Escuela
Politécnica
Superior

NjoyCooking

Grado en Ingeniería Multimedia

Trabajo Fin de Grado

Autor:
Luis Cobo García
Tutor/es:
Pedro A. Pernías Peco
Manuel Marco Such

Diciembre 2016

Universitat d'Alacant
Universidad de Alicante

NjoyCoking

Web de geolocalización de recetas

Autor

Luis Cobo García

Directores

Pedro Pernías Peco

Lenguajes y sistemas informáticos

Manuel Marco Such

Lenguajes y sistemas informáticos

Grado en Ingeniería Multimedia

Escuela
Politécnica
Superior

ALICANTE, 26 de noviembre de 2016

Preámbulo

Desde mis inicios en el grado de Ingeniería Multimedia, siempre me ha despertado mucho interés todo lo relacionado con el desarrollo web y con la gestión de contenidos. En otros campos, la programación se volvía monótona y aburrida, mientras que con la programación web crecía mi interés y las ganas de aprender, debido a la manera tan gráfica y sencilla como se representa la información. Esto, más el factor extra de mi interés por el diseño web, hicieron que empezara a desarrollar mis primeros proyectos web.

Conforme han ido pasando mis años de estudio, he ido conociendo nuevas herramientas para desarrollo, nuevos lenguajes de programación web y nuevas técnicas que me han ido proporcionado una base sólida. Fue tal mi interés por el tema web que, durante la realización de mis prácticas de empresa, aprendí algunas técnicas basicas de marketing digital. Aunque sin profundizar mucho en cada una, había aprendido las bases de cada una de las fases del desarrollo web integral.

Entonces llegó el momento de presentar mi propuesta de Trabajo de Grado. Fue entonces cuando, el que luego sería mi tutor: el profesor Pedro Pernías, me propuso una idea de proyecto interesante que yo decidí desarrollar, aplicando los conocimientos que ya tenía y englobando todas las fases de un proyecto web: planificación,arquitectura,diseño,desarrollo y marketing.

Durante este trabajo aplicaré asignaturas como Usabilidad y Acesibilidad, Negocio Multimedia, Análisis y Especificación de Requisitos, Sistemas Multimedia, Programación

Hipermedia 1 y 2, Sistemas Multimedia Avanzados.

Agradecimientos

Me gustaría empezar dando las gracias a mi tutor Pedro Pernías. Por sus ideas y consejos que me ha proporcionado durante todo este aprendizaje. Y por su entusiasmo y pasión que me ha transmitido en todas sus clases. Sin él este trabajo no habría cobrado vida.

En segundo lugar, agradecer también a mis padres Luis y Raquel, todo el apoyo e interés que me han demostrado durante el desarrollo de este proyecto. Y agradecer todos sus consejos que me han proporcionado durante estos años de carrera.

También quiero dar las gracias a la empresa 3dids.com empresa donde he trabajado durante 7 meses, por todo lo que me ha enseñado, ya que sin esos conocimientos no podría haber desarrollado el proyecto con tanta fluidez.

Por último agradecer a mi compañero de clase Pablo Pernías, por sus consejos de desarrollo y sugerencias de ampliación y mejora de la aplicación que han servido de gran ayuda para mejorar el proyecto.

Índice general

Índice de figuras

1 Introducción

1.1. El marketing digital en la web

1.1.1. ¿Qué es el marketing digital?

El marketing digital son técnicas y estrategias de comercialización usando medios digitales, tales como dispositivos móviles, televisiores digitales y ordenadores.

¿Cuáles son las principales diferencias respecto al marketing tradicional?

- **Personalización**: El marketing digital pretende obtener información del usuario más personalizada. Por ello, aplica técnicas que permiten que sugerir a los internautas información sobre aquello en lo que está interesado, basando estas recomendaciones en búsquedas previas o en sus preferencias definidas.
- **Masivo**: Se puede obtener un gran número de usuarios que formen parte del público objetivo invirtiendo mucho menos dinero que en el marketing tradicional.

1.1.2. Evolución

El concepto de marketing digital ha ido ido variando desde sus inicios hasta la actualidad. En los años noventa, con la aparición de los primeros banners, aparecen las primeras técnicas de marketing en páginas web, aunque este primer concepto que se desarrolla es mucho más básico y se basaba principalmente en hacer publicidad hacia los usuarios para captar posibles clientes [8].

Con la llegada de las redes sociales y la aparición de los smartphones, el concepto de marketing cambia: no se basa únicamente en promocionar un producto sino que se pretende crear una estrategia de venta basada en la perspectiva del cliente. Se reconoce e investiga en las áreas que ayuden a que los clientes piensen que sus opiniones importan, para generar fidelidad hacia la marca. Aquí es donde las redes sociales juegan un gran papel, ya que permiten compartir todo tipo de contenido (videos, enlaces, textos) que se asocian a nuestras opiniones y gustos personales. Todo este contenido permite a las empresas conocer más detalladamente a un potencial cliente y así crear estrategías de marketing para fidelizarlo.

1.1.3. Técnicas

Disponer de una web es una de las mejores oportunidades para incrementar el prestigio y visibilidad de una marca y para alcanzar un mayor rango de clientes con efectividad. Para ello, entran en juego diversas técnicas de marketing digital que nos permiten obtener una mayor visibilidad de la marca o producto:

Posicionamiento SEO

El posicionamiento en buscadores mejor conocido como posicionamiento SEO es un conjunto de técnicas que implican una mejora de la página web con el fin de mejorar su posición en los resultados de los buscadores para unos términos de búsqueda específicos [6]. Cuanto mejor esta optimizada la página web obtiene una mejor posición en los buscadores y por tanto una mayor visibilidad. En la figura 1.1 se observan las principales técnicas de posicionamiento SEO, que se describen a continuación:

- **Uso de keywords**. Palabras clave, son un conjunto de datos asociados a la página que tienen relación con una posible búsqueda por parte de los usuarios en un buscador. Se asocian como metadatos a una página de la web y son unos de los elementos más básicos para el posicionamiento SEO.
- **URL amigables**. Usar palabras cortas y amigables como Urls en el sitio web de vez

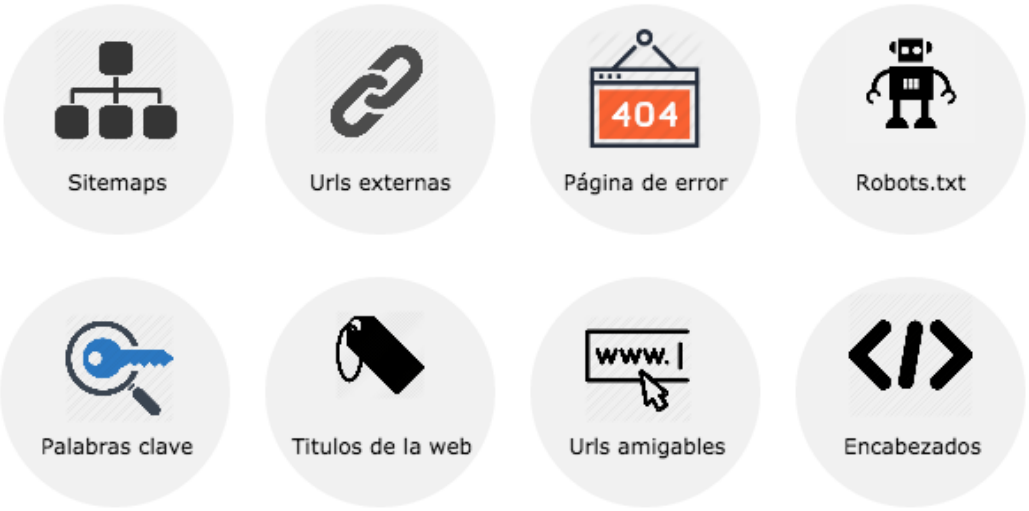

Figura 1.1: Técnicas de SEO

de urls complejas, permite al buscador disponer de palabras clave para interpretar su contenido. Además es mucho más fácil de interpretar por las personas.

- **Uso de etiquetas de título**. Cada página del sitio web, tiene unos metadatos asociados. Una de las etiquetas es el title(título), que indica el nombre de la página web. Es importante que cada página de la web tenga un título diferente y que el texto tenga una información relacionada con la página para facilitar la indexación por parte de los buscadores.

- **Ofrecer un mapa de contenido del sitio a los robots buscadores**. Conocido como sitemaps(en inglés), es una lista de las páginas del sitio con información adicional tal como la importancia de la página o la frecuencia con la que cambia de contenidos. Generalmente los sitemaps se generan como fichero XML.

- **Crear una página de error personalizada**. Una página de error amigable y personalizada mejora la experiencia para el usuario y evita problemas de indexación por parte de los buscadores.

- **Insertar links externos**. Disponer de enlaces desde otras páginas consideradas por los buscadores también favorece el buen posicionamiento de una web.
- **Proporcionar un fichero robots.txt**. Fichero de texto que sirve de guía a los buscadores para rastrear qué información del sitio hay que indexar a fin de posicionarla. Mediante este fichero se delimitan que páginas no queremos que aparezcan posicionadas (página de admin o de política de privacidad por ejemplo) y se facilita la lectura por parte de los crawlers(rastreadores) en el sitio.
- **Ofrecer encabezados de la página**. Se deben utilizar las etiquetas de encabezado(h1,h2,h3) correctamente, estableciendo una jerarquía de la web y, de paso, que sean utilizadas como palabras clave.

La analítica web

La analítica web permite estudiar la repercusión de las campañas de marketing online. Con esta técnica se pretende entender el tráfico del sitio web y así implementar nuevas mejoras.

En los inicios de la analítica web, el objetivo principal consistía en medir el número de visitas a una determinada página. Cuantas más visitas, existía una mayor probabilidad de generar impacto publicitario. En la actualidad se sigue analizando el número de visitas de un sitio, pero la analítica web ha evolucionado y se miden otros indicadores, como la profundidad de las visitas. Las métricas más importantes utilizadas se dividen en dos tipos: básicas y avanzadas.

Métricas básicas: permiten ver el tráfico de usuarios de la web:

Sesiones. Número de visitas que tiene la página.

Visitas únicas. Número de usuarios que han visto la página.

Tasa de salida. Conocer las páginas de la web en las que los visitantes abandonan la web.

Usuarios. Número de usuarios que visitan la web.

Duración de la sesión. Duración media de una sesión en la web.

Métricas avanzadas(KPI): indicadores clave del rendimiento del sitio web. Estás métricas se basan en la comparación de los objetivos marcados por la empresa con los resultados conseguidos. En función del tipo del sitio web, los objetivos serán diferentes y por tanto, los KPIs variaran. Algunos KPI son:

Tasa de conversión . Porcentaje que se obtiene del número de conversiones que se producen a partir del número de visitas. La definición de conversión dependerá del tipo de web. Para un sitio con contenidos las conversiones son el número de registros en la aplicación. Para el caso de un e-commerce la tasa de conversión es la relación entre el número de ventas del sitio y el nº de visitantes.

Fuentes de tráfico . Analiza las fuentes de donde provienen las visitas de los usuarios. Las fuentes de tráfico pueden provenir de diferentes sitios:

- Búsqueda Orgánica(organic search): El usuario ha accedido a la web a través de buscadores como Google o Yahoo.
- Búsqueda de pago(paid search): El usuario accede mediante enlaces directos de anuncios de herramientas de marketing como Adwords.
- Tráfico directo: El usuario introduce la url directamente en el navegador.
- Social: Proviene de redes sociales cómo Twitter,Facebook o Google+.
- Referenciada: Los visitantes provienen de links directos de otras páginas.

Tasa de rebote . Porcentaje de visitas de una sola página. Es decir usuarios que visitan la página pero no interactúan con ella ya que la abandonan nada más visitarla.

Figura 1.2: Panel de Google Analytics

Para monitorizar toda el tráfico de datos obtenido de las métricas aplicadas se usan herramientas de analítica web. Una excelente herramienta gratuita es Google Analytics, que proporciona un dashboard o panel de control muy completo para monitorizar la información de un sitio web [1]. Como se observa en la figura 1.2 en Google Analytics se muestra toda la información de forma gráfica facilitando el análisis de los datos.

La redes sociales

Con la aparición de las redes sociales y su gran aceptación entre los usuarios de internet, las empresas han encontrado una herramienta muy potente para promover sus marcas. Las principales técnicas de social media marketing, adaptadas a las necesidades de cada empresa pueden reportar muchos beneficios:

- **Difusión**: Se consigue una difusión de la información rápida y económica,
- **Recopilación de datos**: se obtiene una gran cantidad de información(BIG DATA) sobre el público objetivo de la marca utilizando una táctica para las redes sociales.
- **Mayor número de visitas**: mediante la difusión de contenido en redes sociales

y con una buena táctica bien orientada, se consigue una mayor visibilidad del sitio web.

Como se ha expuesto anteriormente, del uso de tácticas de social media se obtiene una recopilación de datos relevante para ser utilizada por una empresa, con el fin de afianzar la fidelidad de sus clientes o de conseguir nuevos. Con los datos obtenidos de las redes sociales, las empresas pueden analizar la información y utilizarla adecuadamente para dar una mayor visibilidad a su marca. Un ejemplo de uso de los datos de redes sociales podría ser la obtención de un segmento del mercado, donde los potenciales clientes poseen unas características y necesidades similares(un nicho de mercado) sobre el que la empresa pueda generar un nuevo modelo de negocio [7].

1.1.4. Targeting de las redes sociales

Aunque todas las redes sociales se emplean generalmente para dar una mayor visibilidad, para conseguir un resultado satisfactorio, se debe elegir previamente la red social adecuada que usan los potenciales clientes del producto [9].

Actualmente existen muchas redes sociales enfocadas a diferentes ámbitos: Linkedin(ámbito profesional), Pinterest(ámbito artistico) o Facebook(ámbito social). Cualquiera de las redes sociales actuales se puede emplear como "target" para llegar a un segmento de clientes. Si el 'target" incialmente no esta definido o engloba un grupo de usuarios muy heterogéneo, es más aconsejable escoger una red social con un mayor número de usuarios ya que crea la posibilidad de ir acotando los distintos "targets" existentes y elegir el que más convenga para definir un nicho de mercado. Tres de las redes sociales más usadas en el mundo son: la web de microblogging Twitter, la web social de amistades Facebook y la aplicación de fotografía Instagram.

Twitter probablemente sea la red social más adecuada para empezar a acotar un target sobre el que generar una nueva oportunidad de negocio. La principal ventaja de Twitter son las datos en tiempo real que pueden publicar los usuarios(Tweets). Mediante

los Tweets, las personas expresan sus gustos,emociones, noticias relevantes, etc. Entre los temas o palabras más repetidos por los usuarios en el momento en una región determinada se generán unas tendencias(Hashtags) que son una representación de los intereses comunes de unos usuarios para una región concreta. Cada una de las tendencias puede ser un potencial target sobre el que la empresa puede crear una relacion fuerte con el usuario. Esta es la principal ventaja de Twitter respecto a las otras redes sociales: la diversidad de temas en tiempo real.

1.2. Big Data

Una herramienta que está comenzando a utilizarse para obtener información sobre los usuarios y los mercados para así, generar mejores estrategias de marketing es el Big Data. Con esta técnica, los encargados de marketing de las distintas empresas pueden conocer mejor sus clientes objetivo y mejorar tanto los productos que ofrecen como los mensajes que desean transmitirles.

1.2.1. ¿Qué es el Big Data?

"Big Data" se concibe como la gestión y análisis de grandes cantidades de información. El "Big Data" nos proporciona soluciones que utilizando métodos convencionales llevaría un tiempo mucho más elevado.

1.2.2. Las tres uves del 'BIG DATA"

Todo el volumen de información puede provenir de diferentes tipos de fuentes. En Big data se definen 3 tipos de propiedades de los datos(ver figura 1.3):

- **Volumen de datos**: hace referencia a el tamaño de los datos.
- **Variedad de datos**: se refiere a los distintos tipos de datos
- **Velocidad de datos**: la velocidad con la que se procesan los datos.

