Anexo IV: Diseño del sistema software

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

Tras la elicitación (especificación) de los requisitos software del sistema realizada en el documento "*Anexo II: Especificación de requisitos software*", y su posterior análisis realizado en el documento "*Anexo III: Análisis de requisitos software*", se debe realizar el diseño de dichos requisitos.

En este anexo se desarrolla el modelo de diseño, el cual describe la realización física de los casos de uso, teniendo en cuenta tanto los requisitos funcionales como los no funcionales y las posibles limitaciones que pueda tener la implementación del sistema.

Durante el desarrollo de este modelo se ha elegido el patrón arquitectónico a implantar (ya implantado por la tecnología WPF) y se han desarrollado los subsistemas y las clases de diseño, dando lugar a la vista arquitectónica del sistema.

Para el diseño de la realización de los casos de uso se han modificado los diagramas de secuencia existentes para adaptarse a la realización física del sistema.

En cuanto a la realización física del almacenamiento de información, se ha realizado el diseño de la base de datos, haciendo posible la visualización de la misma de forma global mediante el diseño de un modelo de la base de datos.

Por último, se ha realizado el modelo de despliegue para desarrollar una visión global de la distribución final del sistema.

2) Modelo de diseño

El modelo de diseño es un proceso que expone la creación de una estructura y planificación detallada para el desarrollo de un sistema software. Consiste en identificar los requisitos del proyecto, diseñar la arquitectura del software, definir los módulos y componentes y establecer las interacciones entre ellos.

El objetivo es crear una base sólida para la implementación del software, asegurando que cumpla con los objetivos y requisitos previamente establecidos.

Para facilitar el desarrollo y la comprensión del sistema se ha utilizado un patrón arquitectónico de diseño.

2.1) Patrón arquitectónico

El patrón MVVM[1] es el patrón por defecto recomendado por Microsoft para el desarrollo de aplicaciones WPF en la plataforma .NET. Esto se debe a que WPF está diseñado orientado al uso de este patrón, lo que facilita su implementación y aprovecha las funcionalidades proporcionadas por la tecnología .NET de manera más eficiente; lo que resulta en un desarrollo más rápido y consistente.

El patrón Modelo-Vista-Modelo de vista (MVVM) ayuda a separar limpiamente la lógica de presentación y de negocios de una aplicación de su interfaz gráfica., lo que ayuda a evitar numerosos problemas de desarrollo y facilita la prueba, el mantenimiento y la evolución del sistema.

Con el patrón MVVM, la interfaz de usuario de la aplicación y la lógica de presentación y negocios subyacente se separan en tres clases distintas:

- **La vista**: encapsula la lógica de la interfaz de usuario y la interfaz de usuario.
- **El modelo de la vista:** encapsula la lógica y el estado de la presentación
- **El modelo**: encapsula la lógica de negocios y los datos de la aplicación.

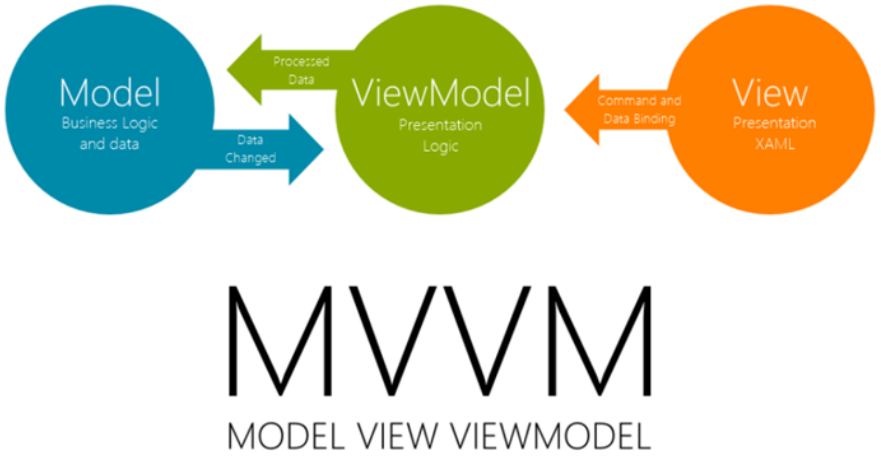

Figura 1. Arquitectura MVVM en proyectos .NET

En la siguiente figura[2] se muestra el esquema general de la arquitectura MVVM aplicado a un sencillo ejemplo: la aplicación expuesta muestra un listado de usuarios en el que se puede seleccionar un usuario.

Como podemos observar, el modelo *Usuario* representa a un usuario "real" con sus atributos nombre y apellidos.

El ViewModel contiene los datos que la vista necesita (usuarios y el usuario actualmente seleccionado). Estos datos preparados por el ViewModel están conectados (bindeados) con las propiedades correspondientes de los elementos de la vista ListView y TextBox.

Cada vez que se selecciona un usuario del listado, el campo de texto enlazado a la propiedad *Usuario Seleccionado* muestra el nombre del usuario correspondiente. Esto es posible gracias a la conexión entre el ViewModel y la Vista a través de la propiedad DataContext, que asigna un conjunto de datos y comportamiento a una vista.

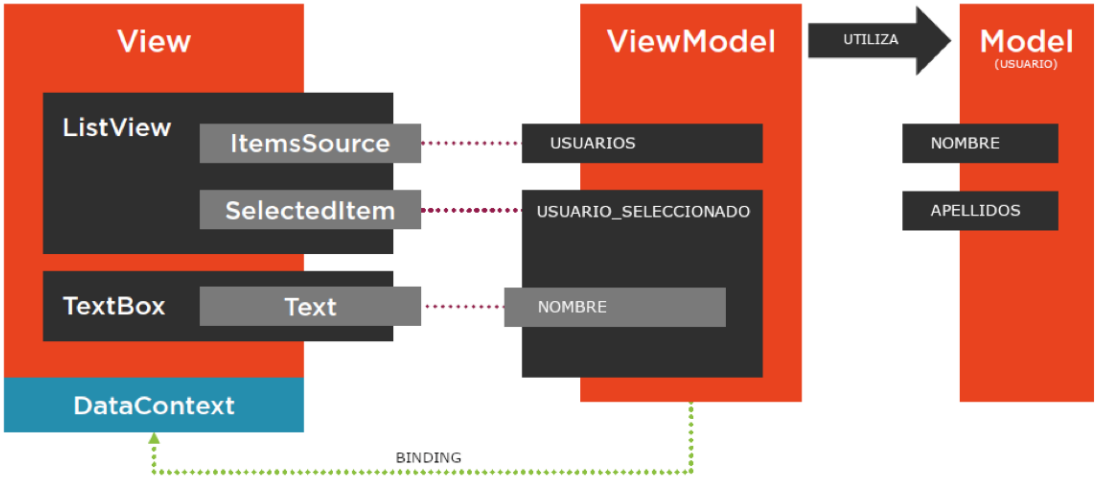

Figura 2. Esquema general de la arquitectura MVVM aplicado a un sencillo ejemplo

Mediante el uso de este patrón en proyectos .NET, podemos conseguir que la conexión de datos sea bidireccional y reactiva, es decir, que el muestreo de datos en las vistas sea en tiempo real y que se reflejen los cambios automáticamente.

2.2) Subsistemas de diseño

Para facilitar la comprensión del diseño se ha dividido el sistema en subsistemas más específicos denominados subsistemas de diseño.

La principal división se ha realizado entre la aplicación ejecutada desde la Tablet incorporada al carro, el lector de códigos de barras y el Arduino, el cual controla el resto de los dispositivos vinculados al sistema.

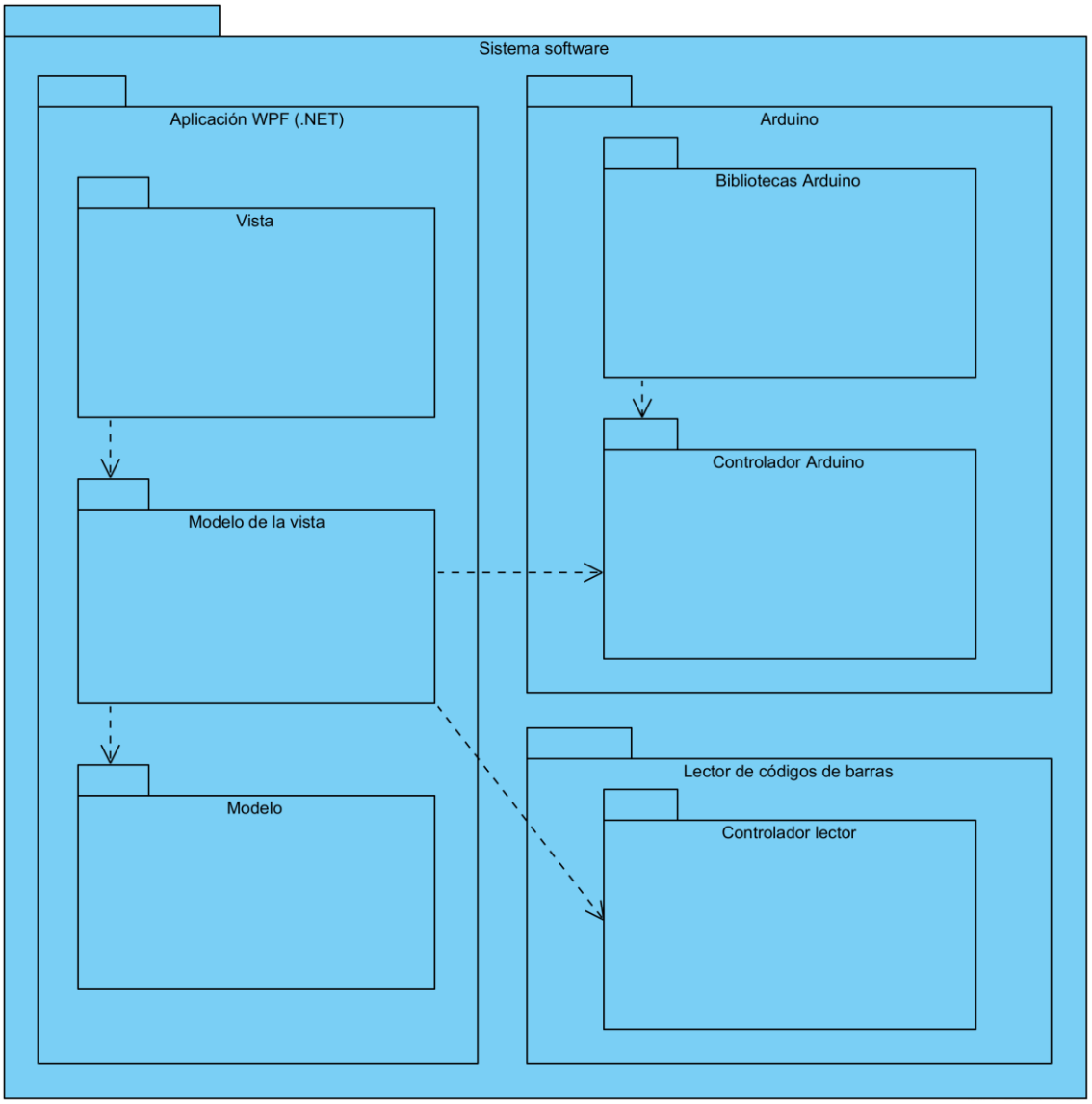

Figura 3. Diagrama de subsistemas de diseño

2.3) Clases de diseño

En este apartado se describen las clases de diseño resultantes de la evolución de las clases de análisis al adaptarse a la realización física de los casos de uso.

