

# **Simulador del protocolo de control de enlace de datos de alto nivel (HDLC)**

**Trabajo de Fin de Grado**

**INGENIERÍA INFORMÁTICA**



**Julio de 2023**

**Autor**

*Raúl Melgosa Salvador*

**Tutores**

*Ángeles M<sup>a</sup> Moreno Montero*

*Sergio Bravo Martín*



Dña. Ángeles Mª Moreno Montero y D. Sergio Bravo Martín, profesores del Departamento de Informática y Automática de la Universidad de Salamanca,

HACEN CONSTAR:

Que el trabajo titulado “Simulador del protocolo de control de enlace de datos de alto nivel (HDLC)”, que se presenta, ha sido realizado por Raúl Melgosa Salvador, con DNI \*\*\*\*5689\* y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado en Ingeniería Informática en esta Universidad.

Salamanca, 4 de junio de 2023

Fdo.: Ángeles Mª Moreno Montero

Fdo. Sergio Bravo Martín



# Contenido

Introducción .....	1
Objetivos .....	3
Conceptos teóricos .....	5
Conceptos básicos .....	5
Estructura de la trama .....	6
Estructura del campo de control .....	7
Trama de información (I) .....	7
Trama de supervisión (S) .....	8
Trama no numerada (U) .....	9
Trabajos relacionados .....	11
Visual HDLC .....	11
Configuración de la estación .....	12
Envío de tramas .....	15
Representación y visualización de tramas .....	16
Funcionalidades adicionales .....	17
Errores en el funcionamiento .....	19
Otras herramientas .....	19
GNS3 .....	19
Cisco Packet Tracer .....	19
Conclusión .....	20
Metodologías, técnicas y herramientas .....	20
Metodologías .....	20
Proceso Unificado (UP) .....	20
Diseño Centrado en el Usuario (DCU) .....	23
Técnicas .....	26
Modelo MVVM .....	26
Metodología de Durán y Bernárdez .....	26
Análisis jerárquico de tareas (HTA) .....	29
Textos de los procedimientos .....	30
Formularios .....	30
Técnica del conductor .....	31
Herramientas .....	31
Visual Studio 2019 .....	31
Visual Paradigm .....	31
Adobe XD .....	32
Windows Presentation Foundation (WPF) .....	32
Visual_HDLC .....	33

Adobe Color .....	33
EZ Estimate .....	33
Windows Project .....	33
Google Forms .....	33
Lucidchart .....	34
Draw.io .....	34
Aspectos relevantes del desarrollo.....	35
Descripción del ciclo de vida .....	35
Modelado de negocio .....	38
Investigación sobre el funcionamiento teórico del protocolo HDLC .....	38
Investigación de proyectos similares existentes .....	38
Investigación sobre técnicas de comunicación entre procesos .....	38
Investigación sobre la creación de hilos en el lenguaje de programación utilizado.....	39
Investigación sobre posibles aplicaciones y utilidades del proyecto .....	40
Investigación de posibles vías de mercado .....	41
Requisitos .....	41
Reuniones con el cliente .....	41
Definición de los objetivos del proyecto .....	41
Definición del ámbito del sistema.....	42
Identificación de actores .....	42
Identificación de requisitos no funcionales .....	42
Identificación de requisitos de información.....	43
Identificación de paquetes funcionales.....	43
Identificación de requisitos funcionales o casos de uso .....	44
Realización de encuestas de contenido .....	45
Identificación de escenarios de uso .....	46
Elaboración del diagrama de casos de uso.....	47
Elaboración de la matriz de rastreabilidad.....	48
Análisis.....	49
Identificación de clases entidad del análisis .....	49
Identificación de paquetes del análisis .....	49
Elaboración del modelo del dominio.....	50
Identificación de otras clases del análisis.....	51
Realización de casos de uso a nivel de análisis.....	51
Análisis de tareas.....	52
Elaboración de la propuesta de arquitectura .....	53
Diseño .....	54

Elaboración del mapa del sitio .....	54
Toma de decisiones de diseño sobre la interfaz gráfica .....	55
Elaboración de un esquema de la visualización de la interfaz gráfica ( <i>Wireframe</i> ).....	56
Identificación de subsistemas .....	57
Identificación de clases de diseño.....	58
Realización de casos de uso a nivel de diseño .....	59
Elaboración del diagrama de subsistemas .....	60
Elaboración del diagrama de despliegue .....	61
Implementación .....	62
Elaboración de un prototipo interactivo .....	62
Implementación del simulador del protocolo HDLC .....	63
Pruebas.....	73
Realización de pruebas en entornos similares .....	73
Definición de casos de prueba .....	74
Definición de pruebas de integración .....	75
Realización de pruebas de usabilidad sobre el prototipo interactivo.....	75
Realización de pruebas funcionales.....	76
Realización de pruebas de integración .....	77
Realización de pruebas de usuario del sistema completo .....	79
Sistema final.....	79
Ventana principal .....	80
Configuración.....	80
Establecimiento de la conexión física.....	82
Envío de tramas.....	83
Visualización del detalle de una trama .....	85
<i>Timeouts</i> .....	85
Envío de tramas erróneas.....	87
Conclusiones y líneas de trabajos futuras .....	88
Bibliografía.....	91

# Índice de figuras

Figura 1: Estructura de una trama HDLC .....	7
Figura 2: Tipos de tramas en función de la estructura del campo de control .....	7
Figura 3: Tipos de tramas de supervisión.....	9
Figura 4: Tipos de tramas no numeradas.....	10
Figura 5: Ventana principal Visual_HDLC .....	11
Figura 6: Configuración de la conexión en Visual_HDLC .....	12
Figura 7: Configuración del protocolo en Visual_HDLC.....	13
Figura 8: Configuración del canal de transmisión en Visual_HDLC .....	14
Figura 9: Configuración del administrador en Visual_HDLC.....	14
Figura 10: Establecimiento de la conexión en Visual_HDLC .....	15
Figura 11: Configuración de la trama a enviar en Visual_HDLC.....	16
Figura 12: Envío de una trama codificada en hexadecimal en Visual_HDLC .....	16
Figura 13: Composición detallada de la trama en Visual_HDLC .....	17
Figura 14: Representación gráfica de un intercambio de tramas en Visual_HDLC.....	18
Figura 15: Modelos del Proceso Unificado .....	22
Figura 16: Modelos del Proceso Unificado y disciplina a la que pertenecen.....	23
Figura 17: Fases y subfases del diseño de la interfaz centrado en el usuario .....	25
Figura 18: Esquema patrón MVVM .....	26
Figura 19: Tareas a realizar en la metodología de Durán y Bernárdez.....	27
Figura 20: Tipos de descomposiciones de la metodología HTA .....	29
Figura 21: Ejemplo de análisis de tareas utilizando la técnica HTA .....	30
Figura 22: Paquetes funcionales del proyecto.....	44
Figura 23: Formato matriz de rastreabilidad utilizado .....	48
Figura 24: Diagrama de clases o modelo del dominio del proyecto .....	50
Figura 25: Ejemplo de realización de un caso de uso a nivel de análisis del proyecto .....	52
Figura 26: Ejemplo de análisis de una tarea del proyecto .....	53
Figura 27: Propuesta de arquitectura del proyecto .....	54
Figura 28: Mapa del sitio del proyecto.....	55
Figura 29: Wireframe de la pantalla principal de la estación del proyecto .....	57
Figura 30: Ejemplo de realización de un caso de uso a nivel de diseño del proyecto.....	60
Figura 31: Diagrama de subsistemas de diseño del proyecto .....	61
Figura 32: Diagrama de despliegue del proyecto .....	62
Figura 33: Prototipo de la pantalla principal de la estación del proyecto .....	63
Figura 34: Primera situación en el establecimiento de la conexión.....	66
Figura 35: Segunda situación en el establecimiento de la conexión.....	66
Figura 36: Tercera situación en el establecimiento de la conexión.....	66
Figura 37: Ejemplo de prueba en el entorno de Visual_HDLC .....	74
Figura 38: Ventana principal de la estación de simulador.....	80
Figura 39: Ventana de configuración en la pestaña de Protocolo de simulador.....	81
Figura 40: Ventana de configuración en la pestaña de Modo de trabajo de simulador .....	81
Figura 41: Ventana de configuración en la pestaña de Canal de simulador .....	82
Figura 42: Ventana de la estación durante el establecimiento de la conexión física en el simulador .....	83
Figura 43: Ventana de la estación después del establecimiento de la conexión física en el simulador .....	83



Figura 44: Ventana de visualización de la trama a enviar en el simulador.....	84
Figura 45: Ventana de la estación tras realizar el envío de una trama en el simulador .....	84
Figura 46: Ventana de visualización del detalle de una trama en el simulador .....	85
Figura 47: Ventana de la estación tras la activación de un timeout ante Trama I en el simulador .....	86
Figura 48: Ventana de la estación tras la expiración de un timeout ante trama I en el simulador .....	86
Figura 49: Ventana de la estación tras el envío de una trama errónea en el simulador .....	87

# Índice de tablas

Tabla 1: Resumen de los requisitos no funcionales del proyecto .....	43
Tabla 2: Resumen de los requisitos de información del proyecto .....	43
Tabla 3: Resumen de los requisitos funcionales del proyecto .....	45
Tabla 4: Resumen de los escenarios de uso del proyecto.....	47
Tabla 5: Resumen de las clases del análisis del proyecto.....	51
Tabla 6: Resumen de las clases del diseño del proyecto .....	59
Tabla 7: Tareas planteadas a los usuarios en la evaluación del prototipo interactivo .....	76

# Resumen

En el transcurso del grado en Ingeniería Informática se han estudiado diversos protocolos de distintos ámbitos de la informática. En el ámbito de las redes de computadores, uno de los protocolos estudiados más importantes es el protocolo a nivel de enlace HDLC.

Actualmente, para el aprendizaje del protocolo HDLC en el grado en Ingeniería Informática, se utiliza una herramienta llamada Visual\_HDLC, desarrollada por la Universidad de Barcelona. Esta herramienta presenta algunos errores en su funcionamiento y no se dispone del código fuente, por lo que no se pueden hacer correcciones sobre la herramienta.

De esta manera, la principal motivación de este trabajo de fin de grado es el desarrollo de un simulador del protocolo HDLC que corrija los errores de funcionamiento de la herramienta Visual\_HDLC, y permita a los alumnos del grado, aprender de manera sencilla el funcionamiento del protocolo HDLC.

En el desarrollo de este proyecto, en primer lugar, se analizará en profundidad la herramienta Visual\_HDLC, para determinar qué aspectos deben mantenerse en el desarrollo del nuevo simulador y que aspectos deben permanecer. También se realizará una investigación de los conceptos teóricos del protocolo HDLC.

Más adelante, se realizará el desarrollo completo del proyecto siguiendo las directrices de la Ingeniería del Software y en concreto del Proceso Unificado (UP). De manera paralela, se realizará un diseño de la interfaz centrado en el usuario, ya que la interfaz gráfica se considera un punto crítico en el desarrollo del proyecto.

En último lugar, se evaluarán los resultados de la implementación del simulador del protocolo HDLC con distintos usuarios y se comentarán posibles líneas de investigación futura que se puedan realizar sobre esta herramienta.

**Palabras clave:** *Redes de computadores, protocolo HDLC, simulador, Visual\_HDLC.*

# Abstract

During the course of the degree in Computer Engineering, several protocols from different fields of computer science have been studied. In the field of computer networks, one of the most important protocols studied is the HDLC link level protocol.

Currently, a tool called Visual\_HDLC, developed by the University of Barcelona, is used for learning the HDLC protocol in the Computer Engineering degree. This tool presents some errors in its operation and the source code is not available, so corrections cannot be made on the tool.

In this way, the main motivation of this final degree project is the development of a simulator of the HDLC protocol that fixes the errors of operation of the Visual\_HDLC tool, and allows the students of the degree, to learn in a simple way the operation of the HDLC protocol.

In the development of this project, first of all, the Visual\_HDLC tool will be analyzed in depth to determine which aspects should be maintained in the development of the new simulator and which aspects should remain. An investigation of the theoretical concepts of the HDLC protocol will also be carried out.

Then, the complete development of the project will be carried out following the guidelines of Software Engineering and specifically the Unified Process (UP). In parallel, a user-centered interface design will be carried out, since the graphical interface is considered a critical point in the development of the project.

Finally, the results of the implementation of the HDLC protocol simulator with different users will be evaluated and possible lines of future research on this tool will be discussed.

**Keywords:** *Computer networks, HDLC protocol, simulator, Visual\_HDLC.*

# Introducción

En la actualidad, el mundo de la informática ha sufrido grandes cambios en un periodo de tiempo muy pequeño, donde la evolución ha sido siempre una constante. El ámbito de las redes de computadores ha sido una de las disciplinas de la informática que ha sufrido esta rápida evolución. En cuestión de 50 se ha evolucionado desde las redes de ordenadores locales a una gran red global como internet que permite comunicar a 2 equipos independientemente de su localización y funcionamiento interno.

Una de las consecuencias del rápido crecimiento de las redes de ordenadores, es la creación de un estándar el cual recoja los distintos protocolos de red y los clasifique en distintos niveles. Este estándar se trata del modelo OSI. El modelo OSI contempla el uso de 7 niveles donde en cada nivel se agrupan los protocolos de red destinados a realizar un conjunto de tareas específicas.

Uno de los niveles más importantes que se definen en el modelo OSI es el nivel de enlace. Las funciones más importantes de este nivel de enlace son:

- Sincronización de trama o entramado
- Control de errores
- Control de flujo
- Gestión del enlace
- Proporcionar servicios a la capa de red

Uno de los protocolos más extendidos y popularizados que proporciona estos servicios a nivel de enlace, es el protocolo HDLC. Este protocolo orientado a bit, permite un intercambio de datos fiable entre dos equipos conectados a través de cualquier medio físico.

De esta manera, este proyecto nace con la intención de profundizar en el funcionamiento del protocolo HDLC y permitir a distintos estudiantes aprender y experimentar sobre el funcionamiento de este protocolo. Existen múltiples variantes de este protocolo, sin embargo, este proyecto se centrará en la variante LAP-M.

La idea principal es desarrollar una herramienta que simule la implementación del protocolo HDLC entre dos estaciones físicamente conectadas. El simulador incluirá una implementación de la variante LAP-M del protocolo HDLC, la cual se implementará en el modo de trabajo no balanceado en la cual las dos estaciones conectadas podrán enviar y recibir tramas en cualquier momento.

En cuanto a la estructura de esta memoria, en primer lugar, se establecerán los objetivos del proyecto, tanto objetivos técnicos como personales. En segundo lugar, se introducirán una serie de conceptos teóricos sobre el funcionamiento del protocolo HDLC. Mas adelante, se citarán las distintas metodologías, técnicas y herramientas utilizadas para el desarrollo del proyecto. A continuación, se presentarán los aspectos más relevantes del desarrollo del sistema, teniendo en cuenta la evolución del ciclo de vida del software. En último lugar, se presentarán los resultados y conclusiones obtenidas del desarrollo del proyecto, así como posibles líneas de investigación futura.

Junto a este documento, se entregarán una serie de anexos, los cuales incluyen información detallada sobre distintos aspectos del desarrollo del proyecto. En el desarrollo de este proyecto, se incluyen los siguientes anexos:

- **Anexo I: Plan de proyecto:** Este anexo incluirá la estimación del esfuerzo del proyecto, la planificación temporal del mismo y un análisis del Diagrama de Gantt obtenido en la planificación temporal.
- **Anexo II.a: Especificación de requisitos software:** Este anexo incluirá la elicitación de requisitos la definición formal de los objetivos del sistema y elaboración del diagrama de casos de uso. También incluirá las primeras actividades relacionadas con el Desarrollo Centrado en el Usuario.
- **Anexo II.b: Análisis de requisitos:** Este anexo incluirá el diagrama de clases del sistema y el análisis de los requisitos previamente identificados. También se incluirá el análisis de tareas.
- **Anexo III: Especificación de diseño:** Este anexo incluirá el diseño de la estructura del sistema y la relación de los casos de uso previamente identificados a nivel de diseño. También incluirá actividades del diseño de la interfaz.
- **Anexo IV: Documentación técnica de programación:** Este anexo incluirá cierta información relevante sobre la implementación del proyecto.
- **Anexo V: Manual de usuario:** Este anexo incluirá información relevante sobre las distintas funcionalidades del sistema, así como la representación visual de la aplicación.

# Objetivos

En el desarrollo de este proyecto se persiguen dos tipos de objetivos. Por un lado, se busca cumplir una serie de **objetivos técnicos** los cuales hacen referencia a los objetivos marcados por los requisitos del software a construir. Por otro lado, se busca cumplir una serie de **objetivos personales** los cuales hacen referencia a los objetivos que se plantea el alumno en el desarrollo del proyecto.

Los **objetivos técnicos** van a guiar desde el principio el desarrollo del proyecto, ya que definen cómo debe ser la herramienta a desarrollar.

