PLAN DE GESTIÓN, ANÁLISIS, DISEÑO Y MEMORIA DEL PROYECTO

Proyecto Software

Proyecto Software | Ebrozon Development Team

Ebrozon Development Team

Contenido

1. ORGANIZACIÓN DEL PROYECTO

El proyecto sobre el que trata este documento consiste en la creación de un sistema de compra-venta de objetos a través de una aplicación web.

El sistema permite publicar anuncios con objetos que se deseen vender, estos productos en venta son mostrados por el sistema y pueden ser comprados mediante el precio fijado por el vendedor o bien se puede enviar una petición de compra para llegar a un acuerdo entre las partes interesadas. Además, existe la posibilidad de que el vendedor pueda poner el producto a subasta y los posibles compradores pujen hasta que un único comprador se haga con dicho producto. El sistema dispone de filtrado para búsquedas para mostrar solo aquellos productos en los que se tenga interés, así como un sistema de geolocalización, el cual permite mostrar productos de vendedores cercanos. Por último, el sistema facilita la comunicación entre comprador y vendedor una vez llegado a un acuerdo sobre el precio de un producto para que la transacción sea ejecutada como deseen.

El propósito del desarrollo de este sistema descrito es la venta del mismo, por el precio acordado con el cliente y cumpliendo con los requisitos establecidos por el mismo, los cuales han sido brevemente descritos en el párrafo anterior.

El objetivo de este proyecto es la realización del sistema descrito partiendo desde cero, cumpliendo los requisitos acordados con el cliente al inicio del mismo, así como la realización de pruebas sobre el sistema final para verificar que se cumplen, además del correcto funcionamiento del sistema. También consta como objetivo la documentación de los pasos a seguir para realizar el objetivo anterior, los cuales quedan reflejados en este documento.

El sistema consta de dos tipos de clientes, un cliente web y un cliente móvil, este último desarrollado de forma nativa para Android, un único servidor y una única base de datos.

Se han acordado con el cliente cuatro entregas, la primera de ellas se llevará a cabo el 6 de marzo de 2019, en la cual se entregará un prototipo de la interfaz de usuario para que el cliente dé el visto bueno a una primera versión sobre la cual irá evolucionando, además de recibir posibles ideas o mejoras por su parte. La segunda entrega se llevará a cabo la segunda semana de abril donde se presentará una primera versión funcional del sistema con el objetivo de obtener retroalimentación por parte del cliente, con el fin de llevar a cabo posibles cambios o seguir por una determinada línea. La tercera entrega se llevará a cabo la tercera semana de mayo, en la que se entregará una versión muy desarrollada, pero no final, del sistema, con el fin de obtener posibles cambios de última hora y con el tiempo suficiente para que estos puedan llevarse a cabo. La entrega final ha sido acordada para final de mayo, en la cual se presentará al cliente la versión final del sistema. En las reuniones que impliquen la presentación de una versión del sistema, se hará entrega también de las fuentes que compongan cada versión.

En los siguientes apartados de este documento se describen la organización del mismo, miembros del equipo, sus roles y en que trabaja cada uno (apartado 2); como se ha construido el sistema, que procesos se han seguido, que herramientas se han utilizado, etc. (apartado 3); fase análisis y fase de diseño del sistema (apartado 4); memoria del proyecto (apartado 5); conclusiones obtenidas por el equipo de desarrollo una vez finalizado el proyecto (apartado 6).

2. ORGANIZACIÓN DEL PROYECTO

Integrantes	Rol	Responsabilidades
Saúl Alarcón Cano	Director del proyecto Coordinador y desarrollador del grupo de back end	Realización de tareas de gestión (edición de memoria y otros documentos) Responsable de redactar algunas actas en reuniones con el profesor Control de la distribución de trabajo (elaboración de calendario) y revisión de esfuerzos Encargado del diseño e implementación de la base de datos Desarrollador de modelos, repositorios y controladores de la API Encargado del despliegue del back end sobre Heroku
Alejandro Cano Somalo	Desarrollador de apoyo para el grupo de back end	Realización de tareas de gestión (edición de memoria y otros documentos), concretamente el punto 5 del plan de gestión, análisis, diseño y memoria, etc. Desarrollador de un para el back-end
Andrés Gavín Murillo	Coordinador y desarrollador del grupo de front end de móvil	Realización de tareas de gestión (edición de memoria y otros documentos) Control de la distribución de trabajo y coordinación dentro del grupo front end de móvil Previsiblemente encargado de unificar las partes de la

		aplicación móvil y llevar a cabo el despliegue
Eduardo Gimeno Soriano	Desarrollador de apoyo para el grupo de back end Encargado de la documentación del análisis y diseño del sistema	Realización de tareas de gestión (edición de memoria y otros documentos) Diseño del sistema mediante diagramas Diseño de un par de modelos y un par de controladores para el back-end
Félix García Rodríguez	Desarrollador del grupo de front end de la aplicación web	Realización de tareas de gestión (edición de memoria y otros documentos) Creación del diagrama E/R a usar en la base de datos Diseñar pantallas de la aplicación de la web Implementar la lógica de la aplicación de la web
Israel Solanas Navarro	Desarrollador del grupo de front end de la aplicación web	Realización de tareas de gestión (edición de memoria y otros documentos) Diseñar pantallas de la aplicación de la web Implementar parte de la lógica de la aplicación de la web Responsable de redactar algunas actas en reuniones con el profesor

Jorge Fernández Muñoz	Coordinador y desarrollador del grupo de front end de web	Realización de tareas de gestión (edición de memoria y otros documentos) Diseñar pantallas de la aplicación de la web Implementar la lógica de la aplicación de la web Encargado de llevar a cabo el despliegue de la web
Sergio Álvarez Peiro	Desarrollador del grupo de front end de la aplicación móvil	Realización de tareas de gestión (edición de memoria y otros documentos) Diseñar pantallas de la aplicación de Android Implementar la lógica de la aplicación de Android
Víctor Sisqués Cortés	Desarrollador del grupo de front end de la aplicación móvil	Realización de tareas de gestión (edición de memoria y otros documentos) Realización del diagrama de Gantt del proyecto y división del trabajo Diseñar pantallas de la aplicación de Android Implementar la lógica de la aplicación de Android

3. PLAN DE GESTIÓN DEL PROYECTO

3.1. PROCESOS

3.1.1. PROCESOS DE INICIO DEL PROYECTO

Para el despliegue de la aplicación se va a hacer uso de Google Cloud, aunque posteriormente, con la entrega del software al cliente, este podrá desplegarlo en el servidor que prefiera utilizar sin ningún tipo de problema. En cuanto a las distintas aplicaciones, la aplicación web funcionará en los navegadores Chrome, Firefox y Opera, y la aplicación para Android podrá ser utilizada en la versión de Android 5.0 en adelante, siendo realizadas las pruebas sobre un Android 5.0.

Para los nuevos miembros del proyecto, en función del equipo al que sea asignado, deberá tener en cuenta las siguientes cuestiones:

En el equipo de backend, necesitará instalar Spring Tool Suite para el desarrollo del código fuente, pudiendo apoyarse en el siguiente tutorial de configuración básica para realizar la configuración inicial del proyecto, también deberá conocer el lenguaje Spring, pudiendo seguir las distintas guías de diseño oficiales disponibles en https://spring.io/guides .

En el equipo de FrontEnd, el entorno de desarrollo utilizado es WebStorm, y en cuanto a los lenguajes básicos utilizados(HTML,CSS,JavaScript) se recomienda utilizar las guias de cada lenguaje que dispone w3schools,https://www.w3schools.com/. En cuanto a lenguajes específicos, se va a utilizar Vue, por lo que se recomienda realizar el tutorial oficial del lenguaje, disponible en https://vuejs.org/v2/guide/ .

En el equipo de desarrollo Android, se necesitará instalar Android Studio para realizar el desarrollo, así como linkar el mismo con el Repositorio de GitHub habilitado para la aplicación Android del proyecto, en cuanto a lenguajes se hace uso de Java y XML, se recomienda hacer uso de la guía de Android estudio disponible en https://developer.android.com/studio/intro

3.1.2. PROCESOS DE EJECUCIÓN Y CONTROL DEL PROYECTO

A la hora de distribuir las tareas del proyecto, principalmente se harán tres equipos:

Front-End en el que se encuentran Félix, Israel y Jorge, siendo este último el encargado del equipo.

Back-End en el que se encuentran Andrés, Alejandro y Saúl el encargado de este.

Android en el que se encuentran Sergio, Víctor y Eduardo, siendo este último el encargado del equipo.

La comunicación entre los miembros del grupo más informal se realiza a través de la aplicación de mensajería WhatsApp, a través del grupo Ebrozon (nombre de la aplicación). Además, existen 3 grupos diferentes en función del equipo de la aplicación Front-end, Back-end y Android.

Para el tema de las reuniones del grupo y paso de archivos de importancia media, se emplea la aplicación Discord, y se emplea un servidor con el nombre Ebrozon. Dentro de este se encuentran 4 canales de voz, uno para cada equipo de la aplicación, y un canal general en el que se realizan las reuniones entre todos los miembros del grupo o entre los líderes de cada apartado

para realizar reuniones de carácter general o hacer puesta en común de los diferentes apartados. Además, existe un canal de texto para cada canal de voz y un canal de texto llamado archivos al que se suben archivos de importancia media, como notas informales de un acta y documentación semi-elaborada en forma de resumen de la documentación formal.

Por último, se emplea el repositorio GitHub, para trasferir los archivos propios de la aplicación, las actas, memorias, y otros ficheros clave en el desarrollo de la aplicación y entrega al cliente. Dentro de GitHub existen un repositorio para cada grupo y un repositorio de documentación que puede ser modificado por todos los miembros del equipo. Es importante destacar que en los repositorios destinados a cada equipo solo puede mergear en la rama master el encargado de dicho equipo.

En las actas de las reuniones se especificará la fecha, miembros presentes y objetivo.

Se realizará aproximadamente una reunión mensual entre todos los miembros del proyecto.

El proceso que se sigue en cada reunión es la siguiente:

1.-Puesta en común de los avances de los miembros del grupo, por ejemplo, coordinación de formatos entre cliente Android y cliente web, o adaptación de los códigos del cliente a las funcionalidades actuales del servidor.

2.-Revisión de las tareas pendientes en el futuro, bien sean del issue traker de GitHub, o la creación de nuevas, las cuales se subirán al issue tracker.

3.-Propuesta del objetivo de avance hasta la siguiente reunión y ver si coincide con el del calendario, sino proponer una modificación para este.

Además de en las reuniones de todos los miembros del grupo, las tareas pueden ser creadas en las reuniones de los encargados de cada grupo o en las reuniones de grupo. Estas son asignadas tras un diálogo entre los mismos, según su disponibilidad. En el caso de que los miembros del grupo no lleguen a un acuerdo el encargado de cada grupo decidido cómo se organizan las tareas.

Por lo general, los tres grupos han decidido hacer un reparto horizontal de las tareas, ocupándose cada uno de unos módulos de la base de datos para el equipo de Back-end o de diferentes pantallas para el equipo de Android o Front-end.

El reparto de tareas de la memoria será diferente, pues algunos apartados son redactados por los miembros del grupo que tienen una mayor disponibilidad de forma voluntaria, y otros como el calendario solo pueden ser redactados por algunos usuarios (generalmente encargados del proyecto). Además, existen apartados más específicos para los que un miembro que ya haya comenzado a redactarlos está más preparado que el resto de miembros del equipo, por lo que preferiblemente ese miembro del equipo siempre es el encargado de ese apartado para ahorrar tiempo.

El proceso de validación de esfuerzos de los diferentes miembros del grupo se realiza una vez por semana y es el siguiente:

1.-Se revisan las horas dedicadas por cada uno de los desarrolladores, comprobando que se han dedicado al menos 5 horas.

2.-Se revisan las tareas realizadas por cada miembro del equipo.

3.-Se revisa el número de commits de cada miembro.

4.-Evaluación entre pares sobre la repercusión del aporte de cada miembro del grupo al sistema.

La entrega de los resultados se hará mediante de la siguiente manera:

-Código fuente necesario para el despliegue el GitHub Ebrozón teniendo la última versión disponible en la rama Master de cada equipo Front-end los fuentes del cliente web, Back-end los fuentes del servidor y Android los fuentes del cliente móvil.

-Base de datos y aplicación web desplegadas en Heroku para mostrar el funcionamiento, y apk android.

En caso de disputa el primer encargado de solucionarla será el coordinador de cada subgrupo y si no será un consejo formado por todos los miembros del grupo quien lo decidirá.

3.1.3. PROCESOS TÉCNICOS

Herramientas y software para el desarrollo del proyecto:

Para el control de versiones del software desarrollado en todo el proyecto será git, mediante un repositorio público en GitHub.

El desarrollo del software en la parte del Backend se utiliza el entorno Eclipse de Spring. En la parte del Frontend para el desarrollo de la web se trabajará con editores con soporte javascript y HTML como Atom y el entorno WebStorm. También para el diseño de pantallas Boostratp Studio. Para el desarrollo de la aplicación en Android se usa el entorno de Android Studio haciendo uso de su integración de git para el control de versiones con el repositorio del Frontend de la aplicación.

Para el despliegue local del backend durante el desarrollo se usará Xampp debido al conocimiento de uso de esta y posteriormente se cambiará a la plataforma en la nube Heroku para poder realizar pruebas con el front-end.

Proceso de despliegue del Backend:

1. El encargado del despliegue unirá todos los fuentes del repositorio del Backend en un único proyecto, asegurándose de que todos están actualizados.

2. El encargado realizará un despliegue local de prueba sobre XAMPP y Spring Tool Suite, comprobando que no haya errores ni de compilación ni de despliegue (por ejemplo, la no existencia de una columna en la base de datos).

   En caso de encontrarse errores, si son simples y de rápida solución, los corregirá el encargado en el momento, en caso contrario, se le notificará el problema al desarrollador del fuente problemático para que lo solucione, y una vez corregido, se volverá al paso 1.

3. El encargado realizará pruebas de las nuevas funcionalidades (en caso de ser posible) mediante peticiones web, para comprobar el correcto funcionamiento.

   En caso de encontrarse errores, si son simples y de rápida solución, los corregirá el encargado en el momento, en caso contrario, se le notificará el problema al desarrollador del fuente problemático para que lo solucione, y una vez corregido, se volverá al paso 1.

4. El encargado realizará el despliegue sobre Heroku, el cual no debería presentar problemas, al haber hecho el paso 2 (los errores que pueden surgir en el despliegue son los mismos).

   En caso de problemas durante el despliegue, el encargado solucionaría estos.

Proceso de prueba de funcionalidades manual de la aplicación Android:

1. El encargado de la prueba mediante Android Studio se sitúa en la rama del repositorio develop y hace pull para asegurarse de tener todos los fuentes actualizados.

2. Se construye el proyecto en el estado que se encuentre para comprobar que no se producen errores. En caso de encontrarse errores se notifica a la persona del equipo que ha implementado esa pantalla o funcionalidad para que se arregle. Si es posible continuar la prueba del resto de funcionalidades se excluyen del proyecto y se sigue, sino se espera a una solución y se vuelve al paso 1.

3. Se lanza la aplicación en el emulador de Android Studio o sobre un dispositivo real conectado con una versión mínima de Android 5.1.

4. Una vez lanzada la aplicación el encargado realiza las pruebas necesarias manualmente probando las nuevas funcionalidades, comprobando que los resultados son correctos y comprobando que no haya problemas en situaciones de error.

5. Por último, si las nuevas funcionalidades implementadas funcionan y superan las pruebas del paso anterior el encargado informa al resto del equipo con las posibles mejoras en las pruebas observadas. Si se encuentra en el final de una iteración y el encargado de hacer el proceso es el encargado del equipo de la aplicación hará un merge de la rama develop a la rama master.

El despliegue del sistema completo se hace de manera manual, primero realizando el despliegue del Backend siguiendo el primer proceso mencionado, y probando la app y la web en la versión en la que se encuentren para comprobar su integración con el Backend.

3.2. PLANES

3.2.1. PLAN DE GESTIÓN DE CONFIGURACIONES

La convención de nombres utilizadas para nombrar los distintos archivos sería la siguiente: nombre.tipo.versión.revisión

Cada vez que se cree una nueva versión, pero sus cambios sean menores, como resolución de errores, se modificará su número de revisión, pero no de versión.

Las versiones solo se modificarán cada vez que se produzcan cambios suficientemente importantes, como por ejemplo la implementación de una nueva funcionalidad.

Para generar la documentación de los ficheros se emplearán herramientas contrastadas que la generen de forma automática a partir de los comentarios del código fuente (p.ej. javadoc en java).

Además, en los ficheros de documentación en los que se expliquen las diversas funcionalidades que tiene la aplicación y que errores se han ido resolviendo, cuando estos sean de una nueva versión o revisión solo se ofrecerá la información sobre los cambios que existan entre esta y la versión o revisión anterior, pero siempre que se cambie la versión se documentarán los cambios

respecto a la primera revisión de la versión anterior (p.ej. La versión 2.8 solo contendrá las novedades respecto a la versión 2.7, pero la versión 3.0 contendrá todos los cambios que hayan sucedido desde la versión 2.0 aunque la mayoría se hayan documentado ya en las revisiones).

El estándar a utilizar en el código, puesto que va ser una aplicación web, se utilizará html5.

Los responsables de realizar la puesta en marcha se irían turnando. Una vez a la semana se elegirá un encargado que será el responsable de realizar la puesta en marcha de la versión más reciente producida.

Otras actividades como la creación de copias de seguridad y semejantes se realizarían de manera automática gracias a las prestaciones de GitHub.

El repositorio que se creará con todos los archivos referentes al proyecto se encontrará en GitHub, para que todos los integrantes del proyecto puedan acceder fácilmente a los archivos. Además, GitHub tiene integrado un gestor de incidencias, lo que hace que no sea necesario realizar estas gestiones mediante herramientas externas.

El proyecto estará dividido en varios repositorios. Uno para la documentación, otro de pruebas, otro para las versiones funcionales del proyecto y dos para el desarrollo del Front-End y el Back-End. Para evitar que se modifique el mismo fichero por dos personas al mismo tiempo y evitar problemas, cada equipo tendrá más sub-ramas de desarrollo, por ejemplo una para cada miembro del equipo, que serán actualizadas con cambios no siempre funcionales y cuando sean más estables se volcarán a la rama de desarrollo principal.

En la rama principal de cada uno de los repositorios sólo podrá haber una versión funcional del sistema, que antes de ser subida será sometida a diferentes test automáticos, entre los que se incluirán test para comprobar la estabilidad del sistema (pruebas de sobrecarga) y test que revisarán las acciones disponibles para comprobar los requisitos que se han resuelto.

En el repositorio de pruebas, se subirán los ficheros que hayan sido modificados y no se hayan vuelto a revisar si el comportamiento es correcto, además de los ficheros que realizan la pruebas.

Para que lo desarrollado en cada uno de estos repositorios pase al repositorio funcional, cada líder de las respectivas partes revisará el código actualizado y si todo está correcto se considerará válido.

Todos son capaces de modificar los ficheros de los repositorios excepto en el de las versiones, el cual solo podrán subir archivos y modificarlos los líderes del Front-End y el Back-End.

