

**IES Carrillo Curso 24/25**

**Desarrollo de aplicaciones multiplataforma**

**Equipo 04:**

**Pedro González Martín**

**Antonio Gómez Camarena**

**David Castro Soriano**

**Tema 02 – Boletín 02**

indice

[Apartado a) 2](#_Toc180861039)

[Generar y Analizar el código generado 2](#_Toc180861040)

[Exportar el código generado 2](#_Toc180861041)

[Analiza y explica el código 2](#_Toc180861042)

[Apartado b) 6](#_Toc180861043)

[Realizar modificaciones al código 6](#_Toc180861044)

[Seleccionar áreas del código 6](#_Toc180861045)

[Modificar el código para personalizar 6](#_Toc180861046)

[Probar los cambios realizados. 11](#_Toc180861047)

# Apartado a)

## *Generar y Analizar el código generado por el editor visual para la interfaz gráfica*.

## Exportar el código generado por Qt Designer a un archivo .ui.

Este apartado está explicado en el mismo “.ui” . Subido a GitHub.

## Analiza y explica el código, tanto su estructura (como está estructurado y organizado) como sus elementos.

Estructura y elementos de InicioSesión.ui:

1. Estructura General

* Ventana Principal: QMainWindow llamada "MainWindow", actuando como el contenedor principal, con propiedades para tamaño (geometry), título (windowTitle = "MainWindow"), y estilo CSS (styleSheet) para un fondo azul claro (rgb(65, 130, 195)).
* Widget Central: Dentro de "MainWindow" se encuentra "centralwidget", un QWidget que contiene el contenido de la ventana principal. Su estilo display: flex; justify-content: center; centra los elementos en la ventana.

2. Proceso de Creación Inicial

* Contenedor y Layout Vertical: "centralwidget" organiza el contenido verticalmente mediante un QVBoxLayout llamado "verticalLayout\_2", que facilita la alineación en bloques verticales.
* Frame de Inicio de Sesión: Dentro de "centralwidget", el QFrame llamado "frame" contiene los elementos de inicio de sesión. Tiene esquinas redondeadas (border-radius: 15px) y un fondo azul oscuro (rgb(40, 81, 121)), definido en su styleSheet.

3. Elementos Específicos y Modificaciones Realizadas

* Título Principal (QLabel “Iniciar sesión”): QLabel llamado "labelIniciarSesion" con el texto "Iniciar sesión", fuente Segoe UI Black de tamaño 29 y estilo CSS que incluye letter-spacing:5px para espaciado entre letras.
* Separador (Line): Un widget Line llamado "lineIniciarSesion" que funciona como separador visual horizontal, con un fondo blanco (rgb(255, 255, 255)).

Campos de Entrada (QLineEdit):

* Usuario/Correo: Este campo de entrada es un QLineEdit llamado "textoUsuarioCorreo" para capturar el usuario o correo. Su estilo (styleSheet) establece un fondo magenta claro (#e200ff) con texto negro y usa una fuente Comic Sans, tamaño 18, en cursiva.
* Contraseña: Otro QLineEdit llamado "textoContrasenna" para introducir la contraseña. Configurado con echoMode en Password, fondo blanco y texto negro.

Botones (QPushButton):

* Iniciar sesión: QPushButton llamado "botonIniciarSesion", con texto "Iniciar sesión" en fuente Arial, tamaño 20, en negrita, y un fondo verde claro (#11ff00). El cursor cambia a PointingHandCursor.
* Registrarse: QPushButton llamado "botonRegistrate" con el texto "Regístrate", fondo naranja (#f2784b) y fuente Calibri de tamaño 14 y en negrita.

Separadores y Espaciadores:

* Espaciadores Verticales y Horizontales: QSpacerItems como "verticalSpacer" y "horizontalSpacer\_2", que mantienen la separación entre los componentes, como entre el título y los campos de entrada.

4. Menús y Acciones

* Barra de Menú: QMenuBar llamada "menubar" con fondo azul oscuro (rgb(40, 81, 121)) y fuente Arial de tamaño 10.

Menús y Submenús:

* Archivo: QMenu llamado "menuOpciones" con un submenú "menuAcerca\_de" y acciones como "actionAcerdaDe" para “Vaciar campos de texto” y "actionNuestra\_empresa" para mostrar “Nuestra empresa”.
* Iniciar sesión y Registrarse: "menuIniciar\_sesi\_n" y "menuRegistrarse" para navegar entre las opciones de sesión y registro.

Estructura y elementos de Registro.ui :

1. Estructura General

* Ventana Principal: La ventana es un QMainWindow llamado "MainWindow", el contenedor principal. Tiene propiedades de tamaño (geometry), título (windowTitle), y un estilo CSS (styleSheet) que establece un fondo azul claro (rgb(65, 130, 195)).
* Widget Central: Dentro de "MainWindow" está "centralwidget", un QWidget que contiene el contenido principal de la ventana. Su estilo incluye display: flex; justify-content: center; para centrar los elementos en la interfaz.

2. Proceso de Creación Inicial

* Contenedor y Layout Vertical: "centralwidget" organiza el contenido verticalmente con un QVBoxLayout llamado "verticalLayout\_2", ideal para alinear los elementos en bloques verticales.
* Frame de Registro: Dentro de "centralwidget" se encuentra un QFrame llamado "frame" que contiene todos los elementos del formulario de registro. Este frame tiene esquinas redondeadas (border-radius: 15px) y un fondo azul oscuro (rgb(40, 81, 121)).

3. Elementos Específicos y Modificaciones Realizadas

* Título Principal (QLabel “Registro”): QLabel llamado "labelRegistro" que muestra el texto "Registro", con fuente Segoe UI Black de tamaño tamaño 29, alineado al centro en la parte superior (AlignHCenter | AlignTop).
* Separador (Line): Un widget Line llamado "lineRegistro" que sirve como separador visual horizontal, con fondo blanco (rgb(255, 255, 255)).

Campos de Entrada de texto (QLineEdit):

* Usuario: QLineEdit llamado "editUsuario" para capturar el nombre de usuario. Su estilo (styleSheet) establece un fondo blanco y texto negro, con fuente Calibri de tamaño 14.
* Correo: QLineEdit llamado "editCorreo" para el correo electrónico del usuario. Tiene un fondo blanco y texto negro, con fuente Calibri de tamaño 14.
* Contraseña: QLineEdit llamado "editContrasenna" para la contraseña. Configurado con echoMode en Password para ocultar los caracteres y un fondo blanco con texto negro.
* Repetir Contraseña: QLineEdit llamado "editRContrasenna" para confirmar la contraseña, también configurado con echoMode en Password, fondo blanco y texto negro.

Etiquetas (QLabel) para Identificar Campos:

* Usuario: QLabel llamado "labelUsuario" con texto "Usuario", usando fuente Calibri de tamaño 14.
* Correo: QLabel llamado "labelCorreo" con texto "Correo", fuente Calibri de tamaño 14.
* Contraseña: QLabel llamado "labelContrasenna" con texto "Contraseña", fuente Calibri de tamaño 14.
* Repetir Contraseña: QLabel llamado "labelRepetirContrasenna" con texto "Repetir contraseña", fuente Calibri de tamaño 14.

Botón de Registro (QPushButton):

* Regístrate: QPushButton llamado "btnRegistro" con texto "Regístrate", fondo naranja (#f2784b), y fuente Calibri de tamaño 14 en negrita. El cursor cambia a PointingHandCursor al pasar sobre el botón.

