la publicación.

[RF24-P8]

Proceso de creación

Para crear una publicación, el usuario primero deberá abrir un borrador en su perfil. El borrador se añadirá a su lista de borradores y podrá ser editado en cualquier momento o descartado de dicha lista.

[RF25-P9]

Campos de un borrador

El borrador constituye una futura publicación que está en proceso de edición. Los distintos campos del mismo que podrá modificar el usuario son los siguientes:

- Nombre de la receta
- Ingredientes utilizados, junto con la cantidad asociada a cada uno.
- Pasos, procedimiento de preparación
- Etiquetas
- Referencias

Al igual que ocurre en la visualización de una receta, en el proceso de edición de un borrador debe aparecer cada componente señalada en un apartado distinto.

[RF26-P10]

Estados de un borrador

El borrador asociado a una publicación deberá tener dos posibles estados: Incompleto y Finalizado. Inicialmente y desde el momento de su creación el borrador estará en estado Incompleto y así deberá indicarlo el sistema. Por consiguiente el usuario deberá tener la posibilidad en cualquier momento de cambiar el estado del borrador a Finalizado, siempre y cuando haya completado todos y cada uno de sus campos.

[RF27-P11]

Publicación de un borrador

En el momento en que un usuario decida finalizar y por tanto publicar un borrador, éste será eliminado de la lista de borradores y el sistema deberá actualizar el tablón del usuario autor de la receta y las listas principales de los usuarios relacionados.

3.3.2. Requisitos no funcionales

Los requisitos no funcionales describen características y restricciones sobre cómo debe operar el sistema.

Seguridad

La aplicación gestionará información de carácter privado asociada a cada usuario, como por ejemplo los datos necesarios para inicio de sesión. El sistema debe proveer de la seguridad necesaria para mantener segura esta información y que solo el creador de la cuenta de usuario pueda tener acceso a la misma.

Usabilidad

La aplicación descrita está dirigida a un público muy amplio incluyendo a usuarios de distintas edades y conocimientos informáticos, por lo que se pretende que la aplicación sea usable, intuitiva, sencilla de utilizar y que evite la carga de memorización por parte del usuario.

Escalabilidad

Se requiere que el sistema sea fácilmente escalable con el fin de poder introducir mejoras y actualizaciones futuras con la mayor facilidad posible y sin tener que rediseñar gran parte del sistema, por ejemplo para introducir los requisitos futuros expuestos más adelante.

Rendimiento

Se pretende que las operaciones de carga más costosas del sistema sean ejecutadas con la mayor rapidez posible, si bien esto garantizará una mayor fluidez de la aplicación e incrementará la sensación de retroalimentación al usuario.

Plataforma

El sistema debe poder ser utilizado a través de la Web vía cualquier navegador web.

Estándares

El sistema debe utilizar el protocolo HTTP como medio básico de comunicación con el usuario.

3.4. Casos de uso

Un caso de uso es una descripción detallada del proceso de manifestación de cierta funcionalidad del software frente a un cierto evento provocado por un actor. Los casos de uso en muchas ocasiones permiten completar la información que pueda haber implícita en uno o más requisitos, como las acciones deben ser ejecutadas por el sistema ante cierta interacción con el usuario.

En este apartado serán descritos de manera general algunos de los casos de uso más representativos de cada subsistema. En el anexo B se encuentra el detalle de cada caso de uso.

Caso de uso Nº1: edición del perfil de usuario.
Caso de uso Nº2: acción de seguir/dejar de seguir a un usuario.
Caso de uso Nº3: búsqueda de usuarios/recetas.
Caso de uso Nº4: crear y guardar un borrador.
Caso de uso Nº5: publicar un borrador.
Caso de uso Nº6: valoración de una publicación.
Caso de uso Nº7: añadir comentario o sugerencia a una publicación.
Caso de uso Nº8: catalogar –añadir al catálogo- una publicación.

4. Diseño

4.1. Restricciones de diseño

La incorporación de ciertos requerimientos requiere tener en cuenta algunos aspectos de diseño que deben ser observados a la hora de elegir una solución.

La primera pauta a tener en cuenta aparece explícita en el requisito no funcional de plataforma, donde se indica que el sistema debe ser implantado sobre la Web y utilizado desde un navegador web. La primera consecuencia que esto origina es que debemos descartar la posibilidad de desarrollar un cliente, puesto que el usuario podrá utilizar cualquier navegador de los que actualmente existen. Lo siguiente a destacar es que la arquitectura subyacente al sistema será ajustada al modelo *cliente-servidor*, por lo que debemos plantear la opción de crear y gestionar un almacén de datos, localizado en la parte del servidor y donde residirán también los componentes necesarios para el procesamiento de peticiones HTTP. Esto implica incorporar una plataforma que disponga de una API o librería que permita realizar ejecuciones mediante el envío de peticiones y respuestas HTTP.

En relación a lo anterior, cabe mencionar también que el sistema debe gestionar una sesión de usuario. Las sesiones de usuario determinan el período de tiempo de la actividad de un usuario del sistema y permiten que el sistema actúe con unos parámetros vinculados al usuario durante un período de tiempo preestablecido. El desarrollo de un sistema como Cooks conlleva implícitamente a la gestión de sesiones en el lado del servidor, por lo que tendremos que elegir una solución que disponga de esta funcionalidad.

Por otro lado a partir de los requisitos podemos deducir que para completar algunas funcionalidades es necesario utilizar peticiones asíncronas, ya que en algunos casos tan solo se solicitará la ejecución de cierta operación de actualización de los datos, más allá del método de formularios web. Las peticiones asíncronas traen consigo una nueva restricción a tener en cuenta relacionada con la consistencia e integridad de los datos de la aplicación.

4.2. Arquitectura de Software

Los patrones de diseño resultan ser plantillas que nos permiten modelar una solución a problemas de desarrollo software. Algunos solo son aplicables a determinados objetivos y su finalidad es permitirnos afrontar problemas de diseño encajando nuestro proyecto en un modelo. En este proyecto la arquitectura del sistema está basada en el patrón de diseño MVC (Modelo-Vista-Controlador).

El MVC es un patrón de diseño que nos permite identificar tres grandes componentes independientes en nuestro sistema: datos –modelo-, funciones de acceso a datos –controlador- y presentación de los mismos al usuario del software –vista-.

En la ilustración 1 se muestran los componentes del sistema, que sigue el patrón de diseño MVC, así como la secuencia de acciones que se producirán en una interacción entre el usuario y el software.

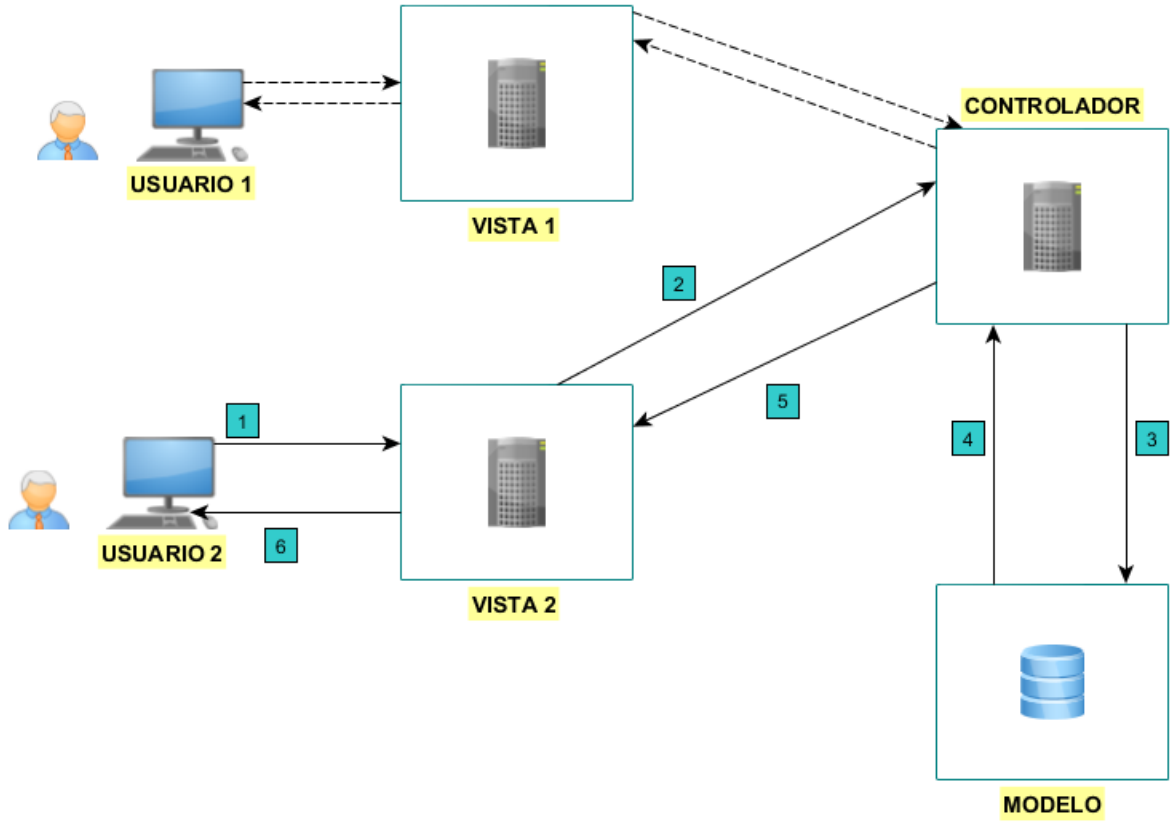

Ilustración 1: Arquitectura de la red social Cooks.

El proceso de funcionamiento del sistema es el siguiente:

En primer lugar, el cliente –vista- recogerá los datos de la interfaz, que serán preparados y enviados al controlador –servidor- mediante HTTP. Por su parte el controlador se encargará de realizar las operaciones adecuadas en el modelo de datos y generará una respuesta que será devuelta a la vista.

Como vemos el único componente que actúa con el modelo vemos que es el controlador, así como la vista es el único componente que tiene contacto con el usuario final. Los elementos de la ilustración de MVC pueden representar servidores físicos distintos.

Con un esquema como éste conseguimos no solo separar componentes de la aplicación, lo que facilita el mantenimiento del sistema, sino permitir que cada componente pueda residir en lugares físicos distintos y simplificar el reemplazamiento de una componente funcional por otra. Además en el caso de la vista se podrían implementar distintas vistas de acceso, por ejemplo vista web, móvil u otros.

4.2.1. Modelo

El modelo de datos es el primer componente funcional e independiente de nuestro sistema.

Los datos constituyen la información que manejará el software y se debe establecer un modelo que formalice correctamente sus características y atributos. El modelo de datos debe garantizar total independencia y abstracción a componentes funcionales externas de cómo se gestiona la información almacenada. En nuestro caso, el modelo de datos albergará toda la información de la red social que ha sido expuesta en la fase de análisis de requisitos.

El modelo de datos resulta ser el almacén de datos que típicamente acaba diseñándose sobre una base de datos. En nuestro caso así será y se procederá a describirla en apartados posteriores.

4.2.2. Vista

Por otro lado tenemos la vista de los datos, que básicamente constituirá la interfaz gráfica de usuario y que utilizará el controlador para acceder a los datos y mostrarlos al usuario del software.

La función principal de esta segunda capa es la de presentar los datos de la aplicación al usuario y ofrecer elementos de interfaz de entrada y salida para que el usuario pueda interactuar con el sistema. A su vez este conjunto de funcionalidades nunca accederá a los datos directamente y en su lugar será el controlador quien haga de capa intermediaria.

Para el caso de este sistema la vista será el sitio web de la aplicación Cooks y en apartados posteriores veremos cómo se ha decido modular esta capa de presentación y el diseño de la misma.

4.2.3. Controlador

Por último tenemos el controlador, que es el conjunto de funciones de acceso a los datos, que nos permitirán leerlos, escribirlos y modificarlos. Estas funciones en ningún momento manipularán los datos directamente, esa tarea está asignada al propio modelo de datos, que deberá proveer a su vez de funciones o de un lenguaje de manipulación de datos.

En nuestro caso esta componente es la API que junto con los datos constituyen el núcleo de la red social, pues podemos decir que la interfaz gráfica es una parte independiente y no tan interesante para este proyecto. Además el diseño del modelo y el controlador deben ser tales que el sistema pueda acoplarse a distintas interfaces sin modificar ninguna de estas dos capas. En apartados siguientes veremos cómo interactúa el controlador con la vista y el modelo de datos y el desglose de funciones que componen esta interfaz de acceso a los datos.

