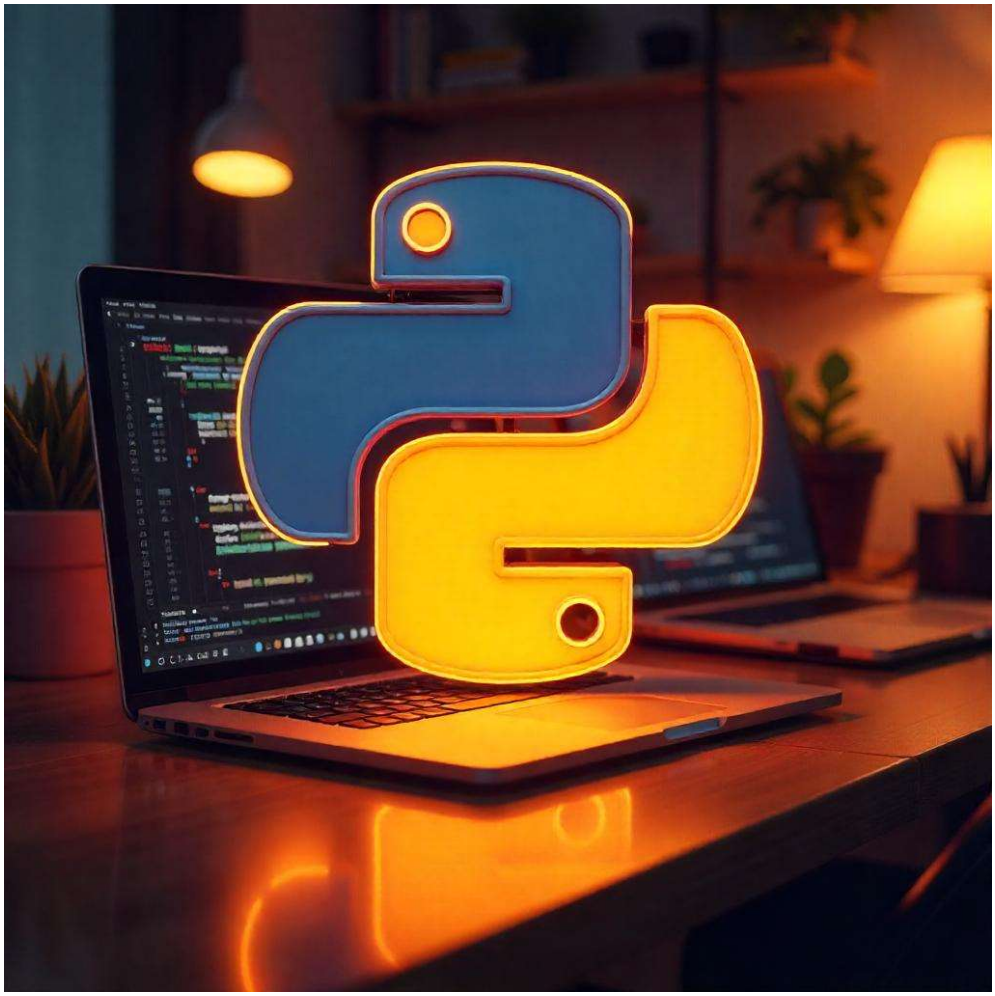
Desarrollo de Interfaces



I.E.S. Carrillo Salcedo

2º Desarrollo de Aplicaciones Multiplataforma

Curso 2024/2025

Tema 02 – Boletín 02

Trabajo realizado por Equipo 01:

Rodríguez López, Julio

Martínez Lorda, Julián

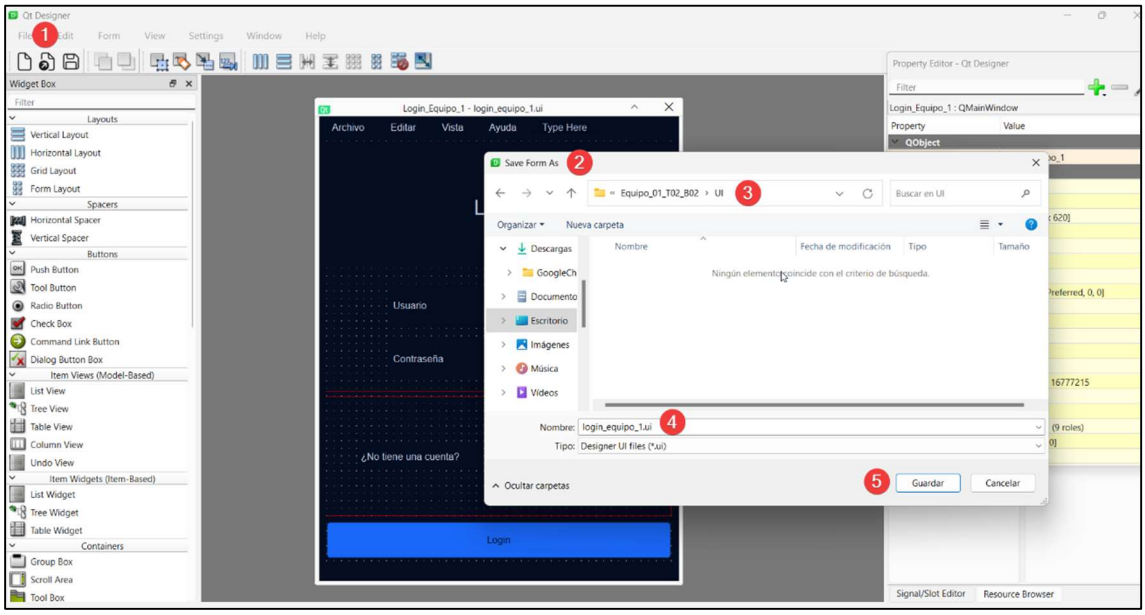
Molero Marín, Juan

Contenido

1.- Generación y Análisis del código generado por el editor visual:

1.1.- Exportar el código genero por Qt Designer a un archivo .ui (un archivo para el diseño de Login y otro para el registro):

Para ello, desde el menú principal de Qt Designer, seleccionamos File y posteriormente save as. Tras esto, los tendremos que guardar en la estructura del proyecto correspondiente:



1.2.- Analiza y Explica el Código, tanto su estructura (como está estructurado y organizado), como sus elementos:

- **<ui> :** Representa toda la interfaz de usuario y actúa como el elemento raíz del archivo. A partir de este elemento principal, se construyen todos los demás componentes de la interfaz.

- **<class>:** Define el nombre de la clase correspondiente al widget principal del archivo .ui Este nombre será utilizado para generar la clase contenedora cuando el archivo XML se convierta en código Python.

- **<widget>:** Define el elemento visual e interactivo dentro de la interfaz de usuario, que será visible para el usuario final. A este elemento se le asignan todas sus propiedades, como el tipo de widget, estilo, disposición (layout), restricciones, señales y slots. Este elemento es la base sobre la cual QT organiza y estructura las interfaces de usuario.

- **<property>:** Es un elemento hijo de <widget> que especifica una propiedad particular del widget padre.

- **<x>:** Hijo de <property>. Define la posición horizontal del widget en el eje X.

- **<y>:** Hijo de <property>. Define la posición vertical del widget en el eje Y.

- **<width>:** Hijo de <property>. Define el ancho del widget en píxeles.

- **<height>:** Hijo de <property>. Establece la altura del widget en píxeles.

- **<string>:** Hijo de <property>. Almacena texto que alguna propiedad del widget puede requerir para mostrar en la interfaz, como el texto de una etiqueta o elementos de hojas de estilo.

- **<enum>:** Hijo de <property>. Referencia un valor enumerado de una biblioteca de QT

- **<bool>:** Hijo de <property>. Define si la propiedad padre es verdadera o falsa.

- **<layout>:** Hijo de <widget> o <item>. Se utiliza para definir la disposición (layout) de los widgets hijos cuando un widget contiene otros widgets. Ejemplos de disposiciones son QVBoxLayout, QHBoxLayout y QGridLayout.

- **<item>:** Hijo de <layout> o de un <widget> de tipo contenedor. Representa un espacio reservado para un elemento, como un widget hijo, un sub-layout, o un elemento espaciador.

- **<number>:** Hijo de <property>. Almacena un valor numérico necesario para una propiedad específica.

- **<size>:** Hijo de <property>. Utilizado para definir el tamaño mínimo o máximo de un widget. Emplea las etiquetas <width> y <height> para establecer el tamaño en pixeles.

- **<addaction>:** Vincula un elemento de acción con un widget específico, permitiendo una asociación funcional.

