**Control de Cambios**

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Fecha** | **Estado** | **Autor** |
| 1.0 | 10/06/17 | Creación | Wilson Julca Mejía |

1. **ESTRUCTURA DE LIBRERÍAS EN GENERAL**

GrupoRPP

Proyecto 2

A

I

N

N

I

PGC

PGCA

Si se modifica interviene el comité de control de cambios

AINNI

Gestión

Negocio

Análisis

Implementación

AINNI

Documentos

Documentos

Codificación

Diseño

Análisis

Clientes (Reléase)

Trabajo (Desarrollo)

Librería principal (aceptados)

PGC=Plan de gestión de la configuración

PGCA=Plan de gestión de la configuración de cambios

1. **ESTRUCTURA DE LIBRERÍAS ESPECÍFICA**

**LIBRERÍA PRINCIPAL:**

**WEB:**

Será desarrollado en lenguaje Python, utilizando el framework Django, que nos servirá como marco base de trabajo para el desarrollo del proyecto. Unas de las principales características del framework antes mencionado, es la facilidad para usar librerías, la facilidad de gestionar las versiones de cada librería, y hasta la facilidad de instalación de las mismas.

En Django, las librerías se suelen subir a un repositorio centralizado que se gestionan con PIP, el foro y comunidad de desarrolladores de Python, se encarga del mantenimiento, gestión y control de las mismas.

En la estructura del proyecto que nos provee el framework, se encuentra un archivo llamado “requirements.txt”, en el cual se procede a instalar los paquetes que usaremos en nuestro proyecto.

**MÓVIL iOS:**

Será desarrollado en lenguaje Swift3, utilizando la arquitectura MVC que XCode genera.

En iOS, las librerías se suelen subir a un repositorio centralizado que se gestionan con POD, el foro y comunidad de desarrolladores de Swift3, se encarga del mantenimiento, gestión y control de las mismas.

En la estructura del proyecto que nos provee el framework, se encuentra un archivo llamado Podfile en el cual se procede a instalar los paquetes que usaremos en nuestro proyecto.

**RESPONSABLES:**

La responsabilidad recae en el Jefe de desarrollo y los programadores, quienes son que van a interactuar en directo con las mismas.

**ACTIVIDADES:**

Las actividades principales en cuanto a la gestión de los paquetes, son sobre todo el control de las versiones, puesto que las gemas (como cualquier librería), tienden a actualizarse cada cierto tiempo, ya sea por errores encontrados o por nuevas funcionalidades que se desea tener. Los roles y el tipo de acceso a cada rol son presentadas en la Tabla 1.

|  |  |
| --- | --- |
| **ROL** | **TIPO DE ACCESO** |
| Gerente del proyecto | * Supervisar * Leer |
| Jefe de desarrollo | * Leer * Supervisar * Escribir * Ejecutar |
| Desarrolladores | * Leer * Ejecutar * Escribir |
| Arquitecto de Software | * Supervisar |

Tabla 1. Roles y Tipos de Acceso

**LIBRERÍAS WEB CONTROLADAS:**

A continuación listamos algunos paquetes que utilizaremos en el desarrollo del proyecto, que serán de gran ayuda en el despliegue de funcionalidades:

* **Django:** Paquete que tiene todas los módulos del framework web Django
* **Django Rest Framework:** Paquete que usaremos para realizar los servicios web siguiendo la filosofía de los formularios de django.
* **Django Debug Toolbar:** Paquete que sirve para mostrar la cantidad de queries, uso del CPU en cada página.
* **Schema Plus (schema\_plus):** Es una Gema que nos facilitará la gestión de llaves foráneas e índices en nuestra base de datos física. Dando paso a una gestión más fácil en la relación de modelos (base de datos a nivel lógico).
* **Pillow:** Paquete que sirve para el manejo de imágenes.
* **django-cryspi-forms:** Paquete sirve para la creación rápida de formularios.
* **psycopg2:** Paquete que sirve para la conexión con la base de datos PostgreSQL.

**LIBRERÍAS MÓVIL iOS CONTROLADAS:**

A continuación listamos algunos paquetes que utilizaremos en el desarrollo del proyecto, que serán de gran ayuda en el despliegue de funcionalidades:

* **Alamofire:** Paquete que permite el manejo de las solicitudes a los servicios REST
* **Alamofire-image:** Paquete que permite la descarga de imágenes asíncronamente.
* **RealmSwift:** Paquete que permite manejar la base de datos en Realm.
* **Schema Plus (schema\_plus):** Es una Gema que nos facilitará la gestión de llaves foráneas e índices en nuestra base de datos física. Dando paso a una gestión más fácil en la relación de modelos (base de datos a nivel lógico).
* **NVActivityIndicatorView:** Paquete que nos permite mostrar un loader mientras se consume los servicios REST o mientras se realiza una acción.