Las diferentes clases de diseño se exponen dentro del contexto de su subsistema para facilitar su estructuración y, por lo tanto, su comprensión.

2.3.1) Aplicación WPF (.NET)

- **2.3.1.1) Vista**

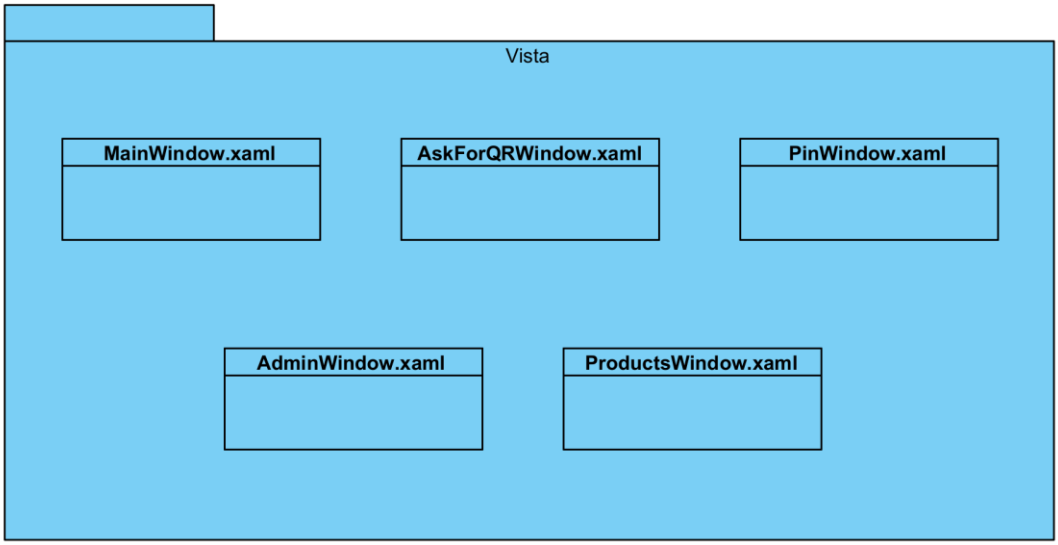

Figura 4. Clases de diseño: Vista

- **2.3.1.2) Modelo de la vista**

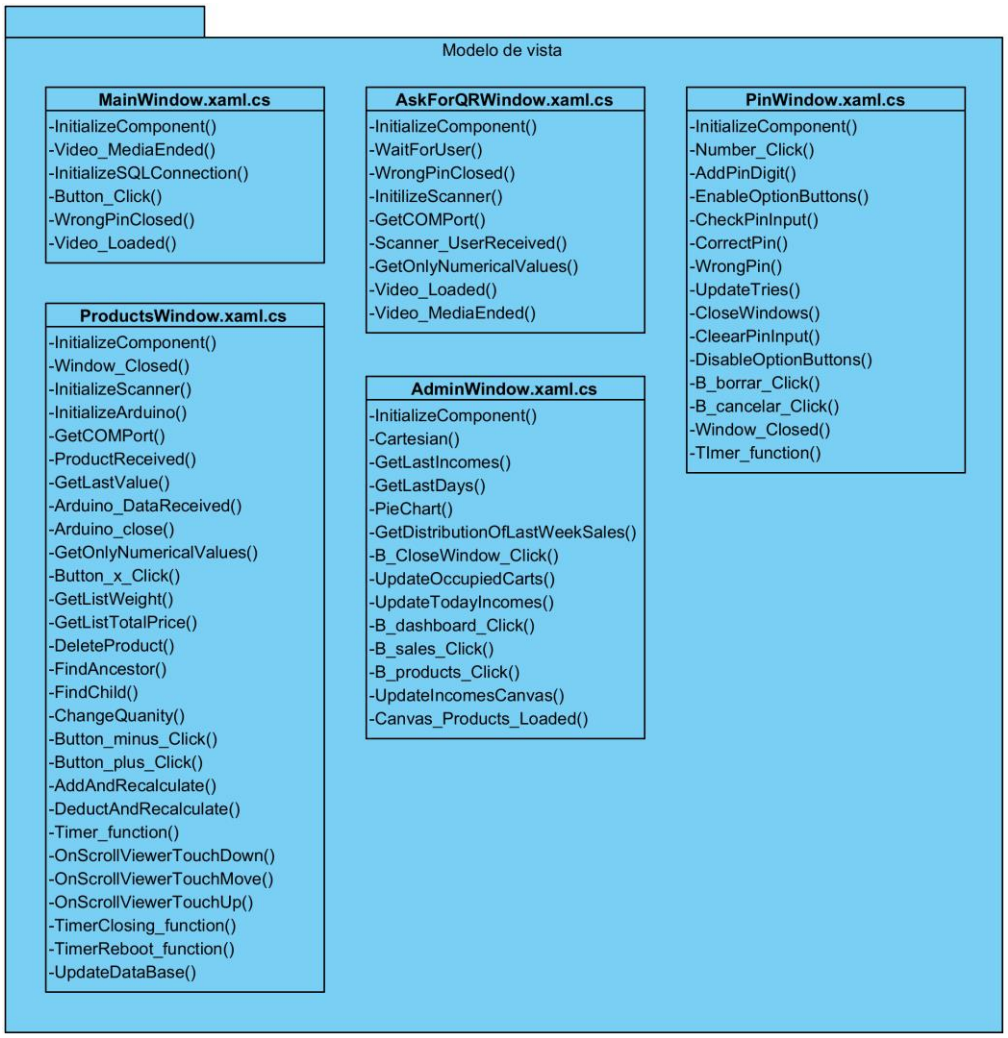

Figura 5. Clases de diseño: Modelo de la vista

- **2.3.1.3) Modelo**

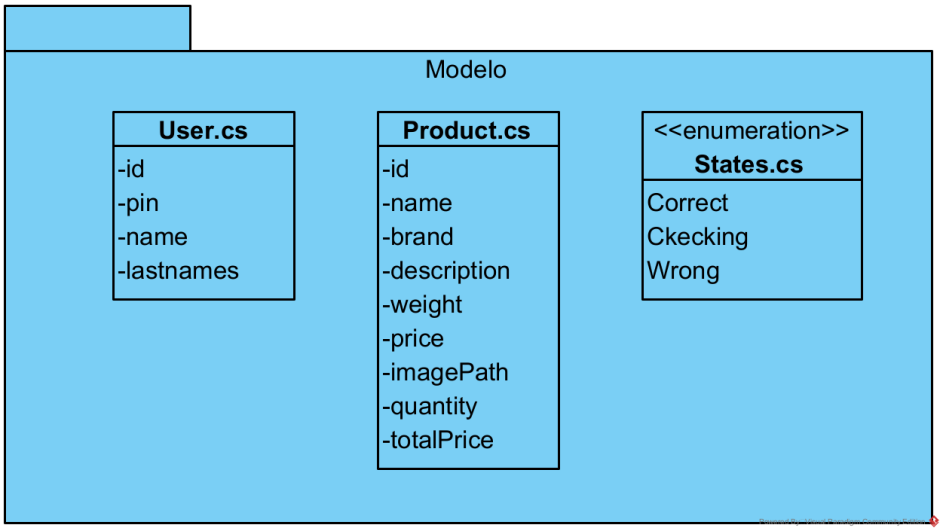

Figura 6. *Clases de diseño: Modelo*

2.3.2) Arduino

- **2.3.2.1) Bibliotecas**

  El controlador de Arduino utiliza bibliotecas de dominio público para controlar de forma correcta y eficiente los diferentes dispositivos. Para no profundizar en exceso en sus funciones, solo indicaré las bibliotecas utilizadas por el controlador.

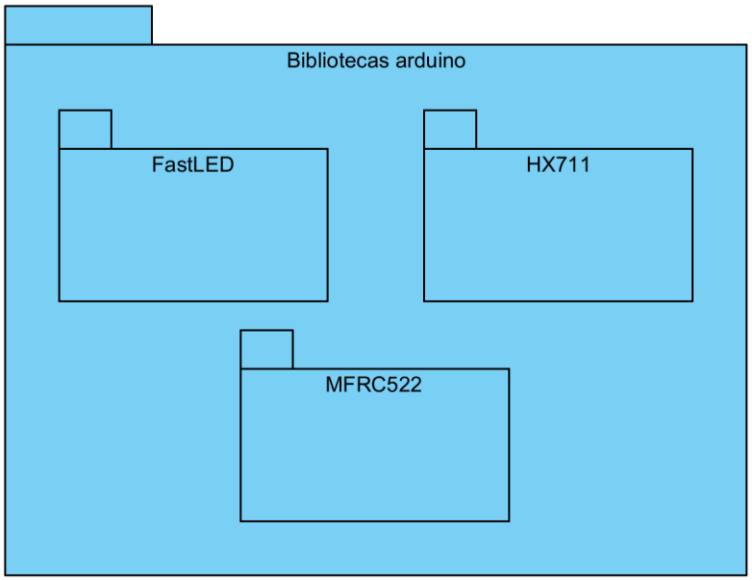

Figura 7. Clases de diseño: Bibliotecas utilizadas por el Arduino

- **2.3.2.2) Controlador**

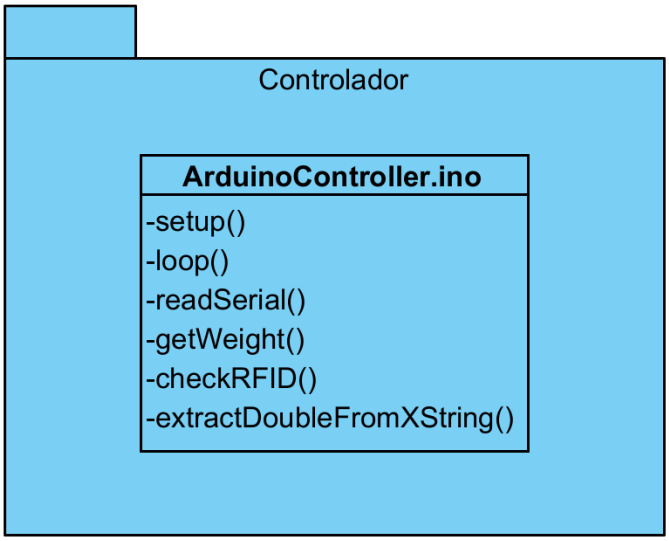

Figura 8. Clases de diseño: Controlador del Arduino

2.3.3) Lector de códigos de barras

La única interacción posible con el lector de códigos de barras se realiza desde el controlador de la aplicación WPF (.NET), ya que no se tiene acceso al código de su controlador. Por esto, se muestra el controlador del lector de códigos de barras vacío, representando su funcionamiento como caja negra.