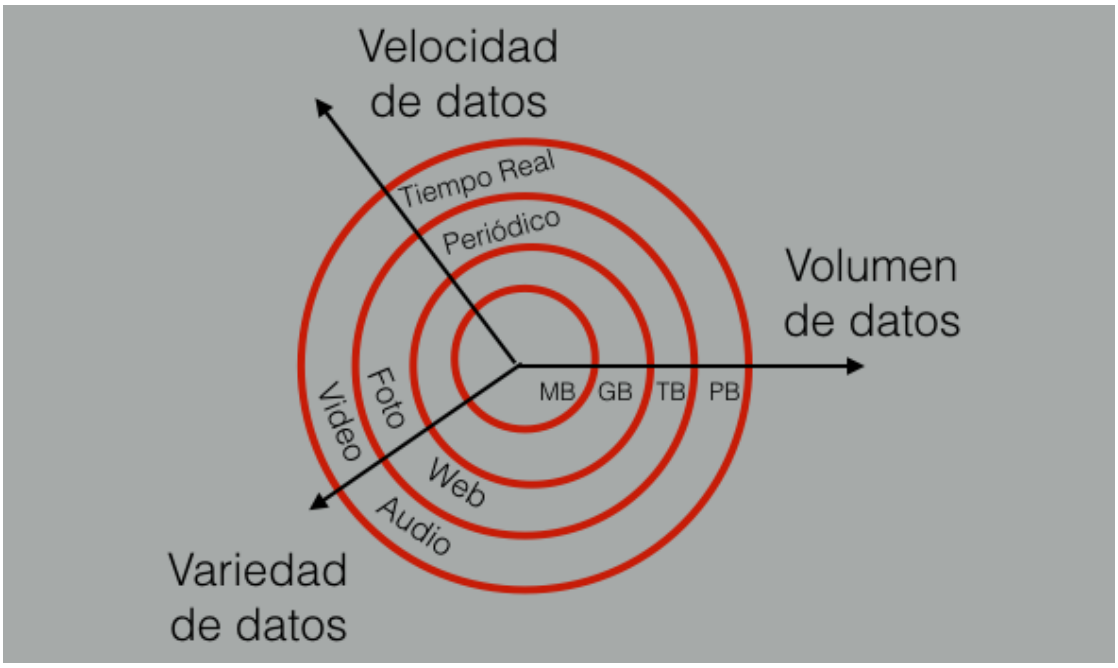

Figura 1.3: Las tres uves del Big Data

1.2.3. Ventajas Principales

- **Reducción de costes**: Las tecnologías de "Big Data" pueden proporcionar una mayor rápidez a la hora de desarrollar el producto y permitir compartir datos de una manera más ágil.
- **Toma de decisiones rápida**: Manejando de forma apropiada la información que proporciona "Big Data" se podrán tomar decisiones más rápidas y eficientes.
- **Nuevos productos y servicios**: Con el análisis de "Big Data" se pueden desarrollar productos que cubran necesidades de los usuarios.

1.2.4. Usos aplicables en las aplicaciones web

Las principales fuentes de datos en la web son los usuarios. Estos proporcionan una información acerca de sus gustos y preferencias que son recogidos y almacenados para posteriormente ser analizados por analistas de datos. Estos datos pueden ser usados en

diversos ámbitos: **Empresarial**, **Deporte** o **Investigación**. En la web el campo más recurrente suele ser el Empresarial.

Las empresas recogen los datos de sus propias webs y de las redes sociales. Utilizando técnicas como la mineria de datos, se obtienen patrones de comportamiento de los usuarios que son usados por las compañías para conseguir mayores beneficios económicos. Un ejemplo sería una web de comercio electrónico(e-commerce) que utilizaría la información de los usuarios para crear anuncios más personalizados incluyendo productos en el que el usuario ya está interesado, aumentando las probabilidades de compra por parte del cliente.

1.3. Planteamiento del proyecto

Actualmente existen millones de aplicaciones web donde existen varios objetivos: desde webs puramente informativas como landings o webs corporativas donde el objetivo principal es dar a conocer algo, hasta un ecommerce donde el principal propósito es la venta de un producto. Para conseguir que se cumplan los objetivos satisfactoriamente, son necesarios algunos factores internos como son la usabilidad o la fluidez de la aplicación. Pero existe un factor externo que es muy imporante para que el proyecto web tenga éxito: la visibilidad, que no es más que conseguir destacar el contenido de el sitio. La principal forma de destacar, es que la web aparezca en los principales buscadores web.

Los buscadores se basan en las técnicas indexación de contenidos para mostrar el listado de páginas web ordenadas según la búsqueda del usuario. Estas se basan en la recogida de datos semánticos de la web(crawling). Pero, ¿Es posible generar visibilidad mediante otra vía que no sean los buscadores y segmentado por áreas?.

Bajo este contexto surge la idea de NjoyCooking. Esta se concibe como una herramienta de marketing que pretende profundizar en el campo de la visibilidad web enfocado a webs de gastronomía y acotandolas sobre las distintas zonas o áreas geográficas.

2 Objetivos

2.1. Objetivo Principal

"Desarrollar una herramienta, orientada para uso en marketing y promoción de un portal de cocina, que permita conocer, según localización geográfica, lo que los usuarios están publicando en distintas redes sociales acerca de temas gastronómicos"

Ello permitirá enfocar mejor las campañas de difusión y promoción de productos de ese sector a un público concreto al localizarlo geográficamente.

2.2. Objetivos Específicos

- Definir los requisitos y especificaciones de la aplicación.
- Estudiar de las diversas API's a utilizar en el proyecto y conocer su uso.
- Diseñar una arquitectura de software capaz de recoger información de las diferentes redes sociales y almacenarla de forma eficaz en la base de datos.
- **Testear y Probar** la aplicación en diferentes navegadores y dispositivos.
- Proporcionar un **producto mínimo viable(PMV)** de la aplicación y desarrollar un modelo de negocio basado en él.

3 Metodología

Para la realización de este proyecto, se estimó una duración de 5 meses donde 1 mes se dedicaría a la investigación, 1 mes para la especificación y diseño y 3 meses para el desarrollo.

Para elegir el marco de trabajo a usar en el desarrollo, se tienen en cuenta algunas de las características de las aplicaciones web:

Escalabilidad La habilidad para manejar el crecimiento continuo del trabajo de manera fluida.

Interoperabilidad La facilidad para comunicarse con diferentes protocolos e interfaces de datos.

Capacidad de Prueba La habilidad para medir el correcto funcionamiento del sistema y sus componentes mediante pruebas.

Con esas características y sabiendo que el desarrollo del sistema se creará de forma iterativa, se opta por utilizar una metodología ágil. Despues de estudiar detalladamente cada una de las metodologías ágiles, finalmente se optó por usar SCRUM [5].

La metodología SCRUM se basa en la realización de pequeños sprints(periodos en el cual se lleva a cabo el trabajo) de 2 o 3 semanas. Al inicio de cada sprint se lleva a cabo una reunión de planificación donde se especifica el trabajo que se va a realizar y se estiman el tiempo en horas que se va a tardar. Cuando

finalize el sprint se realizará una revisión comprobando que se ha completado el trabajo.

Ventajas de trabajar con la metodología SCRUM:

- Reducción de riesgos. Como el alcance está limitado al entregable del sprint comprometido, la aparición de riesgos se limita únicamente a lo que se va a desarrollar.
- Se pueden priorizar los requisitos de la aplicación por valor y coste. En función al valor que aportan que aportan a la aplicación y el coste que supone desarrollarlas, se priorizan para proporcionar el resultado más óptimo en el proyecto. Esta lista de requisitos priorizada se denomina "Product Backlog".
- Flexibilidad y adaptación. Al final de cada sprint se puede aprovechar la parte completada para hacer pruebas y sobre el resultado obtenido tomar decisiones.
- Productividad y calidad. SCRUM se sirve de un proceso de mejora continua, comunicación diaria entre las partes implicadas, estimaciones de esfuerzos conjuntas y entrega de productos de forma regular. Todo ello, con la consigna de mejorar y simplificar la iteración anterior.

4 Desarrollo

4.1. Descripción

4.1.1. Funciones del sistema

La aplicación que se va a desarrollar tiene como objetivo recoger información de las diferentes redes sociales, almacenarlos en una base de datos y mostrarlos en nuestro sitio web. Esos datos almacenados,serán una fuente de información fiable que permitirá conocer que se esta concinando en el mundo en este momento. Para ello se deberá tener en cuenta la gestión de:

- **Gestión de Usuarios**. En la aplicación, se podrán dar de alta usuarios. Existirán tres tipos de roles de usuario:
  - **Administrador**. Tendrá el control de gestionar la información proveniente de las API's. El Filtrado de los datos o la frecuencia con la que realiza las peticiones la aplicación con las API's. También podrá gestionar el contenido del sistema
  - **Colaborador**. Podrá publicar contenido en la aplicación.
  - **Usuario no registrado**. Tendrá acceso a la visualización de los datos de la aplicación.
- **Obtención de datos**. Se podrá consultar información en detalle sobre un elemento de la web.

- **Búsqueda**. Existirán varios métodos que permitirán al usuario filtrar la información deseada.

4.2. Análisis

Para el desarrollo de la aplicación se llevará a cabo previamente un análisis de las diferentes herramientas utilizadas.

4.2.1. Twitter

Twitter es la red social de «microblogging» más conocida actualmente. Esta red social tiene más de 500 millones de usuarios, generando un número aproximado de 65 millones de tweets al dia.

Toda la información proveniente de Twitter es guardada en la base de datos de la aplicación. Para obtener la información, Twitter habilita una REST API que proporciona a los desarrolladores un acceso a la lectura y escritura de gran parte de la información disponible en la red social [15].

Twitter API

```
"statuses" => array:15 [▼
  0 => array:26 [▼
    "created_at" => "Wed Aug 17 09:06:56 +0000 2016"
    "id" => 765837305848401920
    "id_str" => "765837305848401920"
    "text" => "marieclaire_es: Cocina con marieclaire_es Cada día una receta distinta https://t.co/sd7vQ9XNJq https://t.co/X2ZJBuk13g"
    "truncated" => false
    "entities" => array:5 [▼
      "hashtags" => []
      "symbols" => []
      "user_mentions" => []
      "urls" => array:1 [▼
        0 => array:4 [▼
          "url" => "https://t.co/sd7vQ9XNJq"
          "expanded_url" => "http://www.marie-claire.es/lifestyle/recetas"
          "display_url" => "marie-claire.es/lifestyle/rece…"
          "indices" => array:2 [▶]
        ]
      ]
      "media" => array:1 [▶]
    ]
    "extended_entities" => array:1 [▶]
    "metadata" => array:2 [▼
      "iso_language_code" => "es"
      "result_type" => "recent"
    ]
    "source" => "<a href="http://ifttt.com" rel="nofollow">IFTTT</a>"
    "in_reply_to_status_id" => null
    "in_reply_to_status_id_str" => null
    "in_reply_to_user_id" => null
    "in_reply_to_user_id_str" => null
    "in_reply_to_screen_name" => null
    "user" => array:41 [▶]
    "geo" => null
    "coordinates" => null
    "place" => null
    "contributors" => null
    "is_quote_status" => false
    "retweet_count" => 0
    "favorite_count" => 0
    "favorited" => false
    "retweeted" => false
    "possibly_sensitive" => false
    "lang" => "es"
  ]
```


Figura 4.1: Estructura de un tweet

La REST API de Twitter nos proporciona los datos en formato JSON listos para ser procesados. Un Tweet, contiene una estructura(ver figura 4.1) elaborada con toda la información relacionada. La estructura de un Tweet consta de los siguientes elementos:

- **created_at**. Fecha de creación del Tweet, contiene el día, mes, hora, zona horaria y año de cuando se ha creado el Tweet.
- **id**. Identificador único del Tweet
- **id_str**. El identificador del Tweet en formato de cadena de texto. Este campo esta creado para ciertos lenguajes que no soportan enteros mayores de 53 bits, como es el caso de Javascript.
- **text**. Contiene la información del Tweet(máximo de 140 caracteres).
- **truncated**. Booleano que especifíca si el valor del campo text está cortado. El texto truncado terminará con los tres puntos suspensivos.
- **entities**: Proporciona metadatos asociados al contenido del tweet. Entities es un objeto que contiene los siguientes elementos:
  - **hashtags**. Representa un array con los hashtags parseados en el contenido del tweet. Cada hastag contiene un campo de texto con el contenido del hashtag y un array de enteros indicando la posicion inicial y final que ocupa en el Tweet.
  - **media**. Array de objetos que representa el contenido multimedia subido en el Tweet. Este array contiene la url para mostrar a los clientes(display_url). Una versión expandida de la url(expanded_url). El id del fichero y el id en formato de cadena de texto(id_str). Un array con los indíces de la posiciones inicial y final de la url en el texto. Una url apuntando directamente al fichero subido(media_url) y una url para páginas https(media_url_https). Un objeto con los tamaños disponibles para el fichero(sizes).Para aquellos tweets que tiene contenido multimedia asociado a otros tweets, se guarda un

identificador que apunta al Tweet con el contenido original(source_status_id). El tipo de fichero(type) y la url incrustada en el contenido del Tweet(url).

  - **url**. Contiene un array de objetos que representan las urls incluidas en el texto. Contiene la url de visualización, la url expandida y los indices donde se encuentra la url.
  - **user_mentions**. Representa otros usuarios de twitter mencionados en el texto del tweet. Este array tiene el id del usuario mencionado, el id en string, los indices y el nombre(name).

- **metadata**. Array de objetos metadatos asociados al tweet.
  - **iso_language_code**. Codificación iso del lenguaje del tweet.
  - **result_type**. Especifica el tipo de resultado de tweet que quieres recibir. El valor por defecto es recent que devuelve un resultado con los tweets más recientes. También están los valores popular que devuelve los tweets más populares y mixed que incluye ambos tipos mencionados anteriormente.
- **in_reply_to_status_id**. Si el tweet es una respuesta de otro tweet, este campo tendrá el id del tweet orginal.
- **in_reply_to_user_id**. Contendrá el id del usuario del Tweet original.
- **user**. Contiene un array de objetos con toda la información relativa a el usuario que ha publicado el Tweet.
  - **id**. id del usuario que pública el Tweet.
  - **name**. Nombre del usuario.
  - **location**. Cadena de caracteres con el sitio geaográfico al que pertenece el usuario.
  - **description**. Pequeño resumen descriptivo sobre el usuario.
  - **url**. Url proporcionada por el usuario asociada con su perfil.

    - **followers_count**. Número de seguidores que tiene el usuario.
    - **friends_count**. Número de personas a las que sigue el usuario.
    - **protected**. Indica si la cuenta del usuario está protegida, es decir que sus Tweets solo pueden ser vistos con el consentimiento del usuario.
    - **geo_enabled**. Indica si el usuario ha activado la posibilidad de geoetiquetar sus Tweets, es decir, agregar información geográfica en los metadatos del tweet.
- **coordinates**. Representa la posicion geográfica del Tweet. El array de coordenadas esta en geoJSON, donde la longitud es el primer elemento del array y la latitud el segundo.
- **place**. Cuando el Tweet esta asociado a un lugar, pero no necesariamente es orginado de ahí.
- **retweet_count**. Numero de veces que el Tweet ha sido retweeteado.
- **lang**. Idioma en el que esta escrito el Tweet.

Con el análisis de la estructura de un Tweet realizado, se procede a buscar información en Twitter sobre datos relevantes para la aplicación. Para encontrar los datos relevantes que se ajusten a los requisitos para la aplicación, la API de Twitter proporciona unas opciones de búsqueda que permiten buscar los Tweets por palabras clave, idioma, localización y número.

Para el caso de Njoycooking, se pretendía inicialmente buscar Tweets en español de la zona de la península Ibérica que tuvieran la palabra receta, cocina o comida en el Tweet. Para ello utilizaremos los siguientes parametros que Twitter proporciona:

- **q**: parámetro para realizar un consulta de búsqueda por palabras.

- **geocode**: parámetro que permite encontrar Tweets de usuarios localizados en un radio alrededor de un punto central proporcionado en latitud/longitud. Por ejemplo para buscar Tweets de la peninsula establecemos el centro en Madrid que en coordenadas lat/long es (39.8952506,-3.46863775058), y le decimos el radio de alcance, aproximadamente unos 500km.
- **lang**: parámetro que restringe la búsqueda de Tweets al idioma dado. Si solo utilizaramos el parámetro geocode twitter también nos sacaría resultados de Tweets en portugues, para ello se añade este parámetro indicandole que busque por el idioma español(es).
- **count**: parámetro que sirve para definir el número de Tweets a devolver por consulta. Por defecto son 15 y se puede buscar hasta un máximo de 100.
- **until**: devuelve los Tweets creados antes de la fecha dada. Este parámetro no tiene mucho sentido que se utilize, ya que lo que se pretende es buscar los últimos Tweets en tiempo real y no buscar por una fecha determinada.
- **result_type**: parámetro para especificar el tipo de resultado de búsqueda. Los tipos son mixed,recent,popular. Para la aplicación interesa el tipo de resultados recent, ya que nos devuelve los Tweets más recientes.