En este proyecto en concreto, distinguimos los siguientes objetivos técnicos:

- **Simular el protocolo HDLC en concreto su modo de trabajo balanceado (BA):** Dentro del protocolo HDLC existen 3 modos de funcionamiento. Para el desarrollo de este proyecto, se hará énfasis en el modo de trabajo balanceado (BA), en el cual todas las estaciones tienen la capacidad de enviar y recibir tramas en cualquier momento.
- **Implementar el funcionamiento del protocolo HDLC:** El principal objetivo del proyecto es implementar el funcionamiento del protocolo HDLC. En este proyecto en particular, se buscará implementar la variante LAP-M del protocolo HDLC.
- **Permitir seleccionar ciertos parámetros en el funcionamiento del protocolo:** El simulador debe permitir configurar ciertos parámetros sobre el protocolo HDLC, sobre el modo de funcionamiento de las estaciones y sobre el canal de transmisión que une las estaciones.
- **Enviar y recibir tramas:** Cada estación debe tener la capacidad de enviar y recibir tramas a otra estación físicamente conectada.
- **Gestionar los timeout y reintentos de retransmisión de tramas:** Para garantizar un correcto control de los posibles errores que se puedan producir en el intercambio de tramas, se utilizarán distintos *timeouts* que gestionan el reenvío de tramas cuando transcurra un periodo de tiempo determinado.
- **Mostrar las tramas que se generan en cada instante:** Para obtener un mayor entendimiento del funcionamiento del protocolo HDLC, se deberá incluir un mecanismo que muestre el contenido específico de cada una de las tramas intercambiadas, así como un registro con las tramas enviadas y recibidas por cada estación.
- **Generar tráfico con tasas de errores:** El medio de transmisión a través del cual se intercambian las tramas puede fallar. Es por esto por lo que se debe incluir la posibilidad de enviar tramas erróneas, ya que esto permitirá al usuario comprender como el protocolo HDLC soluciona los posibles fallos que se produzcan en la capa física.
- **Permitir seleccionar un modo de trabajo automático (se responde cómo específica el protocolo) o manual (el usuario selecciona la trama a enviar):** En el modo de trabajo automático, las estaciones utilizarán *timeouts*, permitirán el envío de tramas erróneas y responderán automáticamente en situaciones específicas. En el modo de trabajo manual, las estaciones no utilizarán *timeouts*, no permitirán el envío de tramas erróneas y no responderán automáticamente en ninguna situación.

- ***Incluir ayudas que apoyen el aprendizaje del protocolo y del propio software:*** El protocolo HDLC, así como el propio simulador pueden tener cierta complejidad, por lo que se considera imprescindible incluir una serie de mecanismos de ayuda que faciliten a los usuarios el aprendizaje del protocolo HDLC así como el manejo de la herramienta.

Los **objetivos personales**, a diferencia de los objetivos técnicos, no son críticos ya que su cumplimiento no es estrictamente necesario para lograr los objetivos técnicos del proyecto. Sin embargo, sería adecuado cumplir en la mayor medida de lo posible estos objetivos, ya que permitirán al alumno ampliar su conocimiento y evolucionar como persona.

En este proyecto en concreto, distinguimos los siguientes objetivos personales:

- ***Desarrollar un proyecto informático de envergadura en solitario:*** A lo largo del grado, se han desarrollado distintos proyectos. Sin embargo, la mayoría de estos proyectos se realizaron en equipo y no eran de una gran envergadura.
- ***Utilizar las metodologías de Ingeniería del Software para desarrollar un proyecto informático:*** A lo largo del grado, se han desarrollado distintos proyectos. Sin embargo, al tratarse de proyectos relativamente pequeños, no se han aplicado las metodologías de Ingeniería del Software, ya que sería demasiado costoso en comparación con el resto del desarrollo.
- ***Realizar el diseño de una interfaz gráfica de usuario sencilla y amigable:*** Un aspecto muy importante en el desarrollo de un proyecto, es la creación de una interfaz gráfica de usuario que sea fácil de usar y comprender. De esta manera, se aplicarán las metodologías de desarrollo centrado en el usuario (DCU) para lograr este cometido.
- ***Adquirir soltura en un lenguaje de programación orientado a objetos:*** Para realizar la implementación del simulador, se utilizará un lenguaje de programación orientado a objetos. En este caso, el lenguaje de programación elegido será C#.
- ***Desarrollar un proyecto que se diferencie de la competencia:*** Una de las motivaciones de este proyecto es mejorar las prestaciones que ofrecen los simuladores del protocolo HDLC ya existentes, como por ejemplo Visual\_HDLC.
- ***Búsqueda de nuevas funcionalidades:*** Una manera de mejorar el proyecto y diferenciar el proyecto de la competencia, es la búsqueda e implementación de nuevas funcionalidades que sean relevantes y útiles para el usuario. Debido a esto, una de las grandes motivaciones de este proyecto es la búsqueda de nuevas funcionalidades que mejoren el producto y aumenten la satisfacción de los usuarios.



# Conceptos teóricos

Comprender el funcionamiento del protocolo HDLC puede ser un proceso un poco complejo. Es por esto por lo que se ha considerado necesario realizar una introducción sobre los conceptos teóricos y básicos del protocolo HDLC, permitiendo así una mayor y mejor comprensión sobre el desarrollo del proyecto.

## Conceptos básicos

HDLC es un protocolo orientado a bit que proporciona servicios a la capa de enlace como el control de errores, la gestión de flujo, el entramado o la gestión del enlace. Su principal objetivo es permitir el intercambio de información fiable entre dos estaciones físicamente conectadas.

Dentro del protocolo HDLC, se diferencian 3 tipos de estaciones:

- **Estaciones primarias:** Las estaciones primarias tienen el control del enlace y envían un tipo de tramas llamadas *órdenes*.
- **Estaciones secundarias:** Las estaciones secundarias no tienen el control del enlace y envían un tipo de tramas llamadas *respuestas*.
- **Estaciones combinadas:** Las estaciones combinadas pueden actuar tanto de estaciones primarias como secundarias por lo que pueden enviar órdenes y respuestas.

Por otro lado, el protocolo HDLC contempla dos tipos distintos para la configuración del enlace:

- **Configuración no balanceada:** En este tipo de configuración, existe una estación primaria y una o varias estaciones secundarias. Si solo existe una estación secundaria, se utiliza un tipo de transmisión *full-duplex*. Si existe más de una estación secundaria, se utiliza un tipo de transmisión *half-duplex*.
- **Configuración balanceada:** En este tipo de configuración, existen 2 estaciones combinadas donde el tipo de transmisión utilizado es *full-duplex*.

Por último, el protocolo HDLC contempla tres modos de funcionamiento o modos de operación:

- **Modo de respuesta normal (NRM):** Utiliza una configuración del enlace no balanceada donde existe una estación primaria, la cual, puede transmitir en cualquier momento ya que tiene el control del enlace y una estación secundaria la cual solo puede transmitir cuando la estación primaria le da permiso para hacerlo (a través de comandos). El tipo de transmisión utilizado es *half-duplex*.

- **Modo de respuesta asíncrono (ARM):** Utiliza una configuración del enlace no balanceada donde existe una estación primaria, la cual, puede transmitir en cualquier momento ya que tiene el control del enlace y una estación secundaria la cual puede transmitir sin permiso explícito de la estación primaria. Sin embargo, las responsabilidades de la iniciación, mantenimiento y finalización siguen recayendo en la estación primaria. El tipo de transmisión utilizado es *full-duplex*.
- **Modo balanceado asíncrono (ABM):** Utiliza una configuración del enlace balanceada donde cualquiera de las estaciones puede transmitir sin permiso explícito de la otra estación. El tipo de transmisión utilizado es *full-duplex*.

En el caso particular de este proyecto, se hará especial énfasis en el modo de operación balanceado asíncrono (ABM) y en el uso de estaciones combinadas las cuales tienen la capacidad de enviar órdenes y respuestas.

Para la realización de las labores de control de flujo, el protocolo HDLC hará uso de un protocolo de **ventana deslizante**, donde se entiende por ventana, el número máximo de tramas de información que pueden encontrarse sin confirmación en un determinado momento.

Para implementar este protocolo de ventana deslizante, es necesario numerar las tramas enviadas (ya se verá más adelante como HDLC realiza esto) teniendo en cuenta que la numeración se realizará en función del tamaño de la ventana (numeración módulo n, es decir, numeración del 0 al n-1).

## Estructura de la trama

El protocolo HDLC se encarga, entre otras cosas, de realizar el entramado, es decir, delimitar la información a transmitir en tramas. El formato de las tramas en HDLC se encuentra estandarizado y consta de la siguiente estructura:

- **Delimitador o flag inicial:** Conjunto de 8 bits cuyo valor es 01111110 y marca el inicio de una trama.
- **Campo de dirección:** Conjunto de 8 bits que indican la estación a la cual va dirigida el envío de la trama (en el caso de que se trate de una orden o comando) o que indica la estación que realiza el envío de la trama (en el caso de que se trate de una respuesta).
- **Campo de control:** Conjunto de 8 bits que indica el tipo de trama del protocolo. Permite realizar tareas como la sincronización o el control de flujo.
- **Campo de información:** Este campo contiene la propia información que se desea transmitir. Este campo puede estar vacío. Existe un límite máximo (MTU) para el campo de información, el cual limita la cantidad de información que se puede intercambiar en una trama. Generalmente este MTU es de 1500 bytes.
- **Campo de redundancia cíclica (CRC):** Conjunto de 16 o 32 bits, que permite detectar posibles fallos en la transmisión de tramas. El protocolo HDLC utiliza el polinomio  $x^{16} + x^{12} + x^5 + 1$  (CRC-CCITT V.41) para el cálculo del CRC.
- **Delimitador o flag final:** Conjunto de 8 bits cuyo valor es 01111110 y marca el final de una trama.

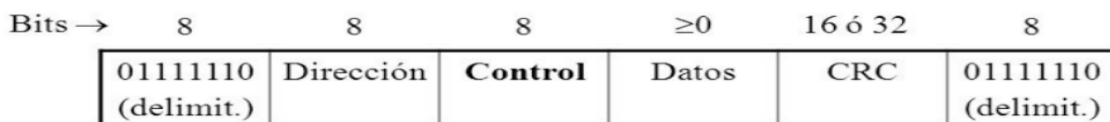


Figura 1: Estructura de una trama HDLC

Una de las problemáticas del uso de delimitadores en las tramas es la confusión de los delimitadores de inicio o fin de la trama con porciones de la propia información de la trama que contengan el mismo valor que los delimitadores. Para evitar este problema, el protocolo HDLC incluye una técnica de **inserción de bits**. En esta técnica, por un lado, el emisor incluye un 0 cada vez que se detectan 5 unos seguidos. Por otro lado, el receptor cuando recibe 5 unos seguidos examinará el sexto bit de manera que este sexto bit es 0, se ignorará y lo eliminará y si el sexto bit es un 1, se tratará de un delimitador.

## Estructura del campo de control

El campo de control de una trama HDLC define 3 tipos diferentes de tramas. Por un lado, se tienen las **tramas de información (I)**, las cuales, se encargan de intercambiar la información generada por el usuario. También realizan tareas de control de errores y gestión de flujo. Por otro lado, existen las **tramas de supervisión (S)**, las cuales, se utilizan para realizar tareas de control de errores y gestión de flujo. Por último, existen las **tramas no numeradas (U)**, las cuales, realizan tareas de establecimiento, mantenimiento y finalización del enlace.

El primer bit o los 2 primeros bits del campo de control hacen referencia al tipo de trama, mientras que el resto de bits del campo de control se utilizan para realizar las labores específicas de cada tipo de trama.

	1	2	3	4	5	6	7	8
Información	0	N(S)			P/F	N(R)		
Supervisión	1	0	S <sub>3</sub>	S <sub>4</sub>	P/F	N(R)		
No numerada	1	1	M <sub>3</sub>	M <sub>4</sub>	P/F	M <sub>6</sub>	M <sub>7</sub>	M <sub>8</sub>

Figura 2: Tipos de tramas en función de la estructura del campo de control

## Trama de información (I)

El campo de control de una **trama de información** tiene la siguiente estructura:

- **Bit 1:** El valor del primer bit del campo de control de una trama de información siempre va a ser 0. Permite identificar una trama de información.

- **Bit 2, 3 y 4:** Los siguientes 3 bits del campo de control de una trama de información hacen referencia al número de secuencia de la trama (NS). Este número de secuencia permite diferenciar tramas de información (se sigue la numeración módulo n teniendo en cuenta el tamaño de la ventana).
- **Bit 5:** El quinto bit del campo de control de una trama de información hace referencia al estado del bit P/F. Si este bit se encuentra activado, la trama se trata de un comando (poll) o una respuesta (final).
- **Bit 6, 7 y 8:** Los últimos 3 bits del campo de control de una trama de información hacen referencia al número de trama esperada (NR). Este número indica el número de secuencia de la trama que se espera recibir de la estación situada en el otro extremo. Este número permite confirmar las tramas recibidas por parte de la estación contraria, realizando así el control de flujo.

Nota: Se utilizan 3 bits para representar tanto el número de secuencia (NS) como el número de trama esperada (NR). Esto se conoce como notación normal modulo 8. Existe también una notación extendida módulo 128 donde se utilizan 7 bits para representar tanto el número de secuencia (NS) como el número de trama esperada (NR).

## Trama de supervisión (S)

El campo de control de una **trama de supervisión** tiene la siguiente estructura:

- **Bit 1 y 2:** El valor del primer bit del campo de control de una trama de supervisión siempre va a ser 1 y el valor del segundo bit siempre va a ser 0. Permite identificar una trama de supervisión.
- **Bit 3 y 4:** Los siguientes 2 bits del campo de control de una trama de supervisión hacen referencia al tipo de trama de supervisión. Existen 4 tipos de tramas de supervisión. El propósito de cada tipo de trama de supervisión se explicará más adelante.
- **Bit 5:** El quinto bit del campo de control de una trama de supervisión hace referencia al estado del bit P/F. Si este bit se encuentra activado, la trama se trata de un comando (poll) o una respuesta (final).
- **Bit 6, 7 y 8:** Los últimos 3 bits del campo de control de una trama de supervisión hacen referencia al número de trama esperada (NR). Este número indica el número de secuencia de la trama que se espera recibir de la estación situada en el otro extremo. Este número permite realizar el control de flujo.

Existen 4 tipos de tramas de supervisión:

- **Trama de Receptor Preparado (RR):** Esta trama permite confirmar tramas de información recibidas y permite indicar a la otra estación la disponibilidad para recibir más tramas de información. Se codifica con el valor 00.
- **Trama de Receptor No Preparado (RNR):** Esta trama permite confirmar tramas de información recibidas y permite indicar a la otra estación la falta de disponibilidad para recibir más tramas de información. Se codifica con el valor 10.
- **Trama de Rechazo (REJ):** Esta trama permite confirmar las tramas de información anteriores al NR y la retransmisión de las tramas de información NR y siguientes. Se codifica con el valor 01.

- **Trama de Rechazo Selectivo (SREJ):** Esta trama permite confirmar las tramas de información anteriores al NR y la retransmisión de la trama de información NR. Se codifica con el valor 11.

	Orden	Respuesta	S <sub>3</sub>	S <sub>4</sub>
Receive Ready	RR	RR	0	0
Receive Not Ready	RNR	RNR	1	0
REJect	REJ	REJ	0	1
Selective REJect	SREJ	SREJ	1	1

Figura 3: Tipos de tramas de supervisión

## Trama no numerada (U)

El campo de control de una **trama no numerada** tiene la siguiente estructura:

- **Bit 1 y 2:** El valor del primer bit del campo de control de una trama de supervisión siempre va a ser 1 y el valor del segundo bit siempre va a ser 1. Permite identificar una trama no numerada.
- **Bit 3, 4, 6, 7 y 8:** Los siguientes 2 bits del campo de control de una trama no numerada hacen referencia al tipo de trama no numerada. Existen múltiples tipos de tramas no numeradas. Los tipos de tramas no numeradas más importantes se explicarán más adelante.
- **Bit 5:** El quinto bit del campo de control de una trama no numerada hace referencia al estado del bit P/F. Si este bit se encuentra activado, la trama se trata de un comando (poll) o una respuesta (final).

Existen 5 tipos fundamentales de tramas no numeradas:

- **Trama de Solicitud de conexión (SABM):** Esta trama permite iniciar el establecimiento de la conexión en el modo de operación balanceado asíncrono. Se codifica con el valor 11100.
- **Trama de Solicitud de desconexión (DISC):** Esta trama permite finalizar el establecimiento de la conexión. Se codifica con el valor 00010.
- **Trama de Asentimiento no Numerado (UA):** Esta trama permite confirmar una petición de inicio o fin del establecimiento de la conexión. Se codifica con el valor 00110.
- **Trama de Modo Desconectado (DM):** Esta trama permite confirmar una petición de fin del establecimiento de la conexión o denegar una petición de inicio de establecimiento de conexión. Se codifica con el valor 11000.
- **Trama de Rechazo de Trama (FRMR):** Esta trama permite indicar que la trama recibida tiene un error irreparable por transmisión. Al recibir esta trama se realiza el restablecimiento del enlace. Se codifica con el valor 10001.