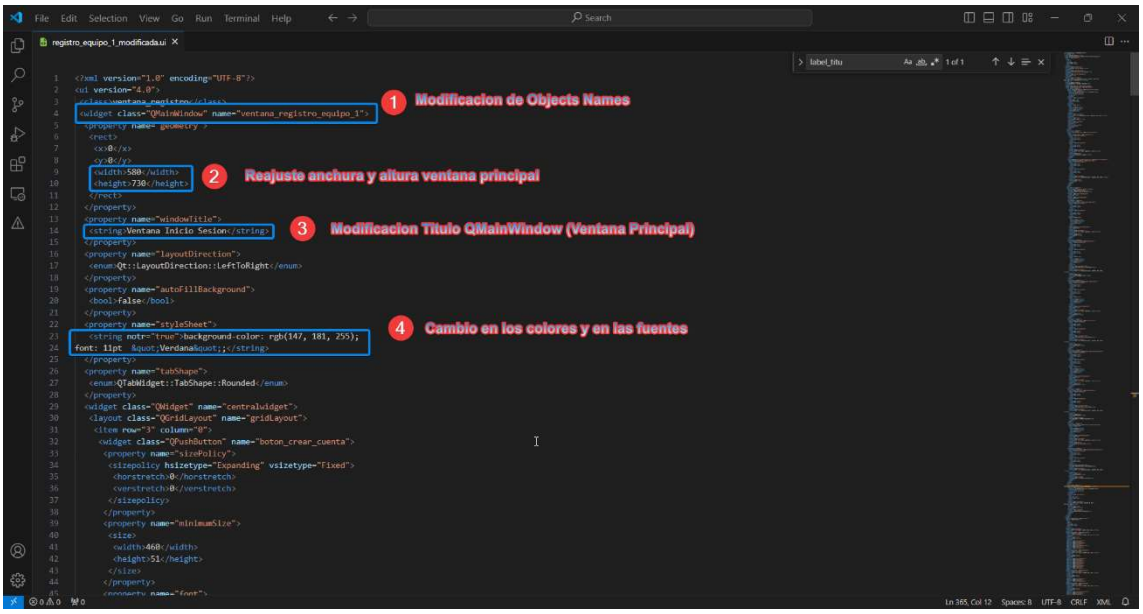
- **<action>:** Define un comportamiento que el usuario puede activar en la aplicación. Este comportamiento se establece mediante propiedades y es reutilizable en diferentes partes de la interfaz.

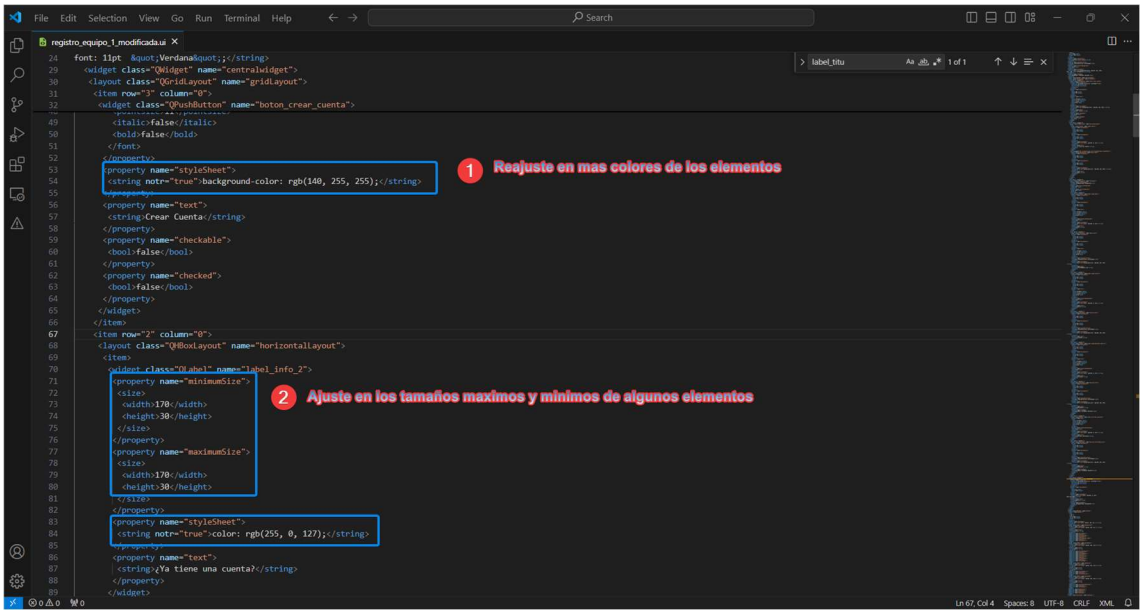
Estructura del Código: La estructura del archivo para del elemento raíz <ui>. Dentro de este, se añaden widgets con sus respectivas propiedades, items, disposiciones (layouts), enlaces a acciones específicas, otros widgets hijos, acciones que representan comportamiento reutilizables, recursos del sistema de archivos que la interfaz necesita, y conexiones entre señales y slots. Esta estructura permite una organización jerárquica de la interfaz, facilitando la construcción modular y reutilizable de los componentes visuales.

