Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías de Telecomunicación

Aplicación web colaborativa de servicios de comidas caseras

Autor: Mario Martínez García

Tutor: Juan Manuel Vozmediano Torres

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2015

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías de Telecomunicación

Aplicación web colaborativa de servicios de comidas caseras

Autor:

Mario Martínez García

Tutor:

Juan Manuel Vozmediano Torres

Profesor titular

Departamento de Ingeniería Telemática

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2015

Proyecto Fin de Carrera: Aplicación web colaborativa de servicios de comidas caseras

Autor: Mario Martínez García

Tutor: Juan Manuel Vozmediano Torres

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2015

El Secretario del Tribunal

Resumen

El tiempo es el recurso más preciado que tenemos en nuestras vidas.

La buena alimentación de una persona, se hace necesaria en contraposición a la llamada comida rápida, para prevenir en un futuro enfermedades de todo tipo.

Cocinar bien, es decir, cocinar comida sana, require de tiempo y de ciertas habilidades. No todo el mundo tiene estas habilidades, ni tampoco a todo el mundo le gusta cocinar, pero lo que sí está claro es que todo el mundo necesita comer.

Y comer todos los días en restaurante, no está al alcance de todos los bolsillos.

Así que, ¿por qué no poner en contacto a personas que sí saben y quieren cocinar, con personas que no tengan tiempo y/o no quieran cocinar?

Es este proyecto hemos desarrollado una aplicación web colaborativa de servicios de comidas caseras donde ponemos en contacto a usuarios cocineros con usuarios comensales.

Abstract

Time is the most precious resource we have in our lives.

Good nutrition of a person, is necessary as opposed to the so-called fast food in the future to prevent diseases of all kind.

While cooking, ie cooking healthy food, time and certain skills are required. Not everyone has these skills, nor everybody likes to cook, but what is clear is that everyone needs to eat.

And eating in the restaurant every day, it is not within reach of all budgets.

So, why not to put people in touch who do know and want to cook, to people who do not have time and / or do not want to cook?

In this project we have developed a homemade meals collaborative web application where we put the chef users in touch with the client users.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

1 INTRODUCCIÓN

Nunca sabes quién está nadando desnudo hasta que baja la marea.

- Warren Buffet -

1.1 Presentación

Cookify es una plataforma colaborativa web que pone en contacto a usuarios para la compra venta de platos cocinados por cuenta propia y envasados en tupperware.

1.2 Objectivos del proyecto

Actualmente, el mercado laboral es mucho más competitivo en que décadas pasadas, en parte debido a la globalización y a la alta cualificación de las personas.

Esto se traduce, que en la mayoría de los casos, tenemos que trabajar más para tener el mismo poder adquisito que antes. Por lo que tenemos menos tiempo para invertir en nuestras actividades diarias, y cocinar probablemente sea una de ellas.

Podemos ir a un restaurante o pedir comida para llevar, y así nos evitamos tener que cocinar, pero mantener este hábito regularmente no está al alcance de todos los bolsillos.

Cocinar bien, es decir, comida sana, no es algo sencillo, porque requiere tiempo y habilidades, y quizás, ni nos guste cocinar, ni tengamos suficiente tiempo para ello o queramos invertir dicho tiempo en otras tareas más satisfactorias para nosotros. Es por ello que muchas personas acuden a la comida rápida, adquiriendo un hábito no saludable que se traduce a largo plazo en enfermedades de cualquier índole.

En las familias, tradicionalmente solo trabaja un cónyuge, mientras que el otro se dedica a hacer las tareas de la casa, como cocinar. Pero cada vez más, debido a la competitividad del mercado laboral, se hace muy necesario que los dos miembros de una pareja estén trabajando para obtener ingresos. Siendo así más difícil tener tiempo para las tareas de la casa.

Toda esta situación, con la llegada de las nuevas tecnologías que tenemos a nuestro alcance, se plantea retos que podrían solventar algunos de los problemas actuales.

Este tipo de aplicaciones se denominan plataformas colaborativas, porque su objetivo es poner en contacto a usuarios para compartir recursos.

Teniendo en cuenta en planteamiento inicial, nos preguntamos que por qué no una plataforma colaborativa que pongan en contacto a personas que saben y quieren cocinar con personas que no tengan tiempo ni quieran cocinar. Por ejemplo, si un amigo o vecino, cocinan y sacan muchas raciones de comida, no veo por qué no podría pedirle una de esas raciones a cambio de hacerle el mismo favor otro dia yo a él.

De este planteamiento nace este proyecto, y le hemos llamado Cookify.

1.3 Motivación personal

Cookify está diseñado y desarrollado con las tecnologías más recientes del mercado y que están teniendo gran aceptación por parte de la comunidad. Por estas razones, he utilizado dichas tenologías para llevar a cabo este proyecto.

1.4 Esquema de la documentación

En este apartado vamos a comentar brevemente en qué consiste cada apartado, para dar un breve avance del contenido de este proyecto.

Bloque 1 Introducción:

Visto hasta ahora, donde hemos presentado el proyecto, sus objetivos y la motivación personal.

Bloque 2 Fundamento teórico

En esta sección se orientará al lector a los distintos conceptos que se deberán de conocer antes de empezar con el resto del documento. Asimismo, se tratará de describir brevemente la situación del contexto actual.

Bloque 3 Recursos y herramientas

En esta sección se presentarán los recursos, tanto tecnológicos como de personal, con los que se contará para llevar a cabo la realización del proyecto.

Bloque 4 Requisitos del sistema

En este apartado se describirán las funcionalidades que deberá presentar el sistema final. Dicha descripción del sistema se dividirá en objetivos, requisitos de información, requisitos funcionales y requisitos no funcionales.

Bloque 5 Análisis del sistema

En esta sección se profundizará en los detalles más importantes del sistema, tales como los tipos que modelan la situación que se desarrolla y sus interrelaciones, las funcionalidades que el sistema debe ofrecer al usuario final y las operaciones que el sistema debe realizar para poder implementar los casos de uso.

Bloque 6 Diseño del sistema

En este apartado se trazarán los medios por los cuales se llevarán a cabo las funcionalidades descritas en el apartado de análisis.

Bloque 7 Implementación

En este bloque se muestran brevemente algunas de las soluciones que se han ideado durante la etapa de implementación del proyecto.

Bloque 8 Pruebas

A lo largo de este bloque se describirán todas las pruebas que se han realizado al sistema para validar los requisitos del proyecto.

Bloque 9 Conclusiones

En este apartado se extraerán las conclusiones finales del proyecto, y la experiencia de desarrollo del mismo. Asimismo, se estudiarán las posibles mejoras que pudiesen aplicarse al sistema.

Bloque 10 Bibliografía

Por último, es en este apartado donde se exponen las fuentes consultadas a lo largo del desarrollo del mismo, necesarias para la documentación.

Bloque 11 Anexos

En los anexos añadiremos un breve manual de usuario para comprender mejor cómo utilizar la aplicación.

2 FUNDAMENTO TEÓRICO

2.1 Hypertext Transfer Protocol (Http)

Hypertext Transfer Protocol o HTTP (en español protocolo de transferencia de hipertexto) es el protocolo usado en cada transacción de la World Wide Web. HTTP fue desarrollado por el World Wide Web Consortium y la Internet Engineering Task Force, colaboración que culminó en 1999 con la publicación de una serie de RFC, el más importante de ellos es el RFC 2616 que especifica la versión 1.1. HTTP define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. Al cliente que efectúa la petición (un navegador web o un spider) se lo conoce como "user agent" (agente del usuario). A la información transmitida se la llama recurso y se la identifica mediante un localizador uniforme de recursos (URL). El resultado de la ejecución de un programa, una consulta a una base de datos, la traducción automática de un documento, etc.

HTTP es un protocolo sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores. El desarrollo de aplicaciones web necesita frecuentemente mantener estado. Para esto se usan las cookies, que es información que un servidor puede almacenar en el sistema cliente. Esto le permite a las aplicaciones web instituir la noción de "sesión", y también permite rastrear usuarios ya que las cookies pueden guardarse en el cliente por tiempo indeterminado.

2.2 Apache Http Server

El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1

El servidor Apache es desarrollado y mantenido por una comunidad de usuarios bajo la supervisión de la Apache Software Foundation dentro del proyecto HTTP Server (httpd)

2.3 Web Services

Un servicio web (en inglés, Web Service o Web services) es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de

ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios Web. Para mejorar la interoperabilidad entre distintas implementaciones de servicios Web se ha creado el organismo WS-I, encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares. Es una máquina que atiende las peticiones de los clientes web y les envía los recursos solicitados.

2.3.1 SOAP

SOAP (siglas de Simple Object Access Protocol) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva de un protocolo creado por Dave Winer en 1998, llamado XML-RPC. SOAP fue creado por Microsoft, IBM y otros. Está actualmente bajo el auspicio de la W3C. Es uno de los protocolos utilizados en los servicios Web.

2.3.2 REST

La Transferencia de Estado Representacional (Representational State Transfer) o REST es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. El término se originó en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo.

2.3 Push

La tecnología push es una forma de comunicación a través de internet en la que la petición de envío tiene origen en el servidor, por oposición a la tecnología pull, en la que la petición tiene origen en el cliente.

2.3.1 Long Polling

Long polling es una variación de la técnica tradicional de polling y permite emular información colocada desde un servidor a un cliente en forma similar al polling normal. Sin embargo, si el servidor no tiene información disponible para el cliente, en vez de enviar una respuesta vacía, el servidor guarda la petición y espera a que alguna información esté disponible. Una vez la información está disponible (o después de un tiempo establecido), se envía una respuesta completa al cliente. Entonces el cliente normalmente realizará un re-pedido de información al servidor, para que éste siempre tenga un pedido en espera, que puede ser usado para responder a un evento.

Long polling no es en sí mismo una tecnología push, pero puede ser usada bajo circunstancias donde un push verdadero no es posible.

2.4 Database

2.4.1 SQL

El lenguaje de consulta estructurado o SQL (por sus siglas en inglés Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar, de forma sencilla, información de bases de datos, así como hacer cambios en ellas.

2.4.2 NoSQL

En informática, NoSQL (a veces llamado "no sólo SQL") es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico del sistema de gestión de bases de datos relacionales (RDBMS) en aspectos importantes, el más destacado es que no usan SQL como el principal lenguaje de consultas. Los datos almacenados no requieren estructuras fijas como tablas, normalmente no soportan operaciones JOIN, ni garantizan completamente ACID (atomicidad, consistencia, aislamiento y durabilidad), y habitualmente escalan bien horizontalmente. Los sistemas NoSQL se denominan a veces "no sólo SQL" para subrayar el hecho de que también pueden soportar lenguajes de consulta de tipo SQL.

3 RECURSOS Y HERRAMIENTAS

3.1 Descripción tenológica

En esta sección vamos a diseminar las entrañas de Cookify comentando con detalles qué tecnologías se han utilizado, por qué y cómo cada componente se encuentra estructurado dentro del escenario principal.

3.1.1 Cliente

La parte del Cliente es la interfaz visual por la que el usuario interactúa con el sistema. En el caso que nos concierne, el cliente es una interfaz web.

Toda interfaz web se construye en base a tres componentes enseciales:

1- Html: Es el lenguaje de marcado que utilizan los navegadores para renderizar el resultado.

2- CSS: Hojas de estilo que se encargan del aspecto visual de los elementos codificados en html.

3- Javascript: Es un lenguaje dinámico que se encarga de las animaciones e interacciones dinámicas del usuario con la aplicación web.

A veces, a medida que la aplicación vaya aumentando en tamaño, se hace recomendable e incluso necesario utilizar algún framework para que familice el desarrollo de la misma.

La parte cliente de Cookify se ha desarrollado con un framework de javascript llamado Angular.js.

3.1.1.1 Angular.js

Angular.js fue creado por Miško Hevery en 2009, quien trabaja para Google y es mantenido por un equipo dentro de esta compañía.

1. Ilustración – Framework Angular.js

Existen otros Frameworks de javascript, como Backbone.js o Ember.js, pero se ha elegido Angular.js debido a la gran respuesta que el mercado está teniendo sobre este framework.

A continuación presentamos una gráfica donde, según el portal de empleo indeed.com, se refleja la tendencia de puestos de trabajo relacionados con estos frameworks.

2. Ilustración – Comparativa de ofertas de trabajo en Indeed.com de Angular.js, Backbone.js y Ember.js

3.1.2 Servidor web

Para el servidor web, se ha elegido Node.js por sus prestaciones que nos ofrecía. Luego hacemos una comparación con los servidores más utilizados para afianzar la importancia de este servidor elegido.

3.1.2.1 Node.js

.Node.js es un servidor web creado por Ryan Dahl y utilizado en su mayoría por el lenguaje javascript. En su página principal se describo como una plataforma de servidor para aplicaciones en tiempo real.

3. Ilustración – Node.js

Node.js usa el mismo motor de javascript que Google emplea en su navegador Chrome, de ahí parte de su relevancia.

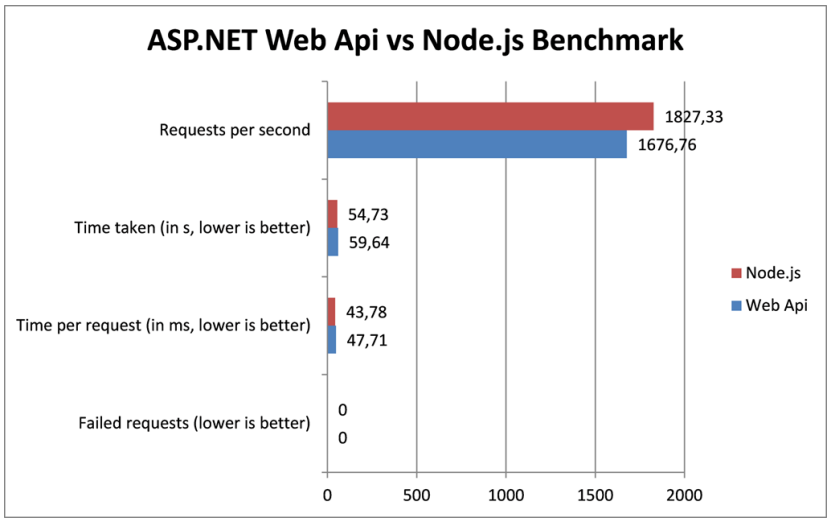
Existen varios servidores de suma importancia en el mercado pero que no hacen sombra a Node.js en

cuanto a rendimientos y características. Algunos de estos servidores son Apache o Microsoft-IIS.

Algunas de las gráficas que corroboran la importancia de Node.js las mostramos a continuación

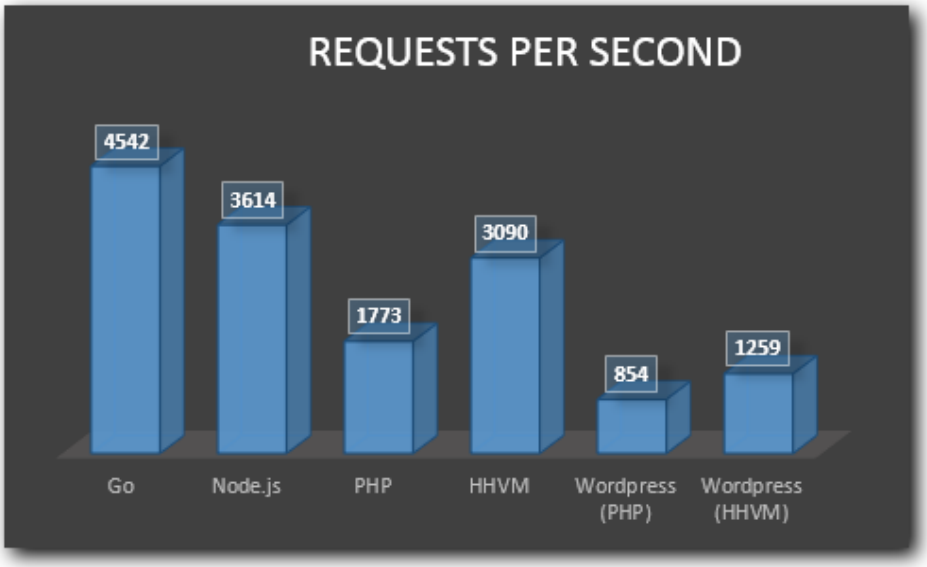
Según la página http://www.haneycodes.net/ donde hacen una prueba para comparar el número de solicitudes por segundo, comparando el servidor ISS y Node.js

Podemos encontrar más detalles en la página web http://www.haneycodes.net/to-node-js-or-not-to-node-js/.



4. Ilustración - Benchmark de los entornos ASP.GET Web API y Node.js

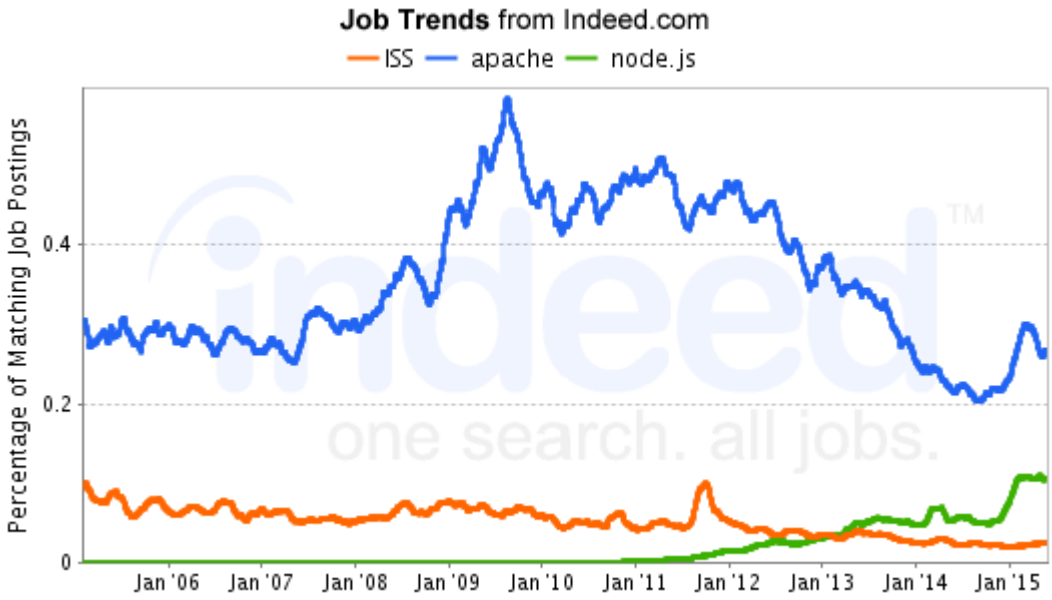
Según http://www.hostingadvice.com/ y comparando los mismos parámetros que antes pero con el servidor Apache en lugar de ISS.



5. Ilustración - Benchmark de algunos de los entornos más populares

A parte de estas dos pruebas, una consideración de gran relevancia para la toma de decisión del servidor tomado es el lenguaje en el que se desarrolla, que en Node.js es javascript, no siendo así en Apache, que es PHP y en ISS es .net. Por lo que hemos tenido en cuenta que tanto cliente como servidor se programa en javascript traduciéndose esto es una mejora de la productividad al tener que aprender solo un lenguaje de programación.

Si comprobamos su importancia comparancias sus tendencias de puestos de trabajos utilizando el portal de empleo indeed.com



6. Ilustración - Comparativa de ofertas de trabajo en Indeed.com de los servidores ISS, Apache y Node.js en términos absolutos

Hay que tener en cuenta, que Apache domina actualmente el mercado porque lleva consolidado en el mercado durante muchos años. En cambio, Node.js nació en 2009 y aún esta por consolidarse frente a Apache.

