Grado en Ingeniería Informática

Trabajo de Final de Grado

Desarrollo de un software con interfaz de usuario y visualizador de llenado de pallet para un robot paletizador

Autor:
Jonatan TOMÁS CARO

Supervisor:
Francisco Javier GARCÍA GANDÍA
Tutor académico:
María de las Mercedes FERNÁNDEZ REDONDO

Fecha de lectura: 12 de Julio de 2021
Curso académico 2020/2021

Resumen

Este documento trata de la memoria técnica de un proyecto que consiste en crear un software para el diseño de los diferentes formatos de pallet que un robot paletizador debe montar. El software tiene que contar con una interfaz de usuario clara y concisa y un visualizador de llenado de pallets. El software debe ser capaz de trabajar con niveles, con diferentes tamaños de cajas y pallets y guardar las configuraciones de los pallets para que sean fácilmente reutilizables. Además de funcionar con los robots paletizadores actuales, también debe funcionar con los más antiguos, de forma que todos los robots trabajen con el mismo software.

Palabras clave

Paletizador, brazo robotico, Visual Studio, C#, HMI, software industrial

Keywords

Palletizer, robotic arm, Visual Studio, C#, HMI, industrial software

Agradecimientos

En primer lugar dar gracias a la Universidad Jaume I como conjunto por la forma de llevar a cabo el aula virtual y la formación académica en general. También por la facilidad de encontrar información que ofrecen, más aun en la época que nos ha tocado vivir por la pandemia a causa del COVID-19.

También dar las gracias a todo el elenco de profesores del grado de Ingeniería Informática en conjunto por su labor y su capacidad de enseñanza a los alumnos.

En lo personal, no puedo olvidarme del apoyo recibido de mi pareja Laura y mis padres José y Juani, ya que durante mi vida académica, ha habido situaciones muy difíciles donde han estado apoyándome en todo momento.

Por último, pero no menos importante, quiero dar mi agradecimiento a mis compañeros de batalla José Carlos Torró Belda y Samir Hanganu, ya que hemos demostrado que cuando unimos fuerzas no hay ningún objetivo o meta que no podamos lograr, es por ello que espero seguir logrando metas juntos.

Índice general

Capítulo 1

Introducción

Índice

1.1. Contexto y motivación del proyecto

El proyecto propuesto en este documento se realiza en el marco de las prácticas para el Proyecto Final de de Grado de Ingeniería Informática. La motivación de estas prácticas viene dada por la posibilidad de aplicar los conocimientos adquiridos durante el grado, así como poner en práctica los conceptos más específicos del Itinerario de Ingeniería de Computadores.

1.1.1. Empresa

Este proyecto se realiza durante la estancia en prácticas en la empresa Robottions, situada en Almazora (Castellón), véase en la figura 1.1, la cual incluye el logotipo de la empresa y su localización. Su gerente y a la vez el supervisor del proyecto es Francisco Javier García Gandía.

Esta compañía ofrece a las empresas soluciones robóticas a problemas logísticos y de producción y también se ocupa de formar personal especializado a nivel de usuario, programador e integrador en robótica industrial. Además, también ofrece servicios a empresas industriales como son:

Figura 1.1: Localización y Logotipo de Robottions

1. Consultoría para la optimización de procesos productivos.
2. Automatización y Robotización de procesos.
   a) Robótica colaborativa.
   b) Intralogística.
   c) Celdas robóticas.
   d) Reparación y puesta a punto de robots.
3. Distribución y formación en SprutCAM. Uno de los softwares de CAM más preparado para el mecanizado con Robots.
4. Desarrollo de Software a Medida.
5. Formación en robótica.
6. I+D+i. Consiste en innovar y no dejar de buscar proyectos interesantes en los que investigar y continuar avanzando.

1.1.2. Motivación del proyecto

En cuanto a robótica industrial, hoy en día, una de las propuestas de automatización más útiles y amortizables que existen son los robots de paletizado. Un robot de paletizado o paletizador es un sistema formado por un brazo robótico y un software que monitorice el mismo. Este brazo robótico se encarga de apilar elementos que salen de una línea de producción, con lo que se automatiza la tarea que siempre se ha realizado manualmente.

El proyecto a desarrollar durante la estancia en prácticas y su principal motivación surge de la necesidad de solucionar un problema de software obsoleto o anticuado. La empresa dispone de robots paletizadores con un software ya antiguo, lo que hace que no puedan realizar algunas tareas que otros robots si ejecutan. Además de este problema anterior, también se han encontrado con el inconveniente de no poder implantar adecuadamente ese software en los robots más modernos de la empresa.

Es por esto, por lo que la empresa Robottions quiere desarrollar un proyecto que consiste en la creación de un software con interfaz de usuario y visualizador de llenado de pallet para los robots paletizadores más modernos.

Este software debe ser capaz de funcionar con los robots paletizadores actuales, pero también con los mas antiguos, de forma que todos los robots trabajen con el mismo software.

El alcance de un robot paletizador es muy importante ya que aporta una gran autonomía en una empresa, aumentando la producción y la capacidad operativa de la misma. Un software adecuado hace que la posibilidad de adaptarlos con facilidad a medida que las necesidades aumenten o cambien las características de los productos, sea sencillo para el cliente, y es con esto, con lo que Robottions aspira a que más empresas se interesen por él.

1.2. Objetivos del proyecto

El principal objetivo de este proyecto es crear un software para el diseño de los diferentes formatos de pallet que un robot paletizador debe montar, este software debe contar con una interfaz de usuario clara y concisa, además de un visualizador de llenado de pallets.

Además del objetivo principal, para este software se han de cumplir los siguientes requisitos:

- Ser capaz de trabajar por niveles.
- Tener varias vistas 2D y como mejora, una vista 3D.
- Ser capaz de utilizar cualquier tipo de tamaño de cajas (deben configurarse en la app).
- Ser capaz de colocar las cajas tanto en vertical como en horizontal.
- Ser multi-pallet (hasta 6) de manera que cada uno de los pallets activos en la configuración actual puede ser del mismo tipo o diferente.
- Tener un identificador de calidad para cada pallet activo, de manera que la palletizadora sepa en qué pallet colocar la caja según su grado de calidad.
- Guardar las configuraciones de pallets para que sean fácilmente reutilizables en un futuro.
- La definición de cada nivel de pallet debe ser rápida.

Por lo tanto, resumiendo con esta aplicación de escritorio se debe poder crear/modificar/activar las diferentes distribuciones de llenado de los pallets. También monitorizar el proceso de paletizado de forma que se pueda seguir en directo y recojamos información de valor.

1.3. Tecnologías que se deben utilizar

En el proyecto se utilizarán varias tecnologías para el adecuado funcionamiento del robot, y además debe ser compatible con el actual software instalado en los robots más antiguos. A continuación, se enumerarán las tecnologías que pueden utilizarse en el proyecto.

- La aplicación se desarrollará en el lenguaje C# y el entorno de desarrollo Visual Studio.

- Para el almacenamiento de los diferentes datos se utilizará el lenguaje extensible o formato XML.
- La iteración con el robot será con el software propio de la empresa KUKA (Fabricantes del brazo robótico).
- Se utilizarán PLCs o Controladores Lógicos Programables para la conexión con el brazo robotico.

1.4. Alcance del proyecto

1.4.1. Alcance funcional

- El producto a desarrollar ha de almacenar en una base de datos en XML toda información sobre los formatos de disposición de las cajas en los pallets, además de los tipos de cajas y pallets de las plantas industriales de los clientes.
- El producto a desarrollar ha de mostrar, mediante vistas de escritorio, elementos gráficos en 2D sobre como están colocadas las cajas en los diferentes niveles de los diferentes pallets de los clientes
- El producto a desarrollar ha de permitir al usuario editar esas vistas para cambiar el formato de los niveles y enviar esa información a un brazo robotico concreto.

1.4.2. Alcance organizativo

- El proyecto se lleva a cabo dentro del departamento de **desarrollo de software a medida** el cual tiene como función estudiar, planificar y elaborar software para empresas industriales de la forma precisa que lo necesite.
- Este producto afectará directamente al departamento de **Automatización y robótica**. Una de las funciones de este departamento es la reparación y puesta a punto de robots, esto muchas veces implica la creación de un software más moderno para que los robots funcionen en base a las nuevas normativas y también solucionar errores de software ya anticuado en las plantas industriales de los clientes.

1.4.3. Alcance informático

- El sistema ha de funcionar con los robots paletizadores actuales, pero también con los mas antiguos, de forma que todos los robots trabajen con el mismo software.
- El sistema ha de conectarse a un HMI de la marca KUKA, para la comunicación entre nuestro software y el brazo robotico.

1.5. Estructura de la memoria

La estructura de la memoria técnica presentada en este documento se compone de 8 capítulos. A continuación se muestra un breve resumen del contenido de cada uno.

En el capítulo 1 se presenta una introducción al trabajo realizado, incluyendo el contexto y motivación del proyecto y los objetivos del mismo.

El capítulo 2 contiene una descripción de la planificación del proyecto, así como un seguimiento de este.

El capítulo 3 tratará de un análisis del sistema desarrollado, incluyendo la definición de los requisitos y las tecnologías utilizadas.

En el capítulo 4 se adjunta el diseño del sistema, incluyendo arquitectura, modo de almacenamiento e interfaces.

El capítulo 5 se presenta una descripción técnica de la implementación del proyecto.

El capítulo 6 contiene una descripción de las pruebas realizadas.

En el capítulo 7 se exponen las mejoras que se podrían realizar sobre el proyecto desarrollado, además del futuro del proyecto total.

Y por último, en el capítulo 8 se verán las conclusiones técnicas de gestión del proyecto y una opinión personal.

Capítulo 2

Planificación del proyecto

Índice

2.1. Metodología

En este proyecto vamos a basarnos en una metodología predictiva, también conocida por metodología waterfall con lo que el proyecto se llevará a cabo siguiendo unas determinadas fases o hitos.

El principal objetivo de este tipo de metodología es su propio carácter predictivo, es la más potente en el entorno en que nos encontramos ya que es estable, hay pocos cambios por parte del cliente.

Esta metodología tiene muchas ventajas, la principal es su presupuesto cerrado, el cual se llega a un acuerdo con el cliente desde el principio, es decir, se pueden hacer pequeños cambios en el software en su proceso de desarrollo, pero el precio no cambiará. También tiene otras ventajas como:

- La facilidad para llevar a cabo la medición del transcurso del proyecto.

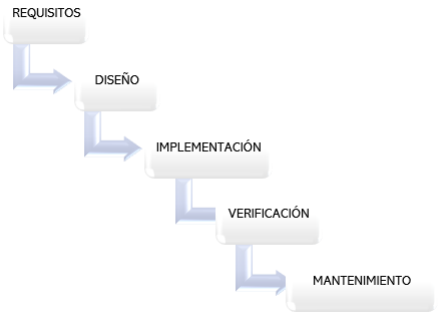

Figura 2.1: Metodología Waterfall

- El cliente no se involucra mucho en el proyecto, cuando se tiene una versión más o menos acabada, se le entrega para que proponga cambios si así lo desea.

También tiene algunas desventajas como la posibilidad que tras unos meses probando el programa, los trabajadores echen en falta alguna funcionalidad sobre la que no se había pensado, en el momento de pedir las características al proveedor del programa. Esto llevaría a un nuevo presupuesto para solucionar el problema, ya que no es un error del proveedor.

2.2. Planificación

El proyecto comenzó el 25 de marzo del 2021 y cumpliendo la planificación la fecha de fin estimada era el 16 de junio del 2021 con el horario acordado entre la empresa y el alumno, de lunes a viernes de 9:00 a 14:00. La planificación inicial para el proyecto es la que se muestra en la tabla 2.1.

TAREAS A REALIZAR	HORAS
Análisis previo del software y del proyecto	25
Profundizar en el lenguaje C# y Visual Studio	30
Desarrollo del software y la interfaz	
Análisis de sistemas y requisitos	10
Diseño y estructura del software	15
Programación e implementación	80
Pruebas y revisión	20
Mantenimiento y actualizaciones	20
Documentación del código fuente	10
Diseño de la usabilidad	10
Comunicación con el brazo robótico (HMI)	40
Comprobación, perfeccionamiento y correcciones	40

Tabla 2.1: Planificación inicial

En la figura 2.1 se ha visto la planificación inicial, en la cual podemos ver una serie de tareas principales del proyecto y otras subtareas. A continuación se desarrollara cada una de las tareas o fases principales de la planificación inicial.

2.2.1. Análisis previo del software y del proyecto

La fase de análisis previo del software y del proyecto tiene dos partes. La primera se realiza antes de empezar el desarrollo, es donde se intenta hacer la primera estructura de la idea basada en lo que se quiere conseguir. Y una vez esa primera estructura se ha desarrollado, es hora de convertirla en el cronograma de un proyecto que es algo más concreto. Este cronograma se puede realizar con varias técnicas, la más utilizada y la que se verá en este proyecto es el diagrama de Gantt.

Diagrama de Gantt

Un diagrama de Gantt es un tipo de diagrama de barras que ilustra el cronograma de un proyecto como se puede ver en la figura 2.2 y en la figura 2.3 usa las pautas y fechas que tendremos para el desarrollo de nuestra aplicación de escritorio. A la izquierda del gráfico hay una lista de las actividades y en la parte superior hay una escala de tiempo. Cada actividad tiene una fecha de inicio, duración y fecha de finalización de la actividad.

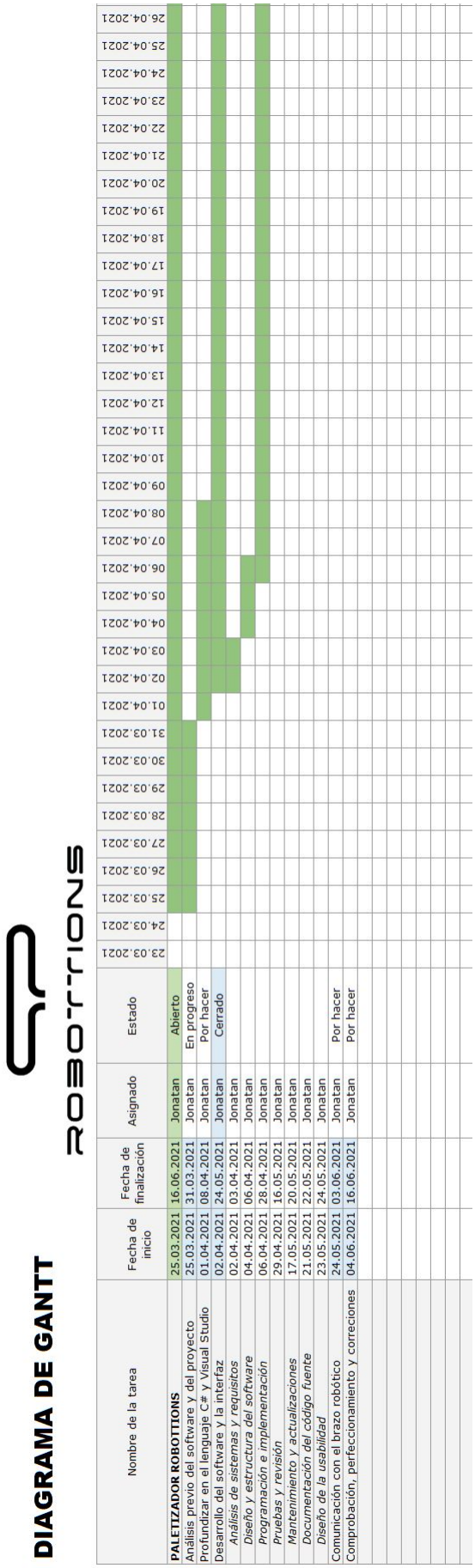

Figura 2.2: Diagrama de Gantt

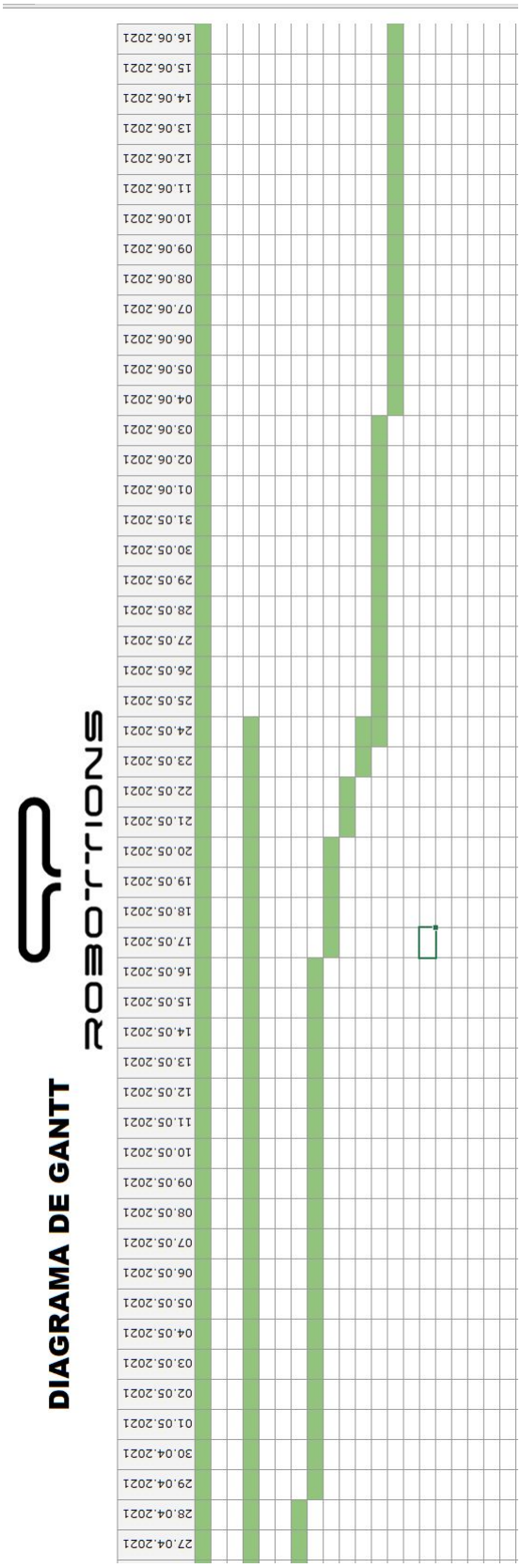

Figura 2.3: Diagrama de Gantt (continuación)

2.2.2. Profundizar en el lenguaje C# y entorno de desarrollo Visual Studio

En esta fase, se busca por parte de la empresa, que conozcamos el entorno de desarrollo donde trabaja, en este caso es Visual Studio, y el lenguaje de programación C#. Como no es habitual que la persona de prácticas venga con este conocimiento aprendido, dejan estas 30 horas para que puedas llevar a cabo un aprendizaje rápido sobre ellos. Al conocer otros lenguajes de programación y que sean tan parecidos, no lleva más de 30 horas saber lo básico del lenguaje.