Orden	Respuesta	M3	M4	M6	M7	M8
SNRM		0	0	0	0	1
SNRME		1	1	0	1	1
SARM	DM	1	1	0	0	0
SARME		1	1	0	1	0
SABM		1	1	1	0	0
SABME		1	1	1	1	0
DISC	RD	0	0	0	1	0
	UA	0	0	1	1	0
	FRMR	1	0	0	0	1
SIM	RIM	1	0	0	0	0
TEST	TEST	0	0	1	1	1
XID	XID	1	1	1	0	1
UI	UI	0	0	0	0	0
RSET		1	1	0	0	1
UP		0	0	1	0	0
AC0	AC0	1	0	1	1	0
AC1	AC1	1	0	1	1	1

Figura 4: Tipos de tramas no numeradas

# Trabajos relacionados

Un aspecto muy importante a tener en cuenta en el desarrollo de este proyecto es el análisis de herramientas y trabajos similares que traten una temática similar. Esto se conoce comúnmente como **estado del arte**. El estado de arte tiene como principal objetivo, recopilar y obtener información sobre la temática del proyecto y utilizar dicha información para el desarrollo del propio proyecto.

Habitualmente, el estado del arte se basa en la investigación de distintos artículos y estudios sobre una temática determinada. En el caso particular de este proyecto, los artículos tratan sobre el funcionamiento teórico del protocolo HDLC, pero no tratan el tema de la implementación del protocolo HDLC en un simulador. Es por esto por lo que el estado del arte en este proyecto se centrará en el análisis de otras herramientas similares que realicen la simulación del protocolo HDLC.

## Visual HDLC

En la asignatura de Redes de Computadores I del grado de Ingeniería Informática, se utiliza un simulador llamado **Visual\_HDLC** el cual ha sido desarrollado por la Universidad de Barcelona. Esta herramienta implementa el protocolo HDLC en su variante LAP-B en el modo balanceado asíncrono (ABM) aunque los desarrolladores también plantearon incluir los modos ARM y NRM en una futura versión de la herramienta.

En el entorno de Visual\_HDLC, cada instancia de la aplicación corresponde con una estación, por lo que será necesario el uso de dos instancias de la aplicación para establecer una comunicación e intercambio de tramas entre estaciones. La visualización de la pantalla principal de la estación es la siguiente:

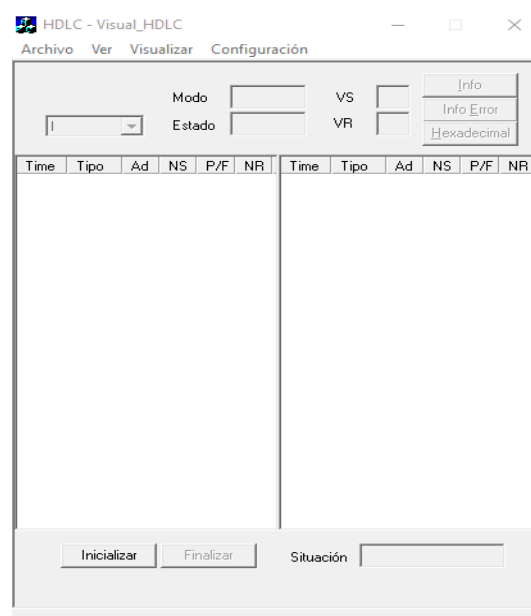


Figura 5: Ventana principal Visual\_HDLC

## Configuración de la estación

Para cada estación, se tienen distintas categorías de configuraciones que se pueden modificar. En primer lugar, se tiene la **configuración de la conexión**. Esta configuración hace referencia al tipo de conexión empleado para conectar las dos estaciones. Existen dos modos de conexión: el **modo local** y el **modo remoto**. En el **modo local**, las dos estaciones se encuentran en la misma máquina mientras que en el **modo remoto** las estaciones se encuentran en máquinas diferentes conectadas a través del protocolo TCP/IP.

Dentro del apartado de configuración de la estación, también se tiene la posibilidad de configurar el nombre y la dirección de la estación. Para el nombre de la estación, se puede elegir entre “Estación A” y “Estación B” mientras que para la dirección de la estación se puede configurar un número cualquiera del 0 al 99. Es imprescindible que los nombres y direcciones de las estaciones sean distintos para realizar la comunicación.

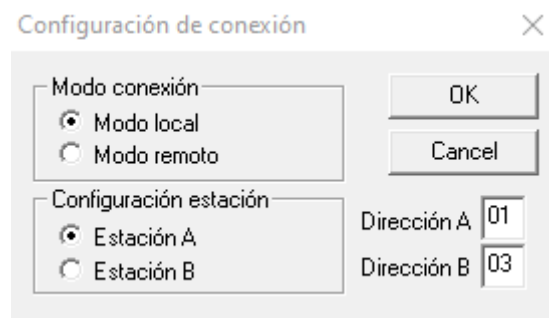


Figura 6: Configuración de la conexión en Visual\_HDLC

En segundo lugar, se tiene la **configuración del modo de trabajo** de la estación. Esta configuración hace referencia al comportamiento de la estación en ciertas situaciones. Existen dos tipos de modos de trabajo para la estación: el **modo manual** y el **modo semiautomático**.

En el **modo semiautomático**, la estación responderá de manera automática en las siguientes situaciones:

- Responderá automáticamente cuando reciba una petición de conexión (SABM) o una petición de desconexión (DISC).
- Responderá automáticamente cuando reciba una trama con el bit de sondeo activado.
- Retransmitirá las tramas rechazadas a través de un comando REJ.
- Enviará una trama de rechazo de trama (FRMR), cuando se produzca una situación de excepción.
- Activará el bit de sondeo automáticamente antes de que se agote el tamaño de la ventana de la estación.
- Generará una trama automáticamente cuando se agote un *timeout*.
- Permitirá la generación de tramas erróneas con una determinada probabilidad especificada en la configuración.

En el **modo manual**, la estación no responderá de manera automática en ningún tipo de situación. Tampoco se gestionarán *timeouts* ni se generarán tramas erróneas.



En tercer lugar, se tiene la **configuración del protocolo**. Esta configuración hace referencia a ciertos mecanismos utilizados para el control de flujo. Por un lado, se permite configurar el **tamaño de la ventana** con un valor entre 2 y 7. También permite configurar el **número máximo de tramas erróneas consecutivas** antes de desconectar el enlace.

Por otro lado, se permite configurar la duración de distintos **timeouts**. La especificación de los valores de los **timeouts** se realiza en milisegundos. Existen 3 tipos de **timeouts**:

- **Timeout ante command:** Tiempo máximo antes de que la estación reaccione cuando no recibe respuesta a un comando con el bit de sondeo activado.
- **Timeout ante trama I:** Tiempo máximo antes de que la estación reaccione cuando no recibe reconocimiento/confirmación a una trama(s) de información enviada.
- **Timeout ante request:** Tiempo máximo antes de que la estación reaccione cuando una estación envía una respuesta que requiere de un comando de vuelta.

Configuración del protocolo utilizado

Tipo de transmisión:  
☐ Half duplex  
☒ Full-duplex

Modo funcionamiento:  
☒ ABM  
☐ ARM  
☐ NRM

Tramas RESPONSE ☐

Tamaño ventana: 7

Tramas erróneas consecutivas permitidas: 3

Timeout ante COMMAND: 20000 ms

Timeout ante trama I: 20000 ms

Timeout ante REQUEST: 20000 ms

OK Cancel

Figura 7: Configuración del protocolo en Visual\_HDLC

En cuarto lugar, se tiene la **configuración del canal de transmisión**. Esta configuración hace referencia a las características del medio de transmisión físico por el que circulan las estaciones.

Por un lado, se permite configurar el **retardo** de la transmisión, es decir, la diferencia de tiempo entre el envío de una trama por parte del origen y la recepción por parte del destino (se especifica en milisegundos). A diferencia de otros parámetros de configuración, se permite aplicar distintos retardos en las estaciones ya que el retardo sólo afecta al envío y no a la recepción.

Por otro lado, se permite configurar la **probabilidad de error** de la transmisión, es decir, la probabilidad de que una trama enviada sea errónea. Esta probabilidad puede tomar valores desde 0 a un valor máximo, el cual se especifica en la configuración del administrador.

Configuración del canal

Retardo 1000 ms

Probabilidad de error 0

OK Cancel

Figura 8: Configuración del canal de transmisión en Visual\_HDLC

En quinto lugar, se tiene la **configuración del administrador**. Esta configuración hace referencia a la configuración del equipo necesaria para conectar vía remota 2 estaciones. Fundamente, se permite configurar la dirección IP y los números de puerto de las estaciones a conectar. También se puede configurar en este apartado, el valor máximo que puede tomar la probabilidad de error.

Opciones del administrador

Puerto A 120

Puerto B 121

Dirección IP Remota 127 . 0 . 0 . 0

Nombre equipo Puesto 1

Máxima prob. pérdida 0.2

Guardar Cancelar

Figura 9: Configuración del administrador en Visual\_HDLC

Para conectar físicamente 2 estaciones y que puedan empezar a intercambiar tramas, se presionará el botón de inicializar en ambas estaciones. Si existe algún tipo de error en la configuración de la estación (por ejemplo, si las estaciones tienen el mismo nombre), no se podrá realizar la conexión física.

Una vez se haya establecido la conexión física entre ambas estaciones, la apariencia de la ventana será la siguiente:

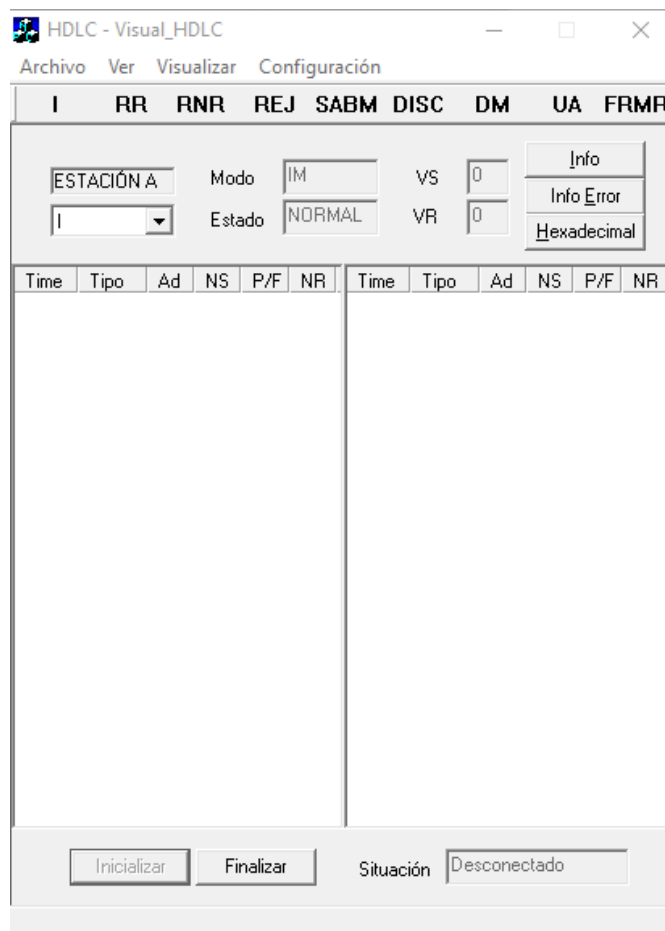


Figura 10: Establecimiento de la conexión en Visual\_HDLC

## Envío de tramas

Aparecerá en la parte superior de la ventana, una nueva barra que permitirá el envío de distintos tipos de tramas. Para enviar una trama, simplemente habrá que pulsar en el botón del tipo de trama correspondiente. Se desplegará una nueva ventana en la que se permitirá al usuario configurar ciertos parámetros de la trama. Los parámetros de la trama que pueden ser configurados son:

- El campo de direcciones o bit C/R (comando o respuesta)
- El estado del bit de sondeo
- Los valores del número de secuencia (NS) y número de trama esperada (NR) en el caso de que estos números tengan sentido en la trama que se va a enviar (por ejemplo, las tramas no numeradas no utilizan ni NR ni NS).

Si la trama a enviar se trata de una trama de información, el usuario también puede configurar el contenido de la información de la trama. Además, puede marcar directamente la opción de enviar una trama de información errónea (aun estando en el modo de trabajo manual).

Trama a enviar

Tipo trama	Address	C/F	NS	P/F	NR
I	B	C	0	0	0

Info

CRC erróneo ☐

Enviar Cancel

Figura 11: Configuración de la trama a enviar en Visual\_HDLC

Por otro lado, también es posible enviar tramas de información y tramas de información de manera directa sin que el usuario tenga que configurar los distintos valores de forma manual. En este caso, se configurarán los valores de la trama en función de los valores NS (número de secuencia de la estación) y NR (número de trama esperada de la estación).

También es posible enviar tramas introduciendo su codificación hexadecimal. En este caso, el usuario debe proporcionar la información del campo de control, el campo de dirección y el campo de información mientras que los *flags* y el CRC son calculados automáticamente por la estación.

Introducir trama en hexadecimal

Únicamente introducir los campos:  
ADDRESS, CONTROL e INFO.

Info

OK Cancel

Figura 12: Envío de una trama codificada en hexadecimal en Visual\_HDLC

## Representación y visualización de tramas

Cada vez que una estación envíe o reciba una trama, se representará el envío o recepción de la trama en la tabla correspondiente. Las tramas enviadas se representarán en color negro en la tabla de la izquierda mientras que las tramas recibidas se representarán en color rojo en la tabla de la derecha.

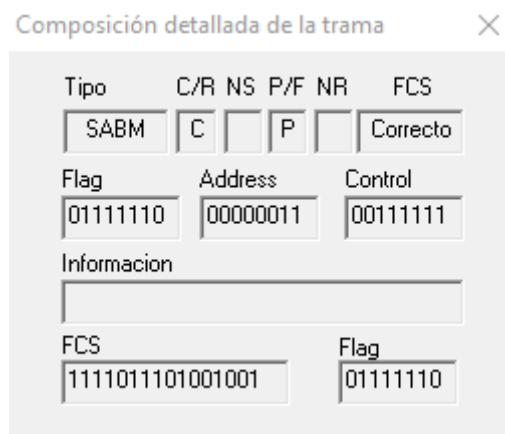
Para cada trama representada, se mostrará la siguiente información:

- El instante en el que se envió o recibió dicha trama.
- El tipo de trama enviada o recibida.
- La dirección de la estación a la que va dirigida la trama en el caso de tratarse de un comando y o la dirección de la estación que responde en el caso de tratarse de una respuesta.
- El valor del número de secuencia (NS) de la trama (solo se representará cuando la trama representada incluya un número de secuencia).
- El estado del bit P/F (solo se mostrará su estado cuando el bit P/F esté activo).
- El valor del número de trama esperada (NR) de la trama (solo se representará cuando la trama representada incluya un número de trama esperada).

Nota: Las 2 estaciones utilizan la misma referencia de tiempo para determinar el instante temporal en el que ha sido enviada o recibida una trama (sincronización).

Para obtener la información detallada de una trama enviada o recibida, se deberá hacer doble click en la trama en cuestión. Al hacer esto, se desplegará una nueva ventana en la que se mostrará la información detallada de la trama seleccionada.

En dicha ventana se mostrará información sobre el tipo de trama, información básica sobre la trama (información sobre el bit C/R, información sobre el bit P/F, información sobre el número de secuencia (NS) y el número de trama esperada (NR)) e información sobre los distintos campos de la trama (*flags*, campo de dirección, campo de control, campo de información y CRC).



Tipo	C/R	NS	P/F	NR	FCS
SABM	C		P		Correcto

Flag	Address	Control
01111110	00000011	00111111

Información

--	--

FCS	Flag
1111011101001001	01111110

Figura 13: Composición detallada de la trama en Visual\_HDLC

## Funcionalidades adicionales

Una de las funcionalidades adicionales que incluye la herramienta Visual\_HDLC, es la representación gráfica de las tramas intercambiadas por una estación. En dicha representación gráfica, se representan las tramas enviadas a través de líneas de color azul y las tramas recibidas con líneas de color verde.

Para cada trama se representa el tipo de trama, la dirección de la estación, el estado del bit P/F si está activado y los números de secuencia (NS) y trama esperada (NR) en las tramas en las que estos números tengan sentido.

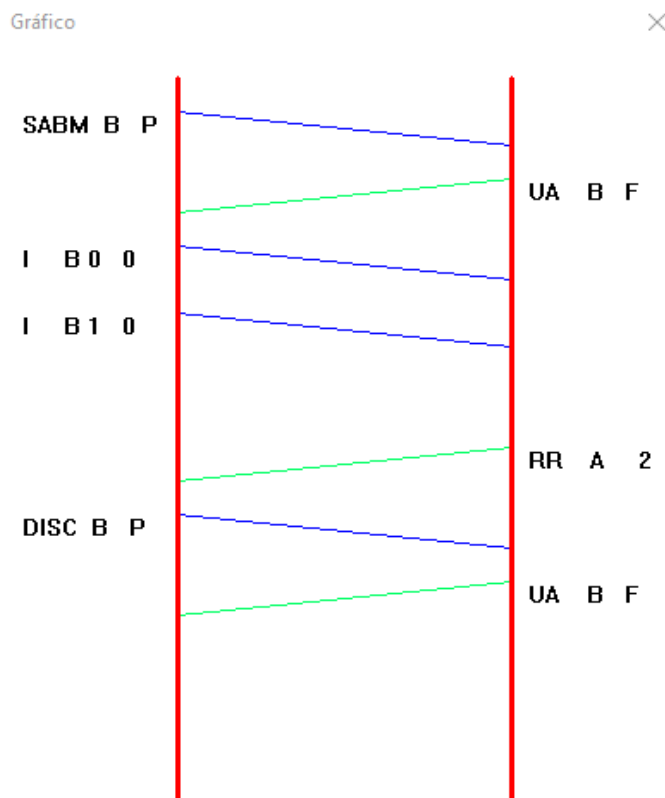


Figura 14: Representación gráfica de un intercambio de tramas en Visual\_HDLC

Por otro lado, en la parte superior de la ventana de la estación se muestra cierta información de interés sobre la estación. En concreto, se muestra el número de secuencia de la estación (NR), el número de trama esperada de la estación (NR), el nombre de la estación y el estado de la estación.

