**Desarrollo de Interfaces**

Imagen que contiene equipaje, tabla, pieza, maleta

Descripción generada automáticamente

**I.E.S. Carrillo Salcedo**

**2ª Desarrollo de Aplicaciones Multiplataforma**

**Curso 2024 / 2025**

**Tema 03 – Boletín 01**

**Trabajo Realizado por Equipo 01:**

**Molero Marín, Juan**

**Rodríguez López, Julio**

**Martínez Lorda, Julián**

**Contenido**

[**0.- Pasos Previos – Despliegue de la Aplicación: 3**](#_Toc183201686)

[**Entorno Virtual: 3**](#_Toc183201687)

[**Docker y BBDD PostgreSQL: 5**](#_Toc183201688)

[**1.- Creación de Componentes: 6**](#_Toc183201689)

[**1.1.- Custom Button: 6**](#_Toc183201690)

[**1.2.- Custom Search Bar: 7**](#_Toc183201691)

[**1.3.- Custom Table View: 10**](#_Toc183201692)

[**2.- Eventos y Señales de los Componentes: 14**](#_Toc183201693)

[**3.- Empaquetado de los Componentes: 16**](#_Toc183201694)

[**3.1.- Archivo setup.py: 18**](#_Toc183201695)

[**4.- Pruebas / Test Unitarios: 19**](#_Toc183201696)

[**4.1.- Pruebas Custom Button – test\_custom\_button.py: 19**](#_Toc183201697)

[**4.2.- Pruebas Custom Search Bar – test\_custom\_search\_bar.py: 20**](#_Toc183201698)

[**4.3.- Pruebas Custom Table View – test\_custom\_tableview.py: 21**](#_Toc183201699)

[**5.- Bibliografía: 22**](#_Toc183201700)

[**6.- Enlace a Repositorio Github: 23**](#_Toc183201701)

# **0.- Pasos Previos – Despliegue de la Aplicación:**

## **Entorno Virtual:**

De forma previa a comenzar la realización de este proyecto, es necesario instalar el entorno virtual para poder trabajar, ya que entre otros aspectos fundamentales, trae integrado todas las dependencias que necesitaremos para poder desarrollar nuestro código, permitiendo instalar en cada proyecto lo necesario sin tener que interferir dentro de otros.

Para ello, lo primero que tenemos que hacer es crearnos un directorio, y desde VSC navegar hacia él. Una vez estemos dentro de la raíz del mismo, ejecutaremos el siguiente comando:

**Captura de pantalla de computadora

Descripción generada automáticamentepython -m venv ‘nombre del entorno’:**

Una vez se haya creado el entorno virtual, lo siguiente será instalar las dependencias respectivamente, las cuáles se incluyen dentro de un archivo denominado “requirements.txt”, el cual deberá ser modificado en función de las especificaciones de cada proyecto, respectivamente. Es por ello que nos traemos dicho archivo a la raíz de nuestro proyecto, y posteriormente, ejecutamos el comando:

**Interfaz de usuario gráfica

Descripción generada automáticamentepip install -r requirements.txt**

Una vez tengamos todas las dependencias que necesitamos instaladas, nuestro siguiente paso será el de activar el entorno previamente. Esto lo conseguimos mediante un comando, así como para desactivarlo y/o eliminarlo tendremos otros respectivamente:

**.\.env\Scripts\activate 🡪** Comando con el que activaremos el entorno virtual.

**deactivate** 🡪 Nos desactiva el entorno virtual.

Captura de pantalla de un celular

Descripción generada automáticamente**Remove-Item -Recurse -Force .\nombre\_entorno 🡪** Nos elimina el entorno virtual, para lo cual primero debe de ser desactivado.

# **Docker y BBDD PostgreSQL:**

Para esta parte, se ha empleado, una herramienta de Docker denominada Docker Compose, la cual nos permite definir, ejecutar y gestionar respectivamente un conjunto de contenedores o un solo contenedor, con el objetivo de poder iniciarlos, detenerlos y/o configurarlos para que puedan gestionar el servicios postgres.

Es por ello que hemos tenido que realizar y configurar el archivo **docker-compose.yml** en la raíz del proyecto, mediante el cual con el comando **docker-compose up** podremos levantar y ejecutar nuestro contenedor de Docker, y mediante **docker-compose down** lo detendremos y eliminaremos posteriormente:

Texto

Descripción generada automáticamente

Con el archivo ya configurado y listo, es hora de ejecutar el comando, crear e iniciar el contenedor, y por lo tanto, iniciar el servicio para poder tener conexión con la base de datos:

Texto

Descripción generada automáticamente

# **1.- Creación de Componentes:**

## **1.1.- Custom Button:**

Captura de pantalla de computadora

Descripción generada automáticamenteNuestro primer componente para la nueva ventana de **“Búsqueda”** será un botón, el cuál será el encargado de redirigirnos nuevamente e la ventana de **“Login”** una vez el usuario haya finalizado las consultas necesarias. Para ello en un proyecto independiente, hemos desarrollado la siguiente clase, ya que posteriormente deberá ser empaquetada dentro de nuestro proyecto principal:

Captura de pantalla de computadora

Descripción generada automáticamenteAdemás de ello, también hemos incluido un módulo para los colores, y otro para las rutas, tal y como podemos apreciar aquí:

## **1.2.- Custom Search Bar:**

**1.2.1.- Custom Seach Bar Controller:**

Texto

Descripción generada automáticamente

En el módulo, hemos implementado todas las funciones que poseerá nuestra barra de búsqueda, donde mediante slots hemos implementado funcionalidades y eventos.

