

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
2. Organización del proyecto	2
3. Plan de gestión del proyecto	2
3.1. Procesos	2
3.1.1. Procesos de inicio	2
3.1.2. Procesos de ejecución y control	3
3.1.3. Procesos técnicos	4
3.2. Planes	4
3.2.1. Plan de gestión de configuraciones	4
3.2.2. Plan de construcción y despliegue del software	5
3.2.3. Plan de aseguramiento de calidad	6
3.2.4. Calendario del proyecto y división del trabajo	8
4. Análisis y diseño del sistema	9
4.1. Análisis de requisitos	9
4.2. Diseño del sistema	10
5. Memoria del proyecto	15
5.1. Inicio del proyecto	15
5.2. Ejecución y control del proyecto	15
5.3. Cierre del proyecto	15
6. Conclusiones	15
Glosario	15
Anexo I. Actas de todas las reuniones realizadas	15
7. Conclusiones	15
8. Bibliografía	15

1. Introducción



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, de acuerdo con las capacidades y conocimientos individuales. Además, se ha designado como director del proyecto a Arturo Calvera, sobre el cual recae la gestión de los equipos de trabajo y de las responsabilidades de los mismos.

■ 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 modificar dicha división en equipos para adaptarse a las necesidades que surjan durante el desarrollo del proyecto.

3. Plan de gestión del proyecto



Procesos

3.1.1. Procesos de inicio

Para la identificación y asignación de recursos a utilizar, se han puesto en común los conocimientos tecnológicos de cada integrante del grupo realizando así un sondeo de las opciones disponibles, el grado de familiarización con las tecnologías y las preferencias individuales de los desarrolladores.

Respecto a los servidores en *cloud* a utilizar para el despliegue del sistema, se barajaron los servicios de *cloud* Amazon Web Services y Microsoft Azure. El criterio de elección se ha basado en el estudio del impacto a nivel global de la empresa, es decir, grado de utilización en el mundo empresarial y la cantidad de servicios ofrecidos gratuitamente. Finalmente, teniendo en cuenta los criterios anteriores, se decide crear una cuenta en AWS, siendo así Amazon el proveedor de *cloud* para el proyecto.

En relación con la base de datos, se ha decidido usar una base no relacional debido al grado de flexibilidad de modelo de datos que ofrece frente al modelo típico relacional. Como SGBD se ha escogido MongoDB. Se ha creado una cuenta en MongoDB Atlas el cual permite crear una base de datos remota accesible por el sistema.

En cuanto a la formación inicial de los integrantes, se han tenido en cuenta en el reparto de responsabilidades los conocimientos de los desarrolladores, sin embargo, todos los desarrolladores dedicarán tiempo de manera individual en autoformación con tutoriales y documentación sobre las tecnologías a usar en el desarrollo del sistema. Además, todos los integrantes del grupo se han comprometido a formar a sus compañeros en las áreas que conozcan a medida que surjan dudas durante el desarrollo del sistema.

3.1.2. Procesos de ejecución y control

Las comunicaciones entre los miembros del grupo de trabajo se realizarán principalmente a través de la herramienta de comunicación Whatsapp, mediante la cual se concretan los horarios de trabajo y se informa 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, las reuniones semanales de puesta en común del trabajo se realizarán utilizando la herramienta de videoconferencia Google Meet. Esta última será el principal medio de comunicación del grupo. Destacar que pese a la división de trabajo previa, se intenta trabajar de manera conjunta por videoconferencia, de manera que se puedan poner en común y resolver problemas en la medida en que surjan.

En cuanto al registro de las decisiones tomadas en las reuniones, se plasmarán en actas gestionadas por Javier Vela. El almacenamiento de estas actas se realizará en un directorio remoto en GitHub.

Partiendo de la división inicial del grupo en equipos de trabajo detallada en el punto 2, los responsables de cada equipo de trabajo se encargarán de monitorizar el avance de sus respectivos módulos, determinar las tareas a realizar, asignar dichas tareas a los desarrolladores y marcar límites temporales y prioridades para cada tarea. En cuanto al trabajo de documentación de los avances sobre el código, todos los desarrolladores serán responsables de registrar los mismos en las *wikis* de GitHub habilitadas para ello en cada repositorio de código. Así mismo, los desarrolladores tendrán que rellenar una tabla de control de esfuerzos para registrar su trabajo y cumplir con el resto de planes especificados más adelante.