Por último, en la parte inferior de la ventana de la estación se muestra el valor de la situación de la estación. Una estación puede tener las siguientes situaciones:

- **Desconectado:** La estación no está conectada por lo que no puede intercambiar tramas de información con otra estación. Sin embargo, sí existe una conexión física entre ambas estaciones.
- **Conectado:** La estación está conectada por lo que puede intercambiar tramas de información con otra estación.
- **Inicio conexión:** Se ha iniciado el proceso de establecimiento de conexión entre las tramas. Se ha enviado/recibido la solicitud de conexión (SABM) pero no se ha respondido a dicha solicitud de conexión.
- **Inicio desconexión:** Se ha iniciado el proceso de establecimiento de desconexión entre las tramas. Se ha enviado/recibido la solicitud de desconexión (DISC) pero no se ha respondido a dicha solicitud de desconexión.
- **Excepción:** Se ha producido un error irreparable en la transmisión y es necesario un restablecimiento del enlace a través de una trama FRMR.

## Errores en el funcionamiento

Aunque el funcionamiento de la herramienta Visual\_HDLC es bastante correcto, la herramienta cuenta con algunos errores en el funcionamiento. Los errores más importantes y destacables son los siguientes:

- La información del campo de control de la trama se representa en el orden inverso al que debería. Por ejemplo, en una trama de información se representa primero el número de trama esperada (NR), después el estado del bit P/F, después el número de secuencia (NS) y por último el bit que indica el tipo de trama y esto debería mostrarse en el orden inverso.
- Al enviar tramas de información utilizando el botón de la barra superior, no se activa de manera automática el bit de sondeo cuando se va a agotar el tamaño de la ventana. Si las tramas de información se envían utilizando los botones de envío directo, esto no sucede.
- Cuando una trama solicita el restablecimiento del enlace a través del envío de una trama FRMR, no reenvía las tramas de información que quedaban pendientes de confirmar antes de que se produjera el restablecimiento del enlace.
- Si las 2 estaciones se encuentran en modo semiautomático, cuando se rechazan tramas de información enviadas a través de una trama REJ, la estación correspondiente reenvía las tramas rechazadas. Sin embargo, la otra estación genera otra trama de rechazo confirmando las tramas reenviadas.  
El comportamiento ideal sería que después de reenviar las tramas, la estación correspondiente espere el asentimiento y si no recibe dicho asentimiento antes de que expire el *timeout*, sondear a la otra estación en busca de una confirmación.

## Otras herramientas

Existen otras herramientas de simulación que, aunque no estén destinadas directamente a la simulación e implementación del protocolo HDLC, permiten observar tráfico HDLC intercambiado entre varios equipos. A continuación, se comentan brevemente alguna de estas herramientas:

### GNS3

GNS3 es un entorno gráfico de simulación que permite diseñar topologías de red y realizar simulaciones sobre ellos. Esta herramienta se ha utilizado en la asignatura de Redes de Computadores II del grado de Ingeniería Informática, para la simulación y aprendizaje de distintos protocolos como OSPF, DNS o RIP.

En este entorno también es posible generar y visualizar tráfico HDLC. Para ello habría que configurar las estaciones para que encapsulen las tramas utilizando el comando HDLC y luego generar las tramas a nivel de enlace utilizando algún comando como pueda ser *ping*.

### Cisco Packet Tracer

Cisco Packet Tracer es un entorno gráfico de simulación desarrollado por Cisco que permite la creación, configuración y simulación de redes en un entorno virtual. Esta herramienta se ha utilizado en la asignatura de Redes de Computadores II del grado de Ingeniería Informática, para la simulación y aprendizaje de distintos protocolos como DNS o DHCP.

En este entorno también es posible visualizar tráfico HDLC. Para ello habría que configurar las estaciones para que encapsulen las tramas utilizando el comando HDLC y luego generar las tramas a nivel de enlace utilizando algún comando de prueba de conectividad como pueda ser *ping*.

En ambas herramientas de simulación se puede observar la estructura de una trama HDLC. Sin embargo, estas herramientas de simulación no permiten generar directamente tramas HDLC. Es por esto por lo que se necesitan comandos de prueba de conectividad como *ping* para generar las tramas HDLC.

## Conclusión

Aunque existe mucha información teórica sobre el funcionamiento del protocolo HDLC, la información disponible sobre la simulación del protocolo HDLC es escasa ya que solo se tiene constancia de una herramienta que realice directamente la simulación del protocolo HDLC.

Esto hace que el desarrollo del proyecto sea más difícil de realizar debido a la falta de referencias directas a su vez representa una gran motivación ya que se va a dar un paso importante en un ámbito poco desarrollado como es la simulación y aprendizaje del protocolo HDLC.

# Metodologías, técnicas y herramientas

En este apartado, se van a comentar las distintas metodologías, técnicas y herramientas utilizadas en el desarrollo del proyecto. El objetivo es explicar las características esenciales y más importantes de las metodologías utilizadas, las técnicas aplicadas y las herramientas utilizadas de manera que el lector pueda adquirir una idea general de cada una de las metodologías, técnicas y herramientas utilizadas.

## Metodologías

### Proceso Unificado (UP)

La metodología principal utilizada para el desarrollo de este proyecto es el **Proceso Unificado (UP)**. El proceso unificado se trata de un *proceso software* el cual nos va a proporcionar las distintas herramientas, tareas, actividades, métodos para elaborar un producto software de calidad.



Existen otros tipos de procesos software, como el método Scrum, la programación Extrema (XP) o modelos tradicionales (secuenciales). Sin embargo, para el desarrollo de este proyecto, se ha elegido el Proceso Unificado debido a los siguientes motivos:

- Experiencia previa en el uso de este proceso software en distintas asignaturas del grado como *Ingeniería del Software* o *Gestión de Proyectos*.
- Equipo de desarrollo muy reducido (solo existe un miembro en el equipo de desarrollo el cual va a ser responsable de realizar todas las tareas).
- Disponibilidad de un espacio o periodo de trabajo amplio para el desarrollo del proyecto (no es necesario una metodología ágil).
- Necesidad de elaborar una documentación amplia, extensa y correcta del sistema a desarrollar ya que se trata de un proyecto de final de grado.

En el Proceso Unificado, el desarrollo del software se realiza a través de una secuencia de ciclos de desarrollo donde cada ciclo de desarrollo finaliza con una versión entregable del producto. Dentro de cada ciclo, distinguimos 4 fases distintas:

- **Inicio:** Se define el alcance del proyecto, se realiza una descripción inicial del proyecto y se realiza el análisis y desarrollo de los casos de negocio.
- **Elaboración:** Se elabora el diseño de la arquitectura del proyecto, así como la especificación de la gran mayoría de casos de uso.
- **Construcción:** Se construye el producto desde la arquitectura inicial hasta obtener una versión operativa del producto.
- **Transición:** Se realizan correcciones y mejoras sobre el producto para obtener una versión entregable la cual será utilizada por los clientes.

Cada fase cuenta con una serie de iteraciones donde en cada iteración se produce un incremento en el desarrollo del producto. Al final de cada fase se produce un hito el cual consiste en un conjunto de artefactos o entregables. Los hitos resultantes del final de una fase se denominan hitos primarios. Al final de cada iteración también se produce un hito el cual se va a conocer como hito secundario.

Estos hitos tienen como objetivo evaluar y realizar un seguimiento sobre el desarrollo del proyecto, así como en la ayuda en la toma de decisiones y la estimación de recursos necesarios para el desarrollo del proyecto.

Por otro lado, en el Proceso Unificado existen una serie de disciplinas las cuales son un conjunto de tareas o actividades las cuales se utilizan para clasificar las distintas actividades a realizar para desarrollar un producto software. Las disciplinas más importantes son: *Requisitos, Análisis, Diseño, Implementación y Pruebas*.

Cada disciplina proporciona distintos modelos donde los distintos modelos van a presentar distintas dependencias entre ellos. Los modelos más importantes que se van a obtener en el Proceso Unificado son los siguientes:

- **Modelo de casos de uso**
- **Modelo de análisis**
- **Modelo de diseño**

- **Modelo de despliegue**
- **Modelo de implementación**
- **Modelo de pruebas**

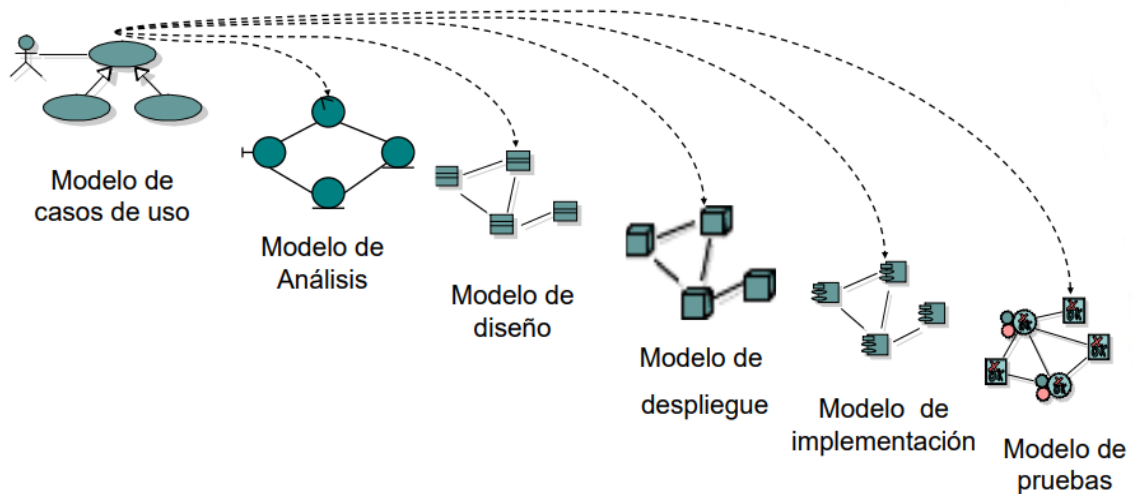


Figura 15: Modelos del Proceso Unificado

El Proceso Unificado cuenta con 3 características principales:

- **Está dirigido por los casos de uso:** Aunque los casos de uso se identifican en la disciplina de Requisitos, la funcionalidad asociada a estos casos de uso tiene una gran influencia en actividades de otras disciplinas como en la elaboración de la arquitectura del sistema, el despliegue del sistema o la definición de casos de prueba.
- **Está centrado en la arquitectura:** Dentro del Proceso Unificado, se distinguen distintos tipos de arquitecturas como la vista lógica, la vista de procesos, la vista de implementación y la vista de despliegue. Estos 4 tipos de arquitectura tienen en común la influencia de la vista de casos de uso. Esto se conoce como el modelo de Arquitectura de 4+1 vistas de Philippe Kruchten.

De esta manera, se van a tener distintas vistas de la arquitectura del sistema donde en cada disciplina del Proceso Unificado se van a tener distintas modelos vistas.

En la siguiente imagen, se muestran las diferentes vistas o modelos del sistema y a la disciplina a la que pertenecen:

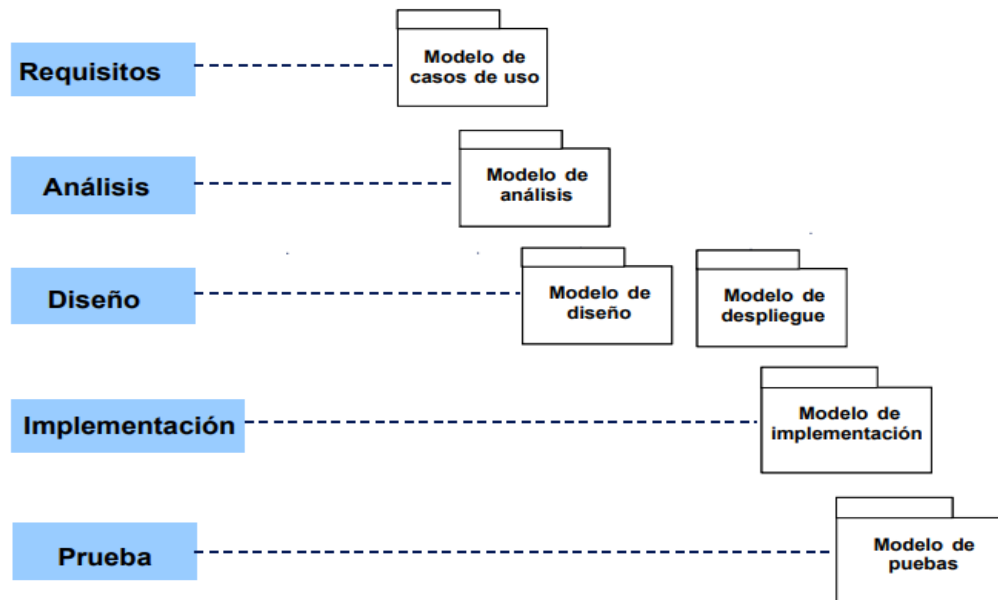


Figura 16: Modelos del Proceso Unificado y disciplina a la que pertenecen

- **Es un proceso iterativo e incremental:** Se divide el esfuerzo de desarrollo del proyecto en una serie de iteraciones donde la realización de cada iteración supone un incremento en el desarrollo del proyecto. En cada iteración, se realizan distintas actividades las cuales desembocan en un hito el cual simboliza el fin de la iteración correspondiente. Cada iteración parte de lo que se ha obtenido en iteraciones anteriores y es por esto por lo que se denomina incremental.

## Diseño Centrado en el Usuario (DCU)

Uno de los aspectos más importante en el desarrollo de cualquier herramienta software, es la correcta elaboración de una interfaz gráfica fácil de usar, comprender y utilizar. Debido a ello, se ha considerado conveniente utilizar una metodología específica que guíe y estructure el desarrollo de la interfaz del usuario.

De esta manera, paralelamente a la metodología del del Proceso Unificado (UP), se utilizará la metodología de Diseño Centrado en el Usuario (DCU). La metodología de Diseño Centrado en el Usuario (DCU) cuenta con un conjunto de métodos y técnicas las cuales pueden ser aplicadas en las distintas fases del diseño de la interfaz del usuario.

La metodología del Diseño Centrado en el Usuario (DCU) tiene una serie de objetivos. Los más importantes son los siguientes:

- Aumentar la satisfacción del usuario con el sistema
- Aumentar la eficiencia y productividad del usuario, es decir, que el usuario pueda realizar fácilmente lo que desea y empleando el mínimo esfuerzo posible.
- Mejorar la usabilidad del sistema
- Aumentar la calidad del producto a desarrollar

Dentro de la metodología del Diseño Centrado en el Usuario (DCU), nos encontramos con 3 fases o etapas principales:

- **Fase de descubrimiento**
- **Fase de conceptualización**
- **Fase de prototipado**

En la **fase de descubrimiento**, se busca recoger los requisitos y necesidades de los usuarios con el objetivo de que el sistema pueda satisfacer dicho requisitos y necesidades del usuario.

En la **fase de conceptualización**, se busca evaluar los resultados obtenidos en la fase de descubrimiento y traducir estos resultados en distintas decisiones de diseño. En esta fase también se tienen en cuenta las posibles alternativas existentes a la hora de realizar el diseño de la interfaz del usuario. No hay que olvidar que en la metodología de Diseño Centrado en el Usuario (DCU) el usuario está involucrado en todo momento y es la piedra angular en torno a la cual se enfoca esta metodología.

En la **fase de prototipado**, se busca elaborar un prototipo de la interfaz de usuario que se desea diseñar y realizar distintas pruebas con usuarios para evaluar la calidad de los diseños realizados. Es esta fase, se pueden obtener una serie de conclusiones sobre el diseño realizado y tomar distintas decisiones con el objetivo de mejorar la calidad de la interfaz de usuario.

Estas fases o etapas se llevarán a cabo de manera iterativa de la misma forma que sucede con el Proceso Unificado (UP) de manera que siempre es posible tener que retroceder a fases o etapas anteriores en el caso de que sea necesario. Estas 3 fases a su vez cuentan con una serie de subfases o subetapas.

La **fase de descubrimiento** se divide en 4 subfases:

- **Análisis de la competencia:** En esta subfase, se busca realizar un pequeño estudio e investigación de interfaces que realicen un cometido similar a la interfaz que se desea desarrollar.
- **Definición de la audiencia:** En esta subfase, se busca definir el público o usuarios objetivo a los que va a ir dirigida la interfaz a desarrollar.
- **Definición de escenarios de uso:** En esta subfase, se define el contexto de utilización de las distintas funcionalidades que incluye y proporciona la interfaz de usuario del sistema, así como la descripción de las experiencias de los usuarios.
- **Encuesta de contenido:** En esta subfase, se define el contenido que debe incluir la interfaz de usuario en función del público objetivo al cual esté dirigido el desarrollo de la interfaz.

