Javier Martínez Jaén

-Día 1

Comienzo buscando información acerca de las distintas herramientas y lenguajes que se proponen para el desarrollo del editor de texto.

<http://joedicastro.com/pages/markdown.html>

<https://doc.qt.io/qtforpython/PySide6/QtCore/QTranslator.html>

<https://doc.qt.io/qtforpython-5/PySide2/QtWidgets/QUndoCommand.htm>l

<https://doc.qt.io/qtforpython/overviews/model-view-programming.html>

<https://www.pythonguis.com/tutorials/modelview-architecture/>

<https://ernestocrespo13.wordpress.com/2011/08/14/tutorial-de-pyqt-editor-de-archivos-sencillo-parte-10/>

<https://linuxhint.com/how-to-write-a-simple-text-editor-in-pyqt5/>

<https://github.com/ralsina/pyqt-by-example>

<https://www.delftstack.com/es/tutorial/pyqt5/pyqt5-menubar/>

<https://acodigo.blogspot.com/2016/07/pyqt-5-cuadros-de-dialogo.html>

<https://www.delftstack.com/es/howto/python/get-filename-from-path-in-python/>

<https://realpython.com/python-menus-toolbars/>

<https://pythonprogramming.net/open-files-pyqt-tutorial/>

<https://www.tutorialspoint.com/pyqt5/pyqt5_quick_guide.htm>

<https://stackoverflow.com/questions/52656526/how-to-insert-a-web-browser-in-python-qt-designer>

<https://github.com/riverzhou/pyqt5webmd>

Empiezo a desarrollar la gui con funcionalidad mínima basándome en los tutoriales encontrados, utilizaré Visual Studio Code como IDE.

Foto del inicio del proyecto, he usado 5 fuentes de información diferentes para la realización:

Una captura de pantalla de una computadora

Descripción generada automáticamente

Añado la función de guardado, y, tras familiarizarme un poco más con la librería PyQt5 procedo a intentar adaptar mi código al modelo MV propuesto previamente. He tenido algún problema a la hora de implementar la menubar pero lo he solucionado rápidamente. La menubar está implementada a falta de añadir la funcionalidad de los diferentes elementos.

Horas día 1: 5h aproximadamente, entre búsqueda de información, toma de contacto con las librerías y comienzo del prototipo.

-Día 2:

Creo mi toolbar para posteriormente añadir las funcionalidades. Añado iconos a la toolbar para hacerla mas orientativa visualmente y cómoda, con funciones nuevas. Una vez añadidos los botones para realizar las diferentes funciones, procedo a desarrollar su usabilidad.

Interfaz de usuario gráfica

Descripción generada automáticamente

Horas día 2: 2’5h

-Día 3: añado las funcionalidades abrir archivo, guardar archivo y crear archivo nuevo, todas funcionan, pero abrir archivo y crear archivo dan error cuando cancelas la acción a la hora de especificar una ruta.

Horas del día 3: 0.5h

-Día 4: comienzo a implementar las funcionalidades copiar, pegar, cortar, deshacer y rehacer. Encuentro diversos problemas con el portapapeles de Windows y el de la propia app.

Horas del día 4: 2h

-Día 5: encuentro la manera de solucionar las funciones copiar y cortar, utilizando QtGui.QTextCursor. Añado la funcionalidad pegar. He completado mi código con un visualizador de markdown en formato HTML, teniendo ahora dos aplicaciones. Mi próximo paso es añadir la app de visualización de markdown a mi aplicación de edición. Añado la visualización de mi archivo markdown en html en paralelo con reacción a los cambios.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Horas día 5: 5h

-Día 6: comienzo añadiendo las funcionalidades de texto Bold, Italic, H1, H2, H3; he tenido que quitar cursiva y subrayado ya que no existen en markdown. Completo la traducción con QTraslator(), estableciendo por defecto el idioma en\_EN y actualizándolo según el idioma del dispositivo, en mi caso detecta es\_ES. Inicio además la implementación de QUndoCommand()

Inserto imagen del editor en la fase actual, con un archivo md de ejemplo:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Paro de implementar QUndoCommand() para comenzar a dividir el proyecto en la arquitectura propuesta.

Completo la primera parte de la actualización de arquitectura.

Horas días 6: 5h

-Día 7:

Continúo realizando el cambio de arquitectura de la aplicación. Al llegar a la tercera adaptación, me falta adaptar tanto ToolBar como MenuBar, para dejar el archivo main.py más limpio y tenerlo todo organizado. Me estoy basando y ayudando del proyecto Proteus facilitado por Amador Durán.

Finalizo la adaptación.

Repositorio antes de cambiar la arquitectura:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Repositorio tras cambiar la arquitectura:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Corrijo un par de errores para finalizar la primera semana.

Horas día 7: 4h

Horas semana 1: 24

Media de horas por día: 3.4h

-Día 8:

Comienzo a realizar la documentación generada a partir del código, voy a usar Sphinx. Primero creo la carpeta docs dentro de mi directorio raíz, utilizo sphix-quickstart para inicializar sphinx dentro de esa carpeta, a continuación, realizo los cambios provenientes para establecer las rutas de los archivos rst a crear y la ruta para autodoc. El primer problema que encuentro es que a la hora de realizar la conversión a html no es capaz de encontrarme la ruta de mainwindow.ui, siendo este el único error que me da.

Horas día 8: 2h

-Día 9:

Tras realizar una serie de cambios y comprobaciones de rutas en el terminal, encuentro el problema por el cual no es capaz de encontrarme esa ruta, he intentado varias soluciones, pero la única funcional es temporal, es decir, se hace un cambio de ruta añadiendo dos puntos, para salir de un folder, y así encuentra el mainwindow.ui; tras realizar la documentación, vuelvo a dejar la ruta como estaba para hacer que el programa vuelva a funcionar.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Hasta ahora he conseguido implementar 3 de las 4 características propuestas por el profesor, las cuales son:

-Internacionalización mediante QTranslator: el programa está completamente hecho en inglés, pero al detectar que mi equipo tiene establecido el idioma español lo cambia automáticamente. Me he ayudado del prototipo de Proteus.

-Uso de la arquitectura Model/View: esta implementación debo corroborar con el profesor que es correcta, pero en principio debería serlo puesto que me he basado en el prototipo de Proteus.

-Documentación generada a partir de código: mediante el uso de Sphinx, como se propuso, he conseguido documentar automáticamente mi código.