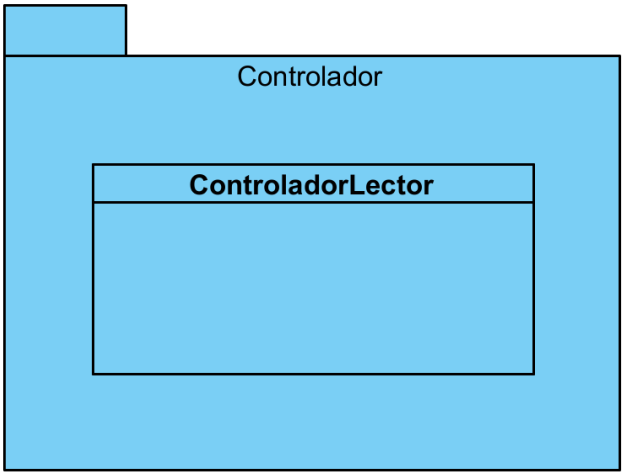

Figura 9. Clases de diseño: Controlador del Arduino

2.4) Vista arquitectónica

En ese apartado se expone como recapitulación un diagrama representativo de la vista arquitectónica, la cual permite la visión global de los subsistemas completos que conforman el sistema software.

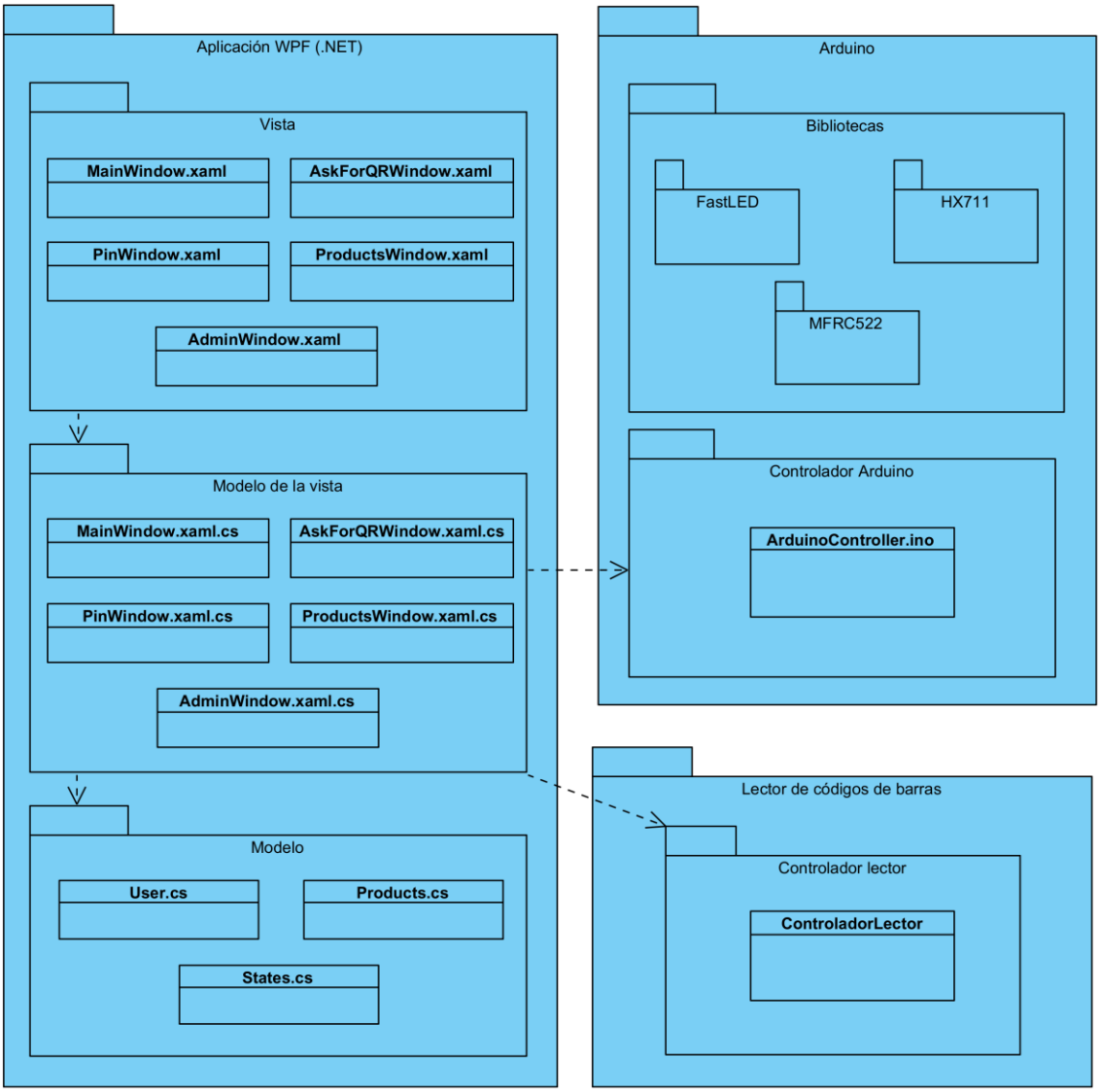

Figura 10. Vista arquitectónica

2.5) Realización de casos de uso

En este apartado se expone la realización física de los casos de uso mediante el desarrollo de diagramas de secuencia a partir de los realizados durante la fase de análisis de requisitos desarrollada en el documento "*Anexo II: Especificación de requisitos software*".

Se han mantenido los estereotipos (boundary, control y entity) de los diferentes elementos para su mejor comprensión visual.

Los casos de uso realizados principalmente por el sistema han sido desarrollados como máquinas de estado en vez de como diagramas de secuencia para su mejor comprensión.