2.2.3. Desarrollo del software y la interfaz

La fase de Desarrollo del software y la interfaz, es la más amplia, en concreto 165 horas. Esto es debido a que este proyecto generalmente se basa en la creación del software. Esta fase se descompone en subfases, que nombraremos y describiremos a continuación.

- **Análisis de sistemas y requisitos (10 horas)**. Extraer los requisitos del producto de software.
- **Diseño y estructura del software (15 horas)**. Determinar cómo funcionará el programa de forma general, consideraciones sobre la red, hardware, etc.
- **Programación e implementación (80 horas)**. Realizar el programa con las estructuras definidas. Esta duración no es exacta, por la complejidad y conocimiento sobre el lenguaje de programación.
- **Pruebas y revisión (20 horas)**. Detectar disconformidades con los clientes y errores varios.
- **Mantenimiento y actualizaciones (20 horas)**. Apoyo de los usuarios, actualizaciones necesarias, implicaciones y soporte.
- **Documentación del código fuente (10 horas)**. Documentación transparente del código fuente.
- **Diseño de la usabilidad (10 horas)**. La experiencia del usuario.

2.2.4. Comunicación con el brazo robótico

En esta fase, el software que estamos desarrollando debe de comunicarse con el HMI de KUKA.

El HMI KUKA es el software principal de los brazos robóticos de la marca KUKA, que son los utilizados por la empresa Robottions, su función es mostrar información en tiempo real, proporcionar gráficos visuales y digeribles que aporten significado y contexto sobre el estado actual del brazo robótico.

Una de las funciones de nuestro software es comunicarse con dicho HMI para enviar los formatos de los niveles de los pallets que se crean o editan en la aplicación que estamos creando.

2.2.5. Comprobación de funcionamiento, perfeccionamiento y correcciones

Esta es la fase final de la estancia en prácticas, y consiste en comprobar que todo funciona bien, realizando las pruebas necesarias con el brazo robotico, en ocasiones al probar físicamente el software con el brazo robotico, podemos observar como alguna de las funciones pueden perfeccionarse y así realizar alguna tarea más rápida, también este en el momento de hacerlo, y finalmente, si hay algún error in situ, se debe corregir en esta fase final.

2.2.6. Planificación final

Aunque en la estancia de practicas se han conseguido realizar todas las tareas propuestas por la empresa, si es verdad que los tiempos marcados en primer momento no corresponden con los finales, ya que la implementación de la aplicación se ha visto afectada por los cambios propuestos por los clientes y ha durado más. Por otro lado la comunicación con el HMI, que solo se trataba de leer los ficheros XML que se creaban en la aplicación, no ha durado más de 5 horas, ya que se realizó un versión básica del HMI y se comprobó que se podían leer los datos. En la tabla 2.2 se muestra cómo ha quedado finalmente la planificación.

TAREAS A REALIZAR	HORAS
Análisis previo del software y del proyecto	25
Profundizar en el lenguaje C# y Visual Studio	30
Desarrollo del software y la interfaz	
Análisis de sistemas y requisitos	10
Diseño y estructura del software	15
Programación e implementación	**130**
Pruebas y revisión	20
Mantenimiento y actualizaciones	20
Documentación del código fuente	10
Diseño de la usabilidad	10
Comunicación con el brazo robótico	**5**
Comprobación, perfeccionamiento y correcciones	**25**

Tabla 2.2: Planificación final

2.3. Estimación de recursos y costes del proyecto

En esta sección se estiman los recursos directos e indirectos requeridos para el desarrollo del proyecto, así como su coste.

2.3.1. Costes directos

Los recursos directos requeridos para este proyecto se pueden dividir en dos clases, que son: costes humanos y de hardware. En cuanto a los recursos de software, todas las tecnologías utilizadas son de licencia gratuita, por tanto no se incluirán en este apartado.

Costes de hardware

En cuanto a los costes de hardware, los recursos usados para desarrollar el proyecto son:

- Monitor ViewSonic Va2719.
- Teclado inalámbrico Logitech MK540
- Ratón inalámbrico Logitech MK540
- Ordenador portátil MSI Apache Pro GE62

Recurso	Precio	Precio mensual	Precio Estimado
Monitor ViewSonic Va271	149,15 €	149,15 € / 72 = 2,07 €/mes	2,07 * 3 = 6,21 €
Teclado inalámbrico Logitech MK540	39,99 €	39,99 € / 72 = 0,55 €/mes	0,55 * 3 = 1,65 €
Ratón inalámbrico Logitech MK540	19,99 €	19,99 € / 72 = 0.27 €/mes	0,27 * 3 = 0,81 €
Ordenador portátil MSI Apache Pro GE62	1.256,54 €	1256,54 € / 72 = 17,45 €/mes	17,45 * 3 = 52,35 €
		TOTAL	**61,02 €**

Tabla 2.3: Costes de Hardware

En la tabla 2.3 se muestra el desglose del coste en hardware estimado para el desarrollo del proyecto. Partiendo de que la duración del proyecto ha sido de 300 horas equivalente a 3 meses aproximadamente, y que el coste de amortización de cada recurso se ha establecido en 6 años, se ha estimado el precio mensual para cada recurso y se ha multiplicado por el total de meses de duración del proyecto para obtener una estimación del coste.

Después de calcular los costes, se ha obtenido un coste total de hardware de **61,02 €**.

Costes humanos y de contratación

Para el calculo de los costes humanos y de contratación, se ha utilizado la herramienta online `https://stackoverflow.com/jobs/salary`, hemos rellenado los datos que nos pedía la herramienta de la siguiente forma:

- Función principal: Ayudante sin titulación.
- Ciudad: Castellón de la Plana.
- Años de experiencia: 0 años.
- Tecnologías: Java, Python, SQL y C#.

Con estos datos, nos ha calculado un sueldo bruto al año de 14.500 €, que mediante la herramienta `https://cincodias.elpais.com/herramientas/calculadora-sueldo-neto/` corresponde a un salario neto de 12.625€ anuales y 1.052€ mensuales.

Con este salario mensual, partiendo de la base de que se trabaja 8 horas al día, durante 5 días a la semana, que hacen un total de 40 horas semanales y 160 horas mensuales equivale aproximadamente a 1.052 € / 160 = 6,57 €/hora.

Si multiplicamos 6,57 €/hora por las 300 horas que hemos realizado en la empresa, nos da un total de **Coste humano y de contratación por tres meses de 1.972,50 €**

Costes directos TOTALES

Con los datos anteriores, el coste directo total obtenido ha sido de:

1.972,50 + 61,02 = 2.033,50 €.

2.3.2. Costes indirectos

Al tratarse de un proyecto de software, para los costes indirectos solo vamos a tener en cuenta los gastos de luz, alquiler del local e internet. Para estos cálculos vamos a suponer que cada usuario de la empresa gasta lo mismo, por lo que habiendo 16 trabajadores, dividiremos los gastos entre 16.

Como la empresa no nos proporciona el dato de alquiler del local, vamos a suponer viendo las naves en alquiler por la zona, que la nave industrial situada en Almazora tiene un coste de 1200 € mensuales, si tenemos en cuenta que hay trabajando 16 personas, nos sale un total de **75€** mensuales dividido entre 30 días son: **2,5€/día** y por horas serían **0,10€/hora**. Esto multiplicado por 300 horas, nos sale que están pagando por nosotros unos **30€**.

Para el gasto de electricidad, suponemos que se paga una media de 150 euros al mes, que son 5€ diarios y 0,20€ por hora. Si esto lo dividimos entre 16 personas que somos, nos queda 0,01 € la hora. Si multiplicamos esto por 300, aproximadamente por nosotros estén pagando unos 3 € por las 300 horas.

Por último una factura de internet por la zona sale por unos 75 € aproximadamente, con lo pagarían por nosotros, 1,5 € por 300 horas.

Costes indirectos TOTALES

Con los datos anteriores, el coste indirecto total asciende a:

30 + 3 + 1,5 = 34,5 € por 300 horas de trabajo.

2.4. Seguimiento del proyecto

Como ya se ha mencionado anteriormente, en el proyecto se va a seguir una metodología predictiva, con lo que el proyecto se llevará a cabo siguiendo unas determinadas fases o hitos. Como hemos observado en la Tabla 2.1 anterior, a cada fase o hito, se le añadirá un identificador de fase, la descripción y el estado se le asignará un tiempo de finalización, este tiempo, en horas, será estimado, ya que no se sabe con certeza cuanto durarán cada una de las tareas. Algunas fases o hitos, a su vez, se descomponen en subfases, por ejemplo la fase de Desarrollo del software y la interfaz, que es la más amplia, en concreto 165 horas, contiene siete subfases.

En cuanto al supervisor del trabajo realizado, cada semana, o cada vez que se finalizaba una de las fases a realizar, el supervisor en la empresa, Francisco Javier García Gandía, hacia un seguimiento de los progresos conseguidos, además de ofrecer su ayuda cuando era necesario. De esta forma, constantemente se recordaba cuáles eran las tareas realizadas, los problemas a resolver en las tareas ya realizadas y las siguientes acciones. Además, la empresa estaba en contacto con el cliente para mostrar los avances y este añadía si quería alguna cosa ya realizada de otra forma, cambiar la localización de algunos botones, o si quería mostrar la información de alguna otra forma. También se comentaban posibles funcionalidades nuevas.

Por otro lado, cada dos semanas, el alumno redactaba informes quincenales donde documentaba e informaba a la tutora del proyecto, la profesora María de las Mercedes Fernández Redondo, de los avances conseguidos en las quincenas anteriores y los objetivos que se deben llevar a cabo durante las siguientes quincenas.

Capítulo 3

Requisitos y Análisis del sistema

Índice

En este capítulo hablaremos de los requisitos y el análisis del sistema, esta sección contiene las tareas que tienen que ver con las funcionalidades del sistema o con el estudio de ellas, dicho de otro modo, es la descripción de como se debe comportar un sistema a desarrollar. Para ello, se deben elaborar unos modelos que permiten ver las iteraciones que el sistema tendrá con los diferentes usuarios.

3.1. Requisitos del sistema (Casos de uso)

Los requisitos del sistema es una parte crucial del proyecto, ya que en ellos se estudian las características o los aspectos que deben cumplir las distintas funciones que tiene un sistema, en cuanto a condiciones tecnológicas, legales, de calidad, funcionales o de seguridad. La definición incorrecta de estos requisitos puede resultar un diagrama de casos de uso inexacto y con ello se pueden perder recursos o desperdiciar tiempo.

3.1.1. Diagrama de Casos de uso

En este apartado veremos el diagrama de casos de uso, este diagrama se utiliza para la obtención de requisitos, es muy útil y una gran herramienta ya que se trata de un esquema gráfico donde se puede ver a simple vista las funcionalidades que un sistema software debe

tener. Este diagrama está relacionado con una serie de actores, que son los que se ven afectados por él, tanto si se aprovechan directamente del software, los actores primarios o secundarios, como si se ven envueltos de alguna otra forma en alguna fase, los actores pasivos. En la figura 3.1 podemos ver el diagrama de casos de uso que corresponde con este proyecto.

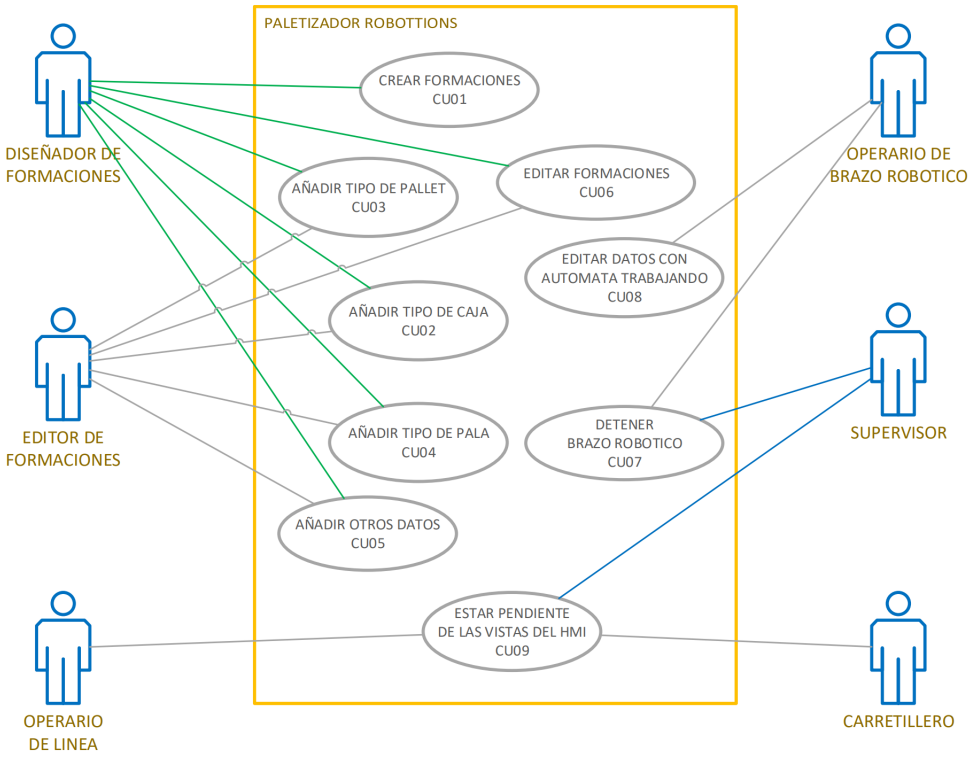

Figura 3.1: Diagrama de Casos de Uso

3.1.2. Resumen de casos de uso y actores

Antes de ver algunos ejemplos de diagramas de casos de uso, se ha realizado el resumen de la figura 3.1 (diagrama de casos de uso), donde se puede ver que funciones realiza cada actor. Este resumen se muestra en la Tabla 3.1.

Actor	Caso de uso
Diseñador de Formaciones	CU01 Crear Formaciones CU02 Añadir tipo de caja CU03 Añadir tipo de pallet CU04 Añadir tipo de pala CU05 Añadir otros datos CU06 Editar Formaciones
Editor de Formaciones	CU02 Añadir tipo de caja CU03 Añadir tipo de pallet CU04 Añadir tipo de pala CU05 Añadir otros datos CU06 Editar Formaciones
Operario de línea	CU09 Estar pendiente de las vistas del HMI
Operario de Brazo Robótico	CU08 Editar datos con Autómata trabajando CU07 Detener brazo robótico
Supervisor	CU07 Detener brazo robótico CU09 Estar pendiente de las vistas del HMI
Carretillero	CU09 Estar pendiente de las vistas del HMI

Tabla 3.1: Resumen Diagrama Casos de Uso

3.1.3. Descripción de los actores

- **Diseñador de formaciones**. El diseñador de formaciones es la persona encargada de crear por primera vez el diseño de cada formato del pallet, esto es, la forma y posiciones de cada caja en cada piso del pallet. Este usuario por regla general tiene el programa en su ordenador personal y está ubicado en las oficinas de la empresa.
- **Editor de formaciones**. El editor de formaciones es la persona que una vez ya tiene las formaciones o diseño de formatos del pallet, hablará con los operarios y tiene la potestad de editar cualquier dato de las formaciones ya creadas. El puesto de trabajo de este usuario suele ser el de encargado de la sección donde está el brazo robótico.
- **Supervisor**. El supervisor es el usuario que está en continua vigilancia observando las pantallas que muestran en tiempo real la tarea del brazo robotico, y en caso de que algo falle o se produzca algún error debe avisar al operario del brazo robótico para que se encargue. Este usuario puede parar el brazo en caso de urgencia.
- **Operario de brazo robótico**. Este operario tiene el HMI, que es el programa en tiempo real del robot en un dispositivo móvil. Es el encargado de editar alguna posición o algún dato imprescindible para que el robot haga bien su función. También puede parar el mismo cuando lo necesite, bien para editar datos o bien por algún tipo de error. Este usuario suele estar al lado del brazo robótico y vigilando que su función in situ esté bien realizada.
- **Operario de linea**. Los operarios de linea tiene que ver el HMI cada cierto tiempo, para comprobar si deben girar alguna caja o moverla en caso que sea necesario en la propia linea de producción para que el brazo robótico coja correctamente la caja.
- **Carretillero**. El carretillero debe de comprobar las pantallas que visualizan los pallets en los que habrá un aviso de que el pallet está lleno, para recogerlo con la máquina elevadora, y poner otro pallet vacío en su posición.

3.1.4. Plantillas de Casos de Uso

La plantilla de casos de uso es una tabla que modela una situación de como un actor concreto utilizará el sistema para lograr un objetivo específico. Un caso de uso efectivo debe proporcionar una descripción detallada paso a paso de cómo el sistema será utilizado por sus actores para lograr el resultado planificado.

CU01 Crear Formaciones	
Autor: Jonatan Tomás Revisor: Jonatan Tomás Fuente: Diseñador de Formaciones	Fecha de creación: 10/06/2021 Fecha de revisión: 10/06/2021 Fecha de aprobación: 10/06/2021 Versión: 1.0
Descripción: El usuario se dispone a realizar desde cero una nueva formación Como empieza: El usuario selecciona los datos correspondientes (Pasta, Formato, Plato, Cliente, Tipo de caja, Tipo de pallet) y pulsa CREAR FORMACIÓN.	
Pasos escenario 1: 1. El sistema demanda los datos de la nueva formación. 2. El usuario selecciona los datos. 3. El sistema valida si la formación existe, y en caso negativo, abre la pantalla de creación de la nueva formación. 4. El usuario rellena el formulario de alturas, selecciona el tipo de pala, crea el layout o layouts que crea conveniente, y pulsa GUARDAR. 5. El sistema comprueba que todo esté correctamente rellenado y en caso afirmativo guarda la nueva formación. Pasos escenario 2: 1. El sistema comprueba que la formación ya existe y quiere modificar sus datos. 2. El sistema muestra los datos de la formación existente. 3. El usuario introduce o modifica los datos a su conveniencia y guarda los datos. 4. El sistema comprueba las modificaciones y guarda la formación ya existente.	

Tabla 3.2: Caso de Uso. Crear Formación

CU02 Añadir tipo de caja	
Autor: Jonatan Tomás Revisor: Jonatan Tomás Fuente: Diseñador de Formaciones	Fecha de creación: 12/06/2021 Fecha de revisión: 12/06/2021 Fecha de aprobación: 12/06/2021 Versión: 1.0
Descripción: El usuario se dispone a introducir un tipo de caja en el programa. Como empieza: El usuario accede al apartado de Configuración de Cajas.	
Pasos escenario 1: 1. El sistema demanda los datos de la nueva caja. 2. El usuario introduce las medidas y el nombre de la caja. 3. El sistema valida si la formación existe, y en caso negativo, introduce la caja. 4. El sistema informa al usuario de que la caja ha sido añadida. Pasos escenario 2: 1. El sistema comprueba que la caja ya existe y quiere modificar sus datos. 2. El sistema muestra los datos de la caja existente. 3. El usuario introduce o modifica los datos a su conveniencia y guarda los datos. 4. El sistema comprueba las modificaciones y guarda la caja ya existente.	