Si tenemos en cuenta los incrementos de puestos de trabajo en lugar de los puestos absolutos, tenemos esta otra gráfica.

7. Ilustración - Comparativa de ofertas de trabajo en Indeed.com de los servidores ISS, Apache y

Node.js en términos relativos

Donde mejor podemos apreciar, que es en Node.js donde más puestos de empleo se están creando relativamente.

3.1.3 Base de datos

En cuanto a la base de datos hemos optado por una base de datos NoSQL orientada a documentos porque es la que major satisfaccia los requermientos del Proyecto, tanto en redimiento como velocidad de desarrollo.

La curva de aprendizaje de mongoDB es bastante suave y ofrece grandes capacidades en cuanto a funcionalidades y grandes expectativas por parte de la comunidad

3.1.3.1 MongoDB

MongoDB fue desarrollado por MongoDB Inc. En el año 2009. Es un proyecto open source y está escrito en tres lenguajes principalmentem C, C++ y javascript.

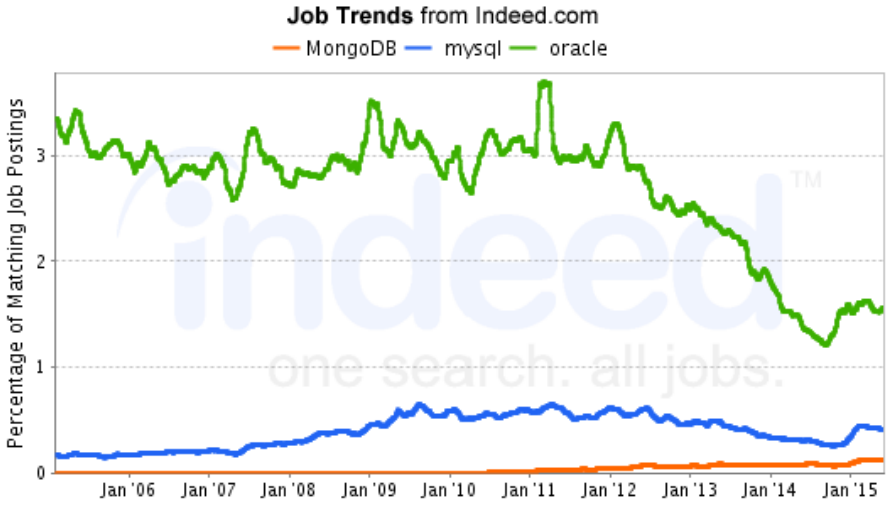
8. Ilustración – Base de datos MongoDB

Una de las diferencias principales con SQL en general, es que mongoDB ofrece grandes rendimientos para los datos basados en geolocalización.

También hemos tenido en cuenta a la hora de elegir una base de datos, es la perspectiva de futuro que tiene por delante.

Como en el caso del servidor, hemos elegido como parámetros el número de empleos laborales que requieren de estas tecnologías.

MongoDB la hemos comparado con Mysql y Oracle, que son las dos primeras que están en el ranking de http://db-engines.com/en/ranking que es un ranking muy valorado en la comunidad.



9. Ilustración - Comparativa de ofertas de trabajo en Indeed.com de las bases de datos Oracle, Mysql y MongoDB en términos absolutos

En esta gráfica se representa el número de ofertas de empleo en términos absolutos que hace referencia a dichas tecnologías.

Vemos que Oracle domina en este ámbito pero hay que recalcar que esta base de datos lleva muchos años en el mercado y lleva tiempo consolidada a su vez como también está en declive al mismo tiempo que nacen nuevas tecnologías.

Si miramos desde la relatividad obtenemos este otro gráfico.

10. Ilustración - Comparativa de ofertas de trabajo en Indeed.com de las bases de datos Oracle, Mysql y MongoDB en términos relativos

Donde podemos observar como en términos de crecimiento, mongo si está liderando el ranking.

3.2 Recursos disponibles

En esta sección vamos a comentar los recursos con los que se ha desarrollado este proyecto.

3.2.1 Recursos humanos

Este proyecto se ha ralizado por un único alumno y gracias a la ayuda del tutor por su asesoramiento en lo referente al plan de negocio.

3.2.2 Recursos hardware

Vamos a comentar los recursos hardware que se ha utilizado para el proyecto.

1. PC de sobremesa,
    a. Sistema operativo Ubuntu 14.04
    b. Procesador Intel Core i7-3770 3,4 GHz
    c. Alamecamiento 500GB
    d. RAM 16 GB
2. Conexión a internet: 10 Mb/s

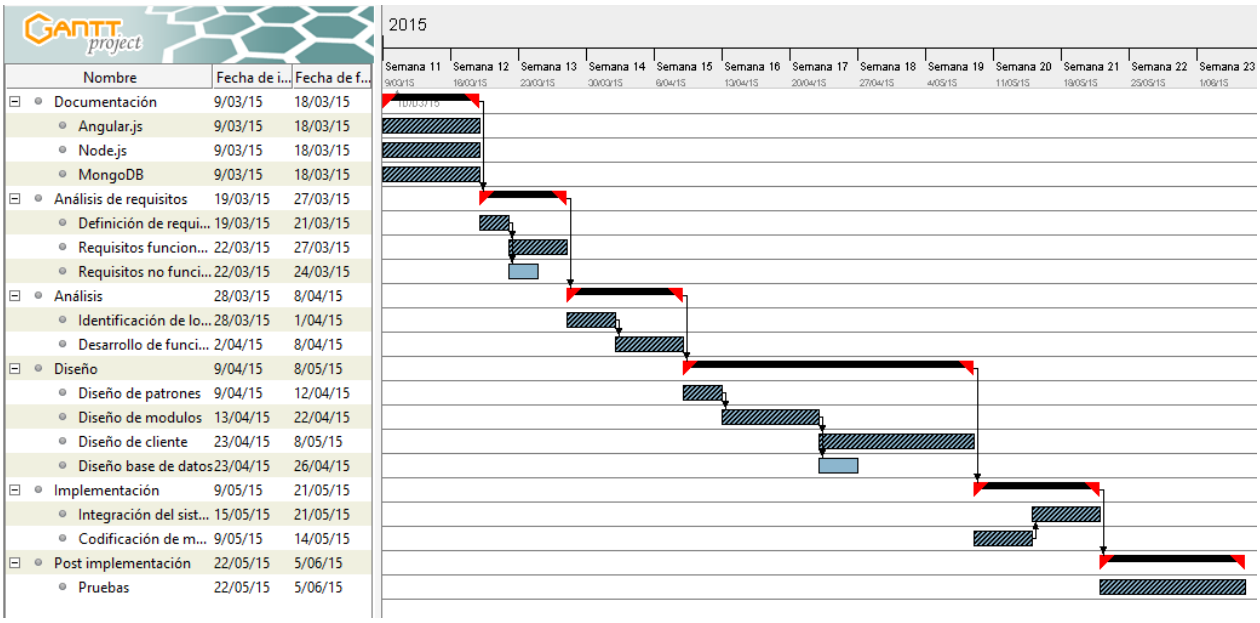
3.2.3 Recursos Software

Indicamos ahora las herramientas de software utilizadas. No son esencial pudiéndose así buscar alternativas para el desarrollo del proyecto.

1. Sistema operativo: Ubuntu 14.04
2. Herramientas de desarrollo
    a. Visual Studio Code como IDE
    b. GIT. Es un sistema que gestiona las distintas versiones de nuestro código
    c. MongoVue, que es un cliente para Windows, para interactuar con la base de datos mongoDB.

4.3 *Marco temporal*

A continuación detallamos cuál ha sido la duración de cada una de las partes del desarrollo del proyecto.



11. Ilustración – Diagrama de Gantt del tiempo de desarrollo del proyecto

Cada día se ha invertido 5 horas al desarrollo del mismo.

Presentamos a continuamos las relaciones entre cada partes y las horas equivalentes.

Documentación	**50 horas**
Angular.js	34 horas
Node.js	10 horas
MongoDB	6 horas

Análisis de requisitos	**45 horas**
Definición de requisitos	15 horas

Requisitos funcionales	26 horas
Requisitos no funcionales	4 horas

Análisis	**60 horas**
Identificación de los subsistemas	20 horas
Desarrollo de funciones	35 horas

Diseño	**135 horas**
Diseño de patrones	25 horas
Diseño de módulos	50 horas
Diseño de cliente	48 horas
Diseño de base de datos	12 horas

Implementación	**65 horas**
Integración del sistema	35 horas
Codificación de módulos	30 horas

Post Implementación	**75 horas**
Pruebas	75 horas

4 REQUISITOS DEL SISTEMA

4.1 Descripción del sistema

Cookify consiste en una aplicación web colaborativa en el campo de la hostelería donde un usuario puede elaborar platos e intercambiarlo por créditos a algún amigo o vecino.

Tiene un sistema de valoración de plato / chef basado en opiniones y puntuaciones, esto permitirá valorar los platos según su reputación y la del chef que elaboró dicho plato.

Estas valoraciones se relacionan directamente con el número de tenedores que un chef puede tener. Un cocinero puede tener desde un tenedor hasta siete tenedores.

La aplicación también ofrece la funcionalidad de poner en contacto a dos usuarios, para que la comunicación sea satisfactoria y así lograr una mejor cooperación y funcionamiento.

4.2 Objetivos del sistema

Para conseguir nuestro cometido, que es realizar con éxito nuestro proyecto, debemos definir claramente cual son los objetivos que deben cumplir dicho proyecto.

OBJ-001	Publicar platos
Descripción	Cada usuario puede crear y publicar platos de comida para que otros usuarios puedan solicitarlo.
Estado	Validado

1. Tabla – OBJ-001

OBJ-002	Obtener geolocalización del usuario
Descripción	Para que el sistema pueda ofrecer platos de comida que estñen cerca de mi posición, el sistema ha de obtener la localización exacta del usuario
Estado	Validado

2. Tabla – OBJ-002

OBJ-003	Poner en contacto a los usuarios entre sí
Descripción	Cualquier usuario puede establecer comunicación vía mensajes privados con otro usuario.
Estado	Validado

3. Tabla – OBJ-003

OBJ-004	Seguir a usuarios y platos
Descripción	Un usuario puede seguir a cualquier plato o usuario para que se le notifique cuando haya algún cambio.
Estado	Validado

4. Tabla – OBJ-004

OBJ-005	Ofrecer una interfaz de usuario sencilla y funcional
Descripción	Cada usuario puede crear y publicar platos de comida para que otros usuarios puedan solicitarlo.
Estado	Validado

5. Tabla – OBJ-005

4.3 Requisitos funcionales

Los requisitos funcionales definen el comportamiento del sistema que se está desarrollando, incluyendo los procesos fundamentales que el software llevará a cabo. La mayoría de estos requisitos funcionales provienen directamente de un requisito de usuario.

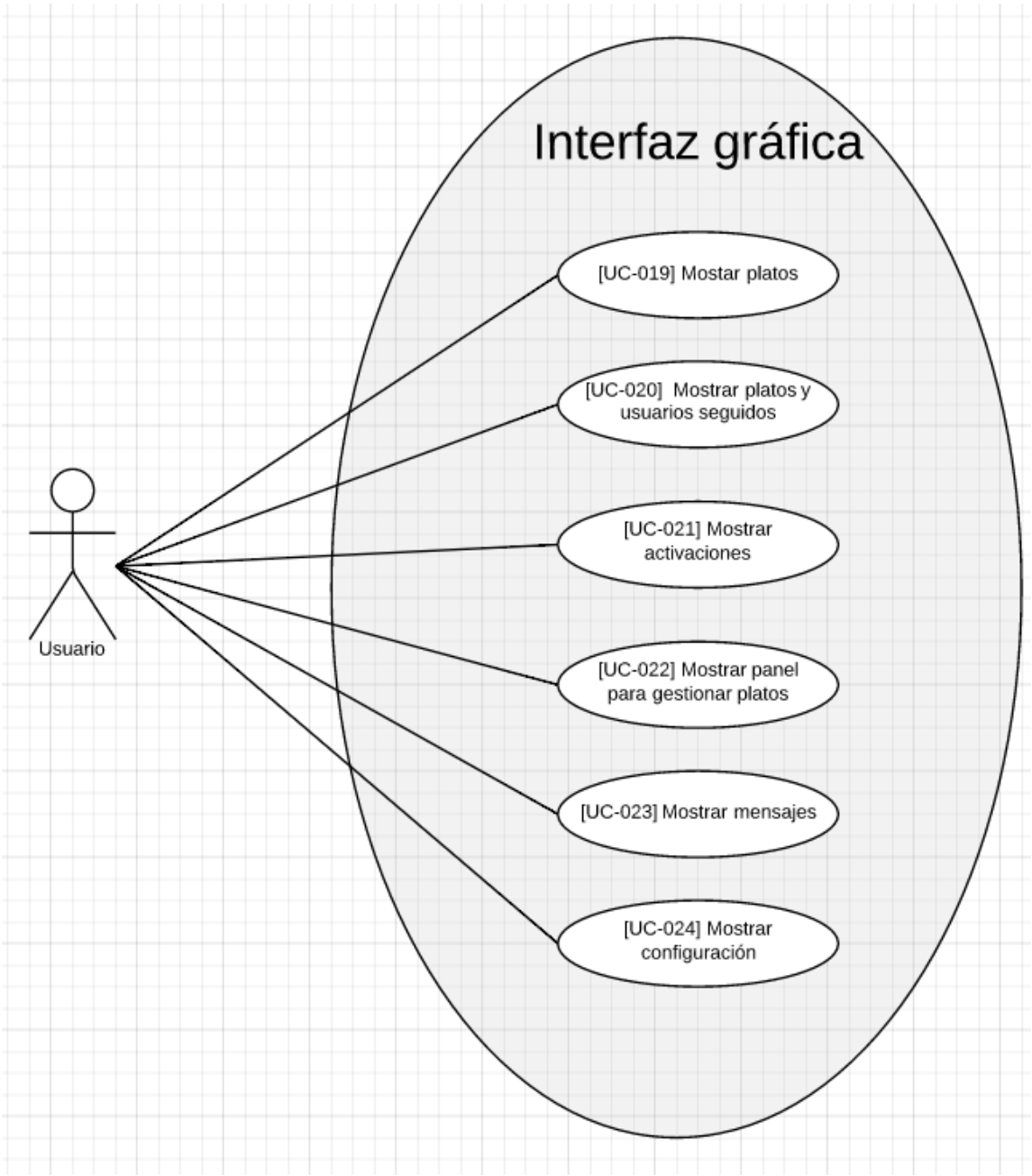
En este apartado vamos a citar los requisitos funcionales de nuestro proyecto. Estos requisitos fueron identificados en la fase de análisis. Los requisitos serán mostrados en forma de diagramas y casos de usos, ya que de esta forma tenemos unos datos más comprensibles.

Los casos de usos vendrán agrupados en distintos subsistemas según al que pertenezcan. Los subsistemas contienen los casos de usos que están relacionados entre sí, de esta forma nos será más fácil buscarlos. Hay que tener en cuenta que no todos los subsistemas tienen porque tener necesariamente casos de uso.

4.1.1 Diagrama de casos de usos

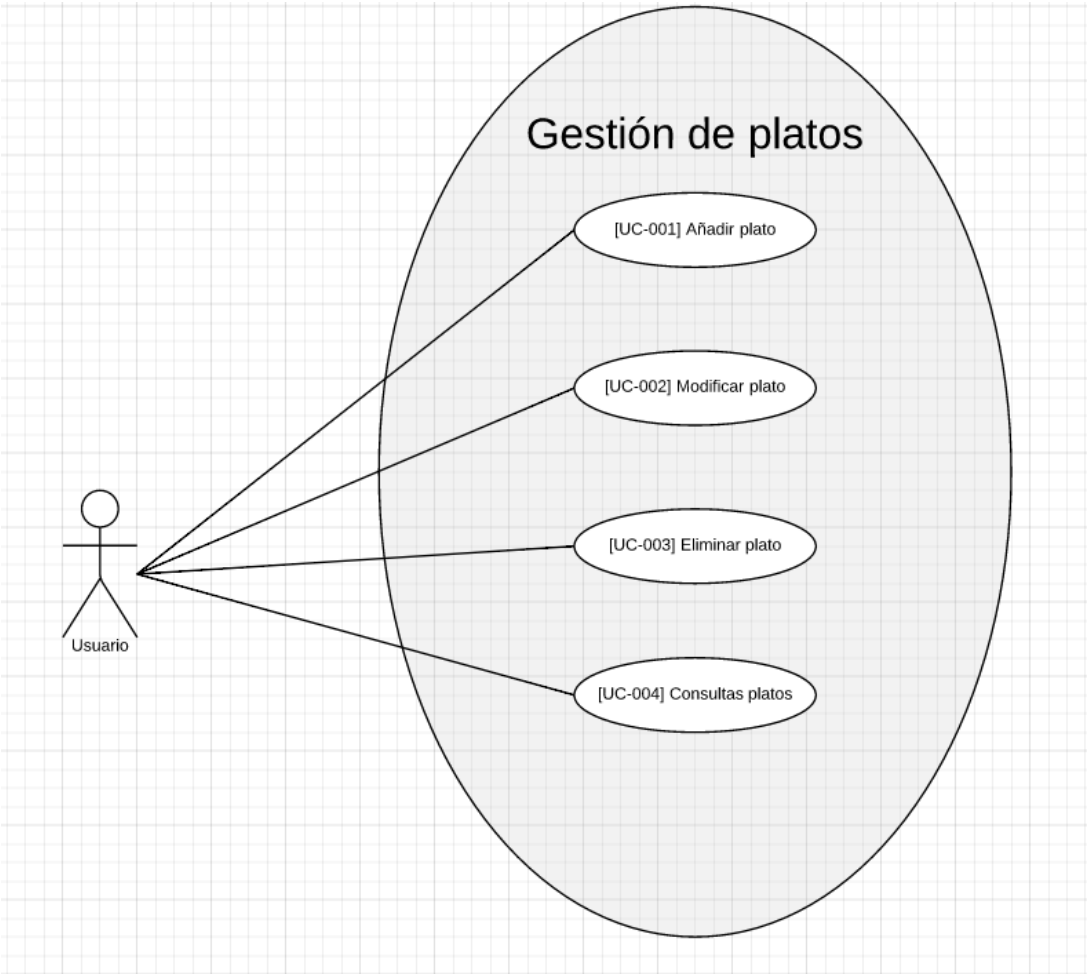
Los casos de uso definen el comportamiento del sistema cuando el usuario interactúa con este. En el caso de nuestro sistema tenemos un único usuario, que es el actor, por lo que vamos a definirlo a continuación.