De la resolución de disputas se encargará el miembro Jorge Bernal, quien actuará como mediador cuando surjan conflictos en cualquier ámbito del desarrollo del proyecto, pudiendo acudir a este en cualquier momento si se necesitase de su ayuda.

Respecto a la monitorización y control del progreso del proyecto, todas las semanas se realizará una reunión conjunta de control en una hora y día fijadas a la cual asistirán todos los miembros del grupo para exponer los avances realizados durante la semana y poder determinarse el estado del proyecto y las siguientes tareas a desarrollar. Así mismo, tal y como se detalla en el punto 3.2.3, existen mecanismos para asegurar la calidad del producto y detectar posibles de rendimiento en el completado de las tareas.

La entrega de resultados se hará de manera continua de tal forma que el cliente pueda ir viendo el progreso del proyecto. El equipo se compromete a entregar los resultados del mismo en los plazos preestablecidos. Los resultados finales, que contendrán los códigos fuente, *scripts* de compilación y despliegue, etc. también serán entregados al cliente.

3.1.3. Procesos técnicos

Para implementar las vistas del sistema se utilizará por una parte Java (Android SDK) para construir el cliente móvil para dispositivos Android y por otra JavaScript (React.JS) para desarrollar el *frontend* 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 desplegado sobre máquinas virtuales en *cloud*. Se seguirá una filosofía de entrega y despliegue continuo utilizando *scripts* y GitHub Actions.

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 lo largo de todos los módulos del proyecto.

Nombrado de archivos

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

$\langle A \rangle \langle B \rangle \langle C \rangle \langle D \rangle$

- 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 de directorios en *backend*:

```
src
├── app.js
├── config
│   └── Db.config.js
├── models
│   ├── Users.model.js
│   └── Admins.model.js
```

Guías de estilo

Los desarrolladores de código se apoyarán en las siguientes 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*.
- Estándares de código para desarrollo en JavaScript: *JavaScript Standard Style*.

Para el control del versionado y actualización del código se utilizarán distintos repositorios de GitHub para los componentes del sistema, es decir, un repositorio de *frontend* web, de app móvil, de *backend* y de documentación del proyecto. A continuación se enumeran los procedimientos a seguir para el uso de estos repositorios.

Dichos repositorios serán privados solo accesibles por el equipo de desarrollo. Se asociará a cada repositorio un conjunto de GitHub Actions para administrar la compilación y puesta en marcha del sistema. 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. Para comprobar el progreso de las tareas, se seguirá el plan de aseguramiento de la calidad presentado en el punto 3.2.3.



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 clúster 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 periódico 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 *frontend* 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 Gradle de Android.

La configuración base de los subsistemas 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

A continuación se presentan un conjunto de actividades de control de calidad del código a llevar a cabo por parte del equipo de desarrollo:

Uso de guías de documentación y de diseño gráfico

Los desarrolladores del proyecto están alentados a utilizar las siguientes guías en el proceso de diseño, documentación y desarrollo del *software*.

- Principios de diseño de las aplicaciones para dispositivos móviles de Google
- *User Interface & Navigation guide* de Android.
- *Google's web fundamentals*.
- *Firefox OS guidelines*.
- Todas las guías anteriormente mencionadas en el punto 3.2.2

Revisión del código por pares

Esta actividad tiene por objetivo realizar revisiones periódicas del código generado por miembros del equipo que no lo han desarrollado, pero que tienen las capacidades técnicas para realizar críticas constructivas sobre este. Se realizarán revisiones por pares abiertas, es decir, revisor y desarrollador se conocen y pueden comunicarse durante el proceso de revisión. Como apoyo a estas revisiones del código, los desarrolladores rellenarán una tabla de versionado del código, la cual permitirá al revisor comprobar los avances realizados en la tarea.

Se rellenará una tabla de versionado (figura 1) por cada módulo importante de software (especificados en el punto anterior), la responsabilidad de rellenar dicha tabla recae sobre el desarrollador del módulo en cuestión. Cuando el desarrollador termine una versión del módulo importante o realice un avance crítico solicitará una revisión al gestor del proyecto y este le asignará un revisor adecuado para la tarea. Las conclusiones de una revisión se guardarán en una tabla de resumen (figura 2) para que sirvan de referencia para futuras revisiones y para que el desarrollador pueda corregir lo necesario.