La **fase de conceptualización** se divide en 4 subfases:

- **Análisis de tareas:** En esta subfase, se define la secuencia de interacciones que deben ocurrir para que se produzca un determinado resultado. Las secuencias de interacción deben estar basadas en la interacción del sistema con el usuario.

- **Mapa del sitio:** En esta subfase, se define la estructura de la interfaz de usuario así como la navegación entre las distintas ventanas que incorpore la interfaz de usuario.
- **Esquemmatización:** En esta subfase, se realiza un esquema o boceto de las distintas pantallas que va a incorporar la interfaz de usuario. El principal objetivo es la distribución de los distintos componentes que van a formar parte de la interfaz de usuario.
- **Diseño:** En esta subfase, se toman diferentes decisiones de diseño sobre tipografía, disposición de los elementos, colores utilizados, etc.

La **fase de prototipado** se divide en 4 subfases:

- **Prototipado de la interfaz:** En esta subfase, se elabora una versión de alta fidelidad de la interfaz de usuario previamente conceptualizada y diseñada con el objetivo de ser probada por diferentes usuarios.
- **Pruebas de usuario:** En esta subfase, se realizan pruebas de interacción con el prototipo de la interfaz con distintos usuarios.
- **Evaluación heurística:** En esta subfase, se evalúan los resultados obtenidos en las pruebas de usuario con el objetivo de conocer los puntos fuertes y los aspectos que pueden mejorarse.
- **Aprobación:** En esta subfase, se aprueba la interfaz de usuario diseñada en el caso de que la evaluación de la interfaz de usuario sea satisfactoria y cumpla lo que se esperaba de ella.

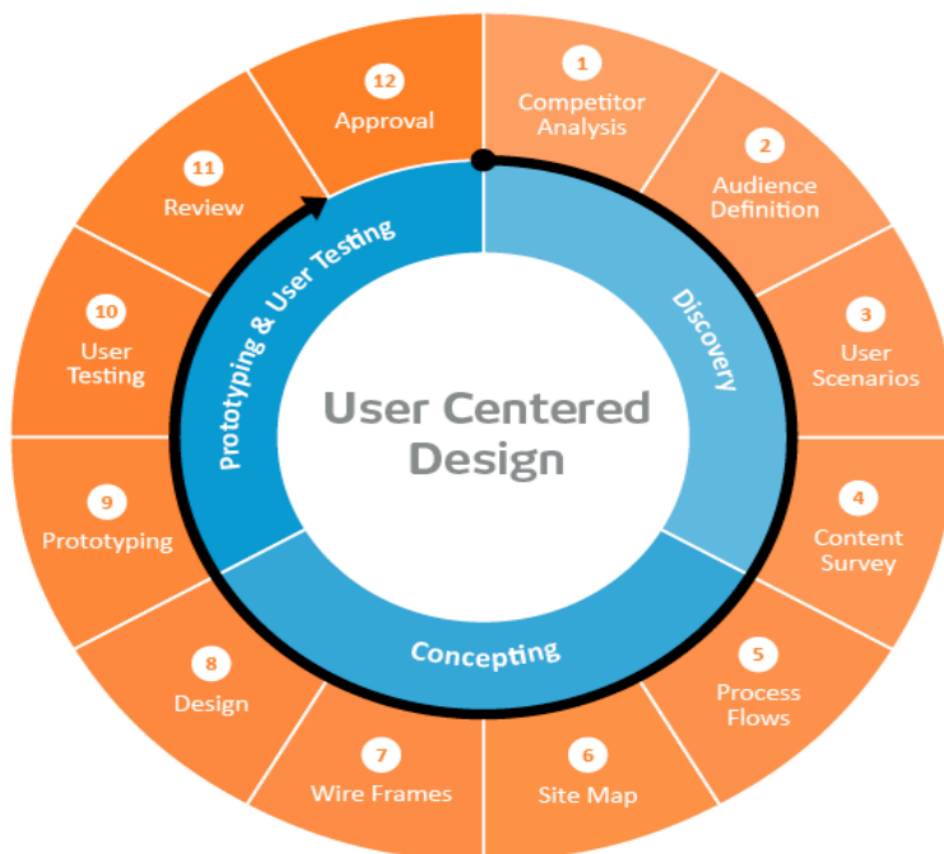


Figura 17: Fases y subfases del diseño de la interfaz centrado en el usuario

# Técnicas

## Modelo MVVM

Para definir la arquitectura del proyecto software a desarrollar, se hará uso del modelo o patrón MVVM (*Model View ViewModel*). Este patrón es similar al clásico patrón MVC con la diferencia esencial de que existe un Modelo de la Vista que reemplaza al Controlador y que se encarga de separar el aspecto visual del sistema de la lógica de negocio interna.

El patrón MVVM cuenta con 3 elementos principales:

- La **vista** la cual tiene como función principal la comunicación e interacción con el usuario.
- El **modelo de la vista**, el cual tiene como función principal implementar la funcionalidad de la aplicación de manera que se encarga de recibir las peticiones de los usuarios que se transmiten a través de la vista y de mostrar la información necesaria de vuelta a los usuarios.
- El **modelo** el cual tiene como función principal realizar el almacenamiento interno de la información que maneja la aplicación.

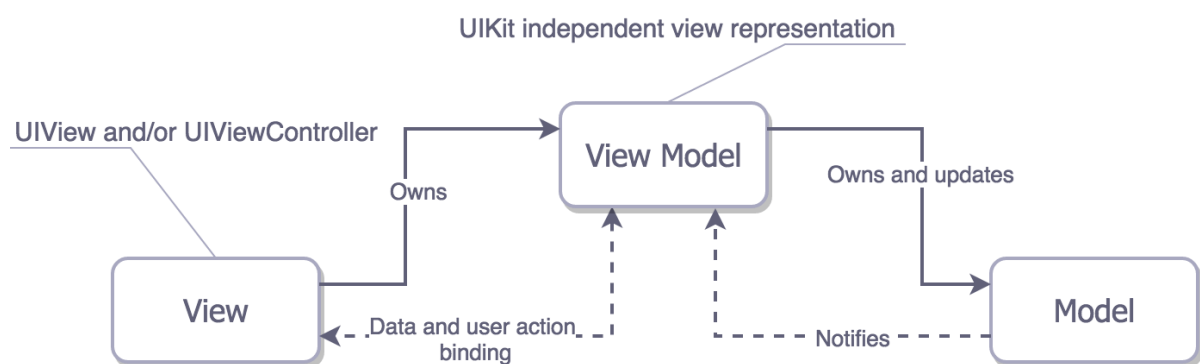


Figura 18: Esquema patrón MVVM

Como se puede observar, el patrón MVVM al igual que el patrón MVC busca separar en capas la aplicación con la diferencia de que la inclusión del ViewModel permite una comunicación más directa entre la vista y el modelo.

## Metodología de Durán y Bernárdez

Aunque se trata de una metodología, se ha encuadra dentro de la sección de técnicas ya que se encuentra incluida dentro de la metodología del Proceso Unificado (UP).

El objetivo de la metodología de Durán y Bernárdez es proporcionar técnicas, herramientas, plantillas para realizar las actividades relacionadas con la actividad o disciplina de captura o elicitación de requisitos. En esta metodología se plantea la existencia de un único producto entregable. Este producto entregable es el Documento de Requisitos del Sistema (DRS).

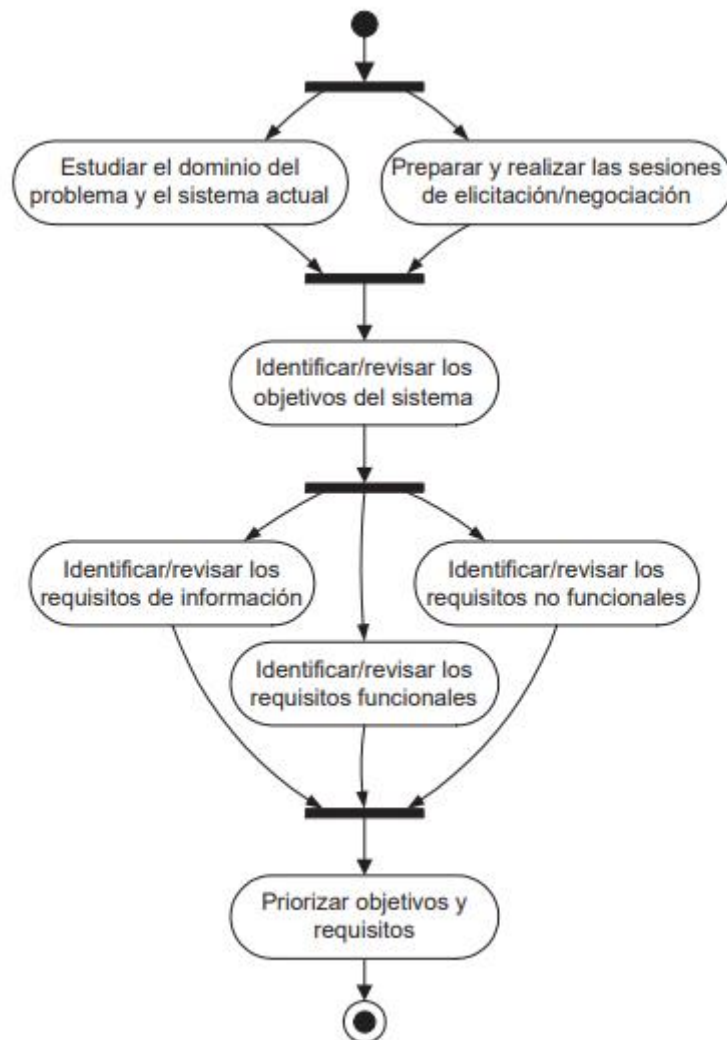


Figura 19: Tareas a realizar en la metodología de Durán y Bernárdez

Dentro de esta metodología, se plantean una serie de tareas que se deberían seguir para realizar una buena Elicitación de Requisitos. Estas tareas tal como se definen en la documentación de la metodología de Durán y Bernárdez son las siguientes:

### **Tarea 1: Obtener información sobre el dominio del problema y el sistema actual**

Esta tarea tiene como objetivo enmarcar el contexto del problema y del sistema antes de realizar la captura de requisitos. Una mala contextualización del dominio del problema o del sistema a desarrollar puede afectar negativamente a tareas posteriores dentro de la captura de requisitos. Esta tarea es de carácter opcional ya que en algunos casos el equipo de desarrollo tiene conocimiento sobre el dominio del problema y tiene conocimiento del sistema a desarrollar.

### **Tarea 2: Preparar y realizar las sesiones de elicitación/negociación**

Esta tarea tiene como objetivo utilizar la información disponible sobre el dominio y sobre el sistema para preparar y realizar la elicitación o captura de requisitos. Esta tarea es sumamente importante ya que en ella se va a obtener y definir qué debe incluir el sistema y que no debe incluir el sistema.

### **Tarea 3: Identificar/revisar los objetivos del sistema**

Esta tarea tiene como objetivo definir los objetivos que debe cumplir el sistema una vez haya acabado su desarrollo y ya se encuentre en explotación. Es posible que los objetivos o parte de ellos se hayan establecido antes de comenzar con el desarrollo del proyecto.

### **Tarea 4: Identificar/revisar los requisitos de almacenamiento de información**

Esta tarea tiene como objetivo identificar las necesidades de almacenamiento de información del sistema a desarrollar y revisar posibles conflictos existentes en la información que se desea almacenar.

### **Tarea 5: Identificar/revisar los requisitos funcionales**

Esta tarea tiene como objetivo identificar las distintas funcionalidades que debe incorporar el sistema a desarrollar (casos de uso) así como el manejo de la información identificada en la tarea anterior. También se identifican aquellas personas o sistemas (actores) que hacen uso de las distintas funcionalidades y gestionan la distinta información del sistema.

En esta tarea, es especialmente importante especificar con claridad y detalle todas las posibles situaciones que se puedan presentar en el uso del sistema, teniendo en cuenta en todo momento posibles excepciones que se puedan producir y alterar el comportamiento normal del sistema.

### **Tarea 6: Identificar/revisar los requisitos no funcionales**

Esta tarea tiene como objetivo identificar aquellos requisitos los cuales no están relacionados con la funcionalidad del sistema pero que son necesarios para el correcto funcionamiento del sistema.

Dentro de los requisitos no funcionales, existen múltiples clasificaciones. Las categorías de requisitos no funcionales más habituales son las siguientes:

- Requisitos de comunicaciones del sistema
- Requisitos de interfaz de usuario
- Requisitos de fiabilidad
- Requisitos de entorno de desarrollo
- Requisitos de portabilidad



## Tarea 7: Priorizar objetivos y requisitos

Esta tarea tiene como objetivo establecer una prioridad a los distintos requisitos y objetivos del sistema con el fin de distinguir aquellos requisitos críticos que son imprescindibles para el funcionamiento del sistema de aquellos que no son tan importantes para el desarrollo del proyecto.

### Análisis jerárquico de tareas (HTA)

El análisis jerárquico de tareas (*Hierarchical Task Analysis*) es una técnica utilizada dentro de la metodología del Diseño Centrado en el Usuario destinada para realizar el análisis de tareas (subfase del DCU).

En la técnica HTA, se realiza una descripción de tareas en términos de operaciones y planes. En esta técnica, las operaciones (descomposición en subtareas) son actividades que realizan las personas para alcanzar un objetivo, y los planes son una descripción de las condiciones que se tienen que dar cuando se realiza cada una de las actividades. Las operaciones se pueden descomponer de forma jerárquica y se asigna un plan a cada una de las subtareas que aparecen.

Para realizar la descomposición, existen 4 tipos básicos de descomposiciones:

- **Secuencia:** Una tarea se descompone en un conjunto de subtareas que se ejecutan de manera ordenada.
- **Selección:** Una tarea se descompone en un conjunto de subtareas en la que se elige cuál de las subtareas se va a ejecutar.
- **Iteración:** Una tarea se descompone en un conjunto de subtareas que se van a estar ejecutando de manera repetitiva.
- **Tarea unitaria:** Representa una tarea que no puede ser descompuesta en tareas más sencillas.

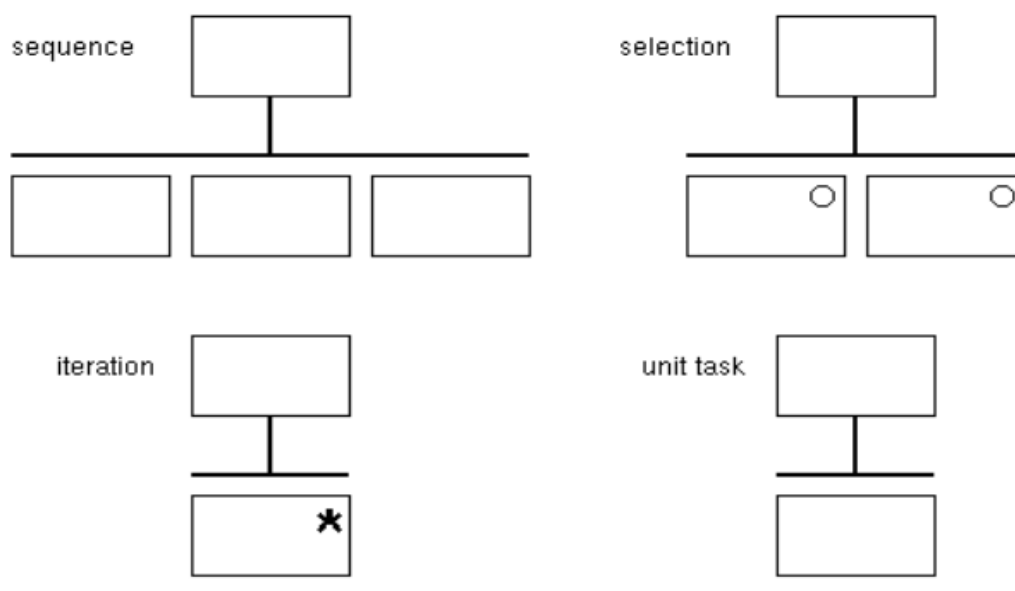


Figura 20: Tipos de descomposiciones de la metodología HTA

El procedimiento a seguir en la metodología HTA (*Hierarchical Task Analysis*) es el siguiente:

- En primer lugar, se realiza una **etapa inicial** en la que se define la tarea principal (objetivo) la cual puede ser dividida en un conjunto de subtareas.
- En segundo lugar, se realiza una **etapa intermedia** en la que se decide el nivel de detalle que se requiere en la descomposición.
- En tercer lugar, se realiza una **etapa final** en la que se revisa y evalúa la secuencia de tareas y subtareas a realizar para alcanzar el objetivo.