4.3. Descomposición modular del sistema

Vista

Los módulos en que se dividirá la capa de la vista serán dos: generador de páginas y acceso a la API. El módulo de generación de páginas –o generador de interfaz- será el encargado de generar los distintos elementos de la interfaz de usuario a partir de datos de entrada. El módulo de acceso a la API por su parte preparará los datos de entrada provenientes de la interfaz de usuario –generador de páginas- para enviárselos a la API y devolverá los datos de ésta al generador de interfaz.

En el proceso petición-respuesta (ver ilustración 2) el cliente –usuario- tendrá contacto únicamente con el módulo de generación de páginas. Éste procesará la petición del cliente y los datos en ella contenidos, los recogerá y solicitará la ejecución de cierta operación de la API, conectando para ello con el módulo de ACCESO_API. Así mismo esta capa manipulará los parámetros de la solicitud y realizará otra nueva petición a la API solicitando la ejecución de uno (o varios) procedimientos. La API realizará las operaciones oportunas y devolverá los resultados a la capa de ACCESO_API, que procesará la respuesta y devolverá los datos extraídos a la capa de GENERADOR_PAGINAS, que en función de los datos recibidos, generará una página u otra para el usuario.

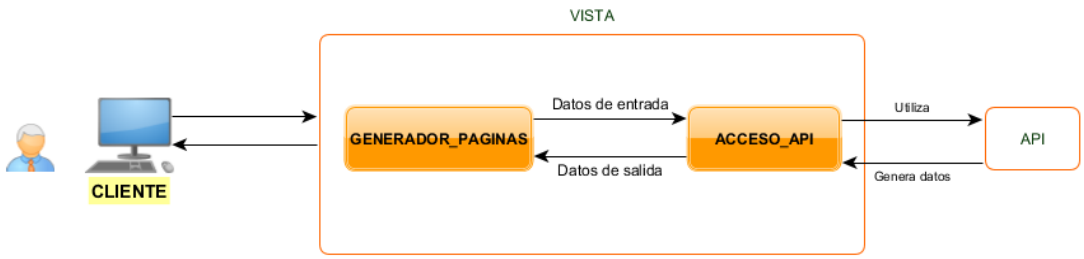

Ilustración 2: Descomposición modular de la vista

Como se ha mencionado antes se ha decidido separar la lógica de la propia interfaz, que es el controlador, de la propia generación de la misma. El motivo de esta decisión tomada es el hecho de que los datos necesitan ser recogidos de la interfaz y preparados para cumplir con los requisitos de la capa del controlador, es decir, deben ser manipulados mediante el proceso pertinente previamente a la interacción con la API.

De esta manera conseguimos de nuevo subdividir esta capa de la vista en conjuntos funcionales distintos, con lo que obtendremos los beneficios aportados por la modularidad, como son la flexibilidad y la portabilidad y mejor localización de errores. No obstante estos módulos no serán independientes entre sí al ser el acceso a la API el módulo intermedio entre la interfaz y la API.

La decisión de incluir la capa de acceso a la API en la vista y no en el controlador está motivada por seguir el principio de independencia del que se ha partido. El módulo de acceso al controlador ACCESO_API manipula datos de entrada del usuario recogidos del generador de páginas y después generará páginas en función de éstos tras procesar los códigos de respuesta y los ficheros de datos provenientes de la API. Si esta funcionalidad se delegara al controlador entonces no sería fácil construir nuevas vistas, que utilicen una tecnología de interfaz gráfica distinta ya que se deberían generar páginas distintas. Podemos pensar que la aplicación puede extenderse a plataformas móviles, con lo cual habría que cambiar parte de la capa de la lógica del negocio, y es justo lo que queremos evitar. En su lugar es deseable que la API tan solo dependa del modelo de datos y de la arquitectura y protocolos subyacentes en el envío de peticiones y respuestas.

Controlador

El controlador se dividirá en dos subsistemas: Usuario y Publicaciones. Esta descomposición otorga flexibilidad al sistema al separar claramente grupos de distinta funcionalidad. De esta manera podemos diseñar y desarrollar ambos componentes por separado.

Adicionalmente se diseñará el módulo de pruebas del controlador que se sitúa al mismo nivel que la vista por acceder ambas de la misma manera al controlador y servirá para validar la funcionalidad del controlador. Al igual que en la capa del controlador, el módulo de pruebas se podrá descomponer de manera independiente. En la ilustración 3 podemos ver un diagrama de interacción del módulo de pruebas y la vista con el controlador.

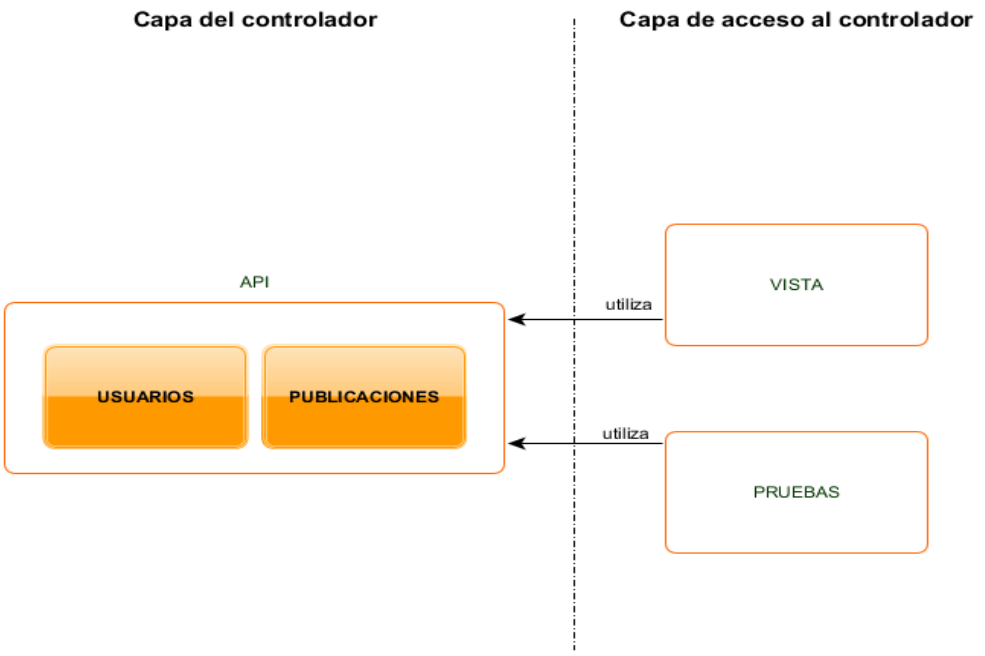

Ilustración 3: Descomposición modular del controlador

La API se ha diseñado de forma que no tenga noción de la existencia de sesiones de usuario y su finalidad se limitará únicamente al acceso a los datos del modelo. Esta funcionalidad será delegada entonces a la capa de ACCESO_API de la Vista, que se encargará de parametrizar la sesión de cada usuario y de controlar el tiempo de la misma. La razón reside en que de este modo los distintos clientes –vistas- que utilicen el controlador serán capaces de gestionar la sesión cada uno de una manera distinta.

4.4. Diseño de la Base de Datos (Modelo)

La base de datos supondrá el almacén de datos del sistema y que manipulará el controlador. Este módulo está compuesto de entidades, procedimientos almacenados y *triggers*.

4.4.1. Diseño conceptual

El primer paso para modelar los datos es identificar aquellos componentes del sistema de los cuales necesitamos guardar información. En el diagrama Entidad-Relación de la ilustración 4 podemos observar un modelado de datos para esta aplicación:

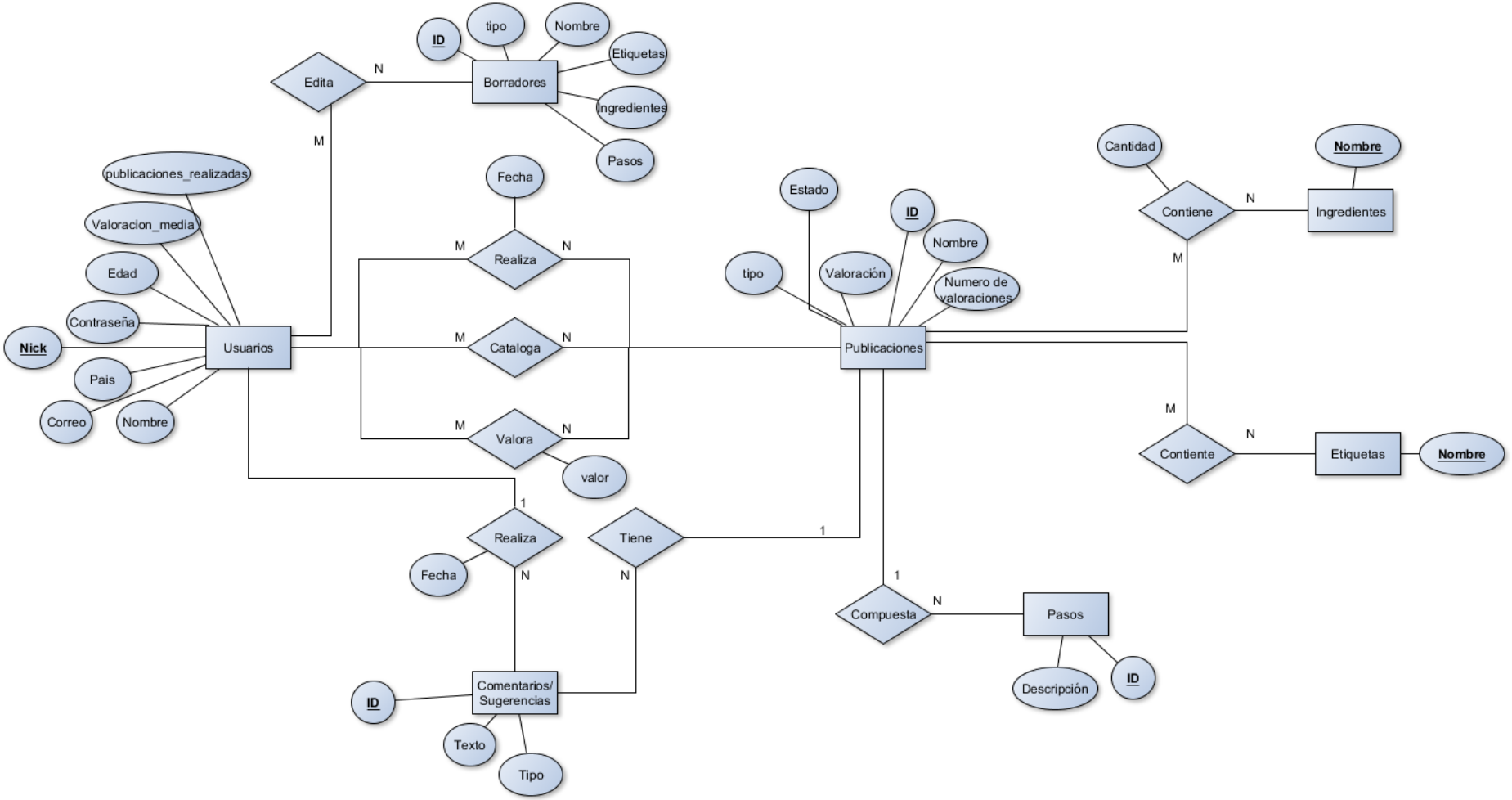

Ilustración 4: Esquema Entidad-Relación de la base de datos

El modelo estará compuesto de 7 entidades fuertes y 9 relaciones. Se le ha añadido un identificador a la mayoría de entidades por carecer de un atributo que los identificara de manera unívoca. Así para las entidades Publicaciones, Borradores, Comentarios/Sugerencias y Pasos tenemos que la clave primaria será un identificador numérico distinto para cada instancia de la entidad.

Para la entidad Usuarios, se ha establecido como atributo unívoco –clave primaria- el Nick de usuario, con lo cual deberá ser distinto para cada usuario de la aplicación. Las entidades Etiquetas e Ingredientes tienen como clave primaria los respectivos nombres asociados. Las entidades Etiquetas, Ingredientes y Pasos son propiedades de una publicación y podrían haberse reflejado como atributos de la entidad Publicaciones. No obstante en los requisitos se establecía que los apartados no supusieran texto libre asociado a la publicación, sino que tuvieran su propio apartado y así se pudieran diferenciar claramente. Además, de cara a las búsquedas de recetas, donde se puede filtrar por ingrediente y etiqueta, resulta ser más cómodo y eficiente que permanezcan en tablas separadas. Esto ocurre porque los sistemas gestores de bases de datos no traen consigo lenguajes de manipulación de datos tan potentes como un lenguaje propio de programación, donde existen estructuras de datos y las operaciones de procesamiento de texto están más optimizadas.