4.2.2. Google Maps

En la actualidad, con el auge de los dispositivos móviles, cada vez es más fácil encontrar el restaurante deseado o la tienda preferida con una sola búsqueda en nuestro smartphone. A los negocios locales les interesa tener visibilidad en internet para poder darse a conocer con facilidad y tener un mayor número de clientes. Aquí es donde entra en juego Google Maps, herramienta que permite representar de manera precisa en un mapa un negocio, punto de

interés o edificio emblemático de una ciudad.

Google maps es una de las herramientas más potentes actuales para la geolocalización y por tanto es la más utilizada. Proporciona un API que permite incluir un mapa en un sitio web, personalizar iconos y estilos del mapa y además manejar los eventos [11]. La estructura de la API de Maps es la siguiente:

Eventos

Eventos de usuario: Sirven para controlar las acciones que realizan los usuarios y especificar cómo se va a comportar la página ante ellos. Algunos de los eventos de usuario disponibles en la API de maps son los siguientes:

- click: Click izquierdo sobre el elemento.
- rightclick: Click derecho sobre el elemento.
- dbclick: Doble click sobre el elemento.
- drag: Arrastrar sobre el elemento.
- mouseover: El ratón está sobre el elemento que tiene el evento.
- mouseout: El ratón esta fuera del elemento que tiene el evento.

Eventos de cambios de estado: Sirven para controlar las modificaciones de las propiedades de los objetos de la API. Se pueden interceptar estos eventos mediante el controlador addListener().

Tipos de Mapa

Existen cuatro tipos de mapas básicos disponibles en la API de Google Maps. Los mapas básicos son:

- MapTypeId.ROADMAP. Vista del mapa de carreteras. Predeterminado.
- MapTypeId.SATELLITE. Imágenes del satélite de Google Earth.
- MapTypeId.HYBRID. Combina las vistas de satélite y del mapa de carreteras.
- MapTypeId.TERRAIN. Mapa basado en información terrestre.

Para personalizar los mapas básicos la API permite cambiar la visualización de elementos como carreteras, áreas de edificios y parques utiliando estilos. Cada elemento del mapa se especifica mediante el tipo MapTypeStyleFeatureType. Las funciones se especifican con la sintaxis featureType: 'característica'. Las características disponibles son:

- road: selecciona las carreteras(autovías,locales).
- landscape: selecciona los elementos naturales(bosques,terrenos).
- poi: selecciona los puntos de interés(negocios,ayuntamientos,parques).
- administrative: selecciona las áreas administrativas(países,províncias,localidades,etc).
- transit: selecciona toda las estaciones de tránsito de transportes públicos(autobús,aeropuerto,etc).
- water: selecciona las zonas de agua(mares, lagos, ríos).

Cada una de las funciones de características del mapa se compone de varios elementos. Los tipos de elementos disponibles son:

- all: selecciona todos los elementos de la función.
- geometry: selecciona todos los elementos geométricos.
  - geometry.fill: selecciona el relleno de la geometría.
  - geometry.stroke: selecciona el trazo de la geometría.

- labels: selecciona las etiquetas de texto asociadas.
  - labels.icon: selecciona únicamente el icono que se muestra dentro de la etiqueta.
  - labels.text: selecciona el texto de la etiqueta.
  - labels.text.fill: selecciona el relleno de la etiqueta.
  - labels.text.stroke: selecciona únicamente el trazo del texto del a etiqueta.

Una vez seleccionado la característica y su correspondiente elemento que se quiere personalizar, se aplican los parámetros de estilo que son opciones del tipo MapTypeStyler y son las que modifican la apariencia de la característica. Las opciones de estilo disponibles son:

- lightness: valor entre -100(negro) y 100(blanco) indica el porcentaje de brillo del elemento.
- saturation: valor entre -100 y 100 indica el porcentaje de intensidad.
- visibility: especicifica si el elemento aparece en el mapa(on/off) y la forma en la que aparece(simplified). Mediante la visibilidad simplified se eliminan algunas funciones de estilo del elemento.
- color: color del elemento(cadena RGB hexadecimal).
- hue: color básico(cadena RGB hexadecimal).
- gamma: valor entre 0.01 y 10.0, se usan para modificar el contraste en varios elementos.

Marcadores

Los marcadores son unos de los elementos más utilizados y relevantes de la API de Google ya que representan una ubicación exacta en un punto del mapa. Los marcadores son objetos del tipo «Marker» y se incializan mediante

el constructor google.maps.Marker. Al construir un marcador se especifican sus propiedades iniciales:

- position: Objeto de tipo LatLng(latitud,longitud) que define el punto inicial del marcador.
- map: Mapa donde debe representarse el marcador.

Para mostrar contenido en el mapa la API proporciona los objeto InfoWindow que muestran la información en una ventana emergente. Las InfoWindow van asociadas a un marcador y sus parámetros iniciales son:

- content: contiene una cadena de texto o una estructura de elementos donde se muestra la información.
- position: objeto Latlng donde se fija la ventana de información. Normalmente la posición se asocia a un marcador(en este caso, la posición se basa en la del marcador).
- maxWidth: ancho máximo de la InfoWindow en píxeles.
- pixelOffset: compensación de la esquina de la ventana de información a la posición donde se fija la ventana.

GeoCodificación

Google Maps proporciona la clase Geocoder, esta clase permite convertir direcciones reales a coordenadas geográficas, para así poder usar esas coordenadas para posicionar marcadores en el mapa. El Geocoder tiene restricciones permite realizar 2,500 peticiones diarías y 50 por segundo, si se quiere ampliar el número de peticiones se debe ampliar al servicio premium.

Para generar un objeto de geocodificación se inicializa mediante google.maps.Geocoder y el método geocode() inicializa la petición. Al inicializar el geocodifacdor le podemos pasar los siguientes parámetros:

- address: la dirección que quieres geocodificar.
- bounds(opcional): establece un cuadro delimitador.
- region(opcional): código de la región. Este parámetro no restringe los resultados del geocodificador solo influye en la búsqueda del resultado.

El resultado de de la geocodificación es un objeto que contiene varios elementos:

- location: objeto LatLng que contiene las coordenas de latitud y longitud.
- location_type: guarda información adicional acerca de la localización específica. Los valores soportados son: ROOFTOP indicando que el resultado devuelto es geocódigo preciso. RANGE_INTERPOLATED reflejando que el resultado es una aproximación entre dos puntos. GEOMETRIC_CENTER indicando que el resultado devuelto es un centro geométrico. APROXIMATE indicando que el resultado refleja un vaor aproximado,
  - place_id: identificador único del lugar geocodificado.
- postcode_localities[]: array que contiene todas las localidades contenidas en el código postal.

4.3. Especificación de Requisitos

Para este apartado se ha seguido el estándar IEEE 830 de especificación de requisitos.

4.3.1. Requisitos funcionales

Para ordenar los requisitos funcionales, se ha considerado que la forma más adecuada es por tipo de rol y relevancia. Los identificadores de los requisitos

que pertenezcan al rol de administrador empezarán por A, los que pertenezcan al rol de usuario empezarán por U, los que pertenezcan al sistema empezarán por S. Para ordenar por relevancia, se especificán tres casos: Los requisitos que sean funcionalidades básicas de la aplicación empezarán por FB, los requisitos que sean funcionalidades secundarias empezarán FS.

Identificador	FB-U-01
Actor involucrado	Usuario
Nombre	Ver listado de entradas
Descripción	Desde la página de blog o un buscador se podrá acceder a ver un listado de las entradas del blog. Cada elemento del listado contendrá una foto de la entrada, un título y un resumen
Requisitos lógicos	La información estará normalizada, existiendo una serie de datos comunes a todas las entradas, pero podrán existir ciertos datos adicionales según el tipo de entrada. También podrán existir una serie de datos generados por usuarios, como valoraciones y comentarios.

Identificador	FB-U-02
Actor involucrado	Usuario
Nombre	Ver detalle de entrada
Descripción	En el detalle de la entrada, se mostrará una información relativa a la notícia tal como su título, descripción e imagen.
Requisitos lógicos	Igual que en el requisito FB-U-01

Identificador	FB-U-03
Actor involucrado	Usuario
Nombre	Comentar Entrada
Descripción	El usuario podrá añadir un comentario opinando sobre la notícia.
Requisitos lógicos	Para los usuarios no registrados será necesario ademas del comentario añadir un nombre de usuario.

Identificador	FB-U-04
Actor involucrado	Usuario
Nombre	Ver listado de tweets
Descripción	Solicitar datos de los tweets geoposicionados
Requisitos lógicos	En el geoposicionado de cada tweet se mostrará la imagen(si tiene) el nombre del usuario que ha escrito el tweet, su contenido y un enlace a la web del autor(si tiene).

Identificador	FB-U-05
Actor involucrado	Usuario
Nombre	Buscar listado de tweets
Descripción	El usuario podrá buscar los tweets que aparecen en el mapa según unos parámetros.
Requisitos lógicos	Los parámetros aceptados son

Identificador	FB-U-06
Actor involucrado	Usuario
Nombre	Login
Descripción	El usuario podrá loguearse en la aplicación como colaborador o administrador mediante un formulario de registro.
Requisitos lógicos	Para loguearse será necesario introducir los campos de usuario y contraseña en el formulario.

Identificador	FS-U-07
Actor involucrado	Usuario
Nombre	Valorar la entrada
Descripción	El usuario podrá valorar la entrada mediante una puntuación que podrá asignar en el detalle de la entrada
Requisitos lógicos	La puntuación tendrá unos valores númericos del 1 al 5.

Identificador	FB-A-01
Actor involucrado	Administrador
Nombre	Dashboard
Descripción	El administrador al iniciar sesión accederá a un dashboard donde encontrará todo el contenido de la web que podrá gestionar.

Identificador	FB-A-02
Actor involucrado	Administrador
Nombre	Ver listado de admin de entradas del blog
Descripción	En la sección de ver entradas, el administrador tendrá la opción de ver un listado de las entradas disponibles en la aplicación. En el caso de que quiera editar una entrada el usuario deberá ir al detalle.

Identificador	FB-A-03
Actor involucrado	Administrador
Nombre	Gestionar una entrada de blog
Descripción	En el detalle de la entrada, el administrador podrá gestionar el contenido de la entrada
Requisitos lógicos	La gestión del contenido implica la modificación de los campos de la entrada y el borrado.

Identificador	FB-A-04
Actor involucrado	Administrador
Nombre	Moderar comentarios
Descripción	En el detalle de la entrada, se mostrarán los comentarios asociados a la entrada que el administrador podrá moderar si sobrepasan las normas de comportamiento.
Requisitos lógicos	La moderación del comentario implica la edición de su contenido o su borrado.

Identificador	FB-A-05
Actor involucrado	Administrador
Nombre	Gestionar Información Mapa
Descripción	El administrador podrá gestionar toda la información relativa a la página del mapa.
Requisitos lógicos	La gestión de la información del mapa implica: La elección por parte del administrador de la palabra clave para la búsqueda tweets y selección del tiempo con el que se actualiza la página del mapa de tweets.

Identificador	FB-A-06
Actor involucrado	Administrador
Nombre	Logout
Descripción	El administrador cerrará su sesión, mediante un logout que encontrará en el menú principal de la web.

4.3.2. Requisitos no funcionales

Usabilidad. La aplicación deberá ser fácil de comprender y manejar de cara al usuario. Para ello contarará con un diseño responsive adaptado a las distintas resoluciones de los diferentes dispositivos, además de una interfaz cuidada e intuitiva.

Escalabilidad. Las funcionalidades de la web se deberán poder ampliar con facilidad. Además la aplicación deberá ser capaz de hacerse más grande sin perder calidad.

Rendimiento. La web deberá ser capaz de soportar un número elevado de conexiones sin resentirse y responder de manera fluida a todas las peticiones realizadas.

Seguridad. Todos los usuarios registrados se autenticarán mediante el uso de un token genreado aletoriamente. Cualquier dato vulnerable del usuario, como por ejemplo la contraseña, se almacenará cifrado en la base de datos.

4.4. Casos de Uso

Para mostrar las actividades se utilizará una representación mediante casos de uso. Los casos de uso se han segmentado para cada rol de la aplicación. En la figura 4.2 se representan los casos de uso generales para cada rol que se explicarán a continuación:

El **usuario no registrado**. Es el rol por defecto en la aplicación cuando el usuario no tiene ninguna acreditación en la web y carece de permisos. El usuario no registrado podrá solicitar los datos de la aplicación para visualizar el contenido. Por ejemplo podrá solicitar los datos del mapa de tweets o

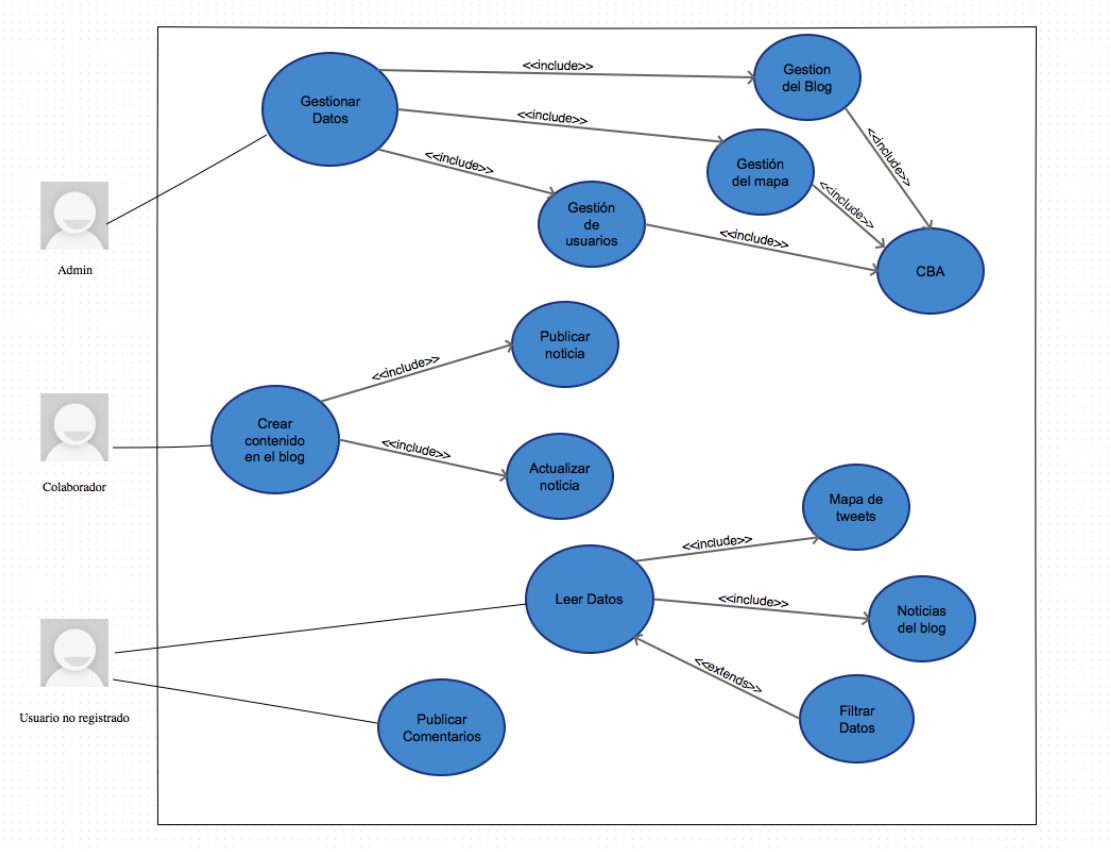

Figura 4.2: Casos de uso generales para NjoyCooking

del blog. Tambíen podrá filtrar la información de los datos solicitados, por palabras clave o categorías. Este usuario podrá también publicar comentarios en el blog.

