Anexo V: Documentación técnica

Desarrollo de una cesta de compra con autodetección de productos orientado al sector retail

Trabajo de Fin de Grado

GRADO EN INGENIERÍA INFORMÁTICA

Julio de 2023

Autor

Pablo Santos Blázquez

Tutores

Sergio García González

Gabriel Villarrubia González

Tabla de contenidos

Índice de figuras

1) Introducción

Con este anexo se pretender desarrollar un documento de ayuda para la sencilla comprensión del código adjunto entregado junto con la documentación del proyecto. El principal objetivo es permitir que cualquier desarrollador con conocimientos en el ámbito informático sea capaz de entender el código del proyecto.

El contenido de este documento describe la estructura y las funcionalidades de cada uno de los distintos directorios y/o archivos entregados.

Para una descripción más detallada de las funciones implementadas se debe consultar la documentación realizada dentro de las clases desarrolladas (.cs) conformantes del propio código fuente o el archivo "*AutoPayDocumentation.xml*" disponible dentro del directorio:

"`~/AutoPay/Documentación/`"

En este documento no se describen los archivos presentados como documentación, pero sí la documentación del código. Para una descripción de dichos archivos debe revisarse el apartado de introducción del documento "*Memoria del proyecto*".

La estructura general de los archivos entregados para el correcto funcionamiento del sistema es la siguiente:

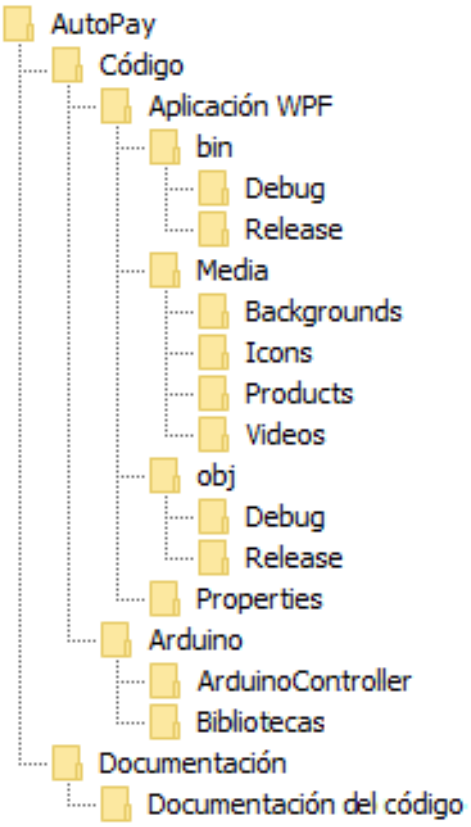

Figura 1. Estructura general del proyecto

2) AutoPay

En este apartado se describe la estructura y las funcionalidades de cada uno de los distintos directorios y/o archivos entregados como parte del proyecto.

Figura 2. Estructura del directorio AutoPay

Sus correspondientes directorios y/o archivos se encuentran en la dirección:

`“~/AutoPay/”`

Dentro de este directorio se encuentran dos subdirectorios:

- **Código**: contiene todos los directorios y/o archivos que forman parte del código fuente o son necesarios para su correcta ejecución.
- **Documentación**: contiene todos los archivos que forman parte de la documentación del proyecto.

3) Código

En este apartado se describe la estructura y las funcionalidades de cada uno de los distintos directorios y/o archivos que forman parte del código fuente o son necesarios para su correcta ejecución.

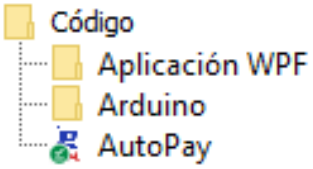

Figura 3. Estructura del directorio Código

Sus correspondientes directorios y/o archivos se encuentran en la dirección:

“~/AutoPay/Código”

Dentro de este directorio se encuentran dos subdirectorios:

- **Aplicación WPF**: contiene todos los directorios y/o archivos necesarios para la correcta ejecución de la aplicación WPF.
- **Arduino**: contiene todos los directorios y/o archivos necesarios para el correcto control de los distintos dispositivos mediante el uso del dispositivo embebido Arduino.

Dentro del directorio también se encuentra el siguiente archivo:

- **Autopay.exe (acceso directo)**: permite la ejecución directa de la versión “Release” del sistema software.

