# Guia de programación Versión 1.0 Victor Rubia

En un principio Amanda se encargará del diseño Frontend, pasando cada boceto realizado por los compañeros a Flutter, mediante el uso de Flutterflow para su facilidad. Tendrá que crear un .dart por cada vista. Ángel se encargará de hacer el backend, pasando el diagrama de clases al modelo. Debe seguirse un patrón MVC mediante el uso de la libreria [mvc\_pattern](https://pub.dev/packages/mvc_pattern) detallada a lo largo del documento, disponiendo todos los getter y setter del modelo al controlador. El controlador se encargará también Ángel, poniendo a disposición de la vista, las funciones que necesite para mostrar los datos. Víctor estará entre medias de ambos para resolver problemas de interconexión entre el backend y el frontend, tomando parte de trabajo del backend y del frontend.

## Primeros pasos

Necesitamos tener Android Studio instalado, junto al plugin de Flutter (buscar en google cómo hacerlo). Descargar el sdk de flutter es obligatorio para completar esto.

## Descargamos el proyecto

Tras hacer git clone del repositorio, pasamos a la descarga de dependencias mediante flutter pub get. Para ello debes tener en el path la carpeta bin del sdk de flutter.

Abriremos el proyecto con Android Studio.

Cada vez que comencemos a programar haremos un git pull.

## Ejemplo útil del que poder fijarnos si tenemos alguna duda

Si tenemos alguna duda de cómo hacer alguna parte de la implementación podemos consultar un proyecto que ya se hizo por parte de Víctor en el siguiente [enlace](https://github.com/VictorRubia/DS_P4_Flutter), en el que se usa MVC y varias pantallas. El ejemplo trata de una aplicación de gestión de banca, como las que tenemos actualmente en nuestros teléfonos. Se necesita un login para entrar y una vez dentro se pueden visualizar los datos bancarios, como tarjetas, cuentas bancarias, movimientos y más.

## Base de datos y Servidor de Archivos

En el repositorio [DB](https://github.com/Asociacion-de-Parchis/DB), el cual debemos de hacer **git clone** [**git@github.com**](mailto:git@github.com)**:Asociacion-de-Parchis/DB.git** o git pull en caso de que ya lo tengamos clonado, se encuentran los archivos necesarios para montar localmente el servidor de archivos y bd mysql.

Basta con situarnos en la carpeta db y con el shell ejecutar el comando *docker compose up -d* **si se ejecuta por primera vez** o bien **si ya se ejecutó antes** *docker compose up*. Para instalar docker en ubuntu puedes seguir [estos pasos](https://docs.docker.com/engine/install/ubuntu/). Si tienes otro SO, puedes buscar en Google “Install Docker Engine” para tu SO.

Para poder **subir al servidor de archivos un fichero** se debe hacer mediante **POST**. Se puede usar **cURL** a modo de ejemplo de la siguiente forma:

curl -F file=@{rutaFichero} localhost:3000

Sustituyendo los corchetes y su contenido por la ruta del fichero local que queremos subir al servidor. Por defecto se ejecuta en localhost sobre el puerto 3000.

Para **comunicarnos con la base de datos** en Dart usaremos la libreria [mysql1](https://pub.dev/packages/mysql1). En su página vienen ejemplos de uso.

Los datos que hemos de rellenar para acceder a PhPMyAdming son

| User | Password |
| --- | --- |
| admin | admin |

**PhPMyAdmin** se ejecuta sobre el puerto **8081**. **MySQL** se ejecuta sobre el puerto **6033**.

## Guías de estilo para la programación

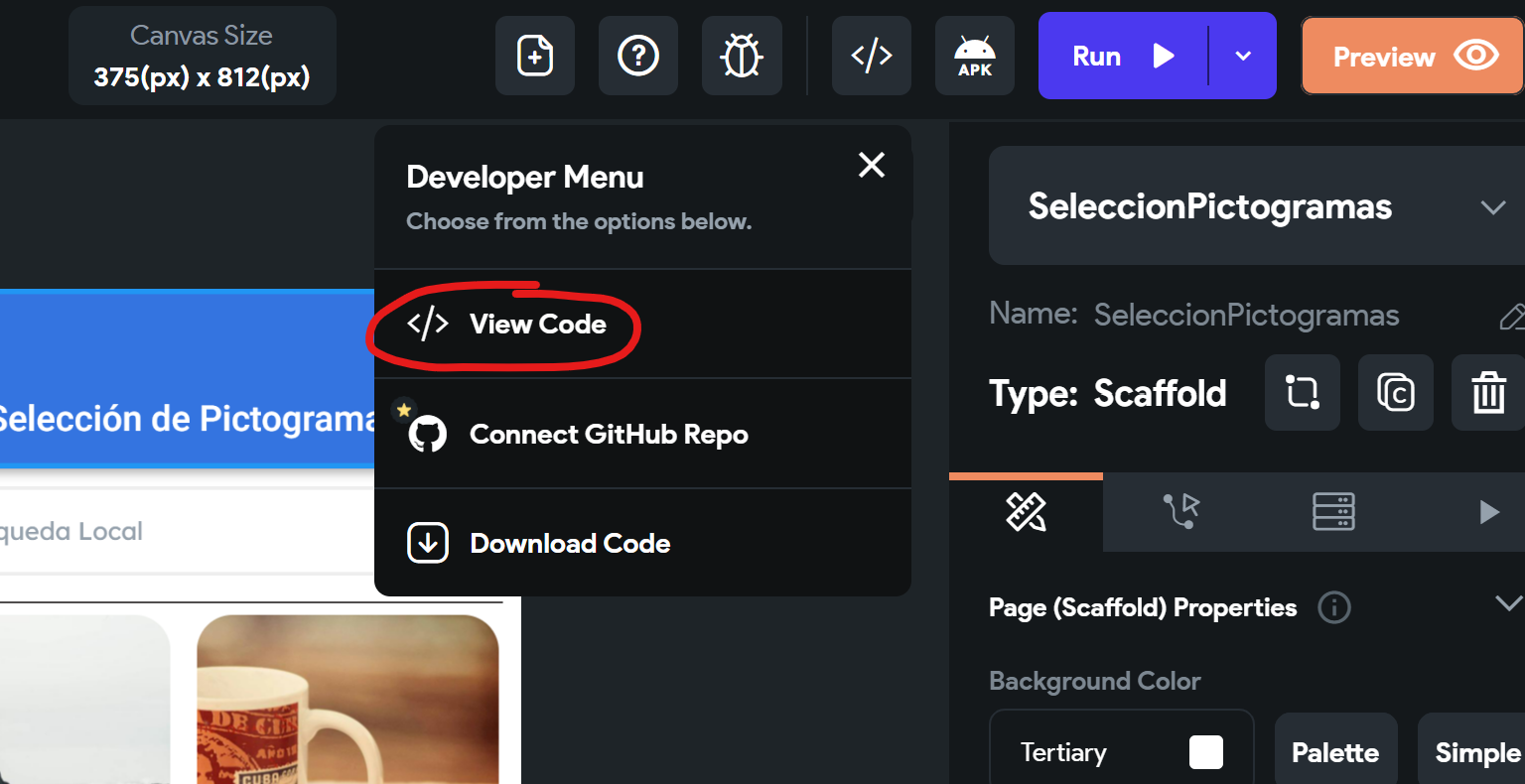
Dart siempre es **UpperCamelCase**, donde las variables se escriben la primera con minúscula y las siguientes con mayúsculas y los nombres de las funciones siempre empezando con mayúscula