Por su parte vemos que los borradores serán instancias de entidades distintas a las publicaciones. Otra opción podría haber sido generar una sola entidad, Publicaciones, que albergara ambos tipos, diferenciados unos de otros mediante un atributo Tipo. Sin embargo del otro modo restringimos a que las etiquetas e ingredientes, que son utilizados en las búsquedas, solo estén ligados a una instancia de receta real y no a un borrador.

Por otro lado, las relaciones que más pueden llamarnos la atención al observar el esquema, a parte de las evidentes, son por ejemplo las tres relaciones distintas que unen las dos entidades Usuarios y Publicaciones, y corresponderán directamente con las tres acciones que un usuario puede realizar respecto a una publicación, a saber: realizar publicación, añadir publicación al catálogo y valorarla

Por último podemos ver que las entidades Publicaciones y Retos tienen un atributo Estado, que señalarán el estado del borrador correspondiente a la publicación y el estado del reto, respectivamente.

4.4.2. Modelo Relacional

En el paso a Modelo Relacional las 7 entidades fuertes pasarán a corresponder directamente con 7 relaciones (tablas) de la base de datos.

Según el algoritmo de transformación, las relaciones M: N originarán nuevas tablas, cuya clave primaria será una clave compuesta por las claves primarias de las entidades que relacionan. De esta manera tenemos que las relaciones Edita, Realiza, Cataloga, Contiene (Publicaciones – Ingredientes), Contiene (Publicaciones – Etiquetas) transformarán en nuevas tablas del Modelo Relacional. Por otra parte las relaciones 1:N no originan nuevas tablas y en su lugar se incorporará la clave primaria de la entidad que participa con 1 a la entidad que participa con N y será clave externa a la clave primara de la primera entidad. Así se deberá incorporar a la entidad Comentarios/Sugerencias los atributos Nick (por la relación Realiza) e ID (por la relación Tiene) y a la entidad pasos deberemos añadir asimismo el atributo ID de la entidad Publicaciones.

Por último tenemos algunas relaciones con atributos en nuestro modelo de datos que indican propiedades que existen cuando se da la relación. Así vemos que en la relación entre Ingredientes y Publicaciones aparece un atributo cantidad, que será añadido a la tabla intermedia generada. En los casos en los que no existe tabla intermedia el atributo se añadirá a la entidad que participa con cardinalidad N, como es el caso de la relación entre Comentarios/Sugerencias y Usuarios. El atributo *valor* de la relación Valora se añadirá a la tabla intermedia resultante como un atributo más.

4.4.3. Procedimientos almacenados y *triggers*.

A partir de los requisitos y los casos de uso expuestos podemos deducir que deben existir respuestas ante ciertos eventos de actualización de la base de datos. El primero de ellos es la acción de valoración de una receta, reflejado en el caso de uso número 5, donde tras tener registro del valor se debe actualizar la valoración media de la publicación valorada. Asimismo esto genera otra actualización necesaria, que será la valoración media del usuario autor de la valoración.

En el sistema puede ocurrir que solo una publicación contenga una etiqueta o ingrediente determinados. Si dicha publicación se elimina del sistema –función realizada en las pruebas de la API- entonces sería deseable que no hubiera constancia de las etiquetas e ingredientes que la componen en la base de datos. La eliminación de etiquetas e ingredientes por tanto son dos nuevos disparadores que se ejecutarán tras eliminar una publicación. Esta funcionalidad se podría delegar opcionalmente en la API, sin embargo planteándolo del modo expuesto conseguimos que la base de datos sea más independiente del propio software que la utilice.

4.5. Diseño de la API (Controlador)

Hemos visto que la parte asociada al controlador será el conjunto de funciones que permitirán a la vista interactuar con el modelo de datos. El conjunto de funciones ha sido descompuesto en dos subconjuntos distintos atendiendo a los diferentes subsistemas que se han descrito en la fase de análisis. En el anexo A se encuentran detalladas las funciones correspondientes indicando para cada una su breve descripción, los parámetros de entrada, la salida esperada, los requisitos cubiertos por cada función y en los casos en los que proceda la lista de funciones que utiliza. En la tabla 1 se expone el conjunto de funciones resumiendo los detalles más relevantes.

Función	Entrada	Salida	Requisitos
nuevo_perfil	nick, nombre, correo electrónico, contraseña	Código de error	U1,U2
editar_perfil	nombre, contraseña, país, edad	Código de error	U3
login	nick/correo, contraseña	Código de error	U4
obtener_catalogo	usuario	JSON	U5
obtener_tablón	usuario	JSON	U6
obtener_borradores	usuario (sesión)	JSON	U7
eliminar_borrador	id_receta, usuario (sesión)	Código de error	U7
seguir	usuario1, usuario2	Código de error	U8
dejar_de_seguir	usuario1, usuario2	Código de error	U8
obtener_seguidores	usuario	JSON	U9
obtener_siguiendo	usuario	JSON	U9
obtener_lista_principal	usuario	JSON	U10, U11
obtener_informacion_perfil	usuario	JSON	U12
es_seguido_por	usuario1, usuario2	Código de error	U8
buscar_usuario	usuario. seguidores/siguiendo/general, nombre, nick, pais	JSON	U13
buscar_publicacion	usuario. catálogo/tablón/general, nombre, ingredientes, etiquetas, valoración, autor	JSON	U14
obtener_top_recetas	-	JSON	U15
borrar_perfil	nick	Código de error	(pruebas)
editar_receta	id_receta, usuario_creador, estado, nombre, tipo, fecha, ingredientes, pasos, etiquetas	Código de error	P1, P2, P7, P8, P9, P10, P11
valorar_publicacion	usuario, id_receta, valoración	Código de error	P4
añadir_comentario	usuario, id_receta, texto	Código de error	P5
añadir_sugerencia	usuario, id_receta, texto	Código de error	P6
catalogar	id_receta, usuario	Código de error	P6
descatalogar	id_receta, usuario	Código de error	P6
obtener_informacion_receta	id_receta	JSON	P3, P7
obtener_ingredientes	id_receta	JSON	P3, P7
obtener_pasos	id_receta	JSON	P3, P7
obtener_comentarios	id_receta	JSON	P3, P7
obtener_sugerencias	id_receta	JSON	P3, P7
obtener_etiquetas	id_receta	JSON	P3, P7
obtener_valoracion_realizada	id_receta, usuario	Código de error	P3, P4, P7
obtener_ha_valorado	id_receta, usuario	Código de error	P3, P4, P7
obtener_ha_catalogado	id_receta, usuario	Código de error	P6
obtener_informacion_borrador	id_receta	JSON	P9
eliminar_borrador	id_borrador	Código de error	P8
eliminar_publicacion	id_receta	Código de error	(pruebas)

Tabla 1: Funciones de la API

Como vemos hay numerosas funciones de obtención de datos que en su totalidad devolverán ficheros en formato JSON.

Por otro lado vemos que existen dos funciones que no están asociadas a ningún requisito y en su lugar están vinculadas a las pruebas del sistema. Estas funciones son *eliminar_usuario*, *eliminar_publicación* y *borrar_perfil*. Estas funciones serán necesarias para la fase de pruebas de la API.

Estas funciones deben ser ejecutables mediante peticiones HTTP, por lo que los parámetros de cada una deberán ser enviados y recibidos mediante los métodos GET o POST. Se puede plantear además la posibilidad de que puedan ser ejecutadas mediante una simple llamada al procedimiento- en local- de cara a las pruebas.

4.6. Diseño de la Interfaz Gráfica de Usuario (Vista)

El primer paso ha sido definir a grandes rasgos la estructura de cada página de la aplicación. Muchas de ellas tendrán una estructura similar y cabe decir que no se ha ahondado demasiado en generar una interfaz muy elaborada y profesional desde un punto de vista de diseño gráfico. Para este proyecto la vista es el componente menos relevante del conjunto Modelo, Vista y Controlador y es suficiente con que cumpla los requisitos de interfaz impuestos: interfaz sencilla e intuitiva.

A continuación se muestra la estructura aproximada de que deberán disponer las páginas web de la aplicación. A todos los tipos de pantalla se les ha asociado un nombre para poder referirnos a cada una de ellas de manera directa más adelante en el mapa de navegación.

4.6.1. Diseño conceptual de las páginas

➔ Pantalla de inicio :: *Pantalla 1*

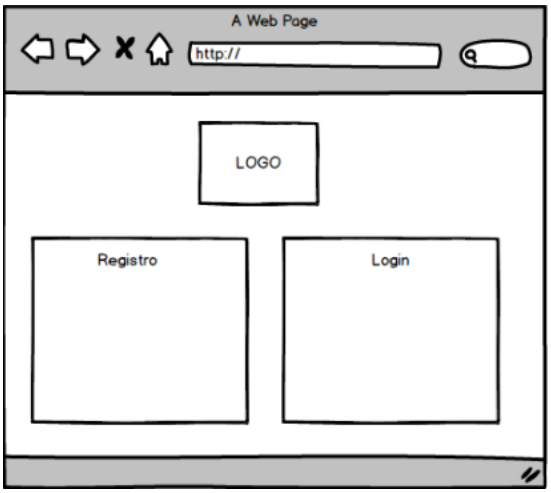

Ilustración 5: Diseño de la pantalla de Inicio

La pantalla de inicio será la primera que encontrará el usuario al conectar con la aplicación. Como vemos dispondrá de una sencilla interfaz con dos formularios: uno para registro y otro para iniciar sesión en el sistema.

➔ Pantalla de navegación usual :: *Pantalla 2*

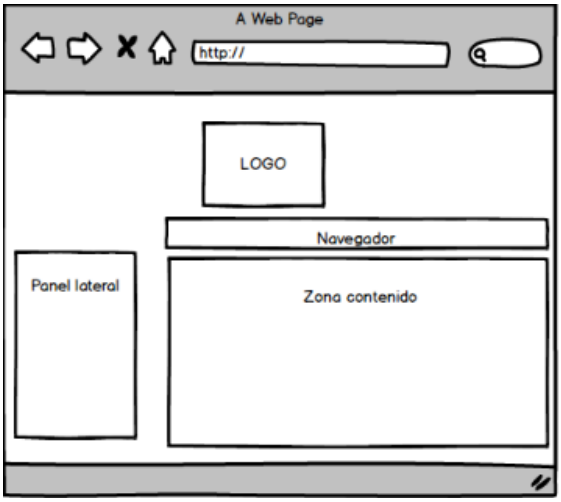

Ilustración 6: Diseño de la pantalla de navegación común

Esta ilustración sirve para representar la estructura de elementos de que deben disponer la mayoría de páginas del sistema.

El *Panel lateral* y *Navegador* son zonas entre las que se deberá distribuir los distintos apartados y funcionalidad a los que tiene acceso un usuario, tales como la búsqueda de usuarios y recetas o los apartados del propio perfil.

En la zona de contenido se reflejará toda la información correspondiente que deba mostrar la aplicación.

➔ Pantalla de búsqueda :: *Pantalla 3*

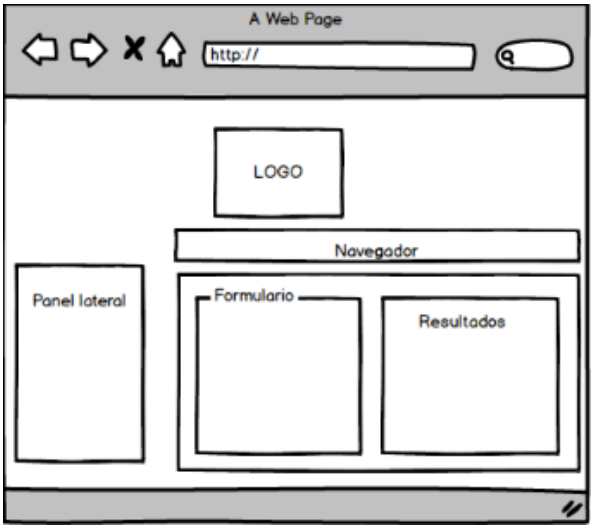

Ilustración 7: Diseño de la pantalla de búsqueda

Este es el esquema de la pantalla de búsqueda, aplicable tanto a búsqueda de recetas como a búsqueda de usuarios. Por supuesto puede verse modificada en diferentes aspectos, el que más llama la atención es quizá el hecho de que la zona de búsqueda –formulario + resultados- no se vea a pantalla completa, pero se pretende que los resultados sean vistos en miniatura, por lo que así puede ser más que suficiente.