El **colaborador**. Este rol, además de poder realizar las mismas funciones que el usuario no registrado, tiene unos permisos añadidos. Este usuario tiene la función extra de publicar contenido en el blog. Un usuario colaborador, podrá crear notícias que luego aparecerán en el blog. Podrá actualizar las notícias, pero únicamente aquellas que hayan sido creadas por él. No tendrá permisos para borrar contenido del blog.

Por último esta el **administrador**. El rol de administrador, es el de super-usuario, ya que posee todos los permisos. El administrador es el encargado de gestionar todo el contenido de la web. Dentro del contenido se encuentran los tres principales modelos de datos. El primer modelo de datos son los «Tweets». El administrador se encargará de gestionar la frecuencia con la que se actualiza el mapa de tweets para tener representada la información actualizada. Otra función será modificar la recolección de tweets en base a una palabra clave, que el administrador introducirá manualmente. El siguiente modelo son las notícias, el administrador podrá generar contenido en el blog, creando o borrando cualquier tipo de contenido, tanto propio como de un colaborador. El administrador también podrá moderar los comentarios de las entradas del blog. Finalmente el último modelo de datos serán los usuarios. El administrador gestionará los datos de los usuarios de la aplicación, siendo el encargado tanto de dar de alta a los colaboradores generando un usuario y contraseña como darlos de baja. Toda la gestión del contenido implica crear, borrar o actualizar información de forma que se ha representado mediante el caso de uso CBA.

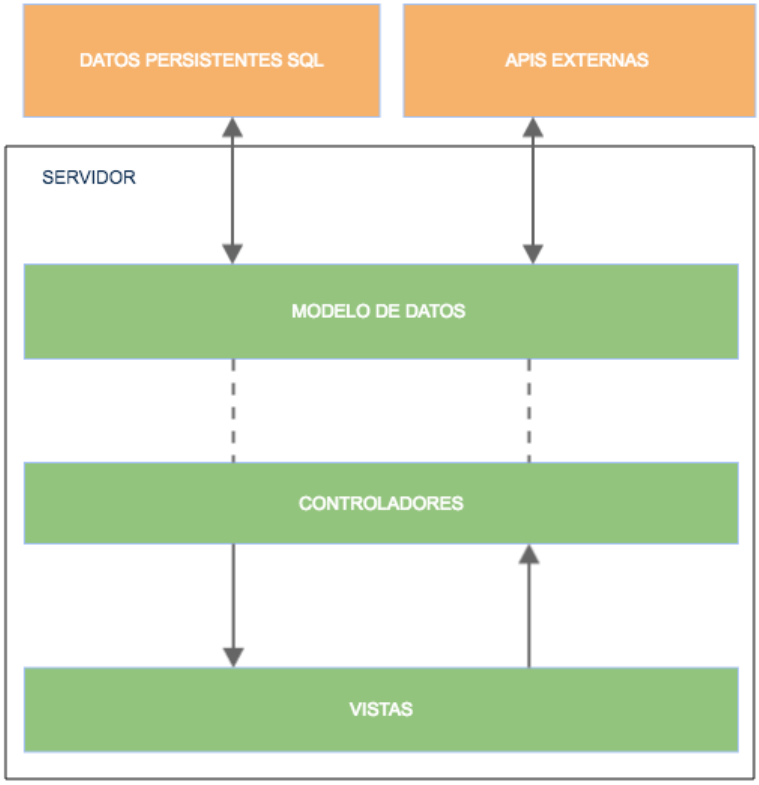

Figura 4.3: Arquitectura de la aplicación

4.5. Diseño de Arquitectura

4.5.1. Arquitectura MVC

La rama de ingenieria del software se preocupa por crear productos robustos y de calidad. Una de las principales soluciones para conseguir esos objetivos es la arquitectura basada en capas, que separa el código en función de sus responsabilidades. Una de las arquitecturas más populares basada en capas para el desarrollo de aplicaciones web, es la arquitectura MVC(Modelo-Vista-Controlador) [10]. En la figura 4.3 se muestra un esquema de la arquitectura de la aplicación.

- **Modelo de datos**. Capa que trabaja con los datos, contiene mecanismos para acceder a la información. A través del modelo accederemos a la base de datos donde,estará la información almacenada. Mediante los modelos también se accederá a los datos de servicios externos que posteriormente se guradarán en la base de datos.
- **Controladores**. Responde generalmente a acciones del usuario, e invoca esas peticiones al modelo. También responde al envío de datos a la vista. Los controladores son el núcleo de la aplicación, ya que es aquí donde se procesan todos lo datos antes de ser enviados a la vista. Un controlador consta de varios métodos, cada método se corresponde con una vista y hace uso de las diferentes clases de la capa de modelos de datos.
- **Vistas**. Es la capa encargada de presentar la información de nuestra página. Los datos provenientes del controlador se presentan en las vistas para que luego ser renderizada por el navegador.

El flujo de la información en la aplicación sería el siguiente. El usuario accederá a una ruta a través del navegador. Cada ruta de la aplicación se corresponde con una vista, que a su vez esta asociada a un controlador. En el controlador correspondiente se llama a las distintas clases de los modelos. Los modelos conectan con la base de datos, esta le devuelve la información, y es recibida y procesada por los controladores que la envían a la vista para que se visualize en el navegador.

Crear una aplicación web basada en la arquitectura MVC nos ofrece ciertas ventajas. Por ejemplo, se puede dividir la lógica del negocio, del diseño del sistema haciendo el proyecto más escalable. Otra ventaja, es la existencia de muchos frameworks basados en MVC como Laravel o Yii Framework que permiten facilitar el trabajo de los desarrolladores. La implementación de URLs amigables, el control del uso de la memoria caché o el control de los

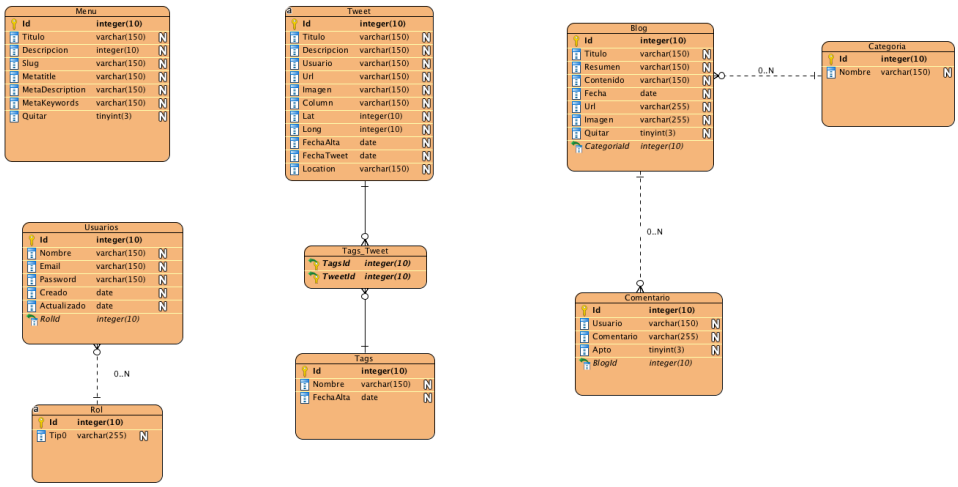

Figura 4.4: Conjunto de tablas del modelo de datos de NjoyCooking

recursos del servidor son tres de las principales ventajas de usar un framework MVC.

4.6. Modelo de Datos

Como se ha comentado en el apartado anterior, el modelo de datos elegido para almacenar la información es el modelo relacional MySQL. El modelo relacional se fundamenta en el uso de relaciones, las relaciones se pueden considerar como un conjunto de datos. Para mostrar una forma más visual esas relaciones se conceptualizan en forma de tablas, que estan formadas por registros y campos. En la figura 4.4 se representa mediante un modelo entidad-relación los conjuntos de datos para la aplicación. Todas las tablas contienen un campo Id que es el identificador de cada tabla(clave primaria) y tienen la clausula de auto-incremento por lo que cada vez que se agregue una fila, MySQL generará otro identificador incrementando en 1 valor más al anterior.

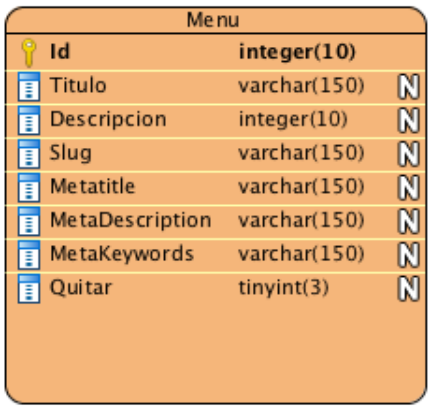

Figura 4.5: Tabla del menú

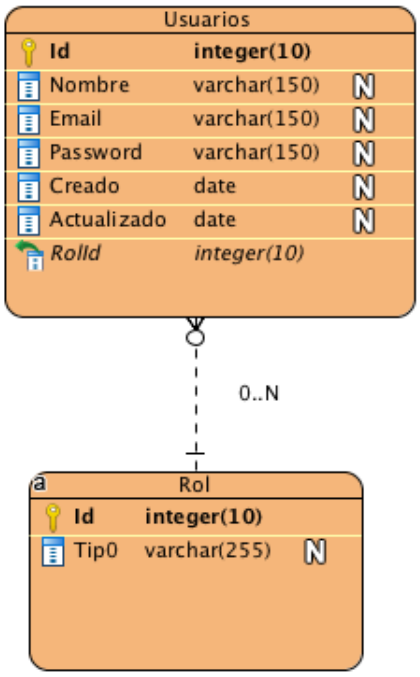

Figura 4.6: Relación de la tabla de Usuarios con la de Roles

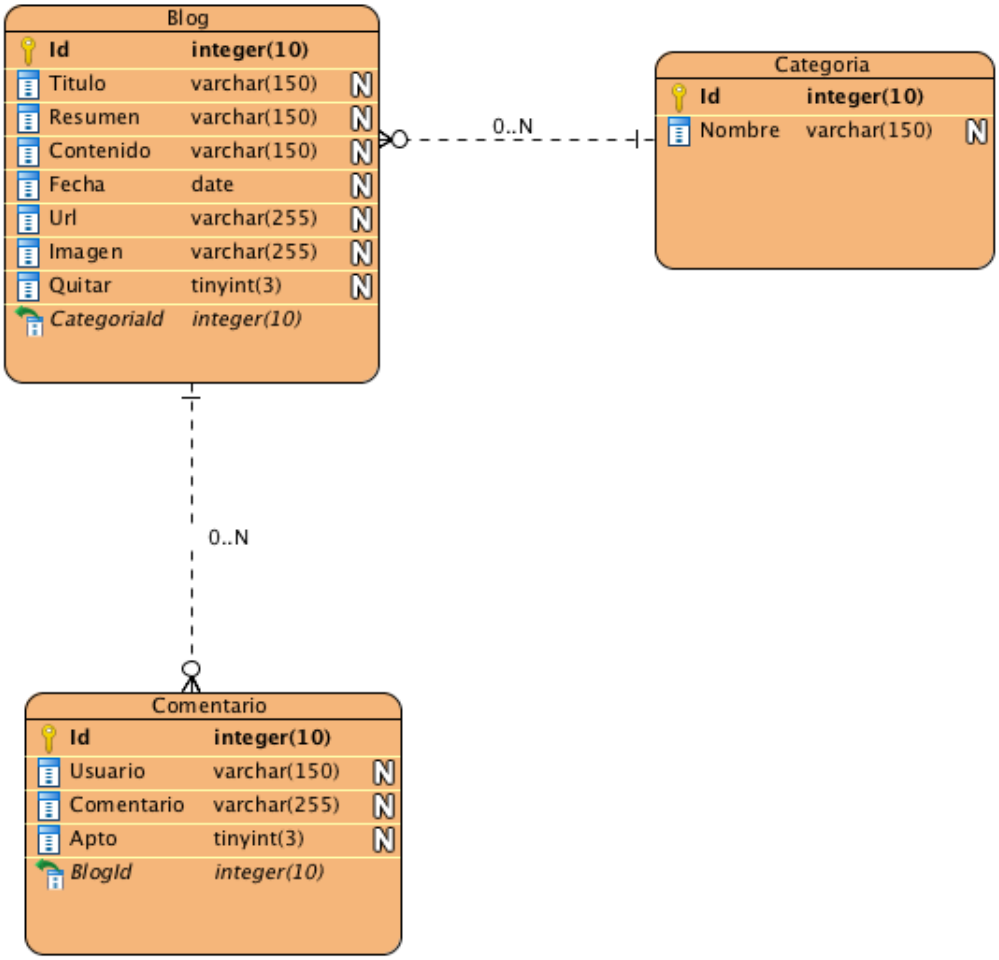

Figura 4.7: Relación de la tabla de Blog con las Categorías y Comentarios

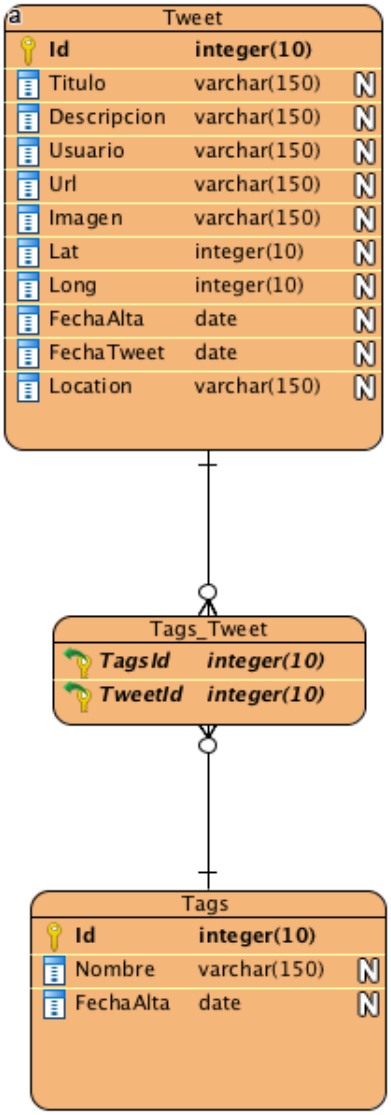

Figura 4.8: Relación de la tabla de Tweets con lo Tags

La primera tabla, y más básica, es la tabla de «Menus», en ella encontramos todos los datos correspondientes al menú de navegación de la aplicación,esta tabla esta pensada para que el menú sea más dinámico y se pueda gestionar de forma más fácil sin modificar código. Como se observa en la figura 4.5, la tabla de «Menus» se compone de los siguientes elementos:

- **Título**: Este es el titulo del menú, y el que aparecerá en la web en la navegación.
- **Descripcion**: Campo para añadir texto introductorio, acerca de la sección.
- **Slug**: El slug corresponde al trozo de la url que aparecerá para esa sección de la web en la aplicación.
- **Campos de MetaAtributos**: MetaKeywords,MetaDescription,MetaTitle, los metas para cada sección del menú. Estos campos son importantes para un buen posicionamiento SEO.
- **Quitar**: Campo de tipo tinyint, que nos permite dejar de mostrar un elemento del menú sin tener que borrar la fila. Por defecto el campo quitar esta a 0 y por tanto mostrará el elemento, en el caso de no se quiera visualizar, se cambia el valor a 1 y ese elemento ya no se muestra en el menu.

En la figura 4.6, se muestra la tabla de Usuarios y su relación con los Roles de la aplicación. Se pretendía que un usuario tuviera un único rol y que un rol pudiera ser adoptado por muchos usuarios. Para representar esa relación se añade un clave ajena en la tabla usuarios generando una relacion uno a muchos y así permitiendonos cumplir las condiciones dictadas previamente. Para la tabla de Usuarios, se generan los siguientes campos:

- **Nombre**. Nombre del usuario, que se muestra en la web.

- **Email**. Correo electrónico, del usuario. Tanto este campo como el nombre sirven de login para el usuario.
- **Password**. Contraseña del usuario. En la base de datos se guarda un hash de la contraseña para mayor seguridad.
- **Creado**. Campo de fecha con la creación del usuario.
- **Actualizado**. Campo de fecha con la actualización del usuario.
- **RolId**. Clave ajena a la tabla de Rol, contiene el Id del Rol.

La tabla de Rol únicamente contiene el campo identificador y el tipo que hace referencia al nombre del rol.

