Patrones de Arquitectura de Software

Unidad 1

García Galaz Rodolfo

20491167

Desarrollo de Aplicaciones Móvil

Jose Ramón Bogarin Valenzuela

Instituto Tecnológico de Mexicali

17 de febrero de 2025

Mexicali, B.C.

ÍNDICE

1. Model-View-Controller **2**

2. Model-View-Presenter **3**

3. Model-View-ViewModel **4**

4. Clean Code **5**

ÍNDICE DE FIGURAS

1. Estructura de archivos de MVC 2
2. Estructura de archivos de MVP 3
3. Estructura de archivos de MVVM 4
4. Estructura de archivos de Clean Code 5

1 MODEL-VIEW-CONTROLLER

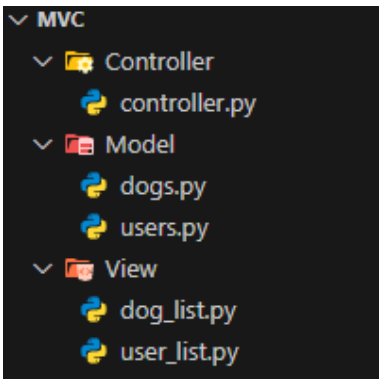

Figura 1: Estructura de archivos de MVC

La estructura con el modelo MVC esta formada por las siguientes carpetas y archivos:

- **Model**. Contiene las clases de objetos y la lógica de negocios relacionada con los mismos.
  - **dogs.py**: Archivo donde se declara la clase y funciones para crear, actualizar, eliminar y leer los objetos de clase Dog.
  - **users.py**: Archivo donde se declara la clase y funciones para crear, actualizar, eliminar y leer los objetos de los usuarios.
- **View**. Contiene los endpoints para el usuario final y solicita la información a través de los controladores.
  - **dog_list.py**: Muestra un listado de los perros actualmente creados en la base de datos.
  - **user_list.py**: Muestra un listado con los nombres de los usuarios de la base de datos.
- **Controller**. Contienen las funciones para proveer a los endpoints con los datos que solicitan, y esto es a través de los modelos.
  - **controller.py**: Incluye la lógica para solicitar actualizaciones, cambios, lecturas y demás a los modelos, así como el código que da formato a los datos para ser visualizados en los endpoints .

2 MODEL-VIEW-PRESENTER

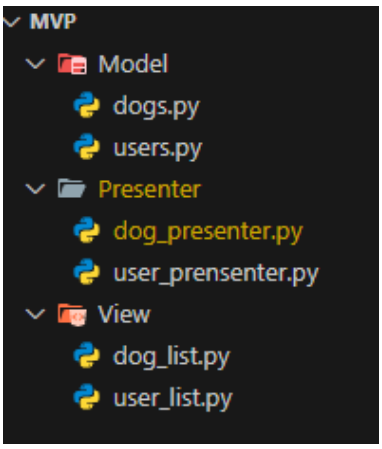

Figura 2: Estructura de archivos de MVP

La estructura con el modelo MVP esta formada por las siguientes carpetas y archivos:

- **Model**. Contiene las clases de objetos y la lógica de negocios relacionada con los mismos.
  - **dogs.py**: Archivo donde se declara la clase y funciones para crear, actualizar, eliminar y leer los objetos de clase Dog.
  - **users.py**: Archivo donde se declara la clase y funciones para crear, actualizar, eliminar y leer los objetos de los usuarios.
- **View**. Las pantallas de la aplicación y la lógica de las funciones de los elementos de la UI.
  - **dog_list.py**: Muestra un listado de los perros actualmente creados en la base de datos.
  - **user_list.py**: Muestra un listado con los nombres de los usuarios de la base de datos.
- **Presenter**. Contienen las funciones para proveer a las pantallas con la información que necesitan en el formato que son solicitadas.
  - **dog_presenter.py**: Incluye la lógica para solicitar el listado de los objetos *Dog*, así como el código que da formato a los datos para ser visualizados en las pantallas.
  - **user_presenter.py**: Incluye la lógica para solicitar el listado de los objetos *User*, así como el código que da formato a los datos para ser visualizados en las pantallas.