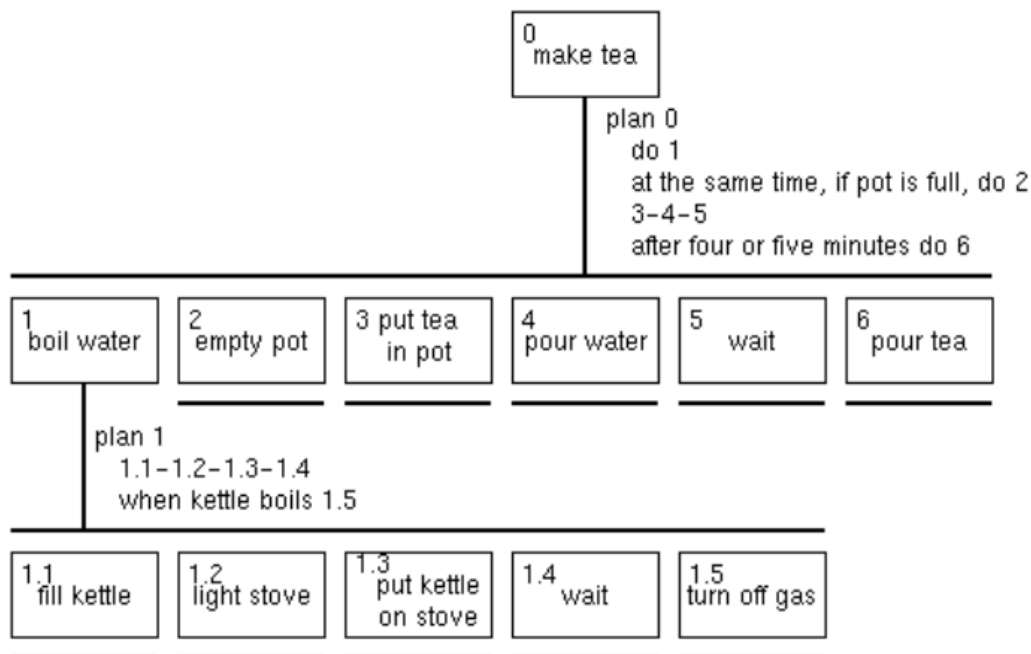


Figura 21: Ejemplo de análisis de tareas utilizando la técnica HTA

## Textos de los procedimientos

La técnica de textos de los procedimientos es una técnica utilizada dentro de la metodología del Diseño Centrado en el Usuario destinada para realizar la definición de los escenarios de uso (subfase del DCU).

En la técnica de textos de los procedimientos, se describe el escenario de uso describiendo pasos a paso las acciones del usuario y las respuestas del sistema. Esta técnica tiene la ventaja que permite elaborar la especificación de los escenarios de uso de manera rápida, sencilla y completa. Además, los textos de los procedimientos utilizan el lenguaje natural por lo que serán comprensibles por todos los usuarios.

## Formularios

La técnica de los formularios es una técnica utilizada dentro de la metodología del Diseño Centrado en el Usuario destinada para realizar la encuesta de contenido (subfase del DCU).

En la técnica de los formularios, se realizan una serie de preguntas sobre el contenido del sistema a desarrollar de manera remota a los usuarios, de manera que el usuario responde a las preguntas en el momento que él decida y tomando el tiempo que él necesite.

Las ventajas de esta técnica son la disminución del efecto Hawthorne debido a la observación, la facilidad para distribuir los formularios a distintos usuarios y la flexibilidad a la hora de elaborar las preguntas y cuestiones en el formulario.

## Técnica del conductor

La técnica del conductor es una técnica utilizada dentro de la metodología del Diseño Centrado en el Usuario destinada para realizar las pruebas de usuario (subfase del DCU).

En la técnica del conductor, el evaluador guía al usuario en su interacción con el prototipo con el objetivo de que el usuario realice unas tareas determinadas. A través de esta técnica, el evaluador puede observar la facilidad o dificultad del usuario para realizar las distintas tareas además de proporcionar realimentación al usuario.

## Herramientas

### Visual Studio 2019

Visual Studio es un entorno de desarrollo integrado (IDE) que permite la edición, compilación, depuración e implementación de aplicaciones en distintos lenguajes de programación.

Este entorno de desarrollo incluye herramientas de completado de código y diferentes extensiones que pueden ser descargadas directamente a través de la tienda de Visual Studio.

Ventajas que proporciona Visual Studio como entorno de desarrollo:

- **Compatibilidad con distintos lenguajes de programación:** Permite el desarrollo de aplicaciones en C++, C#, JavaScript, Python, etc.
- **Instalador selectivo:** El instalador permite descargar solo los módulos necesarios para realizar un determinado cometido.
- **Portabilidad:** Permite trasladar proyectos de una máquina a otra con una gran facilidad.
- **Facilidad en la edición:** La programación en el entorno de Visual Studio es rápida, cómoda, sencilla e intuitiva.

### Visual Paradigm

Visual Paradigm es una herramienta CASE de desarrollo de software que permite la creación de modelos y diseños de software. Además, permite la generación de código fuente en varios lenguajes de programación.

La herramienta Visual Paradigm se caracteriza por:

- Permitir la creación y edición de múltiples tipos de diagramas (diagramas de casos de uso, diagramas de clases, diagramas de secuencia, diagramas de despliegue, etc). En definitiva, esta herramienta permite crear todos los modelos generados dentro del Proceso Unificado (UP).
- La facilidad en la creación de diagramas de distintos tipos debido a la sencillez de su interfaz gráfica.
- La facilidad en la exportación de los distintos diagramas en distintos formatos (PNG, HTML, XML, PDF, etc).

## Adobe XD

Adobe XD es una herramienta destinada a la elaboración de prototipos de interfaces de usuario.

La herramienta Adobe XD cuenta con las siguientes características:

- Permite la creación de prototipos para cualquier dispositivo ya sea móvil u ordenador.
- Permite la creación de prototipos interactivos de forma que se pueden incluir transiciones sencillas entre las distintas pantallas del prototipo.
- Permite compartir los prototipos realizados con otros usuarios de manera sencilla a través de enlaces, lo cual facilita en gran medida su distribución para realizar las pruebas de usuario.
- Permite incluir componentes adicionales que incluyen funcionalidades interesantes que se pueden aplicar en la creación de los prototipos.

## Windows Presentation Foundation (WPF)

Windows Presentation Foundation (WPF) es una herramienta que permite el desarrollo de interfaces gráficas en Windows tomando características de las aplicaciones Windows y las aplicaciones web.

WPF presenta las siguientes características:

- Utiliza el modelo MVVM el cual permite separar la lógica de negocio de la aplicación de su representación visual.
- Utiliza el lenguaje de programación XAML para implementar la parte visual de la aplicación mientras que utiliza el lenguaje de programación C# para implementar la lógica de negocio de la aplicación.
- Incluye una gran variedad de controles de visualización (cajas, botones, tablas, etc.), la mayoría heredadas de Windows.
- Se basa en el uso y gestión de eventos. Las distintas acciones o eventos que realizan los usuarios son los que desencadenan la ejecución de ciertas funcionalidades.

## Visual\_HDLC

Visual\_HDLC es una herramienta que implementa el protocolo HDLC en el modo balanceado asíncrono utilizando la variante LAP-B. Aunque implementa una funcionalidad parecida a la funcionalidad que incluirá el sistema a desarrollar en este proyecto, se utilizará como referencia para definir distintos casos de prueba y confirmar como debe ser el comportamiento del sistema a desarrollar en ciertas situaciones.

## Adobe Color

Adobe Color es una herramienta que permite a los diseñadores elegir los colores adecuados para los diferentes componentes y elementos de una interfaz de usuario.

Las funcionalidades más importantes que proporciona Adobe Color son:

- Una rueda de color interactiva en la que se puede seleccionar los colores con una gran precisión.
- Un mecanismo para aplicar distintas reglas de la teoría de colores para encontrar combinaciones de colores armoniosas.
- Una herramienta de extracción de colores que permite extraer los colores más relevantes de una imagen.
- Una herramienta de accesibilidad que permite obtener el contraste entre el color de un texto y el color de fondo en el que se encuentra el texto. De esta manera, se pueden buscar combinaciones de colores que proporcionen buen contraste y legibilidad.

## EZ Estimate

EZ Estimate es una herramienta CASE que permite estimar el esfuerzo de un proyecto a partir de los puntos de casos de uso. Esta herramienta utiliza factores de complejidad técnica (TCF), factores de complejidad del entorno (ECF), la complejidad de los casos de uso (UUCW) y la complejidad de los actores (UAW) para determinar la estimación del esfuerzo del proyecto.

## Windows Project

Windows Project es una herramienta CASE que permite realizar la planificación temporal de un proyecto, así como el control y el seguimiento de dicha planificación. Una de las grandes ventajas de esta herramienta es que incluye la posibilidad de generar diagramas de Gantt, los cuales son imprescindibles en el desarrollo de este proyecto.

## Google Forms

Google Forms es una de las múltiples herramientas de las que dispone Google, la cual está orientada a la creación de formularios y encuestas personalizadas de manera online y gratuita. Entre sus funcionalidades destacadas se encuentra la posibilidad de compartir rápidamente las encuestas y formularios con otros usuarios y la facilidad en la visualización de los resultados de una encuesta o formulario.

## Lucidchart

Lucidchart es una herramienta que permite la elaboración y edición de diagramas de cualquier tipo (diagramas de flujo, diagramas de procesos, esquemas, etc.). Lucidchart se caracteriza por su gran facilidad para arrastrar distintos componentes por la pantalla y por la fácil exportación y almacenamiento de los distintos diagramas realizados. Útil para realizar esquemas o *wireframes*.

## Draw.io

Draw.io es una herramienta gratuita que permite la elaboración y edición de diagramas de cualquier tipo (diagramas de flujo, diagramas de procesos, esquemas, etc.). Draw.io se caracteriza por su gran facilidad para arrastrar distintos componentes por la pantalla y por la fácil exportación y almacenamiento de los distintos diagramas realizados. Útil para la elaboración del análisis de tareas o para la elaboración de mapas de sitio.

Presenta las mismas ventajas que Lucidchart, con la diferencia de que Draw.io permite al usuario elaborar diagramas ilimitados mientras que Lucidchart solo permite crear 3 diagramas distintos de manera gratuita.

# Aspectos relevantes del desarrollo

En esta sección del documento, se tratarán los aspectos más interesantes del desarrollo del proyecto, en el que se expondrá el ciclo de vida del proyecto desarrollado, así como los aspectos más relevantes del análisis, el diseño y la implementación.

El objetivo fundamental de esta sección es exponer los distintos procedimientos utilizados, así como las decisiones tomadas para llegar a la implementación o solución final junto con la justificación del motivo por el cual se han tomado dichas decisiones.

## Descripción del ciclo de vida

El ciclo de vida del software hace referencia al conjunto de etapas realizadas durante el desarrollo de un sistema software. Este conjunto de etapas será diferente en función del tipo de metodología utilizado.

En el desarrollo de este proyecto, se utilizará el Proceso Unificado (UP) el cual proporciona un marco de trabajo en el que se definen las distintas etapas a realizar durante el desarrollo del proyecto.

Dentro del Proceso Unificado existen 4 fases principales (Inicio, Elaboración, Construcción y Transición) donde cada etapa contará con una serie de iteraciones donde en cada iteración se producirá un incremento en el desarrollo del proyecto. En cada iteración, las tareas a realizar se agrupan en una serie de disciplinas. Para el desarrollo de este proyecto, se han considerado las siguientes disciplinas:

- **Modelado de negocio:** En esta disciplina, se agrupan las tareas relacionadas con actividades de negocio relacionadas con el proyecto a desarrollar, así como las tareas de investigación de información relacionada con la temática del proyecto a desarrollar.
- **Requisitos:** En esta disciplina, se agrupan las tareas relacionadas con la obtención y descripción de los requerimientos del sistema a desarrollar, así como la captura de las necesidades de los usuarios y la definición de objetivos que debe cumplir el sistema.
- **Análisis:** En esta disciplina, se agrupan las tareas relacionadas con el análisis tanto de la estructura como la funcionalidad del sistema a desarrollar.
- **Diseño:** En esta disciplina, se agrupan las tareas relacionadas con el diseño del sistema a desarrollar como puede ser la definición de la arquitectura del sistema o la especificación completa de la implementación de las distintas funcionalidades del sistema.
- **Implementación:** En esta disciplina, se agrupan las tareas relacionadas con la codificación y la elaboración de software.
- **Pruebas:** En esta disciplina, se agrupan las tareas relacionadas con la realización de distintas pruebas sobre el sistema ya desarrollado e implementado.

En el caso particular de este proyecto, se plantea la división del desarrollo en 8 iteraciones las cuales estarán repartidas en las distintas fases del Proceso Unificado (Inicio, Elaboración, Construcción y Transición) y donde cada iteración supondrá un incremento en el desarrollo del sistema.

A continuación, se explicará con mayor detalle las actividades desarrolladas en cada una de las iteraciones:

- **Iteración 1:** Esta primera iteración tendrá lugar en la fase de Inicio. Esta iteración tendrá como objetivo definir la base en la cual se fundamentará el desarrollo del proyecto. Entre otras cosas, se realizarán investigaciones sobre el funcionamiento del protocolo HDLC y sobre el funcionamiento de otros simuladores de protocolos de enlace. También se definirán en esta iteración los objetivos que deberá cumplir el proyecto, así como las distintas funcionalidades que debe incluir el sistema. Por otro lado, se planteará un boceto de la arquitectura inicial y se preparará el entorno de desarrollo en el cual se elaborará el proyecto en cuestión.
- **Iteración 2:** Esta segunda iteración tendrá lugar en la fase de Elaboración. En esta iteración se buscará profundizar en la base definida en la fase anterior. Entre otras cosas, se realizará una investigación sobre modelos de comunicación entre procesos y se identificarán los actores, paquetes, requisitos no funcionales y de información. Por otro lado, se definirán las distintas pantallas o ventanas que incluirá la opción a desarrollar. También se realizarán pruebas con entornos similares para obtener información sobre cómo se desea desarrollar el sistema.
- **Iteración 3:** Esta tercera iteración tendrá lugar en la fase de Elaboración. En esta iteración se comenzará a definir formalmente ciertas funcionalidades del sistema. Por un lado, se realizará una investigación sobre creación y manejo de hilos en el entorno de desarrollo y se definirá las funcionalidades relacionadas con la configuración y el envío de tramas. Por otro lado, se profundizará en la estructura del análisis y se tomarán las primeras decisiones de diseño las cuales estarán relacionadas con ciertos aspectos de la interfaz de usuario. También se implementará un primer programa en el que se implemente un mecanismo de comunicación entre 2 procesos y se definirán casos de prueba.
- **Iteración 4:** Esta cuarta iteración tendrá lugar en la fase de Construcción. En esta iteración se definirán el resto de las funcionalidades del sistema y el análisis cobrará una gran importancia. Por un lado, se identificarán los riesgos principales del proyecto y se definirán las funcionalidades relacionadas con la recepción de tramas, la gestión de capturas de tráfico y la gestión de *timeouts*. Por otro lado, se analizará en mayor profundidad las funcionalidades relacionadas con la configuración y el envío de tramas y se profundizará en la estructura del análisis. También se profundizará en el diseño de la arquitectura del sistema, así como en el diseño de la interfaz gráfica del sistema. Por último, se implementará un programa en el que se implementen varios hilos que realicen tareas simples de manera concurrente y se elaborará una estrategia de pruebas que incluya los casos de prueba identificados anteriormente.



- **Iteración 5:** Esta quinta iteración tendrá lugar en la fase de Construcción. En esta iteración se diseñará gran parte del sistema. Por un lado, se priorizarán los riesgos del proyecto identificados anteriormente, se identificarán los escenarios de uso del sistema y se elaborará el diagrama de casos de uso que organizará la funcionalidad del sistema. Por otro lado, se analizará en mayor profundidad las funcionalidades relacionadas con la recepción de tramas, la gestión de capturas de tráfico y la gestión de *timeouts*. También se diseñará la interacción o secuencia de las funcionalidades relacionadas con la configuración y el envío de tramas. Por último, se implementará un prototipo interactivo completo (no funcional) y se definirán las pruebas de integración a realizar posteriormente.
- **Iteración 6:** Esta sexta iteración tendrá lugar en la fase de Construcción. En esta iteración se realizará el grueso del diseño del sistema y se implementarán ciertas funcionalidades del sistema. Por un lado, se revisará con el cliente los progresos realizados, se elaborará la matriz de rastreabilidad que relacionará los distintos requisitos del sistema y se agruparán las clases de análisis en los paquetes identificados. Por otro lado, se diseñará la interacción o secuencia de las funcionalidades relacionadas con la recepción de tramas, la gestión de capturas de tráfico y la gestión de *timeouts*. También se implementará la interfaz gráfica de la aplicación, así como las funcionalidades relacionadas con la configuración y el envío de tramas. Por último, se elaborarán pruebas de usabilidad sobre el prototipo interactivo realizado anteriormente.
- **Iteración 7:** Esta séptima iteración tendrá lugar en la fase de Construcción. En esta iteración se implementarán el resto de las funcionalidades del sistema y se realizarán las primeras pruebas funcionales. Por un lado, se investigarán posibles aplicaciones del proyecto y se realizará una revisión de los requisitos. Por otro lado, se elaborará una propuesta de arquitectura del sistema y se realizará la distribución física del sistema. Se implementarán las funcionalidades relacionadas con la recepción de tramas, la gestión de capturas de tráfico y la gestión de *timeouts*. Por último, se realizarán pruebas funcionales sobre las funcionalidades de la aplicación relacionadas con la configuración y el envío de tramas
- **Iteración 8:** Esta octava iteración tendrá lugar en la fase de Transición. En esta iteración se implementarán el resto de las funcionalidades del sistema y se realizarán las pruebas pertinentes sobre el sistema ya desarrollado. Por un lado, se investigarán posibles vías de mercado para la aplicación y se contemplarán y analizarán posibles nuevas funcionalidades a incluir. Por otro lado, se realizará una descripción de la arquitectura realizada y se integrará todo el sistema. Por último, se realizarán pruebas funcionales sobre las funcionalidades de la aplicación relacionadas con la recepción de tramas, la gestión de capturas de tráfico y la gestión de *timeouts* y se realizarán pruebas de integración.