4. Menús y Acciones

* Barra de Menú: QMenuBar llamado "menubar" con fondo azul oscuro (rgb(40, 81, 121)) y fuente Arial de tamaño 10.

Menús y Submenús:

* Archivo: QMenu llamado "menuOpciones" con el título "Archivo", que contiene un submenú "menuAcerca\_de" y la acción "actionIniciar\_sesion" para vaciar los campos de texto y "actionNuestra\_empresa" para mostrar “Nuestra empresa”.
* Iniciar sesión y Regístrate: "menuIniciar\_sesion" y "menuRegistrate" con opciones para navegar entre el registro y el inicio de sesión.

# Apartado b)

# Realizar modificaciones al código generado por el editor visual (a ambos archivos .ui), ajustando aspectos funcionales y/o estéticos de la interfaz.

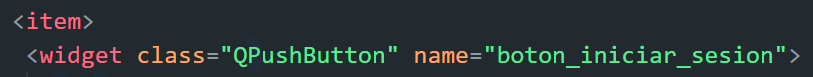
## Seleccionar áreas del código que requieren ajustes (por ejemplo, propiedades de widgets, señales, etc.).

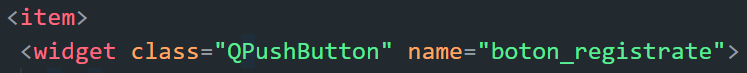
Vimos que de cara al código Python y por hacer más fácil la programación de acción de botones y selección de campo de texto, decidimos cambiar los nombres de los elementos de modo que sean más identificativos.

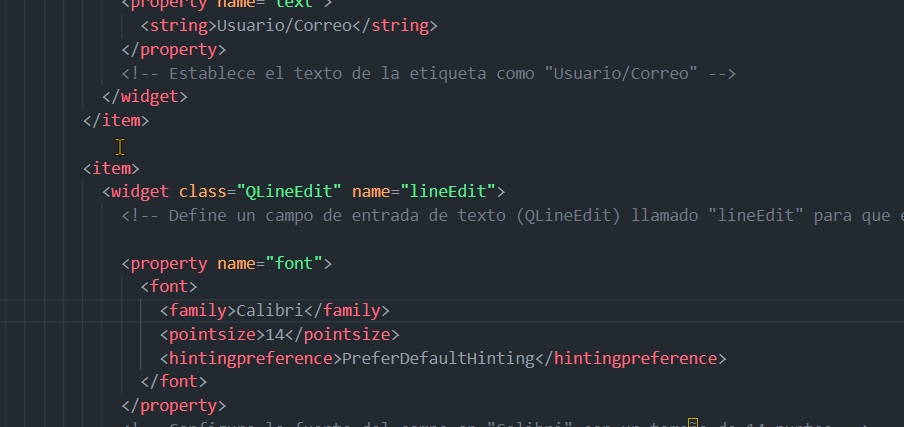
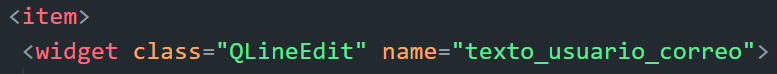
## Modificar el código para personalizar el comportamiento de los componentes.

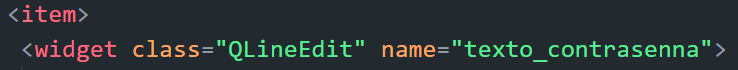
Pantalla inicio de sesión:

Botón iniciar sesión:

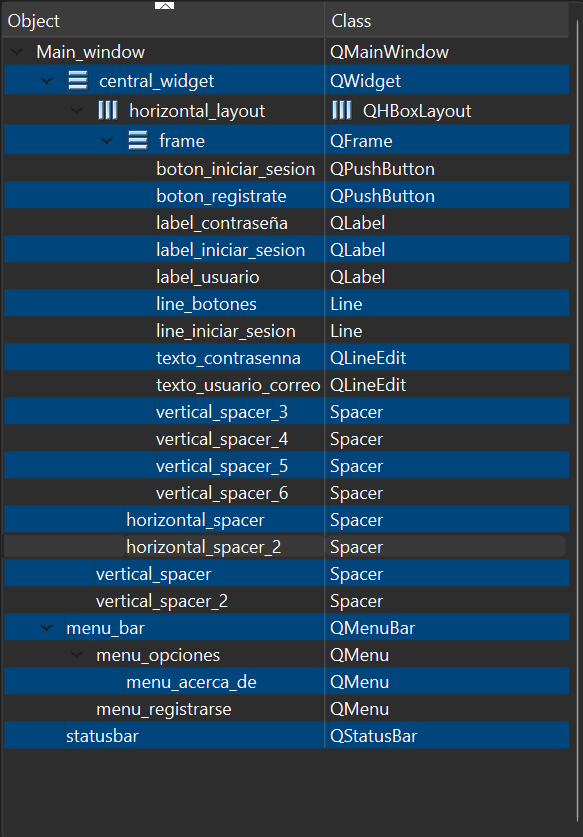


Botón Regístrate:

Campos de texto “QLineEdit”:

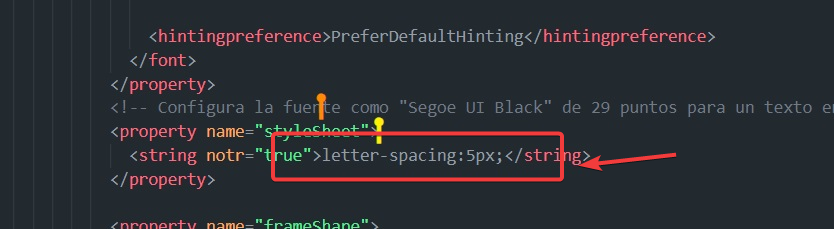


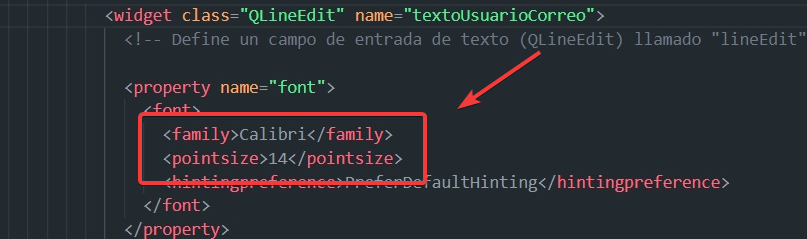
Así hicimos con todos los campos que queríamos diferenciar.

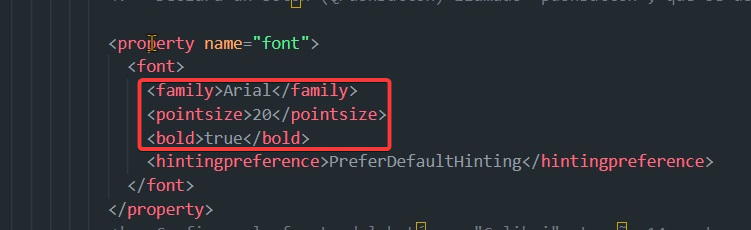
 Pantalla Registro Pantalla inicio sesión:



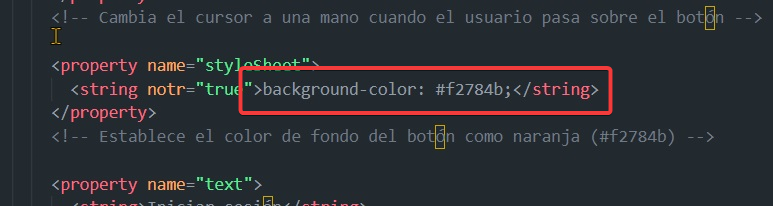
Pusimos especiado entre las letras de iniciar sesión:

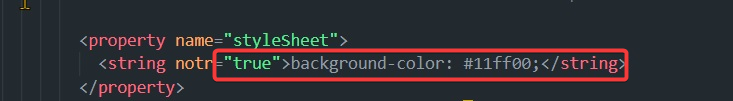
Cambiamos el tipo de letras, de calibri 14 a Arial 20.