4) Aplicación WPF

En este apartado se describe la estructura y las funcionalidades de cada uno de los distintos directorios y/o archivos necesarios para el correcto funcionamiento de la aplicación WPF.

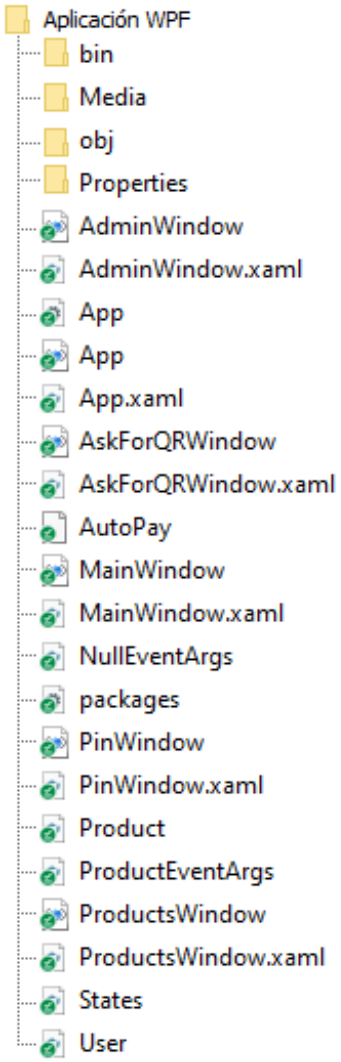

Figura 4. Estructura del directorio Aplicación WPF

Sus correspondientes directorios y/o archivos se encuentran en la dirección:

`“~/AutoPay/Código/Aplicación WPF/”`

Dentro de este directorio se encuentran tres subdirectorios:

- **bin**: contiene los archivos ejecutables (.exe) y bibliotecas de enlace dinámico (como los archivos .dll) necesarios para la correcta ejecución de la aplicación.
- **Media**: contiene los archivos gráficos utilizados por el sistema.
- **obj**: contiene los archivos temporales y de compilación utilizados internamente por el proceso de compilación (como archivos .obj y .pdb). No son necesarios para la ejecución de la aplicación.

- **Properties**: contiene las configuraciones y metadatos relacionados con el proyecto, como la información de ensamblado, las configuraciones de compilación y las configuraciones de recursos específicos de la aplicación WPF.

Dentro del directorio también se encuentran los archivos correspondientes a las clases implementadas:

- **AdminWindow.xaml**: contiene la estructura y definiciones de la interfaz gráfica, incluyendo la disposición de los controles, estilos, plantillas y enlaces de datos necesarios para la representación visual de la pantalla de estadísticas. Es mostrada únicamente a los administradores.
- **AdminWindow.xaml.cs**: contiene la lógica y funcionalidad de la ventana definida en su correspondiente archivo *"AdminWindow.xaml"*. En él se definen y manipulan los eventos, se accede a los controles de la interfaz gráfica y se realiza la lógica de negocio.
- **App.config**: contiene las configuraciones específicas de la aplicación, como cadenas de conexión a bases de datos, configuraciones de servicios o configuraciones de seguridad. Permite ajustar y personalizar el comportamiento y las características de la aplicación sin necesidad de recompilar el código fuente.
- **App.xaml**: define la clase de aplicación principal y proporciona configuraciones y recursos globales para toda la aplicación. En él se establecen estilos, plantillas y recursos que son compartidos por las distintas ventanas.
- **AskForQRWindow.xaml**: contiene la estructura y definiciones de la interfaz gráfica necesarias para la petición del QR a los usuarios. Es mostrada únicamente a los usuarios no identificados.
- **AskForQRWindow.xaml.cs**: contiene la lógica y funcionalidad de la ventana definida en su correspondiente archivo *"AskForQRWindow.xaml"*. En él se definen y manipulan los eventos, se accede a los controles de la interfaz gráfica y se realiza la lógica de negocio.
- **Autopay.csproj**: contiene la configuración del proyecto, incluyendo las referencias a las bibliotecas y ensamblados utilizados, las configuraciones de compilación y las dependencias del proyecto.
- **MainWindow.xaml**: contiene la estructura y definiciones de la interfaz gráfica necesarias para mostrar la pantalla de Standby. Es mostrada hasta que un usuario no identificado comienza el proceso de identificación.
- **MainWindow.xaml.cs**: contiene la lógica y funcionalidad de la ventana definida en su correspondiente archivo *"MainWindow.xaml"*. En él se definen y manipulan los eventos, se accede a los controles de la interfaz gráfica y se realiza la lógica de negocio.