4.3.1.1 Subsistema de interfaz gráfica



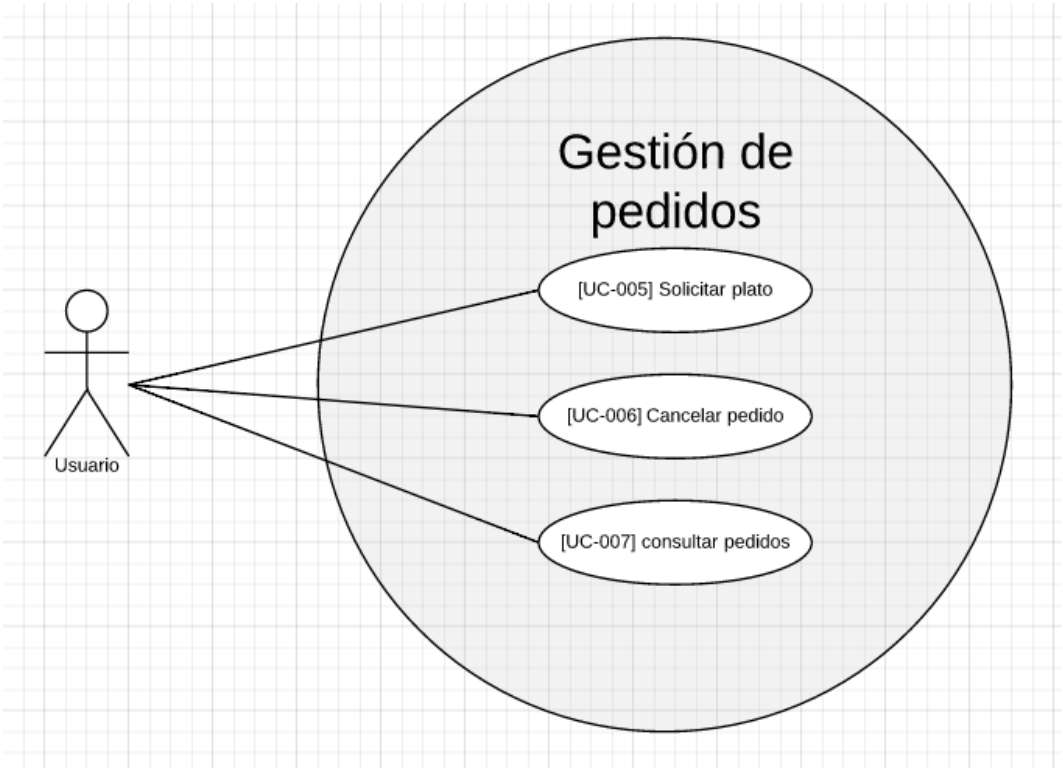
12. Ilustración – Subsistema de interfaz gráfica

4.3.1.2 Gestión de platos



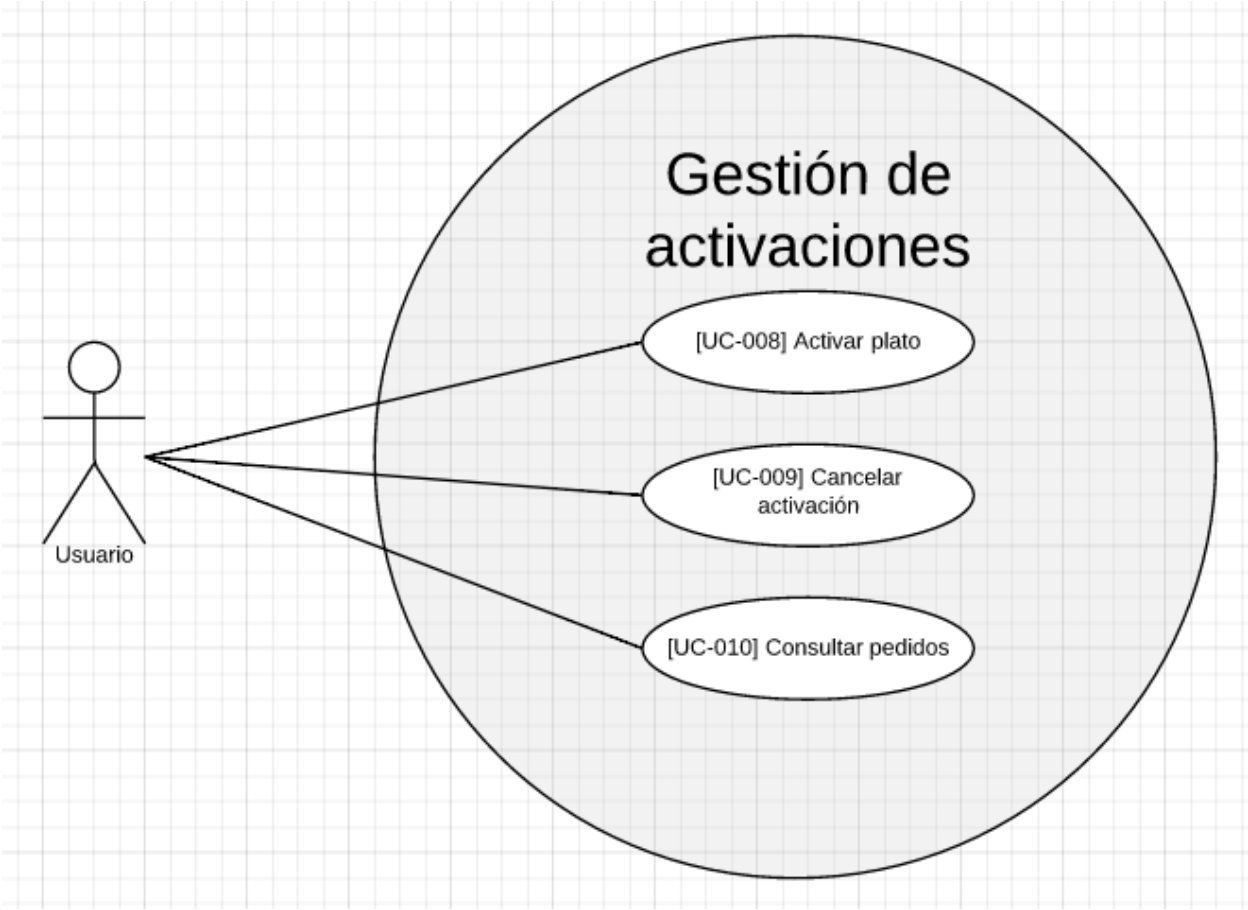
13. Ilustración – Subsistema de gestión de platos

4.3.1.3 Gestión de pedidos



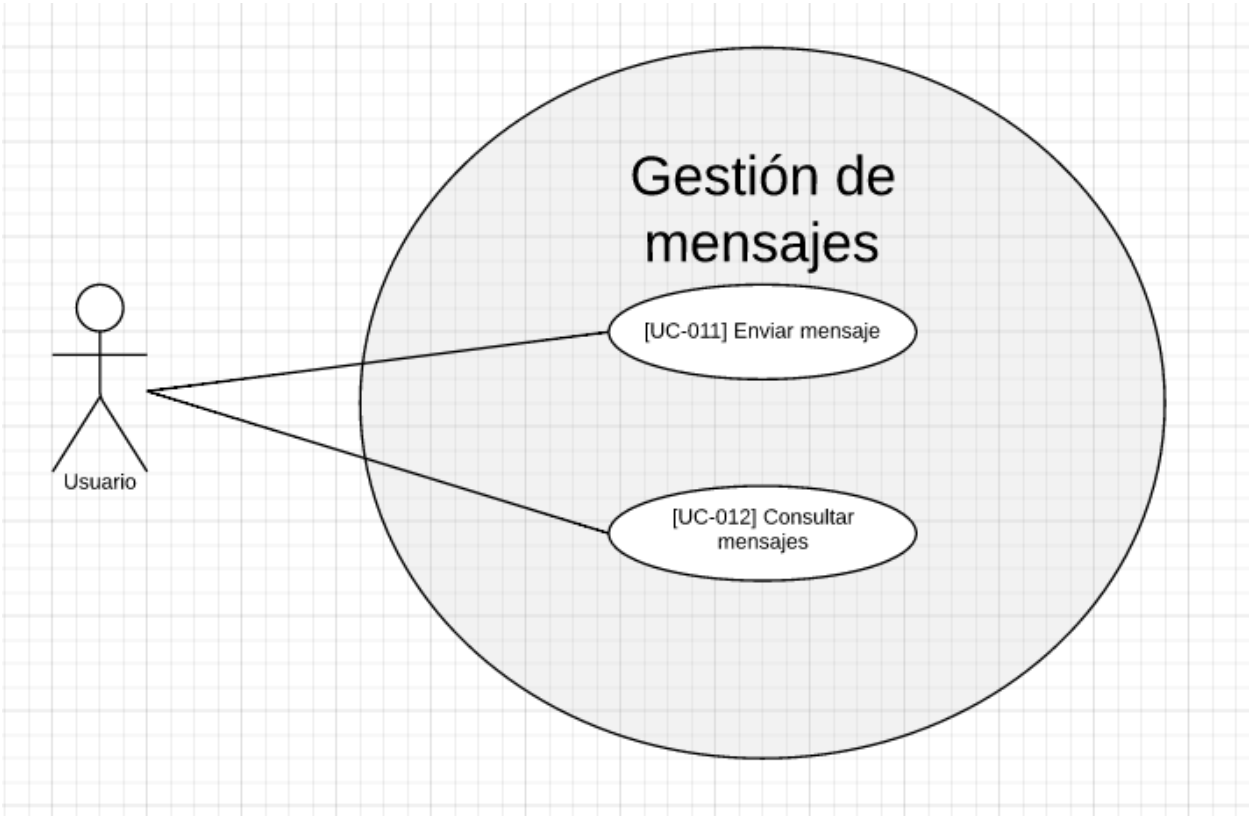
14. Ilustración – Subsistema de gestión de pedidos

4.3.1.4 Gestión de activaciones



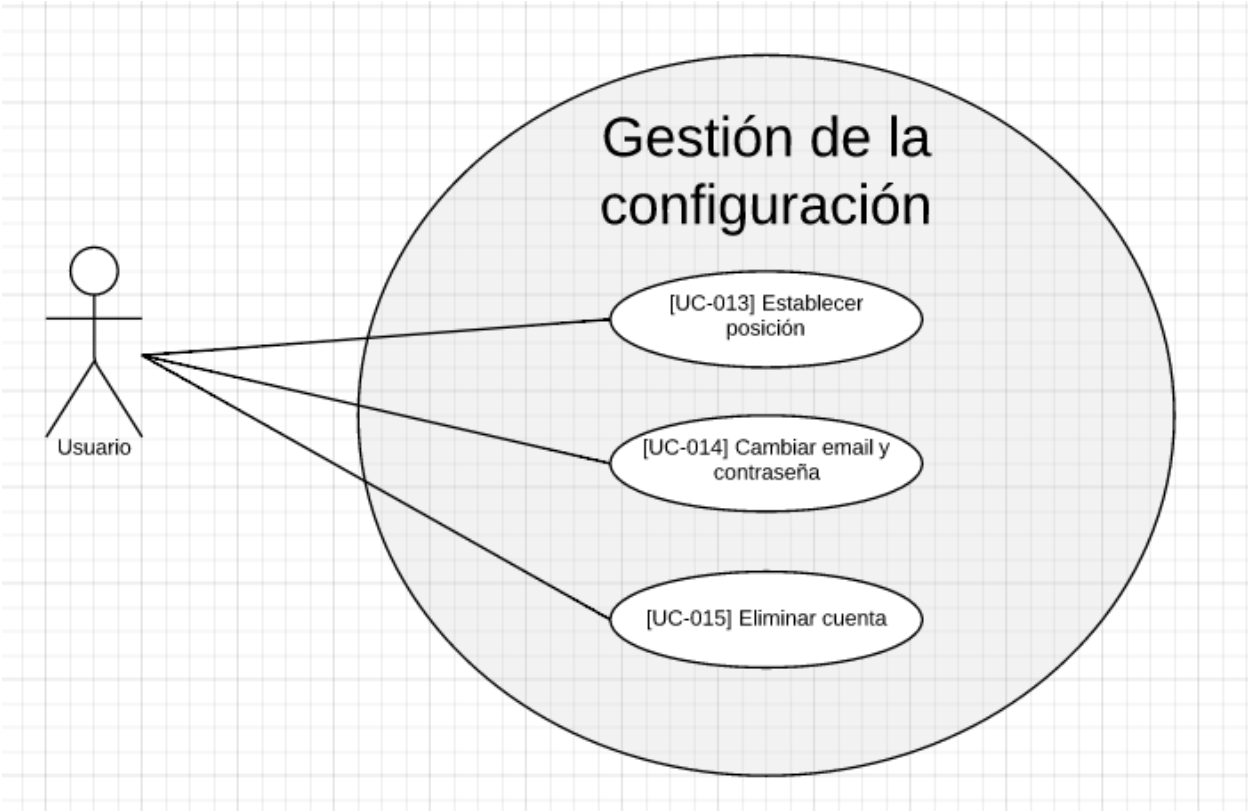
15. Ilustración – Subsistema de gestión de activaciones

4.3.1.5 Gestión de mensajes



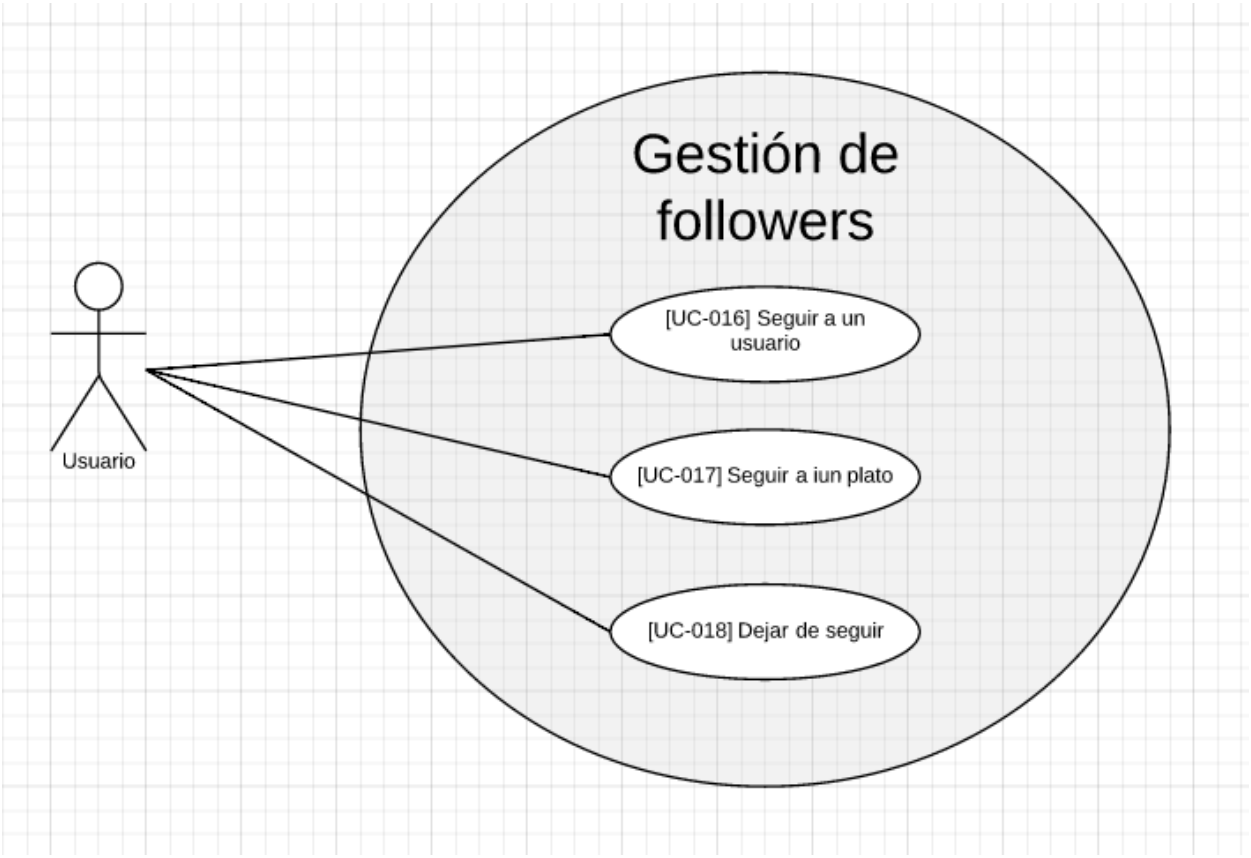
16. Ilustración – Subsistema de gestión de mensajes

4.3.1.6 Gestión de configuración



17. Ilustración – Subsistema de gestión de la configuración

4.3.1.7 Gestión de followers



18. Ilustración – Subsistema de gestión de followers

4.3.2 Definición de actores

ACT-001	Usuario de la aplicación
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Descripción	Este actor representa a cualquier usuario de la aplicación
Comentarios	Ninguno

6. Tabla – ACT-001

4.3.3 Casos de uso del sistema

UC-001	Añadir un plato
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-004] Seguir a platos y usuarios [OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional
Descripción	El usuario podrá crear cuantos platos quiera.
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

7. Tabla – UC-001

UC-002	Modificar un plato
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-004] Seguir a platos y usuarios [OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional
Descripción	El usuario podrá modificar sus platos.

Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

8. Tabla – UC-002

UC-003	**Eliminar un plato**
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-004] Seguir a platos y usuarios [OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional
Descripción	El usuario podrá eliminar sus platos
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

9. Tabla – UC-003

UC-004	**Consultar platos**
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-004] Seguir a platos y usuarios [OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional
Descripción	El usuario podrá consultar sus platos y los de cualquier usuario.
Precondición	Ninguna
Importancia	Importante

Urgencia	Urgente
Estado	Validado

10. Tabla – UC-004

UC-005	**Solicitar plato**
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-001] Publicar platos [OBJ-002] Obtener geolocalización del usuario [OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional
Descripción	El usuario podrá crear cuantos platos quiera.
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

11. Tabla – UC-005

UC-006	**Cancelar pedido**
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-001] Publicar platos [OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional
Descripción	El usuario podrá crear cuantos platos quiera.
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

12. Tabla – UC-006

UC-007	Consultar pedidos
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional [OBJ-003] Poner en contacto a los usuarios entre sí
Descripción	El usuario podrá crear cuantos platos quiera.
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

13. Tabla – UC-007

UC-008	Activar plato
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional [OBJ-001] Publicar platos [OBJ-002] Obtener geolocalización del usuario
Descripción	El usuario podrá crear cuantos platos quiera.
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

14. Tabla – UC-008

UC-009	Cancelar activación
Versión	1.0 (5/8/2015)

Autor	Mario Martínez García
Dependencias	[OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional [OBJ-001] Publicar platos [OBJ-002] Obtener geolocalización del usuario
Descripción	El usuario podrá crear cuantos platos quiera.
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

15. Tabla – UC-009

UC-010	**Consultar activaciones**
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional [OBJ-001] Publicar platos [OBJ-002] Obtener geolocalización del usuario
Descripción	El usuario podrá crear cuantos platos quiera.
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

16. Tabla – UC-010

UC-011	**Enviar mensaje**
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional

	[OBJ-003] Poner en contacto a los usuarios entre sí
Descripción	El usuario podrá crear cuantos platos quiera.
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

17. Tabla – UC-011

UC-012	Consultar mensajes
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional [OBJ-003] Poner en contacto a los usuarios entre sí
Descripción	El usuario podrá crear cuantos platos quiera.
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

18. Tabla – UC-012

UC-013	Establecer posición
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional [OBJ-002] Obtener geolocalización del usuario
Descripción	El usuario ha de establecer su geolocalización
Precondición	Ninguna

Importancia	Importante
Urgencia	Urgente
Estado	Validado

19. Tabla – UC-013

UC-014	Cambiar contraseña y email
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional [OBJ-002] Obtener geolocalización del usuario
Descripción	El usuario ha de establecer su geolocalización
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

20. Tabla - UC-014

UC-015	Eliminar cuenta
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional [OBJ-002] Obtener geolocalización del usuario
Descripción	El usuario ha de establecer su geolocalización
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

21. Tabla – UC-015

UC-016	Seguir a un usuario
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional [OBJ-002] Obtener geolocalización del usuario
Descripción	El usuario ha de establecer su geolocalización
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

22. Tabla – UC-016

UC-017	Seguir a un plato
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional [OBJ-002] Obtener geolocalización del usuario
Descripción	El usuario ha de establecer su geolocalización
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

23. Tabla – UC-017

UC-018	Dejar de seguir
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional [OBJ-002] Obtener geolocalización del usuario
Descripción	El usuario ha de establecer su geolocalización
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

24. Tabla – UC-018

UC-019	Mostrar platos
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional [OBJ-002] Obtener geolocalización del usuario
Descripción	El usuario ha de establecer su geolocalización
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

25. Tabla – UC-019

UC-020	Mostrar platos y usuarios seguidos
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional [OBJ-002] Obtener geolocalización del usuario
Descripción	El usuario ha de establecer su geolocalización
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

26. Tabla – UC-020

UC-021	Mostrar activaciones
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional [OBJ-002] Obtener geolocalización del usuario
Descripción	El usuario ha de establecer su geolocalización
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

27. Tabla – UC-021

UC-022	Mostrar panel para gestionar platos
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García

Dependencias	[OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional [OBJ-002] Obtener geolocalización del usuario
Descripción	El usuario ha de establecer su geolocalización
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

28. Tabla – UC-022

UC-023	Mostrar mensajes
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional [OBJ-002] Obtener geolocalización del usuario
Descripción	El usuario ha de establecer su geolocalización
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

29. Tabla – UC-023

UC-024	Mostrar configuración
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Dependencias	[OBJ-005] Ofrecer una interfaz de usuario sencilla y funcional [OBJ-002] Obtener geolocalización del usuario

Descripción	El usuario ha de establecer su geolocalización
Precondición	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

30. Tabla – UC-024

4.4 Requisitos no funcionales

Los requisitos funcionales son aquellos requisitos que no implican funciones a realizar ni describen información a guardar, sino más bien son de carácter general y que se exigen a cualquier sistema final. Ejemplos de requisitos no funcionales: rendimiento, disponibilidad, estabilidad, etc. Ahora vamos a exponer nuestros requisitos no funcionales.