2.5.1) Gestión de acceso

• UC-01: Identificar usuario

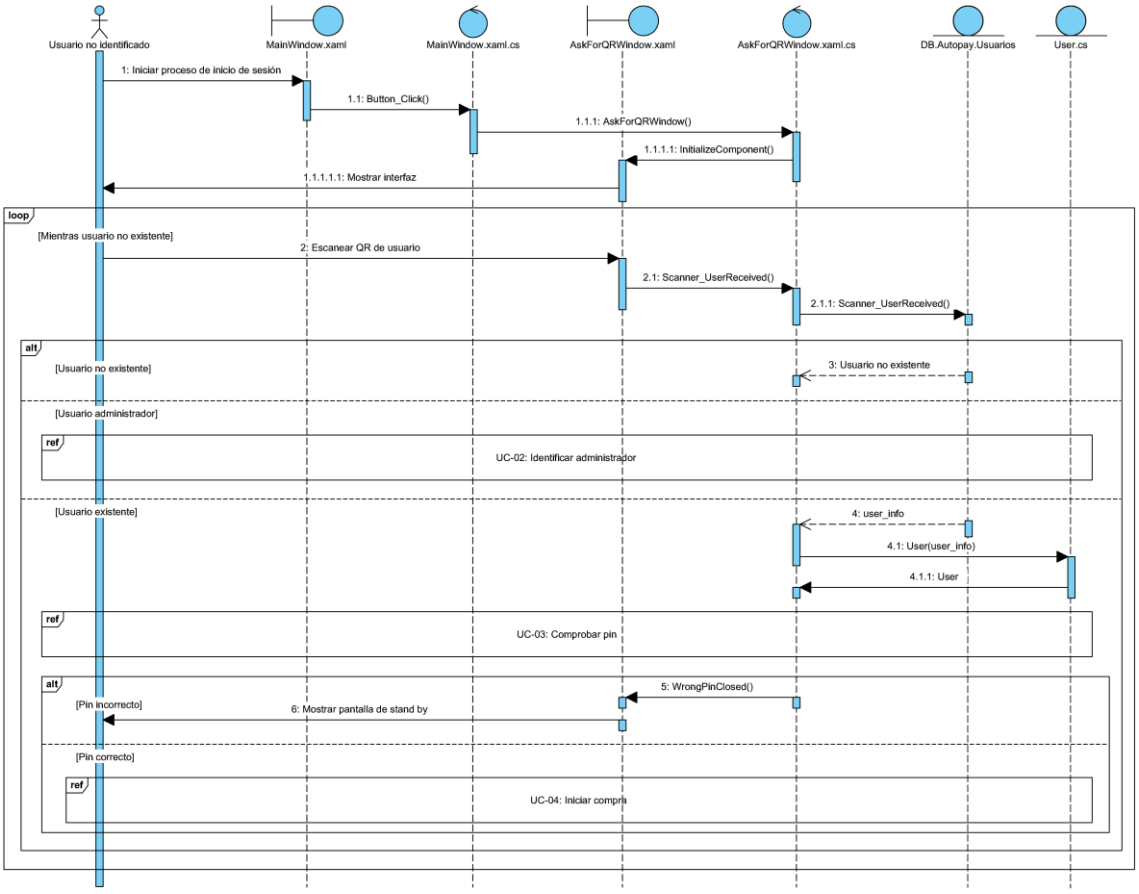

Figura 11. Diagrama de secuencia del UC-01: Identificar usuario

- **UC-02: Identificar administrador**

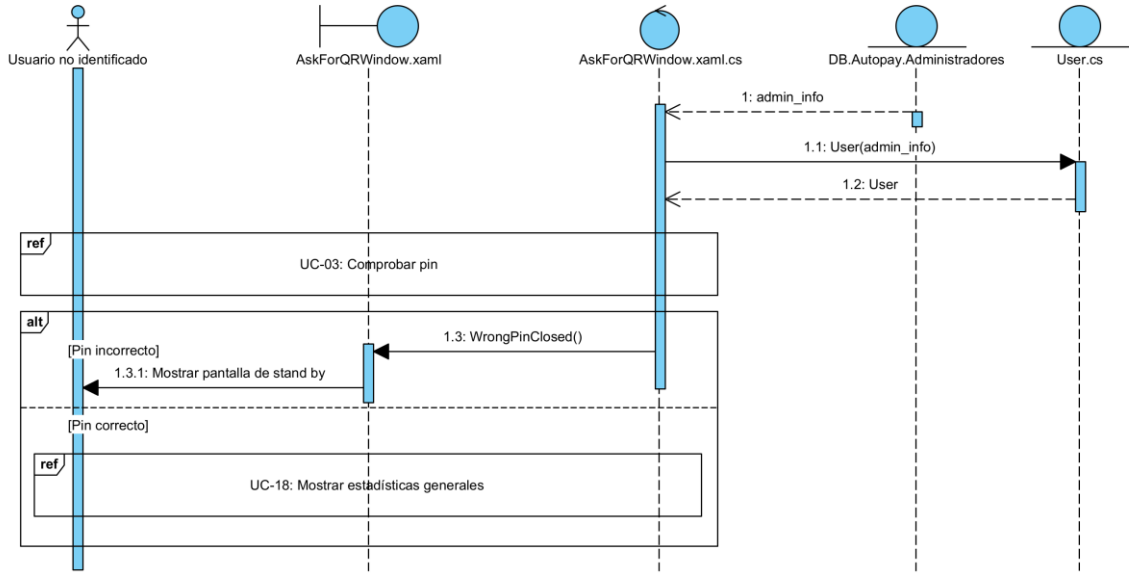

Figura 12. Diagrama de secuencia del UC-02: Identificar administrador

- **UC-03: Comprobar pin**

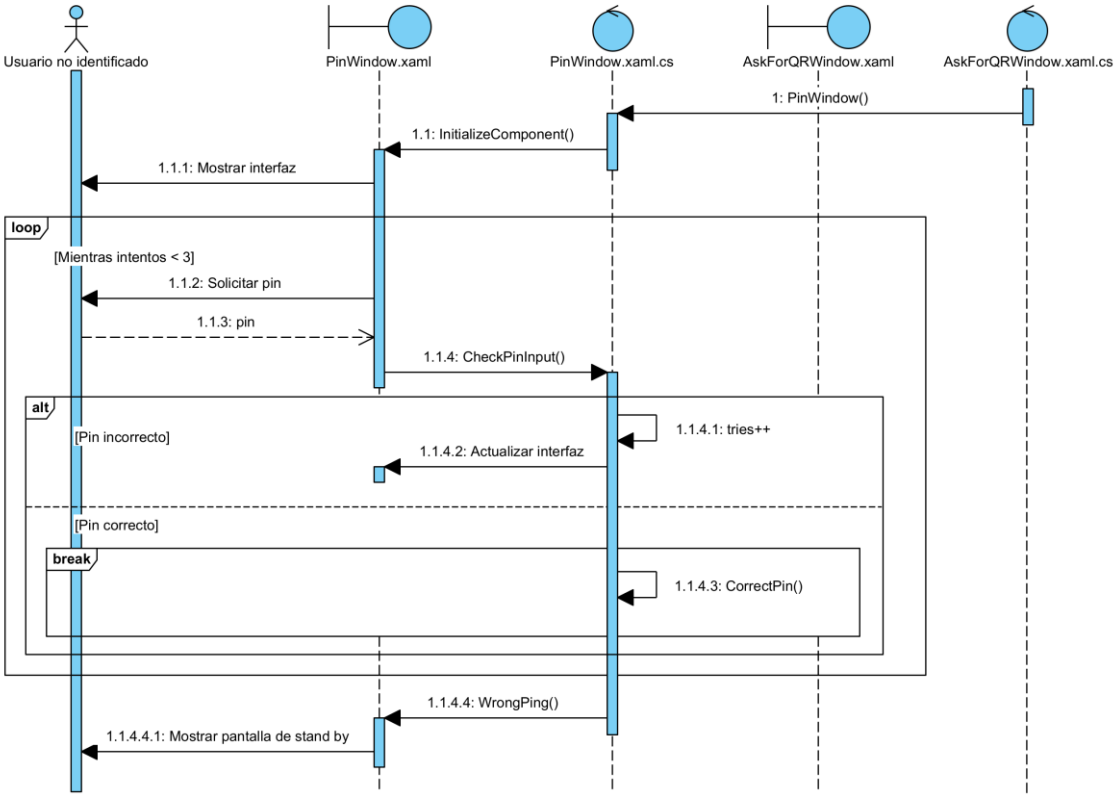

Figura 13. Diagrama de secuencia del UC-03: Comprobar pin

2.5.2) Gestión de productos

· UC-04: Iniciar compra

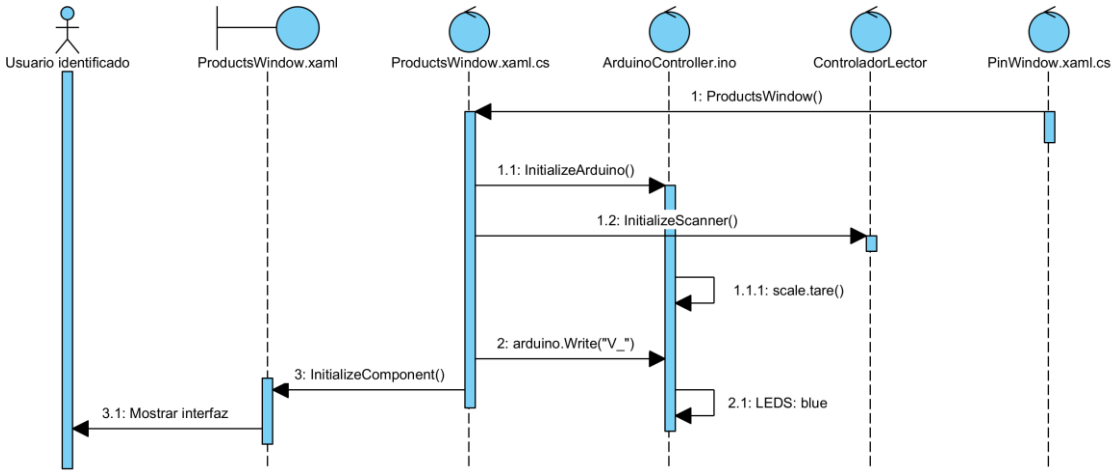

Figura 14. Diagrama de secuencia del UC-04: Iniciar compra

· UC-05: Detectar producto

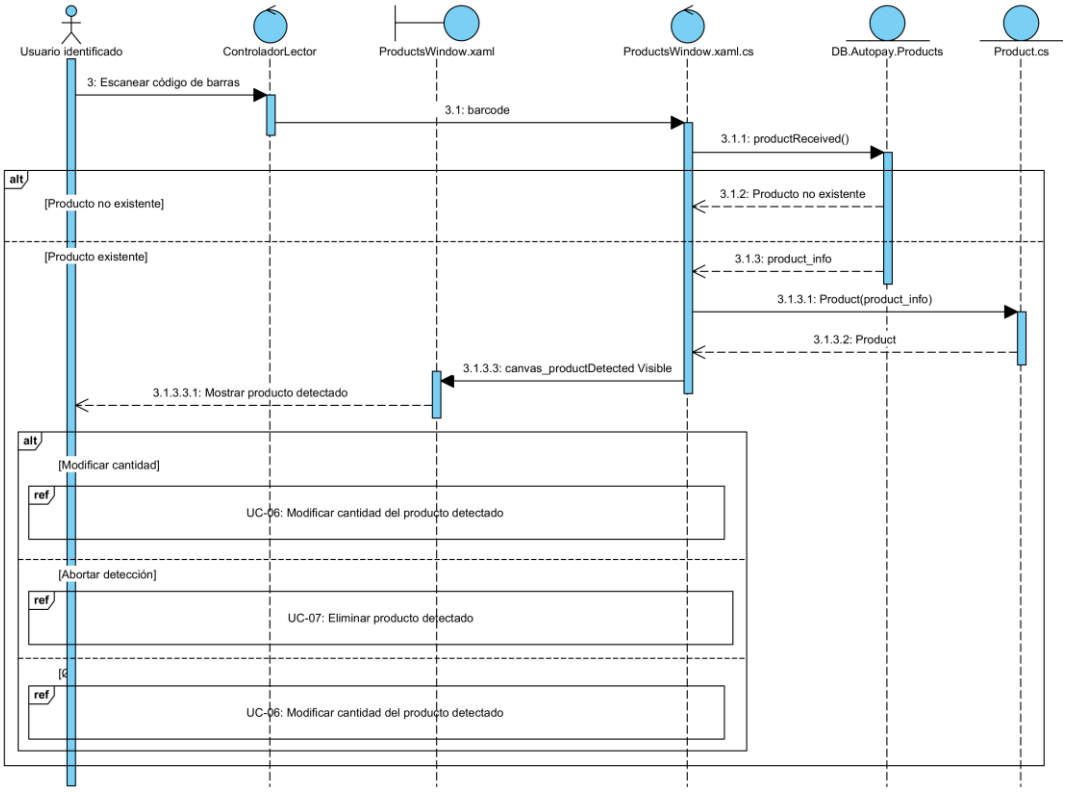

Figura 15. Diagrama de secuencia del UC-05: Detectar producto

- **UC-06: Modificar cantidad del producto detectado**

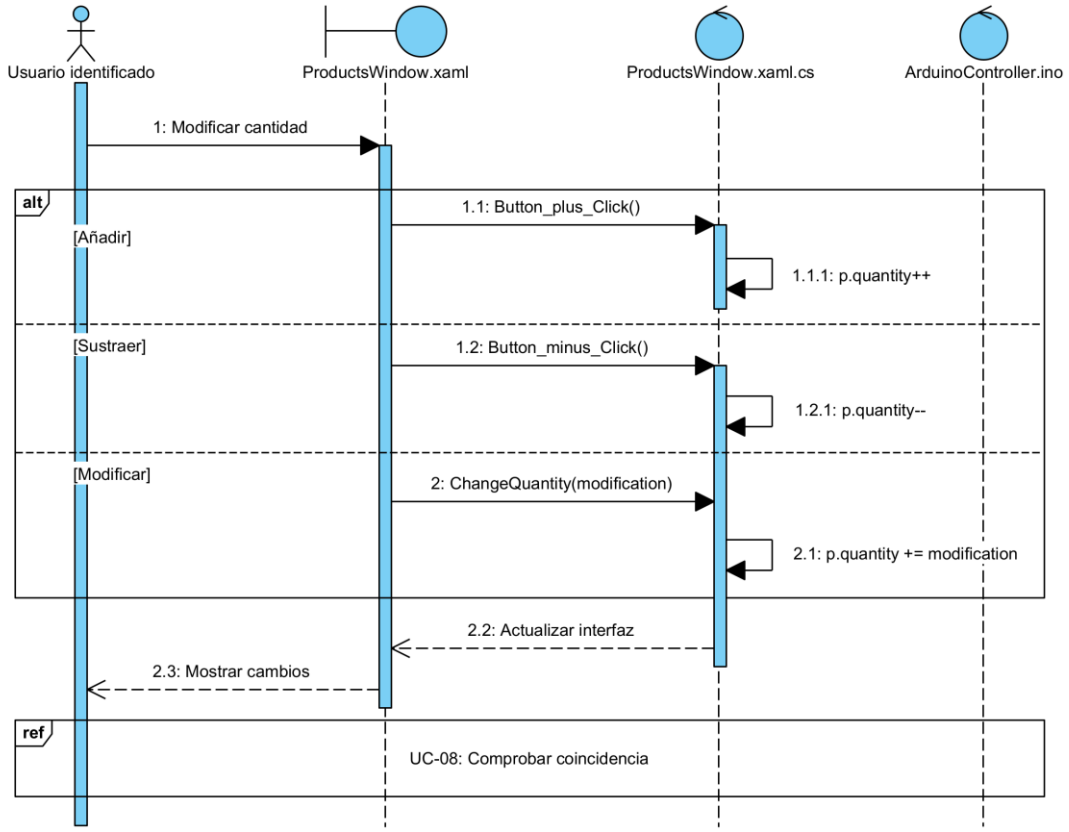

Figura 16. Diagrama de secuencia del UC-06: Modificar cantidad del producto detectado