➔ Pantalla de receta en detalle :: *Pantalla 4*

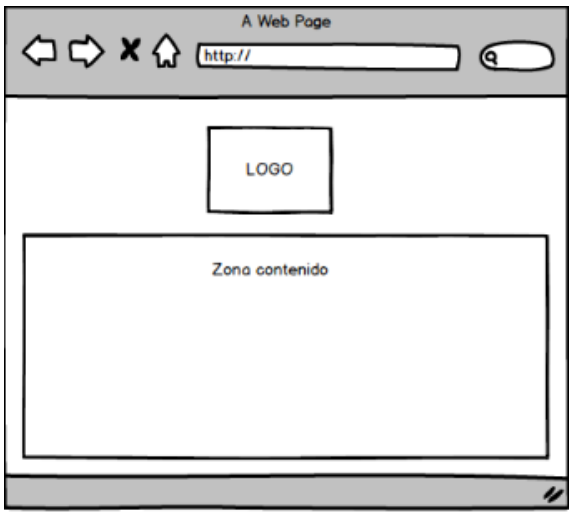

Ilustración 8: Diseño de la pantalla de receta en detalle

Esta pantalla no tiene más trascendencia que las anteriores y tan solo sirve para mostrar una variación de la pantalla 2 en la que hemos obviado las zonas laterales y superiores para mostrar solo el contenido.

Esta pantalla es aplicable por ejemplo a la edición de un borrador o a la pantalla de detalle de receta.

➔ Elementos de receta detalle / edición de borrador

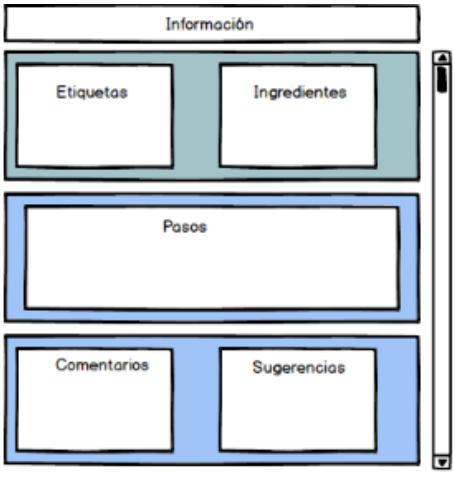

Ilustración 9: Elementos de la pantalla de receta en detalle

En esta última ilustración se muestra un esquema de la vista de una receta en detalle o la edición de un borrador. En *Información* podemos incluir la diversa información asociada a la receta y en los apartados subsiguientes podemos distinguir las zonas que se requería que fueran bien diferenciadas en los requisitos.

A esta imagen no se le asocia ningún número de pantalla porque queda implícito que será la estructura interna de la pantalla 4.

Por su parte las zonas de *Comentarios* y *Sugerencias* no deberían aparecer en la pantalla de edición de un borrador por no ser una receta pública.

➔ Pantalla de información/error :: *Pantalla 5*

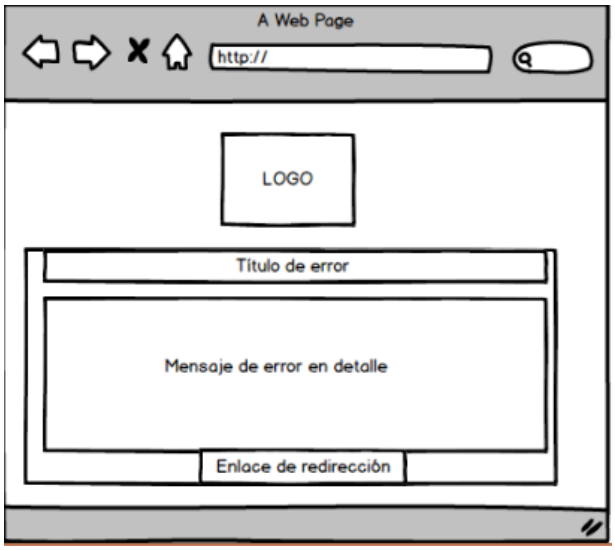

Ilustración 10: Diseño de la pantalla de error/información1

Por último tenemos la página de error o de información, en la que se mostrará con esta estructura cualquier mensaje de error o de información que la aplicación deba mostrar al usuario. En la parte inferior de la página se ha incluido una zona donde típicamente se podrá proporcionar un enlace directo a una página concreta.

4.6.2. Esquema de navegación entre páginas

En el esquema de la Ilustración 11 se muestran las páginas como miniaturas que tendrán asociados el tipo de pantalla, de entre las expuestas antes, el título de la página y la acción del usuario que provoca el cambio entre pantallas.

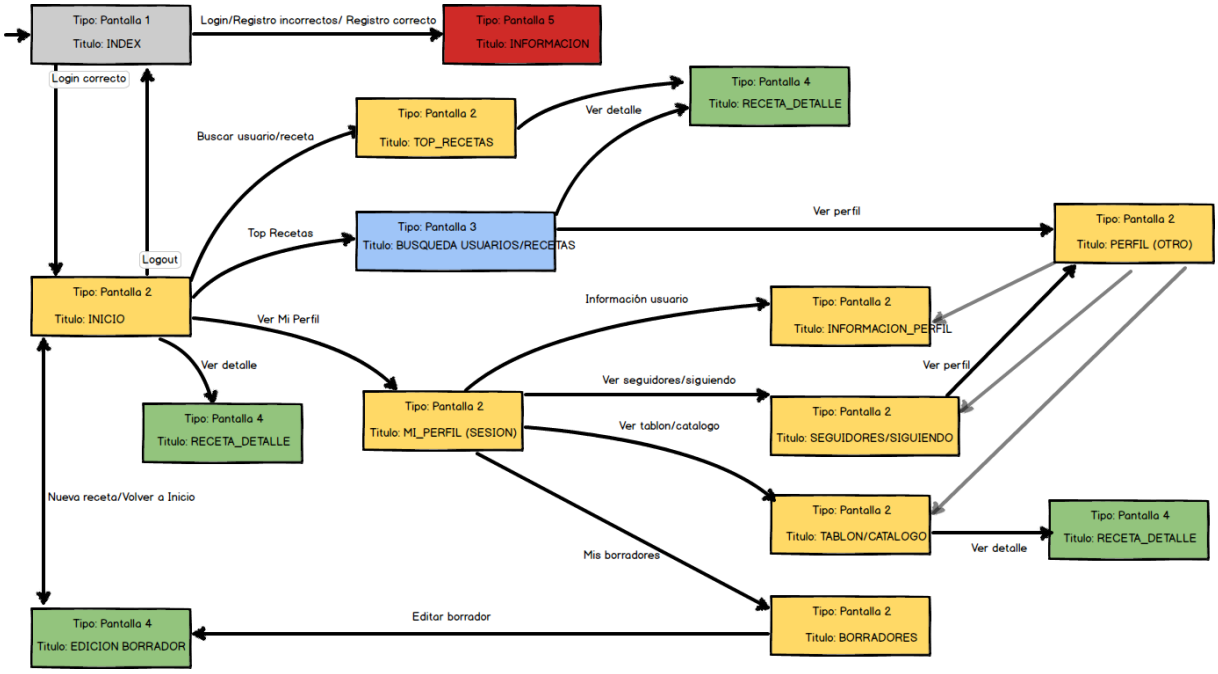

Ilustración 11: Diagrama de navegación entre las pantallas de la interfaz gráfica

No han sido mostradas todas las posibles acciones de un usuario desde cada página con el fin de no saturar el esquema de flechas y cajas. Además algunas de estas relaciones se consideran implícitas. Por ejemplo la página de INFORMACION solo se refleja 1 vez en el esquema y sin embargo esta página será mostrada, como se ha mencionado antes, siempre que el sistema deba informar de algo al usuario.

La página inicial será la de INDEX, desde la cual el usuario será capaz de registrarse o ingresar en el sistema. Si inicia sesión correctamente, navegará a la página principal que será la de INICIO, en la cual se le mostrará la lista principal mencionada en los requisitos y enlaces para acceder a cada publicación allí mostrada.

Desde la página de INICIO el usuario tendrá acceso a TOP_RECETAS, BUSQUEDA y MI_PERFIL. En la pantalla TOP_RECETAS el usuario podrá visualizar el ránking actual de recetas de la aplicación y en la pantalla BUSQUEDA podrá realizar búsquedas de usuarios y recetas –aunque en pantallas diferentes-. Asimismo desde estas dos últimas páginas podrá acceder al detalle de las recetas o al perfil de los usuarios buscados.

Desde la página de INICIO el usuario también podrá acceder a la creación de un borrador nuevo o a todos los apartados dentro del perfil propio del usuario. La pantalla de INFORMACION_PERFIL mostrará sus datos de perfil con opción a la edición de los mismos. La pantalla SEGUIDORES/SIGUIENDO en realidad constituyen dos páginas distintas, pero se han unificado en este esquema por situarse en la misma posición de navegación.

En ellas se mostrarán respectivamente las listas de seguidores y de siguiendo del usuario propietario del perfil que se está visualizando y darán posibilidad de acceder al perfil de cada usuario de dichas listas. Lo mismo también ocurre con las páginas TABLON/CATALOGO, en las que se mostrarán las listas de todas las publicaciones y el catálogo personal del usuario, respectivamente, y también darán acceso a la visualización de la receta en detalle.

Por otro lado podemos observar que en el lado derecho a las pantallas del perfil de usuario existen enlaces de un color más claro, que van desde la página OTRO_PERFIL hasta las correspondientes páginas de perfil de usuario. La página de OTRO_PERFIL pretende ser la pantalla de perfil de un usuario cualquiera, distinto al que ha iniciado sesión. Esta distinción ha sido necesaria para poder reflejar claramente que la lista de borradores, visible en la página BORRADORES, solo pertenece al perfil propio, por lo que cada usuario solo podrá visualizar su propia lista de borradores, que es lo que implícitamente se indicaba en los requisitos.

5. Implementación

5.1. Plataformas y tecnologías empleadas

5.1.1. Base de Datos

Las diferentes opciones de que disponemos para la implementación de la base de datos se encuentran entre los diferentes SGBD (Sistemas Gestores de Bases de Datos) a elegir. Cada gestor ofrece una experiencia distinta con la base de datos y sus respectivos desarrolladores incluyen diferentes garantías respecto a los datos almacenados.

Disponemos de gestores como Oracle, MySQL o SQL Server, entre otros. De entre ellos el gestor elegido ha sido PostgreSQL. Las razones residen en que es una licencia de software libre y es un gestor bastante completo y eficiente que contiene todas las funcionalidades necesarias para este proyecto como son *triggers*, procedimientos almacenados, secuencias, etc.

5.1.2. API

En cuanto a la API se ha valorado entre utilizar los *frameworks* Java y PHP.

En primer lugar tenemos la conocida plataforma Java, que mediante *servlets* y páginas JSP o JSF podemos crear un servidor HTTP muy consolidado. Además ofrece la tecnología JPA(*Java Persistance API*) [10] mediante la cual podemos crear una lógica de negocio orientada a objetos y una base de datos también con este paradigma para hacer que cualquier cambio en la lógica del sistema se haga persistente en la base de datos mediante una sencilla secuencia de operaciones. Sin embargo el diseño de este proyecto no está orientado a objetos, y la API está especificada por medio de funciones remotas del servidor, por lo que no podríamos sacar el mayor rendimiento a esta tecnología.

Por otro lado tenemos PHP, que resulta ser una plataforma usada casi tan frecuentemente como Java pero en sistemas de menor amplitud o a pequeña escala. Para esta aplicación se ha utilizado este lenguaje frente a Java. PHP es un lenguaje multiparadigma[4] que permite tanto orientación a objetos como programación procedural, y también dispone de elementos que recuerdan a la programación funcional. PHP tiene una sintaxis sencilla y ahorra al desarrollador el trabajo de generar código objeto y de desplegar la aplicación. Dispone además de una API de acceso a bases de datos –PDO- que resulta fácil de usar y proporciona funciones necesarias para evitar inyecciones de código SQL en el servicio web. Sin embargo un inconveniente a destacar es la carencia de tipos, o el tipado dinámico de datos, que tendrá como consecuencia realizar comprobaciones más exhaustivas en la entrada de datos por parte del usuario, aunque también resulta ser un lenguaje más flexible y menos restrictivo que Java.

[4] Lenguaje multiparadigma: lenguaje de programación que admite la utilización de diferentes paradigmas de programación.

5.1.3. Vista

Para la toma de decisión de la tecnología a emplear en esta componente tenemos un condicionante de partida, y es que sea una vista web utilizable desde un navegador. Para ello podríamos pensar en cualquier tecnología que permitiera generar código HTML, como Java o el propio PHP, a partir de ficheros de datos devueltos por el controlador. Para ello vamos a utilizar PHP por ofrecer un procesado directo de los ficheros de datos JSON de la API. De hecho la otra alternativa a JSON es XML, pero ha sido elegido JSON por esta misma razón. El procesado de estos ficheros con PHP es directo y además es un estándar orientado para utilizarlo junto al lenguaje de cliente JS –Javascript-.

Respecto a los submódulos ACCESO_API y GENERADOR_PAGINAS, se ha utilizado también PHP para simplificar la comunicación entre módulos.