NFR-001	Compatibilidad navegadores (Chrome 32+, Firefox 29+, IE 9+)
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Descripción	El framework Angular.js es relativamente reciente y no está soportado por los navegadores antiguos.
Dependencias	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

31. Tabla – NFR-001

NFR-002	Rendimiento
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Descripción	Una llamada a la API no tardará más de 1s de procesamiento en el servidor.
Dependencias	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

32. Tabla – NFR-002

NFR-003	Seguridad
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Descripción	El sistema debe garantizar tanto la privacidad del usuario como la seguridad de sus datos almacenados en el dispositivo
Dependencias	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

33. Tabla – NFR-003

NFR-004	Fiabilidad
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Descripción	El sistema debe ser fiable en todo momento de cara a las peticiones realizadas por el usuario
Dependencias	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

34. Tabla – NFR-004

NFR-005	Escalabilidad
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Descripción	Debido a que el mercado avanza muy rápido pueden surgir nuevas necesidades que deberíamos implementar en nuestro sistema, por lo tanto nuestro sistema debe ser fácilmente escalable.
Dependencias	Ninguna

Importancia	Importante
Urgencia	Urgente
Estado	Validado

35. Tabla – NFR-005

NFR-006	**Usabilidad**
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Descripción	Debido a que nos podemos encontrar en el mercado tanto usuario ocacionales como avanzados en nuevas tecnologías, nuestro sistema debe ser intuitivo, directo y sencillo de utilizar.
Dependencias	Ninguna
Importancia	Importante
Urgencia	Urgente
Estado	Validado

36. Tabla – NFR-006

4.5 Plan de negocio

La estrategía a utilizar para retabilizar la aplicación es mediante micro créditos.

Un plato, estaría valorizado según un número de créditos a determinar por el chef. Para poder adquirir dicho plato, el comensal tendría que desembolsar esta cantidad de créditos establecida por el cocinero de dicho plato.

Estos créditos, son recibidos en su mayoría por el cocinero, excepto una pequeña parte que recibe el sistema como comisión.

Cada usuario, ya sea cocinero o comensal, tiene un número de créditos, que puede conseguir comprándolos o ganándolos vendiendo platos.

El sistema aconsejará a los cocineros que dispongan de todos los certificados legales para la manipulación de alimentos así como estar dado de alta como autónomo en la Seguridad Social para poder vender platos.

Cookify no puede comprobar que estos requisitos legales sean cumplidos por parte de los cocineros ya que esta aplicación solo ofrece un servicio de intermediación.

4.5.1 Lanzamiento. Marketing.

Dada la característica colaborativa de la aplicación, para que el sistema pueda funcionar, dos usuarios (comensal y chef o cocinero) han de encontrarse relativamente cerca uno de otro, exactamente dentro de una zona de 2km de radio (establecido por el sistema).

Para haya posibilidades de éxito, se hace muy necesario iniciar una campaña de marketing de carácter local. Local, porque no tiene mucho sentido llegar a comensales que se encuentren a más de 50km de distancia de los cocineros.

5. Análisis del sistema

En el apartado anterior hemos definidos los requisitos del sistema, los cuales describe la idea del proyecto y sus objetivos. Ahora vamos a describir las tareas que deben llevarse a cabo para examinar los requisitos con el objetivo de delimitarlos y definir exactamente cada uno de ellos.

Los objetivos principales del análisis de requisitos consiste en: 9

 Detectar y resolver los conflictos entre requisitos.

 Delimitar el software y establecer con qué elementos externos interacciona.

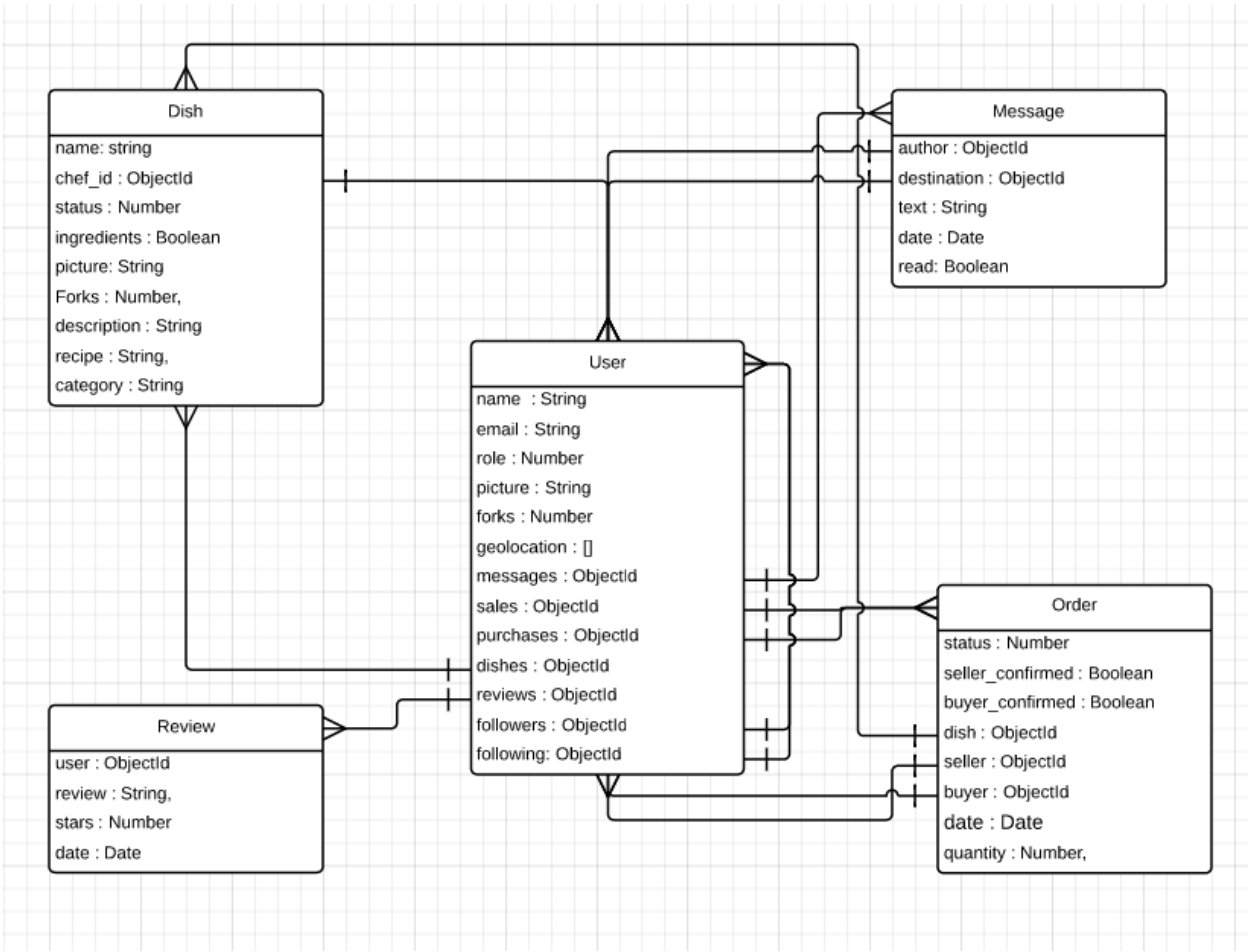
 Elaborar los requisitos del sistema para obtener, a partir de ellos, los requisitos del software a desarrollar.

En dicho análisis se definirán los aspectos más internos del sistema. Para ello recurriremos a una serie de diagramas que se reparten en tres apartados, donde se detallarán los flujos de información que se dan durante la ejecución de la aplicación.

5.1 Modelo estático del sistema

Este modelo nos permitirá definir los tipos que necesitaremos durante el desarrollo del sistema y sus interrelaciones con el propósito de obtener una visión global del mismo.

Todo elemento de un modelo estático debe estar trazado hacia aquellos requisitos que lo justifican.



19. Ilustración – Modelo de la base de datos.

TYP-001	Plato (Dish)
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Descripción	Objecto que modela el plato con todas sus variables ensenciales para la gestión del mismo.

Dependencias	Ninguna
Componentes	Ninguno
Comentarios	Ninguno

37. Tabla –TYP-001

TYP-002	**Usuario (user)**
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Descripción	Objecto que modela el usuario con todas sus variables ensenciales para la gestión del mismo.
Dependencias	Ninguna
Componentes	Ninguno
Comentarios	Ninguno

38. Tabla – TYP-002

TYP-003	**Valoración (Review)**
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Descripción	Objecto que modela las valoraciones con todas sus variables ensenciales para la gestión del mismo.
Dependencias	Ninguna
Componentes	Ninguno
Comentarios	Ninguno

39. Tabla – TYP-003

TYP-004	Mensaje (Message)
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Descripción	Objecto que modela los mensajes con todas sus variables ensenciales para la gestión del mismo.
Dependencias	Ninguna
Componentes	Ninguno
Comentarios	Ninguno

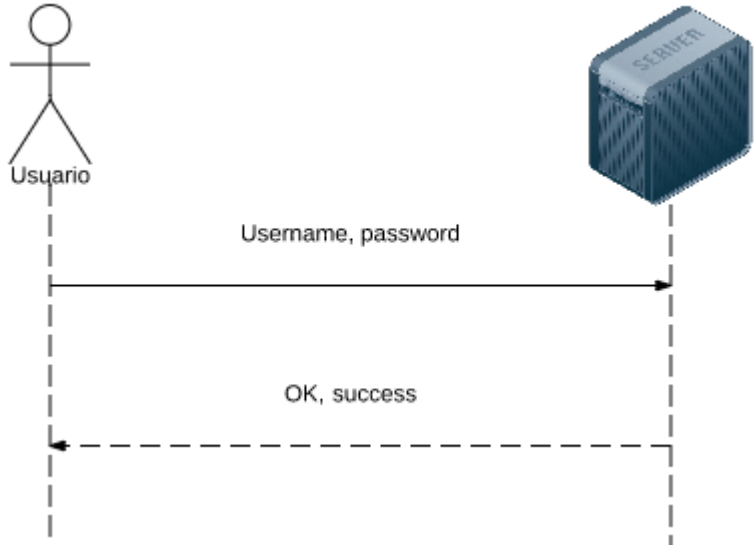
40. Tabla – TYP-004

TYP-005	Pedido (Order)
Versión	1.0 (5/8/2015)
Autor	Mario Martínez García
Descripción	Objecto que modela el plato con todos sus variable ensenciales para la gestión del mismo.
Dependencias	Ninguna
Componentes	Ninguno
Comentarios	Ninguno

41. Tabla – TYP-005

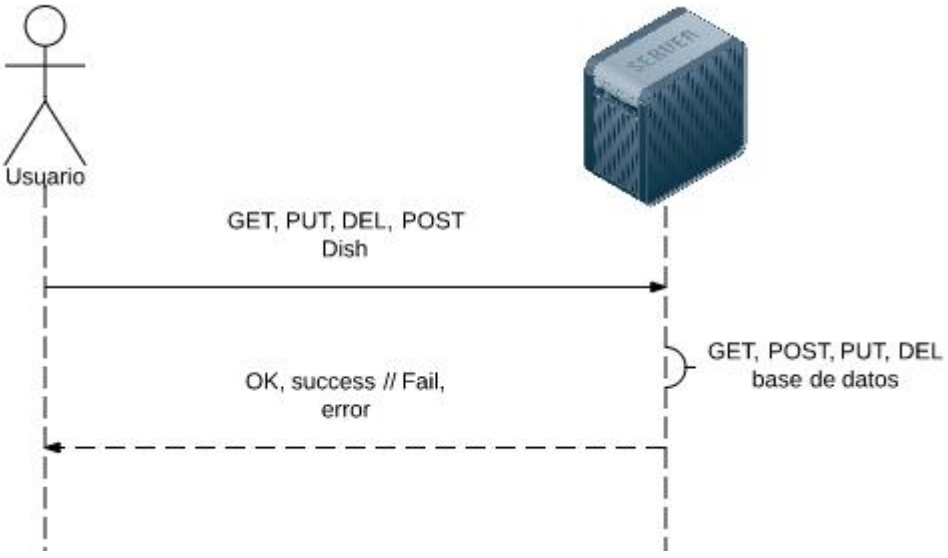
5.2 Modelo dinámico del sistema

5.2.1 Autenticación



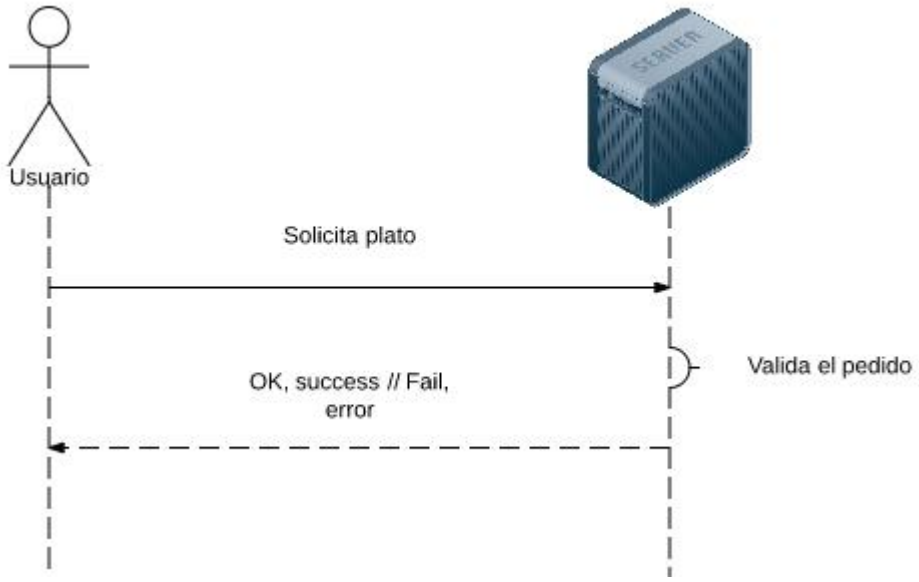
20. Ilustración – Operaciones de autenticación

5.2.2 Gestion de platos



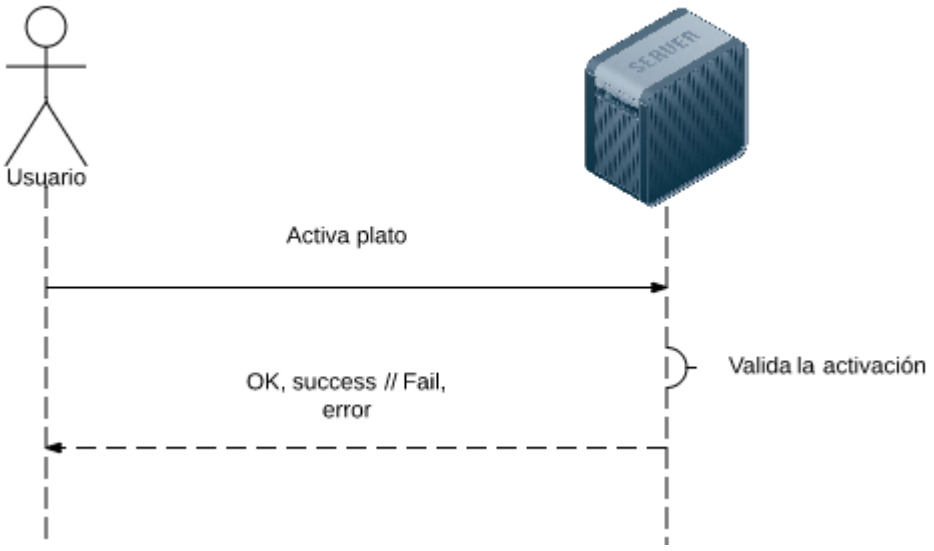
21. Ilustración – Operaciones de gestión de platos

5.2.3 Gestión de pedidos



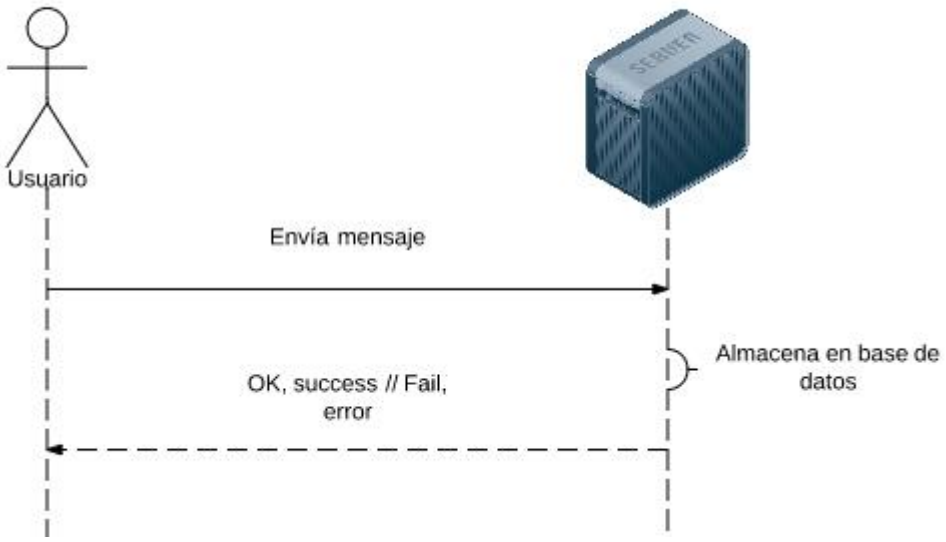
22. Ilustración – Operaciones de gestión de pedidos