3.2.2. PLAN DE CONSTRUCCIÓN Y DESPLIEGUE DEL SOFTWARE

Durante el desarrollo, toda construcción y despliegue del back-end (servidor API y BD) para pruebas se va a llevará acabo con la aplicación XAMPP en su versión 3.2.2, compilada el 12 de noviembre de 2015, y Spring Tool Suite, el cual lanza un servidor Tomcat. Estas nos facilitan un despliegue local rápido y sencillo, que nos permite probar nuevas características de forma muy eficiente. Además, usando estas herramientas concretas, nos aseguramos de que todos trabajamos y realizamos pruebas sobre un sistema con unas características concretas iguales para todos. De XAMPP se usarán los módulos de Apache y MySQL para la base de datos,

permitiendo que se conecten a los puertos que vienen configurados por defecto, siendo especialmente importante para el módulo de MySQL, ya que, si no se controlase, muy fácilmente podría producirse un problema de puertos a la hora de juntar las diversas partes.

Para el despliegue real del back-end, se realizará el despliegue del servidor y la base de datos sobre la plataforma Heroku, usando para la base de datos el add-on Heroku-Postgresql, ya que, es open source y era el que mayor almacenamiento gratuito proporcionaba. Las aplicaciones web y móvil se conectarán a este a la hora de realizar peticiones, tanto durante las pruebas como durante el funcionamiento real. Gracias a esto dispondrán de un servidor abierto constantemente para realizar sus pruebas sin depender del equipo del back-end, mientras el equipo de este continua el desarrollo de nuevas funcionalidades.

La elección de Heroku como plataforma de despliegue para el servidor se debe a su facilidad y rapidez de uso, al encargarse la propia plataforma del despliegue como tal tras haberle concretado algunos detalles como el lenguaje y el tipo de base de datos. Al despreocuparnos despliegue, podremos centrarnos en el desarrollo como tal.

Se procurará que 1 vez a la semana (ya que el back-end es simple de lanzar) se junten todos los cambios realizados en el back-end, para asentarla como una última versión, sobre la que se deberá trabajar la semana siguiente. Esta versión será montada por un encargado de despliegue.

El proceso de despliegue con XAMPP y Spring Tool Suite es el siguiente:

1. Lanzar la base de datos, para lo cual solo es necesario abrir XAMPP y darle a start a Apache y MySQL.
2. Aunar los ficheros fuente en un mismo proyecto Spring, lo cual supone solamente incluir los fuentes nuevos en los paquetes correspondientes.
3. Comprobar posibles errores de compilación, y corregirlos en caso de haber.
4. Darle a Run al proyecto de Spring Tool Suite, y en caso de mostrar errores durante el despliegue corregirlos, y darle a Run de nuevo.
5. Nota: en caso de tener el proyecto corriendo, si se hace alguna modificación en los fuentes, al hacer ctrl+s, obviamente se guardan, y se relanza automáticamente.

El proceso de despliegue sobre Heroku (una vez creada la BD sobre Postgresql) es el siguiente:

1. En la carpeta de nuestro PC donde esté el repositorio dedicado a Heroku, en el proyecto ya configurado, se actualizan los fuentes correspondientes a modelos, repositorios y controladores.
2. Se abre la terminal, se accede a dicha carpeta desde ella y se ejecuta “git add .”
3. Se ejecuta “git Commit –m mensaje”
4. Se ejecuta “git push heroku master”

Configuración para desarrollo en XAMPP y Sprint Tool Suite:

Ruta base de datos: mysql://localhost:3306/ebrozon_bd

Puerto base de datos: 3306

Puertos apache: 80 y 443

Usuario base de datos: root (sin contraseña)

Ruta peticiones: localhost:8080

Configuración heroku:

Host base de datos: ec2-184-73-210-189.compute-1.amazonaws.com

Ruta base de datos:

postgres://ec2-184-73-210-189.compute-1.amazonaws.com:5432/d3311baks5knve

Puerto base de datos: 5432

Usuario base de datos: tvsujbnpveceag

Contraseña base de datos:

db96040cf34b037240cce6d42c971ee642aa8a5ee31206c43132a43de4411c3b

Ruta peticiones: protected-caverns-60859.herokuapp.com/

Respecto a la aplicación móvil solo nombrar que, obviamente, no necesita un despliegue, ya que se ejecuta localmente, y ejecutará sus peticiones sobre el servidor ya nombrado.

Y sobre la web, decir que una vez la web sea funcional y falten mejoras y correcciones menores, se realizará su despliegue sobre un servidor cloud, probablemente Google Cloud, que se configurará en dicho momento, y se realizarán los métodos de automatización correspondientes para facilitar el relanzamiento de la web, para poder realizar actualizaciones de forma eficiente. Este punto del desarrollo se espera que al menos unas 2 semanas antes de la entrega final, para tener tiempo de solucionar todos los problemas que puedan surgir al desplegarlo sobre un entorno real. Puesto que, en todo momento durante el desarrollo, se habrán estado comunicando con el servidor final, este punto no debería generar problemas más allá de complicaciones del despliegue de la web como tal.

3.2.3. PLAN DE ASEGURAMIENTO DE LA CALIDAD

Se seguirán las guías de estilo de cada herramienta utilizada, es decir, para Java https://google.github.io/styleguide/javaguide.html, para HTML https://google.github.io/styleguide/htmlcssguide.html, para Javascript https://google.github.io/styleguide/jsguide.html, para Spring https://github.com/spring-projects/spring-framework/wiki/Code-Style y para MySQL .

Para el diseño gráfico de las GUI se seguirá el estándar de Google Material Design https://material.io/design/ que simplifica el apartado gráfico para el usuario y se garantizará la usabilidad con unas acciones sencillas y una respuesta rápida de estas.

La documentación será realizada a través de herramientas de generación de documentación automática, en este caso Javadoc.

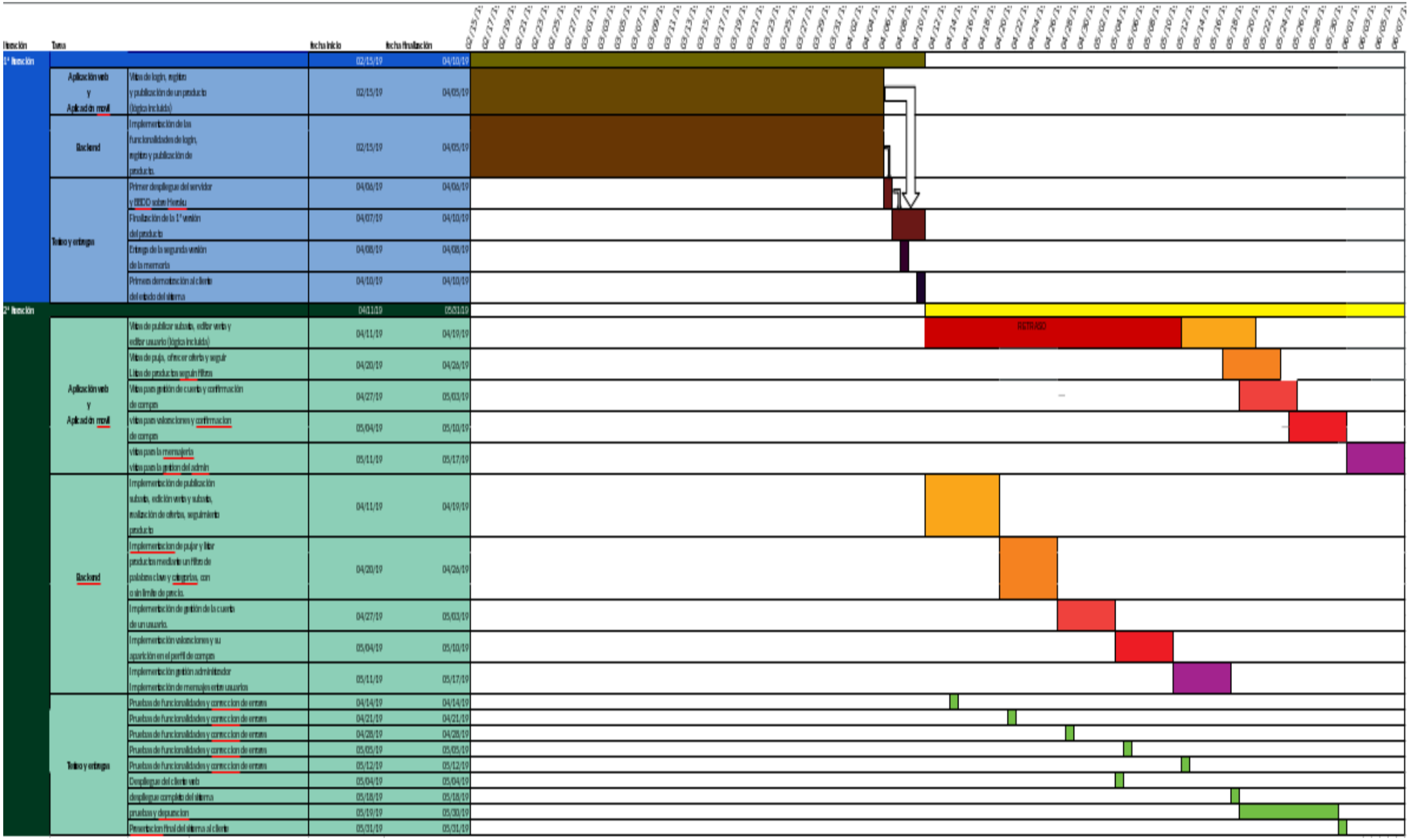
Para garantizar la calidad de código se seguirá una jerarquía de los miembros del grupo facilitada por las herramientas que incluye GitHub, entre las que se encuentran la división de los miembros en tres equipos (Back-End, Front-End y Android) con sus respectivas ramas de desarrollo. Además, la rama principal sólo podrá ser actualizada por el responsable de cada equipo, siendo la rama de desarrollo la única modificable por los demás miembros de cada equipo.

3.2.4. CALENDARIO DEL PROYECTO Y DIVISION DEL TRABAJO

En la primera iteración del proceso de diseño nos centraremos en desarrollar las funcionalidades principales del sistema, mientras que en la segunda iteración se corregirán todos los errores encontrados en la primera, se implementarán las funcionalidades secundarias y se afinara el diseño de la página web y de la aplicación Android para que sean más agradables al usuario.

Para la primera iteración se planea permitir la creación, edición y borrado de usuarios con sus credenciales básicos: nombre de usuario, nombre real, correo, contraseña y ciudad. También se permitirá la puesta en venta de objetos para compra inmediata y para su compra mediante subasta; los objetos se verán definidos por un nombre, foto, descripción, precio y etiquetas. Una etiqueta es una o más palabras clave que el sistema utilizará para clasificar los productos. También se permitirá la edición de la información de productos ya publicados por parte del cliente, además de otorgar la posibilidad al usuario de retirar estos. Adicionalmente, el sistema permitirá a los usuarios realizar ofertas por un producto y a los vendedores aceptar o rechazar estas. Finalmente, el sistema permitirá la búsqueda de productos mediante palabras clave.

Para la segunda iteración se finalizarán los requisitos que, por falta de tiempo, no pudieron ser completados en la primera y se añadirán funcionalidades al sistema. Estas funcionalidades son: permitir información adicional en los perfiles de usuario (como puede ser una foto de perfil, una descripción, etc.), un sistema de calificación/reportes de usuarios tras una compra/venta, el bloqueo de una cuenta en caso de que tenga varias calificaciones negativas y/o reportes, el envío de mensajes entre usuarios, la posibilidad de añadir productos a listas de seguimiento y se mejorara el sistema de búsqueda para que se puedan filtrar los productos en función de localización, categoría, precio etc. Finalmente, el sistema registrara todas las acciones realizadas por los usuarios.


Iteración
Tarea
Fecha Inicio
Fecha finalización
1ª Iteración
02/15/19
04/10/19
Aplicación web y Aplicación movil
Vistas de login, registro y publicación de un producto (lógica incluida)
02/15/19
04/05/19
Backend
Implementación de las funcionalidades de login, registro y publicación de producto.
02/15/19
04/05/19
Testeo y entrega
Primer despliegue del servidor y BBDD sobre Heroku
04/06/19
04/06/19
Finalización de la 1ª versión del producto
04/07/19
04/10/19
Entrega de la segunda versión de la memoria
04/08/19
04/08/19
Primera demostración al cliente del estado del sistema
04/10/19
04/10/19
2ª Iteración
04/11/19
05/31/19
Aplicación web y Aplicación movil
Vistas de publicar subasta, editar venta y editar usuario (lógica incluida)
04/11/19
04/19/19
RETRASO
Vistas de puja, ofrecer oferta y seguir
Lista de productos segun filtros
04/20/19
04/26/19
Vistas para gestión de cuenta y confirmación de compras
04/27/19
05/03/19
vistas para valoraciones y confirmacion de compras
05/04/19
05/10/19
vistas para la mensajeria
vistas para la gestion del admin
05/11/19
05/17/19
Backend
Implementación de publicación subasta, edición venta y subasta, realización de oferta, seguimiento producto
04/11/19
04/19/19
Implementacion de pujar y filtar productos mediante un filtro de palabras clave y categorias, con o sin límite de precio.
04/20/19
04/26/19
Implementación de gestión de la cuenta de un usuario.
04/27/19
05/03/19
Implementación valoraciones y su aparición en el perfil de compras
05/04/19
05/10/19
Implementación gestión administrador
Implementación de mensajes entre usuarios
05/11/19
05/17/19
Testeo y entrega
Pruebas de funcionalidades y correccion de errores
04/14/19
04/14/19
Pruebas de funcionalidades y correccion de errores
04/21/19
04/21/19
Pruebas de funcionalidades y correccion de errores
04/28/19
04/28/19
Pruebas de funcionalidades y correccion de errores
05/05/19
05/05/19
Pruebas de funcionalidades y correccion de errores
05/12/19
05/12/19
Despliegue del cliente web
05/04/19
05/04/19
despliegue completo del sistema
05/18/19
05/18/19
pruebas y depuracion
05/19/19
05/30/19
Presentacion final del sistema al cliente
05/31/19
05/31/19

abril 2019

domingo	lunes	martes	miércoles	jueves	viernes	sábado
31	1	2	3	4	5 Web y apk: vistas de login registro, y publicación de un producto Back: implementación de dichas funcionalidades	6 Primer despliegue del servidor y base de datos sobre Heroku
7 Primera versión de la web y de la aplicación, que permita registrar usuario, logearse y subir un producto	8 Entrega de la segunda versión de la memoria	9	10 Primera demostración al cliente del estado del sistema	11	12 Back: publicación subasta edición venta y subasta, realización de ofertas, seguimiento producto	13 Web y apk: vistas publicación subasta, edición venta, subasta y edición usuario
14 Pruebas de funcionalidades y corrección de errores	15	16	17	18	19 Back: implementación pujar y listar productos por filtro de palabras clave y categorías con límite o sin de precio	20 Web y apk: vistas pujar, ofrecer oferta y seguir, listas de productos según filtros (debería ser la misma para todo)
21 Pruebas de funcionalidades y corrección de errores	22	23	24	25	26 Back: implementación gestión cuenta (ventas activas, historial, productos seguidos	27 Web y apk: vistas para gestión de cuenta confirmación compra
28 Pruebas de funcionalidades y corrección de errores	29	30	1	2	3	4
5	6					

mayo 2019

domingo	lunes	martes	miércoles	jueves	viernes	sábado
28	29	30	1	2	3 Back: implementación valoraciones, y su aparición en perfil confirmación compra	4 Despliegue del cliente web Web y apk: vistas para valoraciones y confirmación compra
5 Pruebas de funcionalidades y corrección de errores	6	7	8 Reunión 5	9	10 Back: implementación gestión admin implementacion mensajes entre usuarios	11 Web y apk: vistas para la mensajería
12 Pruebas de funcionalidades y corrección de errores	13	14	15	16	17 Web y apk: vistas para gestión admin Back: implementación gestión registro acciones y corrección de errores	18 Despligue completo del sistema en la versión actual (faltan algunas funcionalidades menores)
19 Pruebas y depuración	20 Pruebas y depuración	21 Pruebas y depuración	22 Pruebas y depuración	23 Pruebas y depuración	24 Pruebas y depuración	25 Pruebas y depuración
26 Pruebas y depuración	27 Pruebas y depuración	28 Pruebas y depuración	29 Pruebas y depuración	30 Pruebas y depuración	31 Presentación final al cliente del sistema	1
2	3					

	Saúl	Sergio	Eduardo	Félix	Jorge	Israel	Alejandro	Víctor	Andrés
Análisis de requisitos.					TODOS				
Diagramas de análisis del sistema.					TODOS				
Diagramas de diseño del sistema.					TODOS				
Diseño e implementación del interfaz web.				✓	✓	✓			
Diseño e implementación de la lógica del cliente web.				✓	✓	✓			
Diseño e implementación del interfaz móvil.		✓						✓	✓
Diseño e implementación de la lógica del cliente móvil.		✓						✓	✓
Diseño e implementación de la base de datos.	✓								
Diseño e implementación de la capa DAO.	✓		✓				✓		
Pruebas de funcionalidad.					TODOS				
Pruebas de mal uso.					TODOS				

Las tareas de la fase de requisitos, análisis y diseño se han repartido equitativamente entre todos los integrantes ya que todos hemos cursado la asignatura ingeniería software y debido a esto tenemos conocimientos básicos sobre esta fase del diseño de un sistema. Lo mismo ha sucedido con las pruebas. Durante esta primera iteración se realizarán 2 tipos de pruebas; pruebas de mal uso (pruebas en las que el usuario utiliza el sistema de manera incorrecta) y pruebas de funcionalidades (pruebas en las que se comprueba el cumplimiento de los requisitos).

Las personas asignadas a las tareas de diseño de la aplicación web (Diseño e implementación del interfaz web y diseño e implementación de la lógica del cliente), son aquellas con mayor experiencia a la hora de programar en HTML y/o en JavaScript.

Las personas asignadas a las tareas de diseño de la aplicación Android (Diseño e implementación del interfaz web y diseño e implementación de la lógica del cliente), son aquellas con mayor experiencia a la hora de programar en java y/o en XML.

Las personas asignadas a las tareas de Back-End (Diseño e implementación de la base de datos y Diseño e implementación de la capa DAO), son aquellas con mayor experiencia a la hora de programar en java y/o SQL. El trabajo se ha distribuido en base a la modularización del sistema en base a paquetes, y se han centrado en los más cruciales, siendo estos aquellos de los que más depende el sistema.