Para la página del Blog 4.7 tenemos el esquema relacional con tres tablas: Blog,Categoría y Comentario. Cada fila de la tabla pertenece a una notícia del blog de la aplicación, cada notícia puede tener uno o más comentarios y ese comentario podrá aparecer solo en una notícia. Por tanto para cumplir esas condiciones se generan dos tablas(Blog y Comentario) con una relación uno a muchos. Las notícias pueden tener categorías, en el caso de la aplicación una noticía puede pertenecer a una categoría o ninguna, por tanto se añade una relación muchos a uno de la tabla Categoría a Blog. Las tuplas para la tabla de Blog son las siguientes:

- **Título**. Título de la notícia.
- **Resumen**. Campo de texto para mostrar un resumen en el listado de notícias.
- **Contenido**. Contenido de la notícia.
- **Fecha**. Campo de Fecha con la creación de la notícia.
- **Url**. Url amigable que aparece en la ruta de la notícia.
- **Imagen**. Campo de texto para la imagen principal de la notícia.

- **Quitar**. Campo quitar, para poder dejar de visualizar la notícia sin necesidad de borrarla de la base de datos.
- **CategoríaId**. Clave ajena. Id de la categoría a la que pertenece la notícia.

Tuplas para la tabla de comentarios:

- **Usuario**. Usuario que comenta la notícia.
- **Comentario**. Campo de texto con el comentario sobre la notícia.
- **Apto**. Campo para moderar el comentario y que sea valido para mostrar. Por defecto vale 1, no se muestra, cuando se cambia a cero se valida y se muestra.
- **BlogId**. Clave ajena con el Id de la notícia al que pertence el comentario.

Para el mapa de tweets se generan tres tablas(figura 4.8). La principal es la tabla Tweet, donde se almacenan todos los tweets que se van a mostrar en el mapa. Los campos son los siguientes:

- **Título**. Título del tweet.
- **Descripción**. Campo de texto que contiene el tweet.
- **Usuario**. Usuario que ha escrito el tweet
- **Url**. Enlace que se encuentra en el tweet y que conduce a una página externa.
- **Imagen**. Imagen del tweet.
- **Lat**. Coordenada de latitud en el mapa.
- **Long**. Coordenada de longitud en el mapa.
- **FechaAlta**. Fecha de registro del tweet.
- **FechaTweet**. Fecha de creación del tweet.

- **Location**. Campo que guarda una localización.

Para clasificar el contenido de los tweets, se crea la tabla Tags, que son los distintas clases en las que se pueden clasificar los tweets. Los campos para esta tabla son, el nombre de la etiqueta y la fecha de registro de la misma.

Según como esta especificado, un tweet puede tener uno o más tags, y pueden pertenecer varios tweets a un tag. Por tanto es una relación muchos a muchos, para cumplir esta relación se crea una tabla adicional(Tag_Tweets), que contiene los identificadores(es decir las claves primarias) de las dos tablas relacionadas(Tweet y Tag).

5 Implementación

5.1. Software

Para diseñar una interfaz de usuario real, con la que establecer la base para posteriormente representarla mediante HTML y CSS, se utilizan diversos programas de diseño web.

- **Sketch**: Sketch es una aplicación de diseño vectorial, que permite diseñar interfaces para aplicaciones móviles o web de una manera sencilla y con una gran potencia.
- **Illustrator**: Esta herramienta de ADOBE , permite el diseño de logotipos de forma vectorial,
- **Spectrum**: Programa sencillo que permite crear paletas de colores, de forma que se puedan seleccionar colores complementarios o de gama monocromática y utilizarlos para la interfaz de la aplicación.

5.2. Tecnologías

Para la realización de la aplicación web, se ha llevado a cabo un análisis de las tecnologías disponibles y se han seleccionado las que mejor se adaptaban a las necesidades del proyecto.

5.2.1. Front-End(Cliente)

Las dos principales e indispensables tecnologías que se usan para la parte del cliente son: el lenguaje de etiquetas HTML5 y las hojas de etilos CSS3. Estas son dos de las tecnologías fundamentales en las que se basa el desarrollo web.

Con la finalidad de conseguir una apariencia cuidada e intuitiva del sitio web, sin la necesidad de crear las hojas de estilos propias, se consideró la idea de usar el framework Bootstrap. Sin embargo, debido al objetivo de personalizar al máximo la apariencia de la web y a que Bootstrat no permitía muchos grados de libertad en ello, se descartó este framework y se optó por añadir clases propias mediante css.

Para maquetar el sitio web con CSS3 de forma más rápida y refactorizable se utiliza **Sass**. Sass es un lenguaje de preprocesado de CSS, que permite escribir CSS de forma más cómoda, posibilitando declarar variables,mixins, herencia de clases, etc [3]. Hay diversas formas de utilizar Sass en un proyecto. Mediante un programa como Prepros, mediante un automatizador de tareas como Grunt, o mediante un terminal con los comandos de Sass. Para el proyecto se ha optado usar el terminal para ejecutarlo, ya que es mucho más ligera y consume menos RAM que con programas como Prepros. Para que empezar a usar Sass en el proyecto, dentro de nuestra carpeta “padre” donde se encuentre el CSS, se crea una carpeta Sass donde se incluirán todos los ficheros .scss. Con el terminal situado en la carpeta padre se escribe sass –watch sass(nombre de la carpeta donde se encuentran los ficheros .scss). Finalmente, al compilarlo, se generará un fichero style.css que será la hoja de estilo a usar.

Para añadir el dinamismo a la web, se ha optado por utilizar un framework de Javascript como **JQuery** que permite simplificar la manera de interac-

tuar con los elementos HTML mediante funciones propias. También permite funciones propias para animaciones y peticiones HTTP. Además, JQuery es software libre [12].

Como última tecnología Front-End se utiliza la API de Google Maps, indispensable para el sitio web ya que la principal proposición de valor de la aplicación es la geolocalización de recetas, y para su representación necesitamos la API de Google.

5.2.2. Back-End(Servidor)

Después de barajar diversos lenguajes para desarrollar el back-end de la aplicación, finalmente se eligió PHP. Además de ser el lenguaje más utlizado para el desarrollo web, es uno de los lenguajes más potentes y flexibles, pudiendo ser utilizado en la mayoría de los servidores web y sistemas operativos. Además PHP esta publicado bajo licencia de software libre, por lo que no supone ningun coste.

Para montar la arquitectura MVC en la aplicación, se utiliza el framwework Laravel [2]. Laravel agiliza el desarrollo de las aplicaciones web, permitiendo multitud de funcionalidades. Con este framework, desarrollado de forma elegante y simple se evita la creación de código espagueti, facilitando su refactorización y/o su modificación. Algunas de las características de Laravel:

- **Plantillas**. Laravel utiliza platillas Blade. Blade permite tener un sitema de vistas modular de forma que se tenga que repetir la menor cantidad de código. Para ello se genera una plantilla base o layout, que es donde se representa la estructura de la web y se volcará el contenido para cada página. Mediante la directiva include(nombre_template), se podrá incluir una vista parcial de contenido HTML, esta directiva se utiliza para contenido que no cambia, por ejemplo para incluir

la cabecera o el footer de la aplicación. Luego mediante la sentencia yield(nombre_template), permitiremos crear una futura sección en el HTML que se definirá en las vistas que son heredadas de este template. Mediante la sentencia extends(nombre_template) le diremos a Laravel que vistas se van a usar como futuras secciones. Con estas sentencias se conseguirá volcar el contenido especifíco para cada página de la web duplicando el menor número de codigo HTML y de forma más modular.

- **ORM**. Es una implementación de registro activo para trabajar con la base de datos de forma que cada tabla de la base de datos tiene un Modelo correspondiente asociado con el mismo nombre. Esta implementación permite también métodos predefinidos para llamar a la base de datos como: save(),create(),get(),find() [13].
- **Caché**. Laravel, cuenta con un robusto sistema de caché, el cual se puede ajustar, para que se produzca una carga rápida de la web y generar una mejor experiencia al usuario.
- **MiddleWare**. Usa HTTP Middleware, que proporcionan un correcto mecanismo para filtrar las peticiones en la aplicación. Un ejemplo de middleware que incluye Laravel, es el usado para verificar si el usuario esta autenticado en la aplicación.

Para la base de datos, se utiliza el sistema de gestión relacional MySQL, ya que es uno de los sistemas más utilizados y con mayor documentación para el desarrollo web. Además, de la perfecta integración con Laravel.

5.2.3. Estructura de la aplicación

Para comenzar a desarrollar el proyecto, primero se debe instalar Laravel. Para ello se utiliza un manejador de dependencias como composer que nos permite instalar los paquetes y librerías de forma automática sin la

necesidad de hacerlo de forma manual. Mediante nuestro terminal escribimos el siguiente código para instalar composer en el ordenador: curl -sS https://getcomposer.org/installer — php. Una vez instalado, para generar un proyecto con Laravel escribimos en el terminal el siguiente comando: composer create-project laravel/laravel nombre-proyecto.

Si vamos a la carpeta raíz con la que creamos el proyecto se observa que dentro composer ha generado una estructura de directorios que es como organiza Laravel el código de la aplicación. A continuación se detallan brevemente cada una de ellas:

- **app**. El directorio app es donde se encontrará la mayor parte del código personal del back-end de la aplicación. Desde los modelos de datos de la aplicación, hasta los controladores pasando por el middleware.
- **config**. Aquí se encuentran los ficheros de configuración de Laravel y de la aplicación. Por ejemplo en el fichero app.php se especifican parametros tales como la zona horaria y el idioma de la aplicación. También se definen los providers, que son cada uno de los objetos o instancias que se cargarán en el proyecto.
- **database**. Aquí se encuentran todos los ficheros relacionados con la base de datos de la aplicación. Dentro encontramos tres subdirectorios:

  - factories: aquí se incluyen los ficheros que generan automaticamente nuevos datos en tu base de datos para testear la aplicación, sin necesidad de generarlos manualmente.

  - migrations: las migraciones son un tipo de control de versiones para la base de datos. A través de estos ficheros se puede modificar la base de datos.

  - seeds: permiten poblar la base de datos con datos de prueba. Los seeds pueden utilizar factories para poblar la base de datos o introudcirlos

manualmente.

- **public**. En este directorio se encuentran los directorios con ficheros estáticos como son las hojas de estilo(css), los ficheros javascript(js) y las imágenes(images).
- **resources**. En resource encontramos subdirectorios donde se encuentran las vistas en formato blade.php de la aplicación(views) y los archivos de idiomas de la aplicación, para poder pasar de un idioma a otro en la aplicación(lang).
- **storage**. Se encuentran varios subdirectorios que contiene el cache de la aplicación, sesiones, etc.
- **vendor**. Este directorio contiene todo el core de Laravel y los componentes instalados.

Estructura Sass

Como se comenta previamente, se utiliza Sass para maquetar la web. Este lenguaje de preprocesado de CSS permite organizarlo y hacerlo más modular de forma que se pueden generar varios ficheros .scss y compilarlos en un único fichero CSS que es el que se utilizará en la aplicación. Para ello se usa una carpeta sass que se incluye en la carpeta public del proyecto donde se incluyen todos los directorios con los ficheros .scss. Una de las formas de refactorizar más el código Sass, es generar un fichero .scss para cada página web de la aplicación de modo que en futuros cambios resulte todavía más sencillo modificar el estilo. La estructura Sass es al siguiente:

- **base**: en el directorio base se encuentran los ficheros scss con los elementos más básicos de la aplicación. Las tipografías usadas, los formatos de texto de los encabezados y párrafos, etc.
- **config**: en este directorio encontramos los siguientes ficheros.

  - variables: scss donde se declaran las variables que se usan en las hojas

de estilo tales como colores de la aplicación, las fuentes usadas.

- functions: aquí se encuentran los mixins de sass que se pueden utilizar. Los mixins no son más que funciones que permiten reutilizar estilos.

- animations: en este directorio se incluyen los ficheros scss con animaciones hechas con css incluidas en la aplicación.

- **ui**: en el directorio ui(user interface) se incluyen los ficheros scss que modifiquen las propiedades css de elementos de la interfaz gráfica de la aplicación como botones,formularios, etc.
- **modules**: debido a que la aplicación esta basada en la filosofía mobile first en esta carpeta se incluyen todos lo ficheros scss de las páginas de la aplicación en resolución móvil(de 0 a 768px de resolución de pantalla).
- **mediaqueries**: en esta carpeta se incluyen los ficheros scss para las resoluciones de tablet y ordenador de escritorio. Para esta resolución se han marcado las resolución de 768px a 1024px(tablet) y para escritorio las resoluciones de 1024px a 1600px y mayores de 1600px. Para cada resolución habrá un subdirectorio con el nombre de la resolución donde se incluirán sus ficheros scss correspondientes.

5.3. Sprints

Una instalado el framework Laravel y comprendida su estructura se procede a desarrollar la aplicación. Para realizar el proyecto, como se especifica previamente se utiliza la metodología Scrum. El proyecto se divide en los siguientes sprints:

