Plan de gestión, análisis, diseño y memorial del proyecto KeyPaX

Bárbaros Software S.A. (Grupo 3)

Carlos Bellvis, 755452
Jorge Bernal, 775695
Jorge Borque, 777959
Arturo Calvera, 776303
Andoni Salcedo, 785649
Javier Vela, 775593

Organización GitHub: https://github.com/UNIZAR-30226-2021-03

Índice

1. Introducción

2. Organización del proyecto

El equipo del proyecto está formado por los 6 integrantes del grupo. Para dividir el trabajo y las responsabilidades se han formado 2 grupos iniciales, acorde con las capacidades y conocimientos individuales. Además, se ha designado como **director del proyecto** a **Arturo Calvera**, cuya responsabilidad abarca la gestión de los equipos de trabajo y sus responsabilidades.

- **Equipo *Backend***:
  - Responsable: Arturo Calvera.
  - Función: Desarrollo del *backend* del sistema.
  - Integrantes: Arturo Calvera, Andoni Salcedo.
- **Equipo *Frontend***:
  - Responsable: Javier Vela.
  - Sub-equipos: El equipo dedicado a la vista del sistema se divide para el desarrollo de las dos interfaces.
    - **Equipo Web**:
      - ◇ Función: Desarrollo del *frontend* web del sistema.
      - ◇ Integrantes: Jorge Bernal, Javier Vela.
    - **Equipo Android**:
      - ◇ Función: Desarrollo de la aplicación cliente del sistema para dispositivos Android.
      - ◇ Integrantes: Carlos Bellvis, Jorge Borque.
- Se contempla la posibilidad de crear nuevos equipos para las necesidades que surjan durante el proyecto (*e.g.* despliegue, pruebas) o la reconfiguración de los actuales para adecuarse a la situación.

3. Plan de gestión del proyecto

3.1. Procesos

3.1.1. Procesos de inicio

Para asignar los recursos que vamos a utilizar, hemos puesto en común los que conocíamos cada componente del grupo, y a partir de ahí, cuál es el favorito de la mayoría del grupo y en el que nos encontramos más cómodos trabajando.

Los servidores en cloud que más nos llamaron la atención son Amazon Web Services y MIcrosoft Azure. Después de informarnos sobre ambas y compararlas, decidimos crearnos una cuenta en AWS. De primeras, los hemos visto como las 2 plataformas con más impacto a nivel global y que más nos pueden servir para nuestro futuro, a partir de aquí, al no tener ninguno una experiencia previa en ellas, hemos cogido AWS por interés general de todos y por las posibilidades de trabajo que nos ofrece gratuitamente. Respecto a la base de datos, hemos decidido usar MongoDB ya que ha sido utilizada por la mayoría del grupo en asignaturas previas, y por su compatibilidad con AWS. Para su utilización, nos hemos registrado en Mongo Atlas.

Sobre la formación inicial, hemos tenido un debate amplio a cerca de si es mejor que todos conozcamos todas las tecnologías que vamos a utilizar, o si después de haber organizado el trabajo

y el rol de cada uno, se especialice y enfoque su formación en la tecnología que va a utilizar en su caso. Entonces, hemos decidido que todos emplearán tiempo antes de la fecha designada para empezar a realizar el despliegue, en autoformación con tutoriales que vamos poniendo en común sobre las APIs donde vamos a desarrollar nuestro proyecto. Y luego, una vez ya decidido el trabajo de cada uno, enfoque su formación en su tecnología correspondiente, para poder desarrollar sin problemas su tarea y poder ir todo el grupo a un ritmo semejante.

3.1.2. Procesos de ejecución y control

Las comunicaciones entre los miembros del grupo de trabajo se realizan principalmente a través de la herramienta de comunicación *WhatsApp*, mediante la cual concertamos los horarios de trabajo, además de informar acerca de actualizaciones puntuales en las tareas asignadas a cada miembro para que todo el equipo quede informado de cómo avanzan los distintos componentes del trabajo. Además, para las reuniones semanales de puesta en común del trabajo realizado utilizamos la herramienta *meet* desarrollada por *Google*. Esta última sería nuestro principal medio de comunicación. Destacar que pese a tener una división del trabajo a realizar, intentamos juntarnos todos en un *meet* cada vez que nos ponemos a trabajar, de manera que en caso de que alguien tenga una duda acerca de su parte sea más probable que ésta sea resuelta.

En cuanto al registro de las decisiones tomadas en las reuniones, se están plasmando en actas gestionadas por nuestro compañero Javier Vela. Respecto al almacenamiento de estas actas, hemos decidido utilizar la *Wiki* de *GitHub*, que contiene la información respectiva a cada repositorio de código en nuestra organización.