- **NullEventArgs.cs**: contiene la definición de una clase utilizada para representar un argumento de evento cuando no se requiere utilizar ningún dato adicional en la resolución del evento.
- **Packages.config**: contiene una lista de los paquetes NuGet instalados en el proyecto, junto con sus versiones correspondientes.
- **PinWindow.xaml**: contiene la estructura y definiciones de la interfaz gráfica necesarias para mostrar la pantalla en la que el usuario o administrador introduce su pin para su identificación como usuario.
- **PinWindow.xaml.cs**: contiene la lógica y funcionalidad de la ventana definida en su correspondiente archivo "*PinWindow.xaml*". En él se definen y manipulan los eventos, se accede a los controles de la interfaz gráfica y se realiza la lógica de negocio.
- **Product.cs**: contiene la definición de la clase representativa a los productos. Define elementos como atributos o métodos para la obtención de los mismos.
- **ProductEventArgs.cs**: contiene la definición de una clase utilizada para representar los argumentos de evento cuando se requiere utilizar la información sobre un producto en la resolución del evento.
- **ProductsWindow.xaml**: contiene la estructura y definiciones de la interfaz gráfica necesarias para mostrar la pantalla de compra, información del estado del carro y pago.
- **ProductsWindow.xaml.cs**: contiene la lógica y funcionalidad de la ventana definida en su correspondiente archivo "*ProductsWindow.xaml*". En él se definen y manipulan los eventos, se accede a los controles de la interfaz gráfica y se realiza la lógica de negocio.
- **States.cs**: contiene la definición de una enumeración que define los posibles estados del carro: "*Correct*", "*Checking*" y "*Wrong*".
- **User.cs**: contiene la definición de la clase representativa a los usuarios. Define elementos como atributos o métodos para la obtención de los mismos.

4.1) bin

Contiene los archivos ejecutables (.exe) y bibliotecas de enlace dinámico (como los archivos .dll) necesarios para la correcta ejecución de la aplicación.

Figura 5. Estructura del directorio bin

Sus correspondientes directorios y/o archivos se encuentran en la dirección:

`“~/AutoPay/Código/Aplicación WPF/bin”`

Separa los dos distintos modos de ejecución en dos subdirectorios distintos:

- **Debug**: contiene el archivo ejecutable (Autopay.exe) y las bibliotecas dinámicas compartidas generadas en el modo de depuración. Estos archivos están optimizados para facilitar la depuración del código durante el desarrollo.
- **Release**: contiene el archivo ejecutable (Autopay.exe) y las bibliotecas dinámicas compartidas generadas en el modo de lanzamiento. Estos archivos están optimizados para un rendimiento y tamaño de archivo más eficiente en la distribución de la aplicación final, ya que no incluyen información de depuración adicional.

Su archivo ejecutable es el utilizado para lanzar la aplicación WPF.

4.2) Media

Contiene los archivos gráficos utilizados por el sistema.

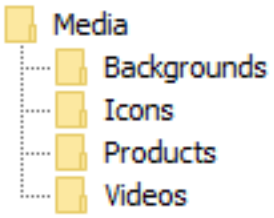

Figura 6. Estructura del directorio Media

Sus correspondientes directorios y/o archivos se encuentran en la dirección:

`“~/AutoPay/Código/Aplicación WPF/Media”`

Separa los distintos tipos de gráficos en distintos subdirectorios:

- **Backgrounds**: contiene las imágenes utilizadas como fondos de secciones o figuras utilizadas para la decoración de las distintas pantallas.
- **Icons**: contiene las imágenes utilizadas como iconos en las diferentes secciones, pantallas de carga o botones.
- **Products**: contiene las imágenes de los productos existentes en la base de datos, siendo el nombre de las mismas el código de barras del producto correspondiente.
- **Videos**: contiene los videos mostrados durante la pantalla de standby (MainWindow.xaml) y Petición de QRs (AskForQRWindow.xaml).

4.3) obj

Contiene los archivos temporales y de compilación utilizados internamente por el proceso de compilación (como archivos .obj y .pdb). No son necesarios para la ejecución de la aplicación.