5.2.4 Gestión de activaciones



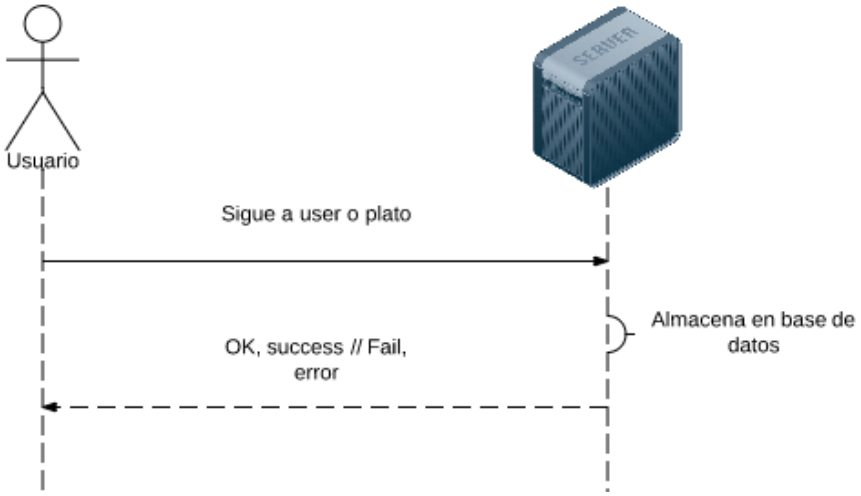
23. Ilustración – Operaciones de gestión de activaciones

5.2.5 Gestión de mensajes



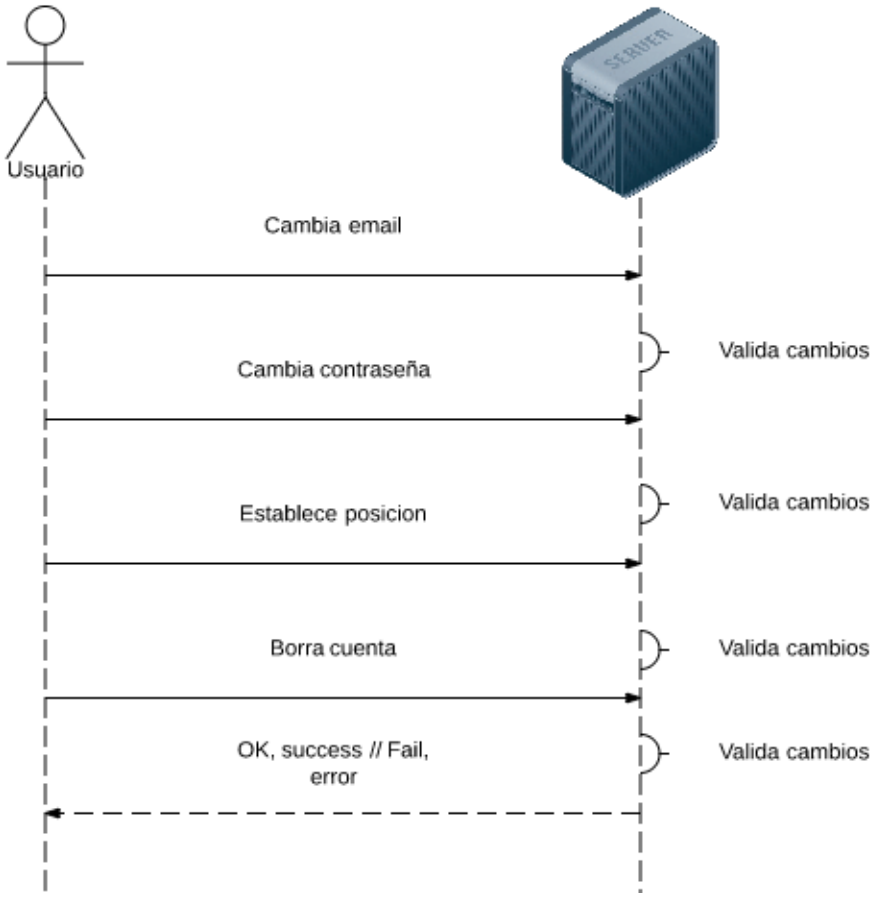
24. Ilustración – Operaciones de gestión de mensajes

5.2.6 Gestión de followers



25. Ilustración – Operaciones de gestión de followers

5.2.7 Gestión de la configuración



26. Ilustración – Operaciones de gestión de la configuración

6. Diseño del sistema

Una vez que hemos acabado con el análisis del sistema continuamos con el diseño del sistema. En esta fase se llevará a cabo dicho diseño que establecerá los pilares de la arquitectura del software a desarrollar así como el diseño de la interfaz de usuario. Está fase es necesario desarrollar un diseño lo más eficiente posible, pues un buen diseño puede marcar la diferencia a la hora de depurar, mantener o ampliar un proyecto software, por lo tanto nos definirá la calidad final del proyecto software.

También se contemplará la parte referente a la interfaz de usuario, ya que es otro de los aspectos más importantes de la aplicación. El diseño de la interfaz de usuario es la encargada de mostrar la información al usuario y que éste interactúe con la aplicación, por lo que es de vital importancia tener una interfaz organizada, sencilla y amigable, con la que el usuario se sienta cómodo trabajando.

6.1 Diseño de la base de datos

En mongoDB, el diseño se basa en schema para modelar los datos antes de introducirlos en la base de datos.

A continuación presentamos los scripts utilizados para definir los schema o modelados los documentos.

Schema Pedido

Hemos de mencionar, que este schema contiene una referencia al plato, al vendedor, comprador y a la activación de la que procede. Estas son referencias a otros documentos.

```
var purchaseSchema = new mongoose.Schema({
 status: Number, // 0 no realizado, 1 Realizado, 2 cancelado
 seller_confirmed: Boolean,
 buyer_confirmed: Boolean,
 dish: {
  type: mongoose.Schema.Types.ObjectId,
  ref: 'Dish'
 },
 seller: {
  type: mongoose.Schema.Types.ObjectId,
  ref: 'User'
 },
 buyer: {
  type: mongoose.Schema.Types.ObjectId,
  ref: 'User'
 },
 activation_id: {
  type: mongoose.Schema.Types.ObjectId
 },
 date: Date,
 quantity: Number,
 review_done : Boolean
});
```

Schema Usuario

```
var userSchema = new mongoose.Schema({
 name: String,
 email: {
  type: String,
  required: true,
  unique: true
 },
 password: {
  type: String
 },
```

```
role: String,
status : {type:Number,default:0},
token: String,
picture: String,
forks: {type : Number, default : 0},
mealcoins: {type : Number, default : 0},
validationEmailToken: String,
validationExpiresToken: Date,
location: {
   type: {
    type: String,
    enum: "Point",
    default: "Point"
   },
   coordinates: {
    type: [Number],
    default: [0, 0]
   }
 },
city: String,
country: String,
phone: String, //Del cooker para verificar la cuenta y del User, para mostrarlo al cooker
date_registered: Date,
cooker_date_expires: Date,
type_account: Number, //1 --> normal, 2 --> Vitalicia
chats: [{
 type: mongoose.Schema.Types.ObjectId,
 ref: 'Chat'
}],
purchases: [{
 type: mongoose.Schema.Types.ObjectId,
 ref: 'Purchase'
}],
sales: [{
 type: mongoose.Schema.Types.ObjectId,
 ref: 'Purchase'
}],
```

```
dishes: [{
  type: mongoose.Schema.Types.ObjectId,
  ref: 'Dish'
}],
reviews: [{
  user_id: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User'
  },
  review: String,
  stars: Number, // 0 - 5
  date: Date
}],
reviews_done: [{
  target_type: String, //dish or chef
  target_id: String,
  review: String,
  stars: Number, // 0 - 5
  date: Date
}],
activations: [{
  dish: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'Dish'
  },
  day: Number,
  date_created: Date,
  timeStart: String, // Sí status es 1
  timeEnd: String, // Sí status es 1
  quantity: Number,
  left: Number,
  price: Number,
  canEdit: Boolean
}],
facebookID: String,
googleID: String,
twitterID: String,
```

```
  user_blocked: [{
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User'
  }],
  user_following: [{
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User'
  }],
  dish_following: [{
    type: mongoose.Schema.Types.ObjectId,
    ref: 'Dish'
  }],
  followers: [{
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User'
  }],
  notifications: [notificationSchema]
})
```

Schema Chat

El schema chat es un contener de mensajes, una conversación entre dos usuarios.

```
var chatSchema = new mongoose.Schema({
  users: [{
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User'
  }],
  messages: [{
    author: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'User'
    },
    text: String,
    date: Date,
    read: Boolean
  }]
});
```

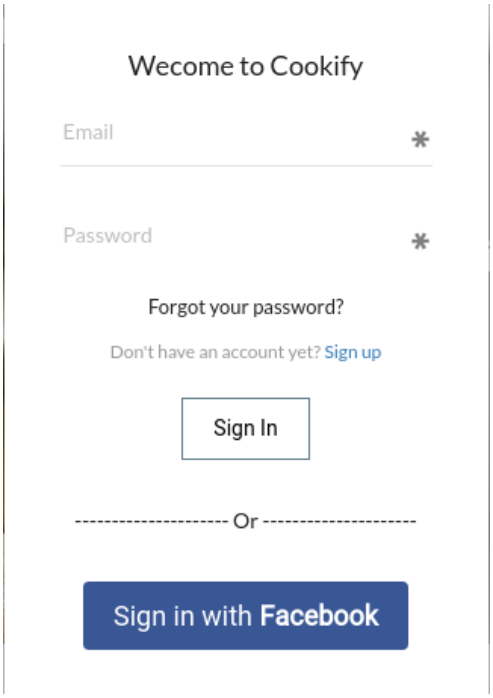
Schema Plato

```
var dishSchema = new mongoose.Schema({
  name: String,
  chef_id: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User'
  },
  status : {type:Number,default:0},
  chef_name: String,
  ingredients: String,
  picture: String,
  forks: {type : Number, default : 0},
  weight: {type : Number, default : 0},
  description: String,
  recipe: String,
  sold_quantity: {type : Number, default : 0},
  category: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'Category'
  },
  reviews: [{
    user_id: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'User'
    },
    review: String,
    stars: {type : Number, default : 0}, // 0 - 5
    date: Date
  }],
  date_created: Date,
  last_modified: Date,
  followers: [{
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User'
  }]
});
```

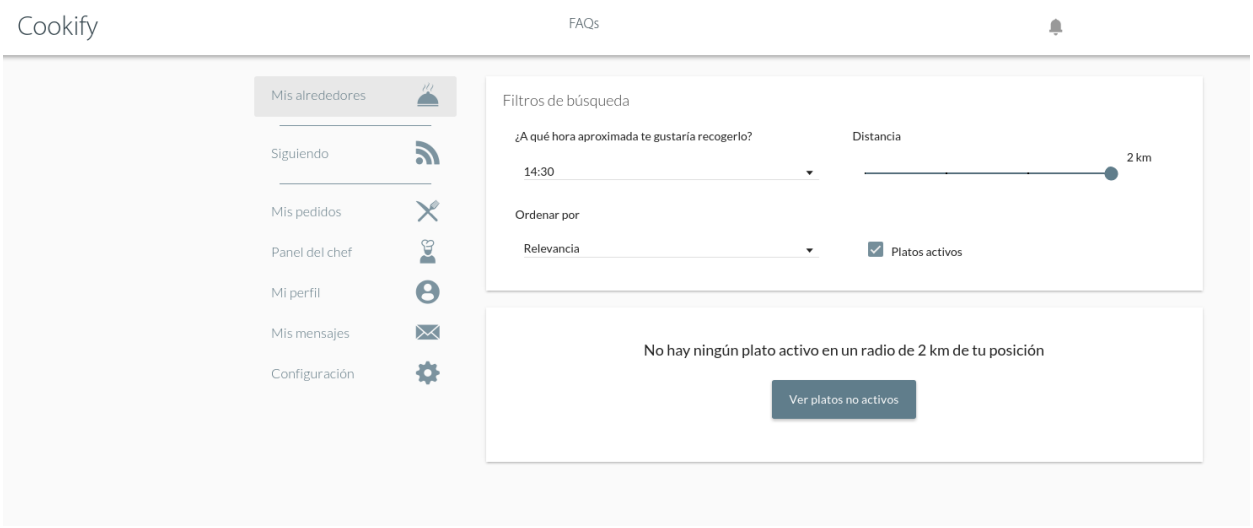
6.3 Diseño de la interfaz

A continuación vamos a mostrar capturas de pantallas donde refleje la interfaz.

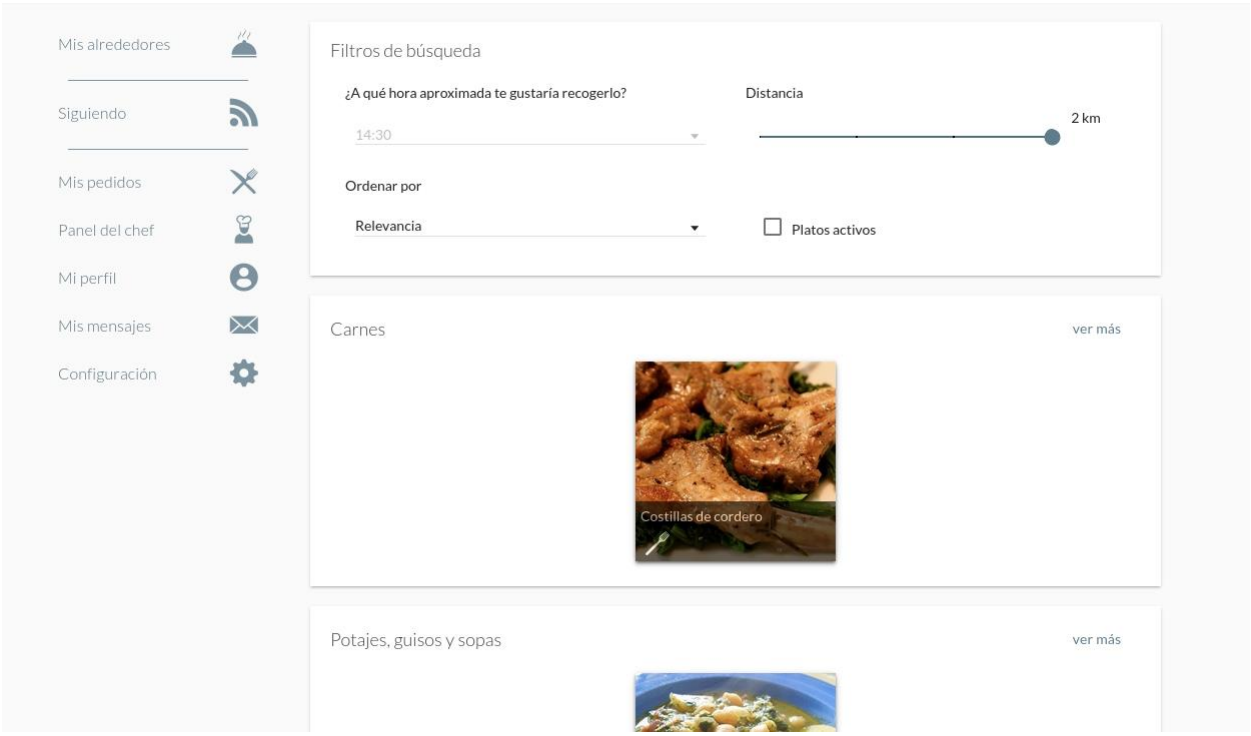
Login



Mis alrededores

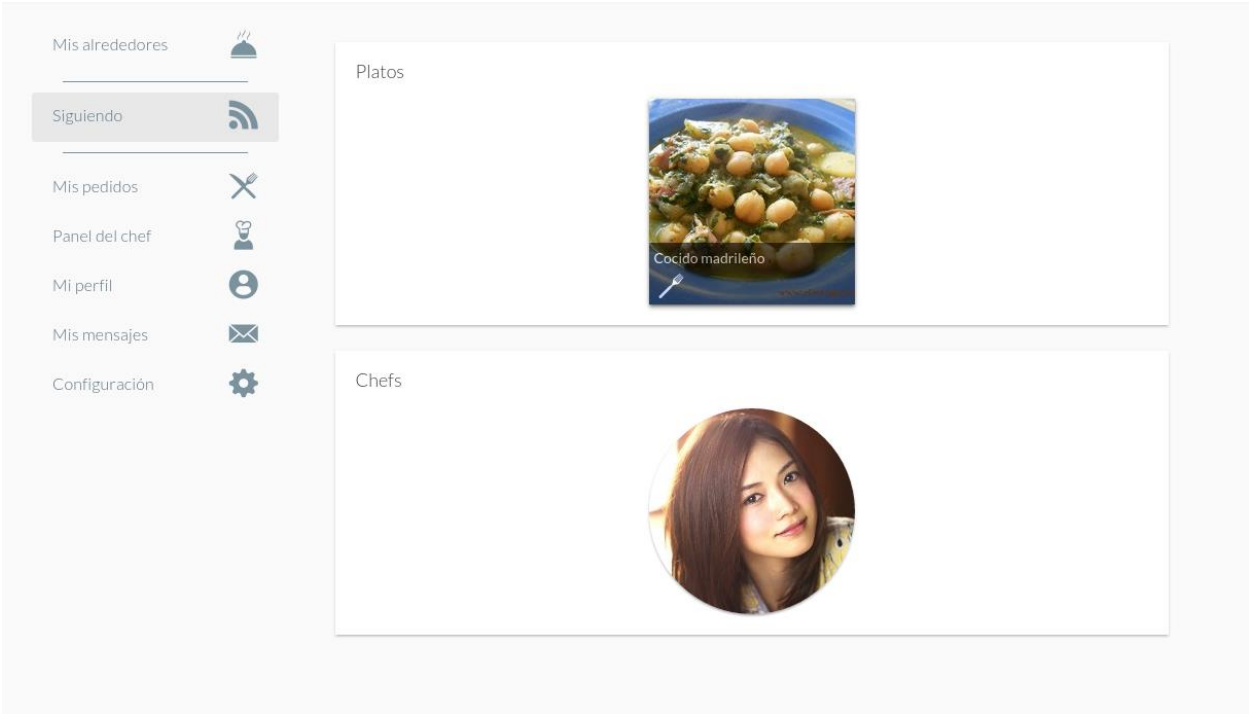


Y si le damos al botón de mostrar platos no activos.