Módulo	Versión	Fecha	Desarrollador	Descripción del trabajo desarrollado
Log-in	1.3	02/06/2021	Jaime García	Nueva interfaz de log-in

Figura 1: Tabla de versionado del código.

Módulo	Versión	Fecha	Revisor	Conclusiones/Correcciones
Log-in	1.3	02/06/2021	Carlos Puente	Nueva interfaz de log-in

Figura 2: Tabla de conclusiones de revisión por pares de código.

Revisión de los requisitos

Para comprobar el correcto cumplimiento de los requisitos del sistema se utilizará una tabla de cumplimiento de requisitos (figura 3). En este tipo de revisión tanto el desarrollador como el revisor rellenarán dicha tabla y discutirán los resultados. Dichos resultados serán recogidos en una tabla de resumen (figura 4) que servirá como para futuras revisiones. Los módulos de software a desarrollar anteriormente van asociados a ciertos requisitos del sistema. Antes de comenzar el desarrollo de un módulo se concretarán los requisitos finales a los que hace referencia y se generará la plantilla de la tabla de adecuación de requisitos. Como en el modelo de revisión anterior, cuando el desarrollador del módulo termine una versión importante o realice un avance crítico solicitará una revisión al gestor del proyecto este le asignará un revisor adecuado para la tarea.

Módulo y Versión	Desarrollador/Revisor			
Log-in	Carlos Puente	CUMPLIMIENTO	REQUISITOS	COMENTARIOS
			15 RF1	No envía 2FA
			20 RF2	Cumple perfecto
			5 RF3	Apenas empezado
0 = No cumple el requisito				
10 = Cumple el requisito parcialmente				
20 = Cumple el requisito completamente				

Figura 3: Tabla de cumplimiento de requisitos.

Módulo	Versión	Fecha	Revisor	Nota media de cumplimiento	Conclusiones/Correcciones
Log-in	1.3	02/06/2021	Carlos Puente	13,3	Nueva interfaz de log-in
0 = No cumple con ningún requisito					
10 = Cumple los requisitos parcialmente					
20 = Cumple con todos los requisitos					

Figura 4: Tabla de conclusiones de revisión de cumplimiento de requisitos.

Todas estas tablas generadas por el proceso de aseguramiento de la calidad quedarán almacenadas en un directorio remoto accesible por todos los miembros del equipo de desarrollo agrupadas por el módulo al que hacen referencia de manera que sirvan como histórico de las revisiones realizadas y como mecanismo de monitorización del progreso de las tareas.

Test sobre el producto

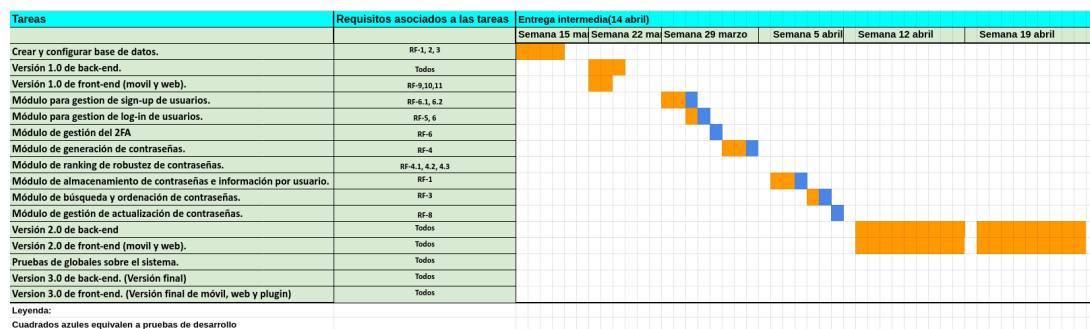
Para asegurar el funcionamiento correcto del código desarrollado se realizarán pruebas de desarrollo sobre los módulos, las cuales consistirán en pruebas unitarias y pruebas de integración con el resto de componentes del sistema. Además, una vez el sistema sea funcional, se realizarán pruebas globales sobre el sistema, las cuales consistirán en pruebas funcionales, de comunicación, de rendimiento, de sobrecarga... etc. Por último, utilizando las herramientas de despliegue continuo mencionadas anteriormente, se implementarán tests básicos sobre el sistema previos al despliegue.