Tabla 3.3: Caso de Uso. Añadir nueva caja

CU03 Añadir tipo de pallet	
Autor: Jonatan Tomás Revisor: Jonatan Tomás Fuente: Diseñador de Formaciones	Fecha de creación: 12/06/2021 Fecha de revisión: 12/06/2021 Fecha de aprobación: 12/06/2021 Versión: 1.0
Descripción: El usuario se dispone a introducir un tipo de pallet en el programa. Como empieza: El usuario accede al apartado de Configuración de Pallets.	
Pasos escenario 1: 1. El sistema demanda los datos del nuevo pallet 2. El usuario introduce las medidas y el nombre del pallet 3. El sistema valida si el pallet existe, y en caso negativo, introduce el pallet 4. El sistema informa al usuario de que el pallet ha sido añadido Pasos escenario 2: 1. El sistema comprueba que el pallet ya existe y quiere modificar sus datos. 2. El sistema muestra los datos del pallet existente. 3. El usuario introduce o modifica los datos a su conveniencia y guarda los datos. 4. El sistema comprueba las modificaciones y guarda el pallet ya existente.	

Tabla 3.4: Caso de Uso. Añadir nuevo pallet

CU04 Añadir tipo de pala	
Autor: Jonatan Tomás Revisor: Jonatan Tomás Fuente: Diseñador de Formaciones	Fecha de creación: 12/06/2021 Fecha de revisión: 12/06/2021 Fecha de aprobación: 12/06/2021 Versión: 1.0
Descripción: El usuario se dispone a introducir un tipo de pala en el programa. Como empieza: El usuario accede al apartado de Configuración de Recetas.	
Pasos escenario 1: 1. El usuario entra en el editor de recetas y pulsa el apartado crear nueva pala. 1. El sistema demanda los datos de la nueva pala. 2. El usuario introduce la longitud y el nombre de la pala. 3. El sistema valida si el pala existe, y en caso negativo, introduce el pala. 4. El sistema informa al usuario de que el pala ha sido añadido. Pasos escenario 2: 1. El sistema comprueba que el pala ya existe. 2. El sistema informa que no se puede añadir la pala porque ya existe.	

Tabla 3.5: Caso de Uso. Añadir nueva pala

CU05 Añadir otros datos	
Autor: Jonatan Tomás Revisor: Jonatan Tomás Fuente: Diseñador de Formaciones	Fecha de creación: 12/06/2021 Fecha de revisión: 12/06/2021 Fecha de aprobación: 12/06/2021 Versión: 1.0
Descripción: El usuario se dispone a introducir nuevos datos en el programa (Tipo de pasta, Formato de pieza, Tipo de plato, Cliente) Como empieza: El usuario accede al apartado de Configuración de Recetas.	
Pasos escenario 1: 1. El usuario entra en el editor de recetas y pulsa el apartado que corresponda para crear el nuevo dato. 1. El sistema demanda los datos. 2. El usuario introduce el nombre del dato que quiera añadir. 3. El sistema valida si el nombre existe, y en caso negativo, introduce el dato en la lista que corresponda.. 4. El sistema informa al usuario de que el objeto ha sido añadido. Pasos escenario 2: 1. El sistema comprueba que el dato ya existe. 2. El sistema informa que no se puede añadir el nuevo objeto porque ya existe.	

Tabla 3.6: Caso de Uso. Añadir otros datos

CU06 Editar Formaciones	
Autor: Jonatan Tomás Revisor: Jonatan Tomás Fuente: Editor de Formaciones	Fecha de creación: 12/06/2021 Fecha de revisión: 12/06/2021 Fecha de aprobación: 12/06/2021 Versión: 1.0
Descripción: El usuario se dispone a modificar una formación. Como empieza: El usuario entra en configurar recetas y pulsa EDITAR FORMACIÓN.	
Pasos escenario 1: 1. El sistema demanda los datos de la formación existente. 2. El usuario selecciona los datos. 3. El sistema valida si la formación existe, y en caso afirmativo, abre la pantalla de edición de la formación. 4. El usuario edita el formulario como deseé y pulsa "GUARDAR". 5. El sistema comprueba que todo este correctamente rellenado y en caso afirmativo guarda la formación.	

Tabla 3.7: Caso de Uso. Editar formaciones

CU07 Detener brazo robótico	
Autor: Jonatan Tomás Revisor: Jonatan Tomás Fuente: Supervisor.	Fecha de creación: 12/06/2021 Fecha de revisión: 12/06/2021 Fecha de aprobación: 12/06/2021 Versión: 1.0
Descripción: El usuario se dispone a parar el brazo robótico. Como empieza: El usuario comprueba las vistas en tiempo real del brazo robótico.	
Pasos escenario 1: 1. El usuario comprueba que el robot está funcionando mal mediante las vistas. 2. El usuario pulsa el botón de parada del brazo robótico. 3. El sistema desconecta el autómata 4. El sistema muestra en pantalla que el brazo no está funcionando.	

Tabla 3.8: Caso de Uso. Detener brazo robótico

CU08 Editar datos con Autómata trabajando	
Autor: Jonatan Tomás Revisor: Jonatan Tomás Fuente: Operario de Brazo Robótico	Fecha de creación: 12/06/2021 Fecha de revisión: 12/06/2021 Fecha de aprobación: 12/06/2021 Versión: 1.0
Descripción: El usuario se dispone a parar el brazo robótico. Como empieza: El usuario comprueba el funcionamiento del robot.	
Pasos escenario 1: 1. El usuario comprueba que el robot no está funcionando correctamente. 2. El usuario pulsa el botón de parada del brazo robótico. 3. El sistema desconecta el autómata y muestra la pantalla de edición del mismo. 4. El usuario edita los datos correspondientes para su correcto funcionamiento. 5. El sistema comprueba si los datos son correctos y guarda la configuración 6. El sistema pulsa el botón para volver a conectar el autómata. 7. El sistema reanuda el trabajo y vuelve a poner en marcha el brazo robótico.	

Tabla 3.9: Caso de Uso. Editar datos con autómata trabajando

CU09 Visualizar HMI	
Autor: Jonatan Tomás Revisor: Jonatan Tomás Fuente: Operario de línea, Carretillero	Fecha de creación: 12/06/2021 Fecha de revisión: 12/06/2021 Fecha de aprobación: 12/06/2021 Versión: 1.0
Descripción: El usuario está pendiente de las pantallas en tiempo real. Como empieza: El usuario comprueba las vistas de los pallets en directo.	
Pasos escenario 1 (Operario de línea): 1. El sistema muestra la posición de las cajas en las vistas. 2. El usuario comprueba que una caja no está en la posición que indica la vista en directo del pallet. Pasos escenario 2 (Carretillero): 1. El sistema muestra un aviso en pantalla de que un pallet esta completo. 2. El usuario comprueba que el pallet esta lleno y lo retira. 3. El sistema indica que no hay pallet en esa posición. 4. El usuario pone de nuevo el pallet en la posición del sensor. 5. El sistema comprueba que ya hay pallet y indica al brazo que puede seguir llenando en esa posición.	

Tabla 3.10: Caso de Uso. Visualizar HMI

3.2. Análisis del sistema

El objetivo del análisis del sistema es transformar la definición de requisitos en una especificación de software. La especificación de software es una documentación completa y precisa de que tiene que realizar el sistema para cubrir los requisitos de usuario.

En este apartado se mostrará el diagrama de clases y su documentación, este diagrama consiste en una visión general de las clases, los atributos y las asociaciones que se puede ver en la aplicación software.

3.2.1. Diagrama de clases

El diagrama de clases representa una primera abstracción de la información que se debe empaquetar y manipular en el sistema de software. Se muestran las características, las relaciones y las funcionalidades de alto nivel, para ello se debe tener en cuenta los casos de usos y definirlos de manera explícita.

En la figura 3.2 puede verse el diagrama de clases de nuestra aplicación, en el podemos ver todas las clases de nuestro software y como se relacionan con las demás clases. En los siguientes apartados explicaremos con detalle para que se utiliza cada una de las clases, así como cada objeto de la misma.

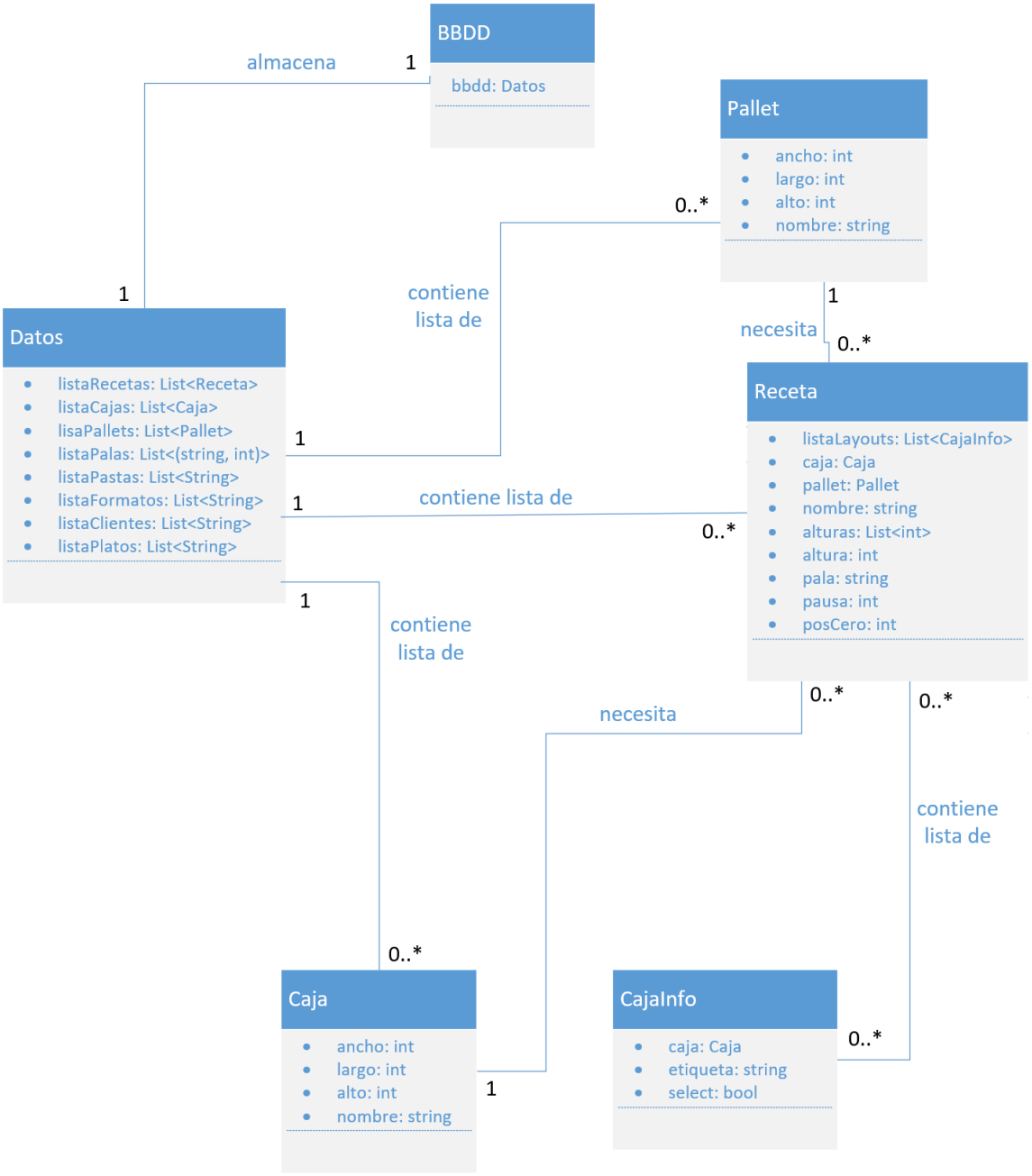

Figura 3.2: Diagrama de clases

3.2.2. Documentación del diagrama de Clases

La Documentación del Diagrama de Clases proporciona una explicación de todas las clases de nuestra aplicación, así como una vista lógica de las tablas que incluye todas las características, atributos y relaciones que hemos visto en la figura 3.2.

Desde la tabla 3.11 a la tabla 3.16, se describirán cada una de las clases del diagrama de clases (figura 3.2), así como la definición de sus atributos u objetos. En la parte inferior de cada tablas se explicará porque es necesario que exista tal clase, esto es lo que se denomina Validación.

Clase: Datos	
Autor: JTC Revisor: JTC Fuente: App	Fecha de creación: 14/06/2021 Fecha de revisión: 14/06/2021 Fecha de aprobación: 14/06/2021 Versión: 1.0
Descripción: (1) Esta clase se utiliza para agrupar todas las listas con los diferentes objetos de la aplicación. (2) Cada lista de esta clase contiene todos los objetos existentes de cada tipo.	
Descripción de cada atributo: (no obvio)	ListaRecetas: Contiene las formaciones, puede estar vacía ListaCajas: Puede estar vacía, siempre que formaciones esté vacía, y no puede tener nombres repetidos. LisaPallets: Puede estar vacía, siempre que formaciones esté vacía, y no puede tener nombres repetidos. ListaPalas: Cada pala tiene que incluir una longitud. ListaPastas: Lista utilizada como parte del nombre de las formaciones. ListaFormatos: Lista utilizada como parte del nombre de las formaciones. ListaClientes: Lista utilizada como parte del nombre de las formaciones. ListaPlatos: Lista utilizada como parte del nombre de las formaciones. ***El nombre de la formación será la unión de: pasta-formato-cliente-plato**
Validación: Esta clase es necesaria para agrupar todos los datos de la aplicación y luego enviarlos a la base de datos.	

Tabla 3.11: Clase: Datos

Clase: BBDD	
Autor: JTC Revisor: JTC Fuente: App	Fecha de creación: 14/06/2021 Fecha de revisión: 14/06/2021 Fecha de aprobación: 14/06/2021 Versión: 1.0
Descripción: (1) Esta clase se utiliza para almacenar todos los datos de la aplicación. (2) Esta clase es la encargada de almacenar los datos en el disco C, mediante xml.	
Descripción de cada atributo: (no obvio)	bbdd: Almacena un objeto de la clase datos, que a su vez, almacena todas las listas de cada objeto que se debe guardar en el disco duro de nuestro PC.
Validación: Esta clase es necesaria para guardar en una serie de archivos XML de nuestro ordenador todos los datos del programa y después cargarlos de ellos.	

Tabla 3.12: Clase: BBDD

Clase: Pallet	
Autor: JTC Revisor: JTC Fuente: App	Fecha de creación: 14/06/2021 Fecha de revisión: 14/06/2021 Fecha de aprobación: 14/06/2021 Versión: 1.0
Descripción: (1) Esta clase representa un pallet para el brazo robótico. (2) Esta clase es la encargada de proporcionar el tamaño del contendedor donde añadiremos las cajas en la vista de creación del layout.	
Descripción de cada atributo: (no obvio)	Ancho: Número en mm de máximo 4 cifras y mínimo de 3. Largo: Número en mm de máximo 4 cifras y mínimo de 3. Alto: Número en mm de máximo 3 cifras y mínimo de 2. Nombre: Solo puede contener números o letras y no repetirse.
Validación: Esta clase es necesaria para almacenar todos los tipos de pallets que utilizará el brazo robótico.	

Tabla 3.13: Clase: Pallet

Clase: Caja	
Autor: JTC Revisor: JTC Fuente: App	Fecha de creación: 14/06/2021 Fecha de revisión: 14/06/2021 Fecha de aprobación: 14/06/2021 Versión: 1.0
Descripción: (1) Esta clase representa una caja para el brazo robótico. (2) Esta clase es la encargada de proporcionar el tamaño de una caja, que luego añadiremos en la vista de creación del layout.	
Descripción de cada atributo: (no obvio)	Ancho: Número en mm de máximo 3 cifras y mínimo de 3. Largo: Número en mm de máximo 3 cifras y mínimo de 3. Alto: Número en mm de máximo 3 cifras y mínimo de 2. Nombre: Solo puede contener números o letras y no repetirse.
Validación: Esta clase es necesaria para almacenar todos los tipos de cajas que utilizará el brazo robótico.	

Tabla 3.14: Clase: Caja

Clase: Receta	
Autor: JTC Revisor: JTC Fuente: App	Fecha de creación: 14/06/2021 Fecha de revisión: 14/06/2021 Fecha de aprobación: 14/06/2021 Versión: 1.0
Descripción: (1) Esta clase representa una receta o formación para el brazo robótico. (2) Esta clase es la encargada de guardar el formato o diseño de cada layout en el pallet y almacenar cuantas alturas y que tipo de layout debe tener cada una de las alturas.	
Descripción de cada atributo: (no obvio)	ListaLayouts: 3 objetos, uno por cada layout, guarda el formato de cada uno de los tres layouts. Caja: El tipo de caja utilizada en la formación, máximo:1. Pallet: El tipo de pallet utilizada en la formación, máximo:1. Nombre: Formado por pasta-formato-plato-cliente-caja-pallet. Alturas: Una lista de que layout va en cada altura (1,2 o 3). Altura: Es la altura máxima del pallet. Pala: Tipo de pala que usará la máquina elevadora. Pausa: Si se necesita parar el robot en cada altura (Si o No). PosCero: En que esquina del pallet debe empezar el robot.
Validación: Esta clase es necesaria para que el brazo robótico sepa como colocar cada una de las cajas en cada altura de cada pallet.	

Tabla 3.15: Clase: Receta

Clase: CajaInfo	
Autor: JTC Revisor: JTC Fuente: App	Fecha de creación: 14/06/2021 Fecha de revisión: 14/06/2021 Fecha de aprobación: 14/06/2021 Versión: 1.0
Descripción: (1) Esta clase se utiliza para que el robot paletizador sepa la posición de la caja. (2) Esta clase es la encargada guardar la posición de la caja mediante la etiqueta, el robot sabrá como girar la caja sabiendo en que parte va la etiqueta de la misma.	
Descripción de cada atributo: (no obvio)	Caja: El objeto Caja, cada CajaInfo equivale a una caja. Etiqueta: Izquierda, Derecha, Arriba o Abajo Select: Booleano utilizado para editar la caja
Validación: Esta clase es necesaria para que el brazo robótico sepa la posición de la etiqueta, con lo que también sabrá el giro que tendrá que darle a la caja.	

Tabla 3.16: Clase: CajaInfo

3.2.3. Asociaciones

En el diagrama de clases podemos ver las relaciones que existen entre ellas, estas conexiones contienen una multiplicidad, que viene dada por los objetos que intervienen en la relación. Cada una de las asociaciones tiene dos multiplicidades, una para cada extremo y puede ser de diferentes tipos, podemos ver en la tabla 3.17 los tipos de multiplicidad que existen.

0..1	- No es obligatorio que tenga ningún objeto conectado. - Como máximo puede estar conectado con uno.
1..1	- Para todos, es necesario estar conectado con almenos un objeto. - Como máximo puede estar conectado con un objeto.
0..*	- Puede ser que no tenga ningún objeto (no es obligatorio). - Puede estar conectado con más de un objeto.
1..*	- Para todos, es necesario estar conectado con un objeto. - Puede estar conectado con más de uno.