- **UC-07: Eliminar producto detectado**

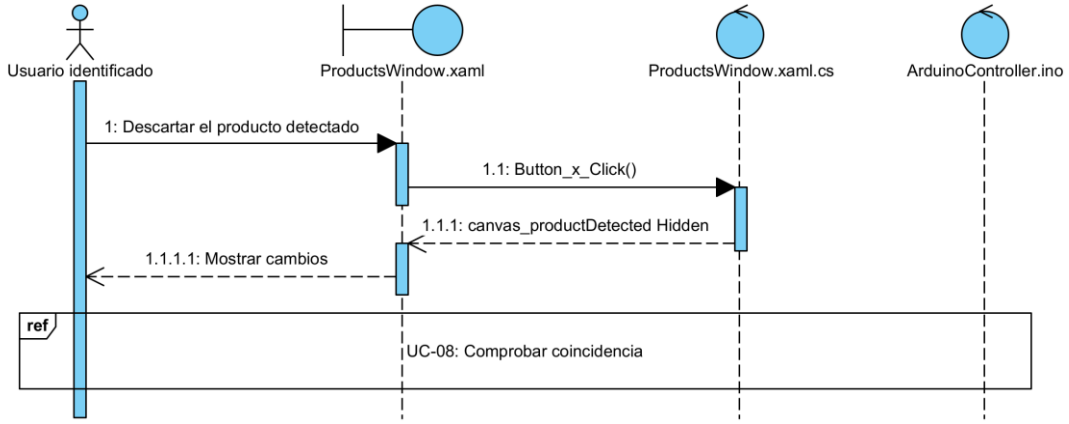

Figura 17. Diagrama de secuencia del UC-07: Eliminar producto detectado

- **UC-08: Comprobar coincidencia**

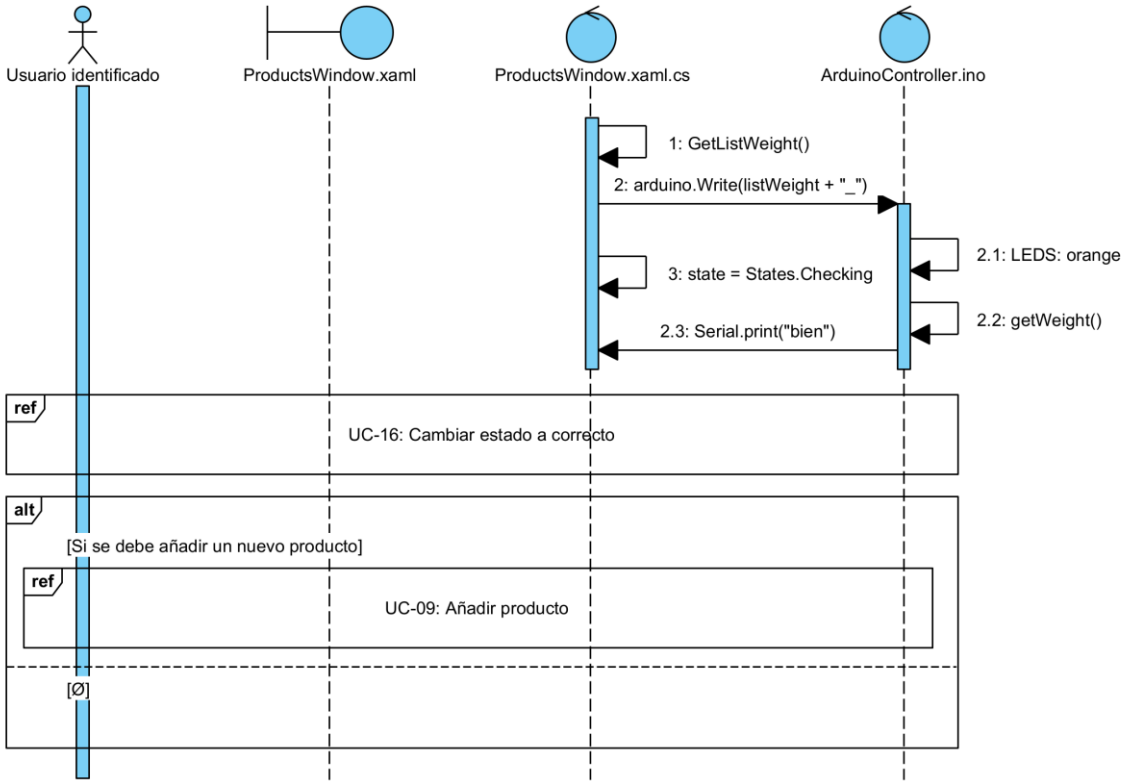

Figura 18. Diagrama de secuencia del UC-08: Comprobar coincidencia

- **UC-09: Añadir producto**

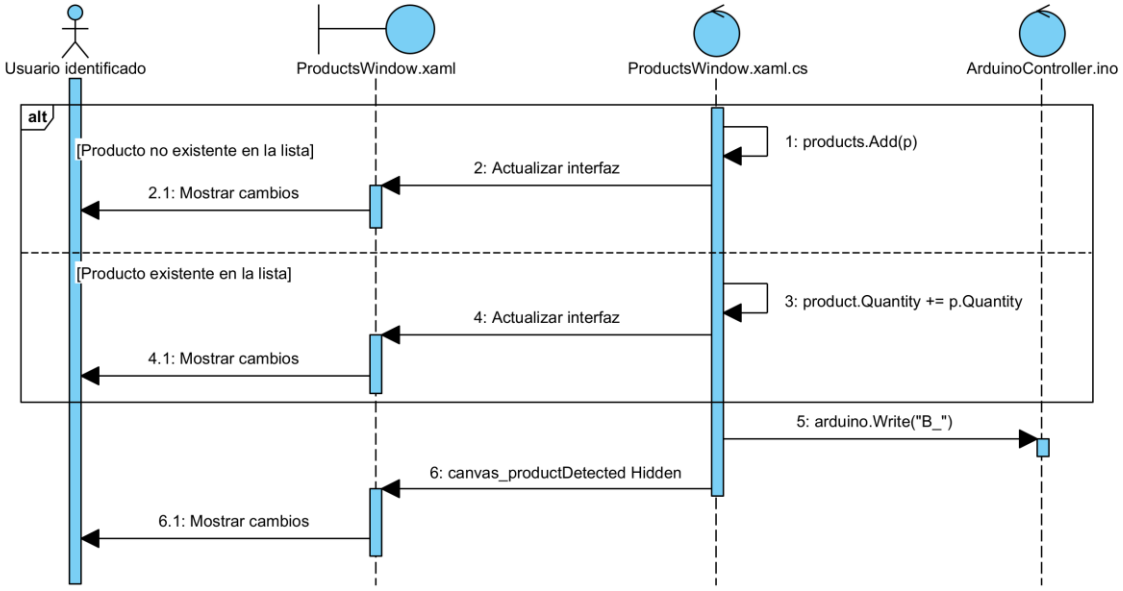

Figura 19. Diagrama de secuencia del UC-09: Añadir producto

- **UC-10: Modificar cantidad de un producto añadido**

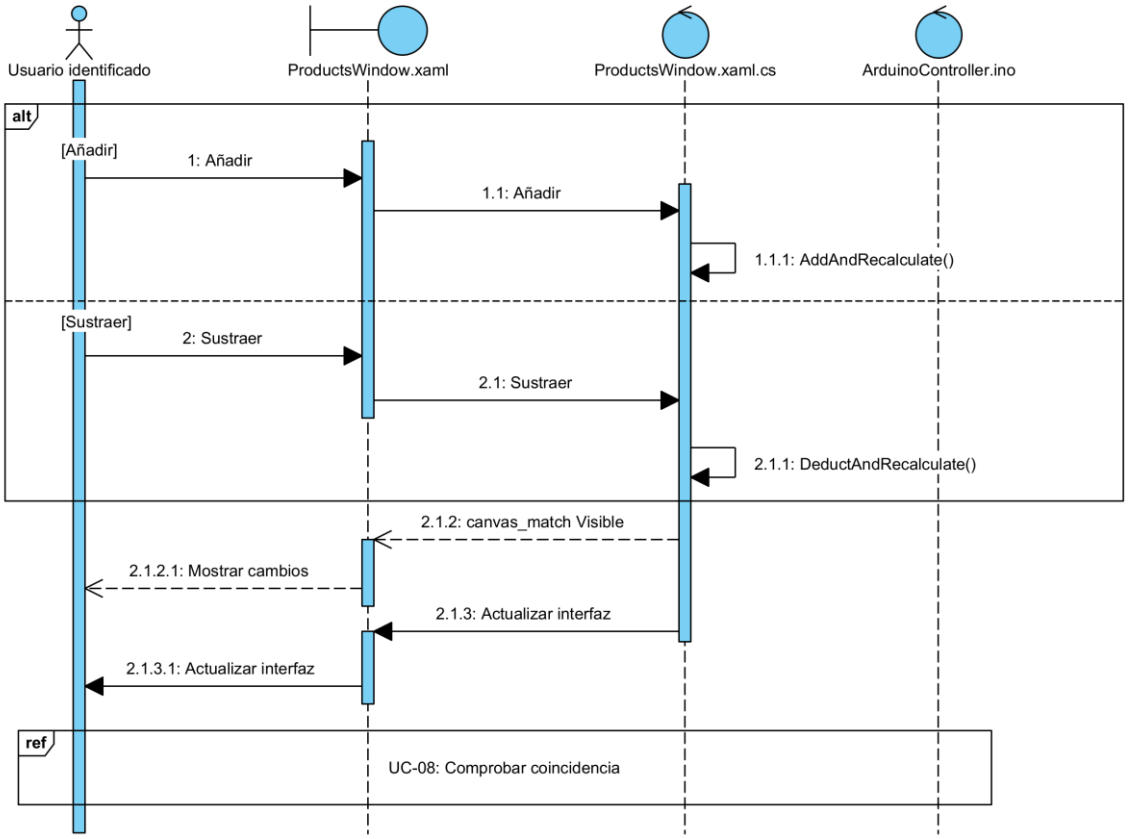

Figura 20. Diagrama de secuencia del UC-10: Modificar cantidad de un producto añadido

- **UC-11: Eliminar producto añadido**

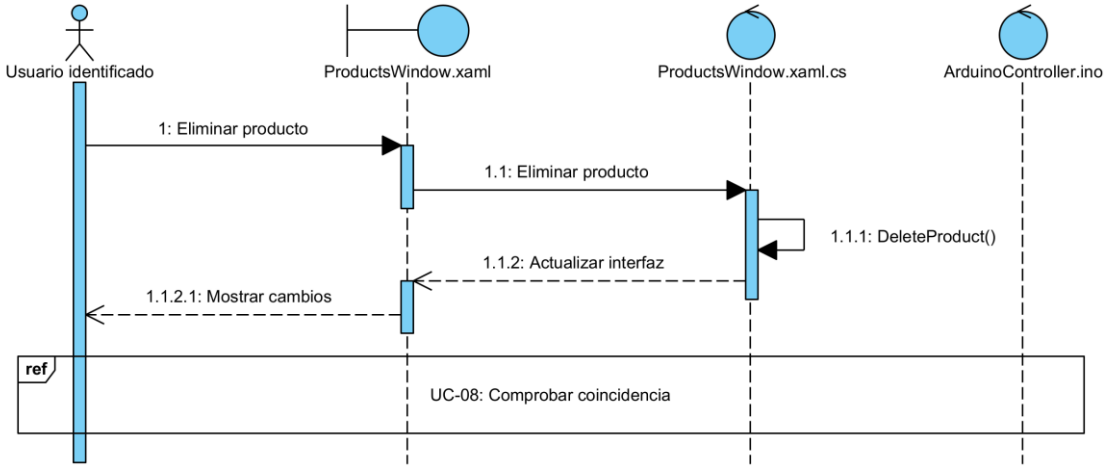

Figura 21. Diagrama de secuencia del UC-11: Eliminar producto añadido

2.5.3) Gestión del estado del carro

· UC-12: Reaccionar ante un cambio no previsto

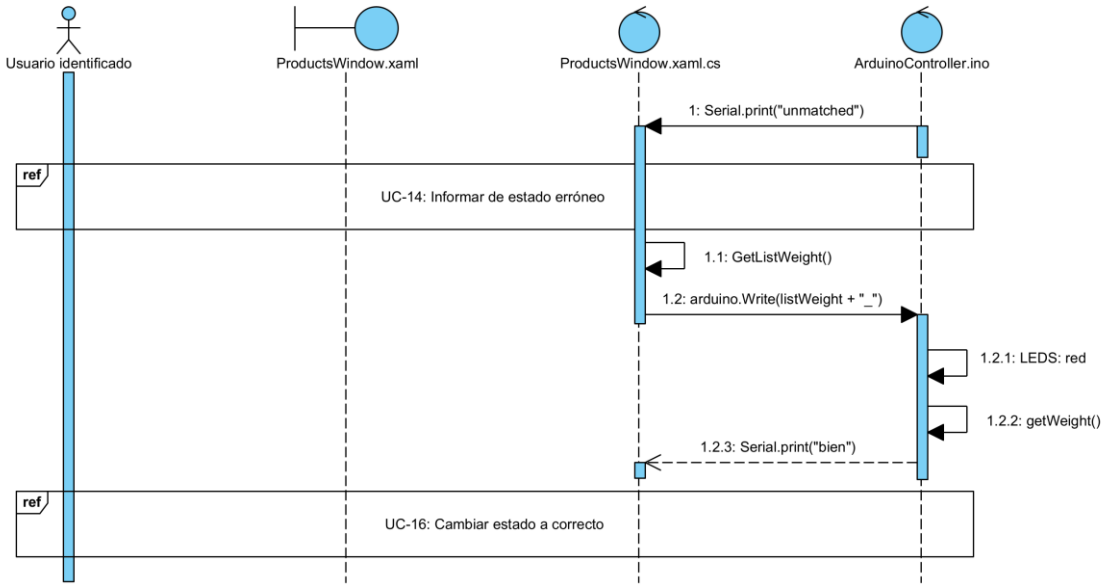

Figura 22. Diagrama de secuencia del UC-12: Reaccionar ante un cambio no previsto