Una vez se han explicado las distintas tareas a realizar dentro de cada una de las interacciones, se explicará con mayor grado de detalle las tareas realizadas en cada una de las 6 disciplinas.

## Modelado de negocio

Esta disciplina, se centra en labores de investigación sobre información relacionada con la temática del proyecto a desarrollar y en actividades de negocio relacionadas con el proyecto a desarrollar.

Las principales actividades o tareas realizadas en la disciplina de modelado de negocio son las siguientes:

### Investigación sobre el funcionamiento teórico del protocolo HDLC

Para el desarrollo de cualquier proyecto es muy importante realizar una investigación sobre los conceptos teóricos básicos a tratar en el proyecto. En el caso particular de este proyecto, esto es muy importante ya que la temática principal del proyecto es el protocolo HDLC, el cual cuenta con unos conceptos teóricos de cierta dificultad.

De esta manera, para poder realizar un simulador que implemente el protocolo HDLC será imprescindible tener bien definidos los conceptos básicos sobre el funcionamiento del protocolo HDLC. Dichos conceptos básicos del protocolo HDLC se explican con detalle en el apartado de [Conceptos Teóricos](#).

### Investigación de proyectos similares existentes

Para el desarrollo de cualquier proyecto es muy importante realizar una investigación sobre proyectos relacionados que traten de la misma temática o una temática similar. Esta investigación permite obtener información sobre cómo otros programas implementan ciertas funcionalidades y permite obtener ideas sobre nuevas funcionalidades que no se encuentran implementadas en otros programas similares.

En el caso particular de la simulación del protocolo HDLC, existe una herramienta llamada Visual\_HDLC la cual simula la implementación del protocolo HDLC en el modo balanceado asíncrono (ABM). Dicha herramienta cuenta con una interfaz de usuario algo anticuada y algunos errores en el funcionamiento, por lo que uno de los objetivos de este proyecto es corregir estas debilidades. En el apartado de [Trabajos Relacionados](#) se explica con mayor detalle el funcionamiento de Visual\_HDLC.

### Investigación sobre técnicas de comunicación entre procesos

Para la simulación del protocolo HDLC, es necesario disponer de dos estaciones que se encuentren físicamente conectadas a través de un medio físico. En este caso, cada estación será un proceso por lo que es necesario encontrar un mecanismo que nos permita comunicarlos.

Se investigaron múltiples mecanismos de comunicación entre procesos. Los más relevantes son:

- **Tuberías:** Mecanismo que permite la comunicación unidireccional y local entre 2 procesos ubicados en la misma máquina.
- **Sockets:** Mecanismo que permite la comunicación bidireccional entre 2 procesos ubicados o bien en la misma máquina o bien en máquinas distintas.
- **Memoria compartida:** Mecanismo que permite la comunicación unidireccional y local entre 2 procesos ubicados en la misma máquina. Requiere de mecanismos de control de acceso para evitar que ambos procesos accedan al mismo tiempo a la zona de memoria compartida.

Tras realizar la investigación, se tomó la decisión de utilizar las tuberías como mecanismos de comunicación entre procesos debido a las siguientes razones:

- **Sencillez:** Las tuberías permiten comunicar a dos procesos de manera sencilla ya que solo es necesario crear la tubería en un extremo y conectarse a dicha tubería desde el otro extremo.
- **Eficiencia:** No consume recursos de red (sockets) o memoria compartida lo que permite una comunicación rápida entre procesos.
- **Sincronización:** Las tuberías cuentan con mecanismos de sincronización automática, evitando posibles problemas de concurrencia.
- **Validez:** Al tratarse de un simulador del protocolo HDLC, los procesos o estaciones se encontrarán en la misma máquina por lo que las tuberías es un mecanismo válido para comunicar ambos procesos.
- **Interoperabilidad:** Las tuberías pueden utilizarse en una gran variedad de lenguajes de programación. Es posible utilizar tuberías en el lenguaje C#, el cual será utilizado como lenguaje de desarrollo.

De esta manera, se creará un proyecto de WPF en el cual se realizará una conexión entre dos procesos a través de una tubería creada para tal efecto. Es importante tener en cuenta que la comunicación a través de tuberías es unidireccional, por lo que la primera versión del proyecto contará con un esquema de comunicación unidireccional donde solo se pueden intercambiar la información en un sentido.

Sin embargo, para realizar la implementación del protocolo HDLC en su modo balanceado asíncrono (ABM) es necesario que ambas estaciones tengan la posibilidad de enviar información al otro extremo en cualquier momento. De esta manera, se creará una segunda versión del proyecto en el que se establecerá una comunicación bidireccional entre los dos procesos a través de la creación de dos tuberías.

## Investigación sobre la creación de hilos en el lenguaje de programación utilizado

Uno de los aspectos más importantes en el desarrollo de ciertos proyectos, es la gestión de la concurrencia y la realización de varias tareas de manera paralela. En el caso de la simulación del protocolo HDLC, la gestión de la concurrencia es ciertamente importante ya que dentro del protocolo HDLC se realizan distintas labores de manera simultánea.

Por ejemplo, cada estación debe estar permanentemente escuchando por su tubería correspondiente a la espera de recibir nuevas tramas por parte del otro extremo y al mismo tiempo debe encargarse de otras labores, como la detección de eventos del usuario o el envío de tramas. También deben implementarse de manera paralela, un mecanismo de control de cada uno de los *timeouts* existentes.

Para solucionar este tipo de situaciones, se han investigado distintas técnicas sobre el manejo de hilos, procesos y tareas. Las alternativas más relevantes son:

- **Creación de hilos que ejecuten una determinada porción de código:** Se trata de la concepción más extendida de lo que es un hilo. En la creación del hilo, existirá un método *start()* o con un nombre similar que contendrá la porción de código que ejecutará el hilo en cuestión.
- **Uso de tareas:** En esta concepción, los hilos se encargan de realizar una tarea específica donde dicha tarea suele ser la ejecución de un método.
- **Programación asíncrona:** Busca la ejecución de tareas paralelas sin bloquear la ejecución del hilo principal. De esta manera, mientras los hilos secundarios realizan distintas tareas, el hilo principal puede encargarse de otras labores. Este concepto permite al hilo principal gestionar los resultados de una tarea cuando ésta haya finalizado.

Tras realizar dicha investigación, se ha tomado la decisión de utilizar la estrategia de la programación asíncrona para implementar el simulador del protocolo HDLC ya que dicha técnica permite crear distintos hilos secundarios que realicen tareas como la gestión de los *timeouts* o la escucha de la tubería mientras el proceso principal se mantiene activo realizando otro tipo de gestiones como el mantenimiento y actualización de la interfaz gráfica.

Además, esta técnica permite gestionar los resultados de una tarea cuando ésta haya finalizado. Por ejemplo, cuando expire un *timeout* nos interesa gestionar la finalización de dicha tarea para decidir cuáles deben ser las próximas acciones a realizar.

Se creará un proyecto de WPF en el que se realicen varias tareas simples de manera paralela utilizando hilos asíncronos para aprender más sobre dicha técnica de concurrencia.

## Investigación sobre posibles aplicaciones y utilidades del proyecto

Una actividad bastante importante en el desarrollo de cualquier proyecto es la investigación de las posibles utilidades y aplicaciones que puede tener el sistema a desarrollar. De esta manera, se puede obtener información sobre los contextos en los cuales se pueda hacer uso del sistema.

En el caso particular de este proyecto, la utilidad básica del sistema a desarrollar es facilitar el aprendizaje del protocolo HDLC a los alumnos del grado de Ingeniería Informática. Otra utilidad es la evaluación del rendimiento del protocolo HDLC en ciertas situaciones, como puede ser la medición de la eficiencia de utilización del enlace, la capacidad de detección y corrección de errores o la estabilidad en el funcionamiento del protocolo.

## Investigación de posibles vías de mercado

En el desarrollo de cualquier tipo de proyecto, es importante realizar una investigación sobre las posibles vías de mercado para la comercialización del producto. Esto se debe a que la mayoría de los proyectos desarrollados tienen como objetivo compensar los gastos y recursos empleados en el desarrollo del proyecto y generar una serie de beneficios.

En el caso particular de este proyecto, no se contempla ningún tipo de comercialización posible ya que se trata de un proyecto de final de grado orientado al aprendizaje el cual trata una temática bastante poco extendida como es la simulación del protocolo HDLC.

No se considera que exista un conjunto grande de personas que estén dispuestas a pagar una determinada cantidad de dinero por el sistema, por lo que la comercialización del producto final queda totalmente descartada.

## Requisitos

Esta disciplina, se centra en la obtención y descripción de los requisitos que debe incluir el sistema a desarrollar, así como la captura de las necesidades de los usuarios relacionadas con la temática del proyecto a tratar y la definición de los objetivos que debe cumplir el sistema.

Las principales actividades o tareas realizadas en la disciplina de requisitos son las siguientes:

### Reuniones con el cliente

Un aspecto crucial en el desarrollo de cualquier proyecto software, es la obtención de información sobre qué debe incluir el sistema a desarrollar. Para ello, se realizarán distintas reuniones y entrevistas en las cuales el cliente nos expone cómo quiere que sea el sistema a desarrollar.

En el caso particular de este proyecto, al tratarse de un proyecto de final de grado, partiremos de un enunciado básico en el que se planteará de manera resumida las características básicas de un sistema. En nuestro caso dicho sistema se trata de un simulador del protocolo HDLC.

Este enunciado cuenta con escasa información sobre qué debe incluir el sistema a desarrollar por lo que serán necesario realizar distintas reuniones con los tutores del proyecto (que, en este caso, actuarán de clientes) para definir con mayor detalle que debe incluir el sistema a desarrollar.

### Definición de los objetivos del proyecto

En el desarrollo de cualquier proyecto software, se plantean una serie de objetivos que deben cumplirse una vez haya concluido el desarrollo del proyecto. De esta manera, es importante definir de manera adecuada cuáles son los objetivos a cumplir, ya que el desarrollo del proyecto va a estar orientado a la consecución de dichos objetivos.

En el caso particular de este proyecto, los objetivos técnicos están relacionados con la simulación del protocolo HDLC en el modo balanceado asíncrono, la realización de tareas específicas del protocolo HDLC (envío/recepción de tramas, *timeouts*, respuestas automáticas, etc.) y la inclusión de mecanismos de ayuda. Estos objetivos se definieron de manera clara en el enunciado del proyecto. También se han definido una serie de objetivos personales los cuales no son críticos para el desarrollo del proyecto.

## Definición del ámbito del sistema

Un aspecto importante que suele ser pasado por alto en el desarrollo de un proyecto software, es definir el ámbito del sistema y definir la audiencia a la que va dirigida. De esta manera, es importante definir los límites del sistema y definir a qué tipo de usuarios va dirigido el sistema. Si no se tiene esto en cuenta, los resultados finales del desarrollo puede que no sean los esperados.

En el caso particular de este proyecto, el ámbito de este proyecto es la simulación del protocolo HDLC en el modo balanceado asíncrono en su variante LAP-M, con todo lo que ello supone. Con respecto a la definición de la audiencia, el sistema va dirigido a usuarios que tengan conocimiento sobre el funcionamiento del protocolo HDLC o bien dispongan del conocimiento y las bases necesarias para poder aprender y comprender el funcionamiento del protocolo con facilidad.

## Identificación de actores

En la identificación de actores, se identifican los distintos tipos de usuarios que van a interactuar con el sistema a desarrollar, con el objetivo de definir las funcionalidades a las que va a tener acceso cada tipo de usuario identificado.

En el caso particular de este proyecto, se identifica un actor principal llamado Usuario el cual tendrá acceso a todas las funcionalidades del sistema. No tiene sentido definir distintas categorías de usuario en este proyecto ya que se trata de un entorno de simulación en el que desea que todos los usuarios puedan acceder a todas las funcionalidades. También se identifica un actor sistema que responderá de manera automática en ciertas situaciones.

## Identificación de requisitos no funcionales

Los requisitos no funcionales, aunque no traten de las funcionalidades del sistema, son críticos para el desarrollo de cualquier proyecto software, ya que el incumplimiento de alguno de estos requisitos afectará directamente a la calidad del sistema desarrollado.

En el caso particular de este proyecto, se han identificado 11 requisitos no funcionales donde la mayoría de estos requisitos no funcionales identificados están relacionados con algún comportamiento del producto (fiabilidad, usabilidad, etc.).

En la siguiente tabla, se muestra un resumen de los requisitos no funcionales identificados en el desarrollo del proyecto:

ID	Nombre
RNF-01	<i>Interfaz</i>
RNF-02	<i>Usabilidad</i>
RNF-03	<i>Fiabilidad</i>
RNF-04	<i>Eficiencia</i>
RNF-05	<i>Mantenibilidad</i>
RNF-06	<i>Productividad</i>
RNF-07	<i>Rendimiento</i>
RNF-08	<i>Manual de usuario</i>
RNF-09	<i>Manejo de errores</i>
RNF-10	<i>Entorno de desarrollo</i>
RNF-11	<i>Estándar de proceso</i>

Tabla 1: Resumen de los requisitos no funcionales del proyecto

## Identificación de requisitos de información

Los requisitos de información son utilizados para definir la información que va a almacenar y gestionar el sistema, así como las posibles restricciones que debe cumplir la información almacenada.

En el caso particular de este sistema, las necesidades de almacenamiento de información no son muy grandes. Aun así, se deberá almacenar cierta información sobre la estación, sobre el contenido de las tramas intercambiadas y sobre la configuración de la estación (protocolo, modo de trabajo y canal de transmisión). Por otro lado, parámetros como el número de secuencia, número de trama esperada o el tamaño de la ventana estarán sujetos a ciertas restricciones de numeración.

En la siguiente tabla, se muestra un resumen de los requisitos de información identificados en el desarrollo del proyecto:

ID	Nombre
IRQ-01	<i>Información sobre las tramas</i>
IRQ-02	<i>Información sobre la estación</i>
IRQ-03	<i>Información sobre la configuración del protocolo</i>
IRQ-04	<i>Información sobre el modo de trabajo</i>
IRQ-05	<i>Información sobre la configuración del canal de transmisión</i>
CRQ-01	<i>Limitación en el tamaño de la ventana</i>
CRQ-02	<i>Número de tramas erróneas consecutivas permitidas</i>
CRQ-03	<i>Limitación en el número de secuencia (NS)</i>
CRQ-04	<i>Limitación en el número de trama esperada (NR)</i>

Tabla 2: Resumen de los requisitos de información del proyecto

## Identificación de paquetes funcionales

En el desarrollo de cualquier proyecto software que siga el marco de trabajo del Proceso Unificado (UP), la primera arquitectura estará basada en la funcionalidad. Para agrupar dicha funcionalidad se utilizarán distintos paquetes funcionales que agruparán los distintos casos de uso del sistema.

En el caso particular de este proyecto, se identifican 3 paquetes funcionales principales. Estos paquetes funcionales son:

- **Gestión del intercambio de tramas**
- **Gestión de las capturas de tráfico**
- **Gestión de la configuración de la estación**

El paquete de **Gestión del intercambio de tramas** incluye una gran parte de la funcionalidad del sistema por lo que se subdivide también en 3 subpaquetes funcionales llamados **Gestión del envío de tramas**, **Gestión de la recepción de tramas** y **Gestión de los Timeouts**.

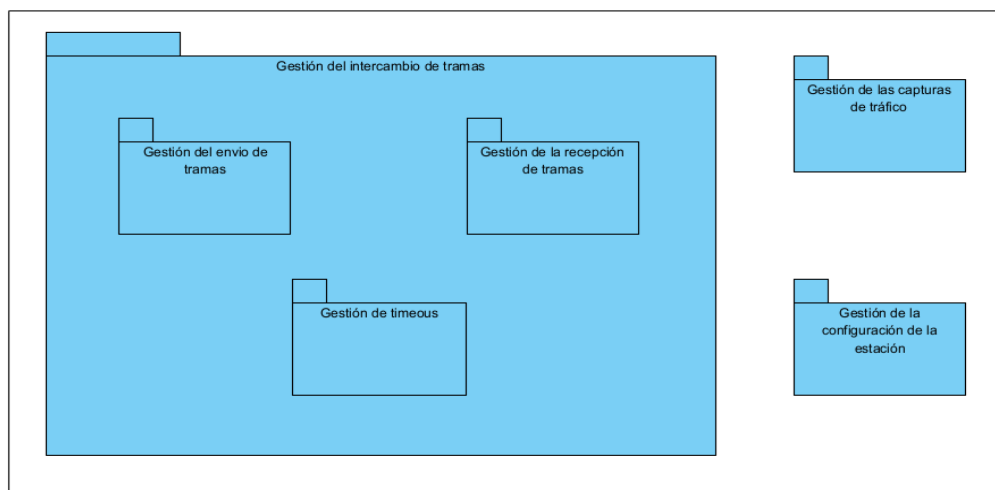


Figura 22: Paquetes funcionales del proyecto

## Identificación de requisitos funcionales o casos de uso

Los requisitos funcionales son utilizados para definir las distintas funcionalidades que debe implementar el sistema a desarrollar. Estos requisitos van a utilizar la información almacenada en el sistema (definida en los requisitos de información) y buscan lograr los distintos objetivos que se han planteado al inicio del proyecto.