Tabla 3.17: Multiplicidad en las asociaciones

A continuación mostraremos una serie de tablas, cada una pertenece a una de las relaciones de nuestro diagrama de clases.

Asociación: contiene lista de	
Autor: JTC Revisor: JTC Fuente: App	Fecha de creación: 14/06/2021 Fecha de revisión: 14/06/2021 Fecha de aprobación: 14/06/2021 Versión: 1.0
Clases 1: Datos - Pallet Clases 2: Datos - Receta Clases 3: Datos - Caja Clases 4: Receta - CajaInfo Descripción: La Clase 1 contiene una lista de varios objetos o ninguno de la Clase 2	
Multiplicidad	Clases 1 - 3: Rol 1: Datos 1..1 (1) Rol 2: Pallet, Receta, Caja 0..* (1) Clase 4: Rol 1: Receta 0..* Rol 2: CajaInfo 0..*

Tabla 3.18: Asociación: contiene lista de

Asociación: almacena	
Autor: JTC Revisor: JTC Fuente: App	Fecha de creación: 14/06/2021 Fecha de revisión: 14/06/2021 Fecha de aprobación: 14/06/2021 Versión: 1.0
Clases 1: BBDD - Datos Descripción: La base de datos (BBDD) almacena los datos del programa.	
Multiplicidad	Rol 1: BBDD 1..1 (1) Rol 2: Datos 1..1 (1)

Tabla 3.19: Asociación: almacena

Asociación: necesita	
Autor: JTC Revisor: JTC Fuente: App	Fecha de creación: 14/06/2021 Fecha de revisión: 14/06/2021 Fecha de aprobación: 14/06/2021 Versión: 1.0
Clases 1: Receta - Pallet Clases 2: Receta - Caja Descripción: La Clase 1 necesita un objeto de la segunda clase.	
Multiplicidad	Clases 1 - 2: Rol 1: Receta 1..1 (1) Rol 2: Pallet, Caja 0..* (1)

Tabla 3.20: Asociación: necesita

Capítulo 4

Diseño del sistema

Índice

El diseño del proyecto consiste en la especificación física del software que se ha documentado en el análisis de manera lógica. Tiene que ser fiable, mantenible, eficiente y con una interfaz adecuada. Esta fase se centra en el cómo, muestra el aspecto físico de los datos.

4.1. Diagrama de clases de diseño

Los diagramas de clases de diseño generalmente se crean en paralelo con el diagrama de clases. El procedimiento de trabajo es el mismo pero con algunas modificaciones. El cambio consiste en agregar operaciones de programación a las clases. En la figura 5.1 podemos observar el diagrama de clases de diseño, añadiendo estas modificaciones.

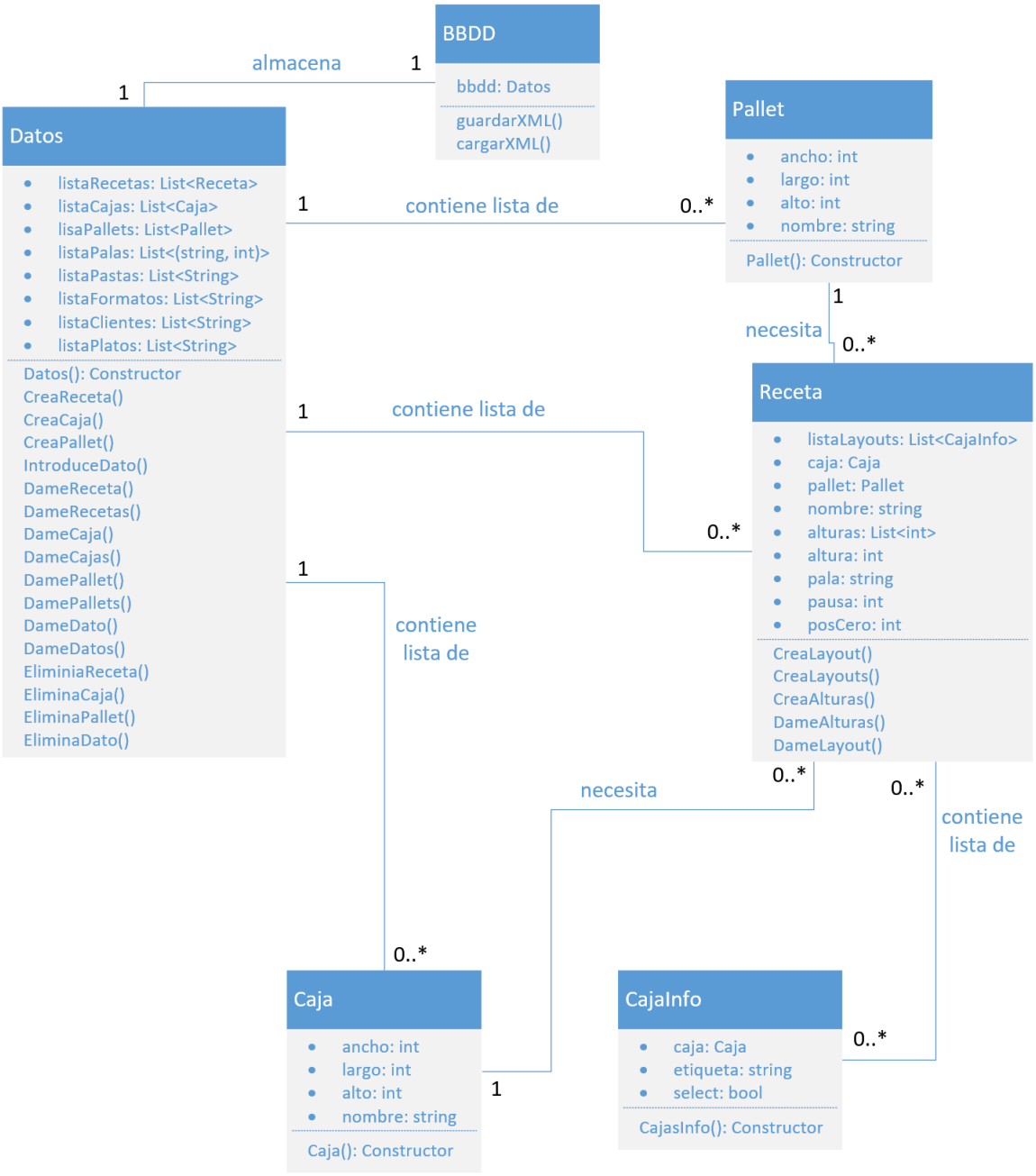

Figura 4.1: Diagrama de Clases de Diseño

4.2. Diseño de la arquitectura del sistema

Después de haber visto el análisis del sistema, y como quedaría el diagrama de clases con las operaciones incluidas, pasamos ha hablar del diseño de la arquitectura del sistema.

Como ya hemos visto varias veces durante la memoria, el proyecto total consta de tres partes, la primera parte es la aplicación de escritorio, donde se crearán los diseños de formaciones de cajas que tendrá cada altura de cada pallet, la segunda parte, es otra aplicación de software, la llamada HMI, esta aplicación es la que se comunicara con la anterior y la que monitorizará en tiempo real al brazo robótico. Por último tenemos el KCP, un dispositivo que será el cerebro del brazo robótico, con el que se manejará el robot manualmente, en caso de ser necesario.

Estás aplicaciones de escritorio y el dispositivo, deben funcionar de forma aislada o independientes. Es decir, una vez el primer software ha creado una serie de formaciones o recetas para utilizar en los pallets, no se tiene que volver a arrancar, a no ser que se quieran agregar nuevas combinaciones, lo mismo sucede con el HMI, aunque debe estar en funcionamiento para ver en tiempo real la tarea que está realizando el robot, debe poder funcionar sin él. Lo único que tiene que estar en funcionamiento obligatoriamente al mismo tiempo es el dispositivo KCP y el brazo robótico.

4.2.1. Almacenamiento de datos

Nuestro sistema, utilizará ficheros con formato XML para el almacenamiento de los datos, cada una de nuestras aplicaciones tendrán una clase encargada de generar o bien editar ficheros XML y otra clase para cargar o leer los datos.

El XML es muy bueno en nuestro caso por dos grandes razones, una por su simplicidad para tomar grandes fragmentos de información y introducirlos en un solo documento, y la segunda es por el poco espacio que ocupan.

4.2.2. Diseño de arquitectura del sistema completo

En la 4.2 podemos ver la arquitectura del sistema completo a desarrollar. En la parte superior de la figura, se encuentra el software paletizador, que es el que hemos desarrollado en esta memoria. Este software almacena los datos en una serie de ficheros con el formato XML. La siguiente imagen de la figura es la aplicación es el HMI, que cargara los datos desde el fichero XML creado con el software anterior para saber el formato que tendrá cada altura de cada pallet mostrado en la imagen en tiempo real. Esta aplicación enviara sus datos directamente al KCP, el dispositivo que tendrá el operario del brazo robótico en mano, para saber si el robot realiza bien su tarea o si por el contrario tiene que modificar algo. El KCP se comunicara de forma inalámbrica con el autómata, enviando los datos que necesite el brazo robótico para su funcionamiento.

Figura 4.2: Arquitectura del sistema

4.2.3. Diseño de la arquitectura del Software Paletizador

Hay que tener en cuenta que la arquitectura vista en la figura 4.2, se trata del sistema final del proyecto, esta memoria y así las practicas que he realizado en la empresa, solo se ha desarrollado el Software Paletizador, que es la primera imagen de la figura, y también los fichero XML que creará este programa, almacenados en un servidor.

La elaboración del software paletizador ha sido basada en un modelo conocido como Cliente/Servidor, es una de las arquitecturas que más se utilizan para este tipo de aplicaciones.

El objetivo principal de esta arquitectura es poder compartir los datos de una manera muy rápida y fiable, puede ser a través de la red, o incluyo en el mismo ordenador. Además a nuestro proyecto le concede una ventaja extra, ya que al tener dos aplicaciones que necesitan los mismo datos, se pueden compartir de una forma rápida entre ellas, y cualquier modificación que se realice en una, automáticamente se vera reflejada en la otra al reiniciar la aplicación.

Utilizando el modelo basado en la arquitectura Cliente/Servidor se puede conseguir un sistema en el que el control del mismo está totalmente centralizado, esto es debido a que los recursos y los accesos tienen que pasar por el servidor, y puedes permitir o denegar los mismos. Esto convierte nuestro sistema en un sistema muy seguro.

En la figura 4.3 se puede observar la arquitectura del software paletizador siguiendo el modelo anteriormente definido.

Figura 4.3: Arquitectura del Software Paletizador

4.3. Diseño de la interfaz

En el diseño de interfaces de usuario hay que tener en cuenta que tipos de usuarios van a utilizar la aplicación, y en base a ellos, se define la forma, la función, utilidad, ergonomía, y otros aspectos que afectan a la apariencia de la aplicación. En el caso de nuestro software, el diseño se centra en que cualquier persona sea capaz de utilizar nuestro software, ya que hay muchas clases de personas que lo utilizarán, desde el administrativo o encargado del diseño de formaciones, hasta el operario del brazo robótico o carretillero, es por esto por lo que requiere de una buena comprensión, debe ser fácil de usar, y tener las vistas claras para que los usuarios entiendan lo que están haciendo.

4.3.1. Guía de estilos

En este apartado definiremos la tipografía utilizada, y como cambiará su diseño, dependiendo del lugar donde esté y del tamaño que tenga. También se mostrará la paleta de colores utilizados en el software, y un ejemplo de formato de los botones. Todo debe quedar uniforme y claro.

- Título 3: Segoe UI 16 en negrita.

  Robottions

- Título 2: Segoe UI 18 en negrita.

  Robottions

- Título 1: Segoe UI 24 en negrita.

Cuerpo: Segoe UI 9 en negrita.

Esto es un ejemplo de cuerpo para ver como quedaría el texto en la aplicación, aunque realmente no tiene tanto texto, solo un par de líneas o tres como máximo. Este tipo de letra es la más común, ya que es la que viene predefinida en Visual Studio.

Colores

Color	Código	Uso	Paleta de colores
Black.	000000	Cuerpo y Títulos 1 y 2	Secundario.
Blue	4169E1	Botones principales	Primario.
CadetBlue	008080	Links.	Secundario.
LigthBlue	B0E0E6	Fondos en las tablas.	Terciario.
White	FFFFFF	Títulos en página de configuración	Terciario.
DarkSlateGray	778899	Títulos varios	Secundario.

Ejemplo uno de los botones principales de la aplicación

Letra Tipo Segoe 20 de color **Blue**, y borde con degradado **LightBlue y DarkSlateGray**

4.3.2. Prototipos

En una primera instancia, la empresa nos proporciona los prototipos de dos ventanas distintas de la aplicación de software que el cliente solicitó en primer lugar. Estos prototipos se muestran en las siguientes figuras 4.4 y 4.5.

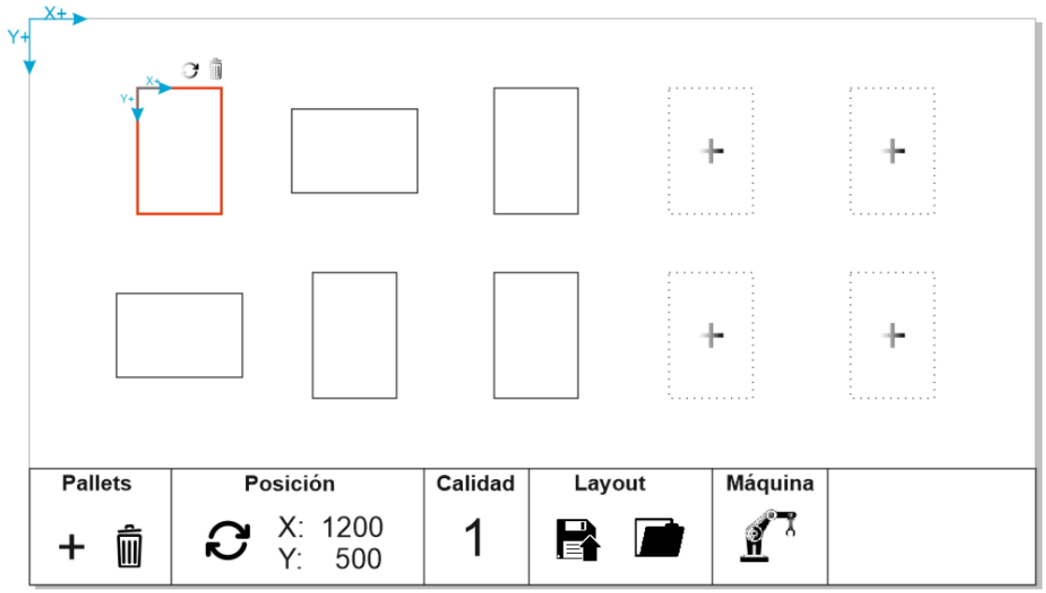

Figura 4.4: Prototipo 1

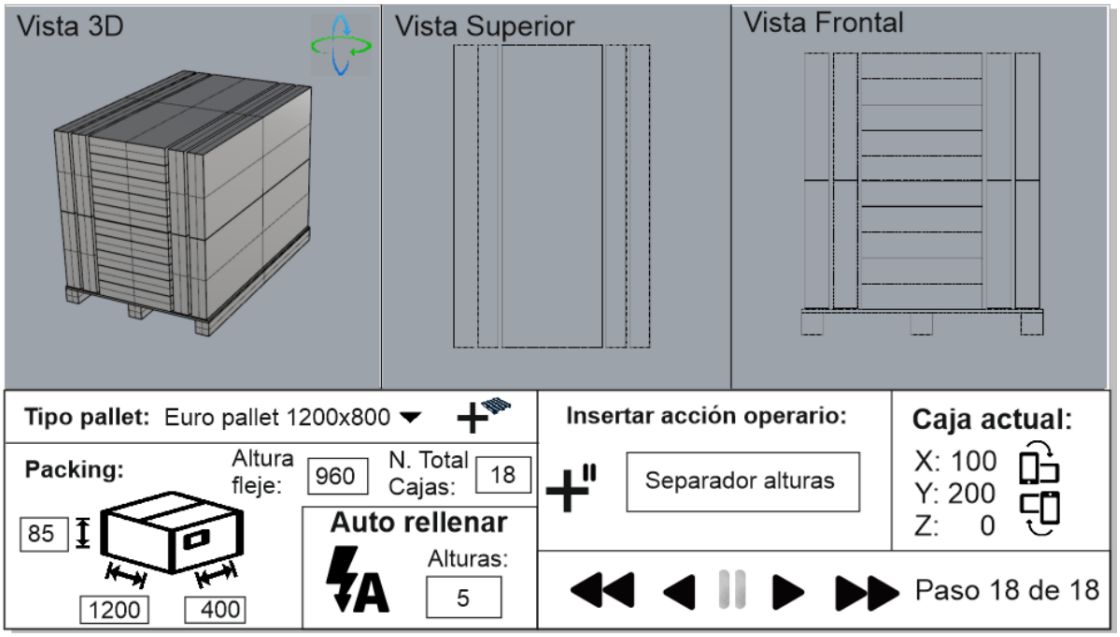

Figura 4.5: Prototipo 2

4.3.3. Interfaces

En este apartado se mostrarán capturas de las interfaces finales, el objetivo es mostrar como ha quedado la aplicación finalmente (a falta de mejoras) y que la usabilidad se ha tenido en cuenta. En el software, como ya se ha visto varias veces en este proyecto se buscaba que la aplicación fuera fácil e intuitiva para los usuarios que la utilicen, ya que puede tratarse de personas con experiencia, pero también otras que no tienen experiencia en aplicaciones de escritorio o informática en general.

Vista principal

Esta ventana o vista es la que se carga cuando se entra en la aplicación, en ella se puede ver de forma sencilla y clara los tres apartados donde, en los dos primeros botones podemos entrar para crear, editar o borrar una caja o un pallet, y el tercer botón para entrar en el menú de configuración de las formaciones, la parte más importante de la aplicación.