4. Calendario del proyecto y división del trabajo










La división del trabajo se consiste en la partición del trabajo en los siguientes módulos:

Módulo de gestión de *sign-ups* de usuarios, módulo de gestión de log-ins de usuarios, módulo de gestión del Two-Factor-Authentication, módulo de generación de contraseñas, módulo de ranking de robustez de contraseñas, módulo de almacenamiento de contraseñas e información por usuario, módulo de búsqueda y ordenación de contraseñas y módulo de gestión y actualización de contraseñas e información. La documentación será del proyecto será actualizada por cada grupo de trabajo usando las *Wikis* de GitHub después de cada sesión de trabajo, del tal manera que los avances realizados en el desarrollo queden registrados haciendo posible el seguimiento del desarrollo al resto del equipo. El diseño gráfico será realizado por todos los miembros del equipo mediante la creación de un *mockup* de la misma en una reunión conjunta. Las instalaciones y los despliegues serán automáticos y continuos mediante *scripts* de GitHub Actions. Las pruebas (Pruebas de desarrollo sobre los módulos y pruebas globales sobre el sistema) se realizarán conforme los desarrolladores vayan terminando sus designaciones y serán los mismos desarrolladores los cuales pasarán las pruebas, en el punto 3.2.3 se explica en detalle este aspecto.



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	Las entradas de contraseñas tienen un nombre asociado.
RF-1.3	El sistema permite asociar a las contraseñas una URL del sitio web al que corresponden.
RF-1.4	El sistema registra la fecha de creación y actualización de la contraseña.
RF-1.5	El sistema permite almacenar una descripción de texto asociada a la contraseña.
RF-1.6	El sistema permite almacenar ficheros de imagen (jpeg, png, ) y ficheros PDF, asociados a la contraseña.
RF-1.7	El sistema asocia cada contraseña a una  categoría del usuario
RF-1.8	El usuario puede crear, renombrar y eliminar categorías
RF-2	El sistema permite ordenar las contraseñas por categorías, fecha de creación y actualización.
RF-2.1	El sistema permite la visualización de las entradas de contraseña y todos sus datos y ficheros asociados.
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 usuario inicia sesión al sistema mediante su correo electrónico y la contraseña maestra. Únicamente podrá acceder con esta contraseña, que no se puede recuperar.
RF-6	El sistema requiere 2FA para iniciar sesión desde un dispositivo nuevo, distinto a los utilizados con anterioridad.
RF-6.1	El sistema permite el registro de un usuario con: correo electrónico, contraseña maestra,  nombre, apellidos.
RF-6.2	El registro de sesión se deberá confirmar, para verificar la identidad, mediante un correo al usuario registrado.
RF-7	La interfaz de usuario permite copiar al  apapeles del dispositivo la contraseña de una entrada.
RF-8	 sistema permite la modificación de todos los campos de las contraseñas.
RF-9	Se accede al sistema mediante una  cación móvil.
RF-10	Se accede al sistema mediante una  rfaz web.
RF-11	El sistema ofrece un <i>plug-in</i> para  egador web.