4.. ANÁLISIS Y DISEÑO DEL SISTEMA

4.1. ANÁLISIS DE REQUISITOS

Código requisito	Descripción
RF-1	- El sistema debe permitir la existencia de usuarios.
RF-2	- Un usuario debe tener un nombre y un correo electrónico.
RF-3	- El sistema debe permitir que un usuario tenga una descripción y una foto.
RF-4	- El sistema debe permitir que un usuario incluya una geolocalización aproximada, es decir, un radio (en kilómetros) en el que se encuentra.
RF-5	- El sistema debe permitir que un usuario modifique su información.
RF-6	- El sistema debe permitir un sistema de calificación de usuarios.
RF-7	- El sistema debe permitir que un usuario pueda valorar a otro usuario.
RF-8	- El sistema debe permitir el bloqueo de la cuenta de un usuario en caso de que tenga varios reportes y un alto porcentaje de las valoraciones que haya recibido sean negativas.
RF-9	- El sistema debe permitir a los usuarios poner en venta objetos, proporcionando información del producto y un precio de compra inmediata.
RF-10	- El sistema debe permitir especificar la siguiente información acerca de un producto: un nombre o título del producto, fotos y una descripción del mismo.
RF-11	- Una etiqueta será una o más palabras clave que se entenderán a modo de categoría, para poder clasificar los productos.
RF-12	- El sistema debe autogenerar unas etiquetas para las ventas y subastas a partir de la descripción y nombre del producto dados.
RF-13	- El sistema debe permitir a los usuarios poner en subasta objetos, siendo estas definidas por un precio mínimo, una fecha límite y un precio de compra inmediata, además de la información del producto.
RF-14	- El sistema debe permitir al vendedor editar tanto la información como el precio de un producto propio antes de que se haya realizado la compraventa.
RF-15	- El sistema debe permitir al vendedor retirar la venta de un producto propio (siempre que el producto no haya recibido pujas o compras inmediatas).
RF-16	- El sistema debe permitir a los usuarios realizar ofertas a los vendedores por un objeto en venta inmediata.

RF-17	- El sistema debe permitir a los vendedores aceptar o rechazar ofertas de un producto propio en venta inmediata.
RF-18	- El sistema debe retirar un producto de la lista de productos en venta, cuando el vendedor ha aceptado una oferta por este o se ha llegado a la fecha límite.
RF-19	- El sistema debe permitir a los usuarios realizar pujas por un producto en subasta.
RF-20	- El sistema debe permitir a los usuarios añadir productos a su lista de "Siguiendo", siendo esta la lista de productos en los que ha indicado que está interesado.
RF-21	- El sistema debe permitir a los usuarios retirar productos de su lista de "Siguiendo".
RF-22	- El sistema debe permitir la búsqueda de productos mediante palabras clave (incluidas en el nombre y descripción de un producto en venta).
RF-23	- El sistema debe permitir filtrar una lista de productos a la venta en función de localización, categoría, precio, provincia, fecha (ordenados de los productos más recientes a los menos recientes y viceversa) etc.
RF-24	- El sistema debe permitir a los usuarios ver las valoraciones propias recibidas y las de otros usuarios.
RF-25	- El sistema debe permitir a los usuarios reportar a otros usuarios en caso de una infracción las reglas de uso (p.e. caso de estafa).
RF-26	- El sistema debe permitir a los usuarios enviar mensajes a otros usuarios.
RF-27	- El sistema debe permitir a los usuarios recibir mensajes de otros usuarios.
RF-28	- El sistema deberá guardar la información de todas las acciones realizadas por los usuarios.
RF-29	- Las acciones de los usuarios abarcan el logearse, el editar su perfil, el seguir productos, el realizar ofertas y pujas por productos, el crear ventas y subastas, el editar la información de una venta y la creación de una etiqueta.
RNF-1	- El producto resultante será una aplicación web soportada en la mayoría de navegadores.
RNF-2	- La aplicación web tendrá un diseño responsive enfocada a los dispositivos móviles.
RNF-3	- El producto resultante será una aplicación web (soportada al menos en Google Chrome) y una aplicación móvil (para sistema operativo Android).

4.2. DISEÑO DEL SISTEMA

Para la aplicación web se plantearon Vue, React y Angular como tecnologías de desarrollo, decidiéndose finalmente el uso Vue. Para la aplicación se barajaron Android Studio (XML y Java), WebPack (HTML) y Flutter (Dart), y se eligió finalmente un desarrollo con XML y Java (Android Studio), por el hecho de que era un entorno y lenguajes con los que ya se tenía experiencia, y porque de este modo la aplicación sería completamente nativa, mejorando el rendimiento y por

ende la experiencia de uso. Y para el servidor de la API se barajaron dos opciones, SQLAchemy (Python) y Spring Boot (Java), eligiéndose finalmente Spring Boot, ya que se tenía mucha experiencia con Java en comparación con Python, junto con una base de datos MySQL para pruebas y Postgresql para despliegue real. Para la interacción entre el cliente (tanto web como móvil) y el servidor se desarrollará una API RESTful, es decir, una API sin estado que responde a las peticiones basándose únicamente en la información de la base de datos, y la pasada en la propia petición. Se planteó el no hacerla REST, pero se consideró que este enfoque facilitaba el desacoplamiento entre cliente y servidor, facilitando la modularización del trabajo y quitando problemas de dependencias entre ambas partes, que podría hacer afectado al ritmo de desarrollo (p.e. si es el servidor el que se encarga de la redirección entre páginas, el back-end ya depende de que estén las páginas para saber a cuál redirigir y qué información dejarle, o por el contrario el cliente no puede probar realmente la navegación).

En cuanto a las operaciones, se podría decir que todas son síncronas, ya que siempre que el cliente lance una petición, este esperará la respuesta y actuará acorde a ella. Las únicas operaciones asíncronas serían algunas relacionadas con el registro de información de las acciones de los usuarios, de la cual el cliente no necesita saber nada, por lo que no esperará nada relacionado con ellas.

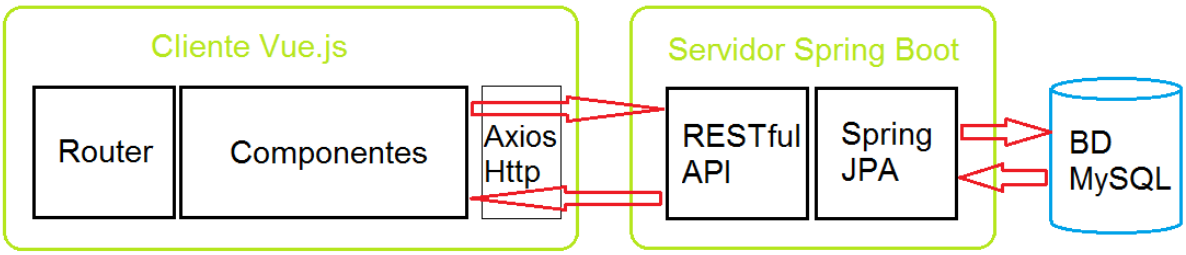
Tecnologías concretas:

– Java -Vue -XML

– Maven -Vue Router

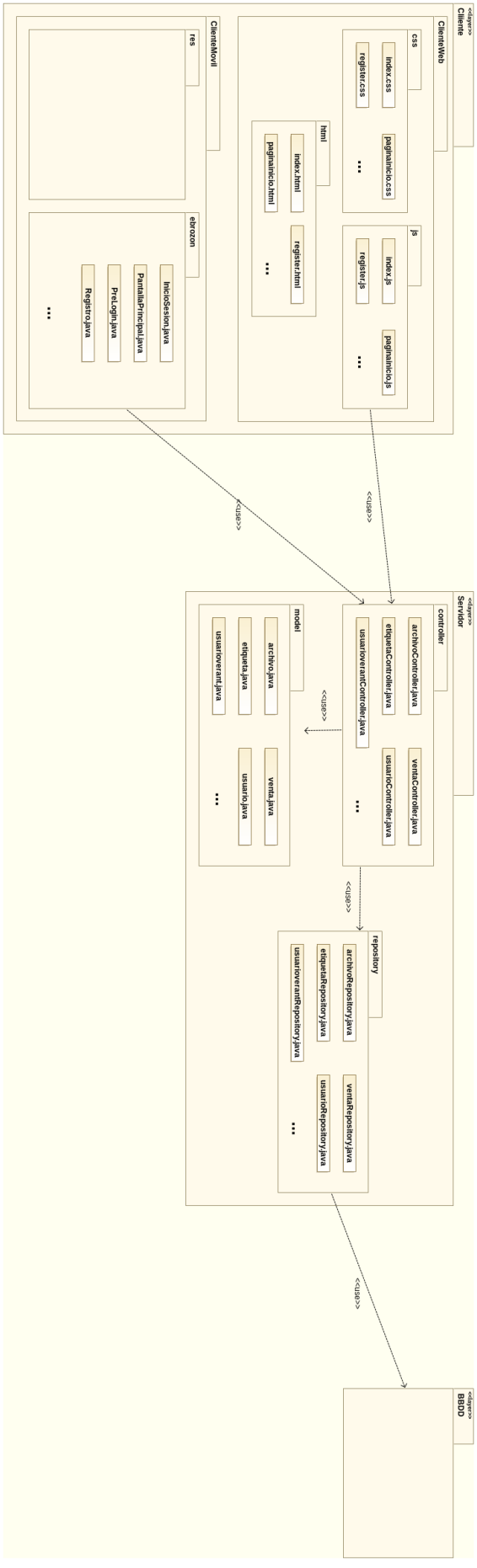
– Spring Boot -Axios

Esquema arquitectura web



En cuanto a la arquitectura, solo destacar que toda la lógica relacionada con la gestión de datos se realizará en el servidor, limitándose el cliente (tanto web como móvil) a gestionar la lógica de la GUI y la sesión, y a realizar las peticiones correspondientes al servidor. De esto se concluiría que la arquitectura que seguiremos será la de cliente ligero y servidor pesado, al delegar al servidor los “cálculos” más complejos y costosos.

Vista de módulos



Se propone una arquitectura 3-tier, la cual se describe a continuación.

- **Capa cliente**
  Capa más externa, contiene la capa de presentación. Se divide en dos bloques, el correspondiente a la web y a la aplicación Android.

  - **Cliente Web**
    Contiene los elementos que conforman la web de la aplicación, el módulo provee una interfaz gráfica al usuario, mientras que hace uso de la interfaz proporcionada por el servidor para llevar a cabo las operaciones correspondientes.

    - css
      Contiene los estilos de los elementos que hay en las vistas.

    - html
      Los elementos html indican que elementos hay en las vistas.

    - js
      Contiene la lógica de la web, para poder llevar a cabo las acciones requeridas por el usuario.

  - **Cliente Móvil**
    Contiene los elementos que conforman la aplicación Android, el módulo provee una interfaz gráfica al usuario, mientras que hace uso de la interfaz proporcionada por el servidor para llevar a cabo las operaciones correspondientes.

    - ebrozon
      Contiene la lógica necesaria para poder llevar a cabo las acciones requeridas por el usuario en las distintas pantallas.

    - res
      Contiene las pantallas que conforman la aplicación, proporcionan una interfaz gráfica al usuario.

- **Capa servidor**
  Capa intermedia, contiene la capa de la lógica de la aplicación o de negocio.

  - **Controller**
    Contiene la API con la declaración de peticiones que es capaz de responder, y a las cuales da respuesta gracias a las funciones de repository.

    - usuarioController.java
      Implementa la API con las peticiones necesarias sobre un usuario. Hace uso de las interfaces ofrecidas por usuario del módulo Model y usuarioRepository del módulo Repository. Ofrece la siguiente interfaz.

      - registrar

Operación utilizada para guardar los datos de un usuario en la base de datos.

- actualizarUsuario
  Operación utilizada para actualizar los datos de un usuario en la base de datos.

- logear
  Operación utilizada cuando el usuario hace login en la aplicación.

- recuperarUsuario
  Operación utilizada para recuperar la información de un usuario.

- banearUsuario
  Operación utilizada para banear un usuario de la aplicación.

- desbanearUsuario
  Operación contraria a la anterior.

- cambiarContrasenaRec
  Operación utilizada para recuperar la contraseña.

- **Model**
  Contiene las estructuras de datos que representan las tablas de la base de datos y sus restricciones.

  - usuario.java
    Implementa la tabla usuario. Ofrece como interfaz los getters y setters sobre sus atributos.

- **Repository**
  Contiene las funciones necesarias para la persistencia de los modelos en la base de datos.

  - usuarioRepository.java
    Ofrece las operaciones necesarias para la persistencia de un usuario en la base de datos.

    - existsBynombreusuario
      Operación utilizada para saber si existe en la base de datos un usuario con un nombre determinado.

    - existsBycorreo
      Operación utilizada para saber si existe en la base de datos un usuario con un correo determinado.

    - findBynombreusuario
      Operación utilizada para encontrar en la base de datos un usuario con un nombre determinado.

    - findBycorreo

Operación utilizada para encontrar en la base de datos un usuario con un correo determinado.

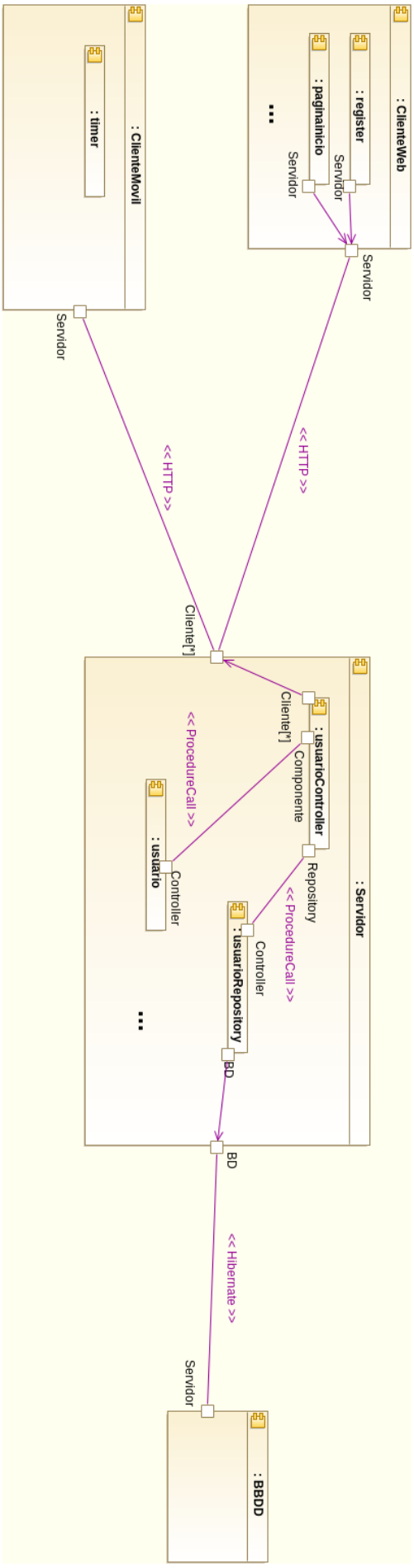
    - registrarLogin
      Operación utilizada par guardar el login de un usuario en la base de datos.

- **Capa base de datos**
  Capa más interna, contiene la capa de gestión de datos o recursos.

Los submódulos que se muestran en la vista, así como la descripción de alguno de ellos y de su interfaz, se han seleccionado a modo representativo. Según crece la aplicación aparecerían nuevos submódulos acordes a la lógica de la aplicación desarrollada.

Las operaciones mencionadas se han descrito a un nivel de diseño, simplemente se describe cual es su uso, en la siguiente vista se documentan sus parámetros y en caso de las operaciones ofrecidas por el servidor, como se deben llamar desde un cliente.

Vista de componente y conector



Componentes

- **ClienteWeb**
  Componente que se encarga de la aplicación web, atiende las peticiones del usuario. Este componente cuenta con una serie de componentes internos que se encargan de tareas específicas.

  - **paginainicio**
    Entre las funciones de este componente se encuentra la presentación de la página de inicio, así como gestionar las acciones que realice el usuario sobre dicha página.

  - **register**
    Entre las funciones de este componente se encuentra la presentación de la página de registro, así como gestionar las acciones asociadas al registro que el usuario tiene disponibles.

  Estos componentes requieren una interfaz para la comunicación con el servidor.

- **ClienteMovil**
  Componente que se encarga de la aplicación Android, atiende las peticiones del usuario. Este componente cuenta con una serie de componentes internos que se encargan de tareas específicas.

  - **timer**
    Subcomponente encargado de la gestión de aspectos que requieran una temporización en la aplicación.

- **Servidor**
  Componente que se encarga del servidor del sistema, atiende la peticiones que llegan desde los clientes y proporciona las respuestas necesarias. Este componente cuenta con una serie de componentes internos que se encargan de tareas específicas.

  - **usuarioController**
    Entre las funciones de este componente se encuentran gestionar las peticiones de los clientes. Ofrece una interfaz que estos pueden utilizar, requiere una interfaz para gestionar los modelos de datos y otra para poder realizar peticiones sobre la base de datos.

  - **usuario**
    Entre las funciones de este componente se encuentra gestionar los modelos de datos. Ofrece una interfaz al componente Controller.

  - **usuarioRepository**
    Entre las funciones de este componente se encuentra gestionar las peticiones sobre la base de datos. Ofrece una interfaz al componente Controller, requiere otra para realizar operaciones sobre la base de datos.

- **BBDD**
  Componente que se corresponde con el motor de base de datos SQL transaccional

encargado del almacenamiento de datos, permitiendo su creación, modificación, borrado y consulta. Ofrece una interfaz utilizada por el servidor.

Conectores

En la vista se pueden observar tres tipos de conectores: conectores ProcedureCall entre subcomponentes de componente servidor, DBAccess para el acceso a la base de datos por parte del componente servidor y RequestReply entre los componentes cliente y el servidor para la gestión de peticiones por parte del usuario.

- **ProcedureCall**
  Los conectores ProcedureCall modela el flujo de control entre los componentes mediante invocaciones. Realizan además la transferencia de datos de datos entre los componentes que interactúan mediante el uso de parámetros y valores de retorno.

  Por ello, son conectores de comunicación y de coordinación. En este caso, todos los conectores son llamadas a métodos Java, que proveen la transferencia de sus parámetros por referencia, un valor de retorno, su accesibilidad puede ser pública o privada, entre otras características.

  Todos los conectores que permiten la comunicación entre subcomponentes de Servidor son de este tipo.

- **Hibernate**
  El conector Hibernate, del tipo DBAccess (acceso a base de datos), permite el acceso a los datos almacenados en otro componente almacén de datos. Provee servicios de comunicación. En la vista se presenta un conector que permite la comunicación entre la base de datos y el servidor.

  El servidor envía peticiones y consultas (queries) a la base de datos gracias a la interfaz proporcionada por esta y la base de datos devuelve la información correspondiente a cada petición.

- **HTTP**
  El conector HTTP, del tipo RequestReply (petición-respuesta), es un conector de tipo evento, el flujo se desencadena por la ocurrencia de un evento, esto permite que exista comunicación asíncrona entre los distintos clientes y el servidor, utlizando el protocolo con el mismo nombre.

  Todos los conectores que permiten la comunicación entre el cliente web, el cliente Android y el servidor son de este tipo.

Interfaces

De entre las interfaces proporcionadas por los componentes anteriormente mencionados destaca las proporcionadas por los componentes usuarioController, usuario y usuarioRepository.