En caso de no existir la restricción impuesta en los requisitos de ser accesible vía web, podríamos sopesar la opción de crear una vista para dispositivos móviles ya que la API ha sido diseñada para permitir la creación de múltiples vistas independientes.

Por otra parte debemos dar detalles respecto a la tecnología utilizada para las peticiones asíncronas, que serán necesarias para refrescar ciertas zonas de las páginas generadas sin actualizar la página entera. Por esta razón PHP queda fuera del alcance, puesto que al ser un lenguaje de lado del servidor no permite este tipo de funcionalidad. Para estas peticiones utilizaremos AJAX (*Asynchronous Javascript and XML*) que mediante Javascript permite realizar peticiones HTTP asíncronas.

5.1.4. Versiones de las herramientas y paquetes utilizados

Para la base de datos se ha utilizado la versión de PostgreSQL 8.4.8 junto con la herramienta PgAdmin III, que se trata de un programa para administrar bases de datos PostgreSQL en un entorno gráfico bastante amigable.

Servidor web Apache 2 junto con los paquetes *apache2 2.2.14* y *apache2-mpm-prefork 2.2.14*.

En cuanto al desarrollo web se ha utilizado la versión 5 de PHP con los paquetes *php5 5.3.2 y php5-pgsql 5.3.2*, siendo este último necesario para obtener los drivers que posibilitan la conexión de PHP con PostgreSQL. Por otro lado tenemos la inclusión de la librería cURL[5] de PHP que permite al servidor actuar como cliente y así ser capaz de enviar y recibir peticiones HTTP a otro sistema terminal.

El sistema operativo de desarrollo ha sido Ubuntu 10.04 en su versión de los laboratorios de la UAM junto con el navegador web Mozilla Firefox.

[5] http://php.net/manual/es/book.curl.php

5.2. Detalles de implementación

5.2.1. Base de Datos

El primer aspecto a tener en cuenta es la solución a implantar para los campos de identificadores numéricos de aquellas tablas que los contengan. Estos identificadores deberán ser números enteros autoincrementales[6].Para ello se han creado secuencias SQL, que permiten al gestor crear una serie de números enteros únicos entre un intervalo de valores, así como acceder al valor actual de una secuencia y establecerlo. Las secuencias son objetos independientes de cualquier otro objeto de la base de datos y se pueden utilizar por ejemplo para vincularlas a un campo de una o varias tablas, indicándole al gestor que dicho campo continúa la serie numérica establecida por una secuencia concreta.

En concreto, se han creado 4 secuencias vinculadas a cada uno de los campos de las 7 tablas del modelo de datos que poseen identificador numérico, a saber: PUBLICACIONES, COMENTARIOS_SUGERENCIAS, PASOS y BORRADORES. Estos campos numéricos sirven para identificar cada instancia de manera unívoca, por lo que el gestor automáticamente creará índices para dichos campos.

Por otra parte se han añadido algunas restricciones que no estaban explícitas en el diseño, como es la restricción de clave única añadida al campo de correo electrónico de un usuario de la aplicación. Aunque pueda parecer innecesario, pues se da por supuesto que no hay dos direcciones de correo electrónico idénticas, sin embargo en los requisitos se establece que el usuario deberá ser capaz de iniciar sesión con la cuenta de correo electrónico como alternativa a su nick asociado, por lo que esta restricción se convierte en necesaria para controlar además la entrada de datos del registro de usuario con el fin de evitar que se registren dos usuarios distintos con la misma dirección de e-mail.

5.2.2. API

En las restricciones de diseño se mencionaba la utilización de la API mediante el protocolo HTTP. Para PHP existen varios métodos de resolver este problema y se han barajado dos de ellos: RPC y HTTP *wrappers*. Se podría incorporar el uso de la arquitectura REST, pero está orientada al uso y representación de recursos, en contraposición con RPC, que está orientado a funciones que es lo que deseamos. Por su parte RMI es una tecnología basada en Java, por lo que esta opción queda descartada.

RPC es una tecnología que resuelve el problema de llamadas a procedimientos remotos. Existen librerías implementadas en PHP para utilizar la funcionalidad de RPC, por ejemplo el repositorio PEAR2[7] ofrece una API para utilizar el protocolo XML-RPC.

---

[6] Se refiere a los campos numéricos que son incrementados por el SGBD con la inserción de cada registro nuevo.
[7] http://pear2.php.net/

La otra opción son HTTP *wrappers*. En realidad este método es una ligera variación de RPC, o una simplificación. En esta solución existirá una página PHP por función PHP de la API. Estas páginas serán envoltorios de las funciones –o *wrappers*- que manejarán las peticiones HTTP entrantes -*handlers*-. El manejador extraerá de la petición los parámetros de ejecución de función, la ejecutará y devolverá los resultados en las respuestas HTTP. Esta ha sido la alternativa escogida frente a RPC por simplicidad y por evitar la implantación de una nueva tecnología, que es estándar pero requiere la existencia de una API implantada sobre HTTP en el lado del cliente aparte de la propia de este protocolo. Podemos observar que con esto ganamos en independencia del controlador por restringirse únicamente a HTTP y en rendimiento al evitar el envío de datos XML.

Así la API deberá ofrecer un manejador de peticiones HTTP por cada función, que realizará la llamada al procedimiento de la API solicitado. Para ello el catálogo de funciones debe ser público para que así la vista pueda tener conocimiento de las funciones disponibles. Cada página PHP irá vinculada a una única función PHP, por lo que prescindimos de realizar las consecuentes comprobaciones. La estructura final del módulo API quedaría finalmente como se indica en la ilustración 12.

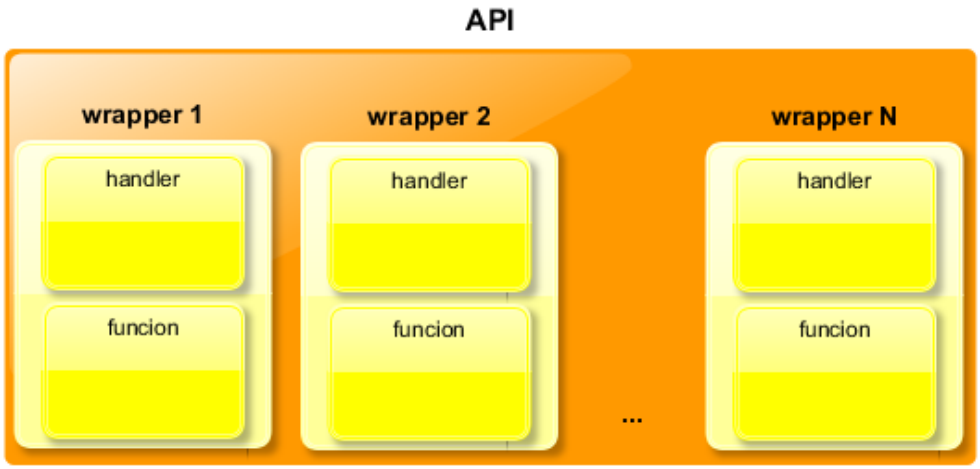

Ilustración 12: Estructura de los módulos en la implementación de la API

En lo referente a la seguridad del sistema, los datos de los usuarios, en particular a sus contraseñas, se ha decidido guardar en la base de datos únicamente el hash MD5 de la contraseña. De esta manera aunque un intruso del sistema pudiera obtener las contraseñas de los usuarios, nunca sería capaz de utilizarlas puesto que lo que ha obtenido no es el texto plano de la contraseña.

Otro aspecto de seguridad al que se ha dado respuesta es a la inyección de código SQL. La inyección de código SQL es un método muy común utilizado en sistemas web que acceden en *back-end* a una base de datos, con el fin de obtener datos de tablas internas del sistema para lograr obtener información del resto de la base de datos.

Para ello el usuario interesado intentará introducir en los formularios de las páginas del sistema fragmentos de sentencias SQL, con el fin de manipular las consultas de obtención de datos tipo SELECT y así obtener datos privados. Para evitar la inyección de código SQL PHP provee de una serie de funciones que permiten crear sentencias SQL preparadas con las que se puede vincular variables a las mismas en lugar de concatenar su contenido dentro de la consulta.

Además, se ha implementado un comprobador de formularios. Este módulo interno a la API se encargará de validar los datos de entrada recibidos por el controlador y verificar que cumplen con los requisitos impuestos y con los campos correspondientes en la base de datos. Este proceso bien podría realizarse en el lado del cliente mediante lenguajes del lado del cliente como Javascript. Sin embargo se ha optado por delegar la validación a aquella componente de la arquitectura que realmente accede y actualiza los datos del modelo de datos. Sin embargo, la gran mayoría de sitios web también realizan esta validación en el lado del cliente por motivos de usabilidad y evitar que las páginas de formularios se recarguen con cada "submit". De esta manera, también se consigue evitar reducir las peticiones al servidor. Podemos concluir por tanto que es condición necesaria y suficiente la validación en el lado del servidor –para nuestro caso, controlador- y además es una buena opción incluir la validación en el lado del cliente –en nuestro caso, vista- y que aunque por el momento no se haya añadido, constituye una mejora futura interesante.

5.2.3. Vista

El primer detalle de implementación es la manera de enfocar el diseño previo realizado de dos subcapas para la vista.

Por un lado tenemos la capa de acceso a la API. Este primer módulo estará compuesto exclusivamente de páginas PHP que incluirán la funcionalidad necesaria para recibir parámetros de entrada por el método POST de HTTP, preparar las estructuras y tipos de datos necesarios y enviarlos al controlador mediante otra petición HTTP. Esta capa también se encargará de procesar las respuestas conteniendo códigos de error y JSON y utilizará las funciones de generación de interfaz para generar la vista que será presentada al usuario.

Por su parte el generador de interfaz estará compuesto de las propias páginas que serán mostradas al usuario, que serán rellenadas con los datos pertinentes obtenidos de la API.

Por otro lado se ha hecho especial énfasis en separar la declaración de componentes de la vista de sus estilos asociados. Así el generador de páginas tan solo producirá páginas HTML, cuyos estilos estarán contenidos a parte en una hoja de estilos CSS.

Por otra parte tenemos la inclusión del lenguaje Javascript, desde algunas sencillas funciones que cambian dinámicamente algún componente de la vista hasta funciones que utilizan la tecnología AJAX para realizar las peticiones asíncronas adecuadas.

6. Pruebas y resultados

6.1. Base de Datos

Las pruebas realizadas contra la base de datos constituyen un conjunto de pruebas de integridad para asegurar el correcto funcionamiento de las reglas y procedimientos de los datos.

Para verificar los datos y las reglas impuestas sobre ellos se ha probado a crear y eliminar registros y comprobar que restricciones como las de unicidad, claves primarias o ajenas y campos no nulos son cumplidas.

Por otro lado también se han realizado pruebas de los procedimientos almacenados y *triggers* de la base de datos. Para ello se ha procedido como en el caso anterior, insertando registros de prueba para usuarios y publicaciones y actualizándolos para observar la ejecución del *trigger* asociado a la tabla actualizada. Por ejemplo para la comprobación de los dos *triggers* que actualizan la valoración media de publicación y valoración media de usuario se ha probado a valorar una publicación y observando estos campos en el registro implicado.

Los resultados de las pruebas de la base de datos han sido satisfactorios. Todas las restricciones han sido cumplidas, los *triggers* responden correctamente ante los eventos asociados y las secuencias establecen valores únicos para los identificadores de las tablas.

6.2. API

El tipo de pruebas realizadas a la API constituyen pruebas de caja blanca. En este tipo de pruebas se conoce la funcionalidad interna de cada módulo o función, así como los distintos flujos posibles que puede seguir el programa. Para ello en este tipo de pruebas se deberán verificar todos los flujos de control, de estados y de datos posibles.

Las pruebas realizadas verifican los siguientes aspectos:

- Parámetros nulos.
- Parámetros no válidos.
- Parámetros vacíos.
- Parámetros de distinto tipo al esperado.
- Cadenas con espacios en blanco.
- Llamada correcta al procedimiento.
- Campos repetidos.

Inicialmente se deben insertar registros de prueba, como son usuarios, seguimiento y publicaciones con el fin de poder probar toda la funcionalidad. Estos registros deben ser eliminados al finalizar las pruebas.

Para implementar las pruebas de la API se ha generado un fichero de funciones de prueba. Para la realización de algunos casos de prueba se han creado grupos de funciones (ver tabla 2) con objetivos comunes, por lo que no siempre una función de prueba se corresponderá con una función de la API.