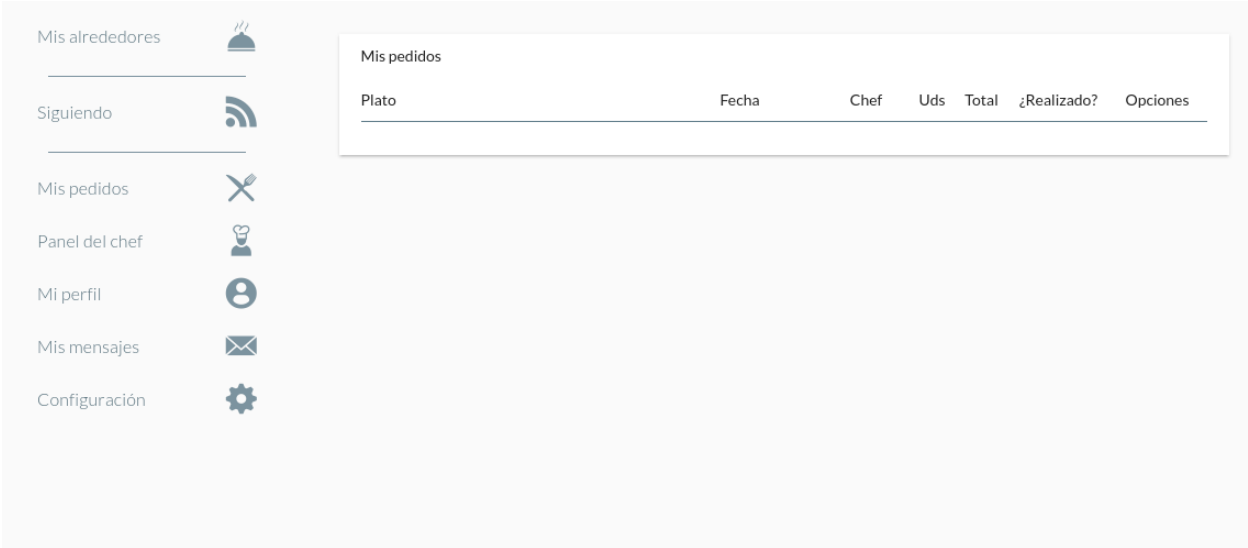


Siguiendo

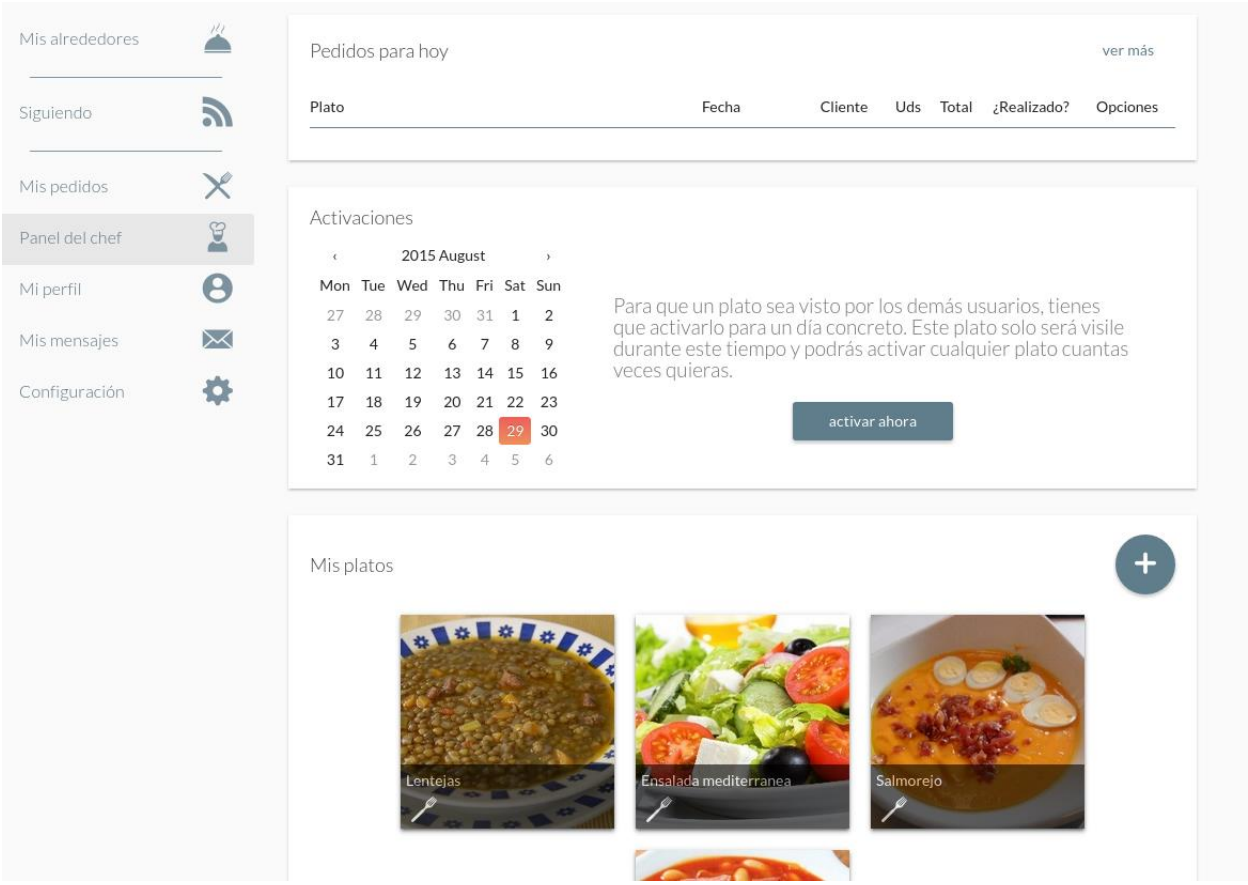
En este apartado se presentan tanto los usuarios como a los platos que estamos siguiendo



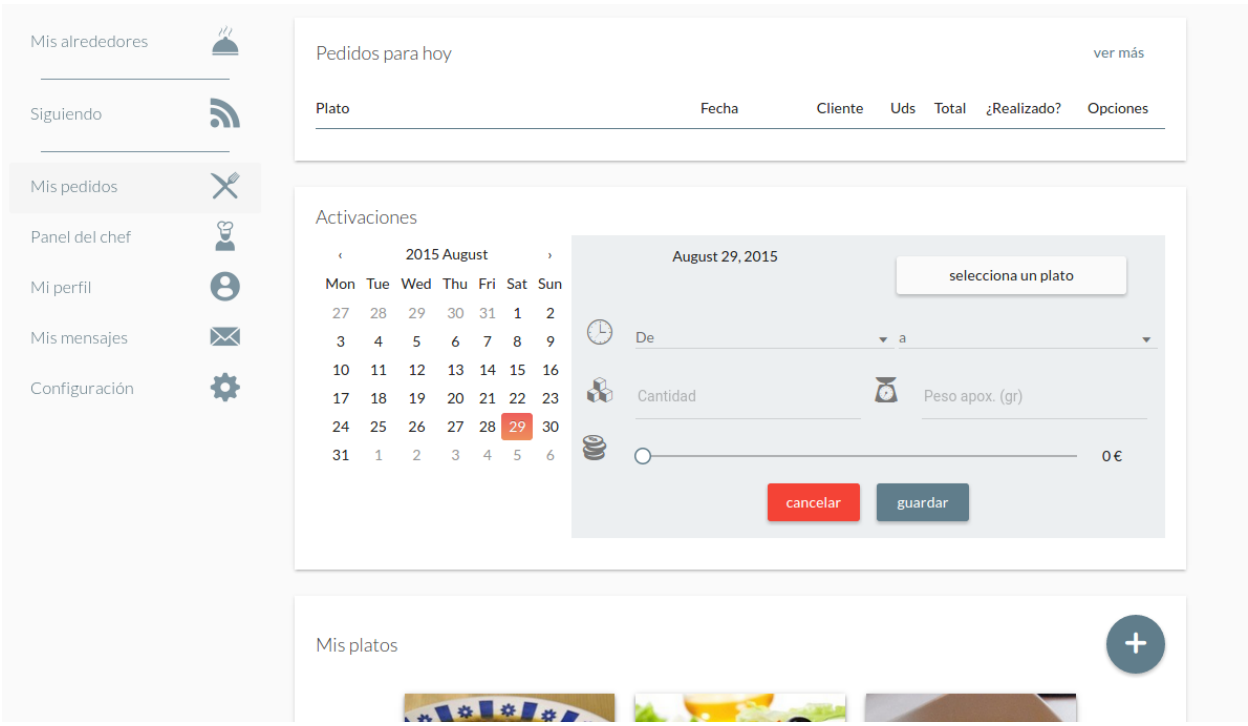
Mis pedidos



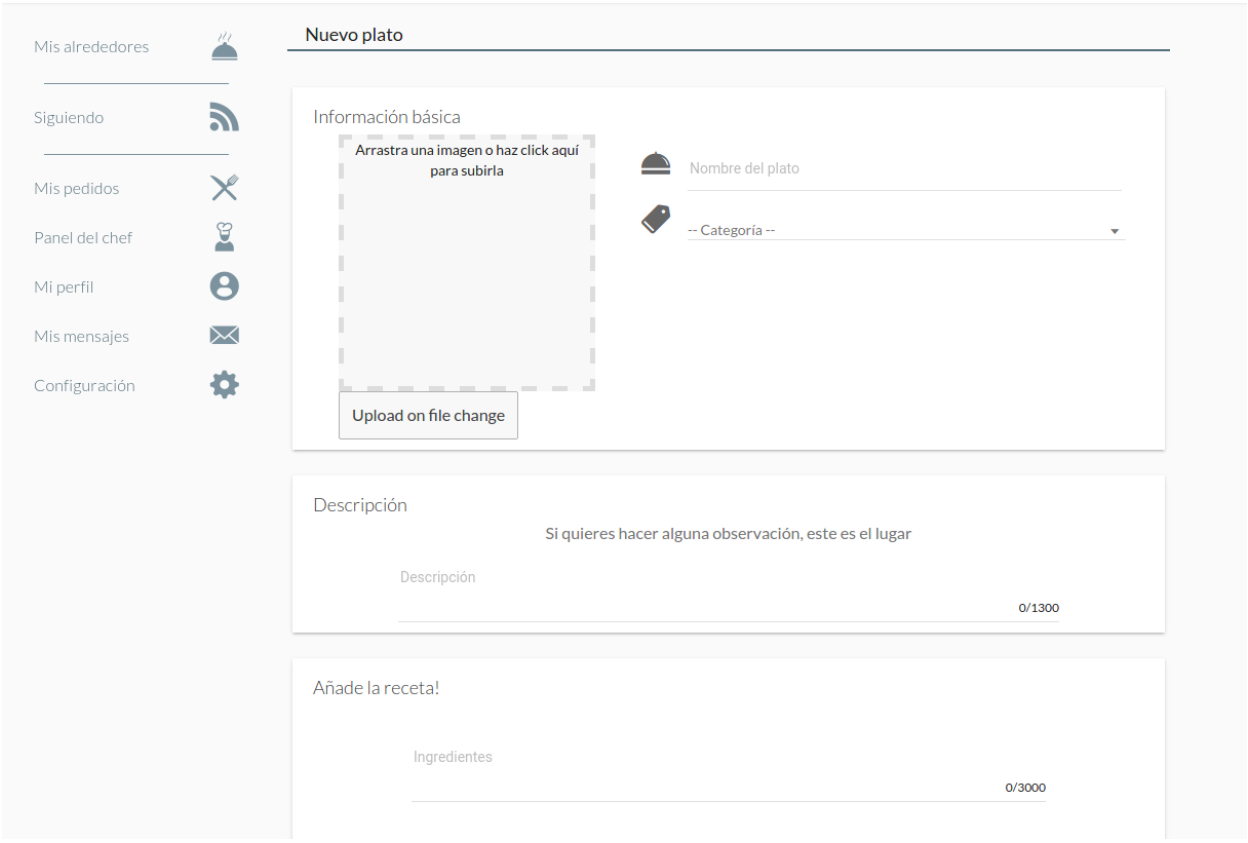
Panel del chef



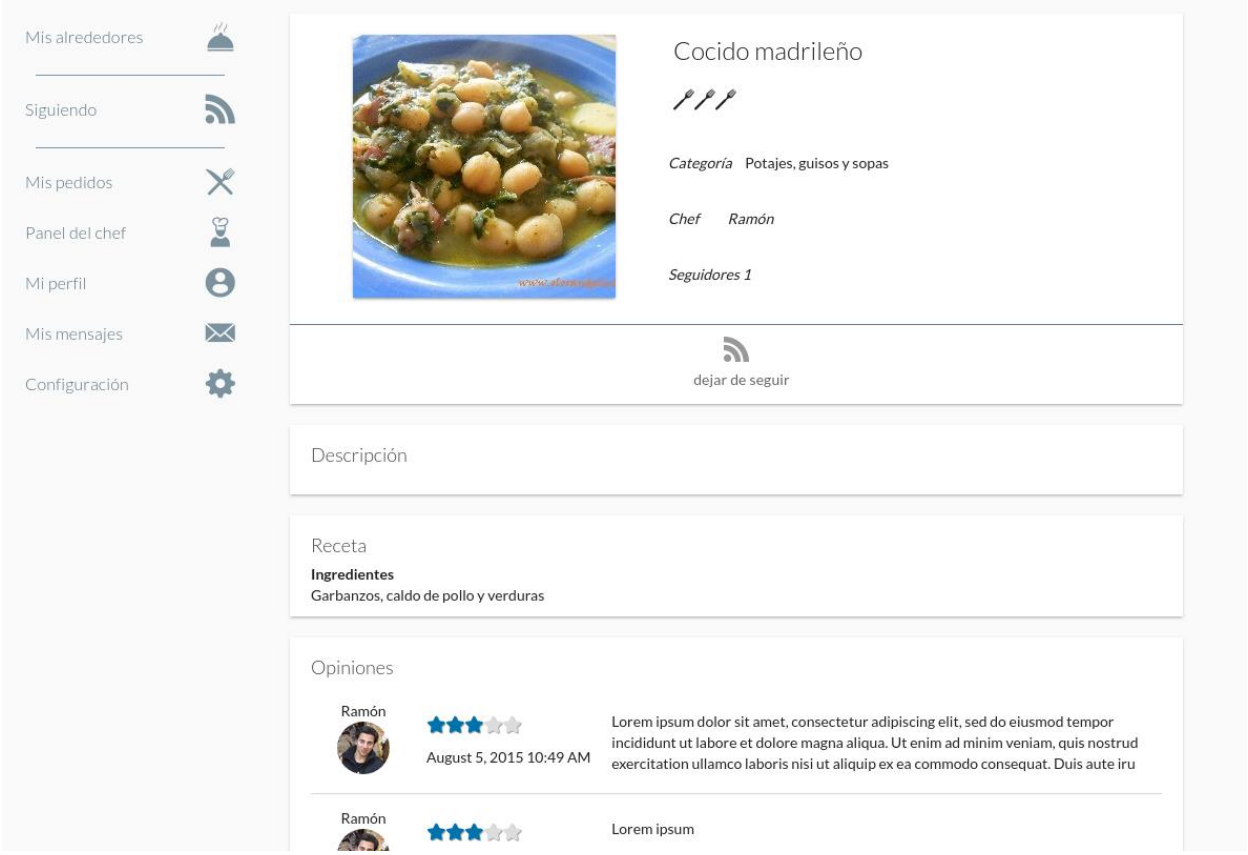
Y si le damos al botón para activar un plato