- **usuarioController**
  Todas las acciones que llegan desde los clientes referentes a la gestión de usuarios se realizan mediante llamadas a las funciones ofrecidas por Controller

  - String registrar(@RequestParam("un") String un, @RequestParam("cor") String cor, @RequestParam("pass") String pass, @RequestParam(value = "tel", required=false) Integer tel, @RequestParam("na") String na, @RequestParam("lna") String lna, @RequestParam(value = "cp", required=false) Integer cp, @RequestParam(value = "ci", required=false) String ci, @RequestParam(value = "pr", required=false) String pr, @RequestParam(value = "lat", required=false) String lat, @RequestParam(value = "lon", required=false) String lon, @RequestParam(value = "im", required=false) MultipartFile im)

    /registrar?un=param&cor=param&pass=param&tel=param&na=param&lna=param&cp=param&ci=param&pr=param&lat=param&lon=param&im=param

  - String actualizarUsuario(@RequestParam("un") String un, @RequestParam("tel") Integer tel, @RequestParam("name") String na, @RequestParam("lna") String lna, @RequestParam("cp") Integer cp, @RequestParam("ci") String ci, @RequestParam("pr") String pr, @RequestParam("lat") String lat, @RequestParam("lon") String lon, @RequestParam("im") MultipartFile im)

    /actualizarUsuario?un=param&tel=param&na=param&lna=param&cp=param&ci=param&pr=param&lat=param&lon=param&im=param

  - String logear(@RequestParam("un") String un, @RequestParam("pass") String pass)

    /logear?un=param&pass=param

  - usuario recuperarUsuario(@RequestParam("un") String un)

    recuperarUsuario?un=param

  - String banearUsuario(@RequestParam("un") String un)

    /banearUsuario?un=param

  - String desbanearUsuario(@RequestParam("un") String un)

    desbanearUsuario?un=param

  - guardar(usuario u)

    /guardar?u=param

  - String cambiarContrasenaRec(@RequestParam("un") String un)

    /recuperarContrasena?un=param

- **usuario**
  La gestión de los datos por parte del componente usuarioController se realiza mediante llamadas a las funciones de usuario.

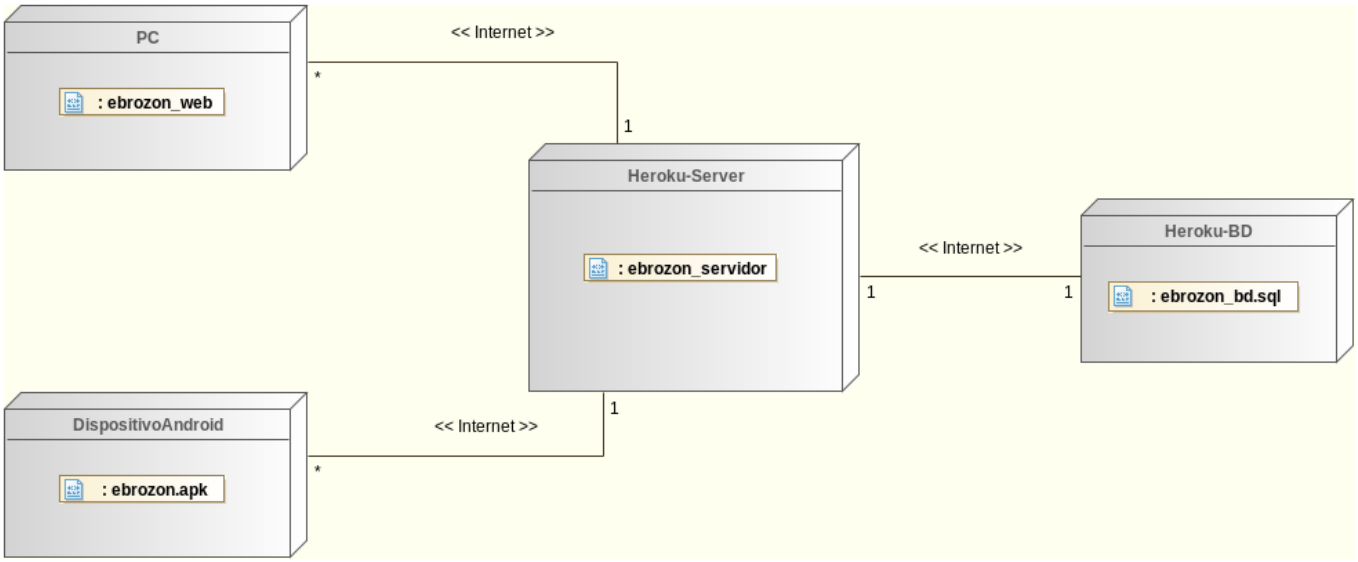
  - **usuario**(String nombreusuario, String correo, String contrasena, String nombre, String apellidos)
  - **usuario**(String nombreusuario, String correo, String contrasena, int telefono, String nombre, String apellidos, int codigopostal, String ciudad, String provincia)
  - **usuario**(String nombreusuario, String correo, String contrasena, int telefono, String nombre, String apellidos, int codigopostal, String ciudad, String provincia,float latitud, float longitud, int archivo)
  - String getNombreusuario()
  - setNombreusuario(String nombreusuario)
  - String getCorreo()
  - setCorreo(String correo)
  - String getContrasena()
  - setContrasena(String contrasena)
  - getTelefono()
  - setTelefono(int telefono)
  - String getNombre()
  - setNombre(String nombre)
  - String getApellidos()
  - setApellidos(String apellidos)
  - getCodigopostal()
  - setCodigopostal(int codigopostal)
  - String getCiudad()
  - setCiudad(String ciudad)
  - String getProvincia()
  - setProvincia(String provincia)
  - float getLatitud()
  - setLatitud(float latitud)
  - float getLongitud()
  - setLongitud(float longitud)
  - int getArchivo()
  - void setArchivo(int archivo)
  - int getActivo()
  - setActivo(int activo)
  - setUrlArchivo(String url)
  - String getUrlArchivo()
- **usuarioRepository**
  Todas las operaciones de base de datos se manejan dentro de la lógica de gestión de usuarios se realizan mediante llamadas a funciones contenidas en usuarioRepository.

  - boolean existsBynombreusuario(String nombreusuario)
  - Optional<usuario> findBynombreusuario(String nombreusuario)
  - boolean existsBycorreo(String correo)
  - Optional<usuario> findBycorreo(String correo)
  - registrarLogin(String un)

Los subcomponentes mencionados dentro de los componentes de cada tipo de cliente son representativos, no representan la totalidad de la aplicación, según crece la misma, aparecen nuevos subcomponentes respectivos a las funcionalidades añadidas.
Respecto al componente que representa al servidor, se menciona únicamente lo referente a la lógica de gestión de los usuarios, para el resto de funcionalidades sería de la misma forma.
Los conectores serían los mismos para el resto de funcionalidades que no se ven reflejadas en esta vista.

La documentación completa de la API del servidor se encuentra en: http://protected-caverns-60859.herokuapp.com/swagger-ui.html#/

Vista de despliegue



5. MEMORIA DEL PROYECTO

5.1. INICIO DEL PROYECTO

Con respecto al tema del despliegue, se hizo al final un despliegue en local en lugar de utilizar los servicios de Google Cloud. Se sigue dejando libertad al cliente de desplegarlo en el servicio que prefiera. Además, el servidor de la base de datos, se desplegó en Heroku. Este servicio se deja abierto para que realicen las pruebas necesarias, pero se deja el despliegue final al cliente ya que Heroku no es apto para utilizarlo en una aplicación con un potencial de conexiones elevado. La aplicación web funciona en los navegadores Chrome y Edge, eliminando el soporte en Firefox y Opera.

Todos los miembros del equipo realizaron las distintas instalaciones de las herramientas a utilizar. Además, todos realizaron diversos tutoriales, ya fuera para instalar correctamente las herramientas, o, sobre todo, para familiarizarse con las tecnologías a emplear y facilitar las tareas. Estos tutoriales también se utilizaron para crear una estructura del proyecto con la que se pudiera trabajar fácilmente.

5.2. EJECUCIÓN Y CONTROL DEL PROYECTO

Debido a cambios que se produjeron en mitad del proyecto, como la adición de crear una aplicación nativa en Android, el reparto de tareas se modificó.
En primera instancia el equipo se dividió en dos grupos, uno centrado en la creación del back-end y otro en la parte web de la aplicación en el grupo de front-end. Se asignó un líder de cada grupo que coordinaría éstos, y entre otras cosas, repartiría las tareas a realizar entre los diferentes miembros. Posteriormente, ya que se decidió hacer una aplicación en Android, se creó un grupo más, haciendo que fueran tres grupos de tres personas cada uno. Como es evidente en este nuevo grupo creado se asignó un líder con las mismas funciones que los anteriores.
La comunicación interna se ha realizado mediante dos vías, grupos en la aplicación de WhatsApp y servidores en Discord.
Se crearon varios grupos en WhatsApp, uno para cada equipo (Back-End, Front-End y posteriormente Android) y uno general.
En los propios de cada equipo se comunicaban cosas relacionadas con esa parte y que no fuera de importancia para el resto de los equipos. Utilizados más para el reparto de tareas, y consultas acerca de cómo realizar ciertas partes.
El general, como es evidente, tenía un propósito de comunicación entre miembros de todos los grupos, especialmente para temas más propios del grupo en conjunto, como reuniones, y semejantes. También se utilizó como puente de unión entre los grupos para decidir temas de utilización de interfaces, peticiones de que parte hacer primero para hacer pruebas, etc.
En Discord se siguió una estructura muy semejante a la de WhatsApp, creando varios servidores de texto y voz para cada uno de los distintos grupos, así como uno de voz y texto de propósito general.
Al igual que los grupos de WhatsApp, cada grupo tenía su propio servidor, utilizado para reuniones de éstos. Los servidores de texto se utilizaron para el envío de ficheros de cierta importancia o ficheros que quizás tuvieran que pasar una cierta revisión, pero que no fueran imprescindibles para la implementación de la aplicación, como memorias o ficheros semejantes.
El general se utilizó entre otras cosas para los repartos de tareas al inicio del proyecto, así como la división en grupos y reparto general de tareas, y más adelante como comunicación entre los distintos grupos en cualquier momento.

El progreso del proyecto, así como número de horas realizadas se ha ido realizando a través de Git. Se creó un repositorio de documentación, con entre otras cosas, ficheros para controlar el

número de horas y las tareas asignadas a cada persona. En el fichero de horas, cada persona puede actualizar en cualquier momento su contribución con el formato de nombre, horas, día y en qué estuvo trabajando. Al actualizar las horas de trabajo puedes hacer todas las inclusiones que quieras, es decir, puedes hacer una única actualización del fichero con varias entradas al final de una semana en caso de no querer estar todos los días que hagas algo actualizando. En el fichero de tareas se presenta una tabla con el nombre y la tarea asignada a esa persona. De esta manera se puede saber fácilmente que parte tienes que realizar en caso de que no haya quedado claro, además de que también puede ser útil ver las tareas de los demás en caso de que el trabajo de una persona pueda tener problemas con el de otras (uso de interfaces, etc.) y poder saber con exactitud quien está realizando eso y comunicarse con esa persona.

El trabajo pendiente y el trabajo realizado estaba dividido en función del grupo. Cada parte se encargaba de sus propias tareas, haciendo que la gestión fuera diferente.
En el caso del back-end, aunque sea desaconsejable, se utilizó una gestión heroica. Debido a como era el desarrollo del back-end este tipo de gestión era más que suficiente para repartir las tareas. El líder se encargaba de repartir las distintas tareas que quedaban por realizar a los distintos miembros. No hubo registros para saber lo realizado y lo que faltaba por realizar.
En el desarrollo de la aplicación Android, el grupo responsable añadía a Git las tareas que quedaban por realizar como issues y las iban asignando a los respectivos integrantes. Cuando alguien terminaba su tarea se le asignaba la siguiente. No hay registros documentados que indiquen lo que se ha realizado.
Con respecto al conjunto general, no había una manera de saber todo lo que se había hecho en el proyecto, por ejemplo, con el uso de los requisitos. Con estemétodo se podría saber lo que se ha realizado y lo que falta, sin embargo, nunca se utilizó ningún método de este estilo.

Cada equipo encargado de una de las tres partes tenía un calendario propio, que tenían que seguir lo máximo posible siempre y cuando se pudiera.
De los tres grupos solo uno acabó dentro de lo establecido por el calendario, el back-end. Por lo tanto, no hubo que realizar ningún tipo de ajuste ni semejantes. Durante los últimos días este grupo ha estado depurando pequeños problemas que podían ir apareciendo.
Los otros dos grupos no avanzaban con respecto al calendario, pero debido a que cada grupo era independiente, afrontaron la compensación del calendario de distintas maneras.
La aplicación Android sufrió un retraso con respecto al calendario alrededor de 1 ó 2 semanas. Pero se recuperó ese tiempo durante las últimas 2 semanas. Es decir, frente a ese retraso, surgido por problemas de entregas de otros proyectos, etc. el grupo decidió simplemente hacer jornadas intensivas de desarrollo para alcanzar al calendario.
La página web lleva un retraso respecto al calendario de alrededor de un mes. Los problemas principales por los que ha surgido este retraso son coincidencias de asignaturas, junto con entregas de trabajos, al igual que la aplicación Android. Sin embargo, el motivo principal de que este retraso haya sido mucho más elevado ha sido que no ha habido un ritmo constante en su desarrollo. Es decir, 2 de los integrantes de este grupo no empezaron a trabajar en la parte web hasta hace un par de semanas, lo que hace que tan solo una persona se estaba encargando del apartado web. Esto hace, como es evidente, que el calendario fuera imposible de cumplir. Una única persona no podía seguir un calendario ajustado para tres. Desgraciadamente, debido a las pocas personas que han trabajado en la web y que se han unido especialmente tarde, no se ha podido llegar a ninguna solución para solucionar este problema, dejando un producto incompleto. Como medidas para solucionar esto, se realizó una gran insistencia en las distintas vías de comunicación que tenían para que realizaran la parte de las tareas que les correspondía, pero sin demasiados resultados. Finalmente, esto ha dado que se pusieran a trabajar estos últimos días, pero ya con el calendario excesivamente roto.

Para el desarrollo del proyecto se han utilizado varias tecnologías y herramientas ya nombradas en secciones anteriores. No ha habido ningún cambio de herramientas frente a las que se habían nombrado. Tampoco surgió ningún problema destacable con el uso de estas tecnologías. Se tuvieron que añadir tecnologías ya que en durante el principio del desarrollo no estaba pensada la versión de Android, por lo que se tuvo que añadir sus respectivas tecnologías posteriormente. Como es evidente en un grupo tan grande, no todo el mundo tenía experiencia en todos los campos. Fundamentalmente la separación en grupos se hizo en función de los campos que más experiencia se tenía, aunque también se hizo en función de las preferencias de cada uno.

A la hora de integrar el código de los distintos grupos en general no hubo demasiados problemas. Los pocos problemas surgidos fueron con el envío de las peticiones en la web. A la hora de integrar el código en esta parte hubo problemas ya que el servidor no aceptaba sus peticiones. Afortunadamente el problema se solucionó rápidamente añadiendo una etiqueta.
Para un mejor funcionamiento se añadió una etiqueta para que las respuestas se pudieran interpretar correctamente en Android y Java.
Otro problema que surgió al integrar el código de web y Android fue producido por las imágenes. Para que las imágenes web funcionasen en Android hubo que realizar ciertos cambios.
No hubo problemas destacables a la hora de realizar el despliegue de la aplicación, tanto en web como en Android.

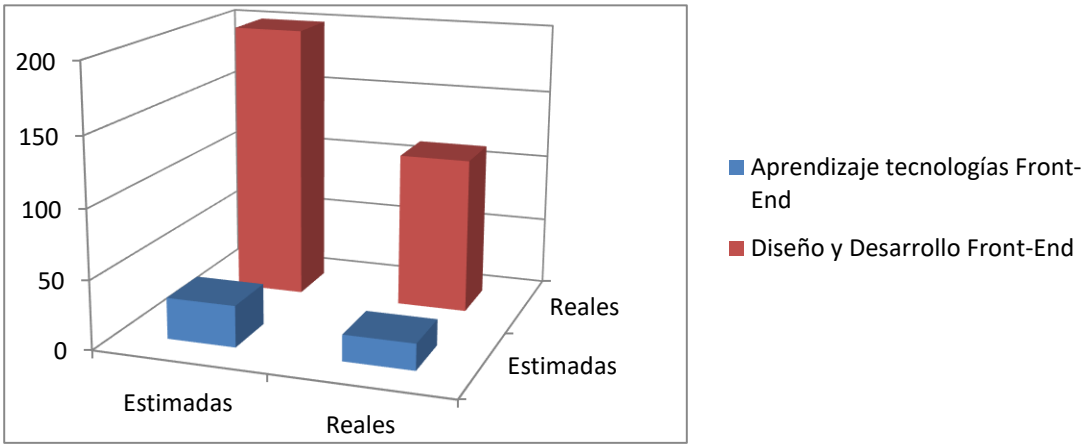
En una ocasión se perdió parte del código de la aplicación Android por un fallo humano. Para solucionarlo, como es evidente, se utilizó una de las funcionalidades de Git, que es volver a versiones anteriores de un repositorio. Esto hizo que se pudiera recuperar la funcionalidad que se había perdido al eliminar el código.

Al realizar pruebas más adelante en el proyecto, se descubrieron errores de funcionamiento que no se habían visto durante las pruebas iníciales.

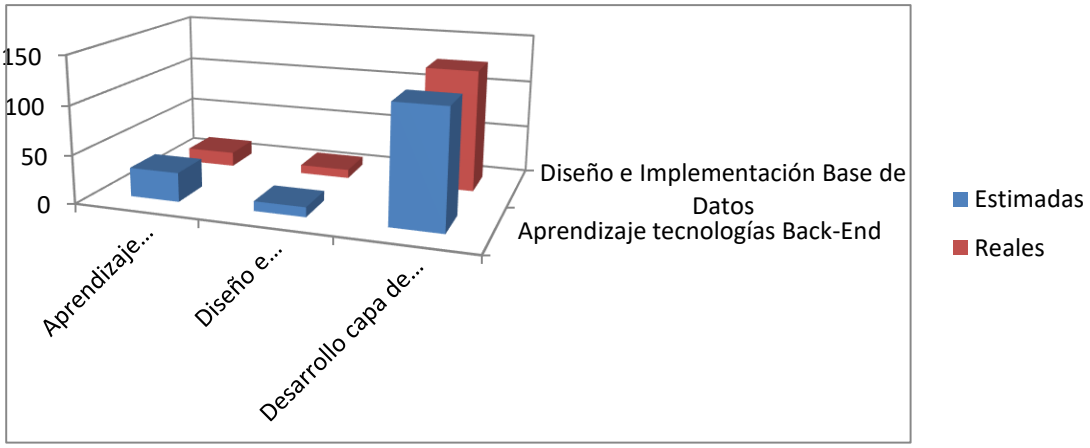
Como ya se ha dicho anteriormente, no se ha cumplido gran parte del calendario, lo cual ha hecho que no se hayan podido realizar todas las pruebas que se hubieran esperado o pruebas de comportamiento general con todas las funcionalidades disponibles.