Teniendo en cuenta las tareas a realizar, las más significativas serían los desarrollos del *backend* y del *frontend*, además de completar las memorias correspondientes a cada una de las partes del proyecto. Por supuesto, podríamos añadir la gestión del tiempo de trabajo dedicado a reuniones y trabajo personal, además de las tareas que conllevan rellenar actas. Centrándonos en el reparto de la memoria, trabajamos todos en ella y procuramos distribuirnos los apartados a tratar de manera lo más equitativa posible. Para el desarrollo del *backend* vamos a dedicar dos miembros del grupo y para el *frontend*, dedicaremos dos al desarrollo web y dos al de la aplicación móvil. Puesto que optamos a una opción que incluye un *plugin* para navegador, una vez que terminemos todo el *frontend*, nos pondremos con esa tarea. En cuanto al resto de tareas posibles a realizar, procuramos rellenar el control de esfuerzos de manera individual, y pese a que el encargado de actualizar nuestra *Wiki* de *GitHub* es Javier Vela, contribuimos todos a completarla.

De la resolución de disputas se encarga nuestro compañero Jorge Bernal, quien actúa como mediador cuando existen opiniones opuestas relativas a la implementación de las diferentes partes del trabajo, ya bien sean de código o completar documentación. Resaltar que puesto que somos un grupo muy compenetrado, la labor de gestionar el equipo no es realmente complicada. El éxito de nuestro proyecto se basa en la paciencia, el respeto por los compañeros de equipo y la alta implicación por querer aprender acerca de aspectos que no dominamos.

Centrándonos más en la parte de monitorización y estado del proyecto, la gestión de estos aspectos se podría decir que es semanal, puesto que va sincronizada con las reuniones semanales que, decidiendo por mayoría, quedamos en que tuviesen lugar todos los martes de 12:00 a 14:00. De este modo, estas dos horas nos sirven para poner encima de la mesa todo el trabajo realizado durante la semana, de tal manera que todos los miembros del grupo nos enteramos de como van avanzando las diferentes partes del proyecto y, en función del plan de trabajo a completar establecido la semana anterior, se toman decisiones del tiempo a invertir para terminar el trabajo atrasado. Por otra parte, fijándonos en la carga de trabajo propuesta para la semana anterior y el porcentaje del mismo que hemos completado, proponemos un plan de trabajo a completar para el martes de la semana siguiente. Estimamos el poder completar cada semana un 10 % del trabajo total.

La entrega de resultados se hará de manera continua de tal forma que el cliente pueda ir viendo

el progreso del proyecto. Nos comprometemos a entregar los resultados del mismo en los plazos preestablecidos. Los resultados finales, que contendrán los códigos fuente, *scripts* de ejecución automática, etc. serán entregados al cliente también.

3.1.3. Procesos técnicos

Para implementar las vistas del sistema se utilizará por una parte Java (Android SDK) para construir el frontend para móviles Android y por otra JavaScript (React.JS) que corresponderá con los distintos navegadores web. En cuanto al backend, se desarrollará utilizando Node.JS y la framework Express. La base de datos utilizará el SGBD MongoDB. El despliegue del sistema se realizará en un entorno de contenedores Docker. Finalmente se ejecutarán unos scripts que llevarán a cabo las pruebas necesarias de las distintas funcionalidades para la comprobación del funcionamiento correcto.

3.2. Planes

3.2.1. Plan de gestión de configuraciones

A continuación se detallan los planes establecidos para la gestión continua de las configuraciones del proyecto.

Para asegurar la correcta comprensión del código y la navegabilidad del mismo se establece una convención de nombrado de archivos, una estructura de directorios y guías de estilo a seguir a los largo de todos los módulos del proyecto.

→Nombrado de archivos:

Todos los archivos del proyecto quedarán nombrados con el siguiente formato:

$< A >< B >< C >< D >$

A = Nombre identificativo principal del archivo, el más identificativo. B = Conjunto de nombres auxiliares opcionales para mejorar la identificación. C= “Subextensión” opcional para marcar el directorio padre y a su vez tipo de función. D = Extensión del archivo.

En estos campos solo se permiten cadenas de texto que cumplan la siguiente ER para evitar el uso de caracteres especiales: [a-zA-z]+[0-9]*. Se intentará además el uso en la medida de lo posible de nombres anglosajones. Todos los nombres comenzarán por una letra mayúscula. (Campos A y B).

Ejemplos:

Users.controller.js : Nombre de un archivo en subdirectorio controllers.

FeedUsers.css : Nombre de un archivo css para componente de feed de usuarios.

→ Estructura de directorios:

Todos los módulos seguirán una estructura de directorios de agrupamiento por tipo de fichero. Es decir, los ficheros quedarán agrupados bajo un directorio padre que indique su uso dentro del módulo o funcionalidad.

Ejemplo de estructura en back-end:

src — app.js #App entry point ———— config #Environment variables and configuration related stuff ————-¿Db.config.js ———— models #Database models ————-¿Users.model.js ————-¿Admins.model.js — ...