2.- Modificación del código generado por el editor visual:

2.1.- Ventana Registro Equipo 01:

Lo primero que hemos hecho ha sido modificar el tamaño de las fuentes, pasando respectivamente todos los label, todos los LineEdit y todo el texto que contienen los botones respectivamente a la fuente Verdana, tamaño 11. Además de todo ello, han sido modificados todos los colores, tanto de los elementos del QMenuBar, como los que acabamos de mencionar anteriormente y se han modificado algunos Object Name que en versiones anteriores inducían a una confusión:

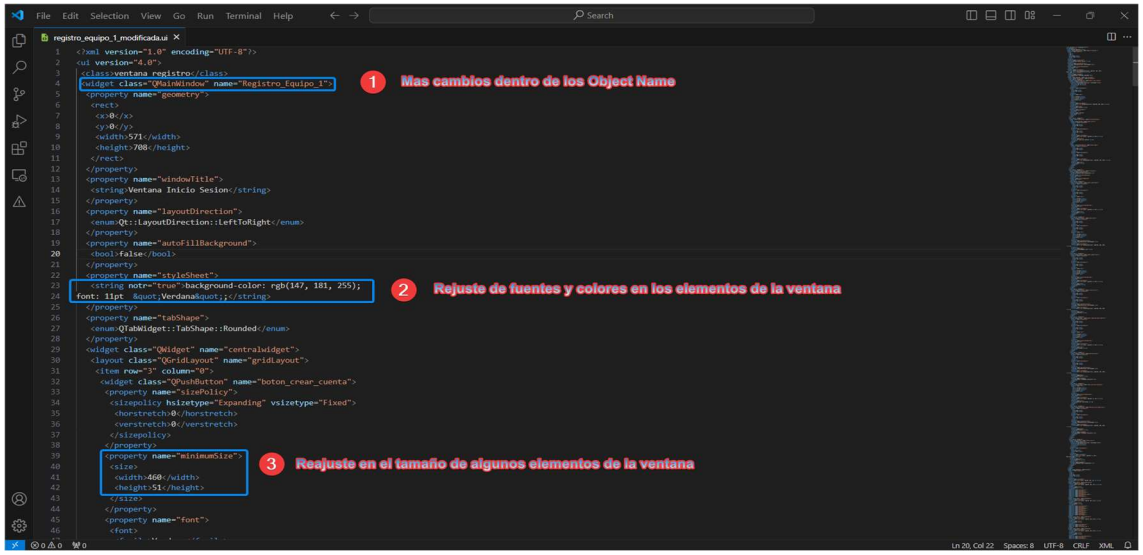


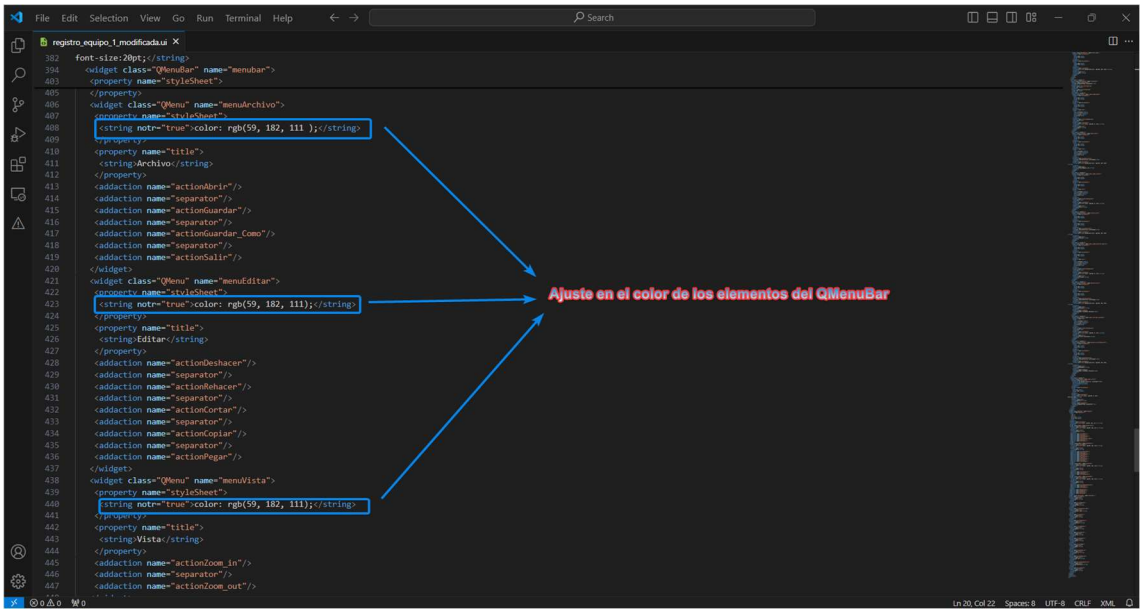


En cuanto a la distribución de nuestros layouts, hemos decidido respetar la estructura debido a que nos parece idónea de cara a seguir desarrollando nuestro proyecto de forma posterior en las siguientes prácticas.