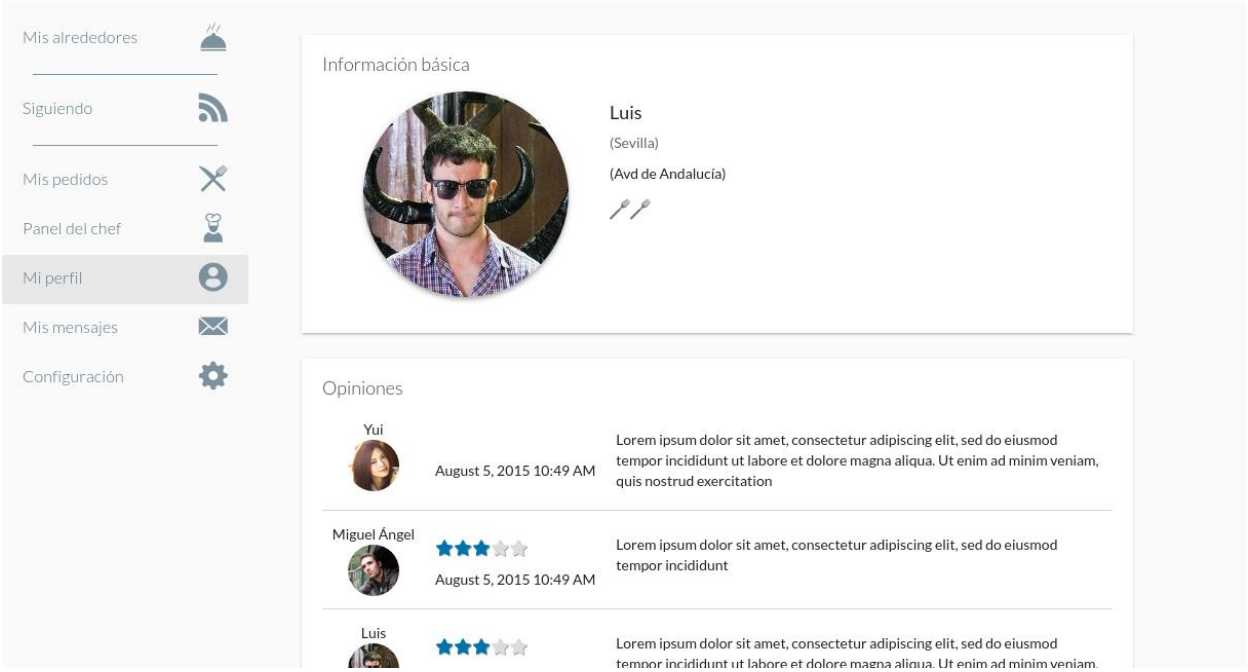
Si creamos un plato…



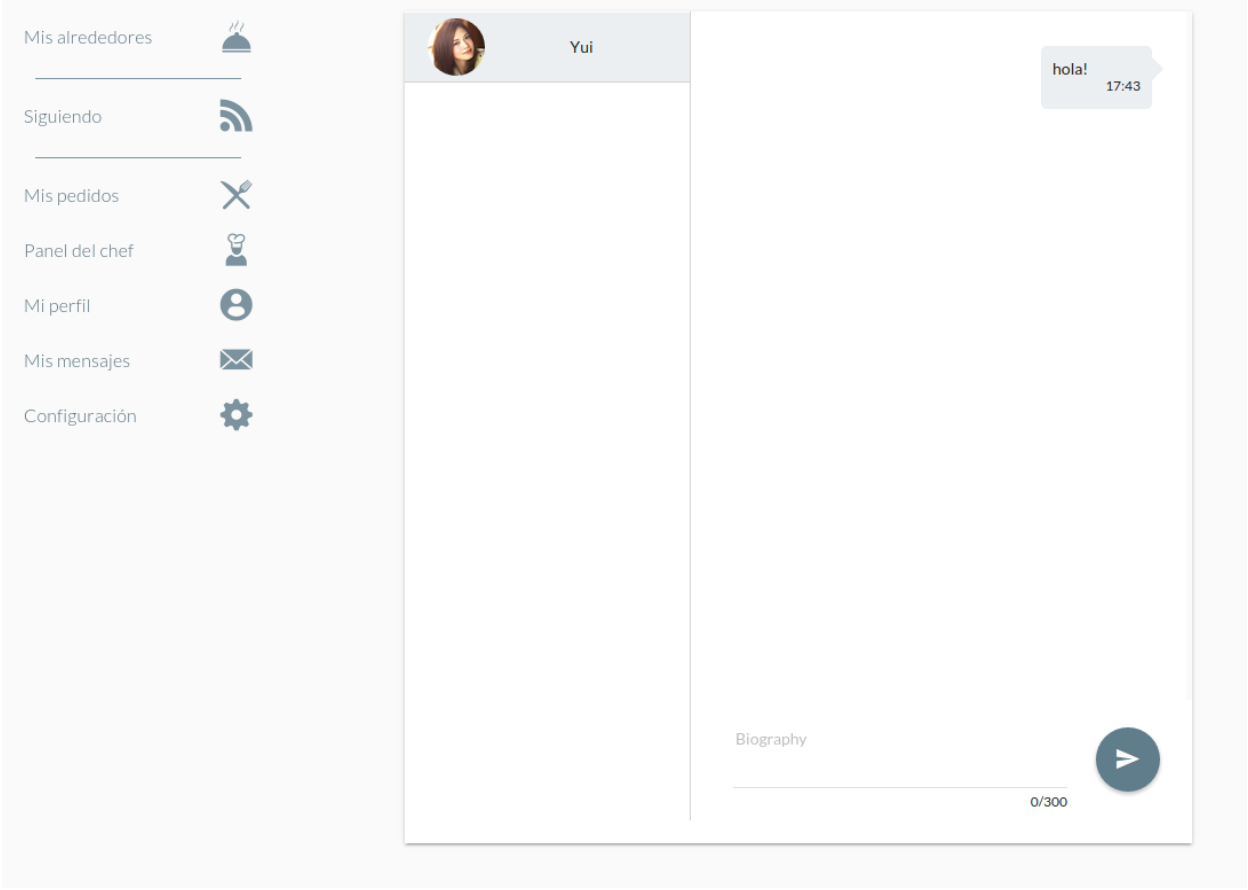
Plato



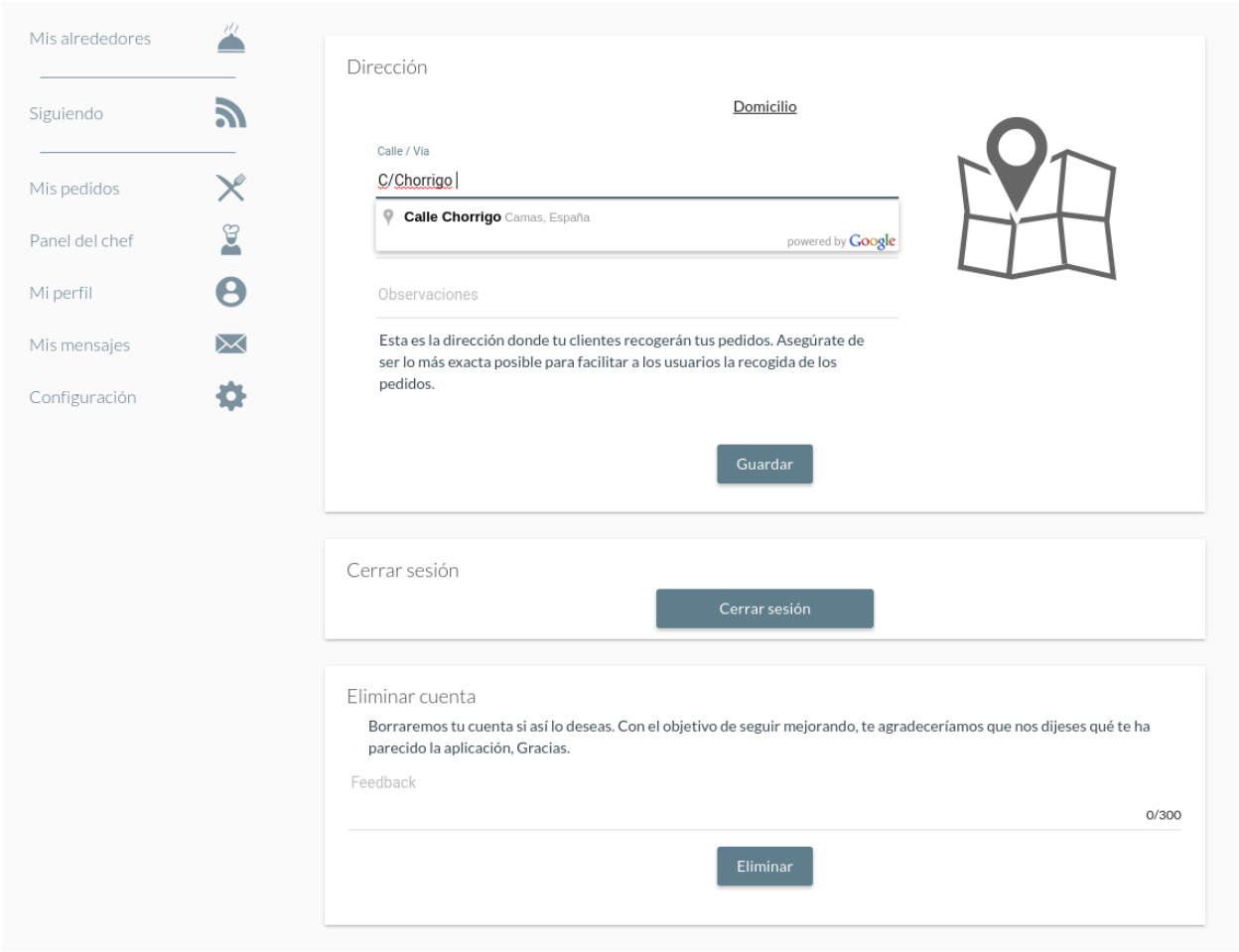
Mi perfil



Mis mensajes



Configuración


Mis alrededores
Siguiendo
Mis pedidos
Panel del chef
Mi perfil
Mis mensajes
Configuración
Dirección
Domicilio
Calle / Vía
C/Chorrigo
Calle Chorrigo Camas, España
powered by Google
Observaciones
Esta es la dirección donde tu clientes recogerán tus pedidos. Asegúrate de ser lo más exacta posible para facilitar a los usuarios la recogida de los pedidos.
Guardar
Cerrar sesión
Cerrar sesión
Eliminar cuenta
Borraremos tu cuenta si así lo deseas. Con el objetivo de seguir mejorando, te agradeceríamos que nos dijeses qué te ha parecido la aplicación, Gracias.
Feedback
0/300
Eliminar

7. Implementación

7.1 Comunicación cliente-servidor

Para llevar a cabo esta tarea se ha optado por implementar una api Restful.

A continuación vamos a listar todos los 'End Points' realizados para la correcta comunicación del sistema.

url	**/api/v1/background**
Versión	1.0.0 (5/8/2015)
Método	GET
Descripción	Es un cron que se encarga de comprobar si hay actualizaciones.
Autor	Mario Martínez García

42. Tabla – REST GET Background

url	**/api/v1/auth/login**
Versión	1.0.0 (5/8/2015)
Método	POST
Descripción	La función que valida el usuario
Autor	Mario Martínez García

43. Tabla – REST POST Login

url	/api/v1/auth/signup
Versión	1.0.0 (5/8/2015)
Método	POST
Descripción	Recibe datos del usuario para el registro
Autor	Mario Martínez García

44. Tabla – REST POST Signup

url	/api/v1/auth/facebook
Versión	1.0.0 (5/8/2015)
Método	POST
Descripción	Gestiona el login de facebook
Autor	Mario Martínez García

45. Tabla – REST POST auth Facebook

url	/api/v1/user/validate/email
Versión	1.0.0 (5/8/2015)
Método	GET
Descripción	Valida el email
Autor	Mario Martínez García

46. Tabla – REST GET validate email

url	/api/v1/user/:id
Versión	1.0.0 (5/8/2015)
Método	GET
Descripción	Consulta un usuario
Autor	Mario Martínez García

47. Tabla – REST GET user

url	/api/v1/user/picture
Versión	1.0.0 (5/8/2015)
Método	POST
Descripción	Modifica la imagen de perfil del usuario
Autor	Mario Martínez García

48. Tabla – REST POST picture

url	/api/v1/activate
Versión	1.0.0 (5/8/2015)
Método	POST
Descripción	Activa un plato
Autor	Mario Martínez García

49. Tabla – REST POST activate

url	/api/v1/setClientLocation
Versión	1.0.0 (5/8/2015)
Método	POST
Descripción	Actualiza la posición del usuario
Autor	Mario Martínez García

50. Tabla – REST POST setClientLocation

url	/api/v1/messages
Versión	1.0.0 (5/8/2015)
Método	GET
Descripción	Consulta los mensajes

Autor	Mario Martínez García

51. Tabla – REST GET messages

url	**/api/v1/messages**
Versión	1.0.0 (5/8/2015)
Método	POST
Descripción	Escribe un mensaje
Autor	Mario Martínez García

52. Tabla – REST POST messages

url	**/api/v1/following/dishes**
Versión	1.0.0 (5/8/2015)
Método	GET
Descripción	Consulta los platos seguidos por el usuario
Autor	Mario Martínez García

53. Tabla – REST GET dishes

url	**/api/v1/following/users**
Versión	1.0.0 (5/8/2015)
Método	GET
Descripción	Consulta los usuarios seguidos por el usuario
Autor	Mario Martínez García

54. Tabla – REST GET following users

url	/api/v1/dish/:id
Versión	1.0.0 (5/8/2015)
Método	GET
Descripción	Consulta los platos
Autor	Mario Martínez García

55. Tabla – REST GET dish

url	/api/v1/dish
Versión	1.0.0 (5/8/2015)
Método	POST
Descripción	Crea un nuevo plato
Autor	Mario Martínez García

56. Tabla – REST POST dish

url	/api/v1/dish
Versión	1.0.0 (5/8/2015)
Método	PUT
Descripción	Modifica un plato existente
Autor	Mario Martínez García

57. Tabla – REST PUT dish

url	/api/v1/dish
Versión	1.0.0 (5/8/2015)
Método	DELETE
Descripción	Elimina un plato
Autor	Mario Martínez García

58. Tabla – REST DELETE dish

url	/api/v1/order
Versión	1.0.0 (5/8/2015)
Método	GET
Descripción	Consulta los pedidos
Autor	Mario Martínez García

59. Tabla – REST GET order

url	/api/v1/order
Versión	1.0.0 (5/8/2015)
Método	POST
Descripción	Realiza un nuevo pedido
Autor	Mario Martínez García

60. Tabla – REST POST order

url	/api/v1/orderDone
Versión	1.0.0 (5/8/2015)
Método	POST
Descripción	Confirma que se ha realizado la transacción de un pedido
Autor	Mario Martínez García

61. Tabla – REST POST orderDone

url	/api/v1/review
Versión	1.0.0 (5/8/2015)
Método	POST
Descripción	Crea una valoración sobre un plato o un usuario
Autor	Mario Martínez García

62. Tabla – REST POST review

url	/api/v1/activate
Versión	1.0.0 (5/8/2015)
Método	GET
Descripción	Crea una activación de un plato
Autor	Mario Martínez García

63. Tabla – REST GET actívate

url	/api/v1/activate
Versión	1.0.0 (5/8/2015)
Método	DELETE
Descripción	Elimina una activación existente
Autor	Mario Martínez García

64. Tabla – REST DELETE activate

url	/api/v1/follow
Versión	1.0.0 (5/8/2015)
Método	POST
Descripción	Sigue a un usuario o plato
Autor	Mario Martínez García

65. Tabla – REST POST follow

url	/api/v1/unfollow
Versión	1.0.0 (5/8/2015)
Método	POST
Descripción	Deja de seguir
Autor	Mario Martínez García

66. Tabla – REST POST unfollow

url	/api/v1/account
Versión	1.0.0 (5/8/2015)
Método	DELETE
Descripción	Borra la cuenta
Autor	Mario Martínez García

67. Tabla – REST DELETE account

7.2 Cron en Background

El funcionamiento de http, es unidireccional, es decir, realizo una petición y el servidor devuelve una respuesta.

El servidor nunca va a enviar datos al cliente sin haberlo solicitado antes.

Para el correcto funcionamiento de los mensajes, tenemos que tener un cron en el cliente, que periódicamente vaya solicitando al servidor si hay mensajes nuevos, en cuyo caso se transmiten a este.

8. Pruebas

El apartado de pruebas es vital en un proyecto, puesto que es la última oportunidad de comprobar que se cumplen todos los requisitos y exigencias antes de dar el proyecto por finalizado. A continuación, se describirán los procedimientos que se han empleado para probar el sistema y garantizar así la calidad del producto final.

8.1 Condiciones generales

Se debe comprobar que la funcionalidad del sistema es completa y exacta para llevar a cabo la validación. Por lo tanto se realizarán pruebas unitarias a cada uno de los módulos, y una vez acabadas se analizará el funcionamiento del conjunto del programa.

Se comprobará que todas las excepciones son manejada de forma correcta según el caso, de esta forma se evitará rupturas de ejecución y otras situaciones no deseadas. Evaluaremos los tiempos de respuesta del sistema a las distintas peticiones del usuario. También se validará que los datos almacenados sean correctos, y que se resuelva en un tiempo razonable aquellas operaciones que bloqueen el sistema o en caso contrario, proporcionar una alternativa al usuario.

8.2 Alcance de las pruebas

Los distintos niveles de pruebas que se van a realizar al sistema durante su desarrollo son los que se citan a continuación:

- Pruebas unitarias, de cada una de las funciones de las clases del sistema. Se realizarán durante las primeras fases de programación del sistema. Deben realizar correctamente su funcionalidad y devolver los valores esperados para ser validadas.
- Pruebas de integración, entre las distintas funciones. Se realizarán junto con las pruebas unitarias. Para ser validadas las funciones de cada una de las clases deben interoperar entre sí de manera satisfactoria.

8.2.1 Pruebas unitarias

Estas pruebas consisten en probar los módulos que componen el sistema de forma individual, para una vez garantizado su correcto funcionamiento de forma independiente, realizar las pruebas de integración.

Prueba	PU-001
Objectivo	Validar las funciones de autenticación y registro del usuario.
Descripción	Dada unas hipotéticas condiciones las funciones tienen que comportarse de forma correcta.
Resultado	Superada

68. Tabla – PU-001

Prueba	PU-002
Objectivo	Validar las funciones de consulta de usuario
Descripción	Dada unas hipotéticas condiciones las funciones tienen que comportarse de forma correcta.
Resultado	Superada

69. Tabla – PU-002

Prueba	PU-003
Objectivo	Validar las funciones de modificación del perfil de usuario
Descripción	Dada unas hipotéticas condiciones las funciones tienen que comportarse de forma correcta.
Resultado	Superada

70. Tabla – PU-003

Prueba	PU-004
Objectivo	Validar las funciones de gestión de platos
Descripción	Dada unas hipotéticas condiciones las funciones tienen que comportarse de forma correcta.
Resultado	Superada

71. Tabla – PU-004

Prueba	PU-005
Objectivo	Validar las funciones de gestión de followers.
Descripción	Dada unas hipotéticas condiciones las funciones tienen que comportarse de forma correcta.
Resultado	Superada

72. Tabla – PU-005

Prueba	PU-006
Objectivo	Validar las funciones de geolocalización de usuario
Descripción	Dada unas hipotéticas condiciones las funciones tienen que comportarse de forma correcta.
Resultado	Superada

73. Tabla – PU-006

Prueba	PU-007
Objectivo	Validar las funciones de gestión de pedidos
Descripción	Dada unas hipotéticas condiciones las funciones tienen que comportarse de forma correcta.
Resultado	Superada

74. Tabla – PU-007

Prueba	PU-008
Objectivo	Validar las funciones de gestión de mensajes
Descripción	Dada unas hipotéticas condiciones las funciones tienen que comportarse de forma correcta.
Resultado	Superada

75. Tabla – PU-008

Prueba	PU-009
Objectivo	Validar las funciones de gestión de activaciones
Descripción	Dada unas hipotéticas condiciones las funciones tienen que comportarse de forma correcta.
Resultado	Superada

76. Tabla – PU-009

Prueba	PU-010
Objectivo	Validar las funciones de gestión de valoraciones
Descripción	Dada unas hipotéticas condiciones las funciones tienen que comportarse de forma correcta.
Resultado	Superada

77. Tabla – PU-010

8.2.2 Pruebas de integración.

A continuación vamos a listar todos los 'End Points' realizados para la correcta comunicación del sistema.

Prueba	PI-001
End Point	GET /api/v1/background
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

78. Tabla – PI-001

Prueba	PI-002
End Point	GET /api/v1/login
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

79. Tabla – PI-002

Prueba	PI-003
End Point	GET /api/v1/signup
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

80. Tabla – PI-003

Prueba	PI-004
End Point	GET /api/v1/facebook
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

81. Tabla – PI-004

Prueba	PI-005
End Point	GET /api/v1/email
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

Prueba	PI-006
End Point	GET /api/v1/user/:id
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

82. Table – PI-006

Prueba	PI-007
End Point	GET /api/v1/user/picture
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

83. Table – PI-007

Prueba	PI-008
End Point	GET /api/v1/activate
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

84. Table – PI-008

Prueba	PI-009
End Point	GET /api/v1/ setClientLocation
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

85. Table – PI-009

Prueba	PI-010
End Point	GET /api/v1/ messages
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

86. Table – PI-010

Prueba	PI-011
End Point	POST /api/v1/ messages
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

87. Table – PI-011

Prueba	PI-012
End Point	GET /api/v1/ following/dishes
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

88. Table – PI-012

Prueba	PI-013
End Point	GET /api/v1/following/users
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

89. Table – PI-013

Prueba	PI-014
End Point	GET /api/v1/ dish/:id
Descripción	Validación en su conjunto de este punto REST.

Resultado	Superada

90. Table – PI-014

Prueba	**PI-015**
End Point	POST /api/v1/ dish/:id
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

91. Table – PI-015

Prueba	**PI-016**
End Point	PUT /api/v1/ dish/:id
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

92. Table – PI-016

Prueba	**PI-017**
End Point	DELETE /api/v1/ dish/:id
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

93. Table – PI-017

Prueba	**PI-018**
End Point	GET /api/v1/order
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

94. Table – PI-018

Prueba	**PI-019**

End Point	POST /api/v1/order
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

95. Table – PI-019

Prueba	**PI-020**
End Point	POST /api/v1/orderDone
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

96. Table – PI-020

Prueba	**PI-021**
End Point	POST /api/v1/review
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

97. Table – PI-021

Prueba	**PI-022**
End Point	GET /api/v1/activate
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

98. Table – PI-022

Prueba	**PI-023**
End Point	POST /api/v1/activate
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

99. Table – PI-023

Prueba	PI-024
End Point	DELETE /api/v1/activate
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

100. Table - PI-024

Prueba	PI-025
End Point	POST /api/v1/ follow
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

101. Table - PI-025

Prueba	PI-026
End Point	POST /api/v1/ unfollow
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

102. Table – PI- 026

Prueba	PI-027
End Point	DELETE /api/v1/ account
Descripción	Validación en su conjunto de este punto REST.
Resultado	Superada

103. Table – PI-027

9. Conclusiones

Una vez finalizado este proyecto, pasamos a comentarios algunas de las conclusiones que podemos extraer de todo ello.

Llevar a cabo una plataforma web de dichas dimensiones, ha sido un gran desafío no solo por las tecnologías de reciente incorporación en el mercado con las que se ha desarrollado si no también por los nuevos desafíos que esto suponía. La web está cambiando, y está cambiando para quedarse con más fuerza, sustituyendo a las aplicaciones nativas de todo tipo de plataformas. Las capacidades hardware de los dispositivos actuales así como el buen diseño e implementación de los estándares en los navegadores webs hacen posible el desarrollo de aplicaciones webs que puedan competir directamente con aplicaciones nativas en todos los aspectos.

9.1 Futuras líneas de desarrollo

El objetivo del proyecto ha sido desarrollar un primer producto sencillo con las funcionalidades fundamentales para ir poco a poco y a medida que la gente va conociéndolo, mejorándolo en todos los aspectos así como integrarlo en dispositivos móviles o tablets.

Este sistema es el que persigue la metodología LEAN, cuyo principio es sacar un producto mínimo viable para seguir mejorándolo teniendo el cuento el feedback que el comensal está realizando sobre el proyecto y así digirir el desarrollo hacia las necesidades del usuario.

10. Bibliografía

1. https://angularjs.org/
2. https://www.mongodb.org/
3. https://jquery.com/
4. http://www.w3schools.com/
5. https://www.javascript.com/
6. https://nodejs.org/
7. http://expressjs.com/
8. https://www.wikipedia.org/

11. Anexos

11.1 Manual de instalación

Este manual está orientado a la instalación del entorno de desarrollo en Ubuntu 14.04

11.1.1 Instalando Node.js

En primer lugar debemos instalar Node.js.

Para ello, primero agregamos el repositorio de Node.js, escribiendo lo siguiente en un terminal de Ubuntu

```
curl --silent --location https://deb.nodesource.com/setup_0.12 | sudo bash -
```

y seguidamentos podemos instalar node.js

```
sudo apt-get install -y nodejs
```

11.1.2 Instalando MongoDB

Una vez instalado el servidor web, procedemos igual con la base de datos MongoDB.