Figura 7. Estructura del directorio obj

Sus correspondientes directorios y/o archivos se encuentran en la dirección:

`“~/AutoPay/Código/Aplicación WPF/obj”`

Separa los dos distintos modos de ejecución en dos subdirectorios distintos:

- **Debug**: contiene los archivos y metadatos relacionados con la compilación en modo de depuración. Estos archivos son útiles durante la etapa de desarrollo para permitir la depuración de la aplicación, como archivos de símbolos (.pdb)
- **Release**: contiene los archivos y metadatos relacionados con la compilación en modo de lanzamiento. Estos archivos son optimizados y están destinados a ser utilizados en el entorno de producción. Incluyen el archivo ejecutable de la aplicación (.exe) y otros archivos necesarios para la ejecución de la aplicación en un entorno de producción, como bibliotecas compartidas (.dll).

4.4) Properties

Contiene las configuraciones y metadatos relacionados con el proyecto, como la información de ensamblado, las configuraciones de compilación y las configuraciones de recursos específicos de la aplicación WPF.

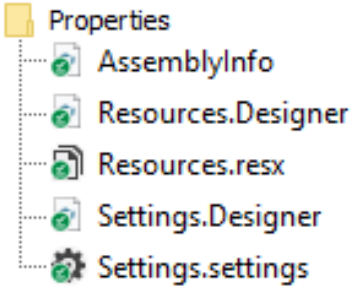

Figura 8. Estructura del directorio Properties

Sus correspondientes directorios y/o archivos se encuentran en la dirección:

`“~/AutoPay/Código/Aplicación WPF/Properties”`

Dentro del directorio se encuentran los siguientes archivos:

- **AssemblyInfo.cs**: contiene información y atributos específicos del ensamblado de la aplicación, como el nombre y descripción de la aplicación, la versión o el número de versión del ensamblado.
- **Resources.Designer.cs**: contiene clases y código que permiten acceder a los recursos definidos en el proyecto, como imágenes, archivos de texto, cadenas de texto o iconos.
- **Resources.resx**: junto con "*Resources.Designer.cs*", contiene clases y código que permiten acceder a los recursos definidos en el proyecto, como imágenes, archivos de texto, cadenas de texto o iconos. Permite la gestión centralizada de los recursos utilizados en la aplicación, lo que facilita su acceso y reutilización en diferentes partes del proyecto.
- **Settings.Designer.cs**: contiene clases y código que permiten acceder a las configuraciones definidas en el proyecto.
- **Settings.Settings**: contiene configuraciones y valores de configuración personalizados para la aplicación.

5) Arduino

En este apartado se describe la estructura y las funcionalidades de cada uno de los distintos directorios y/o archivos necesarios para el correcto funcionamiento de los distintos dispositivos mediante el uso del dispositivo embebido Arduino.

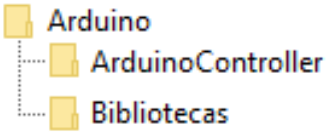

Figura 9. Estructura del directorio Arduino

Sus correspondientes directorios y/o archivos se encuentran en la dirección:

"~/AutoPay/Código/Arduino"

Dentro del directorio se encuentran los siguientes subdirectorios:

- **ArduinoController**: contiene el archivo "*ArduinoController.ino*", responsable del control de los dispositivos incorporados como el sensor de peso, el lector RFID o las luces LED.
- **Bibliotecas**: contiene las bibliotecas utilizadas para la facilitación de incorporación y comunicación con los diferentes dispositivos. Son utilizadas por el archivo "*ArduinoController.ino*".

5.1) Bibliotecas

Contiene las bibliotecas utilizadas para la facilitación de incorporación y comunicación con los diferentes dispositivos.

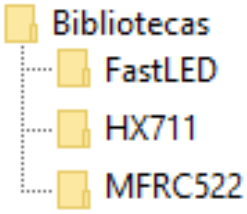

Figura 10. Estructura del directorio Bibliotecas

Sus correspondientes directorios y/o archivos se encuentran en la dirección:

"~/AutoPay/Código/Arduino/Bibliotecas"

Dentro del directorio se encuentran los siguientes subdirectorios:

- **FastLED**: contiene los archivos de la biblioteca FastLED, la cual proporciona funcionalidades y utilidades para controlar y manipular de manera eficiente las tiras de LED RGB.