**1.2.2.-Custom Search Bar:**

Texto

Descripción generada automáticamente

En este módulo hemos diseñado la vista que poseerá nuestra barra de búsqueda. Posteriormente, será integrada dentro de la nueva ventana **“Búsqueda”** que vamos a realizar dentro de nuestro proyecto principal.

Texto

Descripción generada automáticamenteAdemás del controlador y de la vista respectivamente, también incluye dos iconos .png (una papelera y una lupa que acompañarán respectivamente a la barra, vinculándose en el módulo que hemos visto anteriormente), y un directorio utils, donde recogeremos los módulos **path.py, etiquetas.py, estilos.py y colores.py:**

Captura de pantalla de computadora

Descripción generada automáticamente

## **1.3.- Custom Table View:**

Captura de pantalla de computadora

Descripción generada automáticamente**1.3.1-Custom Table View Controller:**

Módulo donde vamos a implementar las funcionalidades de nuestra vista o tabla, donde se van a recoger los datos de nuestras instancias **(Videojuegos),** para lo cual vamos a necesitar por un lado una clase Controlador también para este tipo de objeto, así como el modelo donde lo hemos desarrollado también.

**1.3.2.- Videojuego Controller:**

Texto

Descripción generada automáticamente

Módulo donde se inicializará el controlador, cargando los videojuegos desde la base de datos para ser conectado posteriormente con la vista de la tabla, módulo que vamos a mostrar a continuación.

**1.3.3- Modelo Videojuego:**

**Captura de pantalla de computadora

Descripción generada automáticamente**

Módulo donde se ha desarrollado la clase Videojuego, la cuál a partir de ahora será determinante en el desarrollo de la asignatura, tanto para este proyecto como para otros que deberemos desarrollar posteriormente.

Es la encargada de conectarse con el controlador anteriormente mencionado para poder transmitir los datos de las instancias a la vista, respectivamente.

**1.3.4.- Custom Table View:**

Texto

Descripción generada automáticamenteMódulo donde hemos configurado el aspecto visual de la tabla, es decir, su representación gráfica dentro de nuestra nueva ventana. Además de ello, hemos implementado otras funcionalidades a destacar, como por ejemplo el de filtrar, es decir, se ha manejado que se vaya filtrando lo que el usuario va a escribir dentro de la barra de búsqueda:

Para finalizar, también vamos a mostrar su contenido dentro del directorio utils, en este caso nuevamente serán dos módulos **(estilos.py y etiqueta.py):**

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

# **2.- Eventos y Señales de los Componentes:**

Estos tres componentes van a ser integrados dentro de una nueva ventana de Búsqueda, tal y como ya hemos ido explicando. Para ello, en nuestro proyecto principal han tenido que ser importados, tal y como vamos a explicar ahora un poco más adelante en el apartado del empaquetado. El funcionamiento verdaderamente es muy simple: La nueva ventana la vamos a crear mediante una nueva instancia de QMainWindow, a la cual se dirigirá el usuario tras loguearse dentro de la ventana anterior. A continuación, vamos a explicarlo con más profundidad:

Pantalla de computadora

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Escala de tiempo

Descripción generada automáticamenteCabe destacar que es en el constructor donde todo posteriormente se llevará a cabo, menos el botón de volver al Login que se le asocia al botón en el momento de crear el widget o componente. Además de todo esto, también hemos tenido que implementar una funcionalidad nueva dentro de la clase Loguin para que el usuario, tras iniciar sesión, pueda ser redirigido a esta ventana:

Nuestro resultado final ha sido el de poder navegar con nuestros usuarios creados a una nueva ventana de búsqueda, la cual tendrá tanto una barra para filtrar, como una tabla donde se mostrarán los resultados, así como un botón para volver:

Una captura de pantalla de una computadora

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente

# **3.- Empaquetado de los Componentes:**

Para poder usar estos tres componentes en nuestro proyecto primero ha sido necesario, tras elaborarlos en proyectos por separado como ya habíamos indicado, el proceso de empaquetarlos tanto en formatos comprimido **(tar.gz)** como en formato de instalación de módulos para Python **(whl).** Cabe destacar que para que este proceso resulte exitoso, deberemos incluir en cada directorio que deseamos exportar como un paquete distinto el archivo **‘\_\_init\_\_.py’** respectivamente.