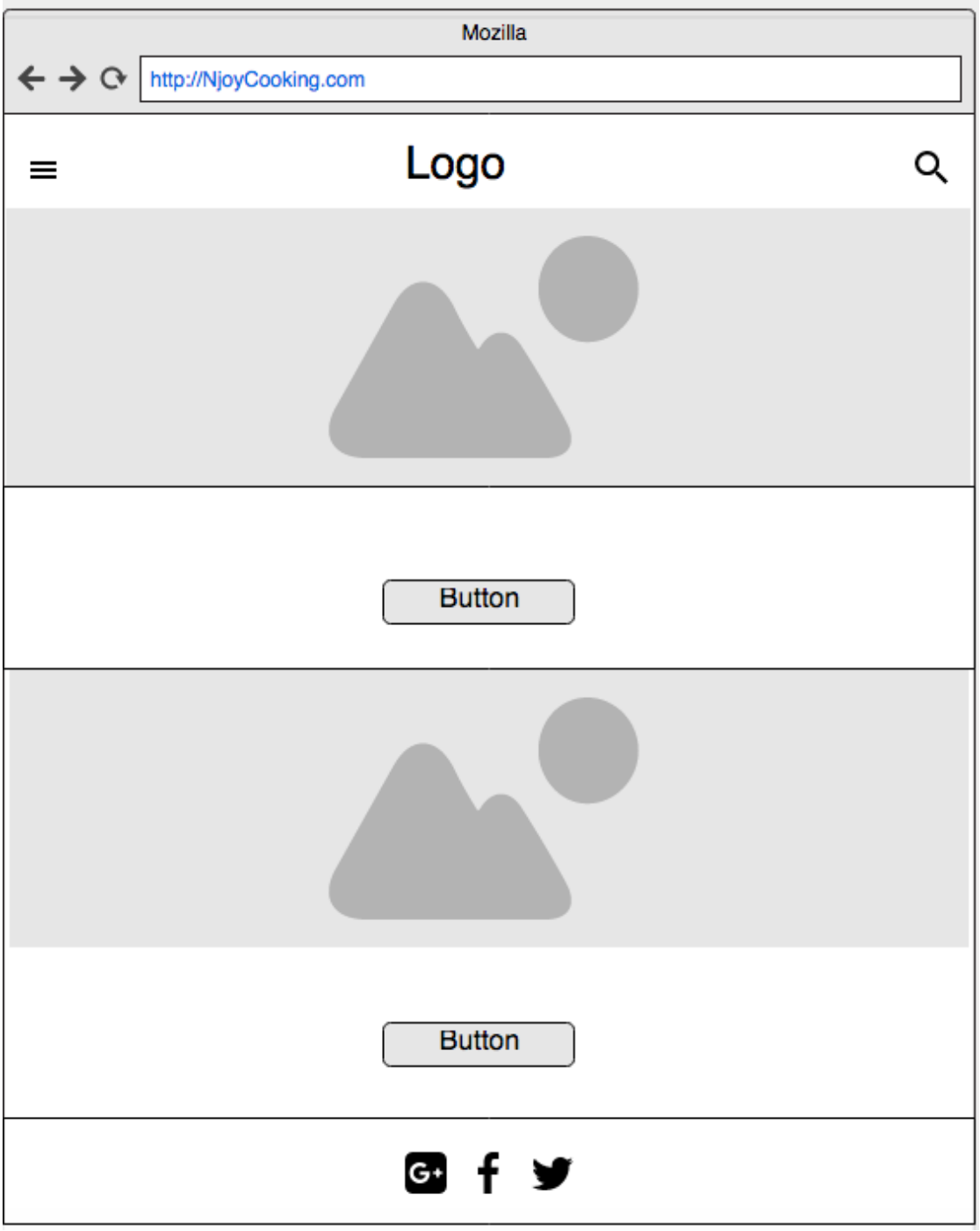

Figura 5.1: Diseño de la página de captación o Landing

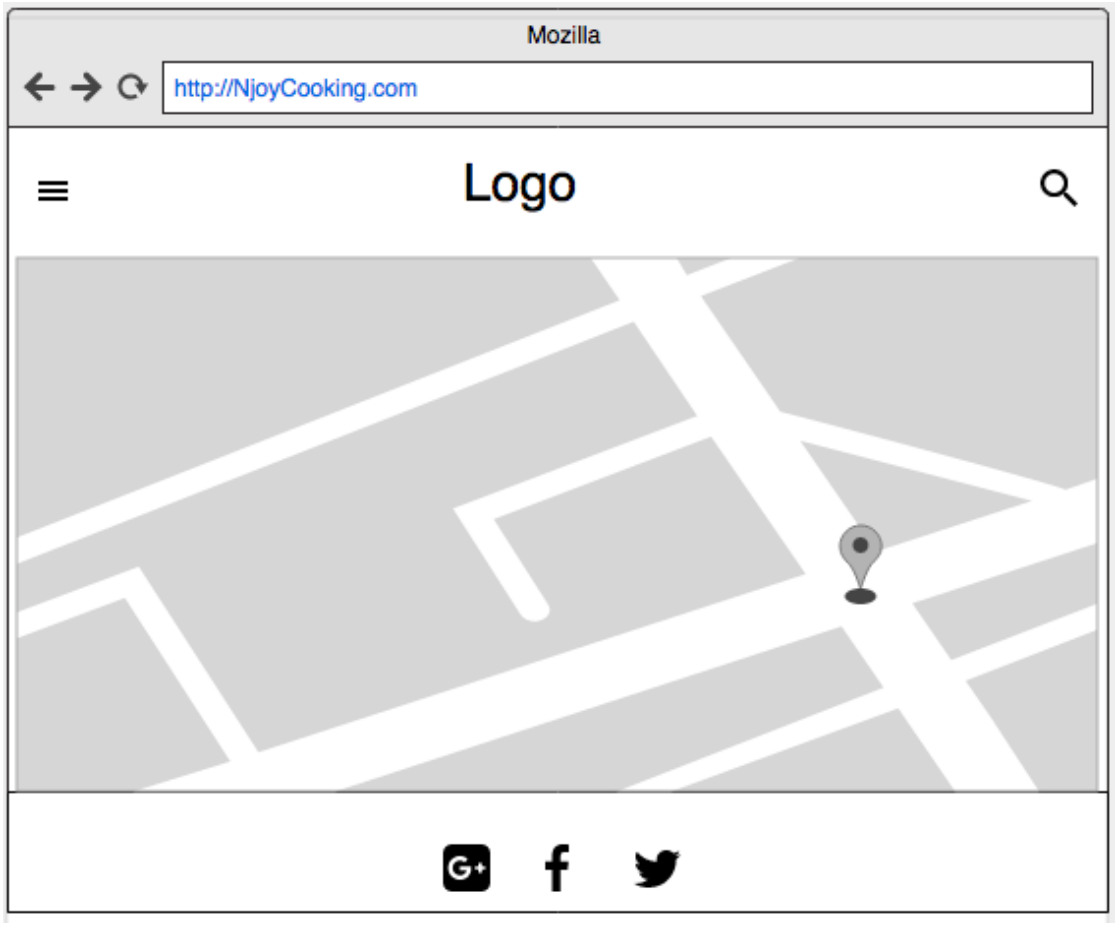

Figura 5.2: Diseño de la página del Mapa de la aplicación

Figura 5.3: Listado de noticias

Figura 5.4: Detalle de noticia

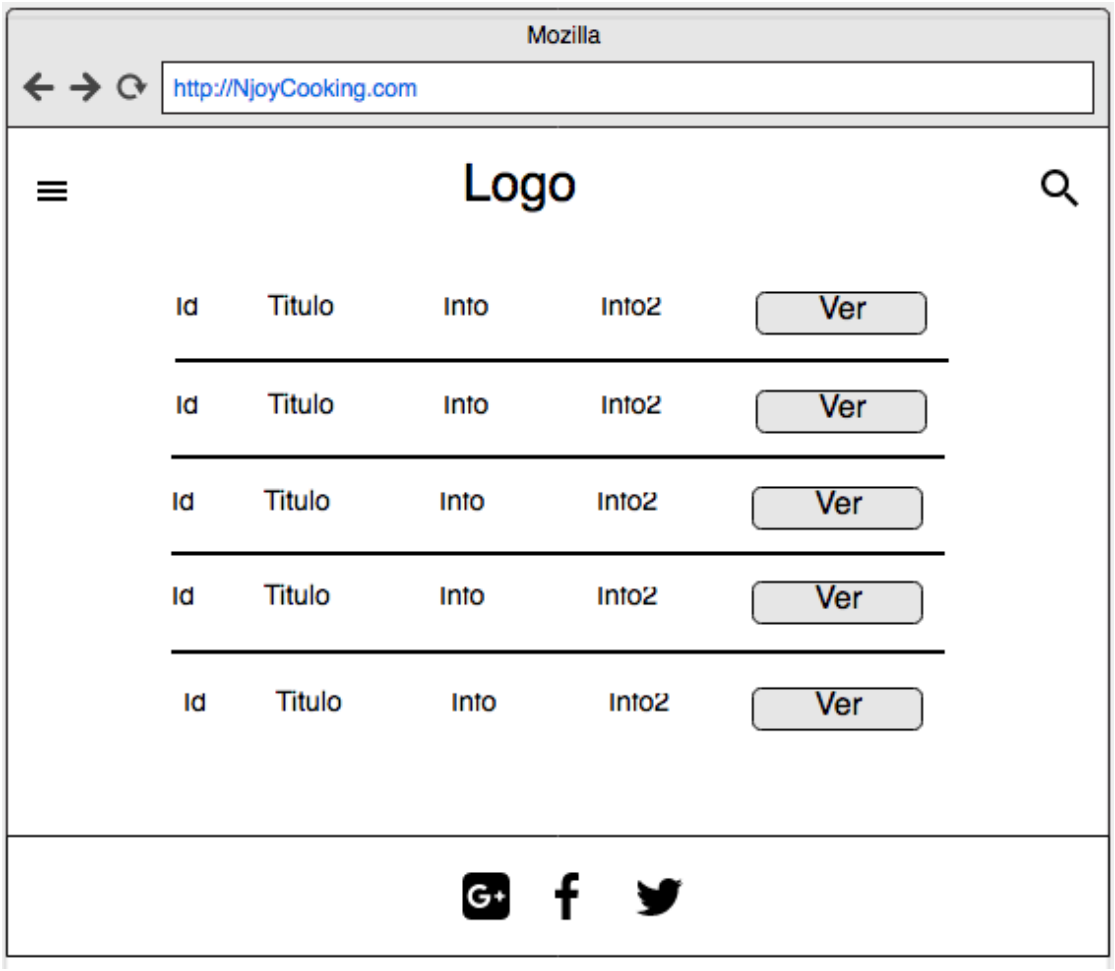

Figura 5.5: Listado de elementos

5.3.1. Diseño de la interfaz

Inicialmente se llevan a cabo unos bocetos de la web mediante mockups para saber como se va a organizar el contenido de la aplicación.

Cualquier empresa que se encuentre en el mundo de las aplicaciones web sabe que una landing, es la mejor forma de promocionar el producto o marca que se quiere dara conocer. Por eso a la página inicial donde el usuario accederá será una landing donde se mostrará información acerca de los servicios que ofrece la web.

Como se observa en el boceto(figura 5.1) la landing tiene un diseño sencillo y usable. En esta página se mostrarán una series de secciones informativas acerca de los contenidos disponibles de la web y otras informaciones relevantes. Las secciones irán acompañadas de una imagen de fondo, un texto informativo y un botón que dirige a la página correspondiente.

La cabecera de la landing, que será común a todas las páginas de la web, tendrá un diseño responsive para todas las resoluciones. El menú será desplegable de forma que al pulsar sobre el icono de la hamburguesa se verán todas las secciones por las que se podrá navegar por la web. Este menú mobile se ha aplicado para la resolución de escritorio ya que favorecía a conseguir una mayor limpieza de la interfaz,no empeoraba su usabilidad y le daba un aspecto más minimalista. Además de este menú en la cabecera encontraremos el logo de la web y un buscador global de la web. Esta cabecerá siempre estará fija para favorecer la usabilidad al usuario.

Por último, el footer o pie de página, otro elemento común a todas las páginas. En esta sección se mostrarán las distintas redes sociales de la empresa, los textos legales(copyright,privacidad) y el logo de la web.

La siguiente página, es la del mapa (figura 5.2). En esta página se mostraban un mapa con los marcadores de las recetas geoposiconadas. El diseño de esta página es muy sencillo ya que además de los elementos comunes de cabecera y pie, en el contenido se muestra un elemento con el mapa que va a mostrar la información.

La próxima sección, es la sección del blog que se compone de dos páginas: el listado de noticias y el detalle de noticia. En el listado de noticias(figura 5.3), el primer elemento que se mostrará será un encabezado con una foto y descripción de la sección del blog. A continuación se mostrarán un listado con las diferentes noticias del blog y en cada una se mostrará un foto, un título y un resumen. Para esta sección la disposición de los elementos de la notícia en movil cambiará colocando los elementos en una sola columna en vez de dos como se muestran en tablet y pc.

En el detalle de la receta(figura 5.4) se muestra toda la información asociada a esa noticia. El primer elemento que aparece es el encabezado de la noticia donde se muestra su título y una foto de fondo. A continuación se muestra en dos columnas la información de la receta, una columna con un primer bloque de información y la segunda con la imagen principal de la noticia. Después se muestran los demás bloques de información en una sola columna y por último se muestran otra fotos asociadas a la noticia. Al final de la noticia se muestra un formulario para escribir comentarios asociados a la noticia y un listado con los comentarios que tiene la noticia. La estructura del contenido cambia para la resolución móvil mostrando todo el contenido en una columna.

Si el usuario que accede a la página esta registrado, aparecerán nuevas secciones disponibles. Estas secciones serán vistas como listados de elementos(figura 5.5), se usaran para mostrar cualquier tipo de datos que visualice el usuario registrado. La información se listará en una disposición de tabla,

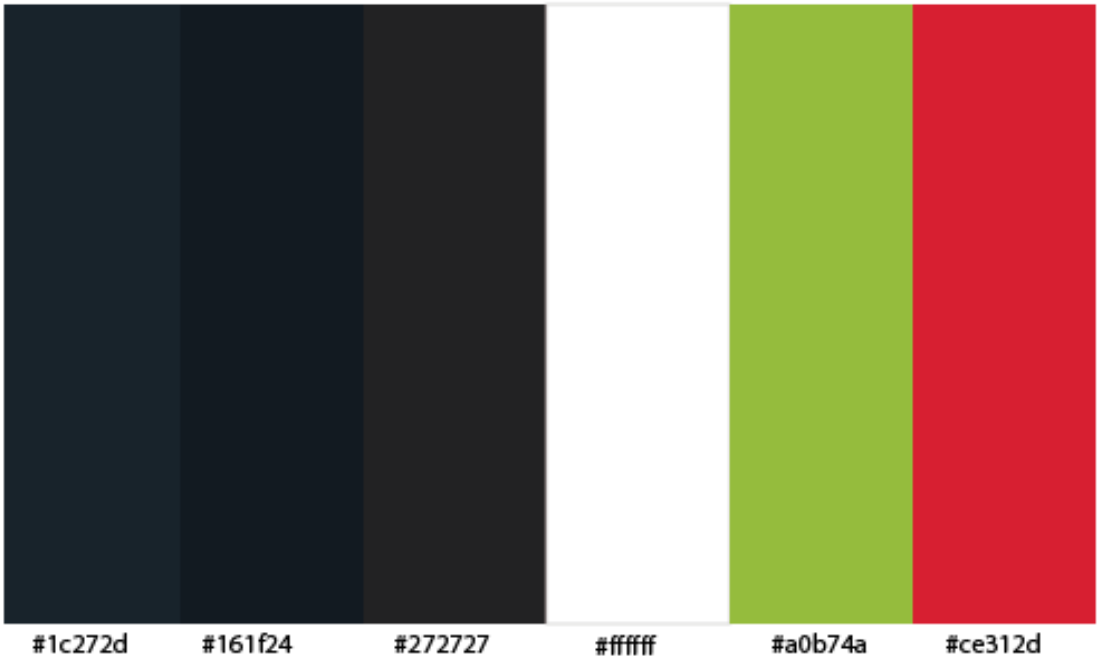

Figura 5.6: Paleta de Colores

con el contenido del elemento(id,titulo,etc) y un botón para ver el detalle. La presentación de este contenido es mucho más simple, que el listado para los usuarios no registrados, ya que esta sección tiene que ser mucho más pragmática y menos visual.

Paleta de Colores

Para la aplicación se ha seleccionado una paleta de colores(5.6) con el programa spectrum. Se ha elegido una gama de azules y grises oscuros que dan un aspecto de seriedad y confianza. También se utilizan una series de colores mas vivos como el verde y el rojo que se utilizan para los botones, pequeños detalles y resaltar los textos para dar una aspecto más vivo.

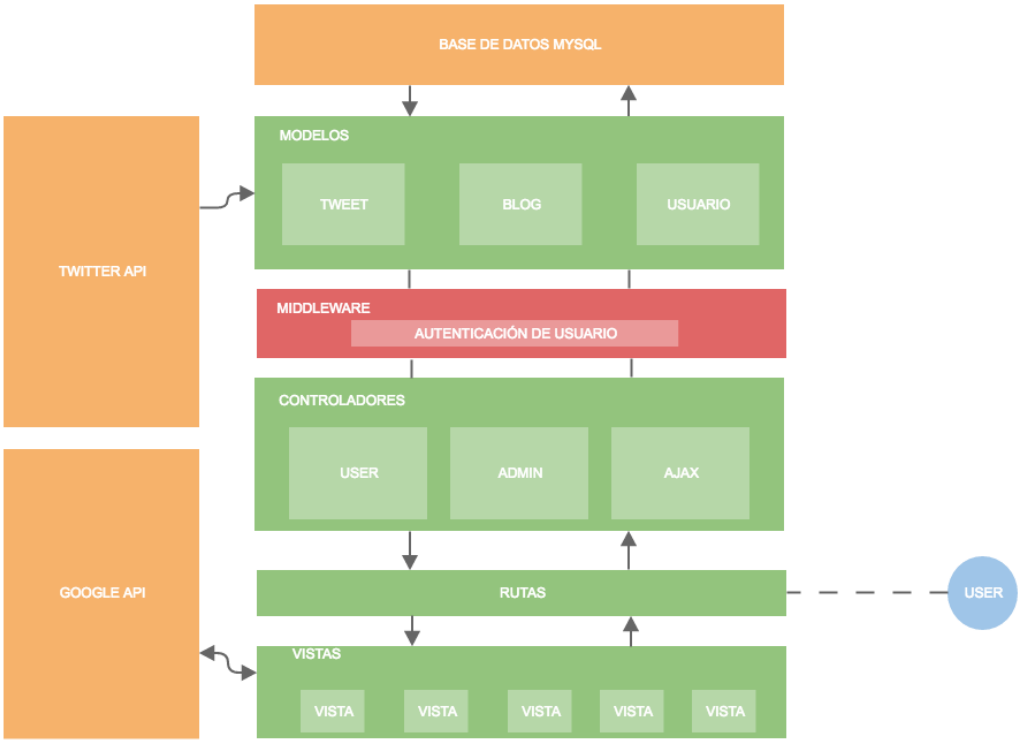

Figura 5.7: Arquitectura específica

5.3.2. Implementación de la arquitectura

En el segundo sprint del proyecto, se lleva acabo un esquema específico de la arquitectura usada para la aplicación. En este esquema de arquitectura ya no es tan general, se especifican detalladamente cada una de las capas de la arquitectura,se enumeran cada uno de los elementos que se va a componer y se detallan cada una de las APIs externas empleadas.