5.3. CIERRE DEL PROYECTO

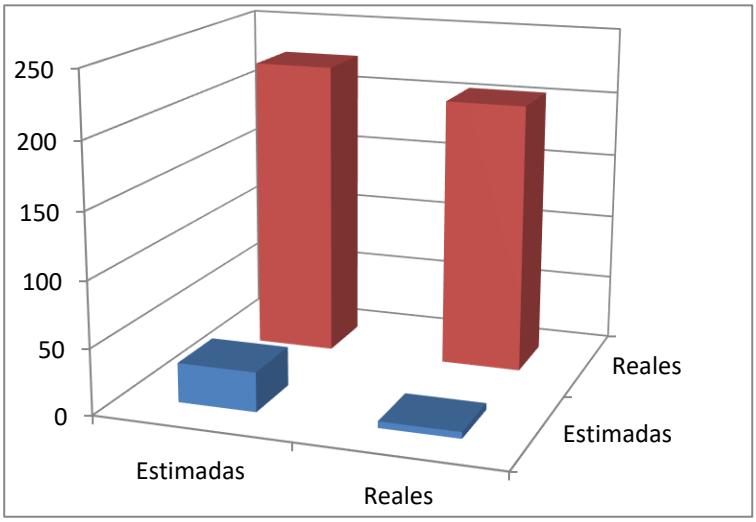
Debido a ciertos cambios surgidos durante el desarrollo del proyecto, los resultados finales respecto a esfuerzo en cada uno de los ámbitos han cambiado, especialmente en el apartado web y en la aplicación Android. Debido a que en un principio solo se iba a realizar un port de la web a Android, la estimación inicial se había presupuesto baja, pero al realizar una aplicación nativa, los costes se han incrementado considerablemente.
Con respecto al desarrollo del front-end, se había hecho una estimación de unas 230 horas repartidas entre aprendizaje y el propio desarrollo. La división era de 30 horas de aprendizaje y 200 para diseño y desarrollo. Los resultados finales fueron de 19 horas para aprendizaje de nuevas tecnologías entre todos los integrantes de ese grupo y 113 horas de diseño y desarrollo, contando la creación de una versión prototipo. Es decir, se dedicó un total de 132 horas al front-end.



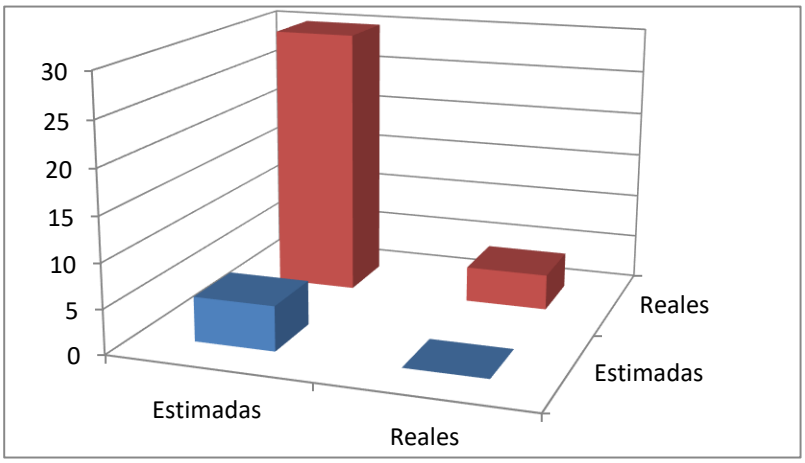
En el back-end se había estimado un total de 30 horas para familiarización de nuevas tecnologías, 10 horas para diseño e implementación de la base de datos y 120 horas para el desarrollo de la capa de interacción con la base de datos, sumando un total de 160. Finalmente se dedicaron 15 horas a aprendizaje de tecnologías, 9 horas para diseño e implementación de la base de datos y 126 horas para el desarrollo de la capa de interacción. Por lo que, con estos datos, se obtiene que se dedicaron 150 horas al back-end.



Las estimaciones iníciales para la aplicación Android fueron bajas con respecto a las estimaciones de las otras partes debido a que se iba a hacer un port, estimándose unas 25 horas. Cuando se decidió crear la aplicación nativa se volvieron a estimar los costes. Las nuevas estimaciones fueron 30 horas de familiarización con el entorno AndroidStudio, 100 horas para el diseño de la interfaz, 120 horas para el desarrollo de la interfaz, haciendo un total de 250 horas. Los resultados obtenidos fueron de 5 horas de familiarización y 205 horas para el desarrollo de la aplicación.



Para las pruebas se hizo una estimación de 5 horas de realización de pruebas de carga y 30 horas de pruebas funcionales, sumando 35 horas totales de pruebas. Finalmente se realizaron 4 horas de pruebas.



A continuación se presenta en forma de tabla las horas estimadas y reales en cada uno de los ámbitos descritos:

	Estimadas	Reales
Aprendizaje tecnologías Front-End	30	19
Diseño y Desarrollo Front-End	200	113
Aprendizaje tecnologías Back-End	30	15
Diseño e Implementación Base de Datos	10	9
Desarrollo capa de interacción con BD	120	126
Aprendizaje tecnologías Android	30	5
Diseño y desarrollo de la interfaz Android	225	205
Pruebas de carga	5	0
Pruebas funcionales	30	4

Como se ha podido ver, la cantidad de horas dedicadas a cada parte ha muy semejante a las estimadas, excepto en la parte web, debido fundamentalmente al retraso en el calendario que tenían, lo que hizo que no se pudiera trabajar tanto en ese aspecto. También puede ser debido a una ligera sobreestimación del coste de horas.
Otro punto en el que hay diferencia de horas generalizadas en cada equipo es en el aprendizaje de nuevas tecnologías. Esto es debido a que se estimó que los integrantes tenían un conocimiento muy básico de las herramientas a utilizar y de esa manera se tendría una estimación máxima. Como muchos de los integrantes tenían algún conocimiento previo con las herramientas que utilizaban este coste se redujo.
Con respecto a la diferencia en esfuerzo al front-end ha sido producida por lo recién nombrado y al hecho de que, al no cumplirse el calendario y llevando un gran retraso, como se mencionó en apartados previos, el número de horas es bastante inferior al supuesto ya que no se dedicó tanto trabajo.
En back-end, las estimaciones propuestas fueron bastante realistas, llegando a ser ligeramente superior el coste para la implementación mayor que el estimado, aunque el coste en familiarización con las distintas tecnologías fue reducido.
Las estimaciones para la aplicación Android sufrieron grandes cambios. Como ya se ha dicho se hizo una estimación muy alta para el diseño de la interfaz, mientras que el coste de su implementación iba a ser muy bajo. Pero debido a que se hizo una aplicación nativa el coste de su producción se incrementó. La diferencia entre el coste estimado y el real es muy bajo, por lo que la estimación que se hizo fue bastante ajustada a la realidad.
También se hizo una estimación de las pruebas que no se pudo cumplir del todo, ya que debido a los retrasos producidos en la aparte web y ligeramente en Android, hizo que no se puedan realizar todas las pruebas esperadas. Con lo cual la cantidad de esfuerzo dedicado a las pruebas es menor.

Como ya se ha dicho anteriormente, había gente que no controlaba demasiado las tecnologías utilizadas, así que el hecho de participar en este proyecto ha hecho que todos los integrantes puedan aprender algo más acerca de distintas tecnologías.
En primer lugar, Git. Muy pocos de los integrantes habían utilizado esta clase de herramienta, lo que ha permitido que este proyecto haya sido útil para aprender esta herramienta ya que, de no haberla utilizado, hubiera sido muy complicado la división del trabajo y la unión de todo lo realizado por cada integrante. El hecho de haber aprendido a utilizar esta herramienta es útil ya que, en el futuro, al intentar hacer cualquier proyecto de tamaño medio-grande sería necesaria esta herramienta, por lo que esa experiencia ya se ha obtenido.
Ciertas personas no habían aprendido ciertas tecnologías del back-end o front-end, pero por distintos motivos han participado en ese equipo, lo que ha hecho que hayan tenido que aprender las tecnologías correspondientes desde prácticamente el principio. Esto hace que si en algún momento tienen que participar en algún proyecto en el que necesiten eso ya tendrán un conocimiento lo suficientemente alto de esa tecnología como para desenvolverse relativamente bien.
Puesto que al final se decidió hacer la versión para Android nativa, la gente que no hubiera estado en contacto con esta tecnología también pudo aprender a utilizar Android Studio para la creación de aplicaciones Android.

Las horas de esfuerzo dedicadas por cada integrante y las tareas realizadas son las siguientes:
Jorge ha trabajado 69 horas y 30 minutos trabajando en la realización de la propuesta técnica y económica, aprendiendo tecnologías para el front-end y creación de la página web.
Saúl ha trabajado un total de 100 horas y 45 minutos repartidas entre reuniones, creación de la propuesta técnica y económica, creación y rediseño de la base de datos, familiarización con las tecnologías a usar en back-end, desarrollo de back-end, correcciones en la memoria, despliegues, realización de pruebas y corrección de bugs.

Israel ha trabajado 65 horas y 15 minutos (además las últimas 2 semanas es posible que no estén contabilizadas en la anterior cantidad y ha invertido en ellas aproximadamente 20h más), las cuales han sido repartidas entre reuniones, familiarización con tecnología empleada para la aplicación web, e implementación de la web.
Andrés ha trabajado un total de 79 y 15 minutos horas repartidas entre reuniones, creación de la propuesta técnica y económica, aprendizaje de tecnologías para Android y desarrollo de la aplicación Android además de correcciones de bugs.
Félix ha trabajado un total de 39 horas y 45 minutos repartidas entre reuniones, creación de la propuesta técnica y económica, familiarización con tecnologías para front-end y desarrollo de la parte web.
Víctor ha trabajado un total de 85 horas repartidas entre reuniones, realización de la propuesta técnica y económica, familiarización con el entorno para la aplicación Android, desarrollo de la aplicación Android y corrección de fallos.
Eduardo ha trabajado 54 horas repartidas entre reuniones, realización de la propuesta técnica y económica, familiarización con tecnologías empleadas en back-end y el desarrollo de éste, y realización de la documentación.
Alejandro ha trabajado un total de 31 horas y 30 minutos repartidas entre reuniones, realización de la propuesta técnica y económica y la memoria final, familiarización con las tecnologías utilizadas para back-end y su desarrollo.
Sergio ha trabajado un total de 72 horas repartidas entre reuniones, realización de la propuesta técnica y económica y memoria final y desarrollo de la parte web.

6. CONCLUSIONES

Conclusión general

Teniendo en cuenta que no se ha seguido una metodología de gestión basada en procesos, la manera de afrontar el proyecto sería diferente. El cambio principal sería realizar una documentación inicial de todas las tareas a realizar por cada equipo para tener claro cuánto queda por hacer.

Uno de los cambios fundamentales sería el hecho de que todos los miembros conocerían los procesos a realizar a la hora del desarrollo. Debido a que las tareas que faltaban por hacer no estaban documentadas en ningún lugar en dos de los grupos no tenían ninguna documentación de las tareas que faltaban por realizar, lo que hacía que cada vez que se acabara una habría que preguntar al líder acerca de la siguiente tarea que tendría que realizar en lugar de escoger una de las pendientes.

Lo que sí que se hizo de manera semejante fue el hecho de que los líderes proveían las distintas herramientas y tutoriales para la familiarización de las tecnologías a emplear, aparte de la supervisión del trabajo realizado, al revisar el código creado por cada integrante antes de darlo por válido.

Saúl Alarcón Cano

De este proyecto, y sobretodo del resultado obtenido (producto resultante), se puede concluir que un buen producto empieza por una buena gestión. Esta conclusion se ve clara al darnos cuenta de que nuestra mediocre gestión ha dado lugar a un producot acorde, mediocre. Si bien podría haber sido peor tanto la gestión como el producto resultante, también habría sido muy mejorable.

Parte del problema de la gestión proviene de la falta de comunicación adecuada y una falta de trabajo en equipo, pero también por una falta de responsabilidad por parte de algunos integrantes. Lo cual me ha hecho ver uno de los mayores problemas que encontraremos en el “mundo real” al entrar en un proyecto, el hecho de que no todos trabajan ni con las mismas ganas, ni al mismo ritmo, y esto hace que si se quieren alcanzar los plazos, alguien deba compensar la falta de trabajo de otros.

Centrándonos en la gestión como tal, si tuviésemos que iniciar un nuevo proyecto, aplicando los conocimientos adquiridos de esta experiencia, lo primero que haria sería aclarar desde un comienzo los papeles de cada trabajador y establecer una jerarquía clara (lo cual si hicimos), y establecería unos procesos de control y medidas de avance del Proyecto lo más concretos posibles (lo cual no hicimos bien), para intentar seguir el calendario lo más possible. En general, aplicaría una metodología de procesos mucho más metódica, ya que la que hemos llevado acabo, dejaba mucho que desear, al dejarlo todo bastante abierto e indefinido.

Y diría que uno de los mayores problemas, ha sido la indulgencia generalizada, en la que pese a ir retrasados, se iba perdonando y diciendo el “ya lo recuperarás”. En un ambiente laboral, esto

sería sin duda lo primero que se solventaría al jugarse el sueldo, ya que aquí al no haber dicho riesgo, nadie sentía ese nivel de responsabilidad.

Victor Sisqués Cortés

Tras cursar esta asignatura me he dado cuenta de la importancia de la gestión de un proyecto y la organización de este. Aunque me parezca tedioso, llevar una documentación actualizada al día y con todos los calendarios establecidos ayuda enormemente a la hora de obtener información sobre el estado del proyecto.

En cuanto a implementación del producto final; considero que, en la app, hemos hecho un buen trabajo que cumple la gran mayoria de los requisitos establecidos a lo largo de la asignatura. El uso de GitHub ha ayudado bastante a la hora de juntar las distintas versiones de la app implementadas por cada uno de los miembros de este grupo.

A pesar de todo esto, considero que se podria haber mejorado el reparto de trabajo y la comunicación entre los integrantes del grupo. En resumen, me ha gustado pero no lo volveria a repetir.

Sergio Álvarez Peiro

Mi conclusión de este proyecto es la importancia que tiene una buena gestión. Tanto al principio del proyecto como saber como adaptarse y cambiar tras calendarios no cumplidos o falta de comunicación. Ya que este proyecto ha sido algo que en el mundo laboral se escalaría mas, con mas componentes en cuanto a la implementación y más personas involucradas pienso que algo así no se podría realizar con la gestión llevada acabo. Esto se mejoraría teniendo una mejor comunicación entre los equipos encargados de cada parte del proyecto, para que nada se quede atrás y en caso de que se haga reajustar calendarios establecidos de forma más habitual. También, el reajustar los equipos si se ve una carga de trabajo que no está equilibrada en ellos o responsabilidades de gestión mejor repartidas.

En cuanto a la parte de la implementación del producto final, se podría haber usado otras tecnologías dentro del desarrollo del código para facilitar y acelerar la implementación pero se descartaron por no dificultar más la tarea de aprendizaje, pero al final podría haber resultado en algo más eficiente. También se aprovecharon herramientas para el control de versiones y fijar objetivos que han facilitado el trabajo individual pero se podrían haber aprovechado más para la comunicación entre el equipo.

Al final creo que se ha conseguido un producto decente a pesar de los fallos mencionados, con la mayoría de requisitos cumplidos.

Andrés Gavín Murillo

Resulta interesante realizar un proyecto de estas caracterísiticas debido a su aproximación con un futuro laboral, ya que en cualquier trabajo habrá que coordinarse en grupo. En este sentido ha sido de gran utilidad, y además en el uso de herramientas comerciales que puede utilizar cualquier empresa destinada al desarrollo software.

El principal problema que hemos tenido (dada la magnitud del proyecto) ha sido que cada uno llevábamos un ritmo de trabajo diferente (unos han trabajado más al principio, otros más al final), esto seguramente se haya debido a que cada uno llevaba unas asignaturas diferentes con más o menos carga de trabajo, resultando en el incumplimiento del calendario fijado desde el primer momento.

Para un próximo proyecto habría que asegurar el cumplimiento del calendario en todo momento, repercutiendo en la responsabilidad y objetivos de cada uno.

Respecto al grupo de Android, hemos gestionado y planificado todo el desarrollo de la aplicación mediante el conjunto de herramientas que ofrece GitHub (branches, control de versiones, sistema de issues), lo cual me ha parecido muy acertado y mantendría para futuros proyectos.

Eduardo Gimeno Soriano

Como conclusión a este proyecto extraería la importancia de ofrecer una buena documentación de la API del servidor, ya que para los encargados de desarrollar el front-end deben conocer de buena mano que se está ofreciendo desde el back-end para poder gestionar las acciones del usuario. También al cliente le interesa conocer que ofrece el servidor, por tanto, esta documentación, además, no deja de ser un producto más que entregar. También es importante establecer unos buenos canales de comunicación con los desarrolladores del front-end para poder recibir de forma clara que necesitan que no se encuentre ya implementado.

A la hora de iniciar un nuevo proyecto intentaría ajustar de mejor manera el número de personas necesarias para cada parte del sistema a desarrollar, el reparto no tendría porque ser equitativo, la programación de GUI puede resultar más costosa que la programación de una API rest. También intentaría mejorar los primeros pasos en la gestión del proyecto, como por ejemplo, definir desde un primer momento como automatizar las pruebas para poder realizar una integración continua más eficiente. Obviamente mantendría el uso de GitHub para el control de versiones, para poder trabajar sobre distintas funcionalidades al mismo tiempo en distintas ramas, etc.

Israel Solanas Navarro

El proyecto me ha resultado de utilidad para adquirir una visión más global de lo que cuesta llevar a cabo un proyecto de software de tamaño considerable, en el que hay que coordinarse con un número elevado de personas para tomar decisiones de diseño, repartir tareas, etc. Lo que resulta importante de cara al mundo empresarial, principalmente si se trabaja en proyectos de un tamaño mediano o grande.

He aprendido a trabajar más en grupo, preguntando dudas, siguiendo algunas directrices de diseño, realizando tareas que se me han asignado, etc.

La necesidad de sacar las tareas adelante me ha ayudado a buscar tecnologías que no conocía (JavaScript, con sus apartados de JSON, AJAX y DOM) para ahorrar tiempo y esfuerzo, en definitiva, ser más eficiente en el desarrollo de estas.

¿Cambios respecto a los procesos seguidos durante este proyecto?

Somos 9 personas y nos dividimos equitativamente en tres grupos para el proyecto (Back-end, front-end web y front-end Android).
Ciertamente, en mi opinión para los primeros meses del proyecto (por ejemplo, todo la primera iteración) esto es efectivo porque puede desarrollarse paralelamente todos los apartados del proyecto y se evita que un apartado del proyecto se quede mucho más retrasado que el resto.
Sin embargo, hacía el final del proyecto, juntándose además otros trabajos que realizar, la coordinación se ha distanciado mucho, concretamente la web tuvo un retraso mayor que el resto por falta de dedicación y de conocimiento de las tecnologías en mi caso.
Esto lo evitaría estableciendo de una forma más cerrada los plazos para alcanzar objetivos y asignando tareas concretas semanalmente o cada dos semanas.

¿Qué cosas haría de otra forma?

No emplearía HTML puro para diseñar las pantallas estáticas porque ocupa mucho tiempo aprender algunos detalles para saber colocar adecuadamente el contenido, etc.
Y para solventarlo emplearía Bootstrap Studio u otra aplicación similar que facilite el diseño de las pantallas (por ejemplo, arrastrando los elementos sobre una pantalla,…).

Además, para la parte de la lógica de la web, intentaría aprender algún Framework de JS para facilitar el diseño y ser más eficiente en tiempo y número de líneas de código programando.

En vez de que todos los grupos se repartan equitativamente, dejaría una persona que se encargue solamente de coordinar e incluso formarse en lo básico de las tecnologías necesarias para apoyar a algún grupo que lo necesite en tareas puntuales.

¿Qué cosas seguiría haciendo más o menos igual?

Seguiría usando GitHub como repositorio compartido y la estructura de los grupos de front-end probablemente no la modificaría mucho.

Jorge Fernández Muñoz

Como conclusión personal, se trata del primer gran proyecto de desarrollo completo al que nos enfrentamos, el cual nos hace comenzar a conocer mas en profundidad la metodología de trabajo llevada a cabo en cualquier proyecto de desarrollo de software, con todas las ventajas que esto conlleva con respecto a nuestro futuro laboral.