Función de prueba	Parámetros	Salida	Funciones de la API	Casos de prueba generados
insert_prueba	-	id_publicacion, id_borrador	nuevo_perfil, seguir, editar_receta	17
test_acciones_receta	id_publicacion	-	catalogar, descatalogar, obtener_ha_catalogado, valorar_publicacion, obtener_ha_valorado, obtener_valoracion_realizada, anadir_comentario, anadir_sugerencia	95
test_acceso_receta	id_publicacion	-	obtener_informacion_receta, obtener_ingredientes_receta, obtener_etiquetas_receta, obtener_pasos _receta, obtener_comentarios, obtener_sugerencias, obtener_informacion_borrador	35
test_acceso_perfil	-	-	obtener_informacion_perfil, obtener_lista_principal, obtener_seguidores, obtener_siguiendo, obtener_tablon, obtener_catalogo, obtener_lista_borradores, es_seguido_por, obtener_top_recetas	41
test_registro	-	-	nuevo_perfil	30
test_login	-	-	login	15
test_edicion_perfil	-	-	editar_perfil	33
test_seguimiento	-	-	seguir, dejar_de_ seguir	23
test_edicion_receta	id_borrador, id_publicacion	-	editar_receta	59
test_busqueda_recetas		-	buscar_receta, editar_receta, catalogar, valorar_publicacion	119
test_busqueda_usuarios		-	buscar_usuario	26
delete_prueba	id_borrador, id_publicacion	-	dejar_de_seguir, borrar_perfil, eliminar_publicacion, eliminar_borrador	17

Tabla 2: Funciones de prueba de la API

En total se han realizado 510 pruebas contra la API de la red social Cooks. El programa de ejecución de las pruebas se encuentra en un fichero PHP aparte y llamará a las funciones de prueba mostradas. Los resultados de todas las pruebas han sido satisfactorios.

6.3. Vista

Las pruebas de la vista, al contrario que las pruebas de la API, no requieren una implementación previa a la ejecución. Sin embargo sí que es necesario establecer los elementos que deben ser probados, como la navegación correcta entre pantallas, validación de formularios y actualización correcta de las páginas.

En las ilustraciones 13, 14, 15 y 16 podemos ver un ejemplo de la prueba del caso de uso número 4 en el que se crea y se guarda un borrador. En la ilustración 16 se muestra una captura del tablón del usuario autor de la receta, donde se puede observar que la receta se ha añadido correctamente.

Ilustración 13: Pantalla 1 de creación de receta

Ilustración 14: Pantalla 2 de creación de receta

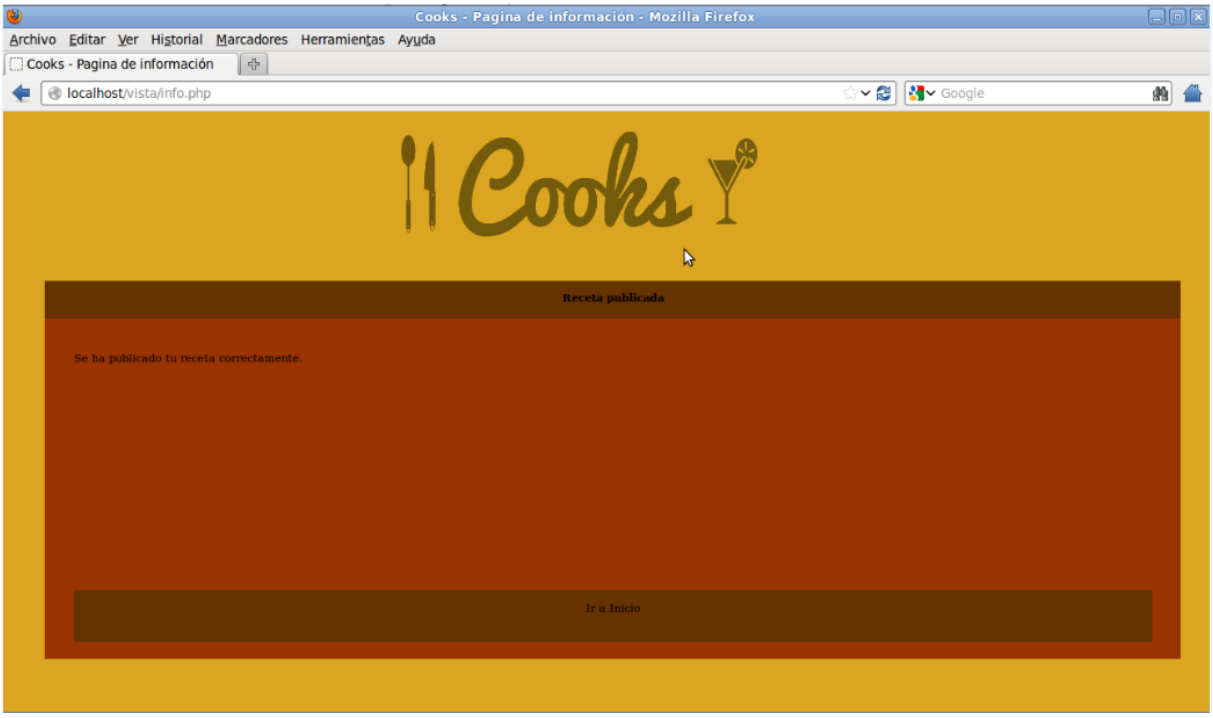

Ilustración 15: Pantalla de confirmación de publicación de receta

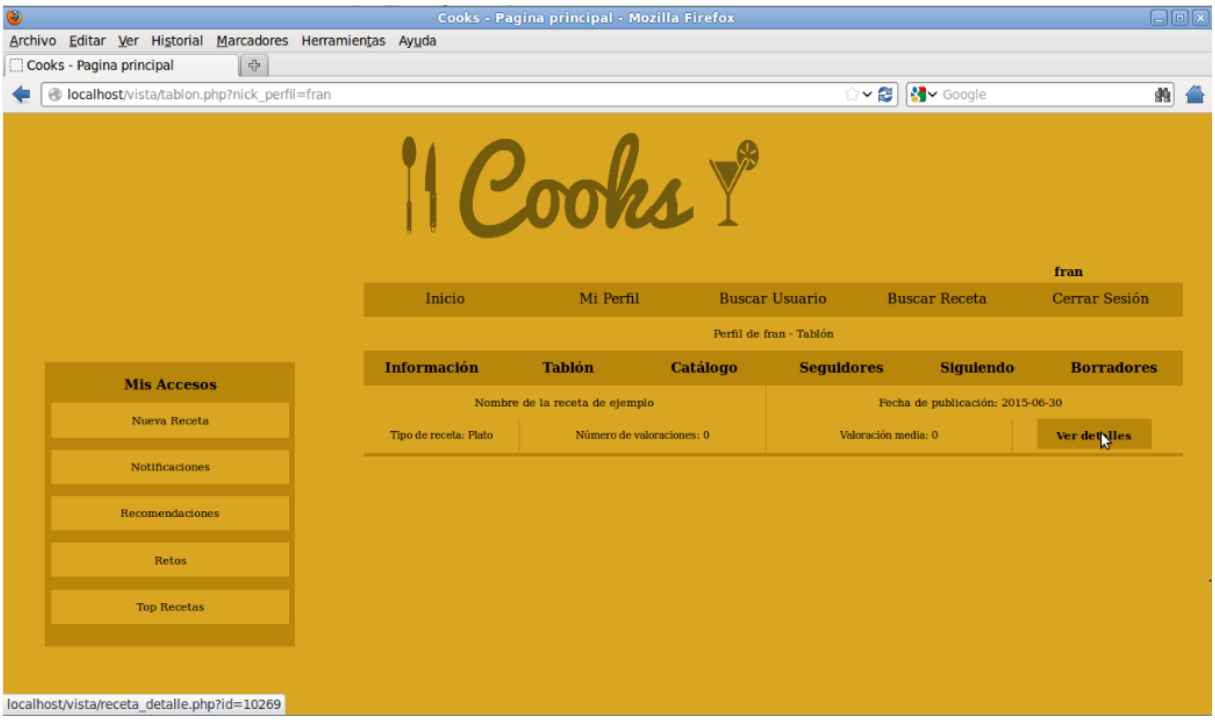

Ilustración 16: Pantalla de visualización de la receta creada

7. Conclusiones y trabajo futuro

En este trabajo se ha diseñado e implementado una red social –Cooks- orientada a cocina y coctelería que permite a los usuarios realizar un seguimiento de la actividad de otros y construir un recetario personal, al que podrá añadir recetas e incorporar las de otros usuarios.

El proceso seguido para el desarrollo de Cooks ha sido en primer lugar realizar un análisis de los requisitos que debe cumplir, donde se han identificado las funcionalidades y objetivos del sistema. Después se ha realizado un diseño de la aplicación, estableciendo la arquitectura a adoptar para resolver el problema planteado, tras lo cual se ha llevado a cabo la implementación de la aplicación, donde se han estudiado las distintas soluciones técnicas adaptables al diseño previo y donde también se han construido todos los componentes del sistema para su despliegue y utilización. Por último se han realizado las tareas de la fase de pruebas, donde se ha verificado el cumplimiento de todos los requisitos.

Para afrontar el diseño del sistema primero se ha descrito la arquitectura de software, que está basada en el patrón de diseño MVC. Para ello se han identificado los distintos componentes que conforman el sistema, constituyendo el modelo una base de datos, el controlador la API de acceso y la vista como la interfaz gráfica de usuario. En la descripción de la arquitectura también se ha establecido la forma de comunicación e interacción entre ellos, haciendo hincapié en que la API fuera completamente independiente de la vista para poder ser utilizada por cualquier interfaz gráfica de usuario. Asimismo también se ha detallado la descomposición en módulos del controlador y la vista.

Las tecnologías utilizadas han sido una base de datos relacional en PostgreSQL para el modelo de datos, PHP y JSON en la API de acceso a la base de datos y el conjunto de lenguajes web PHP, HTML, CSS y Javascript –con la tecnología AJAX- para generar páginas dinámicas del lado de servidor, que constituye la vista. Para la comunicación entre la vista y la API se ha utilizado el protocolo HTTP.

En la implementación primero se han considerado las tecnologías propuestas al inicio del proyecto y se han descrito los detalles de implementación tenidos en cuenta en la toma de decisiones al respecto. En la implementación de la API se ha puesto el mayor esfuerzo de desarrollo por constituir la componente más significativa del sistema, donde ha sido necesario un exhaustivo procesamiento de los parámetros de entrada de las funciones. Aquí también se ha adaptado la tecnología utilizada –PHP- a la decisión de diseño propuesta para que los procedimientos que componen la API sean ejecutables remotamente desde cualquier vista. En la parte de la interfaz de usuario se ha optado por desarrollar una interfaz sencilla que permitiera validar el funcionamiento global del proyecto.

En la fase de pruebas se ha procedido a validar el correcto funcionamiento del sistema y de cada componente por separado. En la base de datos ha sido probado el cumplimiento de las restricciones fijadas y la correcta ejecución de los procedimientos. En la API se ha probado cada función haciendo énfasis en el control de todos los flujos posibles. Por último en la interfaz gráfica de usuario se ha comprobado el funcionamiento de todas las páginas y formularios junto con pruebas de integración de todos los componentes del sistema.

Tras el proceso de pruebas podemos concluir que el sistema implementado cumple con todos los requisitos expuestos. Los usuarios son capaces de interactuar con las publicaciones, propias y de otros usuarios, y realizar todas las acciones detalladas en los requerimientos. La estructura de red social también está reflejada en la aplicación, donde los usuarios pueden seguir y dejar de seguir a otros y visualizar su contenido publicado.

Mi experiencia con la realización de este proyecto ha sido completamente positiva, pues me ha permitido crear una aplicación web escalable y lista para su puesta en producción en Internet y finalizar con buenas sensaciones de garantía de un éxito futuro entre los usuarios. Además en este proyecto he tenido la oportunidad de reforzar conceptos relacionados con la programación web en el lado del servidor y sobre todo con PHP, así como descubrir los beneficios y consecuencias aportados por este lenguaje.

Los aspectos más desafiantes los he encontrado en la fase de diseño, donde ha sido necesario analizar con detalle las distintas soluciones posibles para poder crear un sistema fácilmente escalable. Además esta fase ha sido la más importante del proyecto, si bien es en este período donde se debe ampliar el horizonte y donde se define la calidad del software.

Por otro lado las tareas más laboriosas han sido las relacionadas con la fase de pruebas, en particular al validar el correcto funcionamiento de funciones como la de búsquedas de usuarios y recetas para poder proveer de un filtrado completo y consistente.

Como trabajo futuro hemos identificado las siguientes posibles líneas de mejora de la aplicación:

Referencias: se puede valorar la posibilidad de que el usuario pueda incluir en una receta referencias o vínculos a otras recetas de la aplicación, con el fin de incluirlas como un ingrediente más o por ejemplo para incluir casamiento entre bebidas y comidas –maridaje-

Publicaciones conjuntas: en muchas ocasiones son varias personas las que participan en el proceso de realización de cualquier receta, por lo que no estaría de más dar la posibilidad de creación de borradores compartidos por varios usuarios.