· UC-13: Comprobar estado del carro

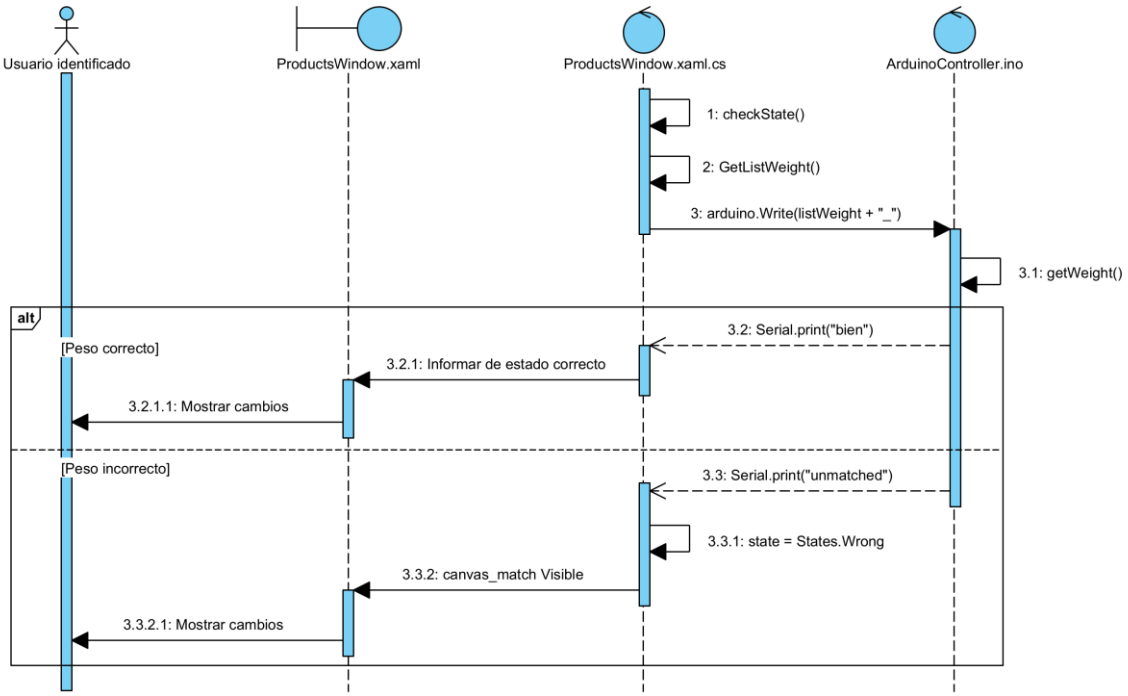

Figura 23. Diagrama de secuencia del UC-13: Comprobar estado del carro

· UC-14: Informar de estado erróneo

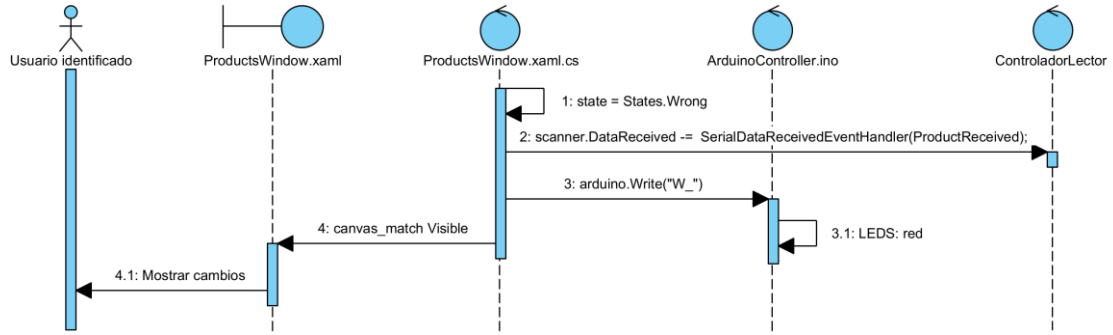

Figura 24. Diagrama de secuencia del UC-14: Informar de estado erróneo

· UC-15: Actualizar ocupación del carro

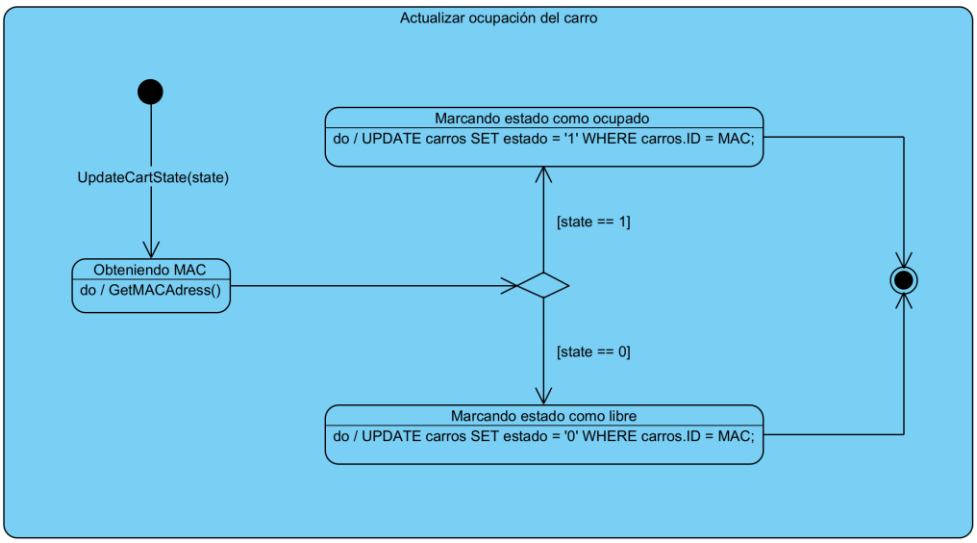

Figura 25. Máquina de estados del UC-15: Actualizar ocupación del carro

· UC-16: Cambiar estado a correcto

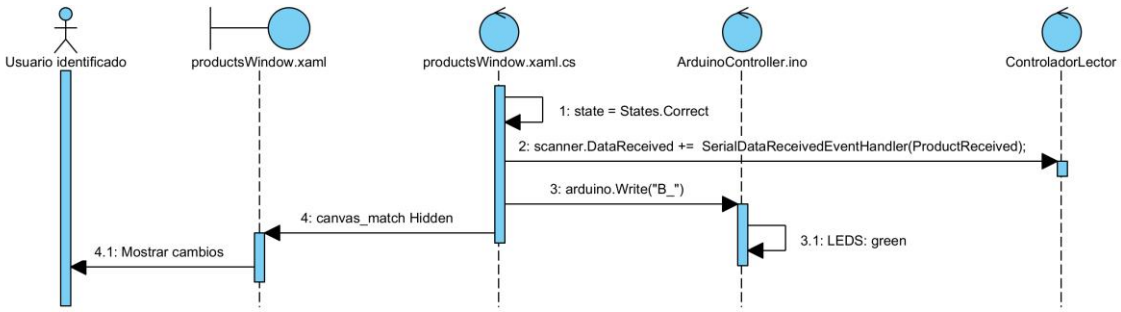

Figura 26. Diagrama de secuencia del UC-16: Cambiar estado a correcto

· UC-17: Reiniciar el dispositivo

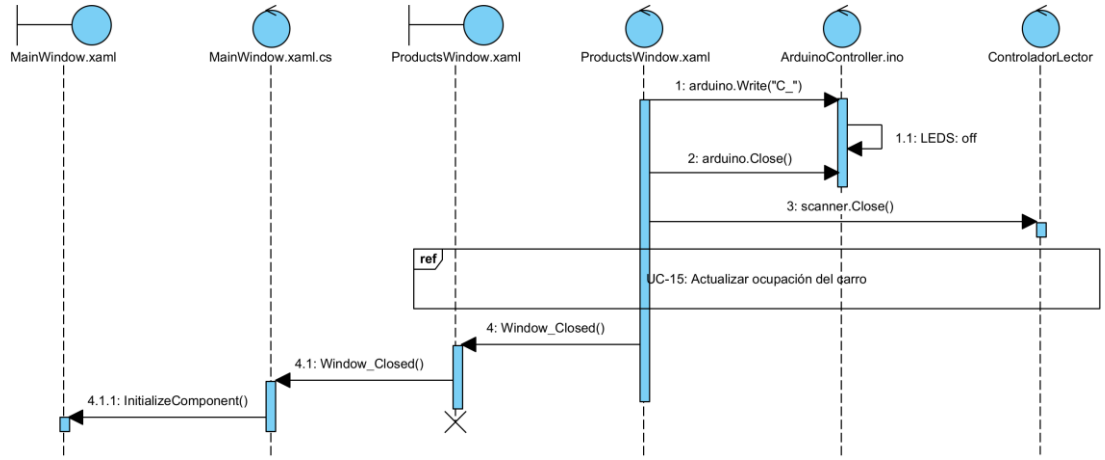

Figura 27. Diagrama de secuencia del UC-17: Reiniciar el dispositivo

2.5.4) Gestión de estadísticas

· UC-18: Mostrar estadísticas generales

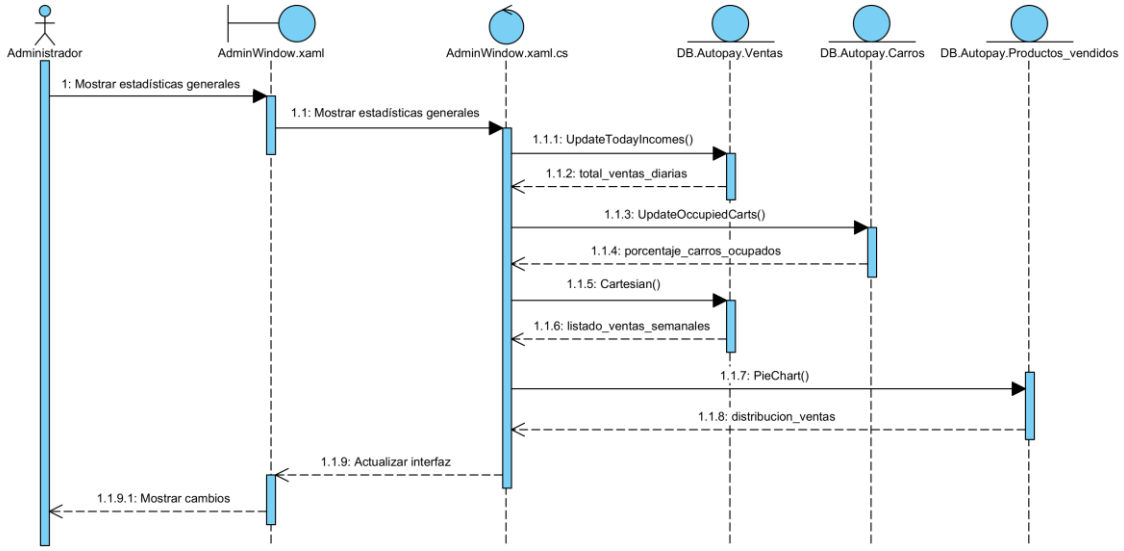

Figura 28. Diagrama de secuencia del UC-18: Mostrar estadísticas generales

· UC-19: Mostrar estadísticas de ventas

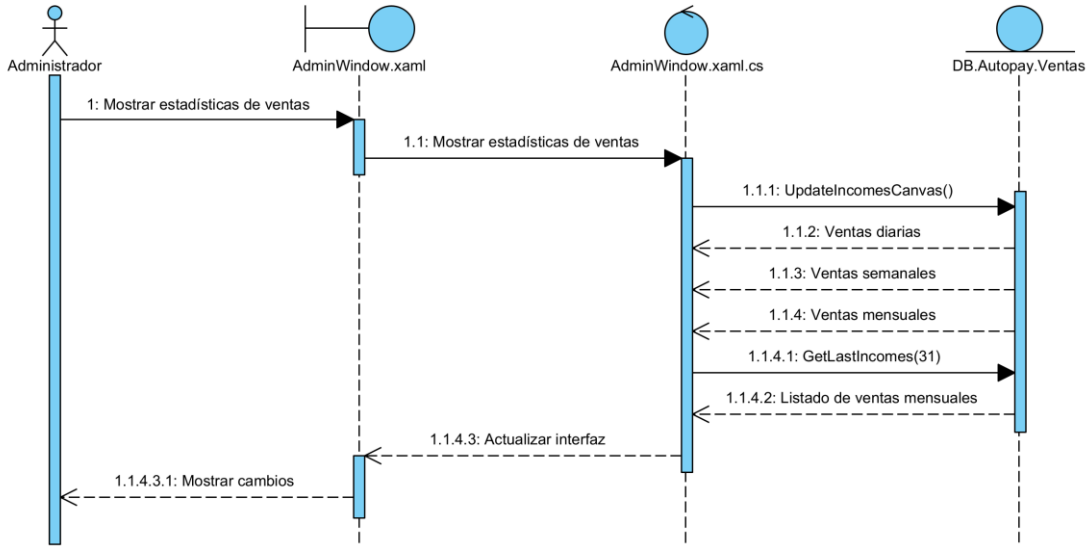

Figura 29. Diagrama de secuencia del UC-19: Mostrar estadísticas de ventas

· UC-20: Mostrar estadísticas de productos

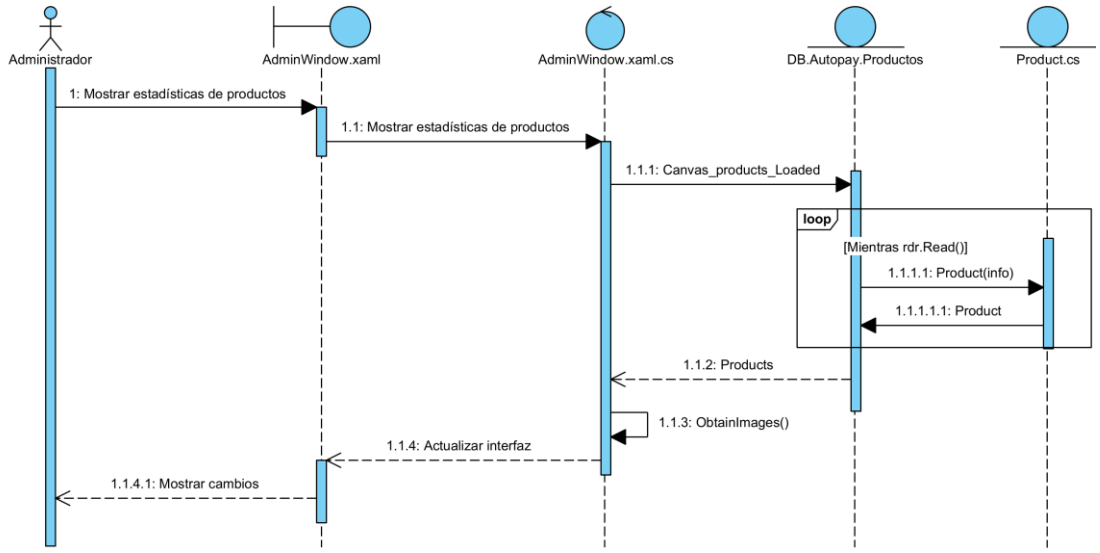

Figura 30. Diagrama de secuencia del UC-20: Mostrar estadísticas de productos

2.5.5) Gestión de pago

· UC-21: Comprobar posibilidad de pago

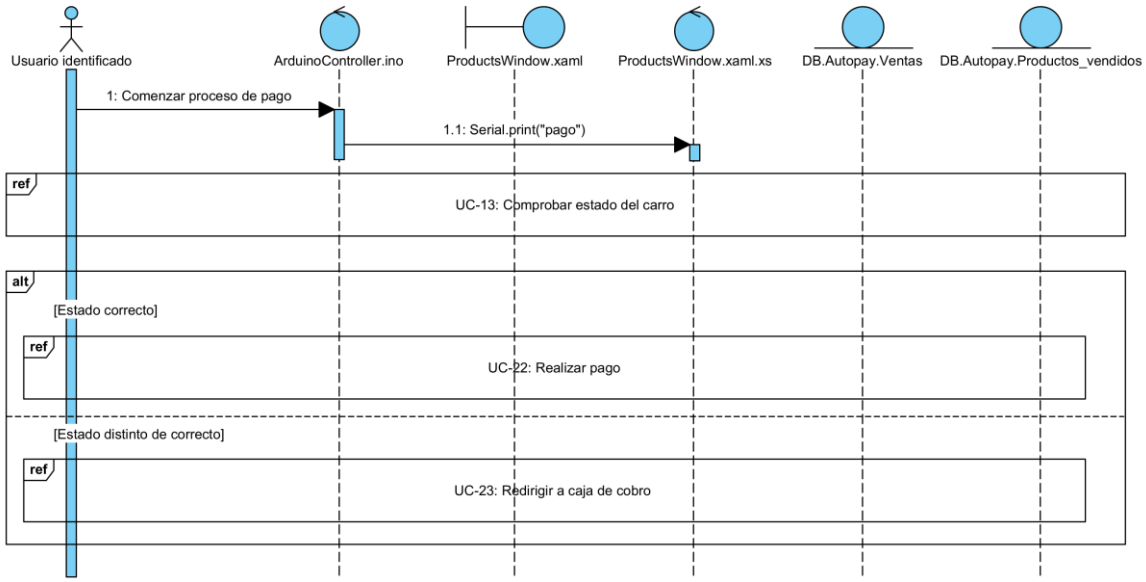

Figura 31. Diagrama de secuencia del UC-21: Comprobar posibilidad de pago