2.2.- Ventana Login Equipo 01:

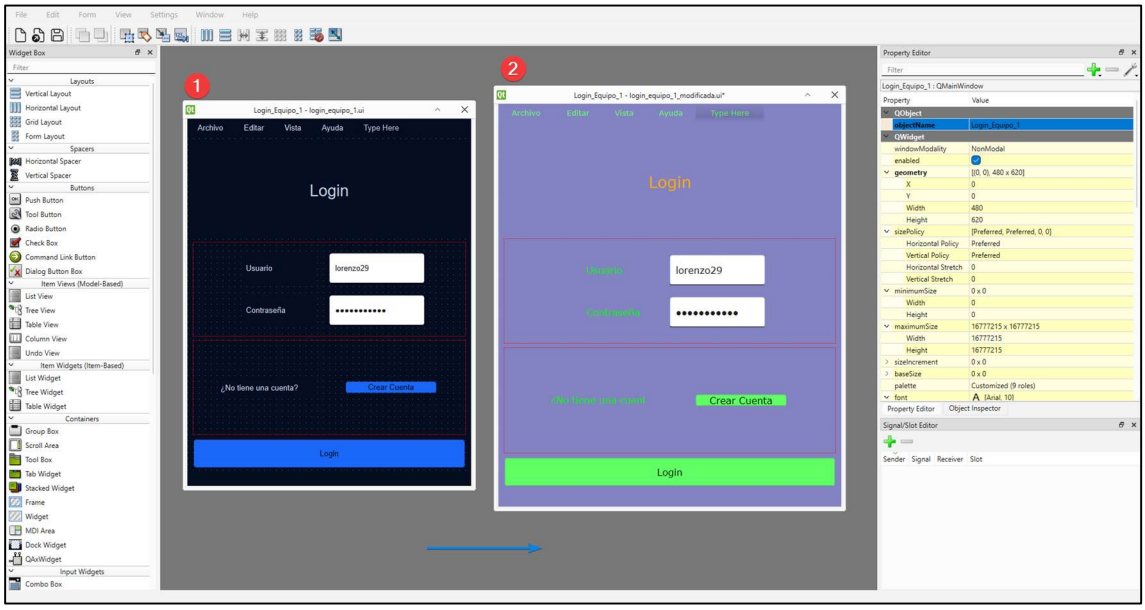
Las modificaciones realizadas son similares al caso anterior: hemos ido retocando todas las fuentes y colores de los diversos elementos: QMenuBar, QMainWindow, QPushButtons, QLabels y LineEdits, respectivamente:



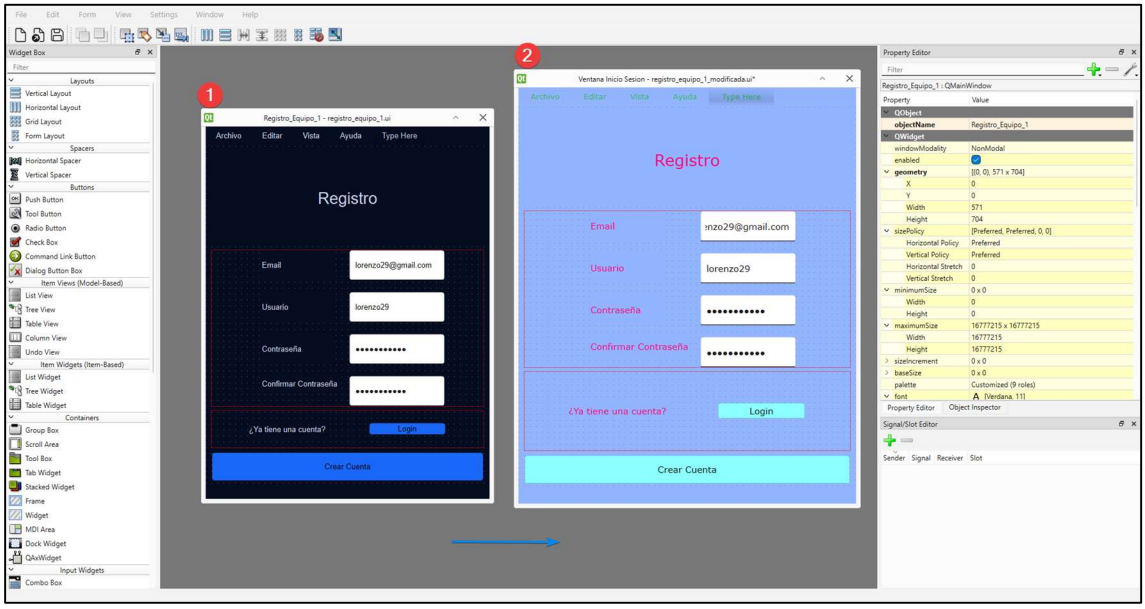


Para terminar, si comparamos los resultados, podemos apreciar claramente las diferencias:

Resultado Ventana Login:



Resultado Ventana Registro:



3.- Ventajas de generar interfaces en XML:

3.1.- Investigar las principales ventajas de emplear XML para la generación de interfaces de usuario:

El hecho de escoger el formato XML a la hora de desarrollar interfaces de usuario trae consigo un conjunto de beneficios dentro del ámbito del intercambio de datos. Para comenzar, **su estructura jerárquica** y **legible** permite una representación de la información de forma coherente, lo que conlleva una **clara interpretación de la misma tanto por parte de las máquinas como de las personas.**

Además, la independencia que posee el formato con respecto a las plataformas y Sistemas Operativos garantiza una amplia compatibilidad sin restricciones y una excelente portabilidad, por lo que su uso permite la elaboración de interfaces en una gran variedad de dispositivos y aplicaciones, sin que esto suponga ningún tipo de problema.

3.2.- Comparar el uso de XML frente a otras técnicas para generar interfaces gráficas:

Normalmente, cuando deseamos elaborar una interfaz de usuario, la primera opción por la que nos podríamos decantar sería mediante el mismo lenguaje de programación con los que se ha realizado también la funcionalidad, como podrían ser C# o Java, por ejemplo.

Como ventaja principal nos encontramos la sencillez y comodidad de no tener que adquirir conocimientos nuevos para ponernos manos a la obra con el proceso. Sin embargo, también nos encontramos el gran inconveniente de poseer una reducida portabilidad.

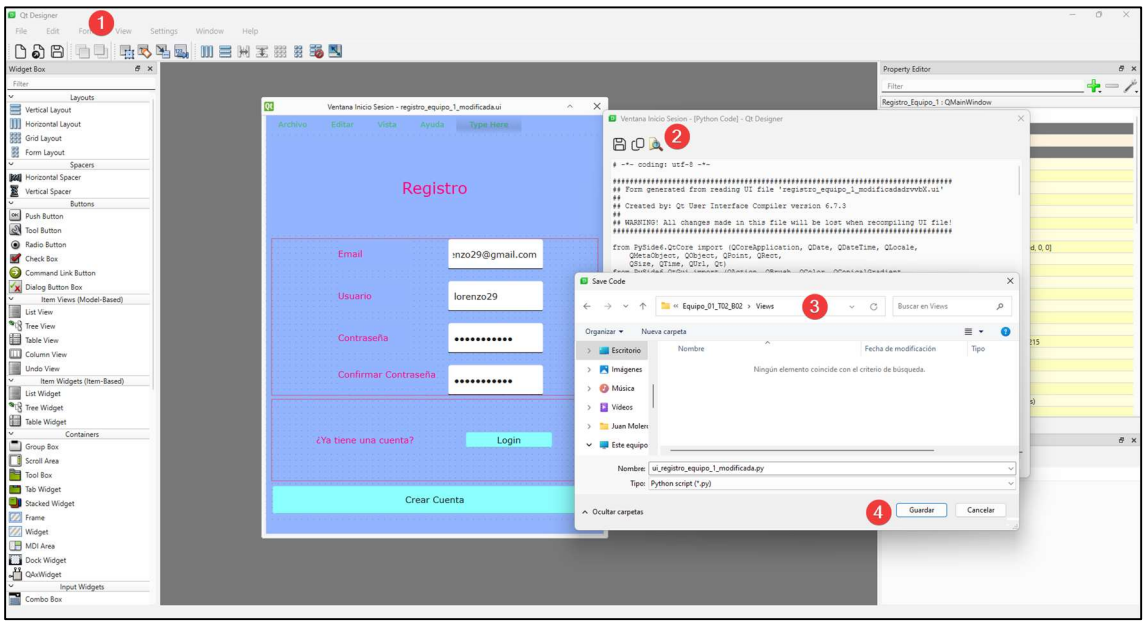
La creación de interfaces de usuarios basados en el lenguaje descriptivo XML nos viene a resolver este problema, ya que el desarrollo de la misma mediante una notación de alto nivel y su posterior almacenamiento en un archivo aparte, hace que posteriormente podamos tratar ese archivo mediante un proceso denominado **mapeo** para obtener la interfaz. Tras esto, podrá ser visualizada en el dispositivo final.