Notificaciones y Eventos: en sistemas de este tipo siempre resulta interesante introducir un sistema de notificaciones que adviertan al usuario de cierta información o de cierto evento de la aplicación que lo involucre directamente.

Imágenes: la primera mejora que podemos tener en cuenta es la introducción de imágenes, tanto para establecerlas como foto de perfil de usuario como para incluirlas en cada receta que sea publicada, para así poder visualizar imágenes del proceso y del resultado final y contribuir a que los usuarios puedan tener una mejor crítica de la publicación.

Intentos: la introducción de Intentos en la red social puede resultar no menos interesante. Se trataría de un apartado más de cada publicación donde los usuarios pueden declarar que han intentado realizar la receta en cuestión por medio de imágenes que reflejen el resultado final.

Retos: los retos constituirían pequeños desafíos o concursos en los que se pediría participar con una receta con determinadas restricciones en cuanto a ingredientes o proceso de realización, por ejemplo. Esto podría contribuir a la diversidad de recetas de este recetario en red.

Rangos: la inclusión de retos en la red social puede despertar el espíritu competitivo entre los usuarios. Para intensificar este efecto en el sistema podríamos asociar un rango de cocinero y otro de coctelero a cada usuario. Basado en un sistema de puntos de experiencia, el rango aumentaría en función de la valoración de las publicaciones de cada usuario y por supuesto se vería influenciado también por la participación en los retos mencionados.

Categorías: otra funcionalidad interesante a añadir al software es la posibilidad de que el usuario sea capaz de categorizar las recetas de su catálogo personal, con el fin de que la consulta de recetas en el mismo, tanto por él como por los demás usuarios, pueda estar más organizada y dirigida.

La gran mayoría de los posibles requisitos futuros aquí expuestos fueron identificados en una primera versión del catálogo de requisitos. Sin embargo finalmente fueron descartados para evitar una extensión excesiva del proyecto.

Referencias

[1] Tía Alia, C. (2015). *El boom de la cocina en España y el papel de los medios de comunicación*.
http://www.directoalpaladar.com/cultura-gastronomica/el-boom-de-la-cocina-en-espana-y-el-papel-de-los-medios-de-comunicacion [Consulta: 1 de Julio de 2015].

[2] Bugueño, P. (2009). *Clasificación de redes sociales.*
http://www.pabloburgueno.com/2009/03/clasificacion-de-redes-sociales/ [Consulta: 1 de Julio de 2015].

[3] Finley, K. (2011). *How Twitter uses NoSQL*.
http://readwrite.com/2011/01/02/how-twitter-uses-nosql [Consulta: 1 de Julio de 2015]

[4] Lavezzo, N. (2013). *What are the limitations of NoSql databases?*
http://www.quora.com/What-are-the-limitations-of-NoSql-databases [Consulta: 1 de Julio de 2015].

[5] Bhattacharjee, A. (2014). *NoSQL vs SQL – Which is a Better Option?*
https://blog.udemy.com/nosql-vs-sql-2/ [Consulta: 1 de Julio de 2015].

[6] Wikipedia. *Middleware*.
https://es.wikipedia.org/wiki/Middleware [Consulta: 1 de Julio de 2015].

[7] Amode, E. (2010). *Servicios web (2): ¿Qué es REST?*
https://eamodeorubio.wordpress.com/2010/07/26/servicios-web-2-%C2%BFque-es-rest/
[Consulta: 1 de Julio de 2015].

[8] Wikipedia. *Remote Procedure Call*.
https://es.wikipedia.org/wiki/Remote_Procedure_Call
[Consulta: 1 de Julio de 2015].

[9] Oracle. *Remote Method Invocation Home*.
http://www.oracle.com/technetwork/articles/javaee/index-jsp-136424.html
[Consulta: 1 de Julio de 2015]

[10] Shekhar, Himanshu. *RPC Vs RMI*.
http://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=7&ved=0CFQQFjAG&url=http%3A%2F%2Fitgs.tistory.com%2Fattachment%2F1406296529.pdf&ei=lU-WVYSwFoqxUaDahLgI&usg=AFQjCNGCQa-IHIRyShPWXiQLQ0dv6sHO0A&bvm=bv.96952980,d.d24&cad=rja
[Consulta: 1 de Julio de 2015]

Anexos

Anexo A: Descripción de las funciones de la API

En este anexo serán descritas en detalle todas las funciones de la API mostradas en la fase de diseño del software.

A.1 Subsistema Usuarios

- nuevo_perfil
  - Descripción: Permite crear un nuevo perfil de usuario.
  - Entrada
    - nombre
    - nick
    - contraseña
    - correo electrónico
  - Salida
    - Código de error
  - Requisitos: U1, U2
- editar_perfil
  - Descripción: Permite editar algunos campos del perfil de usuario.
  - Entrada
    - nombre
    - contraseña
    - país
    - edad
  - Salida
    - Código de error
  - Requisitos: U3

- login
  - Descripción: permite iniciar sesión en el sistema
  - Entrada
    - nick/correo-electrónico
    - contraseña
  - Salida
    - JSON con el perfil del usuario
  - Requisitos: U4
- obtener_catalogo
  - Descripción: permite obtener el catálogo del perfil de un usuario
  - Entrada
    - usuario
  - Salida
    - JSON con el catálogo del usuario
  - Requisitos: U5
- obtener_tablón
  - Descripción: permite obtener el tablón del perfil de un usuario
  - Entrada
    - usuario
  - Salida
    - JSON con el tablón del usuario
  - Requisitos: U6

- obtener_borradores
    - Descripción: permite obtener la lista de borradores creados por un usuario.
    - Entrada
        - usuario (sesión)
    - Salida
        - JSON con la lista de borradores del usuario
    - Requisitos: U7
- eliminar_borrador
    - Descripción: permite eliminar un borrador de la lista de borradores.
    - Entrada
        - id_receta
        - usuario (sesión)
    - Salida: Código de error
    - Requisitos: U7
- seguir
    - Descripción: permite a un usuario seguir a otro usuario
    - Entrada
        - usuario1
        - usuario2
    - Salida
        - Código de error
    - Requisitos: U8

- dejar_de_seguir
  - Descripción: permite a un usuario dejar de seguir a otro usuario
  - Entrada
    - usuario1
    - usuario2
  - Salida
    - Código de error
  - Requisitos: U8
- obtener_seguidores
  - Descripción: permite obtener la lista de seguidores de un usuario
  - Entrada
    - usuario
  - Salida
    - JSON con la lista de seguidores del usuario
  - Requisitos: U9
- obtener_siguiendo
  - Descripción: permite obtener la lista de usuarios a los que sigue un usuario
  - Entrada
    - usuario
  - Salida
    - JSON con la lista de usuarios a los que sigue el usuario
  - Requisitos: U9

- obtener_lista_principal
  - Descripción: permite obtener la lista principal del perfil de un usuario
  - Entrada
    - usuario (sesión)
  - Salida
    - JSON con la lista principal del usuario
  - Requisitos: U10, U11
- obtener_informacion_perfil
  - Descripción: permite obtener la información del perfil de un usuario.
  - Entrada
    - usuario
  - Salida
    - JSON con la información del usuario
  - Requisitos: U12
- es_seguido_por
  - Descripción: permite saber si un usuario es seguidor de otro
  - Entrada
    - usuario1
    - usuario2
  - Salida
    - Código de error
  - Requisitos: U8

- buscar_usuario
  - Descripción: permite realizar una búsqueda de usuarios del sistema.
  - Entrada
    - usuario
    - seguidores/siguiendo/general
    - nombre
    - nick
    - país
  - Salida
    - JSON con la lista de resultados
  - Requisitos: U13
- buscar_publicacion
  - Descripción: permite realizar una búsqueda de recetas del sistema.
  - Entrada
    - usuario
    - catálogo/tablón/general
    - nombre
    - ingredientes
    - etiquetas
    - valoración promedio
    - autor
  - Salida
    - JSON con la lista de resultados
  - Requisitos: U14

- obtener_top_recetas
    - Descripción: permite obtener un ranking de las recetas mejores valoradas del sistema.
    - Entrada
        - (ninguna)
    - Salida
        - JSON con el ranking resultado
    - Requisitos: U15
- borrar_perfil
    - Descripción: permite eliminar un usuario del sistema.
    - Entrada
        - usuario
    - Salida
        - JSON con el ranking resultado
    - Requisitos: módulo de pruebas.

A.2 Subsistema Publicaciones

- editar_receta
    - Descripción: permite crear un borrador o editar uno ya existente. En caso de edición, cualquiera de los campos podrá estar vacío a excepción de los obligatorios para la edición, que serán *id_receta*, *usuario_creador* y *estado*.
    - Entrada
        - id_receta (si es nulo, creará el borrador)
        - usuario_creador
        - estado (1 → publicar receta, 0 → guardar borrador)
        - nombre_receta
        - tipo (plato/cóctel)
        - fecha
        - ingredientes
        - pasos
        - etiquetas
    - Salida
        - Código de error
    - Requisitos: P1, P2, P3, P7, P8, P9, P10, P11
- valorar_publicacion
    - Descripción: permite a un usuario valorar una receta.
    - Entrada
        - usuario
        - id_receta
        - valoración
    - Salida
        - Código de error
    - Requisitos: P4

- añadir_comentario
    - Descripción: permite a un usuario añadir un comentario a una receta.
    - Entrada
        - usuario
        - id_receta
        - texto
    - Salida
        - Código de error
    - Requisitos: P5
- añadir_sugerencia
    - Descripción: permite a un usuario añadir una sugerencia a una receta.
    - Entrada
        - usuario
        - id_receta
        - texto
        - usuario (sesión)
    - Salida
        - Código de error
    - Requisitos: P5
- catalogar
    - Descripción: permite a un usuario añadir una receta a su catálogo personal
    - Entrada
        - id_receta
        - usuario (sesión)
    - Salida
        - Código de error
    - Requisitos: P6

- descatalogar
    - Descripción: permite a un usuario eliminar una receta de su catálogo personal
    - Entrada
        - id_receta
        - usuario (sesión)
    - Salida
        - Código de error
    - Requisitos: P6
- obtener_informacion_receta
    - Descripción: permite obtener la información básica de una publicación
    - Entrada
        - id_receta
    - Salida
        - JSON con el nombre de la publicación
    - Requisitos: P7
- obtener_ingredientes
    - Descripción: permite obtener la lista de ingredientes de una publicación
    - Entrada
        - id_receta
    - Salida
        - JSON con los ingredientes de la publicación
    - Requisitos: P7

- obtener_pasos
    - Descripción: permite obtener la lista de pasos que componen el procedimiento asociado a una publicación.
    - Entrada
        - id_receta
    - Salida
        - JSON con los pasos de la publicación
    - Requisitos: P7
- obtener_comentarios
    - Descripción: permite obtener la lista de comentarios asociados a una publicación.
    - Entrada
        - id_receta
    - Salida
        - JSON con los comentarios de la publicación
    - Requisitos: P7
- obtener_sugerencias
    - Descripción: permite obtener la lista de sugerencias asociados a una publicación.
    - Entrada
        - id_receta
    - Salida
        - JSON con las sugerencias de la publicación
    - Requisitos: P7

- obtener_etiquetas
  - Descripción: permite obtener la lista de etiquetas asociadas a una publicación
  - Entrada
    - id_receta
  - Salida
    - JSON con las etiquetas de la publicación
  - Requisitos: P4
- obtener_valoracion_realizada
  - Descripción: permite obtener la valoración realizada por un usuario a una publicación.
  - Entrada
    - id_receta
    - usuario
  - Salida
    - Valoración realizada por el usuario o código de error
  - Requisitos: P7
- obtener_ha_valorado
  - Descripción: permite saber si un usuario ha valorado o no una publicación.
  - Entrada
    - id_receta
    - usuario
  - Salida
    - 1 → está valorada por el usuario
    - 0 → el usuario no ha valorado
    - código de error
  - Requisitos: P4

- obtener_ha_catalogado
    - Descripción: permite saber si un usuario ha añadido a su catálogo una publicación
    - Entrada
        - id_receta
        - usuario
    - Salida
        - 1 → está valorada por el usuario
        - 0 → el usuario no ha valorado
        - código de error
    - Requisitos: P6
- obtener_informacion_borrador
    - Descripción: permite obtener la información asociada a un borrador.
    - Entrada
        - id_receta
    - Salida
        - JSON con toda la información del borrador.
    - Requisitos: P9
- eliminar_borrador
    - Descripción: permite eliminar un borrador.
    - Entrada
        - id_borrador
    - Salida
        - Código de error.
    - Requisitos: P12

- eliminar_publicacion
    - Descripción: permite eliminar una publicación.
    - Entrada
        - id_receta
    - Salida
        - Código de error.
    - Requisitos: fase de pruebas

Anexo B: Detalle de los casos de uso

B.1 Usuario

Caso de uso Nº1: Edición del perfil de usuario

Actor primario: usuario de la aplicación.