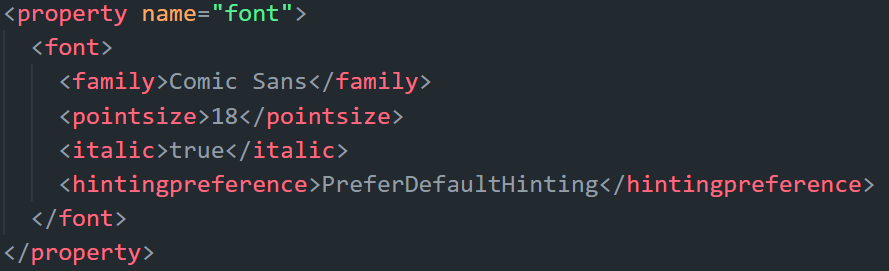


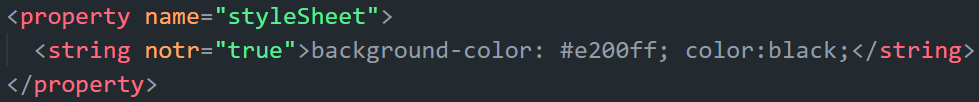
Cambio de color en botón inicio sesión, le cambiamos el #f2784b (naranja) por #11ff00 (verde).





Cambiamos lineEdit texto usuario/correo, cambiamos la fuente comic sans y cursiva, tamaño 18. Tambien le cambiamos el fondo.



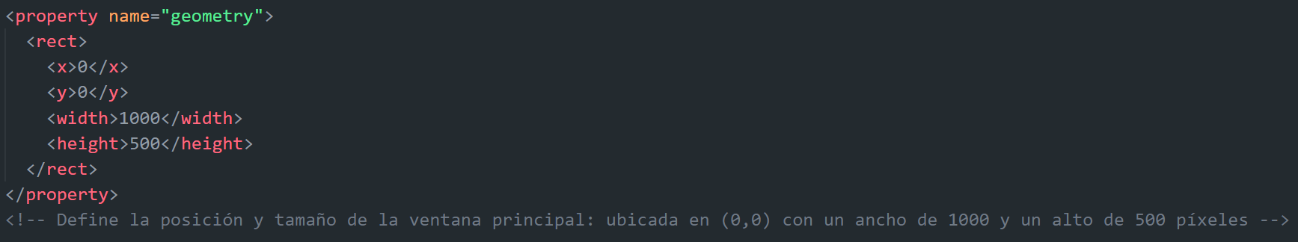


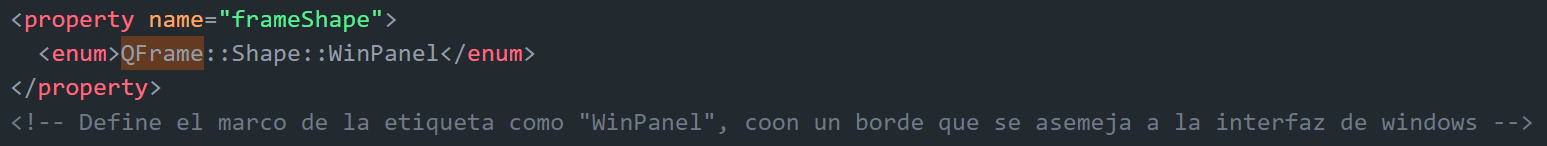
Pantalla Registro:

Se ha añadido un nuevo item, un botón “adicional”.

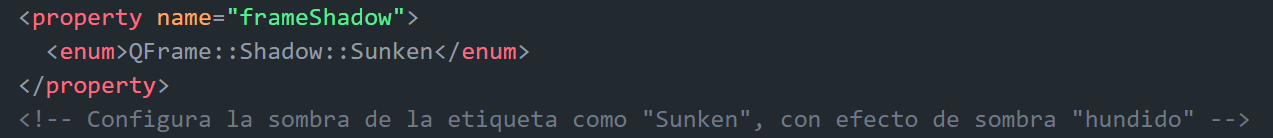


Modificamos el ancho y alto a la pantalla principal “QMainWindow”.

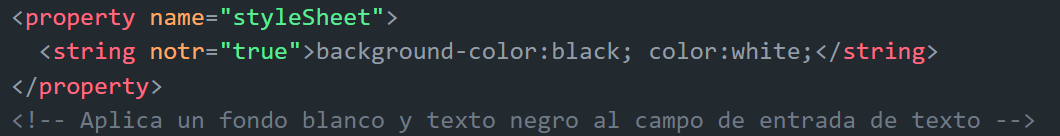
Editamos la propiedad frameshape para poner un borde que se asemeje a la interfaz de Windows al texto REGISTRO.



También editamos la propiedad frameshadow y le añadimos un sombreado con efecto “hundido”.



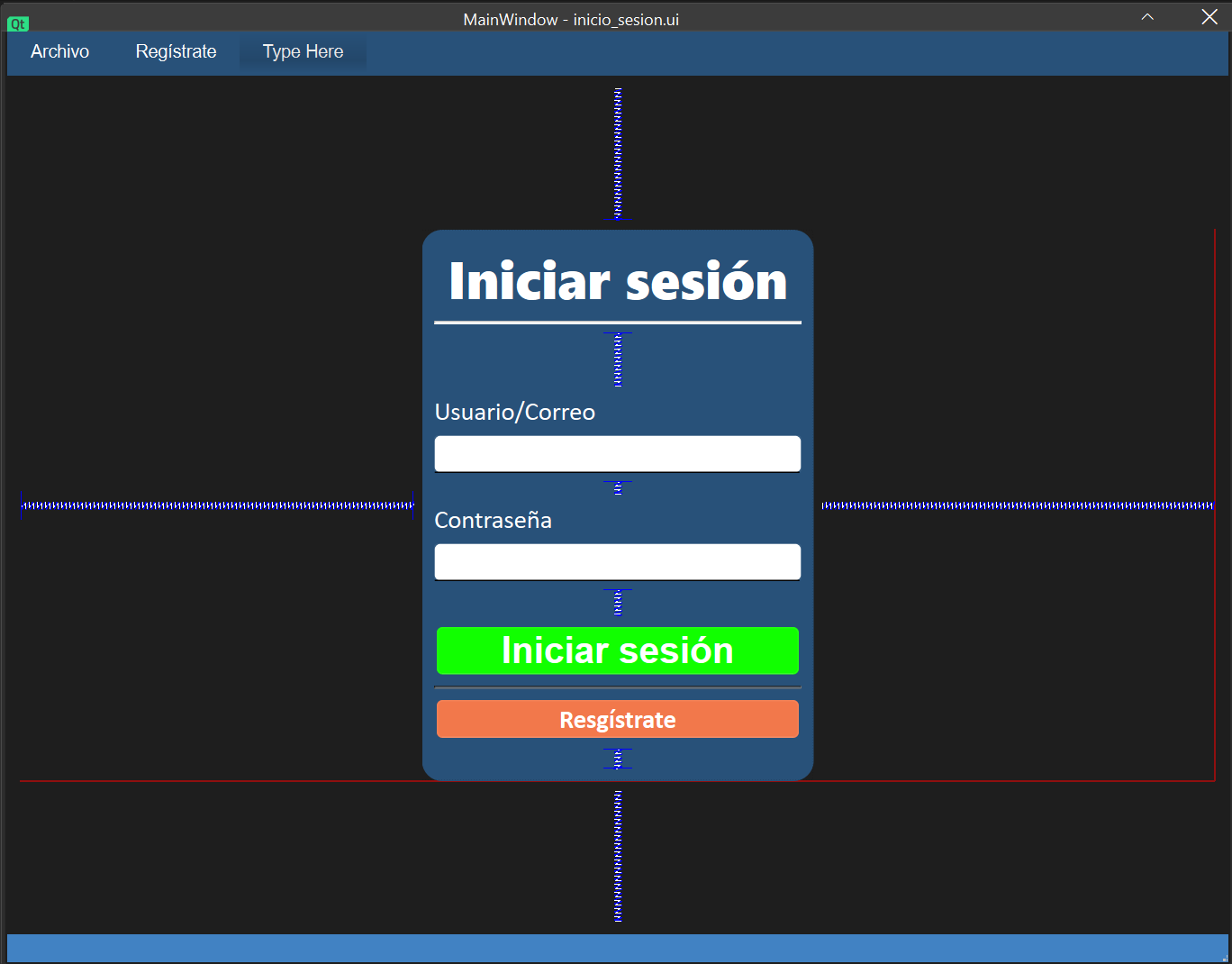
Por último, hemos modificado los line edit para que el fondo sea negro y el texto en blanco.