Además, se ha hecho uso de diversas tecnologías con las cuales el grupo no estaba familiarizado, lo que permite dar un paso mas en nuestra formación, pero ademas se trata de un trabajo en el cual dada la magnitud de los grupos, y la situacion de tener un tiempo limite muy cercano, ayuda a mejorar la planificación para enfrentarse al mismo o, como en mi caso, darse cuenta de los errores de planificación cometidos para poder remediarlos en proximos proyectos, al igual que ayuda a la hora de realizar un proyecto de desarrollo en equipo, gestionando el reparto de trabajo, las diversas comunicaciones necesarias entre el grupo y con el resto de grupos, lo que hace de este proyecto un trabajo mas que interesante e importante por los resultados, tanto positivos como negativos, que genera.

Con respecto a la realización del proyecto, y hablando mas en concreto sobre el desarrollo de la web, en la cual se ha realizado el reparto de trabajo en función a las distintas pantallas de las que se componia la misma, me parece una división correcta de los procesos entre los diversos miembros del grupo, ya que al realizarse este reparto, se ahorra tiempo en cuanto a que cada miembro conoce sus pantallas y los elementos que la componen, lo que facilita su manipulación, por lo que me parece un método de reparto de trabajo muy correcto a pesar de que enc ciertos casos puede llevar a una desigualdad en cuanto al tiempo empleado, ya que no todas las pantallas cuestan el mismo esfuerzo, pero aun así, volvería a apostar por esta metodología.

El mayor problema de este proyecto, y por lo tanto aquello que mas se debería de cambiar, es la manera en que cada uno ha repartido sus esfuerzos sobre el proyecto, esto puede deverse a diversos factores, ya que depende de la carga de trabajo de cada uno de los miembros, pero aun así la realidad es que no se han cumplido los plazos inicialmente planificados, lo que tambien puede ayudar a los miembros a cuestionarse el porque no se han cumplido dichos plazos y que tendría que hacer cada uno para que se cumplieran correctamente.

Por tanto, podriamos afirmar que en opinion personal, el reparto de tareas era el correcto, sin embargo los diversos miembros no supieron repartir correctamente su tiempo, lo que provocó diversos retrasos y por ende diversos problemas con la version final del proyecto.

Félix García Rodríguez

Este proyecto puede servirnos como experiencia de cara al futuro, pues seguramente en nuestra vida laboral nos tocará trabajar en equipo y crear aplicaciones desde el principio sabiendo solo la visión de requisitos que quiere el cliente para ella. Esto nos ha hecho darnos cuenta de la importacia de algunos aspectos a los que antes no les dábamos gran importancia como la documentación, pues al realizar software que solo utilizábamos las personas que lo implementábamos no son surgían dudas sobre que hacía cada cosa, cosa que si ha surgido aqui.

Otro aspecto que hemos visto de gran importancia es la organización y comunicación enttre los distintos miembros, pues debido a las asignaturas cursadas y los asuntos personales de cada uno de los miembros del grupo, hemos trabajado en horarios diferentes, y no todos hemos ido avanzando de forma pareja. Esto, añadido a que las comunicaciones mayormente han sido a través de canales de texto como WhatsApp, han producido retrasos en el calendario, y que algunos miembros del grupo (yo el primero) hayan rendido menos de lo necesario.

Un tercer aspecto que nos ha faltado, y nos habria sido de gran ayuda, es definir desde el principio la automatización de las pruebas.

Por el contrario, hemos encontrado de gran utilidad el repositorio gitHub para trabajar en conjunto, pues podemos trabajar en paralelo, asignarnos tareas con el issue tracker etc.

Alejandro Cano Somalo

Este trabajo, puede servir como práctica en un futuro, debido a que el ambiente de trabajo visto aquí, muy probablemente se asemeje a la realidad laboral, ya que muy probablemente haya que trabajar casi continuamente en equipos de desarrollo.

Lo único malo a destacar es el hecho de la comunicación y coordinación. Debido a que ésta se ha realizado en medio como WhatsApp, puede llevar a problemas. La gestión que se ha hecho de Git, por parte del back-end tampoco fue la mejor debido a que todos trabajábamos en la misma rama, lo cual es desaconsejable, aunque no había problemas puesto que las partes de cada uno eran prácticamente independientes las unas de las otras.

Finalmente, debido a temas de horarios y demás que en la vida laboral no habría a tan alta escala, también afectó, ya que a la hora de preguntar alguna cosa o querer tener reuniones podía ser complicado debido a interferencias con otras asignaturas o trabajos.

ANEXO I. MEJORAS DEL PRODUCTO TRAS LA REUNIÓN 6

Se solucionó el problema de poder eliminar subastas las cuales ya habían recibido alguna puja. Esto no se hizo con ventas normales las cuales ya hubieran recibido ofertas, ya que, como el vendedor es libre de aceptar y rechazar las ofertas, perfectamente en caso de impedirle borrar ventas con ofertas, podría rechazarlas todas y entonces borrar, por tanto el que lo pueda hacer directamente no está mal.
Solución en el back end:

```
@ApiOperation(value = "Deactivate a sale, returns {O:Ok} if ok or error message if not ok", response = String.class)
@CrossOrigin
@RequestMapping("/desactivarVenta")
String desactivarVenta(@ApiParam(value = "sale's id", required = false) @RequestParam("id") int id) {
    Optional<venta> aux = repository.findByidentificador(id);
    if(aux.isPresent()) {
        if(aux.isPresent().getes_subasta() == 1 && repository_sub.numeroPujasRecibidas(id) >0) {
            return "{E:No se puede borrar una subasta cuando esta ha recibido ya una puja.}";
        }
        venta vent = aux.get();
        vent.setActiva(0);
        repository.save(vent);
        return "{O:Ok}";
    }
    else {
        return "{E:No se ha encontrado la venta.}";
    }
}
```

Mejoras en la web

Se han diseñado las páginas de recibirOfertas y enviarOfertas.

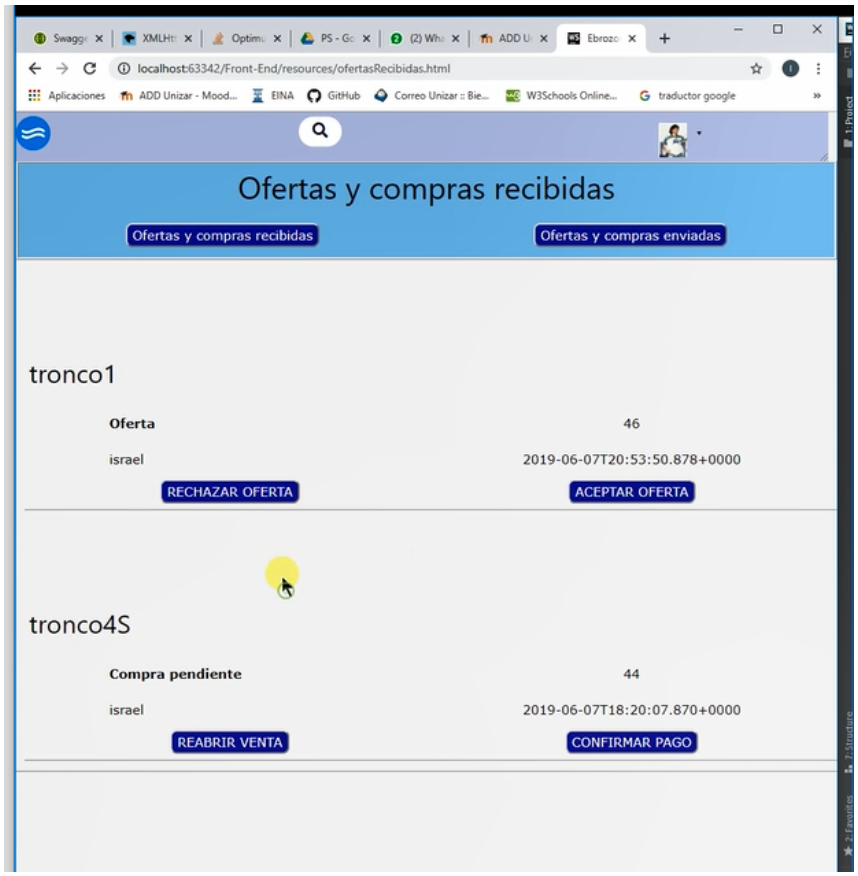
Aunque no se han probado exhaustivamente todas las opciones y se ha encontrado algún error de última hora, la mayor parte de la funcionalidad está presente. Se puede confirmer pago, reabrir venta, etc.

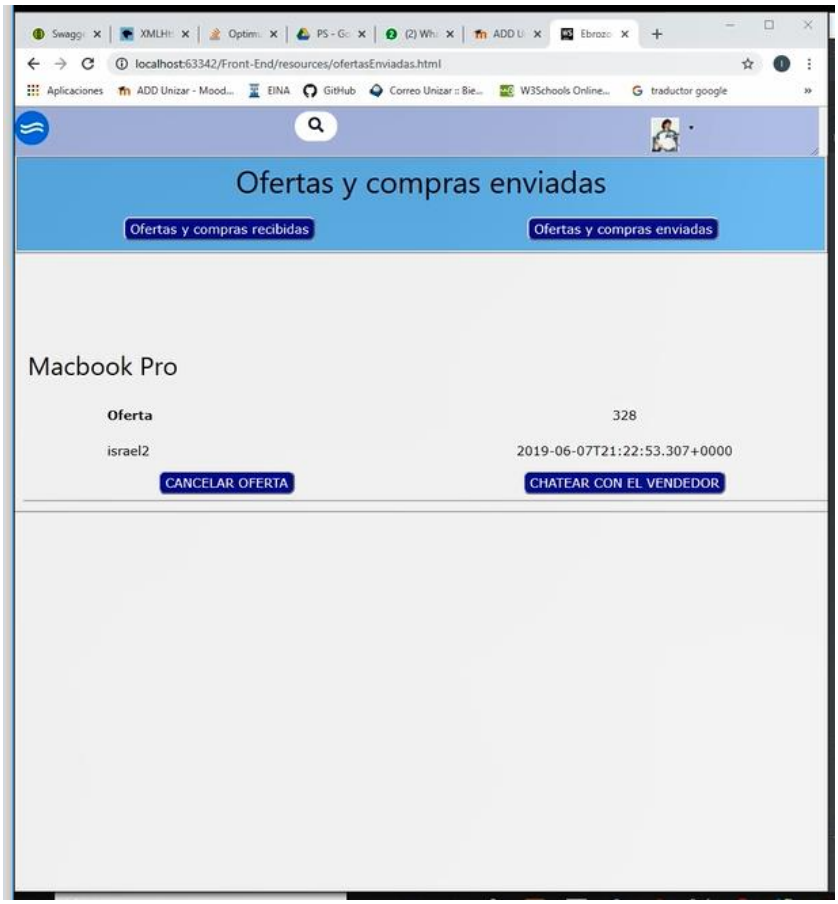
Se incluye un video explicativo en el siguiente link de Google Drive:

https://drive.google.com/file/d/1LGk_xjps5ODfWdGbAQOz7PnJdYWyS_cp/view?usp=sharing

Si no funciona probar este:

https://drive.google.com/drive/u/1/folders/11jtNuWHKTwiEpXXtbny0j4cae5GLUb5q


localhost:63342/Front-End/resources/ofertasRecibidas.html
Aplicaciones
ADD Unizar - Mood...
EINA
GitHub
Correo Unizar :: Bie...
W3Schools Online...
traductor google
Ofertas y compras recibidas
Ofertas y compras recibidas
Ofertas y compras enviadas
tronco1
Oferta
46
israel
2019-06-07T20:53:50.878+0000
RECHAZAR OFERTA
ACEPTAR OFERTA
tronco4S
Compra pendiente
44
israel
2019-06-07T18:20:07.870+0000
REABRIR VENTA
CONFIRMAR PAGO



ANEXO II. ACTAS DE TODAS LAS REUNIONES REALIZADAS

Reunión 1 - grupo 02. Mary Allen Wikes

Lugar: Lab 2.11
Fecha: miércoles 20 de febrero
Hora: 11:00 aproximadamente

Asistentes: todo el equipo y el profesor Rubén Béjar.

Uno de los integrantes del grupo, Saúl Alarcón Cano, lleva a cabo la labor de representante del grupo y además se encarga de tomar anotaciones durante la reunión.

Puntos del día:

- Presentación del grupo y apoyo en la preparación de propuesta técnica y económica.
- Orientación sobre las tecnologías y lenguajes de programación a emplear para el buen desarrollo del proyecto.

Se comparte con el profesor vía USB el documento desarrollado con la propuesta técnica y económica en formato PDF.
Se procede a realizar la fotografía del grupo y el etiquetado: para lo primero, el profesor pide al grupo que se junte para hacer una foto del mismo y, así identificar mejor a cada integrante. Posteriormente, para el etiquetado, cada miembro del grupo dice su nombre y apellidos.

El profesor comienza a revisar el documento de la propuesta técnica entregado.

Ideas de intervenciones del profesor:

Sobre la propuesta técnica

Hay que redactar mejor la introducción: "Todos los sistemas deben cumplir las necesidades del usuario" es demasiado generalista y resulta obvio. Intentamos escribir más complejo de lo que debemos.

Simplificar las explicaciones y no desviar el foco de atención sobre lo importante. [Por ejemplo: "Este proyecto describe una aplicación que va a permitir tal"].

A modo de comentario o sugerencia, se pueden establecer personajes predefinidos con roles y describir sus necesidades. [Por ejemplo: "Juan es un usuario que va a acceder al sistema frecuentemente pero sólo va a utilizar la parte del sistema orientada a realizar tareas de tipo <X>"]

Especificación imprescindible: implementar la geolocalización en base a la distancia [Por ejemplo: "opción para buscar vendedores de un producto <X> dentro de un radio de alcance (por ejemplo, definido alrededor de 5 kilómetros) respecto mi ubicación".

Hay que concretar más los requisitos (p.d. ¿cuándo se puede editar la información de una venta?)

Desde el punto de vista del desarrollo la validación por número de teléfono es "equivalente" a hacerlo mediante correo (e-mail) para simular dicha función.

De cara a un cliente, las características relacionadas con cómo se hará la API y similares no les es interesante para la propuesta

Hitos del proyecto que le pueden interesar al cliente para el programa del proyecto:

- -Para esta fecha he negociado que habrá una primera versión para probarla.
- -Enseñar un prototipo de la GUI por ejemplo para tal fecha.
- -Fechas de presentaciones para el cliente en general.
- -Fecha de entrega.

Hacer un manual para el cliente de cómo desplegar el sistema y otro de usuario (mostrar funcionalidad, navegación, etc.).

Más que trabajadores como tal, resultan de especial interés los perfiles (por ejemplo, especializados en front-end) o grupos de trabajo.

Sobre el presupuesto

Hay que intentar convencer al cliente de que no le cobramos de más aunque realmente se haga.
Evitar demasiado detalle, pero lo suficiente para que se haga a la idea y lo pueda entender.
[Tiempo de diseño de la web, tiempo de la aplicación, tiempo de tal...]

Que pueda entender con términos sencillos (entendibles) lo que le estas cobrando.

Coste

División en módulos o componentes.
Ir dividiendo hasta que tenga el tamaño suficiente como para poder valorar el coste.
Voy a tener que escribir tantos DAOs, tantas cosas o así.
Cuánto va a costar hacer la funcionalidad del login, del registro, etc

Elegir entre división funcional (basada en los requisitos del sistema) o estructural (especificar con un nivel de detalle óptimo los diferentes componentes del sistema [por ejemplo: base de datos relacional, servidor web, aplicación cliente, etc.]).

Los costes de gestión y tal son costes que normalmente se calculan proporcionalmente.
Amortización: saca el valor a partir de un número mágico.
Hay que reflejar las horas de uso de ordenador, por ejemplo 3 semanas de uso.

Mejor dar un precio redondo y redondeando hacía arriba.
Justificar al cliente el precio [Por ejemplo, factura de taller: recambio de motor, cambio de aceite, sustitución de un elemento del sistema eléctrico, etc.]

Formas de pago y fraccionamiento

Se negocia con el cliente
Pensar en pago intermedio por una media entrega o una aplicación a medio desarrollo [Por ejemplo: concertar cita con el cliente pasados 2 meses de desarrollo para mostrar una primera versión mínimamente funcional del sistema y efectuar parte del pago].
Nadie va a pagar algo por nada (sin haberle entregado el producto o parte de él)

Anexo 1

La hoja de cálculo tiene que ir ahí.
Nunca se le entregaría al cliente, pero para esto sí.
En el análisis de requisitos puede ir un miniprototipo de la GUI.
Pensar en nuevas tecnologías es el camino que parece que tenemos que seguir.
Dividir el aprendizaje de nuevas tecnologías y/o lenguajes de programación.
Consejo del profesor (adaptación de las palabras textuales del profesor Rubén Béjar):

> "Creo que debéis subir un poco el nivel de abstracción con nuevas tecnologías…"
> "Tecnologías como los Servlet están un poco desfasadas hoy en día y son de bajo nivel…"
> "Con otros lenguajes de programación casi no hará ni falta utilizar SQL…"

Objetivos acordados:

- Terminar de elaborar el documento final (evaluable) con la propuesta técnica y económica. Especialmente se elaborará la plantilla en hoja de cálculo sobre el presupuesto y ejercicio económico a realizar atendiendo a las indicaciones del profesor, esto es, siguiendo una división funcional o estructural del sistema.

- Aprender nuevas tecnologías para back-end y para front-end. Previsiblemente, se asignará un lenguaje de programación o framework nuevo a emplear por cada pareja de integrantes del grupo.

El encargado de levantar acta en el día de hoy, 20 de febrero de 2019, propone pasar a limpio el presente documento a otro integrante del equipo, que resulta recaer en Israel Solanas.

El encargado de pasar a limpio el acta en el día de hoy, 25 de febrero de 2019, cierra la edición de este documento y propone su revisión al resto de integrantes del grupo, a partir de las 13:08h.

*Un saludo a todos. Y **¡buen trabajo!***

Saúl ALARCÓN CANO e Israel SOLANAS NAVARRO

Reunión 2 - grupo 02. Mary Allen Wikes

Lugar: Lab 2.11
Fecha: miércoles 6 de marzo
Hora: 12:00 aproximadamente

Asistentes: todo el equipo y el profesor Rubén Béjar.

Uno de los integrantes del grupo, Israel Solanas Navarro, lleva a cabo la labor de tomar anotaciones durante la reunión.