4.2. Diseño del sistema

Diagramas de módulos

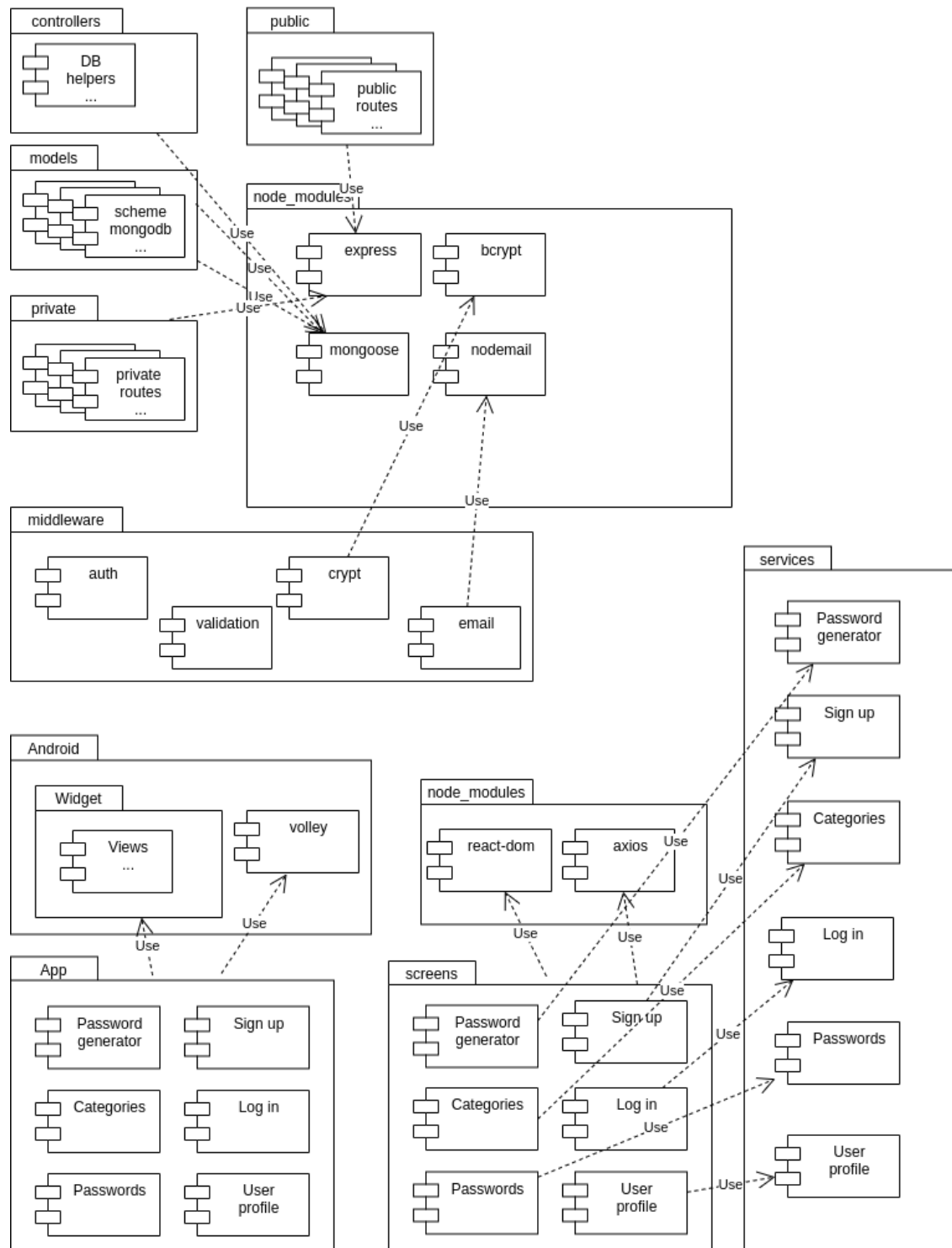


Figura 5: Diagramas de componentes y conectores.

En el diagrama de módulos en la figura 5 se describe las estructuras que el sistema implementa y sus relaciones. Los diferentes paquetes agrupan distintos módulos basándose en su función.

- 'public': las rutas del modelo accesibles públicamente a través de la API.
- 'private': las rutas del modelo que requieren permisos.
- 'models': MongoDB *schemes* que modelan la estructura de la información en la base de datos.
- 'controllers': colección de módulos que interactúan con la base de datos.
- 'middleware': módulos que otorgan funcionalidad variada al sistema.
- 'node modules': módulos importados con NPM que ofrecen funcionalidad específica.
- 'express': *framework* de desarrollo de servidores web para Node.
- 'mongoose': permite la interacción con bases de datos MongoDB.
- 'bcrypt': ofrece funcionalidad de cifrado.
- 'nodemail': ofrece funcionalidad de envío de correos electrónicos.
- 'react-dom': interacción con la vista del navegador web.
- 'axios': permite realizar peticiones HTTP al *backend* de la aplicación.
- 'screens': vistas del servidor web.
- 'services': diferente funcionalidad para la vista de la aplicación.
- 'Android': SDK de Android.
- 'App': diferentes vistas de la aplicación móvil.

Los diferentes módulos externos importados del repositorio de Node o del SDK de Android, ofrecen distinta funcionalidad relativa a las conexiones entre componentes (*e.g. frontend-backend, backend-MongoDB*), componentes de vista básicos y seguridad.