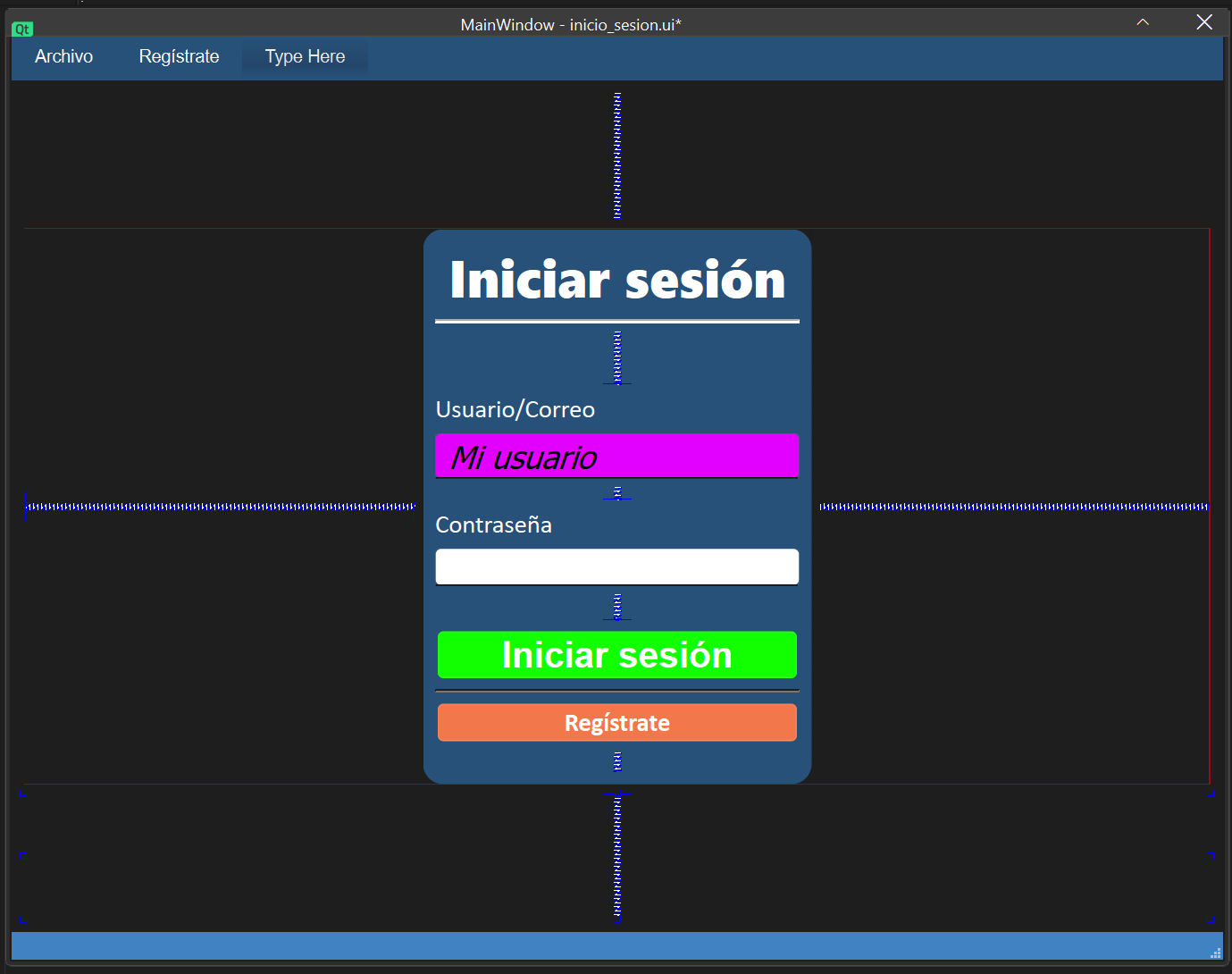
## Probar los cambios realizados mostrándolo mediante Qt Designer y comprobando los cambios gráficamente (también visualizar dichas propiedades para ver que han cambiado).

Pantalla inicio Sesion:

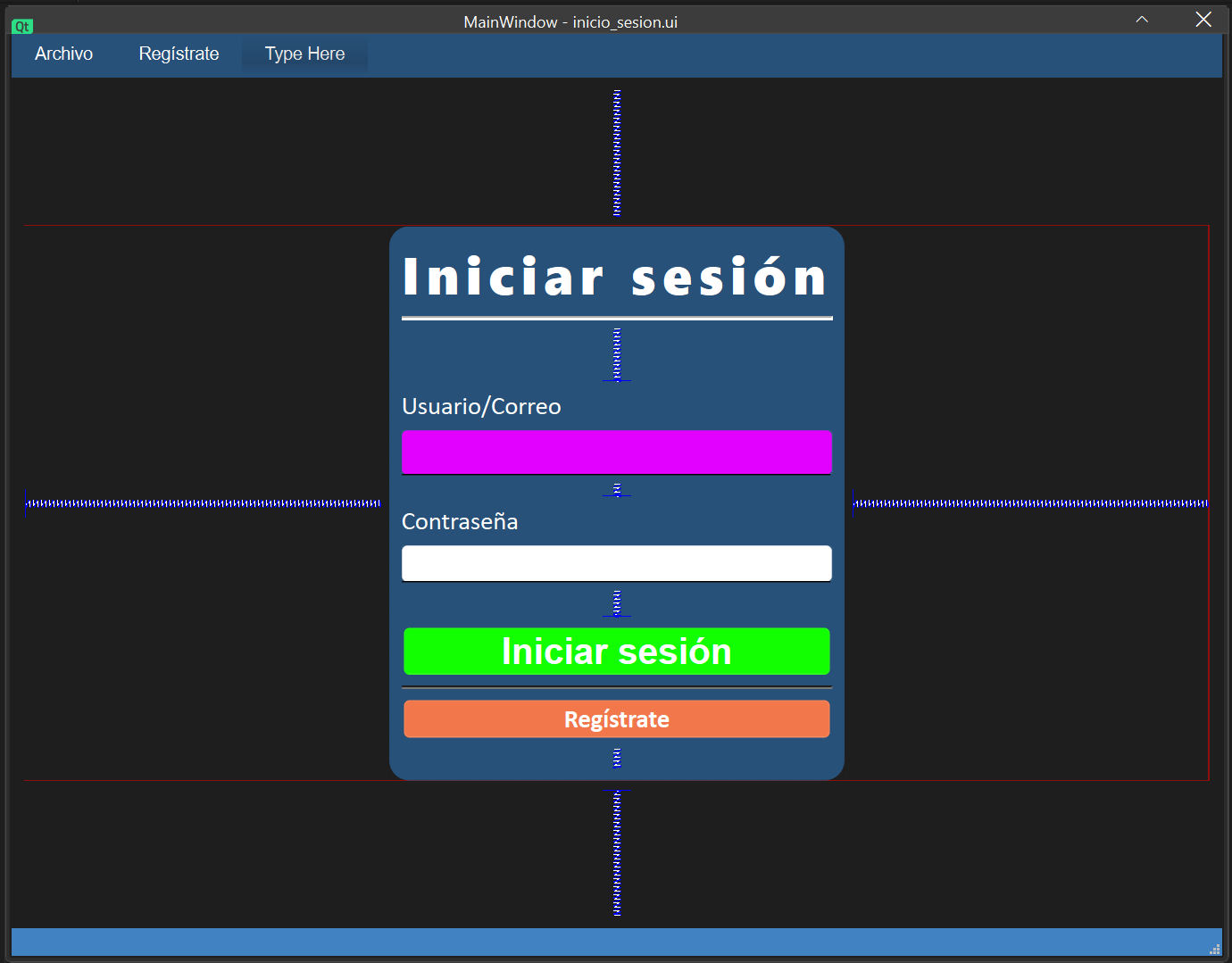
Los cambios realizados en el botón iniciar sesión:



Cambios realizados en el cuadro de texto de usuario/correo:

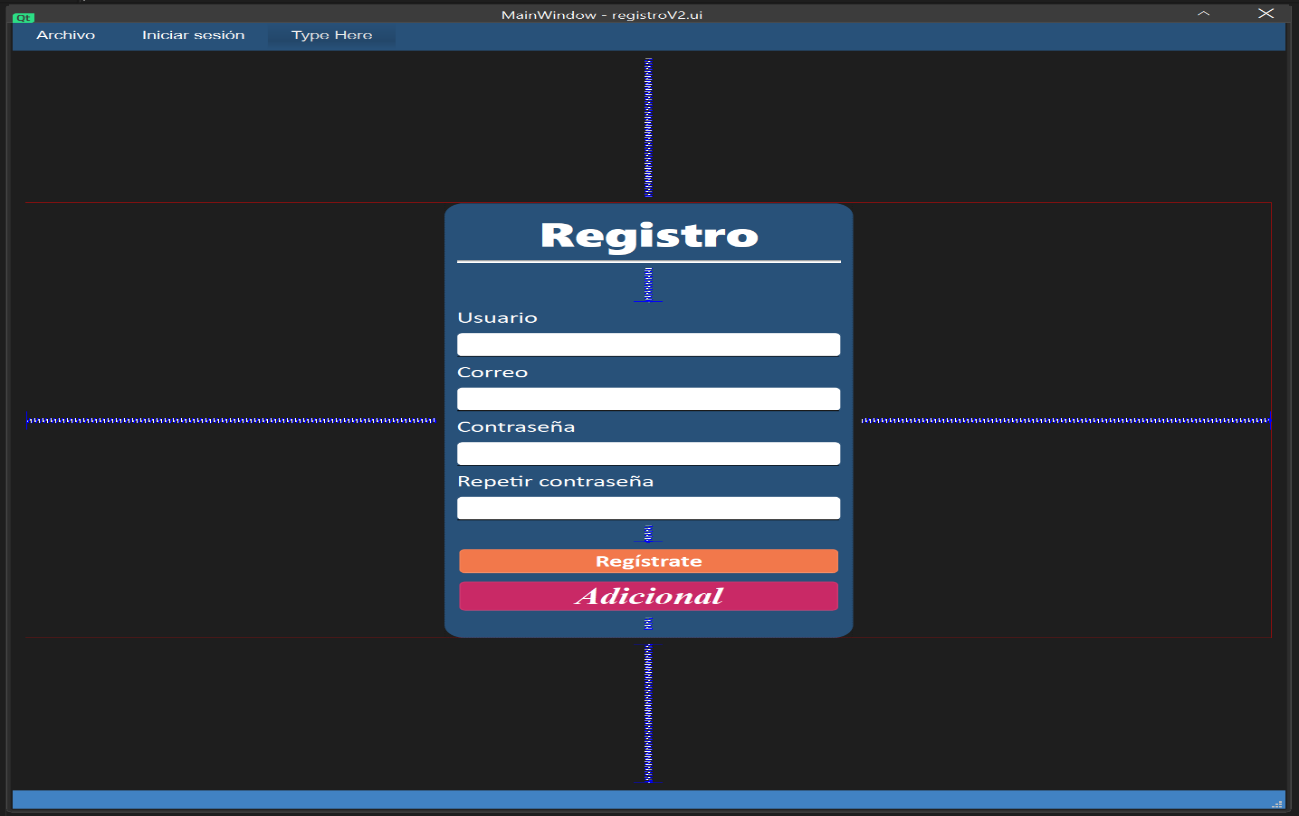


Espaciado de letra en iniciar sesión:



Pantalla de Registro.

Añadimos el botón “adicional” con sus características:



Modificamos el ancho y alto de la qmainwindow a 1000 x 500:



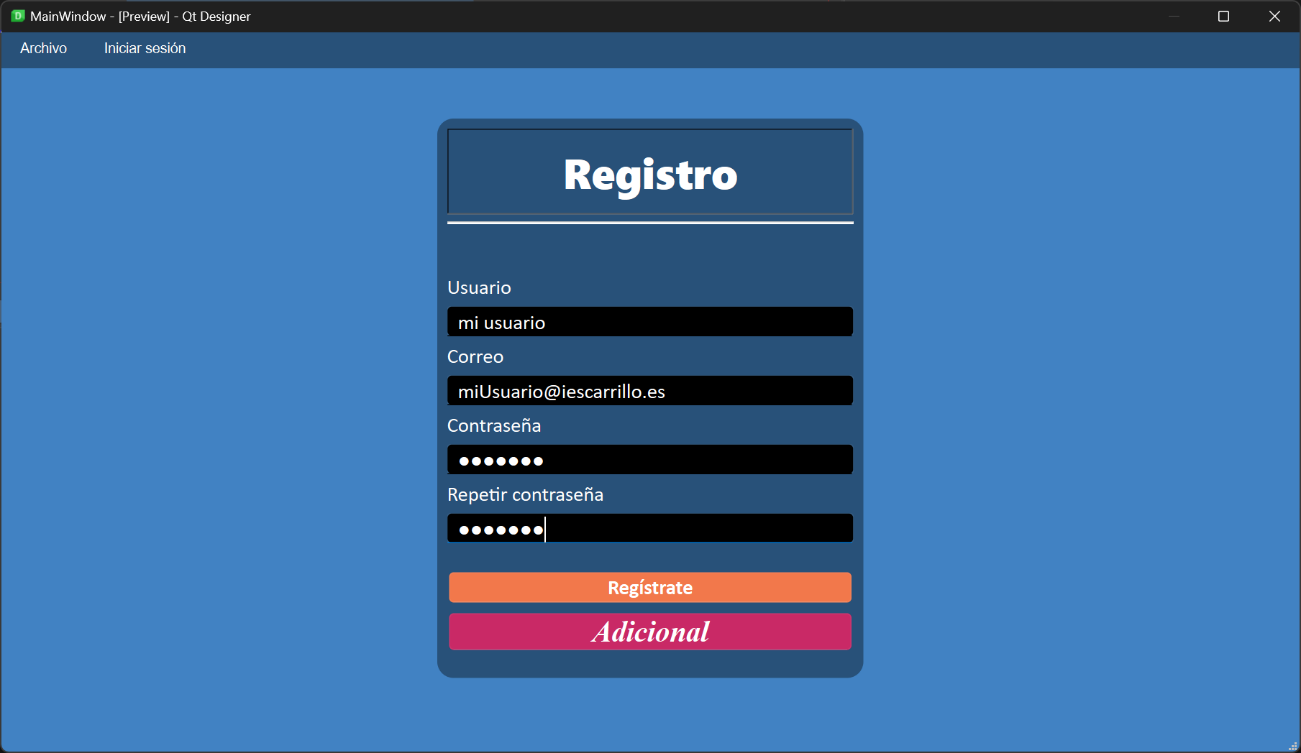
Añadimos un borde a registro:



Añadimos sombreado efecto “hundido” registro:



Modificamos la letra y el fondo de los line edit:



\*Estas modificaciones no son permanentes porque consideramos que el diseño inicial era adecuado, pero las hemos realizado para que se vean que desde el xml se pueden realizar cambios.

# Reconocer y documentar las ventajas de generar interfaces gráficas utilizando XML.

## Investigar las principales ventajas de utilizar XML para la generación de interfaces de usuario.

ofrece varias ventajas para la generación de interfaces de usuario, las principales son las siguientes:

1.**Estructura y legibilidad**: XML proporciona una estructura jerárquica que facilita la organización de datos, lo que hace que el contenido sea más fácil de leer y comprender. Esto es especialmente útil para los desarrolladores que trabajan con interfaces complejas.

2.**Separación de contenido y presentación**: Al utilizar XML, puedes separar los datos (contenido) de la lógica de presentación (cómo se muestra). Esto permite que los cambios en la interfaz no afecten a los datos subyacentes y viceversa, facilitando el mantenimiento y la actualización de la aplicación.

3.**Interoperabilidad**: XML es un estándar ampliamente aceptado y soportado por muchas plataformas y lenguajes de programación. Esto permite que diferentes sistemas y aplicaciones intercambien información de manera efectiva.

4.**Flexibilidad**: Al ser extensible, XML permite definir etiquetas personalizadas que se adaptan a las necesidades específicas de la aplicación. Esto permite crear estructuras de datos que se alineen perfectamente con los requisitos del proyecto.

5.**Compatibilidad con herramientas**: Existen muchas herramientas y bibliotecas que facilitan el trabajo con XML, desde la creación hasta la manipulación de datos.

6.**Soporte para múltiples lenguajes de programación**: XML se puede utilizar con una amplia variedad de lenguajes de programación, lo que permite a los desarrolladores utilizar el lenguaje que prefieran para manipular y procesar datos XML.

7.**Validación de datos**: XML permite definir esquemas (como DTD o XML Schema) que pueden validar la estructura de los documentos. Esto ayuda a garantizar que los datos que se procesan son válidos y cumplen con las especificaciones esperadas.

**Facilidad de integración**: Las interfaces de usuario construidas con XML pueden integrarse fácilmente con otros servicios web y APIs, ya que muchas tecnologías web utilizan XML o JSON (que es más fácil de manipular a partir de XML) como formato de intercambio de datos.

La información la hemos sacado de:

* [**Tech Quintal**](https://www.techquintal.com/advantages-and-disadvantages-of-xml/) **- Ventajas y desventajas de XML**
* [**ThoughtCo**](https://www.thoughtco.com/reasons-to-use-xml-3471386) **- Razones básicas para usar XML:**
* [**IBM**](https://www.ibm.com/docs/en/i/7.3?topic=introduction-advantages-xml) - **Ventajas de XML**:
* [**GeeksforGeeks**](https://www.thoughtco.com/reasons-to-use-xml-3471386) - **Usos de XML**:
* [**Codecademy**](https://www.thoughtco.com/reasons-to-use-xml-3471386) - **Información sobre XML**

## Comparar el uso de XML frente a otras técnicas para generar interfaces gráficas.

**1. XML frente a HTML**

* **Estructura y organización**: XML permite definir etiquetas personalizadas que pueden describir datos de manera más precisa y flexible que HTML. Mientras que HTML tiene un conjunto fijo de etiquetas para presentar contenido en la web, XML permite crear etiquetas específicas que se ajusten a las necesidades del desarrollador.
* **Separación de contenido y presentación**: XML facilita la separación del contenido (datos) de su presentación (cómo se muestra), lo que no es tan sencillo con HTML, donde el contenido y la presentación a menudo están entrelazados. Esto permite un mejor mantenimiento y reutilización del código.
* **Legibilidad**: XML es legible tanto para humanos como para máquinas, lo que facilita el trabajo colaborativo y la comprensión del código. HTML también es legible, pero sus etiquetas no siempre son descriptivas.

**2. XML frente a JSON**

* **Formato y uso**: XML es más verboso que JSON, lo que puede llevar a un mayor tamaño de los archivos y a un mayor uso de ancho de banda en la transmisión de datos. JSON, por otro lado, es más ligero y se utiliza comúnmente en aplicaciones web modernas, especialmente en comunicación entre clientes y servidores.
* **Tipo de datos**: JSON se presta mejor para la representación de estructuras de datos como listas y objetos, mientras que XML es más adecuado para datos que requieren una estructura jerárquica compleja y la posibilidad de incluir metadatos mediante atributos.
* **Interoperabilidad**: Ambos formatos son interoperables, pero JSON tiende a ser más popular en entornos de desarrollo de aplicaciones web modernas, especialmente con JavaScript. XML sigue siendo ampliamente utilizado en sistemas más antiguos y en aplicaciones que requieren interoperabilidad con otros estándares como SOAP.

**3. XML frente a Frameworks de UI (ej. React, Angular)**

* **Flexibilidad**: XML proporciona una forma sencilla de definir la estructura de datos, pero no ofrece las funcionalidades avanzadas de los frameworks modernos que permiten la creación de interfaces dinámicas y responsivas. Frameworks como React y Angular permiten a los desarrolladores crear componentes reutilizables, manejar estados y gestionar la interacción del usuario de manera más eficiente.
* **Desempeño**: Los frameworks de UI suelen tener un mejor desempeño y pueden manejar grandes volúmenes de datos y operaciones de manera más eficiente que XML puro. Utilizan técnicas de virtual DOM y optimización de renderizado que son más efectivas que las basadas en XML.
* **Experiencia de desarrollo**: Los frameworks modernos ofrecen herramientas y ecosistemas robustos (como bibliotecas de componentes, gestión de estado, etc.) que facilitan el desarrollo. XML, por otro lado, requiere más configuración y manejo manual, lo que puede aumentar la carga de trabajo del desarrollador.