3.3.- Redactar las conclusiones obtenidas:

Para concluir, se puede afirmar que como método para crear y desarrollar interfaces, XML es una gran opción, ya que será esencial utilizarlo en proyectos que utilicen varias tecnologías o que deban de moverse en una gran cantidad de dispositivos, debido a su increíble capacidad de portabilidad, lo cual le ofrece una independencia increíble en el desarrollo de este tipo de tareas frente a otros lenguajes, como por ejemplo los mencionados anteriormente.

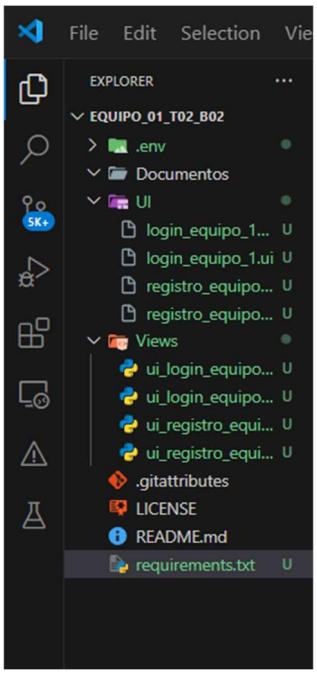
4.- Generación del código correspondiente al interfaz desde XML:

Desde Qt Design, a partir del menú principal (Form → View Python Code…), podremos convertir nuestro código .ui en código .py → llegamos a una ventana donde aparte de visualizarlo, podremos guardarlo en la ruta especificada, en nuestro caso será en una nuevo directorio denominado “Vistas” en nuestro proyecto:

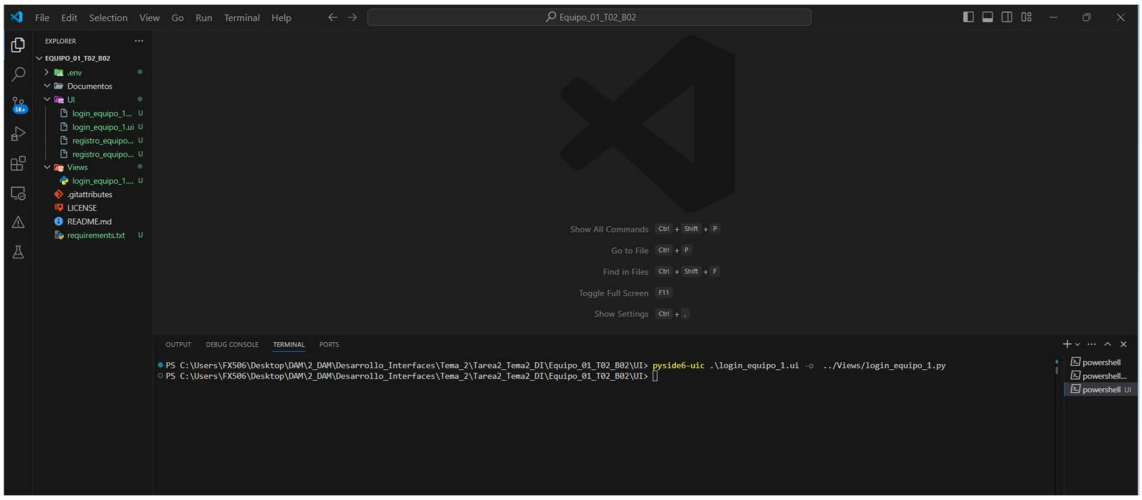


Una vez hemos guardado los archivos, respectivamente, se ha guardado en la estructura del proyecto, y se han intentado ejecutar posteriormente:

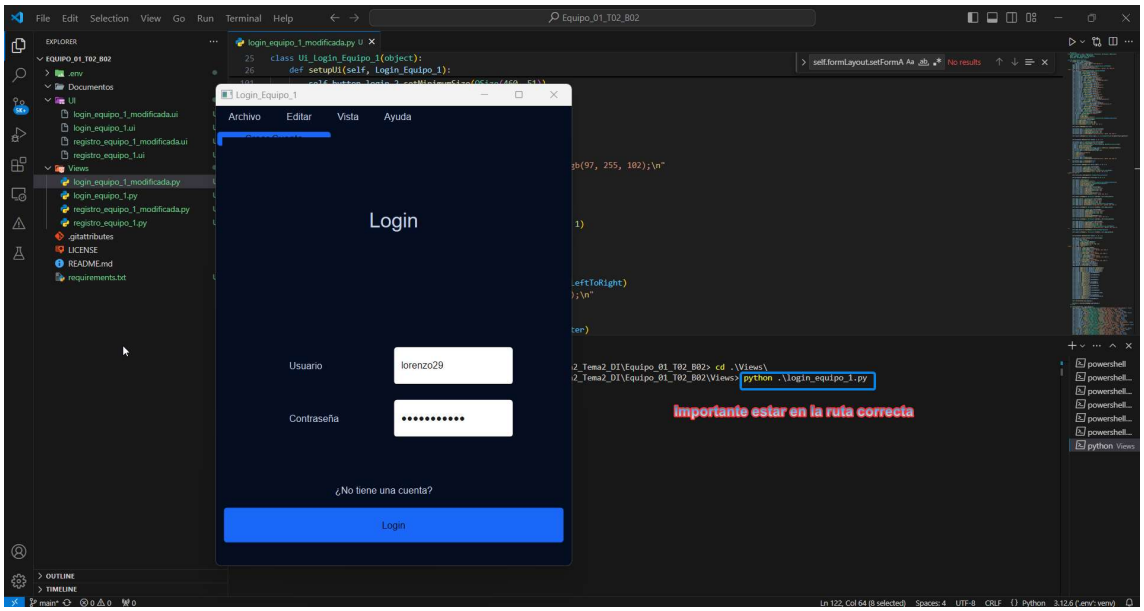
Estructura del Proyecto:

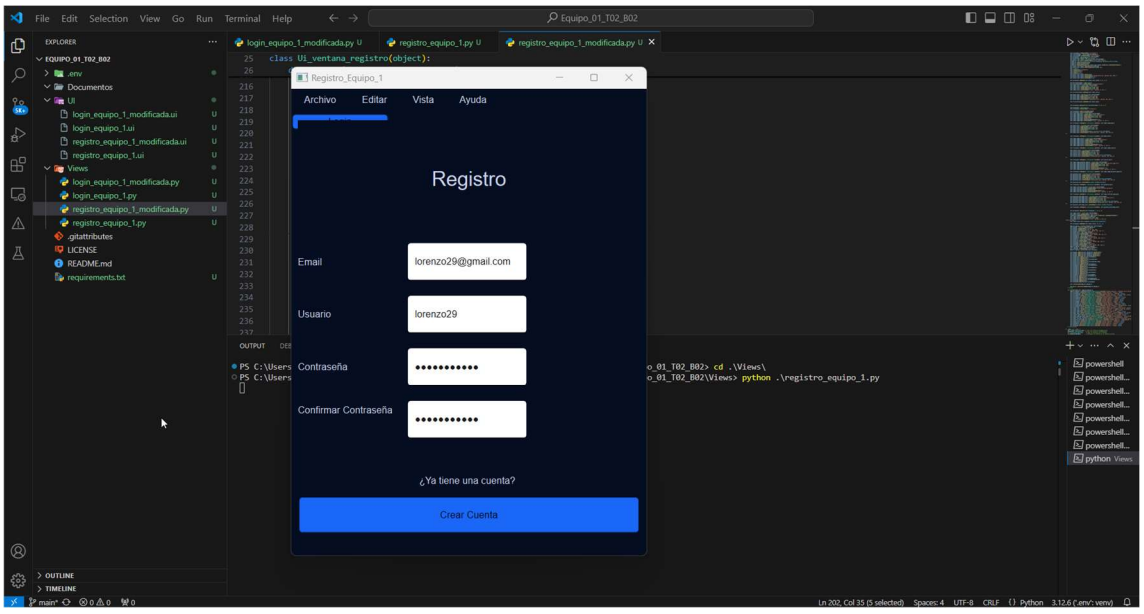


Otra forma que tendríamos de guardar o transformar los archivos .ui a .py es directamente mediante el uso del comando pyside6-uic <nombre archivo ui> -o <ruta y nombre del archivo py>:



Ejecución de los archivos: Para ejecutar los archivos, en su misma ruta realizaremos un python + nombre archivo, respectivamente, aunque previamente tendremos que preparar una clase de ejecución y hemos tenido que eliminar la siguiente propiedad de los mismos: self.formLayout.setFormAlignment(Qt.AlignmentFlag.AlignCenter).





5.- Bibliografía:

Educación y Formación Profesional. Cidead, M. (s. f.). *DI02.- Elaboración de interfaces mediante documentos XML*. Caib.es. Recuperado 28 de octubre de 2024, de https://sarreplec.caib.es/pluginfile.php/9952/mod_resource/content/2/DI02_Completa_Offline/index.html

Formato XML: Funciones y Aplicaciones - IMMUNE. (2023, septiembre 12). Immune Technology Institute. https://immune.institute/blog/formato-xml