En la figura 5.7 se observa un esquema detallado de como se ha implementado la arquitectura de la aplicación. A continuación se procede a profundizar en cada una de las capas de la arquitectura.

```
<!DOCTYPE html>
<!--[if gte IE 6]>
<html class="ie" lang="es-ES">
<![endif]-->
<![if !IE]>
<html lang="es">
<![endif]>
<head>
    @include('includes.head')
</head>
<body>
@include('includes.header')


    @yield('content')


</body>


@include('includes.footer')

<!--[if lt IE 9]>
<script type='text/javascript' src="js/lib/make_ie8_work.js"></script>
<script type='text/javascript' src="js/lib/modernizr.custom.84408.js"></script>
<script type='text/javascript' src="js/lib/selectivizr-min.js"></script>
<![endif]-->
<script type='text/javascript' src="js/lib/device.min.js"></script>


</html>
```

Figura 5.8: Layout principal de la estructura HTML de la aplicación

El usuario que navegue por la aplicación, accederá a las diferentes páginas de la web por medio de una ruta que introduzca en el navegador. Una vez introducida la ruta comienza el flujo de información de la aplicación.

Con la ruta introducida, la aplicación nos manda al controlador correspondiente para obtener la información. Para organizar mejor la información, se estructuran los siguientes controladores:

- **Controller**. Controlador principal de la aplicación, en esta clase se manejan todas las rutas de la aplicación. Este controlador comprueba que la ruta introducida sea correcta, dirigiendo al controlador de cada apartado de la página o por el contrario mostrando la página de error si la ruta es incorrecta.En el constructor de este controlador se crean instancias del resto de controladores en el constructor que luego se llaman para renderizar las distintas vistas de la web.
- **AjaxController**. En este controlador se manejarán todos los datos realizados con peticiones ajax. Apartados como los comentarios, utilizarán peticiones ajax para guardar la información en la base de datos y no tener que recargar la página.
- **SiteController**. Contiene los métodos que manejan la información correspondiente a la parte pública de la web. El mapa, el blog y la landing se gestionan mediante este controlador.
- **AdminController**. La parte privada, es decir aquellas vistas que requieran de autenticación. La gestión de las noticias del blog, los tweets del mapa, se gestionarán con este controlador.

En los controladores se hace uso de los diferentes modelos de datos de la aplicación. Los modelos generalmente se corresponden con una tabla de la base de datos. Para cada objeto de datos se crea la clase php(el Modelo) donde se encuentran los métodos que insertan, borran o actualizan la información

de su tabla correspondiente.

Para obtener los datos de la API twitter, se utiliza la librería TwitterAPIExchange [14], facilitando la obtención de tweets y disponiendo al desarrollador de diversos métodos de búsqueda de resultados. Los datos obtenidos mediante la librería se parsean y se guardan en la base de datos de la aplicación mediante su modelo de datos(Tweet).

Una vez obtenida la información en los controladores, se le pasan los datos a la vista que corresponde para visualizar el contenido. Las vistas estan envueltas en un layout, que contiene la estructura de la aplicación y los trozos de la web que son comunes para toda la web, como la cabecera y el pie de página. En el layout de la aplicación (figura 5.8) se observa su estructura. Mediante la sentencia include incrustamos los elementos comunes de la aplicación en el layout(footer y cabecera) y finalmente mediante yield se vuelca el contenido de la vista. Existen varias vistas en la aplicación y cada una tendrá sus estructura HTML y sus características propias.

En la parte de cliente, una vez renderizada la vista se hace uso de la API de Google Maps mediante Javascript para generar el mapa de la aplicación y todos sus elementos(marcadores,ventanas de información).

Para comprobar la autenticación del cliente, y que cualquier usuario no pueda acceder a partes de la web que solo están disponibles para el administrador, se implementa una capa intermedia entre los controladores y los modelos llamada middleware. Esta capa controla que, el usuario que intenta acceder a las páginas disponibles solo para el administrador, esté registrado como usuario en la base de datos y tenga una sesión iniciada como usuario. Por el contrario si no está registrado, el usuario no podrá acceder y será redirigido a la página principal.

5.3.3. Maquetación

Una vez implementada la arquitectura y definidas las vistas de la aplicación, el siguiente sprint consta de la maquetación de la web.

Previamente se estructuran todos los ficheros PHP con contenido HTML en directorios para organizar mejor la aplicación y que las vistas sean más reutilizables:

- **layouts**: aquí se incluye la plantilla main con la estructura de la aplicación.
- **includes**: en este directorio se incluiran los ficheros que contiene fragmentos parciales de HTML. Estos fragmentos pueden ser tanto elementos fijos de la web como la cabecera o el pie que se incrustan en la estructura principal o render parciales de una vista para estructurar mejor el contenido.
- **site**: aquí se incluyen todas las vistas de la aplicación que tienen asociadas un controlador a ellas, como puede ser la página del blog y del mapa.
- **error**: en este directorio se incluyen todas las plantillas para mostrar los errores de la web. El error más común es el 404 de la página no encontrada.
- **auth**: en la carpeta auth se incluyen los ficheros de autenticación del usuario, como el login del usuario.

La estructura HTML de las vistas sigue una jerarquía de las etiquetas que se aplica para generar un contenido más limpio de las vistas y pasar con éxito el validador del W3C garantizando una correcta escritura de HTML.

Uso de la etiqueta ¡section¿para dividir las partes de la vista. Cada vista se puede componer de uno o más sections. Por ejemplo para la página de

```
<section class="background blogs-intro">
    <div class="blogs-intro-wrapper">
        <h1 class="blogs-intro-title">¡Bienvenidos al blog de NjoyCooking!</h1>

        <p class="blogs-intro-text">Un espacio donde disfrutarás de las últimas técnicas
        de cocina y consejos prácticos.
            Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ad aliquid amet
            asperiores aspernatur at cumque facere fugit</p>
    </div>
</section>
```

Figura 5.9: Ejemplo de jerarquía de clases

blog se divide con dos sections, una para el bloque de introducción del blog y otra para el listado de noticias. Dentro de cada ¡section¿van las etiquetas que se usarán para posicionar el elemento y darle estilos como son las ¡div¿,¡ul¿,¡span¿y ¡p¿.

Cada etiqueta HTML tiene una clase asociada. Las clases son muy importantes ya que pemiten clasificar los elementos para luego poder aplicarle unas reglas CSS comunes. La nomenclatura que se usa para las clases, es una anidación de palabras clave, seguidas de un guión(-). Por ejemplo para el bloque de la intro del blog(5.9) esta el ¡section¿con la clase que contiene la anidación de palabras blogs-intro. Con esta nomenclatura se hace referencia a que es este elemento pertenece a la vista del blog(blog) y la sección de introducón(intro). Dentro de este elemento, se encuentra una etiqueta ¡div¿que contien los elementos y por eso la nomenclatura de la clase es blogs-intro-wrapper(wrapper es una palabra inglesa que significa envoltura). Por último los dos elementos de la sección son los textos, el blogs-intro-text y blogs-intro-title para estas dos clases se le anida como última palabra text o title para aplicar las reglas a cada tipo de texto dentro del bloque de introducción.

Aplicando esta nomenclatura de anidación de palabras clave y gracias a la posibilidades que da Sass, se crea una jerarquía de clases en Sass que permite

```
.blogs-intro{

  //height:100%;
  color: $blanco;

  &-title{

    text-transform: uppercase;
    font-size: 30px;
    font-weight: 700;
    padding:20px 0;
    text-align: center;

  }

  &-text{

    font-size:18px;
  }

}
```

Figura 5.10: Ejemplo de jerarquía Sass

optimizar al máximo las reglas de estilos de cada elemento evitando tener que duplicar reglas, obteniendo una visión más clara de como está organizado y facilitar modificaciones posteriores por parte de cualquier desarrolador. Un ejemplo de la jerarquía Sass sería la figura 5.10.

Para los elementos que son manipulados con jQuery, se les añade un clase extra con el prefijo js-nombre. De forma que cualquier elemento que tenga un evento de jQuery asociado, deberá ser seleccionado mediante la clase con el prefijo js. Estas clases no tendrán estilos asociados, de forma que las clases se dividirán en dos tipos: los selectores de elementos de jQuery y las clases con reglas de estilos asociados. De esta forma estará mejor clasificada la maquetación de los elementos y permitirá que las futuras restructuraciones de la web no supongan tanto trabajo, ya que si únicamente se quiere modificar la apariencia del sitio o optimizar el rendimiento de la web cambiando el código javascript, solo se tendrá que modificar una de los dos tipos de clases

```
Route::get('/',[
    'uses' => 'Controller@init',
    'as' => 'home'
]);

Route::get('/admin/dashboard','Controller@initDashboard');
Route::get('/admin/tweets','Controller@initDashboard');
Route::get('/admin/tweets{id}','Controller@initDashboard');
Route::get('/admin/posts','Controller@initDashboard');
Route::get('/admin/posts/{id}','Controller@initDashboard');

Route::get('/logout', 'Controller@getLogout');

Route::get('/{slug}', array('as' => 'web', 'uses' => 'Controller@init'));
Route::get('/{slug}/{detail}', array('as' => 'web', 'uses' => 'Controller@init'));


Route::controllers([
    'auth'=>'Auth\AuthController',
    'password'=>'Auth\PasswordController',
]);
```

Figura 5.11: Rutas de la aplicación

sin alterar la otra.

5.3.4. Programación

Una vez terminada la maquetación de las páginas, se procede a la programación del back-end de la aplicación. Este apartado es el más extenso de desarrollar por lo que se divide en tres sprints: rutas de la aplicación, lógica de negocio y servicios.

Rutas de la aplicación

Para tener una web con un sistema de rutas amigables que faciliten la navegación del usuario y el posicionamiento del sitio, la mejor opción es generar rutas dinámicas. Mediante las rutas dinámicas el administrador del sitio podrá modificar los valores de las rutas mediante el panel de administración, modificando las urls de cada página cuando sea necesario sin la necesidad de recurrir al programador. Otros campos que el administrador también podrá

modificar son los metas de cada página(MetaTitle,MetaKeywords,etc), que son otro factor importante para el posicionamiento SEO.

Para definir las rutas, Laravel proporciona el fichero routes.php dentro del directorio app que es donde se definen todas las rutas de la aplicación. Dentro de este fichero se definen las rutas especificando el tipo de petición Http(get,post,etc)el nombre de la ruta y el controlador que usa, también podemos definir un alias que identifica la ruta aunque este campo es opcional. Como se observa en la figura 5.11 se definen 4 rutas:

- **Ruta Principal**: la ruta para la pagina principal donde se mostrará la landing se define mediante el caracter /". a esta página se accederá cuando el usuario acceda al dominio de la aplicación.
- **Rutas de Menu**: estas rutas son las páginas de la aplicación que pertenecen al modelo de datos Menu que se corresponde con el menú de navegación de la web. Se generan dinamicamente comprobando que la ruta obtenida se corresponde con la ruta definida en el modelo. Para que la ruta sea dinámica en Laravel se tiene que definir mediante llaves y dentro un nombre de variable.
- **Rutas de detalle**: las rutas de detalle son aquellas que tienen tienen como trozo de url padre una ruta de Menu. No todas las rutas de Menu tienen que tener una ruta de detalle. Esta ruta se usa en la aplicación para definir las notícias del blog de la aplicación y asi poder concatenar a la ruta del blog el nombre de la notícia.
- **Rutas de usuario registrado**: estas rutas son para la vistas de la web de la parte de administrador. También son dinámicas pero para diferenciarlas de las rutas visibles para todos los usuarios se iniciarán con la ruta /admin.

Todas estas rutas se manejan en el controlador principal de la aplicación(Controller).

```php
public function getCoordinates($location){


    $geocode = "http://maps.google.com/maps/api/geocode/json?address=".$location."&sensor=false";

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $geocode);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    $geoloc = curl_exec($ch);
    $geoloc = json_decode($geoloc);

    $random = $this->getRandDecimal();
    $coordinates["lat"] = $geoloc->results[0]->geometry->location->lat + $random;
    $coordinates["long"] = $geoloc->results[0]->geometry->location->lng + $random;

    return $coordinates;
}
```

Figura 5.12: Función: recuperación de datos de geolocalización de una coiudad

Se comprueba que las rutas introducidas sean las correctas y en el caso de no serlo se redirige a la página de error 404 de la web. Para renderizar la vista correspondiente, se instancias de servicios dentro del Controller principal. Estos servicios contienen las acciones que renderizan la vista asociada para cada url y un constructor inicial que guarda los metadatos para cada vista.

Modelos

En esta capa se accede a los datos para ser modificados mediante los modelos datos y utilizando una herramienta de Laravel como ORM, facilita la conexión con la base de datos. Estos métodos de ORM sustituyen a sentencias SQL. Por ejemplo: la función get() de una clase que extiende de la clase Model de Laravel obtendría todos los registros de la tabla que se corresponde con ese modelo. En SQL el equivalente sería hacer un SELECT * FROM Tabla.

En los modelos además de conectarse con la base de datos, también hay métodos que conectan con API externas para recoger datos. Es el caso del método getCoordinates() (figura 5.12) donde se llama a la API de Google Maps enviando un string con la localización de una ciudad y obteniendo sus

```php
public function getTweets($query){

    $tweets = Twitter::getSearch(
        array('q' => $query,
            'count' => 15,
            'lang'=>'es',
            'geocode'=> '39.8952506,-3.46863775058,500km',
            'format' => 'json')
    );

    $this->parseTweets($tweets);
}
```

Figura 5.13: Función que recibe los datos de Twitter

datos de geolocalización.

Controladores

En esta capa se encuentra la mayor parte de la lógica de la aplicación.Como se comentó previamente, existen varios controladores según para que apartados de la web. Pero todas las rutas pasan por el controlador principal que las maneja y controla que existan para mostrar la vista correspondiente.

Como se observa en 5.15, al inicializar el controlador, se generan dos instancias de los otros dos controladores que se encargan de manejar los datos y renderizar las vistas para la parte pública y la parte de administrador.

Existen dos métodos dentro de la clase que controlan las rutas, init() para la parte pública e iniDashboard() para la parte que requiere loguearse.

En el caso de la parte pública, en 5.16 se observa el método init(). Incialmente se le pasan los campos $slug y $detail que son los trozos de la url. Se comprueba que el $slug se encuentre en la tabla de menú de la base de datos y se empieza a buscar la url para mostrar la vista correspondiente. Por

```php
public function parseTweets($tweets){

    $decoded_tweets = json_decode($tweets,TRUE);

    //dd($decoded_tweets);

    foreach($decoded_tweets["statuses"] as $key=>$tweet ){

        $t = $tweet["text"];
        $entities = $tweet["entities"];
        $text = $this->getTextTweet($t);
        $URL  = $this->getUrl($entities);
        //dd($tweet["user"]["screen_name"]);
        $user = '@'.$tweet["user"]["screen_name"];
        $image = $this->getImage($entities);
        $location = $this->getLocation($tweet["user"]);
```

Figura 5.14: Función para parsear los Tweets

```php
public function __construct(Route $route, SiteController $sitecontroller, AdminController $admicontroller){


    $this->siteController = $sitecontroller;
    $this->admiController = $admicontroller;

}
```

Figura 5.15: Constructor del controlador principal

```php
public function init($slug = "home",$detail = "empty"){

    $menuActual = Menu::where("Slug",$slug)->first();

    if($detail == "empty"){
        if($menuActual){

            return $this->siteController->$slug();

        }else{

            if($slug == "home"){

                return $this->siteController->index();

            }else{

                if($slug == "login"){

                    return $this->siteController->login();
                }else{

                    return $this->siteController->error();
                }
            }
        }
```

Figura 5.16: Método de init()

```php
$rules = array(
    'email'    => 'required|email',
    'password' => 'required|alphaNum|min:3'
);

$validator = Validator::make(Input::all(), $rules);

if ($validator->fails()) {

    $errors = new MessageBag(['password' => ['Debes rellenar todos los campos correctamente']]);
    return Redirect::to('login')
        ->withErrors($errors)
        ->withInput(Input::except('password'));
} else {
    $userdata = array(
        'email'     => Input::get('email'),
        'password'  => Input::get('password')
    );

    if (Auth::attempt($userdata)) {

        return redirect()->intended('admin/dashboard');

    } else {

        $errors = new MessageBag(['password' => ['Email o contraseña incorrecta']]);

        return Redirect::back()->withErrors($errors);

    }

}
```

Figura 5.17: Método de login()

```php
$routeSlug = $route->getParameter("slug");
$menuActual = Menu::where("Slug",$routeSlug)->first();

    $menuitems = new Menu;
    view()->share('menuitems', $menuitems->getMenus());

    if($route->uri() != "{slug}/{detail}"){

        if($menuActual != null){

        view()->share('menuactual', $menuActual);
        view()->share('metaTitle', $menuActual->MetaTitle);
        }else{

            view()->share('menuactual', $menuActual);
            view()->share('metaTitle', "Login - NjoyCooking");
        }

    }else{

        if($route->uri() == "/"){

            $menuActual = Menu::where("idMenu",1)->first();
            view()->share('menuactual', $menuActual);
            view()->share('metaTitle', $menuActual->MetaTitle);
        }

        if($route->uri() == "{slug}/{detail}"){

            $slugBlog = $route->getParameter("detail");

            if($slugBlog != null){
                $blogActual = Blog::where("Url",$slugBlog)->first();

                if($blogActual != null){
                    view()->share('menuactual', $blogActual);
                    view()->share('metaTitle', $blogActual->MetaTitle);
                }else{

                    view()->share('menuactual', $blogActual);
                    view()->share('metaTitle', 'Entra como administrador');

                }
```

Figura 5.18: Constructor del siteController

```
public function showblog(Blog $blog){


    $comentarios = new Comentario;
    $comentarios = $comentarios->getComentarios($blog->idBlog);
    return view('site.blog-detalle',compact('blog','comentarios'));
}
```

Figura 5.19: Método shoBlog del siteController

defecto si a los trozos de url que se pasan estan vacíos se establecen valores predeterminados. En el caso del $slug cuando esta vacío es que es la url de inicio y por tanto se debe imprimir la página incial de la web. Para el caso $detail si la url esta vacía se establce a empty, de esta forma sabemos que no se ha pasado ninguna url de detalle y por tanto se renderizan las páginas que no tienen detalle, como por ejemplo la página del mapa. Si la url contiene un detalle entonces pertenece a cada noticia del blog

El login de la aplicación se gestiona también mediante el controlador principal. En el método doLoging() se realiza la identificación del usuario en la aplicación. Como se observa en 5.17 inicialmente se validan los datos que el usuario ha introducido. En el caso de que la validación falle, devuelve al usuario a la página de login con los errores pertinentes. Si las credenciales del usuario son correctas se le redirige a la ruta de inicio del administrador.