## 

## 

## Para FrontEnd

Usaremos la herramienta FlutterFlow. Todos los miembros del equipo deben crearse una cuenta con el correo go.ugr.es y le aparecerá en su espacio personal el proyecto Agenda PTVAL.

Una vez hemos diseñado la pantalla, copiaremos el código mediante

Y pegaremos el código dentro de la carpeta /lib/views del proyecto, **asignándole el mismo nombre que el boceto** del que provenga.

Se debe conectar con las pantallas oportunas. Por Ejemplo:

* Si el boceto que acabo de implementar es Selección de Pictograma, tengo que indicar a la página de *Editar Tarea* que al pinchar el botón *Insertar Pictograma* nos lleve a *Selección de Pictograma*. Además, hay que programar la lógica de, cuando se seleccione un pictograma, esta pantalla debe devolver el valor seleccionado al controlador y volver a la pantalla de la que provenía (*Editar Tarea*)

### Obteniendo información del backend para el FrontEnd

Lo haremos mediante el uso del Controlador. En esta misma carpeta se proporciona un documento plantilla. Las personas que se dediquen al FrontEnd, deben rellenar una tabla por cada función que necesiten para mostrar algo en la vista, dando el mayor número de detalle posible.

### 

### Checklist para el Frontend

* git pull [git@github.com](mailto:git@github.com):Asociacion-de-Parchis/agendaptval.git
* Enlace a [FlutterFlow](https://app.flutterflow.io/project/agenda-ptval-i79nya) para diseñar las vistas.
* Copiar el código de cada vista y colocarlo en un archivo aparte en /lib/views/{Nombreboceto}.dart
* Enlazar las pantallas entre sí, incluyendo las funciones que necesite del controlador para obtener información.
* Rellenar una tabla para **incluir una función** en la vista usando el Controlador.

## Para BackEnd

El backend consistirá en la implementación del diagrama de clases en el apartado de Modelo dentro del proyecto. Esto se realizará en el directorio /lib/controlador/Controller.dart . Se recomienda ver ejemplos que están puestos en [la página de la librería](https://pub.dev/packages/mvc_pattern) para comprender el uso del MVC en flutter.

Tras implementar el diagrama de clases, se debe crear una clase Modelo, que es la encargada de mantener el estado de la sesión dentro de la aplicación. Se recomienda echar un vistazo al ejemplo propuesto [aquí](https://github.com/VictorRubia/DS_P4_Flutter/blob/master/lib/modeloControlador/Controller.dart). En esta clase guardaremos los datos para el uso correcto de la aplicación. Esta clase se deben tener los atributos necesarios (al menos uno por cada clase del modelo que tenemos) para que la app pueda funcionar. Esto no incluye el login (que sí que lo incluirá el controlador), si no que es a partir del logueo correcto cuando se almacenará en el modelo todos los datos correspondientes al usuario que inicia sesión en la aplicación, cargándolos de la BD de una sola vez.

El controlador debe ser una clase intermedia entre los datos que tenemos en el modelo cargados y las vistas. Las funciones que tenemos en esta clase deben ser llamadas en la vista para obtener los datos ya listos y procesados para la vista. Para ello, las personas encargadas del FrontEnd tienen que rellenar un documento que se encuentra en esta misma carpeta para, a medida que vayan haciendo las pantallas, rellenen qué funciones necesitan para rellenar los datos con todos los tipos de datos y una descripción.

### Checklist para el Backend

* git pull [git@github.com](mailto:git@github.com):Asociacion-de-Parchis/agendaptval.git
* Pasar el diagrama de clases a dart.
* Crear un archivo Controller.dart para tener ahí dentro el modelo y el Controlador ([ejemplo](https://github.com/VictorRubia/DS_P4_Flutter))
* Codificar el modelo con todos los getters y setters necesarios
* Crear el controlador con todas las funciones que se vayan añadiendo en el documento de controlador
* Establecer comunicación con la base de datos.
* Establecer comunicación con el servidor de archivos.