Diagramas de componentes y conectores

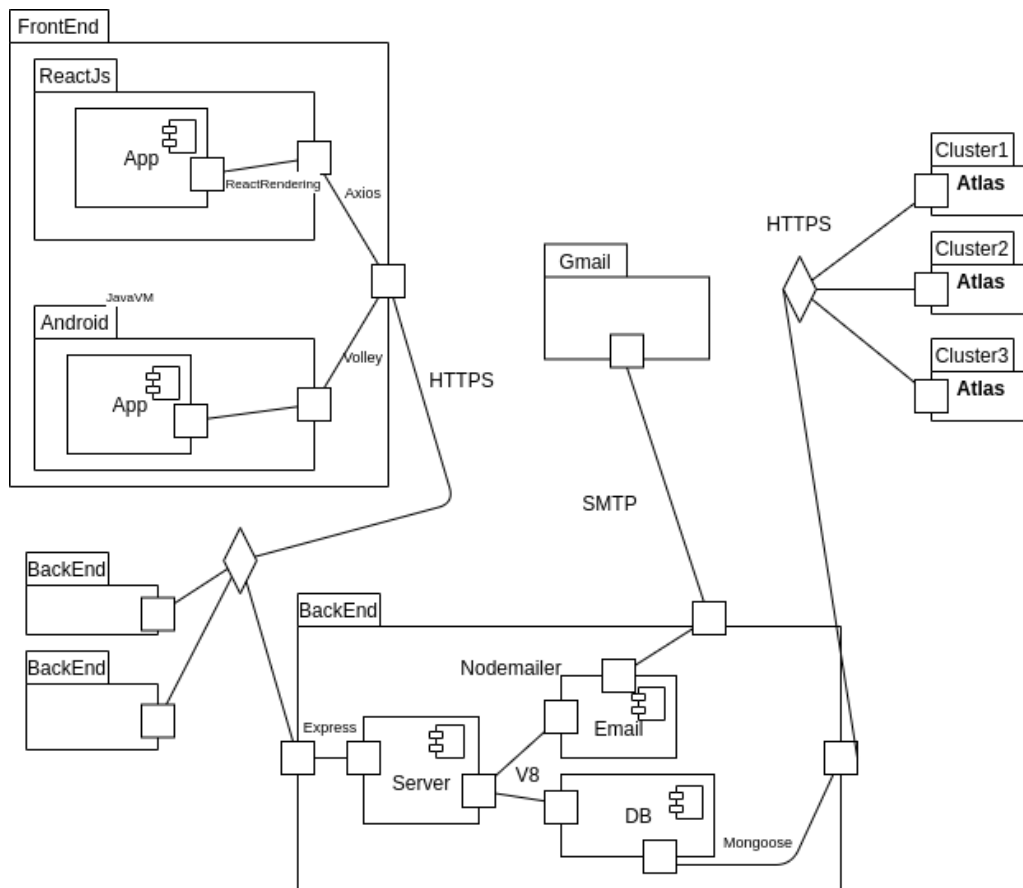


Figura 6: Diagramas de componentes y conectores.

En el diagrama de componentes y conectores en la figura 6 se muestra una visión general del sistema, existen tres componentes principales en la aplicación que se relaciona cada uno con una capa del modelo, estos componentes muestran los elementos de presencia en tiempo de ejecución del sistema.

El *frontend* es el componente de la aplicación con el cual el usuario interactúa con el sistema, la aplicación al ser orientada a varias plataformas usa distintos subcomponentes para darle forma a la aplicación, en concreto React.JS, se comunica con el resto de la aplicación mediante el conector React Rendering, De la misma forma lo hace la parte de Android mediante Java Virtual Machine, ambos interactúan con el resto del sistema usando los puertos que proporcionan las librerías de Axios o Volley.

El conector que relaciona este componente con el componente del *backend* es el protocolo HTTPS a una réplica del servidor.

La interacción entre los componentes de *backend*, es en este caso más compleja el puerto que atiende peticiones utiliza Express, y se comunica con los otros componentes usando el motor V8 como conector, que compila en tiempo de ejecución todos los módulos de JavaScript de la parte del servidor. El sistema a su vez tiene un servidor mail que se comunica mediante el protocolo SMTP y que usa la librería Nodemailer para conectar con Gmail. Por último mantendrá una conexión, también usando el protocolo HTTPS, con la base de datos que se encontrará replicada en varios clústeres y alojada en *MongoDB Atlas*, se usa Mongoose para gestionar la conexión con la base de

datos.

