Capítulo 1

Introducción

1.1. Objetivos

El objetivo de este trabajo de fin de grado (en adelante, TFG) es el desarrollo de una aplicación web para la gestión del escandallo, es decir, controlar el precio de producción de una receta. Para ello usaremos Cloud Servers, concretamente Google Cloud Platform, sin embargo, por algunas de las restricciones que mostraba, las cuales se describirán más adelante, se reorientó el TFG hacia otras plataformas como servicio (SaaS). Para su desarrollo usé la metodología de software ágil basada en el desarrollo dirigido por pruebas (TDD). Como framework, Sinatra (ruby) para desarrollar la aplicación, integrar APIs de terceros, y hacer uso de varias nubes de producción a la vez y también herramientas de soporte para el desarrollo de software como son Github y Travis, entre otras.

1.2. Antecedentes y estado actual del tema

Los antecedentes y estudio de campo no sólo se centran en el tipo de aplicación que vamos a desarrollar, también tendremos en cuenta el entorno de producción, es decir, la nube donde estará disponible la aplicación web. Por lo tanto, tenemos dos puntos iniciales de estudio:

- Sobre la aplicación:
  - ¿Qué ofrece y por qué?
  - ¿Qué otras aplicaciones similares existen?
  - Estudio del sector del mercado.
  - Soporte y aspectos de la usabilidad.
- Y respecto a la nube:
  - ¿Cuál es la mejor nube que se adapte a nuestras necesidades?
  - ¿Qué lenguajes y frameworks soporta?
  - Período, costos, capacidad, etc.

Aplicación

La finalidad de esta aplicación nace de la necesidad de optimizar los gastos, tanto en negocios de restauración como en los propios hogares. Hacer la comida controlando gastos es posible. En la actualidad existen en el mercado aplicaciones con esta idea, algunas de las mas destacadas son:

- Recipe Cost Calculator: Quizás sea la aplicación base como referencia, su fácil gestión y múltiples funciones la hace una herramienta útil y potente. Esta principalmente orientada a negocios.
- Recipe Costing: Esta aplicación va más allá de las funcinalidades básicas, presenta extras como cálculos de menús, gestión de inventario, órdenes de compra a proveedores, etc.
- Pearson Kitchen Manager: Se trata de una API, un banco de información que contiene más de 3000 mil recetas etiquetadas y con otra información como sus valores nutricionales.
- Recipe Costing Calculator: Es más sencilla que las anteriores pero se trata de App disponible en iTunes. Las cosas más sencillas puden ser las más útiles, hay que tener en cuenta que un cocinero/a prefiere una tablet a un ordenador en la cocina.

La segmentación del mercado está orientado especialmente a negocios de restauración: comedores, restaurantes, bares, pastelerías, panaderías, etc. Por otro lado, existe otro sector que no se tiene tan en cuenta debido a que no genera tantos beneficios, se trata de los hogares. Las personas también pueden hacer uso de esta herramienta pues sus necesidades son las mismas pero a menor escala. En el próximo capítulo **??**, se describe toda la funcionalidad de la aplicación, al igual que nos enseña como usarla y veremos las diferencias en la aplicación en función de a qué mercado está dirigido.

Por último, hay que tener en cuenta el soporte para la aplicación. Aprovechando la infraestructura y poder de Internet, la mejor opción es crear una aplicación web, que aunque en esta primera versión se diseñará de forma adaptativa, la idea es poder utilizarla en el futuro en dispositivos móviles y tablets. Además, se ha tenido en cuenta los aspectos de usabilidad durante su diseño.

Nube (cloud)

Inicialmente la idea es trabajar en la nube de Google. Se trata de una plataforma como servicio (PaaS), la cual permite crear y mantener de forma sencilla una aplicación en la infraestructura de Google. Además permite una fácil escalabilidad de transeferencia de datos y almacenamiento gracias a sus módulos.