Para ello, en cada uno de los proyectos por separado, tras finalizarlos hemos tenido que ejecutar por consola el siguiente comando desde la raíz del directorio:

**python setup.py sdist bdist\_wheel**

Tras ejecutarlo, se nos incluirá en la estructura del proyecto lo siguiente:

Texto

Descripción generada automáticamente

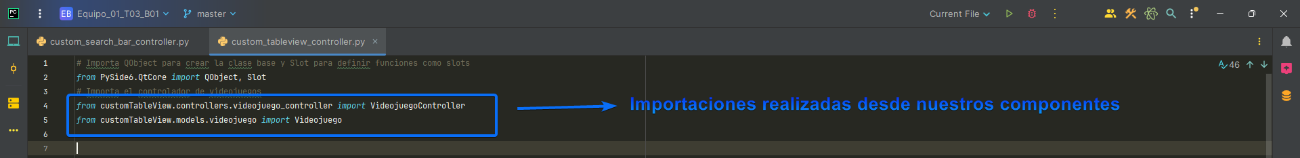
Una vez los tengamos todos listos, es hora de crear un nuevo directorio ‘dist’ dentro de nuestro proyecto, que será donde los hemos trasladados para ser instalados de forma posterior. Una vez allí tendremos que ejecutar los siguientes comandos:

**pip install .\dist\custom\_button-1.0.0-py3-none-any.whl**

**pip install .\dist\custom\_search\_bar-1.0.0-py3-none-any.whl**

**pip install .\dist\custom\_table\_view-1.0.0-py3-none-any.whl**

Tras ejecutarlos, cada componente será una librería nueva dentro de nuestro proyecto, y podrán ser importados dentro de cualquier clase para poder ser usados, tal y como apreciamos en el siguiente ejemplo:



## **3.1.- Archivo setup.py:**

Para que todo este proceso haya sido un éxito, en la raíz de los proyectos individuales de cada uno de los componentes hemos tenido que incluir el archivo **‘setup.py’**, archivo clave dentro del proceso al poseer toda la información y configuraciones necesarias para que el mismo se lleve a cabo.

Para poder realizarlo, lo primero ha tenido que ser importar dentro del mismo la librería **‘setuptools’,** de la cual hemos obtenido el propio **setup** o empaquetado, así como el **find\_packages(),** mediante el cual se localizarán todos los paquetes que deseamos empaquetar, gracias al **‘\_\_init\_\_’** ya mencionado.

En nuestro caso, esta ha sido la estructura de dicho archivo:

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

# **4.- Pruebas / Test Unitarios:**

Para la realización de nuestra pruebas unitarias, nos hemos decantado por **pytest -qt,** herramienta o complemento de Pytest que nos permitirá la realización de las mismas.

En nuestro caos particular, hemos realizado un total de 3 pruebas para cada componente, con lo que tenemos 9 pruebas en total.

## Texto Descripción generada automáticamente**4.1.- Pruebas Custom Button – test\_custom\_button.py:**

En el caso del botón, se ha realizado una prueba para que se inicie con las propiedades correctas, otra para que se inicie con sus estilos correspondiente, así como una tercera para controlar el evento de pasar el ratón sobre él.

## Texto Descripción generada automáticamente**4.2.- Pruebas Custom Search Bar – test\_custom\_search\_bar.py:**

En la barra de búsqueda hemos verificado que se crea e inicia correctamente, que el texto del placeholder (marcador de posición) sea el correcto y que los atajos de teclado estén bien configurados.

## **4.3.- Pruebas Custom Table View – test\_custom\_tableview.py:**

Texto

Descripción generada automáticamente

Para concluir, en la tabla probaremos que se crea correctamente, que los datos han sido cargados en ella y que nuestro filtro de búsqueda funciona correctamente.

Tras el diseño o desarrollo de las mismas, es momento de llevarlas a cabo. Para ello, nos tendremos que ir respectivamente al directorio **‘test’,** y ejecutar el siguiente comando:

**pytest .\tests\nombre\_del\_test.py 🡪** En nuestro caso específico han sido:

**pytest .\tests\test\_custom\_search\_bar.py**

**pytest .\tests\test\_custom\_button.py**

**pytest .\tests\test\_custom\_tableview.py**

Tras ejecutarlos, hemos obtenido resultados positivos con cada uno de ellos, tal y como se va a mostrar a continuación:

**Resultados Custom Table View:**Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente**Resultados Custom Search Bar:**

Texto

Descripción generada automáticamente**Resultados Custom Button:**

# **5.- Bibliografía:**

*pytest-qt — pytest-qt documentation*. (s. f.). Readthedocs.io. Recuperado 22 de noviembre de 2024, de <https://pytest-qt.readthedocs.io/en/latest/intro.html>

*Qt for Python*. (s. f.). Doc.qt.io. Recuperado 22 de noviembre de 2024, de <https://doc.qt.io/qtforpython-6/>

Atoche Ortega, J. (2024). *Tema 03: Desarrollo de Componentes Visuales Personalizados.* Dpto de Informática, IES Carrillo Salcedo.

# **6.- Enlace a Repositorio Github:**

<https://github.com/jmolmar-dev/Equipo_01_T03_B01.git>