Diagrama de distribución

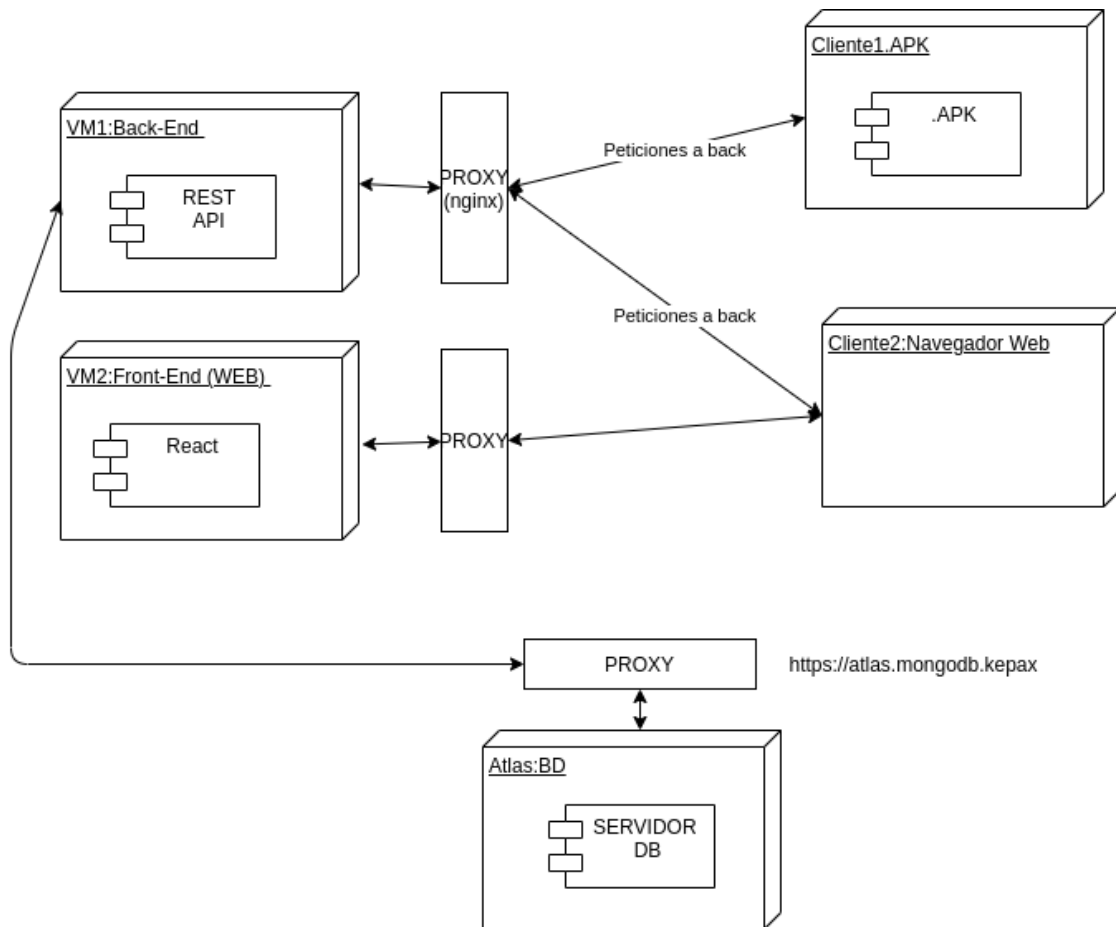


Figura 7: Diagrama de distribución.

En el diagrama de despliegue de la figura 7, se muestra una vista general del sistema en funcionamiento y cómo está distribuido en la nube, en esta vista se asigna cada componente del sistema a un hardware específico.

Tanto el servidor web como el servidor API, se sitúan alojados en los servicios de Amazon AWS y la base de datos se encuentra replicada utilizando MongoDB Atlas.

Se diferencia dos tipos de cliente, el que accede desde un dispositivo móvil y el que accede mediante un protocolo web. El cliente móvil ya tiene compilado el sistema en un dispositivo Android por lo que la capa de acceso a la aplicación este solo tendrá que realizar peticiones directamente a la API REST, a diferencia que el cliente que se conecta mediante un protocolo web, que tendrá en primer lugar que obtener el *template* desde un servidor web para poder realizar peticiones a la API REST.

El servidor cuando requiera realizar operaciones sobre datos persistentes, opera conectándose a un clúster con una réplica de la base de datos MongoDB.

Patrones de diseño y estilos arquitecturales

A diferencia de la mayoría de *frameworks* que usan el patrón arquitectural modelo vista controlador para separar la lógica de la aplicación de la lógica de vistas, React.JS al ser un *framework* moderno que ofrece *server-side-rendering* por lo que no se puede considerar una arquitectura MVC.