· UC-22: Realizar pago

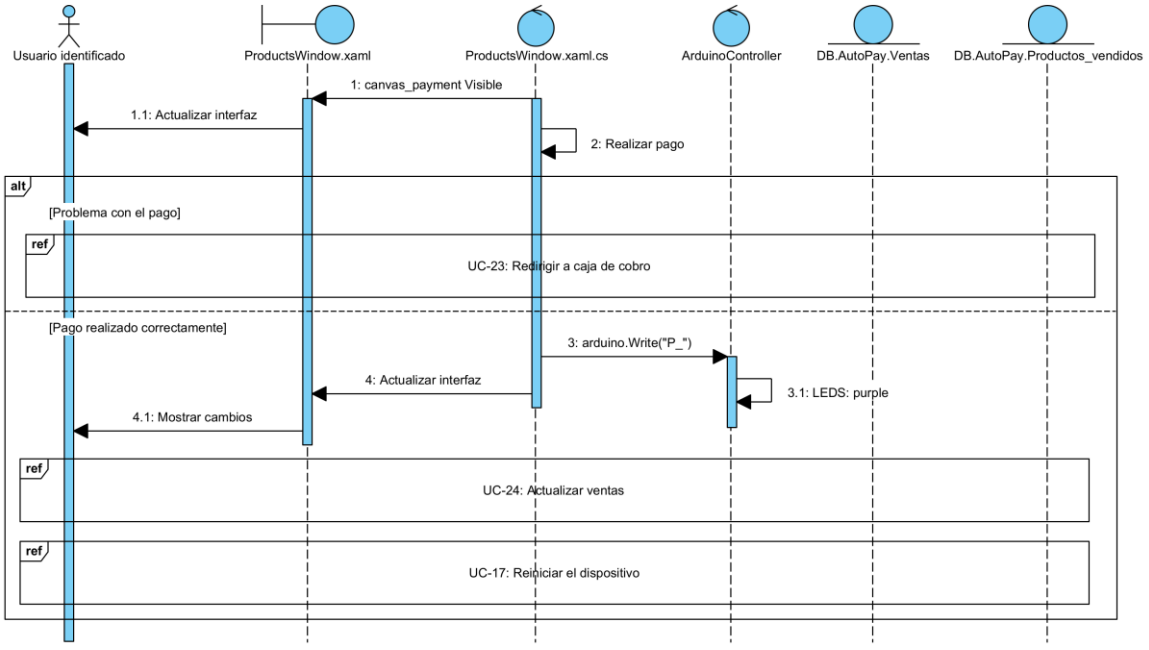

Figura 32. Diagrama de secuencia del UC-22: Realizar pago

· UC-23: Redirigir a caja de cobro

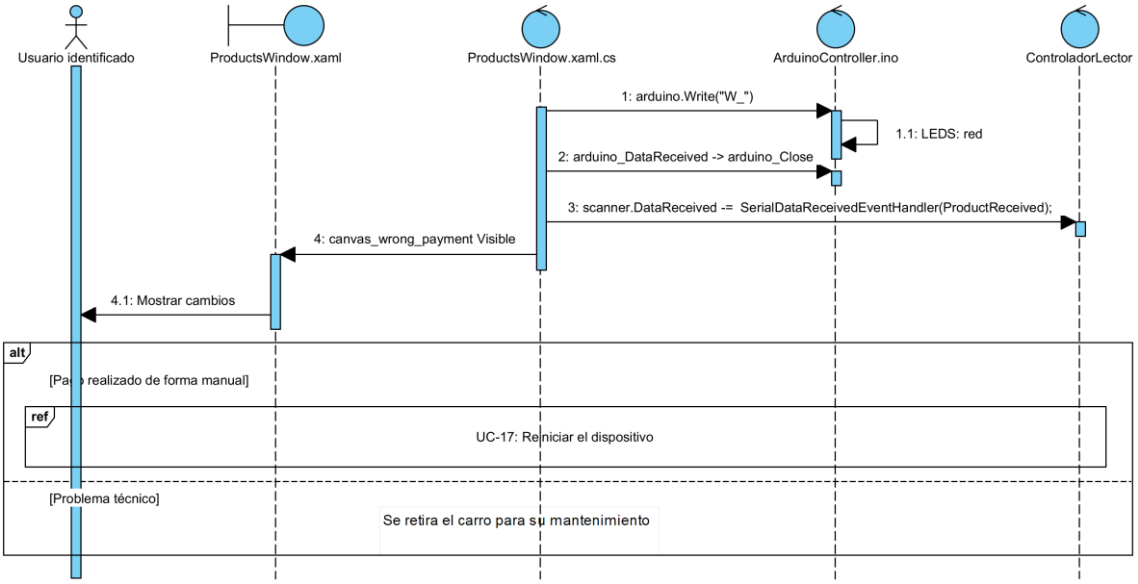

Figura 33. Diagrama de secuencia del UC-23: Redirigir a caja de cobro

· UC-24: Actualizar ventas

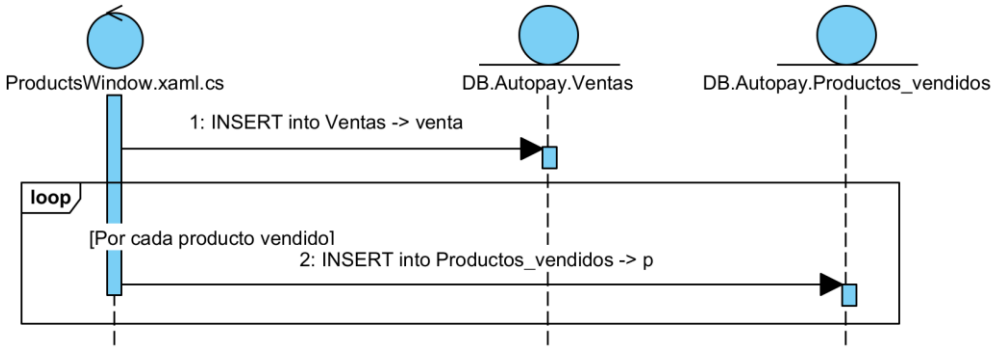

Figura 34. Diagrama de secuencia del UC-24: Actualizar ventas

3) Diseño de la base de datos

En este apartado se expone la estructura de la base de datos creada para el almacenamiento y gestión de los requisitos de información expuestos en el documento "*Anexo II: Especificación de requisitos software*".

Para la mayor seguridad, optimización y rapidez, se ha optado por la implementación de una base de datos relacional basada en MySQL.

Dicha base de datos es alojada en el propio sistema para la simplificación de conexiones debido a que se ha desarrollado un único dispositivo. Para el uso de múltiples dispositivos sería necesaria una ligera modificación en la estructura, alojando la base de datos en un servidor central al que se conectarían cada uno de los distintos dispositivos (clientes).

Para trabajar con dicha base de datos de forma rápida y sencilla se han utilizado la aplicación XAMPP y el plugin web phpMyAdmin.

Figura 35. Aplicaciones utilizadas para el desarrollo de la base de datos