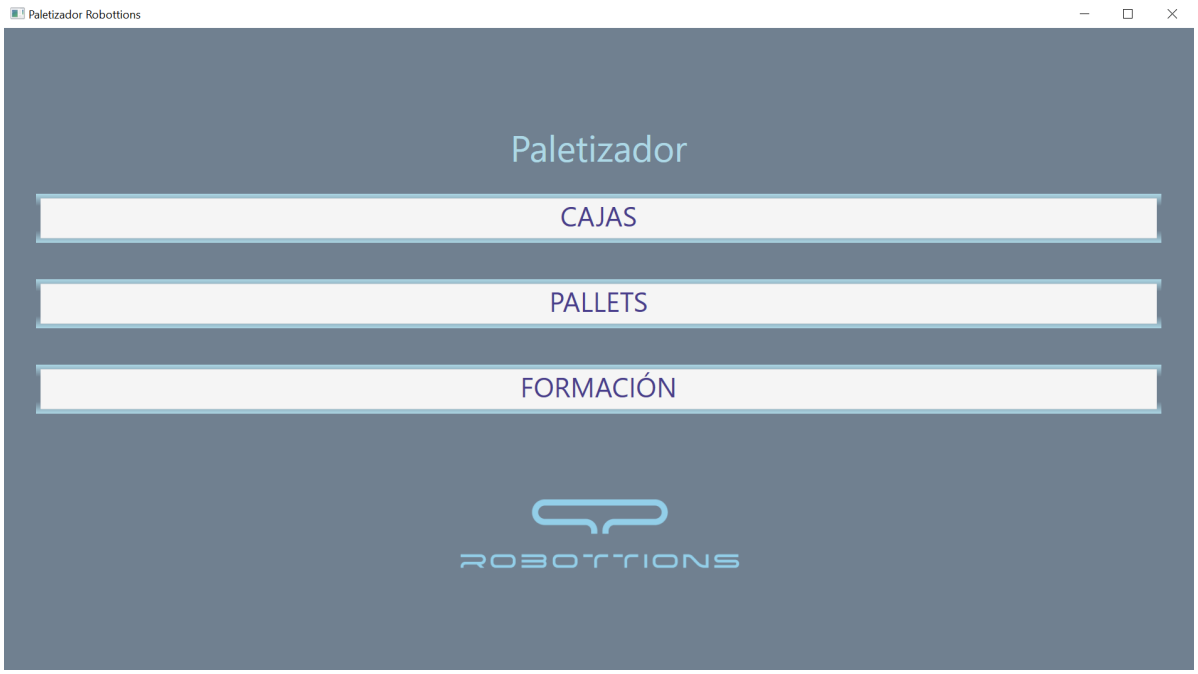

Figura 4.6: Vista principal

Vista de Configuración de Cajas y Pallets

A esta vista podemos acceder pulsando el botón de Cajas en la figura anterior. Utilizamos este panel para añadir una caja nueva, editar una existente o borrarla. Contiene un cuadro donde ver las cajas existentes (si mantienes el cursor encima de una de ellas podemos ver sus datos), un buscador de cajas en el cuadro anterior. Además cuenta con un formulario para rellenar en caso de querer crear o editar una caja, y un dos botones, uno para guardar y otro para eliminar la caja. En la segunda vista podemos ver la vista de Configuración de Pallets con los mismos objetos que la vista anterior.

Figura 4.7: Vista Configurar Cajas

Figura 4.8: Vista Configurar Pallets

Vista de Creación de Formaciones

A este panel se accede pulsando el botón de Formación en la figura 4.6. Como se observa en la figura 4.9 podemos dividir la vista en dos apartados diferenciables, el primero de ellos, es el primer rectángulo con el fondo blanco y el segundo el rectángulo inferior.

En el primer apartado podemos ver varios seleccionables, cada uno de ellos contiene un objeto del tipo indicado en etiqueta superior: Pasta, Formato, Plato y Cliente, Se debe seleccionar cada uno de ellos, además de un tipo de caja y de pallet para crear una nueva Formación. Esto es debido a que por petición del cliente, el nombre de la formación debe estar compuesto por el nombre de cada objeto anterior.

En esta primera vista también tenemos dos botones que son, uno para crear la formación después de seleccionar cada objeto y el otro para ver o editar una formación existente. Este último te muestra otra ventana donde puedes seleccionar la formación a editar, y también puedes borrarla.

En la segunda vista, tenemos 5 botones en la parte superior, cada uno de ellos te muestra un editor personalizado para cada tipo de objeto, donde se podrá crear un objeto nuevo, ver los existente o borrar alguno.

En las figura 4.10 se mostrará una vista con otro editor en la parte inferior, en este caso el de Clientes, y los seleccionables seleccionados. A su vez en la figura 4.11 se puede ver la pantalla perteneciente a Ver/Editar Formación.

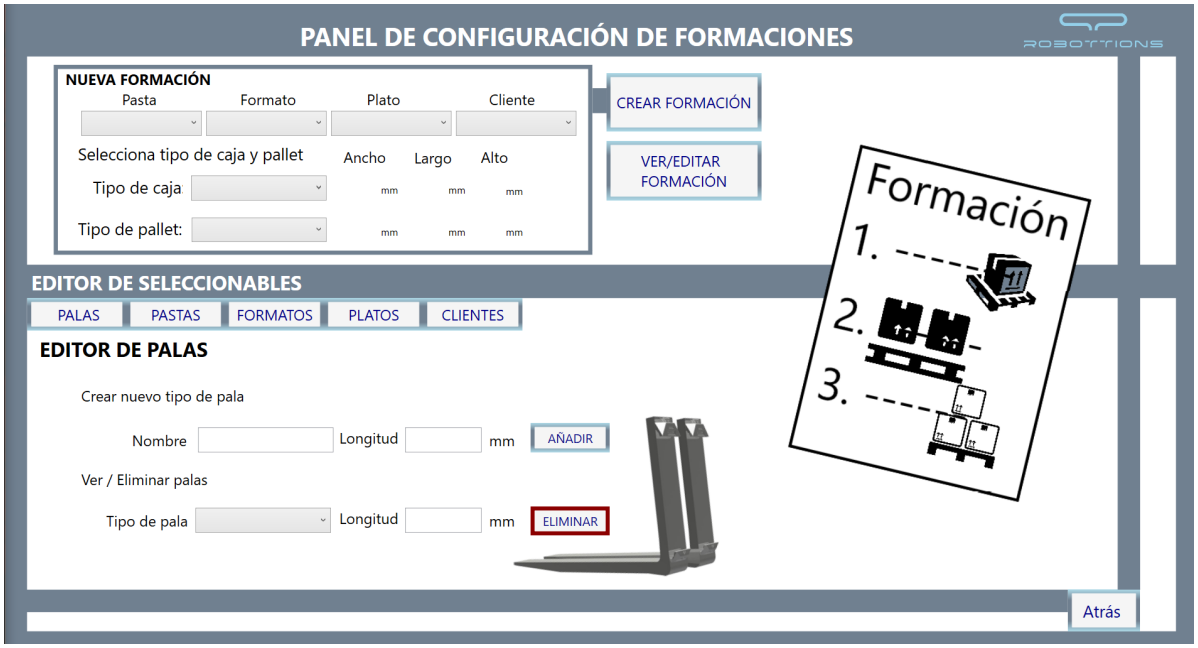

Figura 4.9: Vista de Creación de Formaciones

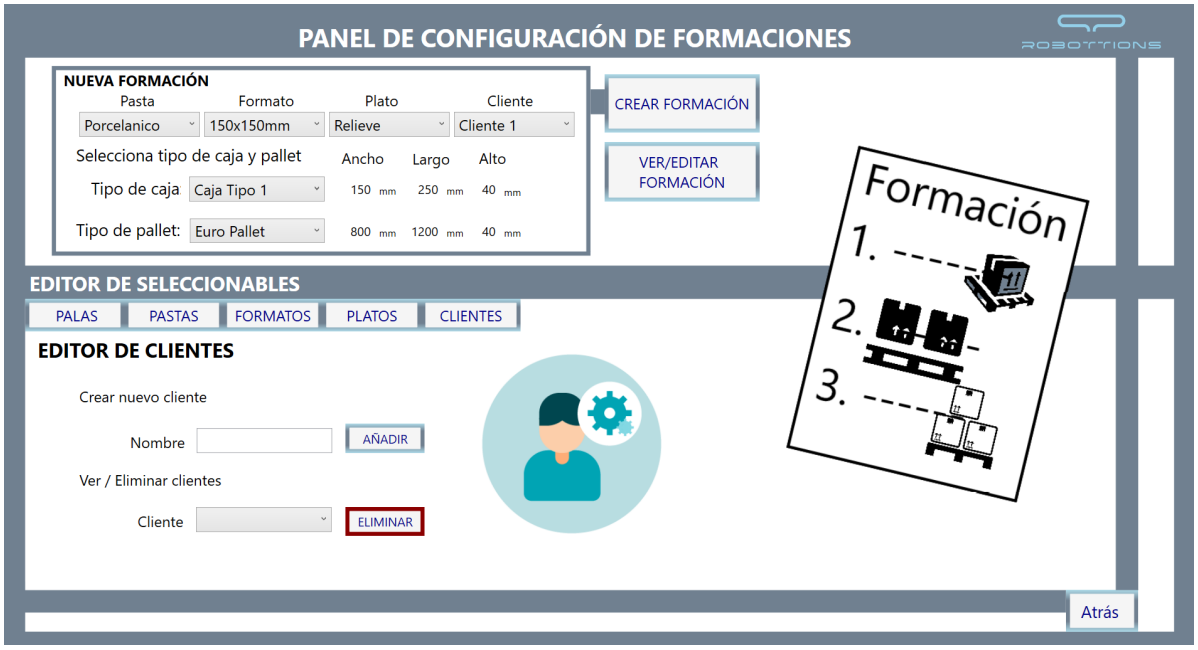

Figura 4.10: Vista de Edición de Formaciones

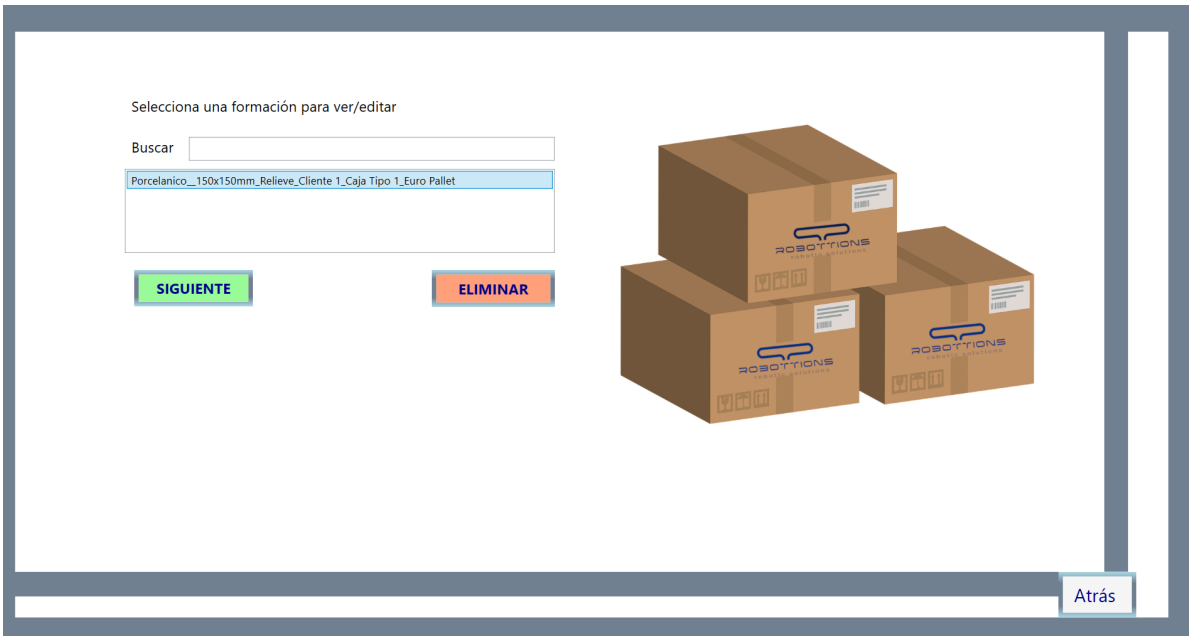

Figura 4.11: Vista de Creación Formaciones

Vista del Editor de Formaciones

A esta vista se acceder cuando se crea una Formación o bien cuando se Edita. En este panel se pueden observar varios botones y formularios.

En primer lugar, podemos ver un botón de **Editar Layout 1**, esto es para entrar en la pantalla más importante de nuestro programa, que es donde diseñaremos los layouts. También se puede observar un botón con el signo +, este agregará un segundo layout y una vez haya un segundo se podrá agregar un tercero. El siguiente botón **Modificar Caja o Pallet** se utilizará para cambiar el tipo de caja o pallet elegido en la creación de la formación. Tenemos dos botones más que es para salir de la vista, una guardando y la otra no.

Los formularios que están en la parte inferior de la vista, crecerá o disminuirán en función de las alturas, hay tantos como alturas, y sirven para introducir que tipo de layout habrá en cada altura. En el seleccionable con la etiqueta **Tipo de palas** se puede seleccionar la pala que se debe poner en la maquina elevadora, y **Pausa entre plantas**, se utiliza para saber si el robot debe parar al llenar toda una planta, o si por el contrario empezar directamente con la siguiente.

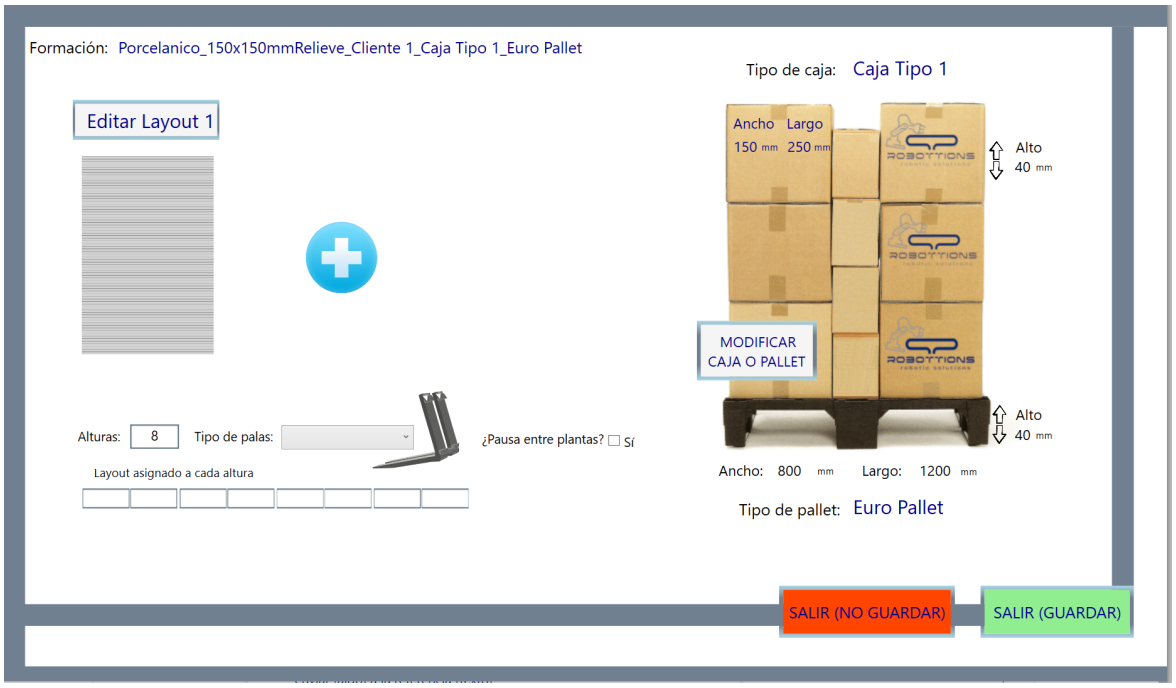

Figura 4.12: Vista de Edición de Formaciones

Layout

A esta vista se puede acceder a través del botón **Editar Layout** que se ha visto en la figura 4.12, tanto si es para crear un nuevo layout como para editar uno existente.

Esta interfaz es la parte crucial del software, en él, se añade caja por caja de la forma que luego recogerá el brazo robótico, se pueden ver 3 partes bien diferenciadas en el panel el primero con la etiqueta Editar Caja, el segundo etiquetado como Visualizador del Pallet y el tercer Con la etiqueta Nueva caja.

Empezando por la parte derecha de la figura 4.13**Nueva caja**, se puede observar un cuadro gris, al pulsar en el mismo aparecerá una nueva caja, en primer lugar, se gira la misma en la posición que se quiera mediante el botón de giro inferior, y a continuación hay dos formas de introducir la caja en el pallet, uno con el Modo Arrastar, el seleccionado en la figura 4.13, donde se se arrastra la caja cogiéndola de la esquina que se quiera solapar y se suelta en la esquina de una caja ya introducida en el pallet para que se solape a la misma. El modo Seleccionar esquina cumple la misma función pero sin seleccionar la caja, pulsando un botón de añadir caja.

Debajo del contenedor gris situado en la parte derecha de la figura 4.13, se puede ver un panel, con el botón **Mover caja** seleccionado, este es para mover una caja dentro del pallet, bien pulsando el botón de la flecha que se quiera o bien con las flechas del teclado, se moverá tantas posiciones como el numero en el interior de las flechas. En caso de seleccionar **Mover plano**, en lugar de moverse una sola caja, se moverán todas en un solo conjunto.

En la parte central de la figura 4.13, etiquetado como Visualizador del Pallet podemos ver un contenedor gris que contiene las cajas, esto hace referencia al pallet, en él, arrastraremos las nuevas cajas e iremos formando el diseño del layout para utilizarlo después. La etiqueta 0,0 y el borde rojo se pueden modificar para que quede en cualquiera de las esquinas, dependiendo de donde esté situado el robot. Abajo podemos ver un botón **Numerar cajas**, en el que al pulsarlo saldrá encima de cada caja el número que seguirá el robot a la hora de coger las cajas.

En la parte de la izquierda de la figura 4.13, se observa un conjunto de botones y textos, en este conjunto podemos editar una caja ya incluida en el pallet, girarla, moverla o eliminarla. También veremos la posición de cada esquina, denominada a,b,c o d, de la forma que se puede observar en el dibujo inferior.

En las siguientes figuras 4.13 y 4.14 se pueden observar los dos modos de añadir caja, en la primera con el Modo Arrastrar y en la segunda con el Modo Seleccionar Esquina. En la figura 4.12 se mostrará una caja seleccionada y los datos de la misma en el panel de Editar Caja.