→ Guías de estilo:

-Estándares de código para desarrollo android: AOSP Java Code Style for Contributors. -Estándares de código para desarrollo en React: React design principles

Para el control del versionado y actualización del código se utilizarán distintos repositorios de github para los módulos del sistema, es decir, un repositorio de front-end web, de app móvil, de back-end y de documentación del proyecto. A continuación se enumeran los procedimientos a seguir para el uso de estos repositorios.

Dichos repositorios serán privados sólo accesibles por el equipo de desarrollo. Se asociará a cada repositorio un conjunto de de "GitHub Actions" para administrar la compilación y puesta en marcha del código. Los commits a estos repositorios irán acompañados de un nombre descriptivo de la tarea asociada. Los commits podrán hacerse en cualquier momento siempre que se haya probado el código previamente de manera local y ateniéndose al estado de las máquinas de despliegue. Se seguirá una filosofía de entrega continua y despliegue continuo. Semanalmente se revisarán los commits realizados para evaluar el progreso de las tareas.

3.2.2. Plan de construcción y despliegue del software

El sistema se fundamenta en el desarrollo de cuatro subsistemas que trabajan aislados de los otros, siendo su comunicación entre los módulos expuesta a través de interfaces que los relacionan, de este modo tanto las pruebas, compilación y dependencias son independientes al resto de subsistemas.

Dos de los cuatro subsistemas, el frontend web y el backend, estarán empotrados en contenedores docker y desplegados en la nube utilizando los servicios de *Amazon Web Services*, se utilizarán scripts de *GitHub actions* para automatizar la compilación y testing y despliegue en AWS de los mismos, utilizando en el caso de entorno web librerías de testing como Jest y Postman en el caso de la *API REST*.

Para el subsistema que concierne a la capa de datos, se despliega en un cluster de MongoDB situado en Bélgica siendo este proporcionado por el equipo de *MongoDB Atlas*, donde se van a desarrollar una serie de diversos triggers que lleven un control de la consistencia de datos tanto tras el uso operaciones además se llevará un control periodico para evitar la replicación y la detección de anomalías a través del uso de scripts.

La interfaz móvil llevará por su cuenta la compilación y testing para cada versión funcional de la aplicación, se integrarán test de unidad, funcionalidad y sobrecarga del sistema.

El punto fuerte de utilizar el despliegue basado en contenedores es que el control de dependencias es indiferente al sistema operativo y al entorno de desarrollo de los programadores, de esta forma cada integrante del equipo podrá configurar y personalizar su entorno por cuenta propia. En el fronted móvil al ser independiente se fija una serie de requisitos para el equipo que trabaja en esta parte, se usará la versión Android 6.1 (*Marshmallow*) y se usará Java como lenguaje de

programación, el control de dependencias es llevado por el propio Gradel de Android.

La configuración base de los subsistema será la siguiente, las interfaces del frontend web y el backend están expuestas en el puerto 80, el modelo de capa de datos proporciona una uri externa con la que acceder a la base de datos. las variables de usuarios y contraseñas serán almacenadas como variables de entorno para evitar ponerlas como texto plano en el código.

3.2.3. Plan de aseguramiento de calidad

3.2.4. Calendario del proyecto y división del trabajo

4. Análisis y diseño del sistema

4.1. Análisis de requisitos

Código	Descripción
RF-1	El sistema permite almacenar contraseñas.
RF-1.1	El sistema permite almacenar pares que constan de nombre de usuario y contraseña.
RF-1.2	El sistema permite asociar a las contraseñas una URL del sitio web al que corresponden.
RF-1.3	El sistema registra la fecha de creación y actualización de la contraseña.
RF-1.4	El sistema permite almacenar una descripción de texto asociada a la contraseña.
RF-1.5	El sistema permite almacenar ficheros de imagen (jpeg, png, ...) y ficheros PDF, asociados a la contraseña.
RF-2	El sistema permite organizar las contraseñas por categorías, fecha de creación y actualización.
RF-3	El sistema permite realizar una búsqueda entre las contraseñas por categoría, nombre de usuario.
RF-4	El sistema permite generación de contraseñas pseudoaleatorias.
RF-4.1	El sistema permite seleccionar la longitud de la contraseña a generar.
RF-4.2	El sistema permite seleccionar el conjunto de caracteres que compone la contraseña a generar.
RF-4.3	El sistema mostrará el grado de robustez de la contraseña al ser generada.
RF-5	El sistema permite al usuario acceder a sus contraseñas únicamente a través de la contraseña maestra.
RF-6	El sistema requiere 2FA para iniciar sesión desde un dispositivo nuevo, distinto a los utilizados con anterioridad.
RF-7	El usuario se registra en el sistema mediante un correo electrónico y una contraseña maestra.
RF-7.1	El registro de sesión se deberá confirmar, para verificar la identidad, mediante un correo al usuario registrado.
RF-8	El sistema permite la actualización de las contraseñas.
RF-9	Se accede al sistema mediante una aplicación móvil.
RF-10	Se accede al sistema mediante una interfaz web.
RF-10	El sistema ofrece un *plug-in* para navegador web.