Interesados y objetivos: el usuario que desea modificar los parámetros de su perfil.

Precondiciones: el usuario debe estar registrado previamente en el sistema y debe haber iniciado sesión en el mismo.

Garantía de éxito (Postcondiciones): el perfil del usuario queda modificado correctamente y así lo debe poder comprobar el usuario de manera inmediata.

Escenario principal de éxito:

1. El usuario navega al apartado de edición de su perfil.
2. Rellena los campos que desea modificar y acepta los cambios.
3. El sistema debe comprobar que los campos introducidos son correctos.
4. Si son correctos, el sistema deberá actualizar el perfil del usuario y reflejar los cambios en la interfaz.

Extensiones (flujos alternativos):

3.a Los datos introducidos por el usuario no son correctos.

3.b.1 En tal caso el sistema debe informar al usuario debidamente.

Lista de variaciones de tecnología y datos:

- Si se amplía el perfil del usuario con más información, se podrán ampliar los campos que puedan ser modificados.

Frecuencia de ocurrencia: baja. Por ahora el usuario solo podrá modificar su nombre, contraseña, edad y país por lo que se supone que son datos que no se cambian constantemente.

Temas abiertos: no hay temas abiertos.

Caso de uso Nº2: seguir

Actor primario: usuario de la aplicación.

Interesados y objetivos: el usuario u1 que desea comenzar a seguir la actividad de otro usuario u2.

Precondiciones: ambos usuarios deben estar registrados previamente en el sistema. El usuario u1 debe haber iniciado sesión en el mismo. No debe existir previamente la relación de seguimiento que se pretende crear.

Garantía de éxito (Postcondiciones): el usuario u2 aparece de manera inmediata en la lista de *siguiendo* del usuario u1, así como el usuario u1 aparecerá de manera recíproca en la lista de *seguidores* de u2. Las publicaciones que realice a partir de ese momento el usuario u2 aparecerán en la lista principal de u1.

Escenario principal de éxito:

1. El usuario u1 visita el perfil del usuario u2.
2. Ejecuta la operación de seguimiento sobre su perfil.
3. El sistema actualiza debidamente los las listas de los perfiles implicados.

Extensiones (flujos alternativos): no tiene flujos alternativos.

Lista de variaciones de tecnología y datos:

- Podemos ampliar las propiedades del perfil de usuario de manera que pueda privatizarse para que sea visible solo ante ciertos usuarios. Con estas condiciones la relación de seguimiento puede no ser posible.

Frecuencia de ocurrencia: muy alta. La creación de relaciones entre usuarios es una de las operaciones que típicamente se realizan con más frecuencia en un sistema de red social.

Temas abiertos: no hay temas abiertos.

Caso de uso Nº3: buscar usuario/receta

Actor primario: usuario de la aplicación.

Interesados y objetivos: el usuario que desea realizar una búsqueda en el sistema.

Precondiciones: el usuario debe haber iniciado sesión previamente en el sistema.

Garantía de éxito (Postcondiciones): el usuario puede visualizar los resultados de la búsqueda correctamente, si los hubiera. En caso contrario, el sistema informará de ello debidamente.

Escenario principal de éxito:

1. El usuario navega al apartado de búsqueda de otros usuarios o recetas.
2. Introduce los parámetros de búsqueda.
3. El sistema comprueba que los parámetros introducidos son correctos.
4. El sistema muestra la lista de resultados de la búsqueda si los hubiera.
5. El usuario puede acceder al detalle de cada elemento de la lista de resultados.

Extensiones (flujos alternativos):

3.a Algunos de los parámetros de búsqueda son incorrectos.
3.a.1 El sistema informará al usuario.
4.a El sistema no encuentra coincidencias con la búsqueda deseada.
4.a.1 En tal caso el sistema informará al usuario de que no se han encontrado resultados.

Lista de variaciones de tecnología y datos:

- Podemos ampliar la variedad de los parámetros de búsqueda para obtener mayor posibilidad de filtrado.

Frecuencia de ocurrencia: muy alta. Un sistema de este tipo tiene como objetivo principal constituir un recetario gastronómico, por lo que la búsqueda de usuarios y sobre todo de recetas se convertirá en una funcionalidad muy solicitada en la aplicación.

Temas abiertos: no hay temas abiertos.

B.2 Publicación

Caso de uso Nº4: crear y guardar un borrador

Actor primario: usuario de la aplicación.

Interesados y objetivos: el usuario que pretende crear un borrador de receta.

Precondiciones: el usuario debe haber iniciado sesión previamente en el sistema.

Garantía de éxito (Postcondiciones): los datos de la receta modificados se guardan correctamente en el borrador y puede acceder posteriormente a él para editarlo o descartarlo.

Escenario principal de éxito:

1. El usuario inicia la creación de un borrador de receta.
2. Introduce los datos de la receta que desee guardar en el borrador.
3. El sistema comprueba que los parámetros introducidos son correctos.
4. El sistema notifica al usuario de que se ha guardado correctamente.
5. El nuevo borrador aparece en la lista de borradores del usuario para poder ser editado o eliminado de la lista.

Extensiones (flujos alternativos):

3.a Algunos de los parámetros del borrador son incorrectos.
    3.a.1 El sistema informará al usuario debidamente y el borrador no se guardará.

Lista de variaciones de tecnología y datos:

- Se puede ampliar el proceso de edición del borrador añadiendo los campos correspondientes a los requisitos futuros mencionados, para lo cual el proceso de modificación y validación del borrador se vería alterado.

Frecuencia de ocurrencia: alta. Esta es una de las funcionalidades principales de la aplicación.

Temas abiertos: no hay temas abiertos.

Caso de uso Nº5: publicar un borrador

Actor primario: usuario de la aplicación.

Interesados y objetivos: el usuario que pretende publicar una receta.

Precondiciones: el usuario debe haber creado un borrador y haber rellenado todos sus campos.

Garantía de éxito (Postcondiciones): la receta se publica correctamente y así debe indicarlo el sistema. La nueva receta se añade al tablón del usuario y a las listas principales de los usuarios que siguen al autor de la receta.

Escenario principal de éxito:

1. El usuario rellena todos los campos del borrador y selecciona la opción asociada a la publicación del mismo.
2. El sistema comprueba que todos los campos del borrador están completos y son correctos.
3. El sistema notifica al usuario que se ha publicado correctamente.
4. La nueva publicación se hace visible en el tablón del autor y en las listas mencionadas.

Extensiones (flujos alternativos):

2.a Algunos de los parámetros del borrador son incorrectos.
    2.a.1 El sistema informará al usuario debidamente y el borrador no será publicado.

Lista de variaciones de tecnología y datos:

- Al igual que en el caso de uso número 5, se puede ampliar el proceso de edición del borrador añadiendo los campos correspondientes a los requisitos futuros mencionados.

Frecuencia de ocurrencia: alta. Esta es una de las funcionalidades principales de la aplicación.

Temas abiertos: no hay temas abiertos.

Caso de uso Nº6: valorar publicación

Actor primario: usuario de la aplicación.

Interesados y objetivos: el usuario que pretende valorar una publicación.

Precondiciones: el usuario debe haber iniciado sesión previamente en el sistema y debe haber seleccionado la publicación en cuestión. Asimismo el usuario no debe haber valorado con anterioridad dicha publicación, o bien no debe ser autor de la misma.

Garantía de éxito (Postcondiciones): la votación se realiza correctamente y se actualiza tanto la valoración media de la receta y el número de valoraciones como la valoración media del usuario autor de la receta.

Escenario principal de éxito:

1. El usuario selecciona una publicación para visualizarla en detalle.
2. El usuario introduce el valor con el que votar entre 1 y 5.
3. El usuario selecciona la opción de "Valorar".
4. El sistema comprueba que los parámetros introducidos son correctos.
5. El sistema notifica al usuario de que su valoración se ha efectuado correctamente.
6. El sistema refresca la pantalla de la publicación en detalle y muestra los datos actualizados.
7. El sistema indicará explícitamente al usuario que ya ha valorado dicha publicación.

Extensiones (flujos alternativos):

3.a El usuario no selecciona la opción de "Valorar".

3.a.1 El sistema en tal caso no actualiza la publicación.

4.a Algunos de los parámetros de la valoración son incorrectos.

4.a.1 El sistema informará al usuario debidamente.

Lista de variaciones de tecnología y datos:

- Se puede observar la posibilidad de poder variar el rango de valores de la votación, así como poder valorar no solo la receta de manera global sino cada componente en particular.

Frecuencia de ocurrencia: alta. La valoración de las recetas es una de las pocas formas de interacción de que dispondrá un usuario con una publicación ajena, además no requiere de rellenar extensos formularios, por lo que se espera que sea realizada con alta frecuencia.

Temas abiertos: no hay temas abiertos.

Caso de uso Nº7: comentar publicación

Actor primario: usuario de la aplicación.

Interesados y objetivos: el usuario que pretende añadir un comentario a una publicación.

Precondiciones: el usuario debe haber iniciado sesión previamente en el sistema y debe haber seleccionado la publicación en cuestión.

Garantía de éxito (Postcondiciones): la agregación del comentario se realiza correctamente y el usuario puede visualizar su comentario en la lista de comentarios de forma inmediata.

Escenario principal de éxito:

1. El usuario selecciona una publicación para visualizarla en detalle.
2. El usuario introduce el texto asociado al comentario en el apartado de *comentarios* de la publicación.
3. El usuario selecciona la opción de "Añadir Comentario".
4. El sistema comprueba que el texto introducido es correctos.
5. El sistema notifica al usuario de que se ha añadido el comentario correctamente.
6. El sistema refresca la pantalla de la publicación en detalle y muestra los datos actualizados.
7. El usuario puede ver en la lista de comentarios aquél que acabara de añadir.

Extensiones (flujos alternativos):

3.a El usuario no selecciona la opción asociada para confirmar la acción.

3.a.1 El sistema no actualiza la lista de comentarios.

4.a El texto asociado al comentario no es válido.

4.a.1 El sistema informará al usuario debidamente.

Lista de variaciones de tecnología y datos:

- Para validar el comentario se podría incorporar un procesador de texto que detectará lenguaje abusivo o comentarios obscenos.

Frecuencia de ocurrencia: media-alta. Aunque los usuarios visualicen la receta en detalle, no todos de modo general se detendrán a añadir comentarios o sugerencias a la receta, por lo que podemos estimar a priori que esto ocurrirá con frecuencia media.

Temas abiertos: no hay temas abiertos.

Caso de uso Nº8: catalogar publicación

Actor primario: usuario de la aplicación.

Interesados y objetivos: el usuario que pretende añadir una publicación su catálogo personal.

Precondiciones: el usuario debe haber iniciado sesión previamente en el sistema y debe haber seleccionado la publicación en cuestión. Dicha publicación no debe pertenecer en el momento actual al catálogo del usuario.

Garantía de éxito (Postcondiciones): la receta se ha añadido satisfactoriamente al catálogo personal del usuario y así puede verificarlo. El usuario será a capaz de acceder al detalle de la receta desde su catálogo.

Escenario principal de éxito:

1. El usuario selecciona una publicación para visualizarla en detalle.
2. El usuario selecciona la opción de "Añadir a Mi Catálogo".
3. El sistema notifica al usuario de que se ha añadido a su catálogo correctamente.
4. El sistema refresca la pantalla de la publicación en detalle y muestra los datos actualizados junto con la evidencia de que la publicación ya ha sido agregada a su catálogo.
5. El usuario puede visualizar en su catálogo personal la publicación recientemente añadida.

Extensiones (flujos alternativos):

2.a El usuario ya ha catalogado la publicación anteriormente.

2.a.1 El sistema informará al usuario debidamente y en .este caso se le prestará la opción de "Quitar del Catálogo".

Lista de variaciones de tecnología y datos:

- Se podría sopesar la opción de añadir una marca a una receta, y asimismo poder clasificar y filtrar el catálogo por marcas establecidas.

Frecuencia de ocurrencia: alta. Esta funcionalidad permite a un usuario guardar una receta cualquiera en una lista personal, para tenerla cerca y no tener que realizar la búsqueda de nuevo. Esto convierte a esta funcionalidad en imprescindible y por tanto podemos estimar que será utilizada muy frecuentemente.

Temas abiertos: no hay temas abiertos.