Al ser una base de datos relacional su estructura se base en tablas que contienen las distintas tuplas que reflejan la información almacenada.

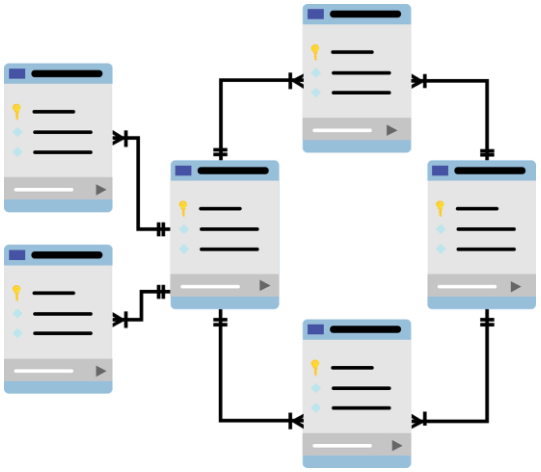

Figura 36. Estructura genérica de una base de datos relacional

A continuación, se muestra el modelo de diseño de la base de datos desarrollada: Autopay.

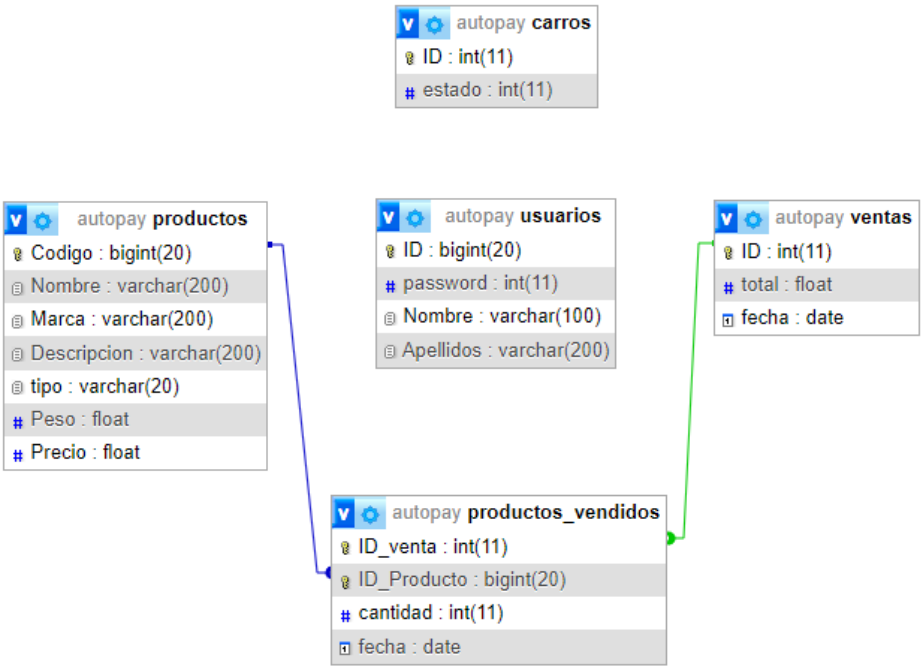

Figura 37. Modelo de diseño de la base de datos

4) Diagrama de despliegue

Para concluir la etapa de diseño se ha desarrollado el diagrama de despliegue, el cual expone las relaciones finales entre los diferentes nodos distribuidos que componen el sistema.

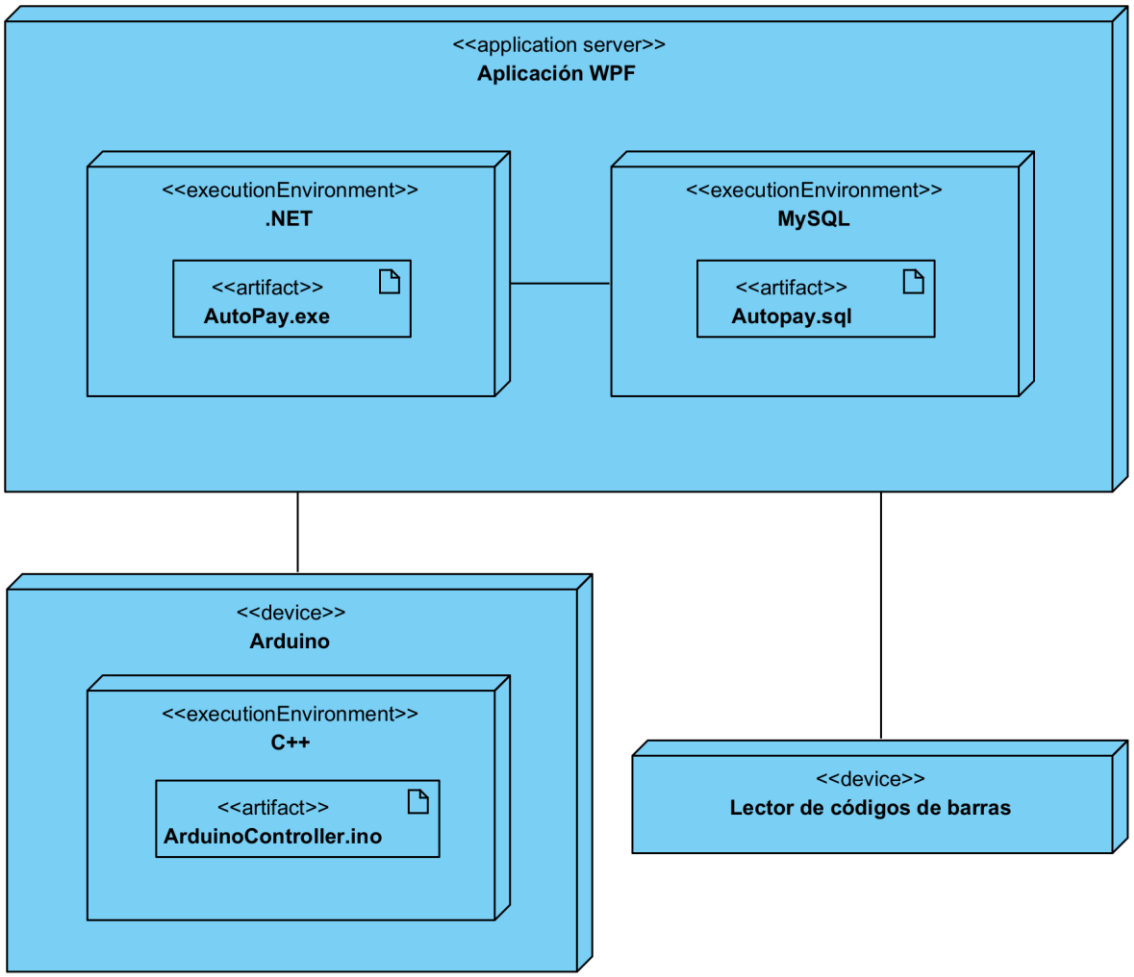

Figura 38. Diagrama de despliegue

5) Bibliografía

[1] Michaelstonis, Erjain, IEvangelist. (2023, 8 junio). *Modelo-Vista-Modelo de vista*. Microsoft Learn. Recuperado 24 de junio de 2023, de https://learn.microsoft.com/es-es/dotnet/architecture/maui/mvvm

[2] Roldan, A. (2020, 18 septiembre). *Patrón de diseño MVVM usando WPF (C#). Parte 1*. Blog Clicko. Recuperado 24 de junio de 2023, de https://blog.clicko.es/patron-diseno-mvvm-usando-wpf-parte-1

[3] Pressman, R. S. (2010). Ingeniería del Software: Un Enfoque Práctico. (7.a ed.). Recuperado 24 de junio de 2023, de http://cotana.informatica.edu.bo/downloads/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF

[4] Moreno García, M. N. *Transparencias de la asignatura Ingeniería del Software II (Tema 7)*. Recuperado 25 de junio de 2023.