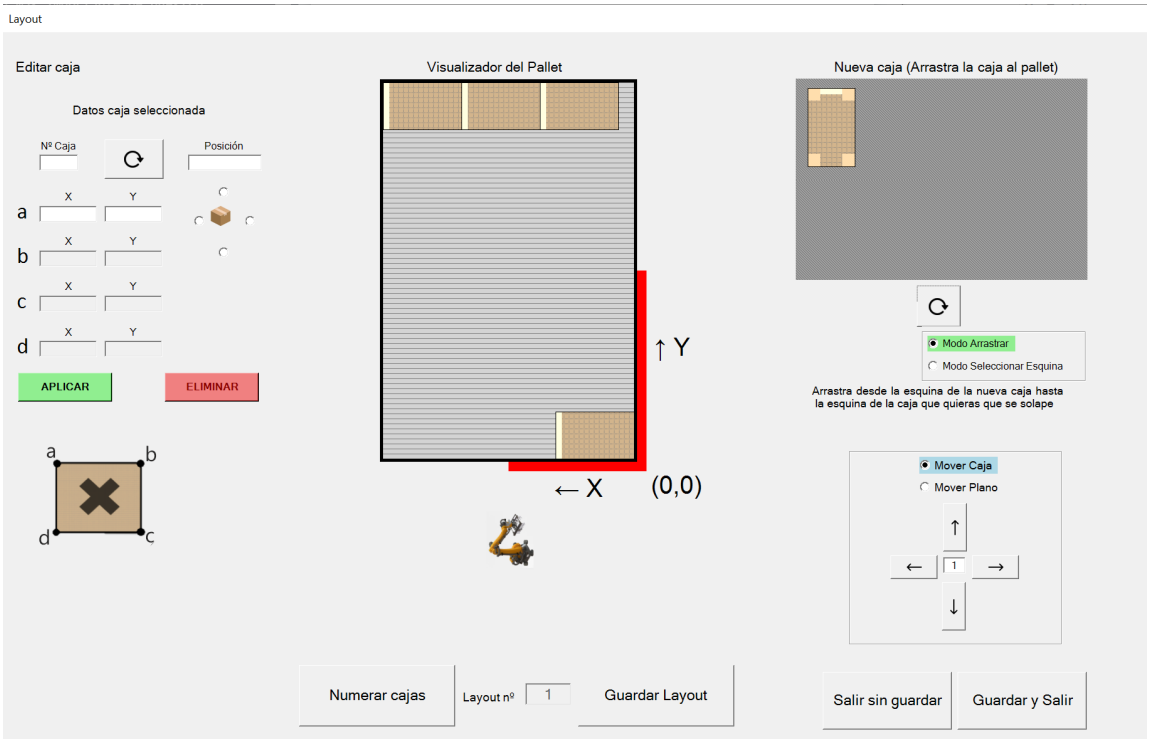

Figura 4.13: Vista de Edición de Formaciones

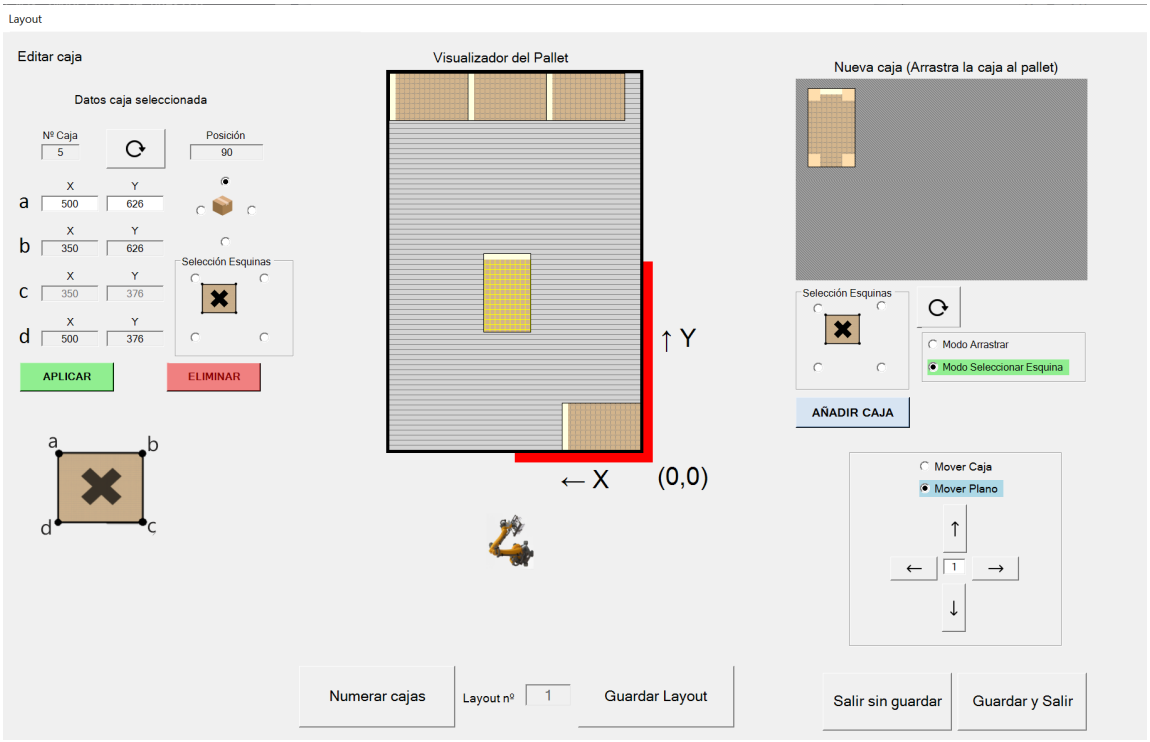

Figura 4.14: Vista de Edición de Formaciones 2

Capítulo 5

Implementación

Índice

En este apartado se desarrollará la parte de la implementación del proyecto, basado en las primeras pautas que me dío la empresa los primeros días de prácticas, estas pautas son las siguientes:

Arquitectura	Windows Forms Windows WPF
Lenguajes	C# xaml xml
Entorno de desarrollo	Visual Studio 2019
Visualizadores	Visualizador propido de Visual Studio Visualizador de Windows
Alojamiento Proyecto	GitHub

Tabla 5.1: Pautas para la implementación

En primer lugar, se ha realizó tanto la ventana funcional para la creación o edición de la caja, como la de la creación o edición del pallet. Después de crear las vistas anteriores se creó la ventana de Creación de Formaciones.

Mas adelantes se creo la vista del Layout. Para esta vista se ha incluido en el proyecto de Visual Studio, la interfaz de programación Windows Forms. En esta ventana el usuario debe ser capaz de diseñar el formato que tendrá una formación, rellenando uno de los pallets creados anteriormente y colocando cada una de las cajas en la posición que desee tanto en horizontal

como en vertical. Para esta vista se está utilizando la técnica de Drag and Drop, que consiste en arrastrar y soltar cada una de las cajas en la posición deseada con el ratón.

Por último y por petición del cliente, se han diseñado los dos Modos de introducción de la caja en el pallet (Arrastrar y Seleccionar Esquina) y los diferentes seleccionables para crear el nombre de las Formaciones.

5.1. Decisiones de la implementación

A lo largo de proyecto se han tomado varias decisiones, algunas tomadas por mí, para que una parte de la aplicación estuviera mas homogeneizada, otras tomadas por la empresa, y otras por parte del cliente, que es lo más común.

En una primera versión de la aplicación se entregó por primera vez al gerente de la empresa, para dar su visto bueno, esté sugirió una serie de cambios que son los siguientes:

- Cuando se actualicen los datos de una caja, aparezca un mensaje: "Se van a sobrescribir los datos de la caja X", el usuario tendrá que aceptar o no. De la misma forma con los pallets.
- Un recuadro superior en la lista de caja creadas, para buscar una caja por nombre. También lo mismo con la lista de los pallets.
- Que al cambiar la etiqueta en una caja, no se cambie sino que la caja gire hacia la dirección de la etiqueta seleccionada.
- Informaciones en varios sitios de la aplicación para que el usuario sepa que está haciendo.
- Algunos botones para poder salir de las vistas sin que se guarden los cambios.

Una vez corregidos esos cambios, y actualizada la aplicación, se paso a generar una segunda versión del programa. Se le proporcionó a la empresa esta segunda versión y la empresa dio su visto bueno, por lo que se procedió a llevar la aplicación al cliente. Cuando el cliente vio la aplicación propuso unos cambios a realizar y el arreglo de algunos pequeños errores que no vimos en su momento.

5.2. Control de errores

Para controlar los errores se han utilizado varios métodos, algunos más comunes como en los formularios y otros propios para el buen funcionamiento de la aplicación.

En primer lugar, veremos el control que se lleva en los formularios, donde no se podrán dejar vacías las casillas en el caso de introducir cualquier tamaño, y en el caso de los nombres, solo se podrán utilizar, números, letras o espacios, estos controles de error son para cada vez que se cree un objeto nuevo en la aplicación, ya sea una caja, un pallet, una pala, cliente o cualquier otro objeto. Podemos ver en la figura 5.1, el error producido al intentar poner un símbolo al crear una caja.

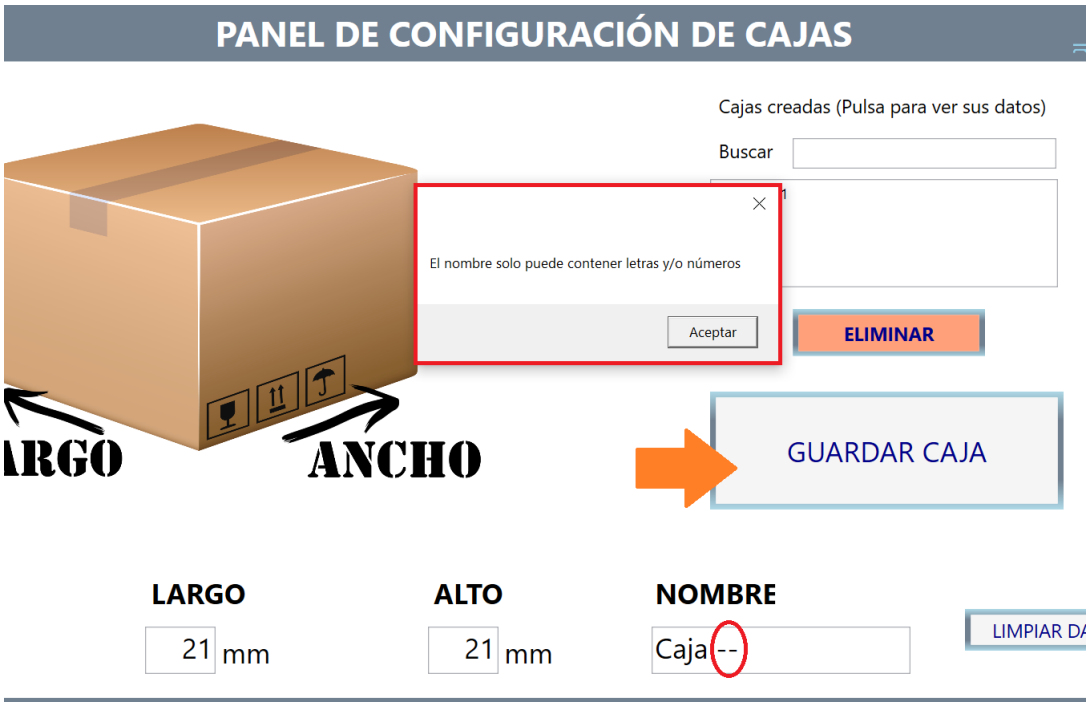

Figura 5.1: Control error en Crear Caja

Otro control de error muy importante es no poder crear ningún objeto con el mismo nombre, para ello se debe comprobar antes de crear cualquier objeto, principalmente las formaciones, si ya existe o no. En caso de existir, mostrará al usuario que ese objeto no se puede crear.

El siguiente control de error se crea al eliminar cualquier caja o pallet, como se ha comentado con anterioridad, la creación de una formación debe contener obligatoriamente un tipo de caja y pallet. Pero, ¿que sucede si se elimina la caja, y esta ya se ha incluido en alguna formación? Pues en en primer lugar, se decidió que no se podía eliminar, pero por petición del cliente, al final se concreto que se podía borrar pero siempre avisando al usuario de que va a borrar una caja utilizada en una formación. Hay que tener en cuenta que aún después de eliminarse la caja, en la formación que la contenía, seguirá estando. Veamos en la figura 5.2, el aviso que se muestra al eliminar una caja que se está utilizando.

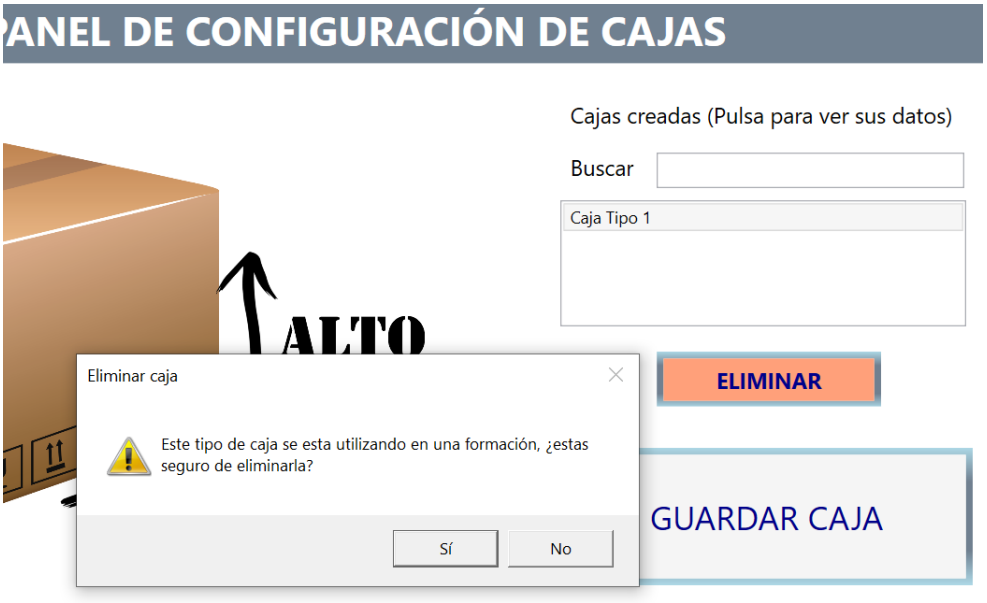

Figura 5.2: Control error en Eliminar Caja

Un error muy común en esta aplicación se produce al añadir cajas en el contenedor del pallet, y es que al poder mover las cajas mediante las flechas del teclado, arrastrándola con el ratón o directamente editando su posición, suele quedar una caja encima de otra, por lo que se produce un solapamiento, y es muy peligroso para el brazo robótico ya que puede intentar poner una caja en un lugar donde ya hay una caja y puede estropear el producto. Para ello, se ha creado un control de error, que se produce al guardar un layout. Al pulsar el botón de guardar, el sistema comprueba todas las cajas para ver si hay alguna encima o debajo de otra, y avisa mediante un popup, si en el mensaje pulsamos que no queremos guardar, el borde las cajas solapadas se teñirá de rojo, para que el usuario sepa que cajas estas solapadas. En la figura 5.3 se mostrara el mensaje y las cajas teñidas después del mismo.

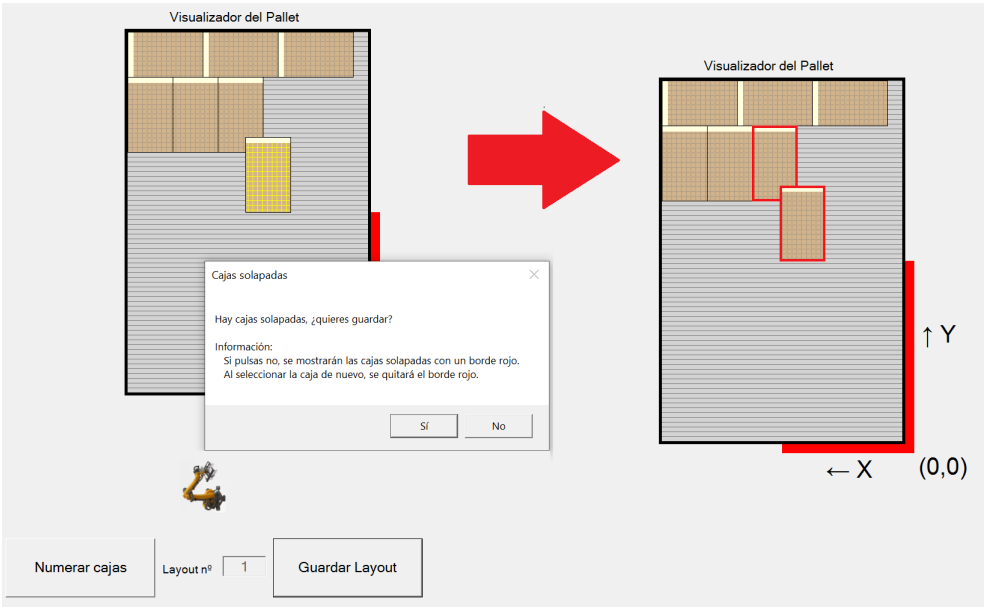

Figura 5.3: Control error en Layout

El último error del que se hablará, se produce al cambiar un tipo de caja o pallet una vez se ha creado la formación, y es que, nos dimos cuenta que en la vista preliminar del layout en el panel de Crear/Editar Formación, al cambiar una caja o un pallet, no cambiaba esta vista. Esto es debido a que la vista preliminar se trata de una imagen en formato bmp que el sistema fotografía al guardar un layout, por lo que no hay forma de realizar esa imagen, si no se ha entrado aun en el mismo. Esto lo solucionamos añadiendo un control de error que nos indicará encima de las vistas preliminares de los layouts, que esa imagen no es válida. Podemos verla en la figura 5.4.

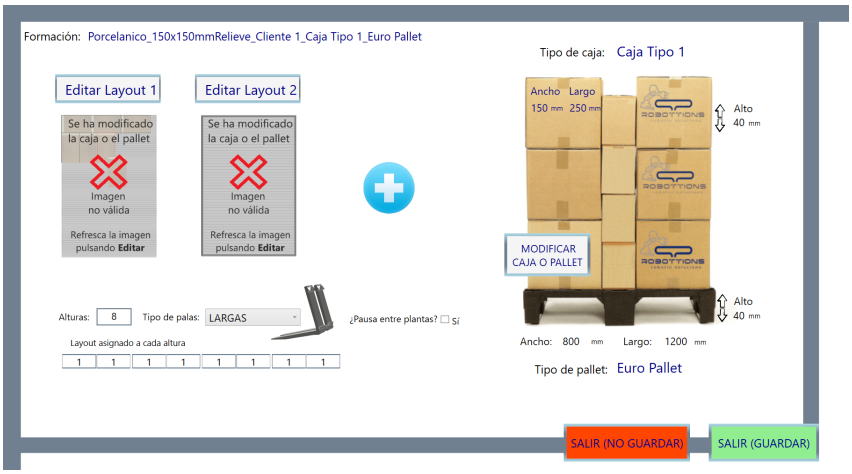

Figura 5.4: Control error en Layout

5.3. Listado de paquete y clases

En este apartado mostraremos los diferentes paquetes y clases utilizados en la implementación del proyecto, con una pequeña descripción de para que se ha creado o sirve cada una.

Para la mayoría de clases se han utilizado dos interfaces de programación para aplicaciones gráficas basadas en Windows como se ha ido viendo durante la memoria, la primera de ellas ha sido Windows WPF, y la segunda Windows Forms. Las dos interfaces de programación son muy parecidas, siendo Forms más antigua y con una diferencia notable para las pantallas de hoy en día, y es que Windows WPF automáticamente redimensiona todos sus objetos de forma que se muestre bien con las diferentes resoluciones en el mercado, HD, Full HD, 2k o 4k, mientras que en Forms puedes crear una resolución fija y no se puede redimensionar, aunque hay técnicas para ello, pero son bastante tediosas.

5.3.1. Paquetes

Como se puede observar en la figura 5.5, la aplicación está dividida en 6 paquetes, incluyendo cada uno partes bien diferenciadas de programa.