La arquitectura usada (figura 8) consiste en tres capas con interfaz web, capa de presentación, capa de negocio y capa datos.

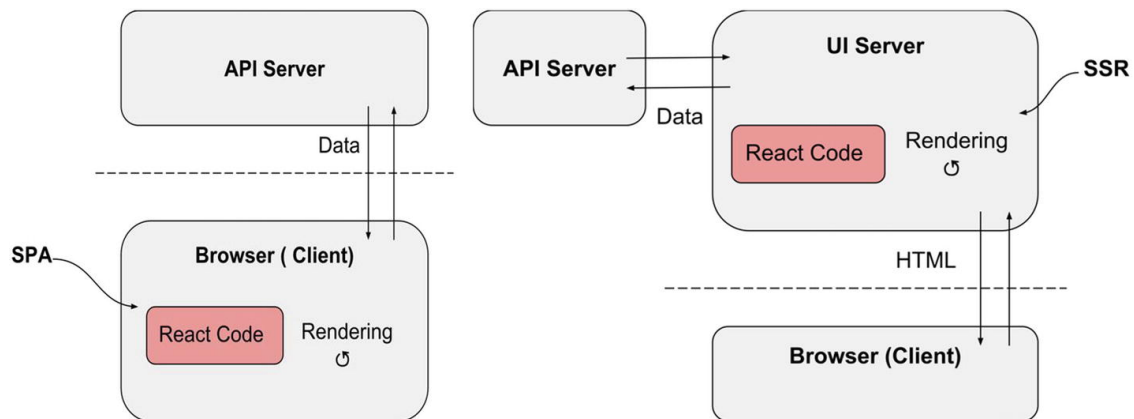


Figura 8: Modelo tres capas.

Como se viene diciendo la capa de presentación o capa de usuario es la que muestra el sistema, le comunica y captura información a la capa de negocio, en esta capa se presentan dos tecnologías Android y React.JS.

La capa de negocio es donde se ejecutan las peticiones del usuario y se envían las respuestas de la ejecución, esta capa se comunica con la capa de presentación para atender a las peticiones y con la capa de datos para almacenar o recuperar datos. Para esta capa se utilizará el entorno de ejecución Node.

La capa de datos es la encargada de almacenar los datos mediante en gestor de base de datos, en el caso de la aplicación se utiliza la base de datos no relacional MongoDB.

Tecnologías elegidas

Se hará uso de lenguajes de programación como JavaScript para la interfaz web (React) y para el modelo (Node). Es la tecnología requerida por el *stack* de lenguajes para aplicaciones web MERN. Es sencillo, debido a su conocimiento previo, y unifica la tecnología para el desarrollo *backend* y *frontend*.

La aplicación móvil se desarrollará en Java (Android), que permite desarrollo nativo para teléfonos, es ampliamente utilizado y se conoce de proyectos anteriores. Se propuso el uso del entorno React Native, pero se descartó debido a su desconocimiento por parte del grupo.

El componente la vista de la aplicación se conecta a la API de del modelo para realizar peticiones relativas a la acción del usuario. Esta segunda conecta con la base de datos MongoDB en los servidores Atlas de Mongo. Se prefiere el uso de Atlas frente al despliegue de la base de datos en una máquina virtual debido a su facilidad de administración y la disponibilidad ofrecida.

La base de datos MongoDB es NoSQL y, frente a una base de datos SQL como Oracle, permite más flexibilidad para recoger y persistir información. Además, MongoDB junto a Mongoose, ofrecen facilidad de interacción con JavaScript mediante ficheros JSON.

Para la creación de la API web ofrecida por servidor del modelo, se ha optado por seguir la estructura API REST, que es ampliamente utilizada y fácil de implementar con las tecnologías utilizadas.

La tecnología Node hace uso de un bucle de eventos el cual permiten la realización de operaciones asíncronas para realizar peticiones en el *background* y recibir respuestas sin necesidad de interrumpir la operativa esperando a esta.

Se ha optado en utilizar el módulo Bcrypt para cifrar los datos de la aplicación y utilizar el protocolo HTTPS para asegurar las comunicaciones.

5. Memoria del proyecto

5.1. Inicio del proyecto

5.2. Ejecución y control del proyecto

5.3. Cierre del proyecto

6. Conclusiones

Glosario

Anexo I. Actas de todas las reuniones realizadas

7. Conclusiones

8. Bibliografía