- **HX711**: contiene los archivos de la biblioteca HX711, la cual proporciona una interfaz conveniente para conectar y leer datos de sensores de carga, amplificando y digitalizando las señales analógicas.
- **MFRC522**: contiene los archivos de la biblioteca MFRC522, la cual permite la lectura y escritura de tarjetas y etiquetas RFID compatibles con el estándar ISO/IEC 14443A.

6) Documentación del código

La documentación del código se ha realizado mediante el uso de cabeceras (comentarios) XML dentro del propio código fuente.

Con estas cabeceras, el propio programa Visual Studio es capaz de generar un fichero XML con la documentación del proyecto.

El fichero XML creado es legible para los humanos, pero su comprensión es muy difícil, por lo que se ha utilizado la herramienta "*KmCsharpHtmlDocGenerator*"[3] para generar una documentación del código mucho más legible en formato HTML enriquecido con CSS.

Sus correspondientes directorios y/o archivos se encuentran en la dirección:

"~/AutoPay/Documentación/Documentación del código"

Para visualizar dicha documentación tan solo hay que abrir el archivo "*index.html*" con cualquier intérprete (como Microsoft Edge).

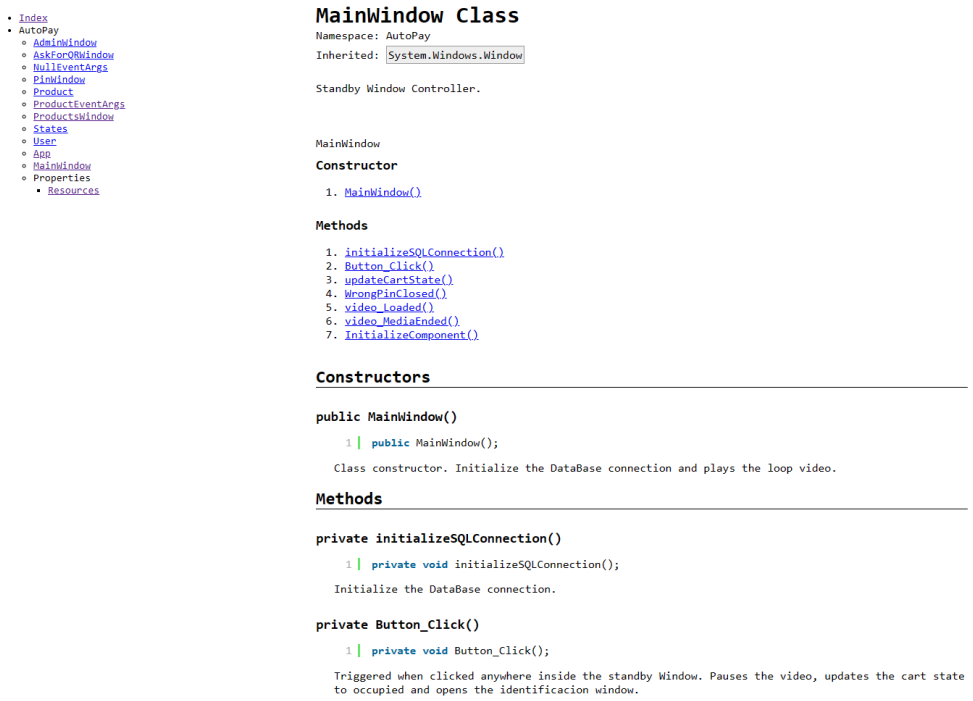

Figura 11. Visualización de la documentación del código para la clase MainWindow

7) Bibliografía

[1] IEvangelist. (2023, 18 marzo). *Working with .resx Files Programmatically - .NET*. Microsoft Learn. Recuperado 28 de junio de 2023, de https://learn.microsoft.com/en-us/dotnet/core/extensions/work-with-resx-files-programmatically

[2] Dotnet-Bot. (2023, 12 abril). *Documentation comments - C# language specification*. Microsoft Learn. Recuperado 28 de junio de 2023, de https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/documentation-comments

[3] AonaSuzutsuki. (s. f.). *GitHub - AonaSuzutsuki/KmCsharpHtmlDocGenerator: Convert C# Xml Doc to HTML*. GitHub. Recuperado 28 de junio de 2023, de https://github.com/AonaSuzutsuki/KmCsharpHtmlDocGenerator