El primero de ellos denominado Almacenamiento, se compone de las clases utilizadas para el almacenamiento tanto interno del programa como externo (la creación o carga de XML). Contiene tres clases que son: BBDD, Datos y Serializar XML.

El segundo paquete, Imágenes y el último, llamado Resources, se utilizan para guardar los recursos del programa, imágenes, iconos o archivos png para incluir en algunos botones, como el de girar en el Layout, o el botón deshacer.

El tercer paquete que podemos observar en la figura 5.5, con el nombre de Layout, contiene todas las clases de Windows Forms utilizadas para el panel de edición del Layout, este panel es el que se puede ver en la figura 4.13 y 4.14, Se pueden ver 4 clases, los dos primeros son los contenedores de Caja y Pallet, el tercero en el panel principal, y el cuarto de ellos, es para cambiar el tipo de caja o pallet una vez escogidos.

Como cuarto paquete se puede contemplar NombreFormación, este se utiliza para los objetos que el cliente nos demando después de enseñárselo por primera vez, son los objetos de los seleccionables que componen el nombre de la formación, formado por Clientes, Formato, Pasta y Platos (Además del tipo de caja y pallet). Se pueden ver los seleccionables nombrados en la figura 4.9 vista en el capítulo anterior.

A continuación se puede observar la figura 5.5 donde se ven los paquetes con las clases que se han nombrado anteriormente.

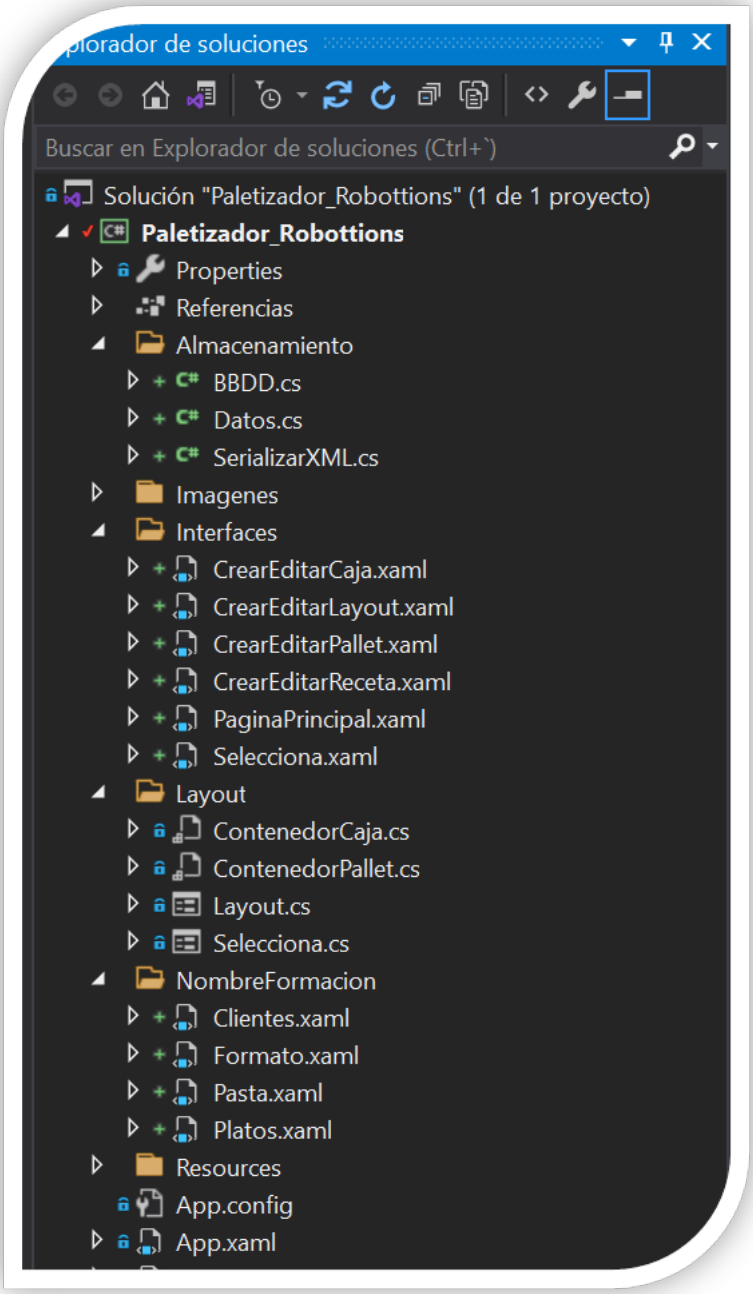

Figura 5.5: Lista de paquetes

5.3.2. Clases

En este apartado mostraremos las clases que incluyen cada uno de los paquetes vistos en el subapartado anterior y una pequeña descripción de cada uno. Exceptuando los paquetes Imágenes y Resources que no contienen ninguna clase.

En primer lugar y como se puede observar en la figura 5.6, tenemos las clases incluidas en Almacenamiento, BBDD, Datos y Serializar XML.

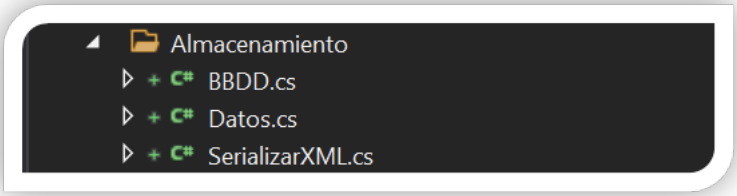

Figura 5.6: Clases del paquete Almacenamiento

BBDD

La clase BBDD se utiliza, como sugiere su nombre, para guardar todos los datos internos del programa en un objeto de la clase Datos para después almacenarlos en el disco duro o servidor, mediante la creación de los ficheros XML. La técnica que utilizamos es, recoger toda la información del programa, y convertirlo en formato String, con este String enviamos los datos a la clase SerializarXML, que será la que se encargará de cifrar el XML, y una vez cifrado se almacenará. En el caso de la carga de datos, la clase BBDD enviara los ficheros a la clase anterior, que descifrará los mismos y se los enviará otra vez convirtiendo de nuevo de formato String al formato de cada objeto de la aplicación.

Datos

La clase Datos se utiliza para almacenamiento interno de cada objeto creado en el programa y sus respectivas listas de objetos. Es la clase más importante de la aplicación, ya que contiene la definición de todas las clases del software. Es a la clase que recurrimos cada vez que queremos crear cualquier objeto.

SerializarXML

Esta clase es la encargada de codificar el código que le enviamos desde la clase BBDD, y devolvérselo listo para guardar en el lugar que corresponda, disco local o servidor. El otro uso que tiene es el contrario, para convertir el lenguaje XML en los objetos que corresponda de nuestro programa, lo realiza de la misma forma que lo codifica.

Una vez visto las clases pertenecientes al paquete de almacenamiento, se mostrarán las clases del paquete de las interfaces.

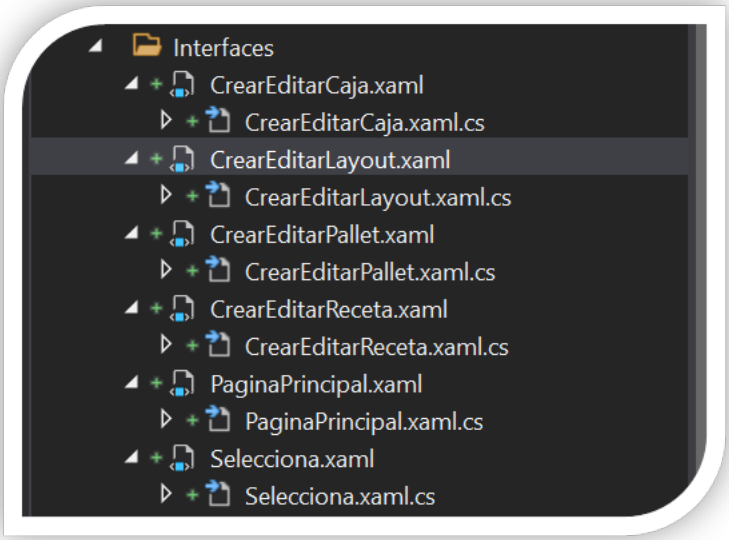

Figura 5.7: Clases del paquete Interfaces

CrearEditarCaja

La Interfaz CrearEditarCaja, es la que el usuario utilizará para la creación de nuevo tipo de caja, deberá introducir, alto, largo y ancho, además de un nombre. Por otro lado, esta interfaz también permitirá al usuario editar los datos de las cajas existentes o eliminar cualquiera de ellas.

CrearEditarPallet

La Interfaz CrearEditarPallet, es muy parecida a la clase anterior, y es que esta es la que el usuario utilizará para la creación de un nuevo tipo de pallet, como en la anterior, deberá introducir, alto, largo y ancho, además de un nombre. Por otro lado, esta interfaz también permitirá al usuario editar los datos de los pallets existentes o eliminar cualquiera de ellos.

CrearEditarReceta

Esta interfaz, es en la que se creará la nueva receta, podemos verla en la figura 4.10 del capítulo anterior, para la creación de la receta, se dispondrá de 4 seleccionables, y otros dos seleccionables más en la parte inferior, el nombre de la nueva receta, sera la unión de los nombres que seleccionemos, separados por una barra baja.

Además de crear una nueva receta, esta interfaz nos servirá para añadir, editar o eliminar cualquier objeto de los seleccionables anteriores, y es que para ello, tenemos en la parte inferior 5 botones, cada uno de ellos nos mostrará un panel distinto de cada objeto para realizar esta función. Por último, también disponemos de otro botón que nos enviará a la interfaz de edición y eliminación de las recetas.

PaginaPrincipal

La interfaz PaginaPrincipal no es más que la interfaz que nos encontramos al abrir el programa por primera vez, donde se pueden ver cuatro botones, que son, para entrar en el editor de cajas, otro para el editor de pallet, el tercero entra en la interfaz CrearEditarReceta anterior, y el último para salir de aplicación.

Selecciona

La última que nos encontramos es la de Selecciona, y se trata de la interfaz de usuario que hemos nombrado anteriormente, a ella accederemos desde un botón en la interfaz de CrearEditarReceta, y es para la edición de las formaciones o recetas, al acceder a esta, nos encontramos con una panel donde se encuentran todas las recetas existentes, seleccionando una de ellas, podremos entrar en la edición de la misma, pulsando siguiente, o eliminar la formación/receta para siempre. También cuenta con un buscador de formaciones para facilitar la tarea de encontrar la que se desee.

A continuación, se mostrarán las interfaces o clases pertenecientes al paquete Layout. Todas estas clases están realizadas con Windows Forms, por eso se distinguen del paquete anterior.

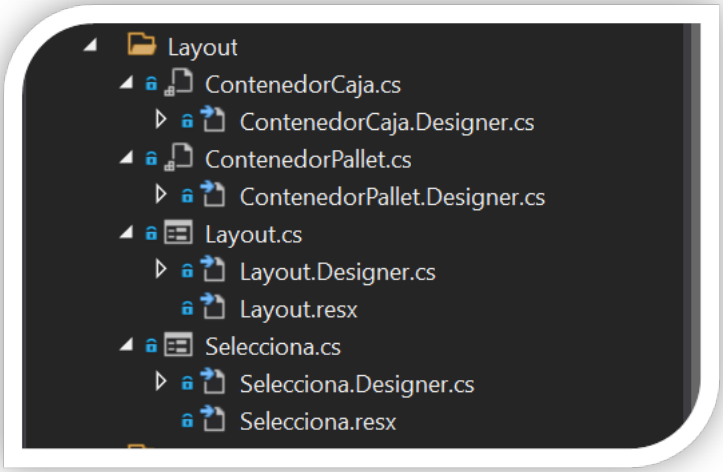

Figura 5.8: Clases del paquete Layout

Aunque se pueden observar cuatro implementaciones de interfaces basadas en Windows Forms, en realidad visualmente solo es una, esto es debido a que la interfaz principal es la del Layout, se puede ver la figura en el capitulo anterior, concretamente la figura 4.13 y 4.14, las interfaces ContenedorCaja y ContenedorPallet están incluidas en el layout, que son los contenedores que simulan la nueva caja o el llenado del pallet. Por otro lado, la interfaz selecciona es utilizada para cambiar la caja, el pallet, o ambos.

ContenedorCaja

La interfaz de ContenedorCaja, como ya hemos comentado, es el contenedor que nos encontramos a la derecha de la interfaz de usuario Layout, este elemento es el encargado de crear una nueva caja, girarla como se desee y después arrastrarla al contenedor de pallet, a la posición que se desee. Podemos ver en la figura 5.9 de que se trata.

ContenedorPallet

La interfaz de ContenedorPallet, como también hemos comentado anteriormente, es el contenedor que nos encontramos en la parte central de la interfaz de usuario Layout, este elemento es el encargado de recoger las cajas que se añaden desde el contenedor anterior. También se pueden editar las cajas una vez puestas, bien con las teclas de dirección del teclado, o bien pulsando en el layout los botones de las flechas. Además es posible girar las cajas una vez puestas, y mover la posición manualmente. En la figura 5.9 mostramos este contenedor y el anterior para su facilidad de entendimiento.

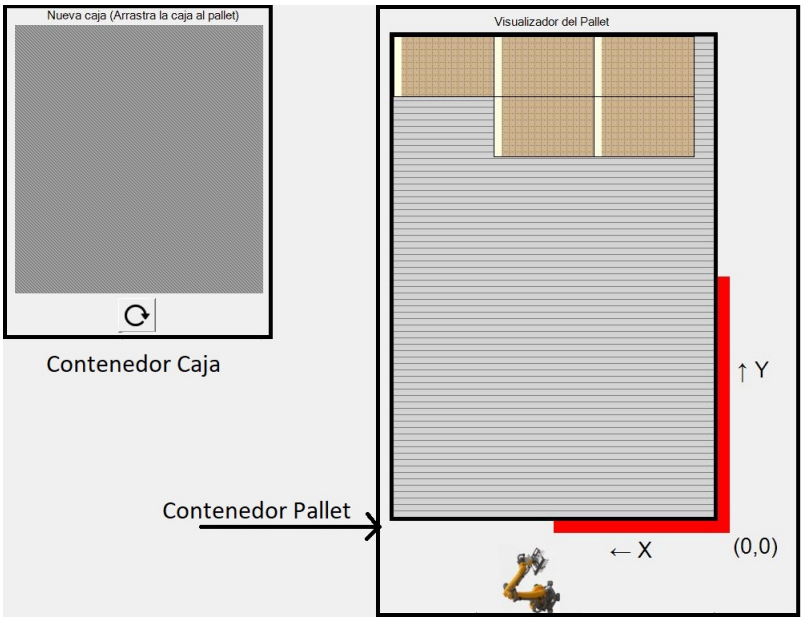

Figura 5.9: Contenedores: Nueva caja y Pallet

Layout

En la interfaz Layout, es donde prácticamente se realizan las tareas más importantes de la aplicación. Al iniciar la interfaz de usuario, nos encontramos con el pallet del tamaño que se haya elegido en forma de contenedor de objetos donde podemos añadir las nuevas cajas. Al lado derecho, y después de pulsar el contenedor de la nueva caja, aparecerá visualmente una caja, la cual podremos arrastrar a la posición deseada dentro del pallet. En el layout se realizan todas las tareas de edición del ContenedorPallet hasta que el pallet quede como lo hayamos imaginado para después el brazo robótico sepa donde situar cada una de las cajas.

Además de lo anterior, también tenemos la posibilidad de mover todo el plano de una vez, para que no queden espacios en el pallet. Por otro lado tenemos un botón de numerar las cajas, esta numeración es el orden que seguirá el robot para colocar las cajas. Y siempre será de la misma forma, la primera caja será la de la esquina izquierda del lado contrario al robot (se puede ver por el dibujo del brazo robótico), luego ira añadiendo hacia la derecha, hasta llegar al final de la linea, y empezará de nuevo con la caja más a la izquierda de la siguiente linea.

Por último, nos encontramos con tres botones inferiores, dos para guardar el Layout, uno saliendo de la interfaz y el otro sin salir, y otro botón para salir sin guardar los cambios. En los botones de guardado, es donde están creados los controles de error de solapamientos de cajas visto en el apartado anterior.

Selecciona

Es la interfaz gráfica que se utiliza para seleccionar desde CrearEditarLayout un cambio de tipo de caja o pallet en la receta ya existente, las cajas ya agregadas en el pallet automáticamente cambiarán de tamaño y se ajustarán al nuevo elegido. De igual forma pasará con el pallet. Por lo tanto es muy común que todo quede mal y haya muchos solapamientos de cajas o cajas fuera del pallet. Por esto es mejor crear un layout desde 0, que utilizar este método.

Una vez vistas las clases del paquete Layout, se verán las clases del paquete NombreFormación, que son las mostradas en la figura 5.10.

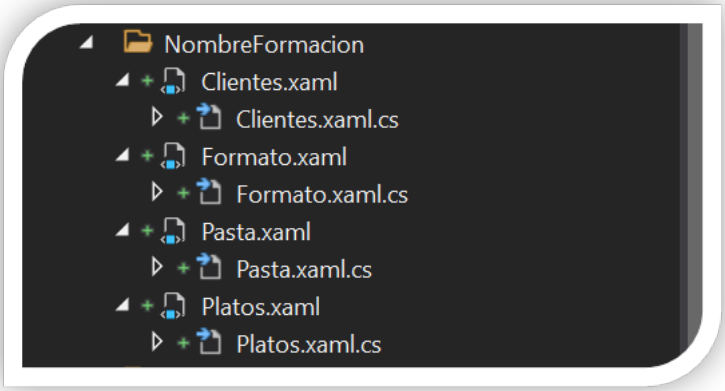

Figura 5.10: Clases del paquete NombreFormacion

En estas cuatro clases no es necesario que las veamos una por una, ya que el funcionamiento es exactamente el mismo, se tratan de las clases:

- Clientes. Los clientes de la empresa.
- Formato. Es el formato de las piezas, no de las cajas.
- Pasta. El material del cual están hechas las piezas de cerámica (porcelana, gres).
- Plato. Es la rugosidad de la pieza (lisa, rugosa).

Estas clases se utilizan para crear el panel de edición de cada una de ellas, como podemos ver en la figura 5.11, cada panel se compone de dos editores, uno para crear un nuevo elemento y el otro para visualizarlos o eliminarlos. Aunque en la figura 5.11 solo se muestra el panel del cliente, los otros tres paneles se trata de lo mismo, solo cambiando la imagen y los nombres. El panel de más a la izquierda, Palas, no tiene que ver con estos, se trata de otro editor para las palas de la máquina elevadora.