Agregamos el repositorio oficial

```
echo "deb http://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.0 multiv
erse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.0.list
```

Actualizamos la base de datos local de paquetes apt

```
sudo apt-get update
```

e instalamos la última versión de MongoDb

```
sudo apt-get install -y mongodb-org
```

Una vez instalada la base de datos, tan solo tendremos que iniciarla.

```
sudo service mongod restart
```

11.1.3 Inicializando Cookify

Una vez instalado tanto el servidor web como la base de datos, tenemos que proceder a instalar las dependiencias y librerías que requiere la aplicación.

Nos situamos en el directorio principal del proyecto y ejecutamos el siguiente comando en el terminal

```
Npm install
```

Con ello estamos instalando las dependencias de Node.js

Y con Bower instalamos las dependencias del cliente.

```
Bower install
```

A continuación solo faltaría iniciar el servidor web

```
Node server.js
```

Donde server.js es el fichero principal del servidor.

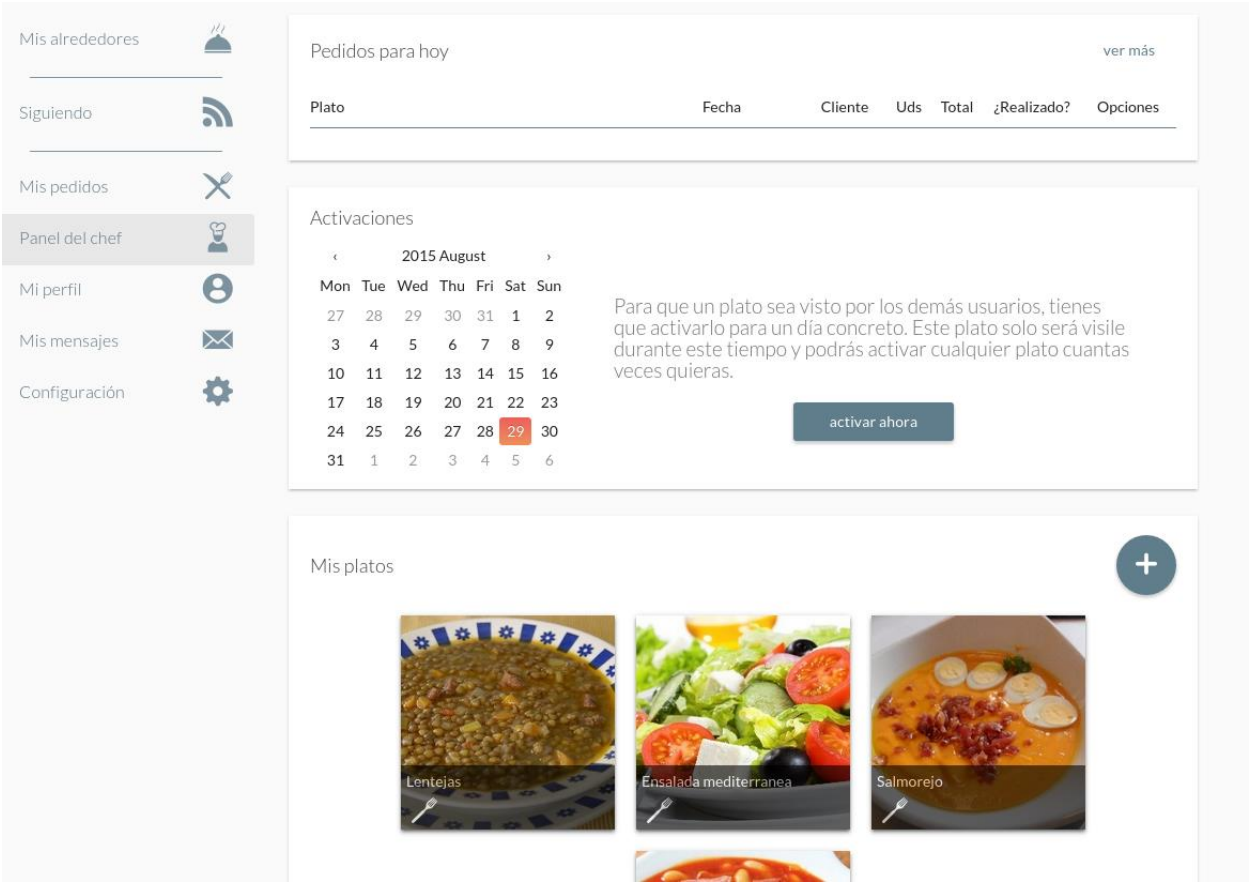
11.2 Manual de usuario

Cookify es una plataforma web de comidas caseras a domicilio en los que el usuario puede jugar dos roles diferences, como chef o cocinero o como comensal.

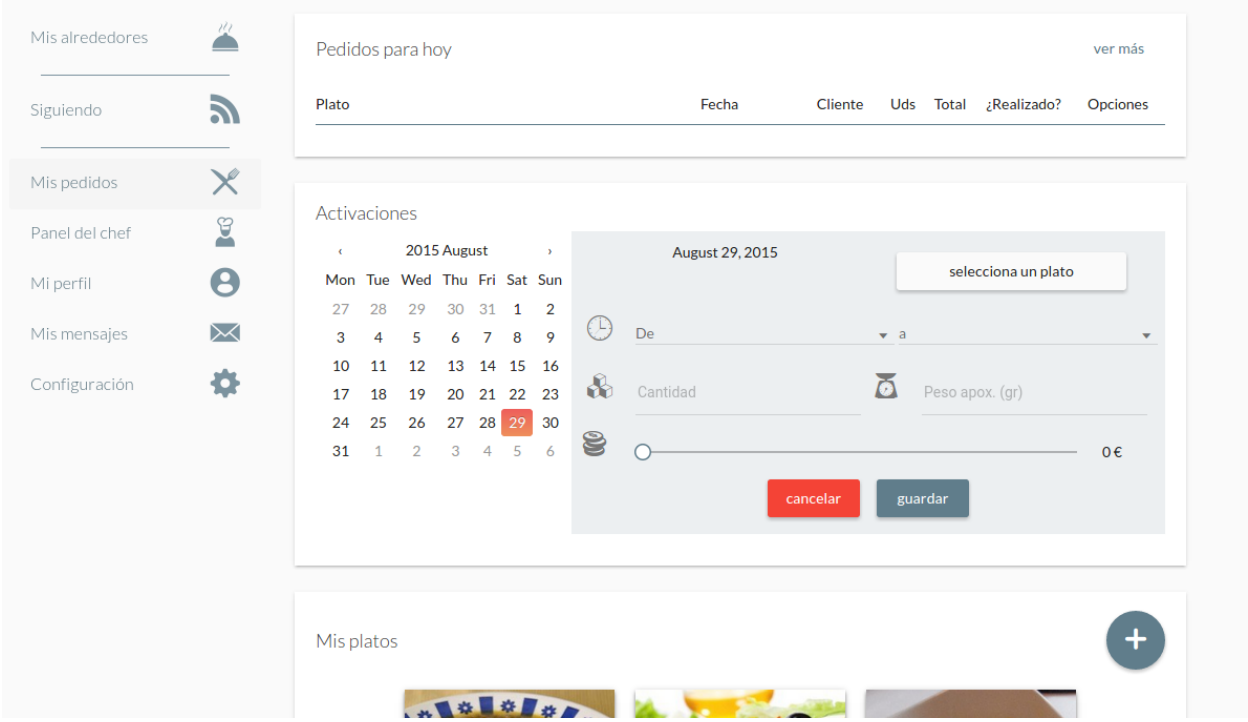
11.2.1 Cocinero

El objetivo principal del cocinero es publicar platos para venderlos en forma de tupper a los vecinos que lo soliciten previamente.

Todo usuario tiene un panel, donde puede crear, editar y eliminar cualquier plato.

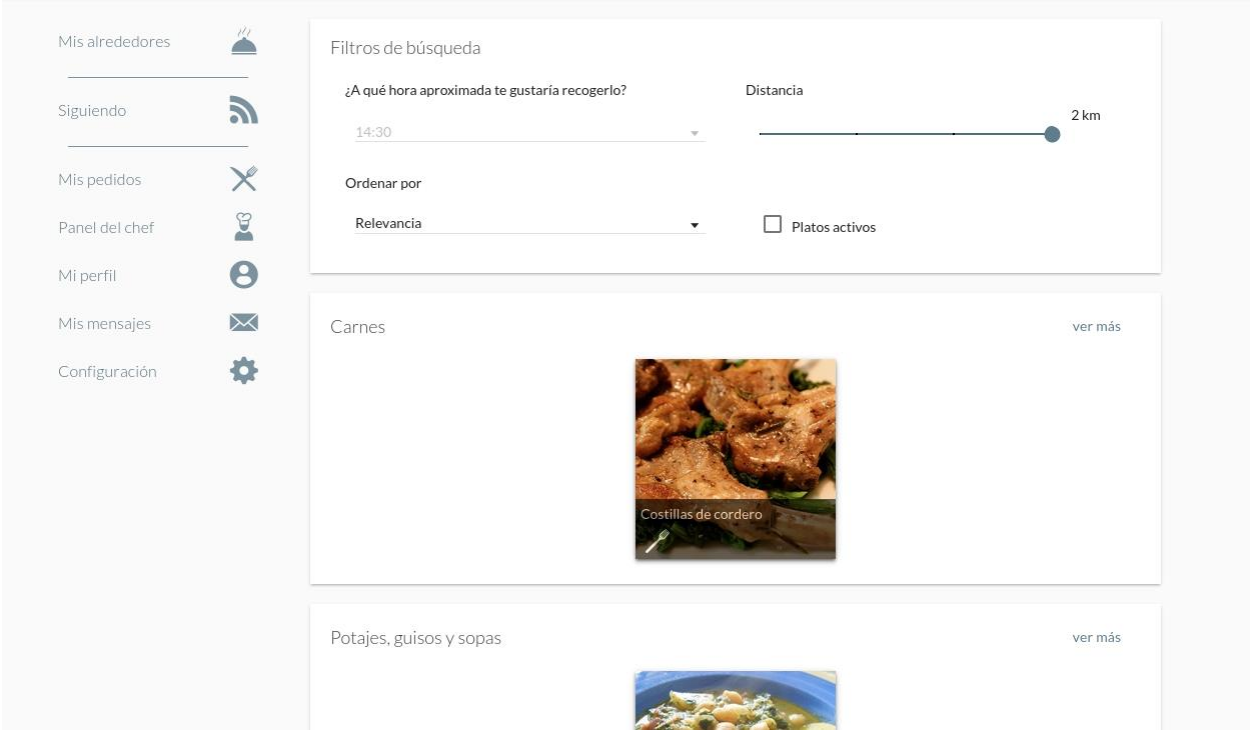


Una vez creado un plato, es te plato está inactivo, esto quiere decir que ningún otro usuario puede solicitar este plato. Para recibir pedidos, se tiene que activar el plato mediante el siguiente formulario



En el que tendrás que especificar el plato, la fecha y horario en el que aceptas pedidos, la cantidad de platos, un peso aproximado para informar al solicitante y el precio.

Una vez activado, cualquier vecino que se encuentre en tus inmediaciones, verá el plato que has activado en la pestaña “Mis Alrededores”



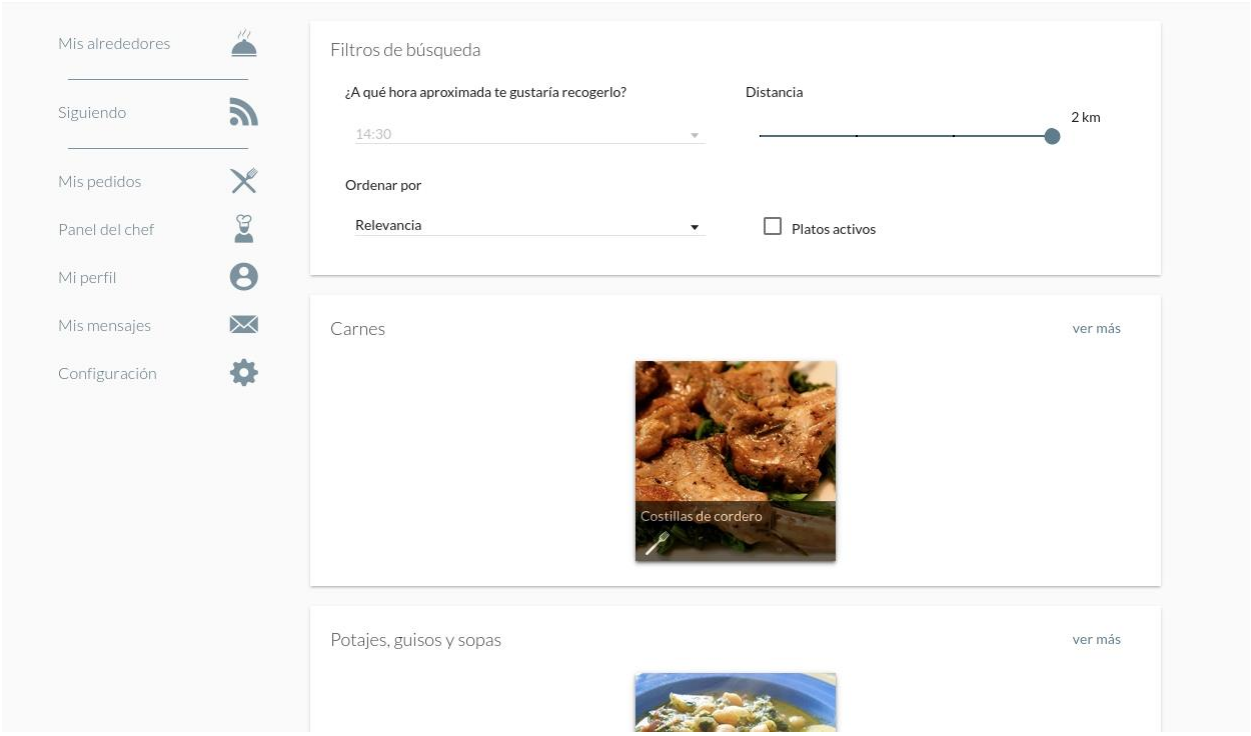
Una vez que un usuario reserve un plato tuyo, saldrá reflejado en la tabla “pedidos” del panel del cocinero.

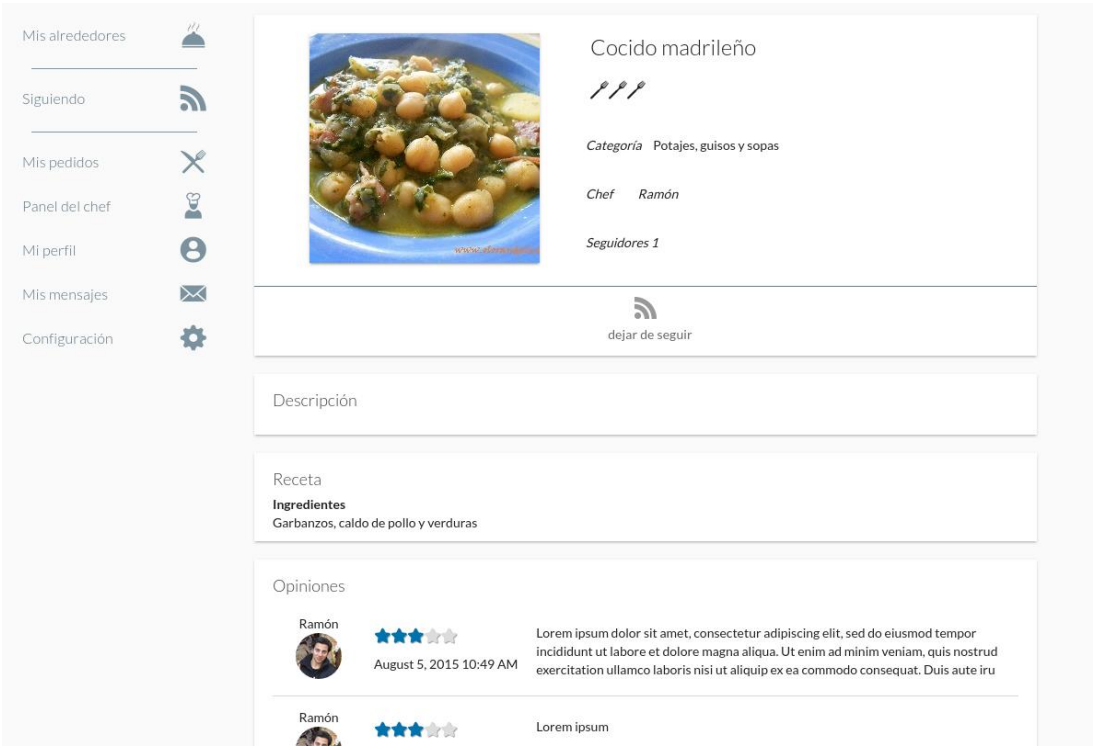
Cuando llegue la hora, el solicitante tendrá que acercarse a tu domicilio para recoger el plato a cambio del precio que le hayas puesto.

11.2.2 Comensal

La función principal del usuario comensal, es la de reservar platos que se encuentren cerca de su geolocalización y recogerlos en casa del cocinero.

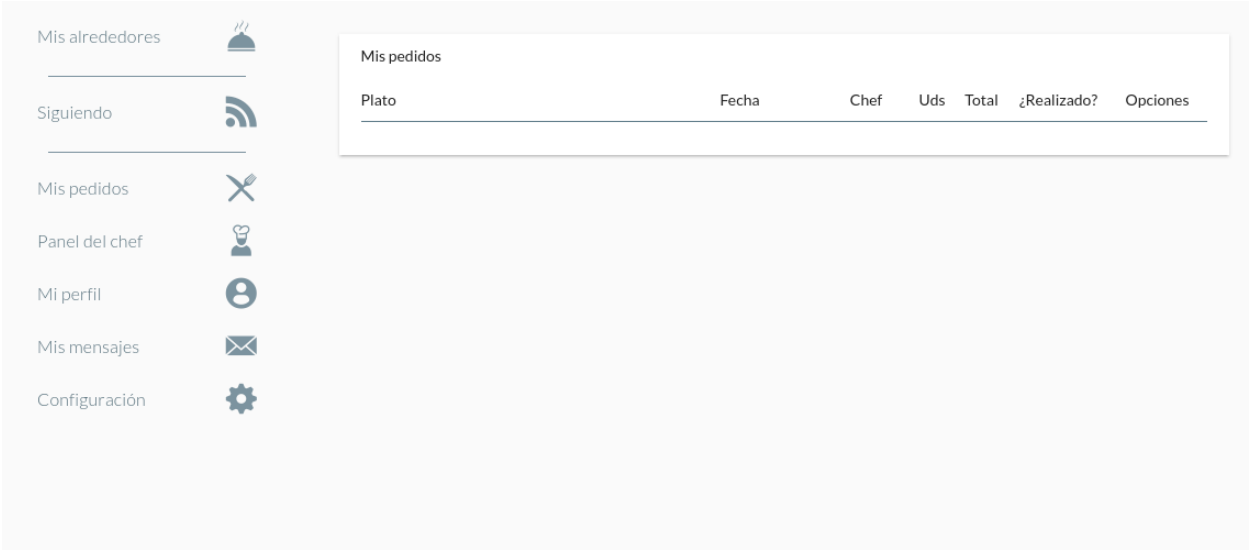
Todos los platos que puede reservar se encuentran en la pestaña “Mis alrededores”.





Si seleccionas un plato desde la pestaña “Mis alrededores” es que se encuentra activo y en cuyo caso aparecerá un pequeño cuadro pudiendo hacer la reserva.

Una vez reservado, aparecerá el resultado en la pestaña “Mis pedidos”



También como usuario, a parte de realizar pedidos de platos, puedes seguir a otros platos y usuarios, y en cuyo caso se te notificará cuando se modifique o agrege un nuevo plato en el caso de seguir a un cocinero, o cuando esté disponible en el caso de seguir a un plato.