Las instancias de la clase siteController y el adminController que se usan en el controlador principal, llaman a las diferentes funciones que gestionan la lógica para cada vista correspondiente con la parte pública y privada respectivamente. Antes de llamar a cualquier función se inicializa la clase mediante el constructor. Un ejemplo del constructor sería 5.18 donde se observa el código al inicalizar el siteController. En el constructor obtenemos el valor de la

url en la que estamos y se obtienen los metas de la página para introducirlos en el HTML. También se guardan los datos de la página.

Con la clase inicializada, se pueden llamar a las distintas funciones disponibles. Un ejemplo podría ser el método showBlog() del siteController, que muestra el detalle de cada noticia del blog. Como se observa en la figura 5.19 se pasa por parámetro una variable blog que contiene el blog al que hemos accedido. Se recogen los comentarios relacionados a la noticia llamando al modelo correspondiente de los comentarios. Y por último se devuelve la vista correspondiente con la información asociada, en este caso los datos del blog y los comentarios.

El caso más complejo de lógica es el del mapa de geoposicionamiento de recetas. Para obtener la información de Twitter se utiliza la una API rest para Laravel que proporciona varias funciones de búsqueda de información. Una vez integrada la API con el proyecto toda la información se procesa en el modelo Tweet. El método básico de este modelo es getTweets($query)(figura 5.13) ya que en este método es donde se hace uso de la API de Twitter para obtener los resultados de la red social. Para realizar la búsquedase utiliza el método getSearch() de la API a la que se le pasa un array con los parámetros de búsqueda:

- **q**: la consulta o palabra clave sobre el que la api va a buscar los tweets. Es parámetro es dinámico ya que se pasa por parámetro en la función para que el administrador pueda modificar la palabra clave desde el dashboard.
- **count**: número de resultados que devolverá la consulta.
- **lang**: idioma en el que están los tweets.
- **geocode**: geocodificador que se le pasa por parámetro la latitud, longitud y un radio que representan el área sobre el que se van a buscar los

tweets. El area utilizado es el de la península ibérica.

- **format**: el formato en el que devuelve los resultados. En este caso json.

Al resolver la petición, devuelve los tweets con toda la información asociada. Como no se van a utilizar todos los datos obtenidos esta información se procesa con el método parseTweets()(figura 5.14). En este método se recorren uno a uno cada tweet y se obtienen los datos relevantes para insertar en la base de datos utilizando otros métodos de la clase.

6 Conclusiones

6.1. Resultados

Durante todo el periodo de desarrollo del proyecto se ha conseguido implementar la arquitectura definida en el capítulo de Desarrollo. Se ha conseguido cumplir el objetivo de diseñar e implementar una arquitectura robusta y que ha conseguido gestionar la información proveninente de la API de Twitter y guardarla en la base de datos de forma satisfactoria.

Otro de los objetivos conseguidos, ha sido conocer el uso de las APIs de Google Maps y Twitter a fondo para luego poder aplicarlas correctamente en la aplicación.

Respecto a los requisitos definidos, se han podido implementar la mayoría de los requisitos exceptuando **FS-U-01**,**FB-A-03**. Para el caso de la valoracion de las entradas del blog(**FS-U-01**) se ha dejado para el siguiente sprint debido a ser una funcionalidad no básica de la aplicación. Para el requisito **FB-A-03** de mostrar los comentarios asociados a la entrada del blog se ha tenido que posponer para un siguiente sprint, debido a los requisitos **FB-U-04** y **FB-A-05** correspondientes al mapa de Tweets y su parte de administración han supuesto realizar mas iteraciones de las previstas. Para el caso **FB-A-03** de gestionar el contenido de la entrada, se ha podido completar pero con la existencia de un fallo a la hora de subir imágenes.

Con la parte de maquetación finalizada, se lleva acabo un testeo y prueba de la aplicación en los distintos navegadores web. Se realizan testeos con los

siguientes navegadores:

- Google Chrome, Mozilla Firefox para los sistemas operativos de Android,IOS,Windows y Mac OS.
- Safari para IOS y Mac OS.
- Internet Explorer 11 para Windows.

El testeo en estos navegadores ha sido satisfactorio y responde adecuadamente a los estilos aplicados y cambios de resolución aplicados mediante CSS.

En definitiva, se podría decir que se ha conseguido el objetivo principal de desarrollar un proyecto que funciona como herramienta orientada a marketing y como un portal de promoción de la gastronomía. Esta versión de la aplicación proporcionará un producto mínimo viable, pero se lanzará como beta, ya que todavía quedan bastantes funcionalidades que mejorar e implementar.

6.2. Problemas Encontrados

El principal problema, a la hora de desarrollar el proyecto ha sido el uso de datos de geolocalización en la aplicación. Debido a temas de privacidad por parte de los usuarios, un porcentaje superior al 90 % no tenía activada la geolocalización en su aplicación de Twitter. De modo que para representar la información en el mapa de la aplicación, se obtenía la ciudad del usuario al que pertenecía el Tweet.

De esta forma los tweets que eran de la misma ciudad se solapaban en un mismo punto (las coordenadas centrales de la ciudad). De modo que para que se visualizarán los datos en el mapa sin solapamiento se decidió por generar dos números aleatorios, uno para la latitud y otro para la longitud, que son los que delimitarán el marcador en el mapa. Para que estuvieran

dentro del rango de los límites de la ciudad se llamaba a la ruta `https://api.twitter.com/1.1/geo/search.json?query` de la API de twitter con el nombre de la ciudad y obteniendo un objeto JSON con las coordenadas que definen el área de esta.

Este método consiguió solucionar el solapamiento de los marcadores en el mapa, para que se vieran todos en el mapa. Pero los datos no se podrían considerar 100 % fiables, ya que no son del todo reales debido a dos razones:

Localización según usuario y no tweet: Los datos de localización que se muestran en el mapa son los de la ciudad del usuario y no los del tweet. Por tanto los tweets de un mismo usuario siempre aparecerán sobre el mismo área de su ciudad independientemente del punto geográfico en el que se encuentre. Por tanto para este caso no sería del todo real la representación.

Punto exacto del tweet : En el caso de que el usuario se encuentre en su ciudad de origen, se representarán los tweets mediante los números aleatorios de latitud y longitud para evitar solapamiento entre tweet de la misma ciudad. Con este método se soluciona la visualización de los tweets, pero no se obtiene la precisión cómo en el caso de la geolocalización. Esta problemática es menos relevante que la primera, pero también supone una pérdida de fiabilidad en la aplicación.

6.3. Mejoras y ampliaciones

La principal mejora a implementar se encuentra en el rediseño de la arquitectura de la aplicación. Para conseguir una web con más dinamismo e interacción con el usuario se utilizará el framework de javascript Angularjs. Además este framework, permite generar un patrón de MVC que permite programar el front-end de la aplicación más modular y tener el código mejor estructurado.

En cuanto a la parte de back-end. Lo primero será terminar los requisitos no finalizados definidos anteriormente y solucionar bugs.

Una ampliación interesante a tener en cuenta podría ser la integración de otras APIs de redes sociales en el proyecto. Ya que puede que en otras redes sociales los usuarios utilicen más la opción de localización y por tanto solucionar el problema surgido con Twitter.

Se deberá refactorizar el código de forma que sea escalable para una implementación real, donde la cantidad de información y el número usuarios es bastante superior al prototipo funcional.

También se tendrán que utilizar herramientas de Big Data, ya que actualmente los datos se gestionan desde un cliente de bases de datos [4]. Existen muchas herramientas de Big Data, y están especializadas en un área, se necesitarán algunas tales como:

- **Hadoop**, una herramienta de almacenado. Un framework open-source que permite trabajar con miles de petabytes de datos.
- **Oracle data mining**,herramienta de mineria de datos muy potente. Permite hacer predicciones acerca de los datos y crear modelos para descubrir el comportamiento del cliente.
- **Chartio**, herramienta de visualización. Proporciona herramientas para crear potentes gráficas y paneles de visualización de datos en unos pocos clicks.

6.4. Modelo de negocio

Despues de desarrollar un producto mínimo viable de la aplicación de Njoycooking, se plantea la siguiente cuestión: ¿Es posible generar un modelo de negocio para la aplicación?

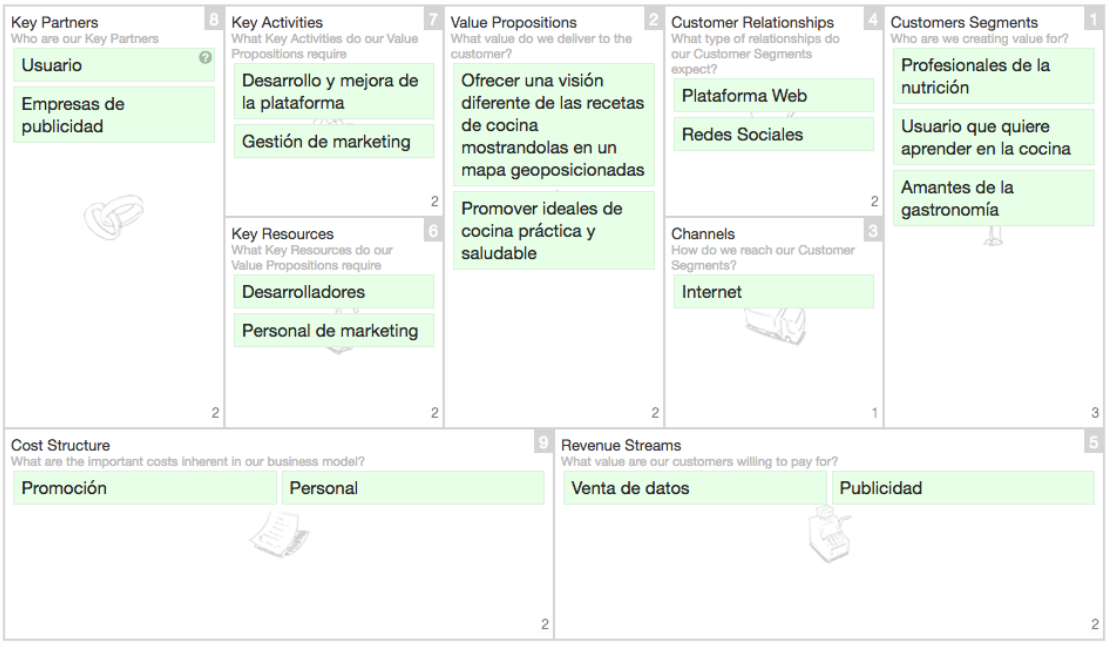

Figura 6.1: Business Canvas de la aplicación

Mediante un business model canvas(figura 6.1) se describe de una forma simple en que consiste la idea de negocio.

Al generar un modelo de negocio, el primer concepto que se tiene que valorar es: ¿Qué propuesta de valor ofrece la aplicación? es decir que elemento diferenciador puede ofrecer la aplicación frente otros existentes en el mercado. NjoyCooking es una aplicación gastronómica pero además de ofrecer un blog de cocina proporciona un visión diferente de los recetas de cocina, mostrando lo que cocina la gente según su zona geográfica mostrando las recetas geoposicionadas en un mapa y en tiempo real. Otra propuesta, es la de promover los hábitos de cocina práctica y saludable.

Otro concepto que se tiene que valorar es como relacionarse con el cliente. El principal punto de contacto con los clientes será la aplicación web, donde los

usuarios podrán interactuar y conocer la propuesta de valor que ofrece. Otra forma de contactar será mediante las redes sociales, ya que son un medio de difusión muy importante para a dar a conocer la aplicación.

¿De dónde se obtendrán los ingresos? Las principales fuentes de ingresos serán dos:

Publicidad: se proporcionará a las empresas la posibilidad de anunciarse en el sitio web mediante banners.

Venta de datos : los datos recogidos en la aplicación se ofrecerán a cambio de un pago económico a las empresas interesadas. Estos datos pueden servir a las empresas, que pueden realizar un estudio de mercado con ellos en función de las gustos y preferencias que tienen los usuarios de NjoyCooking.

¿Cuáles son las fuentes clave? ¿Es decir que personal se necesita para llevar a cabo la propuesta de valor? Para cumplir la propuesta de valor se necesitará la contratación de programadores de software para desarrollar y mejorar la aplicación y especilistas en marketing para dar a conocer la aplicación y ofrecer un mejor uso y experiencia de ella.

El usuario activo de la aplicación será el principal partner de la aplicación, ya que podrá promover la plataforma a sus conocidos y así abarcar un mayor número de usuarios. Las empresas de marketing y publicidad también se tienen en cuenta ya que pueden ayudar a llevar a cabo la propuesta de valor patrocinando el producto.

Por último se debe tener en cuenta: ¿A qué público va enfocada la propuesta de valor? NjoyCooking tiene como objetivo un público concreto como son los amantes de la gastronomía y cocina en general. Aunque el objetivo es

específico, engloba un abánico de usuarios amplio donde existe una variedad por edad,sexo,profesión o aptitudes culinarias.

Bibliografía

[1] Documentación de analytics. https://goo.gl/dJBWi2.

[2] Documentación de laravel. https://goo.gl/0ihWxe.

[3] Documentación de sass. https://goo.gl/wS8AkC.

[4] Herramientas de gestión de big data. https://goo.gl/FzrFwV.

[5] Scrum, métodología ágil. https://goo.gl/ixuest.

[6] Técnicas de posicionamiento sep. https://goo.gl/KHTUA7, 2013.

[7] Marketing en redes sociales. https://goo.gl/w3Y1b2, 2015.

[8] Evolución del marketing digital. https://goo.gl/8pZ049, 2016.

[9] Targeting de cada red social. https://goo.gl/QklCgk, 2016.

[10] Miguel Angel Alvarez. Arquitectura mvc. https://goo.gl/LFw5Ep.

[11] Google. Api de google maps docs. https://goo.gl/G39aMk.

[12] JQuery. Documentación de jquery. https://goo.gl/RFkHX5.

[13] Laravel. Eloquent orm. https://goo.gl/E0aanb.

[14] Thujohn. Framework de la api twitter para laravel. https://goo.gl/srIzPB.

[15] Twitter. Documentación de la api detwitter. https://goo.gl/SxuXms.