## Redactar las conclusiones obtenidas.

Cuando se compara XML con otras formas de crear interfaces gráficas, se puede ver que XML tiene ventajas como una estructura bien organizada, fácil de leer y que separa bien el contenido del diseño. En aplicaciones más modernas, a veces se prefiere usar opciones como JSON, que es más ligero y rápido, o frameworks como React y Angular, que ofrecen más flexibilidad y rendimiento.

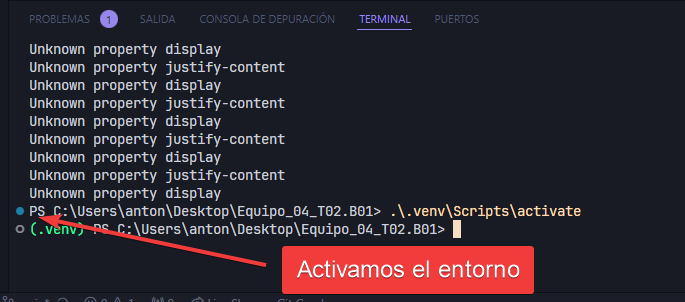
Elegir entre XML y otras alternativas depende mucho de lo que necesite el proyecto. Si la aplicación requiere una estructura clara y organizada o necesita trabajar con metadatos, XML puede ser una buena opción. En cambio, si estamos hablando de aplicaciones web modernas, que necesitan ser rápidas y tener componentes que se actualicen de forma dinámica, los frameworks de UI o JSON suelen ser mejores opciones.

# Apartado d)

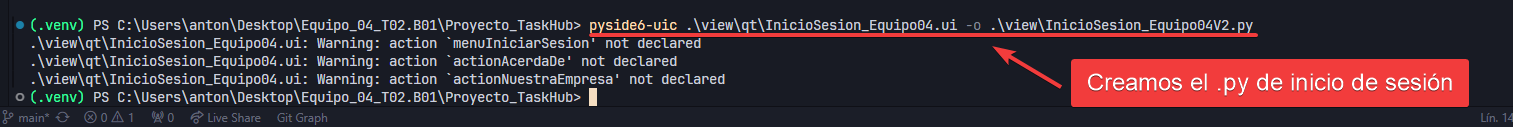
## Generar el código de la interfaz en Python a partir de los archivos .ui (XML) y verificar su funcionalidad.

## Utilizar las herramientas necesarias para generar el código Python desde los archivos .ui (que es la interfaz de la aplicación en código XML).

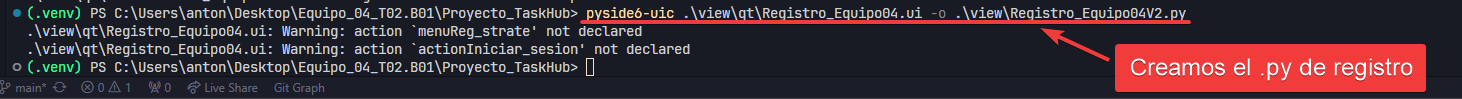
1.



2.

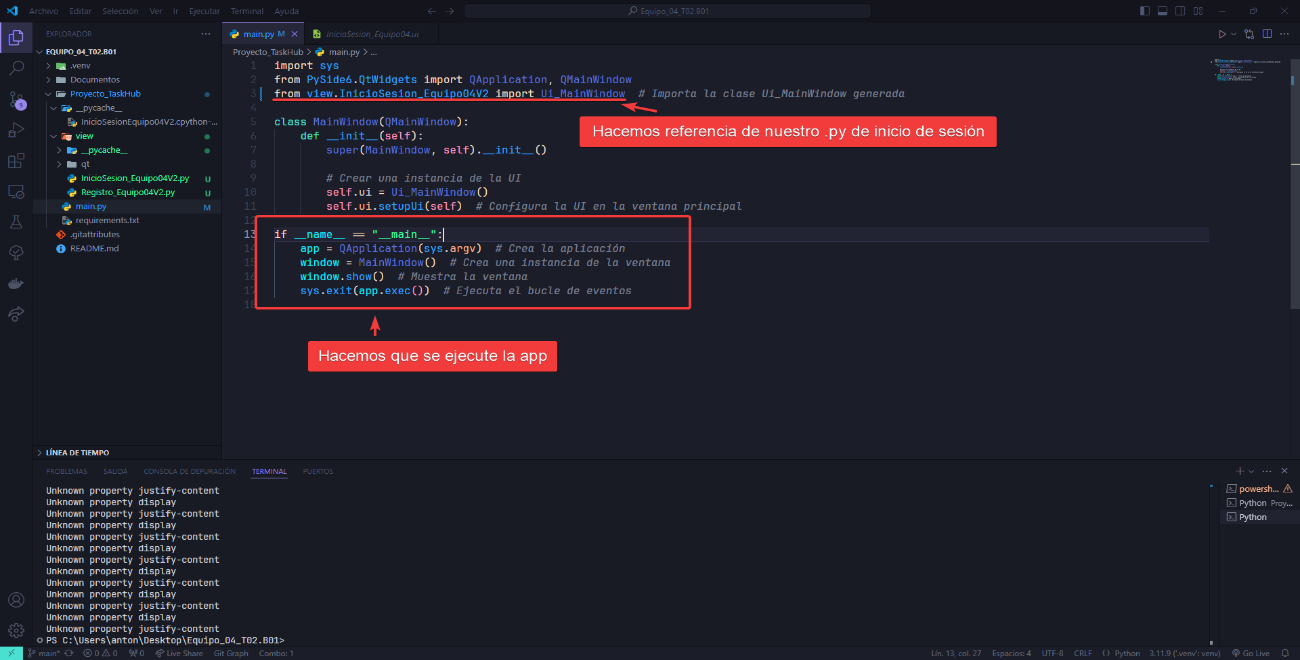


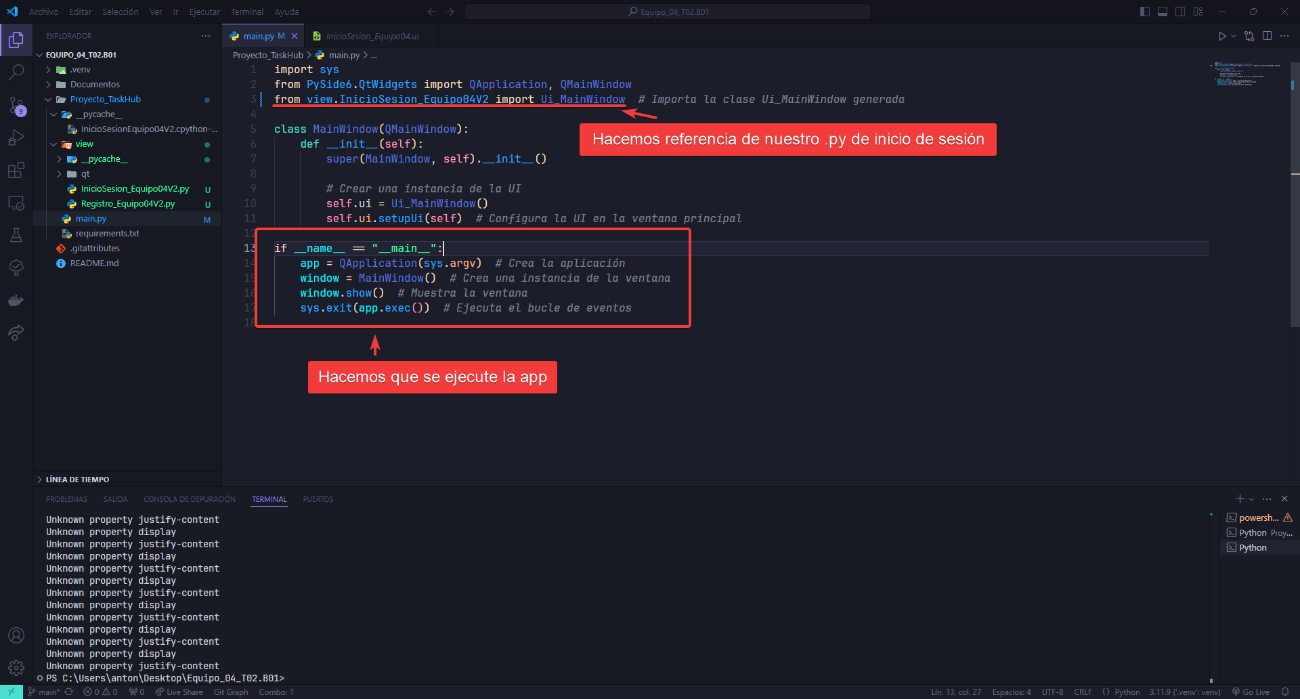
3.