Aprovechando los conocimientos adquiridos durante los últimos cursos en ruby y sus variedad de frameworks (Ruby on Rails, Padrino, Sinatra) lo usaré para crear la aplicación. Tras investigar y ver los servicios que ofrece Google App

Engine (GAE) parece viable la puesta en marcha de la aplicación. GAE soporta cuatro lenguajes y con correspondientes frameworks:

- Python con webapp2 y Jinja2.
- Java con maven.
- PHP con Cloud SQL.
- Go con el paquete html/plantilla.

Sin embargo, también es posible hacerlo funcionar en ruby con la ayuda de java, juntos forman Jruby, el cual es una implementación 100 % del lenguaje ruby. Además funciona como lenguaje embebido dentro de la máquina virtual de Java. Gracias a esta capa, podré programar en ruby y Jruby establecerá una capa intermedia entre el código fuente en ruby y el servidor en java. Para interactuar con la aplicación en la nube provee de una herramienta o kit, en este caso se trata de una gema llamada google-appengine, sin embargo, la gema dedicada a tal fin esta desfasada y el proyecto ha sido archivado.

Tras la incompatibilidad del proyecto de Jruby en GAE busco la opción más parecida posible que me ofrezca soporte, en este caso Google Cloud Platform. Tras investigar encuentro multitud de lenguajes compatibles, entre ellos ruby, sin embargo, los servicios de esta nube son de pago y aunque existe una versión demo durante 60 días. Debido al tiempo que emplearía en el estudio, desarrollo y pruebas, este período puede resultar corto. Me pongo en contacto con el servicio de clientes y soporte de Google para bajar otras opciones, pero no ofrecen nada que me sea viable para desarrollar mi proyecto.

De esta forma se modifica el requisito inicial del TFG de usar los servicios de la nube de Google para encontrar y usar otros entornos de producción. La idea es intentar usar más de uno, demostrando la modularidad de la aplicación, esto significa que el código fuente de la aplicación debe ser único para las distintas nubes de producción. El siguiente paso es investigar las nubes que se ofertan. La siguiente tabla muestra un resumen:

Una vez analizadas se proponen como entornos de producción para la aplicación a desarrollar las nubes Heroku y Openshift, de las cuales se entrará en detalle en el capítulo 3 **??**.

1.3. Metodología de trabajo

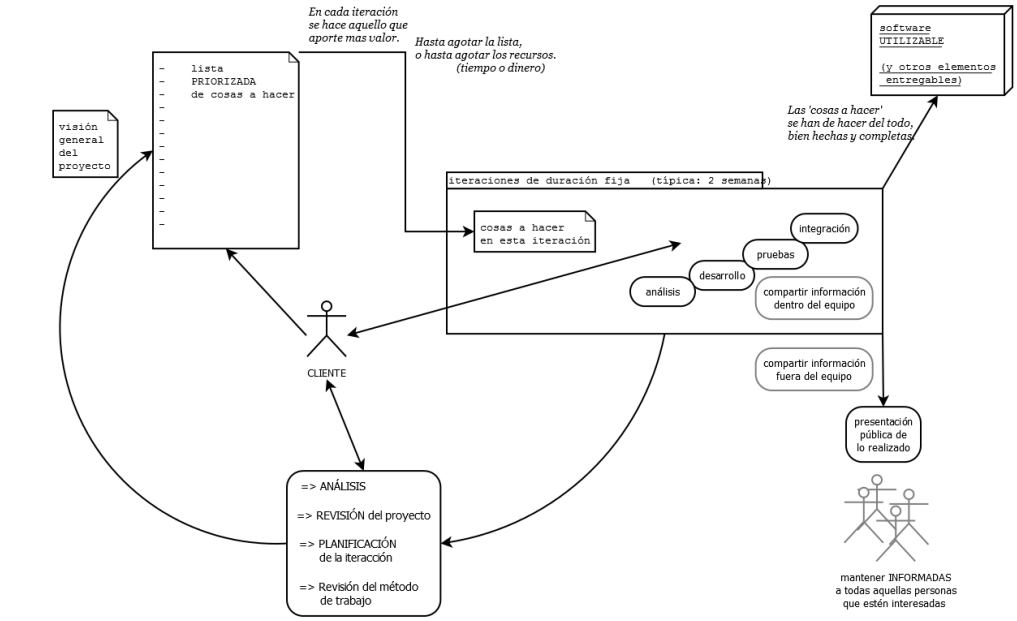
Durante la investigación y desarrollo de este proyecto se ha apostado por las metodologías ágiles, si bien no ha sido en un grupo de trabajo, se puede aplicar en lo que se refiere al análisis, al desarrollo y diseño de la aplicación y a las reuniones con el *cliente*, cuyo rol lo asume el tutor junto con el de *jefe de proyecto*.