3 MODEL-VIEW-VIEWMODEL

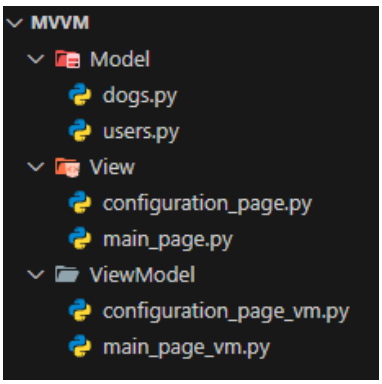

Figura 3: Estructura de archivos de MVVM

La estructura con el modelo MVVM esta formada por las siguientes carpetas y archivos:

- **Model**. Contiene las clases de objetos y la lógica de negocios relacionada con los mismos.
  - **dogs.py**: Archivo donde se declara la clase y funciones para crear, actualizar, eliminar y leer los objetos de clase Dog.
  - **users.py**: Archivo donde se declara la clase y funciones para crear, actualizar, eliminar y leer los objetos de los usuarios, incluyendo funcionalidad para verificar credenciales.
- **View**. Contiene el diseño de las pantallas de la aplicación
  - **configuration_page.py**: Archivo que declara la estructura visual de la página de configuración y declara como su viewmodel a su correspondiente.
  - **main_page.py**: Archivo que contiene la parte visual de la pantalla principal de la aplicación.
- **ViewModel**. Los archivos aquí contenidos alojan la lógica funcional de las pantallas, como pueden ser las funciones de clic de cada una de las pantallas, así como las variables que contienen los valores que se muestren en las mismas.
  - **configuration_page_vm.py**: Archivo contiene la lógica de negocios de la página de configuración y se conecta los modelos pertinentes.
  - **main_page_vm.py**: Contiene la lógica de negocios de la página de principal y se conecta los modelos pertinentes.

4 CLEAN CODE

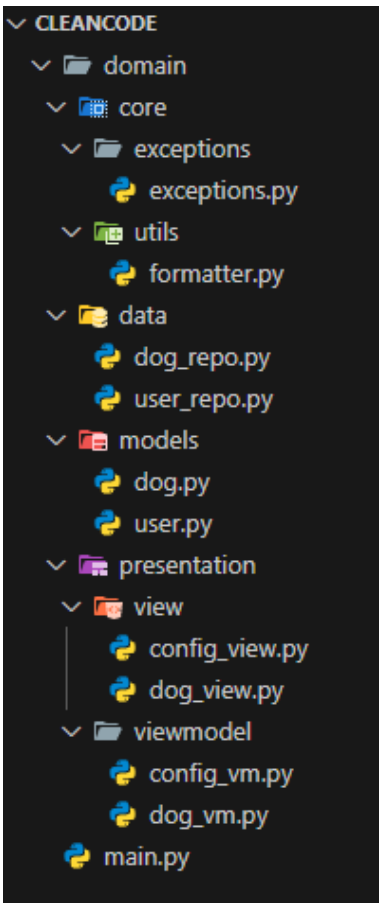

Figura 4: Estructura de archivos de Clean Code

La estructura con el modelo Clean Code esta basada en funcionalidad y debe ser adaptada de acuerdo a la aplicación que se esta creando. El ejemplo siguiente esta basado en una aplicación en donde se muestran un listado de perros y se tiene una configuración de usuario, y por las siguientes carpetas y archivos:

- **domain**. Todo el código de la aplicación.
  - **core**. Contiene funciones y demás utilizados de manera global en la aplicación como excepciones y utilerías para dar formato a múltiples pantallas. Estas funciones son generales y no especificas a una sola pantalla o parte de la aplicación.
  - **data**. Contiene la lógica para obtener la información de la base de datos o servicio que provee los datos que utiliza la aplicación.
  - **models**. Contiene las clases y funciones de los objetos usados por la aplicación.
  - **presentation**. Contiene el código relacionado con las pantallas de la aplicación
    - *view*: Contiene la parte gráfica de las pantallas de la aplicación.
    - *viewmodel*: Contiene la lógica de la funcionalidad de las pantallas y las variables que se muestran en la misma.

- **main.py**: Es el archivo ejecutable que inicia la aplicación.