Puntos del día:

6. Realimentación propuesta económica y documento técnico
7. Revisar documento de plan de gestión del proyecto

<u>Ideas de intervenciones del profesor:</u>

Le pedimos tutoría

Entregar fuente sin poder desplegarlo (funcionando piezas sueltas) -> no vale para nada

El código en sí (intermedio y/o a medias) sin funcionar no vale de mucho

Al menos un informe de avances

Sol: Desplegar en nuestas maquinas

No diferenciar parte del coste (10000) entre mis gastos (interos) y costes reales del proyecto

Otros costes no se le puede poner

Descripcion tecnica -> sobrecargada (demasiada info)

Impresión:

Porcentaje de nota 70-80% -> falta riesgo técnico (a partir del documento técnico)

Muchos requisitos -> no aportan demasiado (solo ampliar requisitos originales/novedosos)

Falta temas de arquitectura especial o app Android

NO PONER LA CUENTA CORRIENTE EN EL PRESUPUESTO

El proveedor no suele exigir nada

--

270h para aprender nuevas tecnologías -> demasiadas

100h GUI de movil -> demasiado

Capa interaccion con la BD (Tiempo: jornada completa de 1 persona durante 2 meses) -> demasiado

Aprender a ser eficiente -> aprox. 600h

Mejora de la usabilidad de la APP -> Parte del diseño de software

Sol: no separarlo de la

MODIFICAR COMENTARIOS en propuesta enconomica

- **FORMULARIO Moodle para <u>CONTROL ESFUERZOS</u>**

De cara a los profesores -> registrar info de control de esfuerzos periódicamente

- Sugerencia: **<u>compromiso de mantener actualizado el control de esfuerzos 1 vez por semana (Moodle -> Saúl se encarga de recordar)</u>**

Sugerencia -> Hacer las tareas entre pocas personas [ej: diseño BD -> 2 personas]

----------PLAN DE GESTION----------

Relacion entre Hitos, responsables, etc. & Back-end y front-end

Obviedad: proyecto divido en front-end y back-end

ROL: quién va a diseñar la BD, la API, la

Sol: marcar actividades explcitamente

Sol1: criterios técnicos [encargado de las plantillas de la GUI y estilo general]

Sol2: Gestión [encargado de [revisar control de esfuerzos, responsable de Git (gestión de conjuraciones) , revisar que se siga criterios de programación (estilo de codificación Google, C++), uno con rol de analista / diseño, alguien que resuleve conflictos en GitHub o arregla chapuzas]]

Sol3: responsable de

- **Si la actividad es importante -> que haya varias perdonas invoklucradas**
- **Objetivo: conseguir partes en que cada una de ellas pueda hacerla como mucho 2 personas en nas cuantas horas**
- **2 divisiones posibles:**
  - **Horizontal: por razones tecnológicas**
    - **[Hay que hacer 7 DAO y se reparten uno por persona]**
    - **Frontend [**
    - **PROGRAMAR PUNTOS DE JUNTAR TODAS LAS PARTES (comprobar funcionamiento en global)**
  - **Vertical: una persona se dedica a una parte desde la BD hasta la GUI**

Paralelizar Android y Web

TODO LO QUE HEMOS DICHO DE HACER -> HAY QUE HACERLO

Identificacion y ¿?? De recursos

Qué necesitamos para realizar el proyecto

Sugerencias:

Definir las tecnologías software

Repositorio de control de versiones usado -> GitHub

Para hacer pruebas: Travis o hacer pruebas en mi PC de vez en cuando, hacer pruebas en Google Cloud, hay que pasársela al profe en Docker,

Entornos de desarrollo -> **DEFINIR**

Sol: Declarar si usamos GitHub y EasyTracker explícitamente
Cuanto más concreto mejor
Entornos de desarrollo
Responsable de configurar el Cloud

NO DEJAR DE RESOLVER COMENTARIOS DE RUBEN (comprueba cualés hemos resuelto)

COMUNICACIONES INTERNAS

Definir en que formato y donde dejamos los documentos

Añadir los servicios de mensajeria/ comunicación que usamos para comunicarnos [DISCORD: dividido en varios canales por tematrica (front end,…) y documentación)

Actas -> solo necesarias en reuniones formales o con resultados interesantes
Sacar las conclusiones/decisiones tomadas en un documento aparte

- Establecer tareas a 3 semanas vista (pull de tareas), evitar hacer reuniones (Sol: easytracker de github o el director manda), tener un sitio con un listado de las tareas, no repartir las tareas por sorteo (asignadas por el director, por pericia, por disponibilidad de tiempo), MINIMIZAR REUNIONES (minimizar num personas para realizar cada tarea), disputas (surgen cuando tenemos **feedback del estado del proyecto** y sabemos quien esta haciendo qué [alguien no esta haciendo nada y Saúl le manda hacer una tarea X])
- Definir el despliegue:
  - Uso Google Cloud
    - Instalo XAMPP
      - De XAMPP no uso PHP pero si MariaDB y voy a teclear el comando 'X' para noseque
- Definir estándar de coficiacion (estilo de codificación)
- Definirir formato archivos (nombre_tipo_version) -> decir que es nombre, tipo y versión en concreto
- Definir para qué vamos a usar el contenedor Docker 'X',
-
- Responsable concreto para DESPLIEGUE semanal -> **AÑADIR A LOS <u>ROLES</u>**
- Corregir **REPOSITORIO por RAMA** MASTER [necesitamos un repositorio para hacer las pruebas….]

PAQUETES, MODULOS Y ¿COMPONENTES?

Definir en concreto c/u

Componentes y modulos tinen que aparecer las interfaces

El encargado de levantar acta en el día de hoy, 20 de febrero de 2019, propone pasar a limpio el presente documento a otro integrante del equipo, que resulta recaer en Israel Solanas.

El encargado de pasar a limpio el acta en el día de hoy, 25 de febrero de 2019, cierra la edición de este documento y propone su revisión al resto de integrantes del grupo, a partir de las 13:08h.

*Un saludo a todos. Y **¡buen trabajo!***

Saúl ALARCÓN CANO e Israel SOLANAS NAVARRO

Revisión de la memoria 1-4-2019

Organizacion -> rol y responsabilidad equivocados
- -Director es rol
- -Rol es una etiqueta, y eso lleva unas responsabilidades

Gestión del proyecto
- -Processos son los comos, un workflow o un algoritmo
  - -Lista de pasos para corregir algo por ejemplo
  - -Procesos para crear un repositorio, para crear un grupo
  - -Procesos de gestión
- -Planes: establecerse los objetivos
  - -Checkeo de los proyectos por ejemplo -> de quí puede salir un proceso de comprobación
    - -Por ejemplo comprobar las horas de esfuerzo o etc
    - -Proceso de verificiación de la calidad por ejemplo cada dos semanas y su consecuencia o acción

Procesos de inicio
- -Cosas que tndría que saber un nuevo empleado nada más entrar al proyecto
  - -Con quien tiene que hablar para darse de alta, que se tiene que instalar o configurar, etc
  - -Chrome firefox y tal no son procesos de inicio
  - -Idea de las tecnologías que se deben conocer o formarse para prepararse
  - -Referenciar los manuales o tutoriales de preparación por ejemplo

Procesos de ejecución
- -Añadir como se determina la distribucion de tareas
- -Distribución a través de la modularización del sistema -> Diseño de modulos o paquetes
  - -En el caso del back-end las tablas
- -Diseñar el modulo de dependencias para saber por donde empezar
- -Modelo de datos pueden dar ciertas pistas de las dependencias aunque no sean definitivas

Reparto de tareas -> no por sorteo, sino por disponibilidad (memoria)

Para las tareas que pueda hacer cualquiera -> el que menos horas de esuerzo tenga dedicadas

Entrega -> concretar algo más que github: claves? ramas? versión?
-Cosas que puedan no estar en github
-Commit etiquetado con versión final

Horas de reunión -> reducirlas, no es realista, además de pérdida de tiempo

Concretar integraciones: al menos 2 además de las entregas (no es necesario estar los 9)
Una o dos personas de cada subequipo debería ser suficiente

Revisión del trabajo semanal puede ser un proceso:
-Mirar tareas pendientes realizada por cada uno
-Revisar horas
-Revisar commits
-Revisar ficheros subidos etc
-Etc

Procesos técnicos
-No tenemos procesos como tal
-Procesos de integración -> ejecución del sistema completo -> como lo vamos a hacer
-Una integración consiste en desplegar el sistema completo en la version que se tenga
-Test exploratorios -> pruebas manuales

Plan de gestión de configuraciones
-No incluir versiones en los nombres de los ficheros
-Si que puede tener sentido darle nombre de versiones a los wars o jars que se generen

Estándares de códigos -> guías de estilo
-HTML5 no es lo que se pide

Puesta en marcha -> integración -> mejor una persona especializada

Deberíamos también tener backups personales -> a tener en cuenta

Qué son los repositorios de Prueba y versiones funcionales? nada

Apache puede no ser necesario, vamos, no debería ser necesario

Donde se va a probar las integraciones, y las personas que se van a asignar, ya que hará falta más de 1

No se concreta que es desplegar sobre google cloud
-Orden de subir ficheros
-Configuración de este

Los repositorios, ramas, etc,no garantiza la calidad del código

Comprobaciones por pares, analizador de código, etc

Afinar un poco más los test que se va a escribir, y semejantes, establecer un objetivo
De qué tipos son los test
De quién se encarga el hacer los test globales, como el de estrés, si van a ser manuales o autómaticos
Por ejemplo el de estrés habría que hacerlo después de la primera integración exitosa.

Calendario de la segunda iteración
En este momento es todo se desarrollará a la vez, y se desplegará todo, eso no representa demasiado el paralelismo
Estaría bien que apareciesen las integraciones, para poder decir: se hacerla la fecha de integración y falta esto...
Si ponemos por ejemplo una integación, insertar un test de estrés
Tareas más pequeñas, que se puedan decir si están o no están
Idealmente se tendría que asignar cada tarea a una persona

Las 9 personas son solo necesarias para decisiones importantes del sistema, pero el resto se debería poder dividir
en más esquemas y fraccionar estos.

Identificar los requisitos, para poder referenciarlos desde los otros puntos fácilmente

---

Preguntas a pensar:

¿Se está cumpliendo el calendario?

¿Si no se está cumpliendo, se ha ajustado?

¿A qué se han debido los ajustes?

¿Ha habido que hacer cambios de tecnologías o algo semejante?
Debe reflejarse, sobretodo si va a cambiar el calendario

¿Qué tal los grupos de comunicación?
Todos los documentos que se escriben son comunicación
Donde registramos las decisiones y tal

¿Medida de progreso?
Los requisitos no son demasiado explicativos
Número de módulos, líneas de códigos, issues, horas, etc

¿Lista de tareas pendientes?
Lista de tareas globales
Unificar el proyecto

¿Habéis hecho algún despliegue?

¿Váis diseñando test de pruebas a medida que escribís código?

Pruebas manuales?

Reunión 4 - grupo 02. Mary Allen Wikes

Lugar: Lab 2.11
Fecha: miércoles 10 de abril
Hora inicio: 12:08 aproximadamente

Asistentes: todo el equipo y el profesor Rubén Béjar.
Uno de los integrantes del grupo, Israel Solanas Navarro, lleva a cabo la labor de tomar anotaciones durante la reunión.

Puntos del día:

8. Realimentación documento de plan de gestión del proyecto
9. Revisión del calendario y diagrama de Gantt

Ideas de intervenciones del profesor:

REQUISITOS

Marcar con código (tipo RF-1) cada requisito para hacer referencia a ellos a lo largo de los documentos

Falta requisito de listar

Añadir mapa de navegación o actualizar/completar lista de requisitos para que se refleje lo que los usuarios pueden hacer en cada pantalla

-REQUISITOS QUE FALTAN:

Filtrado de productos por provincia

Scroll inifinito en pantalla de login (**puede darnos problemas que el scroll sea infinito)**

Filtros: ciudad, etiqueta

- **Uniformidad entre aplicación APP y WEB**
- Detallar bien los **REQUISITOS** y hacer **MAPA DE NAVEGACIÓN** (de la GUI)
- Todo **lo que aparece en la APP o WEB** debe estar **fundamentado** en **un REQUISITO** (no se debería NO implementar un requisito ni hacer algo que NO este en un requisito)

DIAGRAMA DE GANTT

Lo que falta:

- Asignación de recursos a cada persona

- Dependencias (qué tareas dependen de otras [esta tarea depende de que haya acabado la otra])

- Los hitos se marcan con una "especie de DIAMANTE"

- Un plan que no tiene hitos -> NO se revisa hasta el final (**hay que marcar HITOS)**

- Está bien hacer 2 diagramas de Gantt (una para la 1º iteración y otra para la 2º)

Poner las personas que tienen asignados cada tarea

Conclusiones sobre los requisitos funcionales:

5. Sistema debe permitir que un usuario pueda tener una foto -> falta
6. Sistema debe permitir ubicación aproximada -> soporte en la BD pero no hay APP
7. usuario modifique su información -> Back SI \ Front NO
8. calificación usuarios -> nada
9. NADA
10. Se puede bloquear usuario etc. -> si se puede bloquear
11. Sistema debe permitir poner a los usuarios en venta objetos poniendo precio de compra inmediato -> empezado en FRONT
12. Archivos y etiquetas -> Sin probar
13. una etiqueta será 1 o mas palabras clave -> Solo empezado en BACK (sin usar)
14. permitir crear etiquetas al realizar una venta -> NO está
15. subasta -> NO está
16. editar información y precio del producto ->

17. Sistema debe permitir
18. Venta inmediata -> No está
19. Realizar rechazar ofertas -> no esta
20. No está
21. Hacer pujas -> NO
22. Lista productos -> NNO
23. Lista siguiendo > NO
24. Búsqueda productos por Palabras clave -> NO
25. Listar productos en función de localización y etc. -> no
26. Ver valoraciones propias y de otros usuarios -> NO
27. Reportar usuarios -> no
28. Enviar mensajes -> NO
29. Enviar mensajes -> NO
30. Soportado en la mayoría de los navegadores -> NO
31. APP web diseño responsive -> NO
32. Vincular la cuenta de usuario a Google -> desaparecerá
33. Registrar acciones realizadas por los usuarios -> SI está
34. Requisito de reportar y banear -> NO está -> pero con mandar un email al administrador por medio de que un usuario 1 le da aun botón de reportar al usuario 2
35.

2 requisitos completos y 12 a medias, 16 SIN HACER

FALTAN REQUISITOS [listar productos por fecha, …]

Conclusiones finales y recomendaciones:

Subasta, venta y chat -> partes **criticas** (llevan muchas horas -> hay que hacer lógica de la web [invariante, por ejemplo: no puedo retirar un producto de una puja si ya hay alguna oferta en esa puja])

Integracion de front-end y back-end -> es lo más **urgente** (empezar por lo más difícil)

Hacer **Mapa de navegación** (web y APP)

Detallar requisitos (numerarlos, eliminar los que no se vayan a implementar e incluir los que falten)

Lo más **importante** es ir **terminando requisitos**

Marcar requisitos [ejemplo: RF-1 el sistema permite registrar a un usuario con info X e Y]

Ponernos objetivos (por ejemplo, el 1 de mayo hay que llegar a tener los RF-1,3,7, etc.

El encargado de levantar acta en el día de hoy, 10 de abril de 2019, y pasarla a limpio es Israel Solanas.

Tras pasar a limpio el acta durante el mismo día de la reunión, 10 de marzo de 2019, se cierra la edición de este documento y se propone su revisión al resto de integrantes del grupo.

Se ha corregido la expresión de algunas frases el día 15 de abril.

*Un saludo a todos. Y **¡ánimo con el trabajo que nos queda por delante!***

Israel SOLANAS NAVARRO

Reunión 5 - grupo 02. Mary Allen Wikes

Lugar: Lab 2.11
Fecha: miércoles 8 de mayo
Hora inicio: 12:00 aproximadamente

Asistentes: parte el equipo y el profesor Rubén Béjar.
Uno de los integrantes del grupo, Israel Solanas Navarro, lleva a cabo la labor de tomar anotaciones durante la reunión.

Puntos del día:

10. Realimentación documento de plan de gestión del proyecto
11. Revisión del calendario y diagrama de Gantt
12. Preguntas sobre cómo va el desarrollo de proyecto

Ideas de intervenciones del profesor:

MEMORIA

Falta el apartado 5
Descripción de interfaces
Diagrama de componentes y conectores -> Poner 1 ejemplo y decir que los demás son iguales o Detallar MENOS o MAS que actualmente, pero **NO MENTIR**.
Señalar el conector concreto o protocolo de comunicación entre la BD y Servidor o Cliente-Servidor [i.e.: JDBC para l BD o HTTP res para el cliente-Servidor pero no "DBAcess"]
Vistas de instalación -> demasiado detalle (y no son muy interesantes)

Además, cuidado con poner que desplegamos fichero .java porque son .class

NECESARIO:

Diagrama de componentes (hacerlo sin mucho detalle)

Diagrama de módulos (**OPCIONAL, el de componentes es suficiente**)

PRIORIZAR ESTAS PARTES:
Figuras y tablas son PRIORIDAD, deben reflejar bien lo que se ha hecho (ser fieles al contenido)
Introducción, conclusiones y titulo

Gestión del grupo

Faltan las flechas en el calendario, hitos y dependencias
UNIFICAR LAS METRICAS DE LOS 3 GRUPOS (unificar la gestión [todos tengan su lista de requisitos completados])
~~Falta referencias a los requisitos en la memoria entregada -> ESTA HECHO~~

Medida cuantitativa del proyecto (GESTION):

Tenemos un proceso en que cierra un issue cuando el jefe de departamento prueba y certifica que dicho issue funciona y se documenta automáticamente que está cerrado dicho issue

USAR OTRAS MEDIDAS COMO:

Número de líneas de código
Tiempo dedicado

Lista de tareas con asignación a personas:
Diagrama de Gantt, listado de tareas,

Integración (fallos comunes)

Todos los requisitos deben estar en la API y tener su correspondiente botón en la GUI

PREGUNTAS

¿Tenemos calendario y sabemos como vamos respecto al calendario?

Sí en general, salvo que el front end se ha retrasado frente al back end

¿En caso de que cambie algún plazo se ha reajustado el calendario?

ESTO ES IMPORTANTE

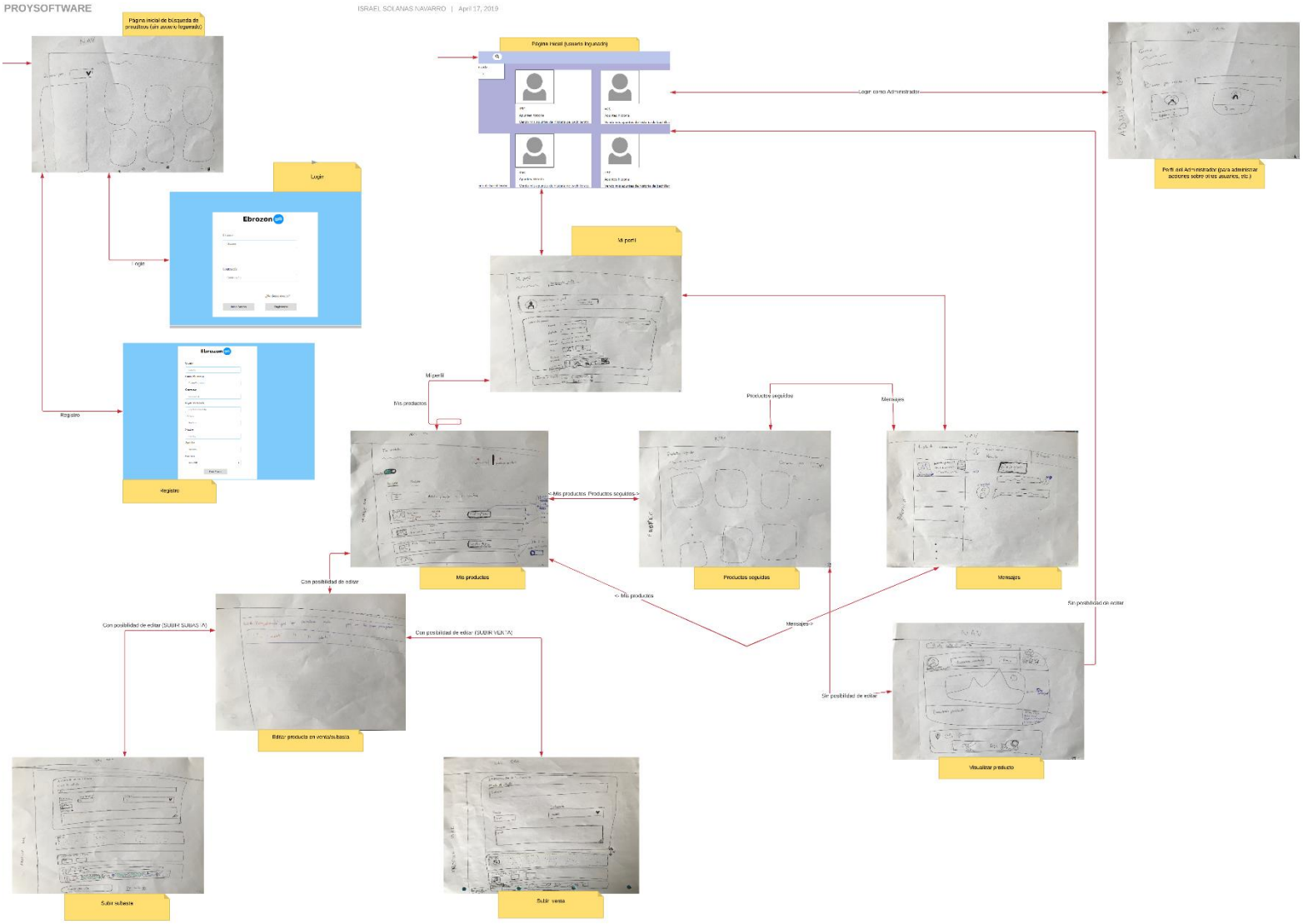
¿En caso de sacrificar algo por falta de tiempo, que sería?