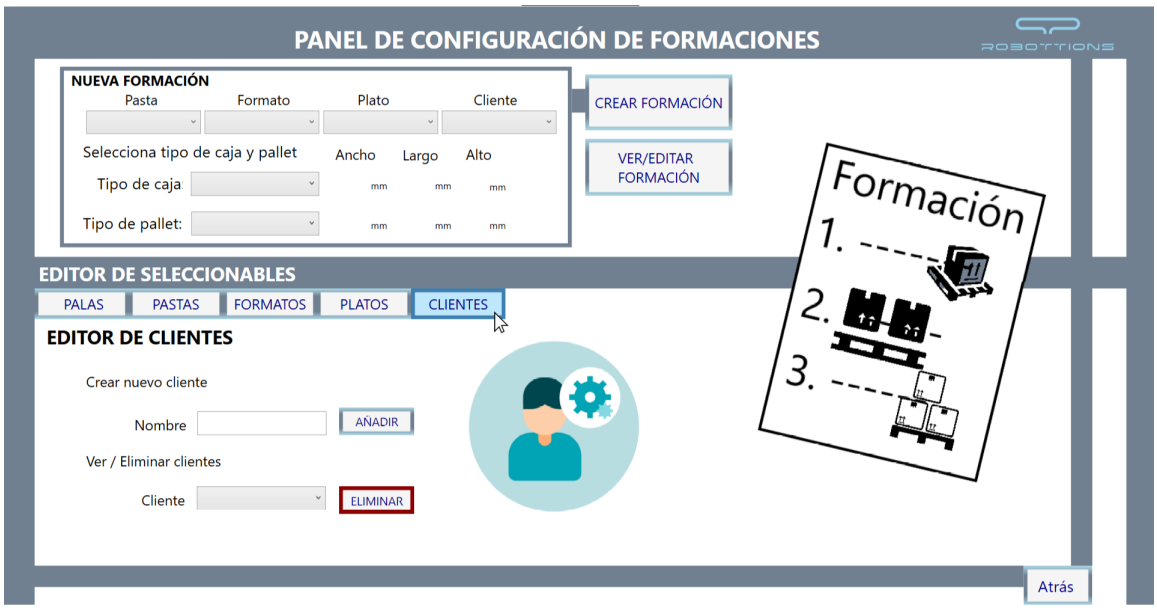

Figura 5.11: Editor del cliente

Capítulo 6

Pruebas realizadas

Índice

En este capitulo observaremos de forma detallada las pruebas realizadas en la aplicación, mientras se estaba desarrollando (verificación) y una vez finalizada (validación), además de las disconformidades con los clientes y otros errores.

6.1. Verificación y validación

Cuando se habla de verificación, nos referimos a la comprobación de la aplicación mientras que todavía se esta desarrollando el proyecto, para ello antes de cada segmento concreto del proyecto, realizábamos unos requisitos o especificaciones que venían dados por parte de la empresa y se comprobaba constantemente que, lo que estamos realizando se adhiere a estos requisitos.

Estas comprobaciones por regla general se realizaban a primera hora de la mañana sobre el trabajo realizado el día anterior, a no ser que tuviéramos que presentar algo ese día. Normalmente estas verificaciones eran algo tan fácil como leer los requisitos y compararlos con lógica del código que hemos implementado. Se comprobaba el código, los controles de errores, el formato de la aplicación, errores gramaticales y el diseño de lo que hemos realizado para ver que concuerde y tenga homogeneidad con el resto del código.

Gracias a la realización de la verificación se ahorra mucho tiempo y costos del proyecto en general, razonar esto es muy fácil, ya que es mucho mas sencillo corregir un error que se ha visto en una fase temprana del proyecto que más adelante, ya que se arrastra el error y se construye encima del mismo, al finalizar, todo lo construido esta mal, por otro lado si se comprueba al final se debe buscar en cientos de lineas, mientras que si se comprueba solo lo escrito el día de antes, es mucho más sencillo encontrar el mismo problema.

Por poner un ejemplo real en la aplicación, del problema anterior, cuando se intento crear el algoritmo de solapamiento de las cajas, no encontraba la forma de hacerlo, y aunque funcionaba no del todo bien, decidí seguir con el proyecto y realizar el algoritmo más adelante

para no perder tiempo. Al final del proyecto, se consiguió crear bien el algoritmo, basándonos en la implementación de arrastrar por esquinas, pero costo mucho más de lo necesario, ya que habíamos construido todo el proyecto con el algoritmo erróneo.

La validación por otro lado se lleva a cabo cuando la creación del software ha llegado al final, o se ha creado una versión definitiva. Cuando se está validando la aplicación, significa que ya hemos finalizado, da igual como, pero lo importante es que hemos llegado y todo funciona como se esperaba. Normalmente se realizan una serie de preguntas, que son:

- ¿La aplicación funciona?
- ¿Puedo crear cajas, pallets y formaciones?
- ¿Se pueden crear diseños de formaciones sin errores?
- ¿Se guardan todos los datos y después se pueden leer?
- ¿El rendimiento de la aplicación es bueno?

Si todo es correcto, es que se cumplen los criterios esperados y que los requisitos están incluidos en la aplicación. Una manera de comprobar que esta validación es correcta, es entregando al cliente la aplicación y que de su visto bueno.

6.2. Disconformidades con los clientes

Durante el desarrollo de nuestra aplicación se ha entregado varias veces el proyecto finalizado en sus primeras versiones a los clientes, para que dieran su visto bueno, en estas entregas los clientes han manifestado una serie de cambios a realizar en la aplicación así como el arreglo de algunos pequeños errores que no vimos en su momento. Parte de esos cambios a realizar por el cliente han sido:

- Algunos atajos de teclado para borrar o editar cajas en la parte del Layout.
- Poner números arriba de cada caja, para saber el orden en que el brazo las cogerá.
- Cuando colocas una nueva caja se seleccione automáticamente.
- Cambiar el nombre de Receta por Formación.
- No guardar la formación si no se han seleccionado palas.
- Avisar si se elimina una caja que se esta utilizando en alguna formación.

Estos cambios implicaron la realización y eliminación de parte del código o diseño, para crearlo de otra forma, o para añadir algunas funciones nuevas. Por esta razón el tiempo planificado en primer momento no fue el que se tardó realmente. Pero gracias a ello, se le otorga a la apelación una garantía de calidad.

Capítulo 7

Futuro del proyecto

Índice

En este capítulo se desarrollarán las mejoras futuras que se realizarán en la aplicación, así como el futuro del proyecto total.

7.1. Mejoras futuras

Durante el proyecto, se han dejado por hacer algunas mejoras por falta de tiempo y también se han visto algunas funcionalidades que serían bastante interesante que tuviera la aplicación. A continuación hablaremos de las más interesantes.

En primer lugar, una de las mejoras de la aplicación, sería añadir un panel de configuración al que se accederá con una contraseña y donde el usuario con acceso a él, pueda editar cualquier apartado de la formación, para así cambiar la pasta, el plato, cliente o formato de la formación, además de otras funcionalidades que ahora no se pueden modificar.

Por otro lado, en el panel de creación del diseño de la formación, en un futuro se debería añadir una serie de funcionalidades sobre los layouts, la primera mejora sería poder copiar un Layout para continuar añadiendo o editando objetos en un diseño diferente que solo necesite unos pequeños cambios en el diseño del Layout anterior. Como segunda mejora, sería además de copiar el Layout, poder girar el plano completo 180 grados, esta mejora fue sugerida por parte de un cliente, ya que muchas veces los diferentes niveles o alturas del pallet son el mismo diseño pero dado la vuelta, para formar una estructura en la que las cajas no puedan caer ya que unas se sujetan con las otras.

Además de las mejoras anteriores, se pueden realizar muchas otras correspondiente al diseño de la aplicación, por ejemplo en los popups que saltan por control de errores, ya que son los predefinidos por WPF, también añadir iconos para que algunos apartados queden más claros y exclusivos de la empresa. Todas estas mejoras es muy probable que se acaben realizando, pero por el tiempo que se tiene que dedicar al resto del proyecto no se han podido realizar ahora.

7.2. Futuro del proyecto

Como ya se ha nombrado durante la realización de este documento, el proyecto total consta de otra aplicación más y de la programación de un dispositivo para el brazo robótico.

En primer lugar, hablaremos sobre la nueva aplicación, esta nueva app, denominada HMI se trata del programa que estará en contacto directo con el brazo robotico, este software será el enlace entre la aplicación de escritorio que hemos creado y el brazo robótico. El HMI se mostrara siempre en la pantalla los operarios del brazo robotico, los operarios de linea y los supervisores. Las características que debe tener son:

- Constará de una ventana que siempre debe estar activa con una vista en tiempo real de la información de cada uno de los pallets que se encargará el brazo robotico.
- Esta información será: Caja actual, Nivel actual, Calidad, Llenado, Activar/Desactivar.
- En la parte derecha debe contar con una serie de pestañas, que al pulsarlas mostraran una serie de opciones dependiendo de la ventana que hayamos pulsado.
- Al pulsar esta ventana debe aparecer en la parte derecha esa información a editar, pero teniendo en cuenta que la ventana principal con la vista de los pallets siempre debe estar activa.

En la figura 7.1 podemos ver un prototipo del HMI que conectara con el dispositivo del brazo robotico, con las características mencionadas anteriormente.

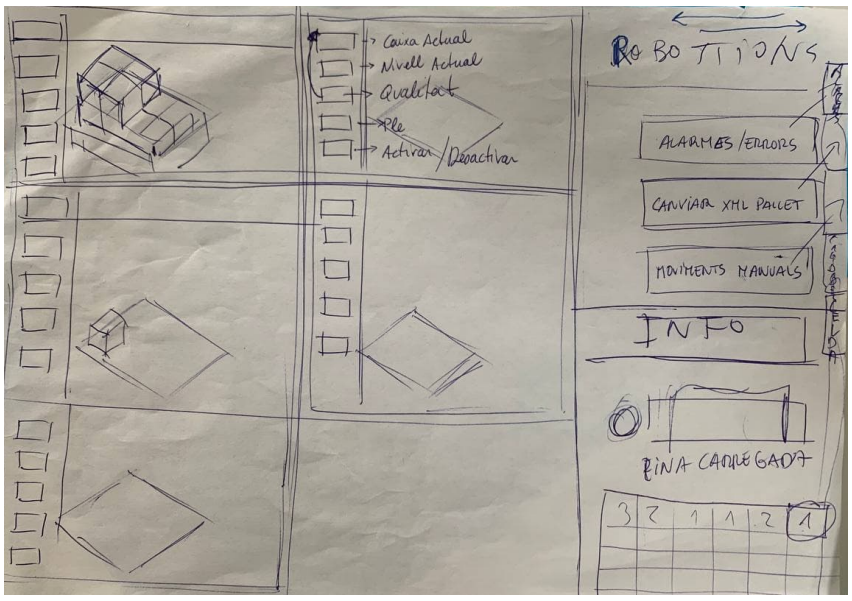

Figura 7.1: Primer boceto del HMI

Mientras se esta realizando esta memoria, se ha empezado con esta segunda aplicación, ya fuera de las prácticas y contratado por la empresa, en la figura 7.2 se mostrará lo que ya se ha realizado hasta ahora del HMI.

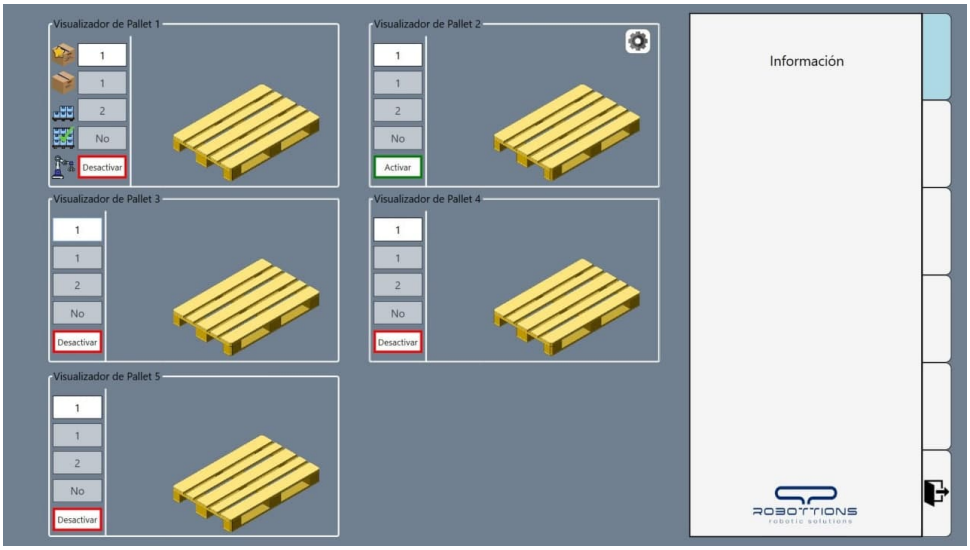

Figura 7.2: Primera versión del HMI

Por otro lado, después de la realización de esta segunda aplicación, también se realizará la programación del dispositivo que se comunicara con esta. Sobre esta parte del proyecto, que es mas centrada en el Hardware, todavía no se ha visto mucho.

Pero sabemos que deberá comunicarse con el brazo robótico mediante conexión inalámbrica, y se comunicará con una serie de sensores externos al robot para saber cuando hay un pallet para llenar, cuando se lo han llevado, o si una altura ya está llena para poder comunicárselo al HMI visto anteriormente.

Capítulo 8

Conclusiones

Índice

En este capıtulo se exponen los resultados del desarrollo del proyecto y las conclusiones finales, tanto técnicas como personales.

8.1. Resultados del proyecto

El resultado final del proyecto que se me ha asignado en las practicas de empresa, ha sido muy positivo. Se han cumplido todos los objetivos con creces hasta llegar al visto bueno final del cliente. Esto quiere decir que además de llegar a los objetivos de la empresa, también se han añadido todas las funcionalidades finales que el cliente ha pedido.

La puesta en marcha de la aplicación se ha realizado con éxito, solo faltando la comunicación con la aplicación siguiente, que se realizará mediante ficheros con formato XML, estos ficheros sí se han preparado en la elaboración del proyecto. Tampoco se ha podido realizar la comunicación con el robot finalmente, ya que esto aún no esta realizado.

Toda mi estancia de prácticas se ha realizado en la propia empresa, la que me ha proporcionado todo el material que he necesitado y ha creado para mí un puesto de trabajo propio donde he podido realizar el proyecto sin ningún problema.

Además durante mi estancia en Robottions donde he desarrollado las practicas, siempre ha habido un buen trato, han resuelto cualquier problema que me ha surgido al crear la aplicación y la colaboración con el supervisor fue constante y eficaz.

8.2. Conclusiones técnicas

Con la finalización del proyecto de prácticas, creo que el haber utilizado el lenguaje C# y el entorno de desarrollo Visual Studio ha sido una decisión muy acertada. Todo los problemas son fácilmente resolubles ya que hay mucha información sobre ellos.

Para la mayoría de clases se han utilizado dos interfaces de programación para aplicaciones gráficas basadas en Windows como se ha ido viendo durante la memoria, la primera de ellas ha sido Windows WPF, y la segunda Windows Forms. Las dos interfaces de programación son muy parecidas, siendo Forms más antigua y con algunos problemas de resolución, pero que se han solventado gracias a algunas técnicas utilizadas.

Decir que nunca había utilizado el entorno de desarrollo de Visual Studio, tampoco el lenguaje C#, ni las interfaces de programación Windows WPF y Forms. Aún así creo que no se podría haber realizado mejor con otros sistemas, ya que una vez he aprendido a utilizarlos, es muy sencillo trabajar con los mismos y el producto final es muy convincente.

8.3. Conclusiones personales y futuro en la empresa

Personalmente, las practicas me han parecido estupendas, ya que además de aprender mucho con ellas, me han servido para comprobar que puedo trabajar con lo aprendido en la carrera, y con mis propios conocimientos puedo crear y programar desde cero una aplicación real, también que mi trabajo es bueno y reconocido tanto por la empresa como por el cliente.

Además gracias a estas prácticas y a la manera de llevar las mismas, he sido contratado por la empresa donde he realizado las practicas, Robottions, en principio para 6 meses y con unas condiciones favorables, por lo que seguiré con el proyecto total hasta el final del mismo, y con la creación de otros proyectos posteriores.

Bibliografía

[1] Human Robot Solutions S.L, Francisco Javier GARCÍA GANDÍA y José CANDEA ROMÁN, *https://www.robottions.com/* [Muy consultada]

[2] Universidad Jaume I, Asignaturas: EI1023, EI1027, EI1054 *https://aulavirtual.uji.es/* [Muy consultada]

[3] Microsfot Visual Studio/Docs *https://docs.microsoft.com/* [Muy consultada]

[4] Community Viusal Studio, Microsoft *https://developercommunity.visualstudio.com/* [Muy consultada]

[5] Stack Overflow Questions *https://stackoverflow.com/* [Muy consultada]

[6] Reddit Questions *https://www.reddit.com/* [Consulta: 27 de Marzo de 2021]

[7] Github FilippoBoido *https://github.com/FilippoBoido* [Consulta: 29 de Marzo de 2021]

[8] Csharp.net *http://csharp.net-informations.com/* [Consulta: 29 de Marzo de 2021]

[9] techotopia *https://www.techotopia.com/* [Consulta: 5 de Abril de 2021]

[10] docs.unity3d *https://docs.unity3d.com/* [Consulta: 7 de Abril de 2021]

[11] codeproject *https://www.codeproject.com/* [Consulta: 15 de Abril de 2021]

[12] visualstudiomagazine *visualstudiomagazine.com/* [Consulta: 17 de Abril de 2021]

[13] c-sharpcorner *https://www.c-sharpcorner.com/* [Consulta: 26 de Abril de 2021]

[14] geeksforgeeks *https://www.geeksforgeeks.org/* [Consulta: 5 de Mayo de 2021]

[15] tutorialsEU *https://www.youtube.com/channel/* [Consulta: 5 de Mayo de 2021]

[16] wpf-tutorial *https://wpf-tutorial.com/* [Consulta: 11 de Mayo de 2021]

[17] soka.gitlab *https://soka.gitlab.io/csharp/wpf-xaml/* [Consulta: 12 de Mayo de 2021]

[18] it-swarm-es *https://www.it-swarm-es.com/* [Consulta: 15 de Mayo de 2021]

[19] guru99 *https://www.guru99.com/* [Consulta: 17 de Mayo de 2021]

[20] Github PKaelin *https://github.com/PKaelin/* [Consulta: 29 de Marzo de 2021]

[21] telerik *https://www.telerik.com/* [Consulta: 3 de Junio de 2021]

[22] collab365 *https://collab365.com/* [Consulta: 3 de Junio de 2021]

[23] AJPDSOFT *https://www.ajpdsoft.com/* [Consulta: 7 de Junio de 2021]

[24] TECHPOWERUP *www.techpowerup.com* [Consulta: 10 de Junio de 2021]

[25] PROGRAMARFACIL *https://programarfacil.com/* [Consulta: 12 de Junio de 2021]

[26] BLOG NOSTATIC *http://blog.nostatic.org/* [Consulta: 12 de Junio de 2021]

[27] DANIWEB *https://www.daniweb.com/* [Consulta: 15 de Junio de 2021]