Estos métodos están basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan mediante la colaboración de grupos.

Nube	Comentario
Google	De pago. Prueba gratuita de 60 días. Multitud de servicios, distintos tipos de bbdd y soporta muchos lenguajes y frameworks.
Azure	De pago. Prueba de 30 días. Soporta varios lenguajes.
cloud9	Se trata de un entorno de trabajo, una nube orientada al desarrollo, no es lo que buscamos.
nitrous	Es similar a la anterior. Orientada al trabajo colaborativo en entornos de desarrollo.
dotCloud	Esta es una propuesta válida. Se trata de una nube de producción con servicios de pago y versión básica gratuita. Soporta varios lenguajes. El uso de módulos como bbdd o rendimiento del servidor son de pago.
Heroku	Ideal para experimentar con aplicaciones en la nube en un módulo limitada. Su uso es ilimitado, pero tiene algunas condiciones. Soporta varios lenguajes y frameworks.
OpenShift	Es una alternativa muy interesante. Existe la opción de uso gratuito de hasta 3 aplicaciones. Soporta varios lenguajes y distintas versiones del mismo, así como 3 tipos de bbdd (PostgreSQL, MySQL y MongoDB) y los cartuchos (módulos) que añadamos son de pago. Podemos crear y subir nuestros propios cartuchos.
appfog	De pago con versión de prueba de 30 días. Soporta varios lenguajes. El enfoque de esta PaaS es facilitar el trabajo al equipo de desarrollo y que solo se centre en la aplicación y en los datos, mientras la nube gestionaría el resto.
Amazon	Presenta una gran infraestructura de servicios y recursos. Es de pago pero tiene una versión de prueba de 12 meses. Soporta varios lenguajes y dispone de multitud de productos orientados a distintos tipos de mercado.
DigitalOcean	Es de pago. Provee de infraestructura remota para que el desarrollador pueda lenvantor el tipo de servidor que desee. Soporta varios lenguajes y frameworks.
Cloud66	Es de pago pero tiene un perídodo de 14 días de prueba. Esta orientada a mejora las aplicaciones que ya tenemos en otras nubes, en cuanto aspectos de crecimiento y seguridad.

Cuadro 1.1: Production Clouds

La idea es minimizar riesgos desarrollando software en períodos cortos. Este período se llama **iteración** cuya duración se define en función de los requisitos del proyecto, especialmente recursos de tiempo y humano.



Cada iteración del ciclo de vida incluye:

- Planificación: organización del trabajo durante una iteración. En este caso, la planificación corresponde con las tutorías y seguimiento semanal con el tutor.
- Análisis de requisitos: cada nueva funcionalidad o idea para la aplicación debe ser estudiada. Así, si por ejemplo, si quiero integrar una API debo considerar si es útil o no, si es compatible con los entornos de producción y con el lenguaje, documentarme para su integración en la aplicación, etc.
- Diseño: interfaz de la aplicación, características y aspectos de usabilidad.
- Codificación: corresponde por lo general a escribir código, aunque esto no se limita solo a realizar la propia aplicación, además se incluyen los tests y todos aquellos prototipos y pruebas durante la etapa de estudio de campo.
- Revisión: incluye superar los tests con éxito y la aceptación por parte del cliente.
- Documentación: para este trabajo en concreto, se realizó un diario semanal, cuya síntesis se engloba en este documento.

Durante este proyecto se ha seguido dicho ciclo de vida con especial importacia. He intentado adaptar la metodología Scrum a este trabajo, aunque no se ha tenido en cuenta el aspecto de los roles. En Scrum se realizan entregas parciales y regulares del producto final. Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.