La estética de la APP y WEB

REQUISITOS COMPLETOS

Faltan muchos requisitos por completar -> recomendable MANTENER lista de REQUISITOS ***UNIFICADA*** *y* ***ACTUALIZADA*** *para WEB, APP Y API (Back-end)*

ANEXO III. MAPA DE NAVEGACIÓN



ANEXO IV. PROPUESTA TÉCNICA Y ECONÓMICA

Contenido

Resumen ejecutivo

En este documento se presenta la propuesta de un sistema de compra-venta de objetos a través de una aplicación web. El sistema permitirá a los usuarios subir información de los productos que desean vender a un servidor. Estos productos se mostraran a los usuarios interesados en comprar, que podrán decidir si comprar el producto al precio propuesto por el vendedor o enviarle una petición de compra por otro precio. También existe la posibilidad de que los usuarios establezcan subastas sobre productos en la que los posibles compradores irán pujando hasta que una única persona gane la subasta.

Los usuarios podrán filtrar las búsquedas para que solo se muestren aquellos productos que cumplen unas determinadas condiciones, también existe un sistema de geolocalización para mostrar a los usuarios los productos más cercanos a ellos.

La aplicación facilitara la comunicación entre vendedor y comprador una vez se haya llegado a un acuerdo sobre el precio para que estos ejecuten la transacción como deseen.

Se realizarán 4 entregas, la primera el día 06/03/2019, en la que se presentará un prototipo de la interfaz de usuario en papel, además de un mapa de navegación del mismo.

La siguiente entrega será para la segunda semana de abril, en la que se presentará una primera versión funcional del sistema y se entregarán los respectivos fuentes; El objetivo de esta segunda entrega es la obtención de retroalimentación por parte del cliente hacia la versión del sistema mostrada, con esta retroalimentación se modificaran diversas características del sistema para acercarse más al producto solicitado por el cliente.

La tercera entrega se realizara para la tercera semana de mayo. En esta se presentara al cliente una versión intermedia del sistema y se entregaran los respectivos fuentes; esta será la última entrega en la que el cliente podrá ofrecer retroalimentación sobre el sistema, por lo que es importante obtener todas las opiniones del cliente sobre el sistema.

La entrega final está prevista para finales de mayo, en esta se presentara la versión final del sistema y se entregaran sus fuentes.

El precio a pagar por el cliente asciende a un total de 22.700€, repartiéndose este entre desarrollo de la aplicación web (4.625 €), desarrollo de la aplicación móvil (2.127€), implementación de la base de datos (5.364€) y costes de gestión, aseguramiento de calidad y otros costes (10.049€).

Objetivos del sistema

El objetivo principal del sistema es facilitar la compraventa de productos entre particulares, permitiendo subir los mismos y poniendo en contacto a compradores y vendedores.

Análisis de requisitos preliminar

13. El sistema debe soportar la existencia de usuarios, que se definen mínimo por un nombre de usuario y un correo electrónico, pudiendo incluir una descripción, geolocalización y/o una foto.
14. El sistema debe permitir modificar la información del usuario.
15. El sistema debe permitir el bloqueo de cuentas en caso de infracción.
16. El sistema debe permitir a los usuarios poner en venta objetos, proporcionando información del producto y un precio de referencia.
17. El sistema debe permitir al vendedor editar tanto la información como el precio de un producto propio a la venta.
18. El sistema debe permitir a los usuarios poner en subasta objetos, siendo estas definidas por un precio mínimo, una fecha límite y un precio de compra inmediata, además de la información del producto.
19. El sistema debe permitir al vendedor retirar la venta de un producto propio (siempre que el producto no esté en subasta y ya haya recibido una puja).
20. El sistema debe permitir a los usuarios realizar ofertas a los vendedores por un objeto en venta normal.
21. El sistema debe permitir a los vendedores aceptar o rechazar ofertas de un producto propio en venta.
22. El sistema debe retirar un producto de la lista de productos en venta, cuando el vendedor ha aceptado una oferta por este.
23. El sistema debe permitir a los usuarios realizar pujas por un producto en subasta.
24. El sistema debe permitir a los usuarios solicitar la compra inmediata de un producto en subasta.
25. El sistema debe permitir a los compradores aceptar o rechazar la compra inmediata de un producto propio en subasta.
26. El sistema debe cerrar la subasta de un producto cuando un vendedor confirma la compra inmediata de un producto en subasta.
27. El sistema debe permitir la búsqueda de productos mediante palabras clave.
28. El sistema debe permitir filtrar una lista de productos a la venta en función de localización, categoría, precio...etc.
29. El sistema debe permitir al comprador puntuar al vendedor, y viceversa, al finalizar la transacción de una compra.
30. El sistema debe permitir a los usuarios ver las valoraciones propias recibidas y las de otros usuarios.
31. El sistema debe permitir a los usuarios reportar a otros usuarios en caso de una infracción las reglas de uso (p.e. caso de estafa).
32. El sistema debe permitir a los usuarios enviar mensajes a otros usuarios.
33. El sistema debe permitir a los usuarios recibir mensajes de otros usuarios.
34. La información de un producto se compone por nombre o título del título, fotos,

vídeos y/o una descripción.

35. El producto resultante será una aplicación web soportada en la mayoría de navegadores.
36. La aplicación web tendrá un diseño responsive enfocada a los dispositivos móviles.
37. El sistema requerirá vincular la cuenta de un usuario a un número de teléfono móvil

Descripción técnica

El producto que se entregará al cliente es una aplicación web que se entregará instalada en un servidor virtual desplegado, concretamente en Google Cloud por ser previsiblemente la oferta más económica de entre los servidores web existentes, y una aplicación nativa de Android,. Se entregarán al cliente las correspondientes claves e información para acceder al sistema en el servidor en que se despliegue la aplicación, además se le entregará el código fuente.
La aplicación web se basará en HTML5, que es compatible con los principales navegadores.
Para almacenar los datos usados por la aplicación se empleará una base de datos relacional con el SGBD PostgreSQL, lo que puede restringir el uso de otro gestor ya que no se puede asegurar su correcto funcionamiento debido a las diferencias que hay entre estos.
La aplicación móvil será nativa para Android, desarrollada en paralelo a la aplicación web.

Plan de trabajo

Fecha	Hito	Descripción	Entregable
6/3/2019	Prototipo de la GUI	Presentación de un prototipo de la versión inicial de la GUI.	Prototipo en papel o Paint, con mapa de navegación
8/4/2019 - 12/4/2019	Demostración inicial del sistema	Presentación y demostración de una primera versión funcional del sistema	
13/5/2019 - 17/5/2019	Demostración de una versión avanzada del sistema	Presentación y demostración de una versión avanzada del sistema	
27/5/2019 - 31/5/2019	Demostración de la versión final del sistema	Presentación y demostración de la versión final del sistema	Versión final del proyecto

Equipo técnico encargado del proyecto

La empresa **Ebrozon Development Team** se especializa en la realización de aplicaciones web, con capacidad de portabilidad a otras plataformas y de comunicación entre usuarios. Pese a especializarse en aplicaciones web, el equipo que compone la empresa está capacitado para realizar un amplio abanico de trabajos de diferentes ámbitos de la informática, como sería el desarrollo de software en general, proyectos que necesiten de aprendizaje automático o gestión de sistemas.
Nuestros clientes han sido fundamentalmente integrantes del Departamento de Informática e Ingeniería de Sistemas de UNIZAR, realizando aplicaciones de diversa temática. El proyecto más grande y reciente, sería el desarrollo de una aplicación web para los profesores de la asignatura de Medioambiente, que buscaban un medio para difundir a un mayor público mensajes de concienciación mediante la publicación de carteles, noticias, etc.

Alejandro Cano Somalo

Capacidades técnicas:

5. Desarrollo de una aplicación para Android, realizando el diseño y la implementación.
6. Conocimientos de lenguajes de programación, fundamentalmente C++ y Java. Otros lenguajes utilizados: Haskell, Elixir, CLIPS, scripts en Bash y gestores de bases de datos relacionales como Oracle y MySQL
7. Conocimientos básicos sobre concurrencia y distribución de programas informáticos.
8. Conocimientos básicos sobre sistemas de recuperación de información.
9. Conocimientos básicos sobre inteligencia artificial y técnicas de aprendizaje automático.

Andrés Gavín Murillo

Participación en el desarrollo de los siguientes proyectos:

- Aplicación de notas para Android (2018)
- Reversi8 para una placa Embest S3CE40 EVB (2018)
- Sistema Linda en un entorno Linux (2019)
- Aplicación para nevera inteligente (2017)
- Sistema de subastas de imágenes en un entorno Linux (2018)
- Sistema distribuido tolerante a fallos (2019)

Conocimientos sobre los siguientes lenguajes y arquitecturas:

- C, C++, Java, CLIPS, Haskell, SQL, Bash, Elixir, Python, MIPS, ARMv4 y ARMv7

Anotaciones extra:

- Participación en el evento HashCode 2018 organizado por Google
- Participación en el evento ImagineCode 2018 organizado por CodeLab
- Conocimientos básicos sobre Linux y gestión de servidores
- Conocimientos básicos sobre inteligencia artificial

Eduardo Gimeno Soriano

Participación en el desarrollo de los siguientes proyectos:

- Aplicación de notas para Android (2018)
- Reversi8 para una placa Embest S3CE40 EVB (2018)
- Sistema Linda en un entorno Linux (2019)
- Aplicación para nevera inteligente (2017)
- Sistema de subastas de imágenes en un entorno Linux (2018)

Conocimientos sobre los siguientes lenguajes y arquitecturas:

- C++, C, VHDL, Python, MIPS, UML, Java, CLIPS, Haskell, SQL, Bash, ARMv4 y ARMv7

Anotaciones extra:

- Participación en el evento HashCode 2018 organizado por Google
- Participación en el evento ImagineCode 2018 organizado por CodeLab
- Conocimientos avanzados sobre el sistema operativo Debian

Félix García Rodríguez

Participación en el desarrollo de los siguientes proyectos:

36. Space Invaders en ensamblador ARMv4 (2017)
37. Analizador sintáctico de Cmms (2017)
38. Sistema distribuido de subastas (2018)
39. Pacman en ensamblador ARMv4 (2018)
40. Aplicación para la gestión de tratamientos sanitarios (2018)
41. Sistema distribuido de servidores de almacenamiento mediante primario/réplica y gestor de vistas (2018)
42. Tres bases de datos relaciones dedicadas a partidos de fútbol, vuelos y películas (2018)
43. Aplicación de notas para Android (2019)

Conocimientos sobre los siguientes lenguajes y arquitecturas:

- ARMv4, Bash, C, C++, CLIPS, Elixir, Haskell, Java, Python, SQL, UML y R

Israel Solanas Navarro

Capacidades técnicas:

- Desarrollo de un proyecto en el ámbito de Sistemas de Información con una interfaz web sobre Medio Ambiente.
- Desarrollo de un proyecto software consistente en desarrollar una aplicación informática para editar y gestionar notas.
- Conocimientos de lenguajes de programación y tecnologías como C, C++, Java, Haskell, scripts en Bash, Servlet, etc.
- Conocimientos básicos sobre concurrencia y distribución de programas informáticos.

Jorge Fernandez Muñoz

Participación en los siguientes proyectos:

6. Desarrollo de un sistema de información sobre el Medio Ambiente (2018)
7. Desarrollo de un proyecto software acerca de una aplicación de creación y gestión de notas (2018)
8. Desarrollo de un sistema distribuido basado en subastas de imágenes (2017)
9. Desarrollo del juego reversi8 sobre una placa (2018)

Conocimientos técnicos adquiridos:

C , C++ , Java, Haskell, ARMv4, ARMv7, HTML, SQL, JSP, Python, Elixir, VHDL, Bash
Participaciones destacadas:

Participación en HashCode2018
Participación en UCode 2018
Participación en ImagineCode 2018

Saúl Alarcón Cano

Participación en el desarrollo de los siguientes proyectos:

- Aplicación de notas para Android (2018)
- Reversi8 para una placa Embest S3CE40 EVB (2018)
- Sistema distribuido de simulación de subastas en consola (2018)
- Aplicación para la gestión de tratamientos sanitarios (2018)
- Sistema distribuido de servidores de almacenamiento mediante primario/réplica y gestor de vistas (2018)
- Analizador sintáctico de Cmms (2017)
- Modificación de un procesador MIPS en VHDL (2018)
- Tres bases de datos relaciones dedicadas a partidos de fútbol, vuelos y películas (2018)
- Aplicación web para subida de noticias, carteles y cuestiones sobre el medioambiente (2018)
- Space invaders en ARMv4 (2017)

Conocimientos sobre los siguientes lenguajes y arquitecturas:

- C, C++, Java, CLIPS, Haskell, Elixir, Python, HTML, UML, XML, SQL, SPARQL, Bash, VHDL, MIPS, ARMv4 y ARMv7.

Sergio Álvarez Peiro

Participación en el desarrollo de los siguientes proyectos:

- Aplicación de gestor de notas para Android.
- Desarrollo del juego Reversi8 para una placa Embest S3CE40 EVB.
- Sistema distribuido de simulación de subastas en línea de comandos.
- Prototipo de una aplicación Android para la gestión de tratamientos sanitarios.
- Modificación de un procesador MIPS en VHDL.
- Tres bases de datos relaciones dedicadas a partidos de fútbol, vuelos y películas.

Conocimiento de lenguajes de programación:

- C, C++, Java, Haskell, ARMv4, ARMv7, VHDL, Bash, SQL y HTML.

Víctor Sisqués Cortés

Participación en el desarrollo de los siguientes proyectos:

- Aplicación de notas para Android (2018)
- Reversi8 para una placa Embest S3CE40 EVB (2018)
- Sistema de subastas concurrentes y distribuidas en un entorno Linux (2017)
- Aplicación de pastillero (2018)
- Aplicación web tomcat sobre Medio Ambiente (2018)

Conocimientos sobre los siguientes lenguajes y arquitecturas:

- -C++, C, Java, JavaScript, Elixir, CLIPS, Haskell, SQL, Bash, ARMv4, ARMv7, MIPS, UML, VHDL, HTML y Python

Anotaciones extra:

- Conocimientos sobre tolerancia a fallos distribuida.
- Conocimientos sobre mecanismos de replicación.
- Conocimientos básicos sobre el sistema operativo Debian.

Presupuesto

El pago del trabajo se realizará a través de una trasferencia bancaria. No será necesario un pago por adelantado para el desarrollo de la aplicación, pero sí que se podrá realizar el pago de una señal negociable con el cliente de entorno a un 20% tras la entrega intermedia de la misma. Además, el cliente tiene el derecho a ejecutar una opción de compra total por adelantado en un plazo de quince días tras la firma de este documento con un descuento del 3%.
La cantidad a pagar es de 16512,13€ que se reparten entre el desarrollo de la web, la aplicación móvil, servidor, costes de gestión, aseguramiento de calidad y otros costes.

Glosario

Front-end: capa de presentación de un software (interfaz que interacciona con el usuario).
Back-end: capa de acceso a datos (parte que procesa la entrada desde el *front-end*).
API: conjunto de rutinas que provee acceso a funciones de un determinado software.

Estimación de costes

ESTIMACIÓN DE COSTES						
					18,50 €	
ESFUERZOS					COSTES	
Tarea/componente	Descripción	Cantidad	Horas/ítem	Estimación final	Coste/hora	Coste (€)
Desarrollo de la aplicación web						
Aprendizaje de nuevas tecnologías	Incluye : Aprendizaje de un nuevo lenguaje Aprendizaje del uso de nuevos frameworks	3	10	30	18,50 €	555,00 €
Diseño de la interfaz		1	100	100	18,50 €	1.850,00 €
Desarrollo de la lógica de la interfaz		1	100	100	18,50 €	1.850,00 €
Desarrollo del back end						
Aprendizaje de nuevas tecnologías	Incluye : Aprendizaje de un nuevo lenguaje Aprendizaje del uso de nuevos frameworks	3	10	30	18,50 €	555,00 €
Diseño de la base de datos		1	5	5	18,50 €	92,50 €
Implementación base de datos		1	5	5	18,50 €	92,50 €
Desarrollo de la capa interacción con base de datos		1	120	120	18,50 €	2.220,00 €
Desarrollo de la aplicación móvil						
Aprendizaje de nuevas tecnologías	Incluye : Aprendizaje de un nuevo lenguaje Aprendizaje del uso de nuevos frameworks	3	10	30	18,50 €	555,00 €
Diseño de la interfaz		1	100	100	18,50 €	1.850,00 €
Desarrollo de la lógica de la interfaz		1	120	120	18,50 €	2.220,00 €
Realización de pruebas y corrección						
Pruebas de carga, volumen...		5	1	5	18,50 €	92,50 €
Pruebas funcionales		1	30	30	18,50 €	555,00 €
TOTAL TAREAS/COMPONENTES				675	18,50 €	12.487,50 €
Gestión		15%		101,25	18,50 €	1.873,13 €
Gestión de configuraciones		5%		33,75	18,50 €	624,38 €
Aseguramiento de la calidad		7%		47,25	18,50 €	874,13 €
TOTAL MACROS				182,25	18,50 €	3.371,63 €
Otros costes						
Viajes		6	15,00 €	90,00 €		90,00 €
Amortización equipos desarrollo		900	0,07 €	63,00 €		63,00 €
Coste del primer mes del despligue		1	500,00 €	500,00 €		500,00 €
TOTAL OTROS COSTES				653,00 €		653,00 €
TOTAL						**16.512,13 €**