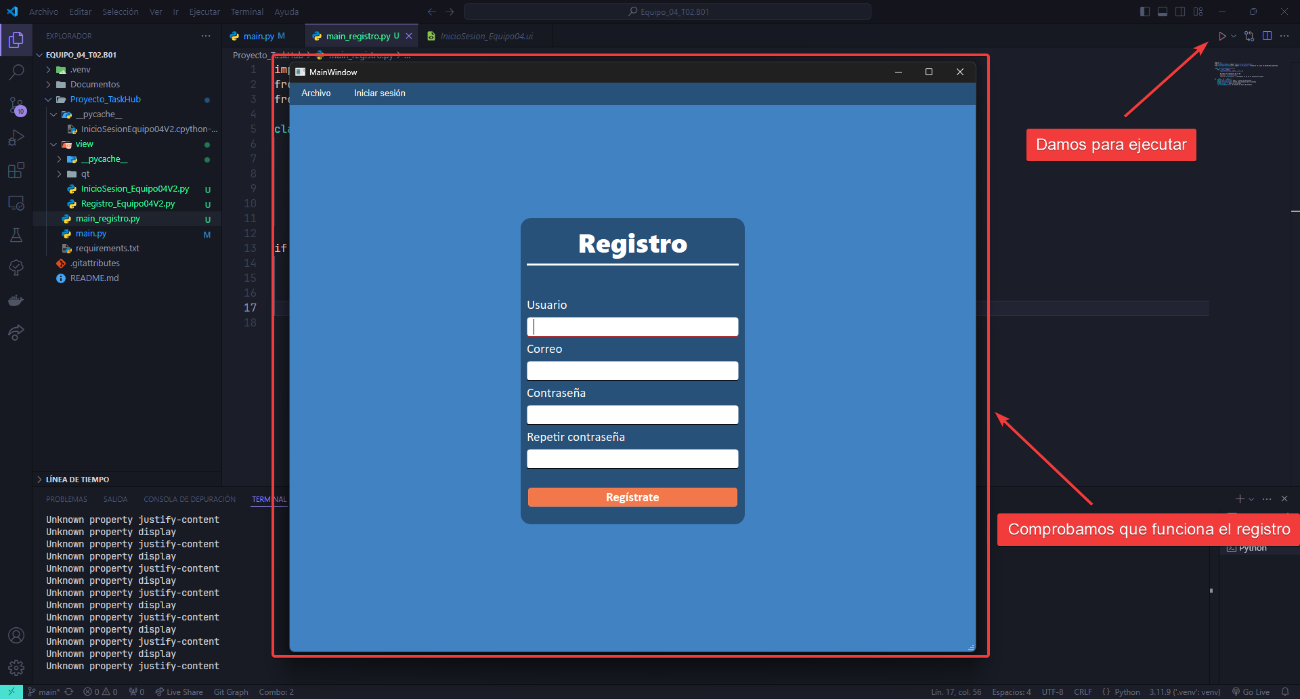


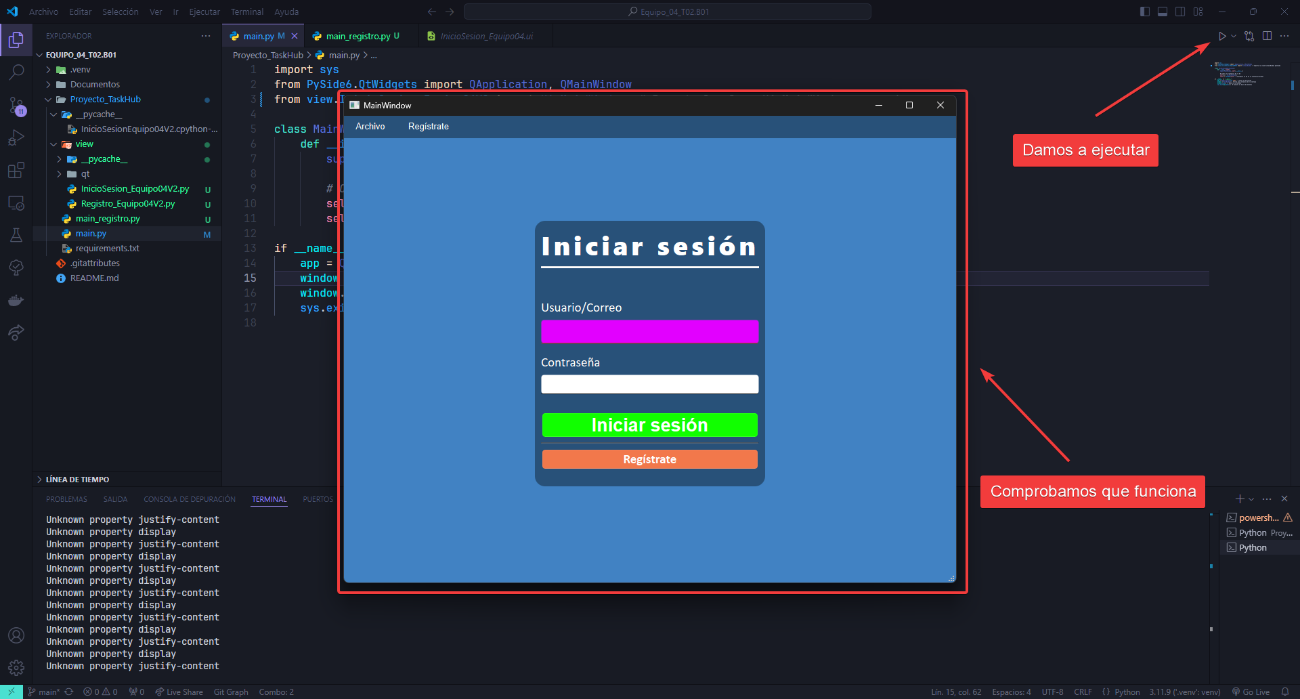
## Comprobar que el código generado funciona correctamente al ejecutarlo en la aplicación.

Para ejecutar el .py genarado hemos creado un main que al ejercutarlo muestre ambas pantallas.









## Realizar ajustes si es necesario para garantizar que la interfaz funcione según lo esperado.

Para que funcione hemos creado un main para cada clase el cual tendremos que modificar mas adelante para conectarlo con la funcionalidad que le demos.