En el caso particular del proyecto a desarrollar, se han identificado 37 requisitos funcionales los cuales estarán distribuidos entre los distintos paquetes funcionales identificados previamente. Para organizar la identificación y descripción de los requisitos funcionales, se ha tomado la decisión de identificar todos los requisitos de un mismo paquete funcional antes de pasar al siguiente paquete. Con esto podemos distribuir la identificación y descripción de los requisitos funcionales en varias iteraciones.

En la siguiente tabla, se muestra un resumen de los requisitos funcionales identificados en el desarrollo del proyecto:



ID	Nombre
UC-01	Consultar información sobre la configuración del protocolo
UC-02	Modificar información sobre la configuración del protocolo
UC-03	Consultar información sobre el modo de trabajo de la estación
UC-04	Modificar información sobre el modo de trabajo de la estación
UC-05	Consultar información sobre la configuración del canal
UC-06	Modificar información sobre la configuración del canal
UC-07	Establecer conexión física
UC-08	Finalizar el establecimiento de la conexión física
UC-09	Enviar trama de información
UC-10	Enviar trama de receptor preparado (RR)
UC-11	Enviar trama de receptor no preparado (RNR)
UC-12	Enviar trama de rechazo (REJ)
UC-13	Enviar trama de rechazo selectivo (SREJ)
UC-14	Enviar trama de petición de conexión (SABM)
UC-15	Enviar trama de petición de desconexión (DISC)
UC-16	Enviar trama de asentimiento no numerado (UA)
UC-17	Enviar trama de modo desconectado (DM)
UC-18	Enviar trama de rechazo de trama (FRMR)
UC-19	Representar gráficamente el envío de una trama
UC-20	Ver detalle de la trama
UC-21	Guardar captura de tráfico
UC-22	Cargar captura de tráfico
UC-23	Implementar timeout ante COMMAND
UC-24	Implementar timeout ante trama I
UC-25	Implementar timeout ante REQUEST
UC-26	Recibir trama de información
UC-27	Recibir trama de receptor preparado (RR)
UC-28	Recibir trama de receptor no preparado (RNR)
UC-29	Recibir trama de rechazo (REJ)
UC-30	Recibir trama de rechazo selectivo (SREJ)
UC-31	Recibir trama de petición de conexión (SABM)
UC-32	Recibir trama de petición de desconexión (DISC)
UC-33	Recibir trama de asentimiento no numerado (UA)
UC-34	Recibir trama de modo desconectado (DM)
UC-35	Recibir trama de rechazo de trama (FRMR)
UC-36	Recibir trama errónea
UC-37	Representar gráficamente la recepción de una trama

Tabla 3: Resumen de los requisitos funcionales del proyecto

## Realización de encuestas de contenido

Una encuesta de contenido tiene como objetivo definir cuáles van a ser los distintos contenidos que va a incluir el sistema a desarrollar, así como la interacción con este contenido y su significado dentro del sistema. En este proceso, se busca obtener de las posibles necesidades de los usuarios y cómo la aplicación desarrollada y la interacción con la interfaz puede satisfacer estas necesidades de los usuarios.

En el caso particular de este proyecto, se han utilizado formularios para obtener y descubrir el comportamiento y las necesidades de los usuarios. Estos formularios contienen ciertas cuestiones específicas sobre la simulación del protocolo HDLC las cuales están centradas en aspectos relacionados con la interacción con el usuario. Posteriormente, se evaluarán los resultados de dichas encuestas de contenido para determinar qué aspectos debe incluir la interacción con el usuario del simulador.

Se ha decidido utilizar formularios para realizar encuestas de contenido debido a la posibilidad de incluir preguntas y cuestiones abiertas y a la reducción del efecto Hawthorne, es decir, la dirección del comportamiento del usuario debido a la observación.

## Identificación de escenarios de uso

Un escenario de uso hace referencia a un contexto de utilización de una funcionalidad del sistema por parte del usuario. Los escenarios de uso suelen ser confundidos con facilidad con los casos de uso. La diferencia entre ambos radica en el grado de formalidad donde los casos de uso utilizan una notación formal mientras los escenarios de uso tienen un concepto más amplio e informal, abierto a la interpretación.

En el caso particular de este proyecto, se definirán los escenarios de uso utilizando una notación llamada textos de los procedimientos. Esta notación permite especificar los escenarios de uso de manera rápida, sencilla y completa, ya que usa lenguaje natural para describir la funcionalidad del sistema.

Para la identificación de los escenarios de uso, se tendrán en cuenta los casos de uso previamente identificados. En la siguiente tabla, se muestra un resumen de los escenarios de uso identificados en el desarrollo del proyecto:

ID	Nombre
Escenario de uso 1	<i>Consultar información sobre la configuración del protocolo</i>
Escenario de uso 2	<i>Modificar información sobre la configuración del protocolo</i>
Escenario de uso 3	<i>Consultar información sobre el modo de trabajo de la estación</i>
Escenario de uso 4	<i>Modificar información sobre el modo de trabajo de la estación</i>
Escenario de uso 5	<i>Consultar información sobre la configuración del canal</i>
Escenario de uso 6	<i>Modificar información sobre la configuración del canal</i>
Escenario de uso 7	<i>Establecer conexión física</i>
Escenario de uso 8	<i>Finalizar el establecimiento de la conexión física</i>
Escenario de uso 9	<i>Enviar trama de información</i>
Escenario de uso 10	<i>Enviar trama de receptor preparado (RR)</i>
Escenario de uso 11	<i>Enviar trama de receptor no preparado (RNR)</i>
Escenario de uso 12	<i>Enviar trama de rechazo (REJ)</i>
Escenario de uso 13	<i>Enviar trama de rechazo selectivo (SREJ)</i>
Escenario de uso 14	<i>Enviar trama de petición de conexión (SABM)</i>
Escenario de uso 15	<i>Enviar trama de petición de desconexión (DISC)</i>
Escenario de uso 16	<i>Enviar trama de asentimiento no numerado (UA)</i>
Escenario de uso 17	<i>Enviar trama de modo desconectado (DM)</i>
Escenario de uso 18	<i>Enviar trama de rechazo de trama (FRMR)</i>
Escenario de uso 19	<i>Ver detalle de la trama</i>
Escenario de uso 20	<i>Guardar captura de tráfico</i>
Escenario de uso 21	<i>Cargar captura de tráfico</i>
Escenario de uso 22	<i>Implementar timeout ante COMMAND</i>
Escenario de uso 23	<i>Implementar timeout ante trama I</i>
Escenario de uso 24	<i>Implementar timeout ante REQUEST</i>
Escenario de uso 25	<i>Recibir trama de información</i>
Escenario de uso 26	<i>Recibir trama de receptor preparado (RR)</i>
Escenario de uso 27	<i>Recibir trama de receptor no preparado (RNR)</i>
Escenario de uso 28	<i>Recibir trama de rechazo (REJ)</i>
Escenario de uso 29	<i>Recibir trama de rechazo selectivo (SREJ)</i>
Escenario de uso 30	<i>Recibir trama de petición de conexión (SABM)</i>
Escenario de uso 31	<i>Recibir trama de petición de desconexión (DISC)</i>
Escenario de uso 32	<i>Recibir trama de asentimiento no numerado (UA)</i>
Escenario de uso 33	<i>Recibir trama de modo desconectado (DM)</i>
Escenario de uso 34	<i>Recibir trama de rechazo de trama (FRMR)</i>
Escenario de uso 35	<i>Recibir trama errónea</i>

Tabla 4: Resumen de los escenarios de uso del proyecto

Elaboración del diagrama de casos de uso

En el desarrollo de cualquier proyecto software que siga el marco de trabajo del Proceso Unificado (UP), la primera arquitectura estará basada en la funcionalidad. Esta primera arquitectura se implementará a través de los diagramas de casos de uso, en la cual se establece una relación entre las distintas funcionalidades identificadas y los distintos actores que van a actuar con estas funcionalidades.

En el caso particular del sistema a desarrollar, el diagrama de casos de uso contará con 2 actores y 37 casos de uso, donde los casos de uso estarán repartidos entre los distintos paquetes funcionales identificados anteriormente. La mayoría de los casos de uso podrán ser accedidos desde el actor usuario, aunque ciertas funcionalidades se desencadenarán por acción directa del sistema.

## Elaboración de la matriz de rastreabilidad

En cualquier proyecto software, existen relaciones entre los distintos objetivos, requisitos funcionales y requisitos de información identificados. La matriz de rastreabilidad permite establecer de manera gráfica y directa dichas relaciones.

En el caso particular de este proyecto, se ha planteado la elaboración de la matriz de trazabilidad siguiendo un formato especial. En dicho formato, se busca establecer una relación entre los requisitos por un lado y los objetivos y requisitos de información por el otro lado. De esta manera, los objetivos y requisitos de información se encontrarán en las columnas de la matriz de rastreabilidad mientras que los casos de uso se encontrarán en las filas de la matriz de rastreabilidad.

En la siguiente imagen, se adjunta un ejemplo del formato de la matriz de rastreabilidad utilizado:

TRM	OBJ	OBJ	OBJ	OBJ	OBJ	OBJ	OBJ	IRQ	IRQ	IRQ	IRQ	IRQ	IRQ	IRQ	IRQ	IRQ	IRQ
0002	000	000	000	000	000	000	000	000	000	000	000	000	000	000	001	001	001
	1	2	3	4	5	8	9	2	3	4	5	6	7	8	0	1	2
UC-0001	<input checked="" type="checkbox"/>	-	-	-	-	-	-	-	-	-	-	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	-	-	-
UC-0002	<input checked="" type="checkbox"/>	-	-	-	-	-	<input checked="" type="checkbox"/>	-	-	-	-	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	-	-	-
UC-0003	<input checked="" type="checkbox"/>	-	-	-	-	-	-	-	-	-	-	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	-	-	-
UC-0004	<input checked="" type="checkbox"/>	-	-	-	-	-	-	-	-	-	-	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	-	-	-
UC-0005	<input checked="" type="checkbox"/>	-	-	-	-	-	-	-	-	-	-	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	-	-	-
UC-0006	<input checked="" type="checkbox"/>	-	-	-	-	-	-	-	-	-	-	-	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	-	-	-

Figura 23: Formato matriz de rastreabilidad utilizado

## Análisis

Esta disciplina, se centra en profundizar en el nivel de detalle tanto de la estructura como la funcionalidad del sistema desarrollado. Se analizarán los requisitos identificados en la disciplina de Requisitos y se planteará de manera más detallada la estructura del sistema a desarrollar y la secuencia de interacción de las distintas funcionalidades a implementar.

Las principales actividades o tareas realizadas en la disciplina de análisis son las siguientes:

### Identificación de clases entidad del análisis

En el desarrollo de cualquier proyecto software, se tienen ciertas necesidades de información. Estas necesidades de información fueron reflejadas en la disciplina de Requisitos a través de los requisitos de información. Estos requisitos evolucionarán en la disciplina de Análisis, convirtiéndose en clases entidad.

En el caso particular del proyecto a desarrollar, se identifican 5 clases entidad. Estas clases entidad identificadas son **Estación**, **Trama**, **Protocolo**, **Modo de trabajo** y **Canal de transmisión**. Estas clases entidad han evolucionado directamente a partir de los requisitos de información planteado en la disciplina de Requisitos.

- La clase **Estación** se encargará de almacenar información básica relativa a la estación,
- La clase **Trama** se encargará de almacenar información relativa al contenido de las tramas intercambiadas.
- La clase **Protocolo** se encargará de almacenar información relativa al protocolo de la estación.
- La clase **Modo de trabajo** se encargará de almacenar información relativa al modo de trabajo de la estación.
- La clase **Canal de transmisión** se encargará de almacenar información relativa al canal de transmisión.

### Identificación de paquetes del análisis

En la disciplina de Requisitos se planteó una primera arquitectura basada en la funcionalidad y en el uso de distintos paquetes funcionales para distribuir la funcionalidad del sistema. Estos paquetes funcionales evolucionarán hasta convertirse en paquetes del análisis, permitiendo así una evolución en la arquitectura del sistema a desarrollar.

En el caso particular de este proyecto, se identifican 3 paquetes del análisis principales. Estos paquetes del análisis son:

- **Gestión del intercambio de tramas**
- **Gestión de las capturas de tráfico**
- **Gestión de la configuración de la estación**

El paquete de **Gestión del intercambio de tramas** incluye una gran parte de la funcionalidad del sistema por lo que se subdivide también en 3 subpaquetes llamados **Gestión del envío de tramas**, **Gestión de la recepción de tramas** y **Gestión de los Timeouts**.

Como se puede observar, la evolución de los paquetes funcionales a paquetes de servicio o de análisis ha sido directa y no se ha considerado la inclusión de algún nuevo paquete.

## Elaboración del modelo del dominio

El modelo del dominio es un diagrama que agrupa las clases entidad identificadas anteriormente. Fundamentalmente, en el modelo del dominio se establecen relaciones entre las distintas clases entidad con el objetivo de definir una estructura básica de la información almacenada en el sistema a desarrollar.

En el caso particular de este proyecto, se identifica una relación directa entre la clase **Estación** y la clase **Trama** ya que una estación envía/recibe un conjunto de tramas. Por otro lado, también existe una relación entre la clase **Estación** y las clases **Protocolo**, **Modo de trabajo** y **Canal de transmisión** ya que una estación tiene asociado con una configuración del protocolo, una configuración del modo de trabajo y una configuración del canal de transmisión.

En la siguiente imagen, se adjunta el modelo del dominio del proyecto:

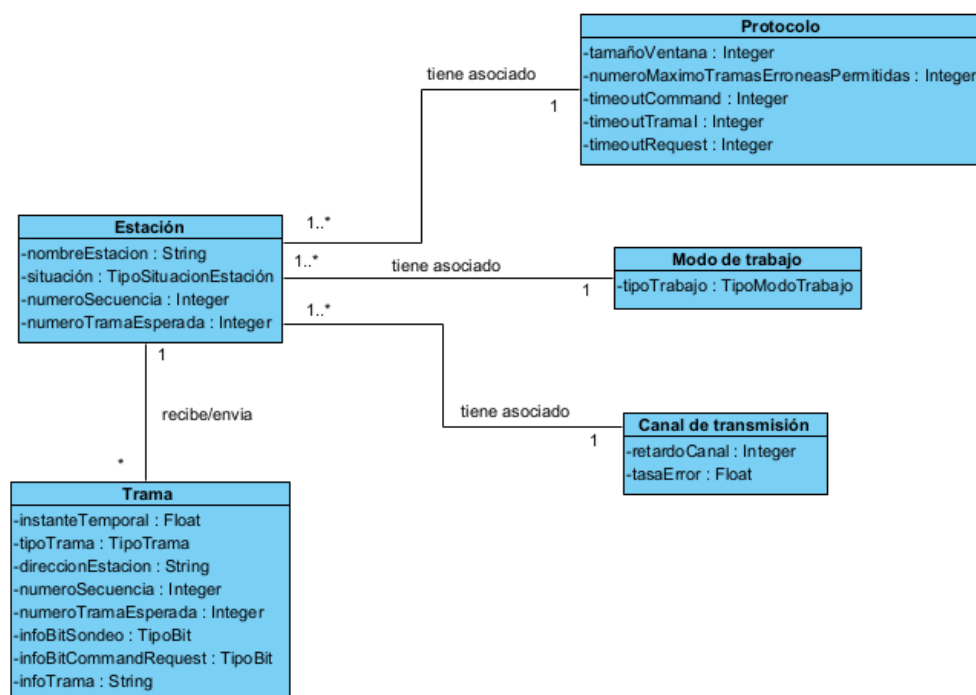


Figura 24: Diagrama de clases o modelo del dominio del proyecto

## Identificación de otras clases del análisis

En la disciplina de análisis, existen 3 tipos de clases: *clases entidad*, *clases de interfaz* y *clases de control*. Las clases entidad fueron identificadas previamente y organizadas en el modelo del dominio. Sin embargo, las clases de interfaz y las clases de control también deben ser identificadas ya que participarán en la interacción del usuario con el sistema.

En el caso particular de este proyecto, se ha considerado la existencia de una clase interfaz y una clase de control asociada a cada clase entidad identificada anteriormente. De esta manera, la clase entidad **Estación** tendrá asociada una clase interfaz llamada **EstaciónUI** y una clase de control llamada **ControlEstación** y así sucesivamente para clase entidad.

En la siguiente tabla, se muestra un resumen de todas las clases del análisis identificadas en el desarrollo del proyecto:

Nombre	Tipo
<b>Estación</b>	Entidad
<b>Trama</b>	Entidad
<b>Protocolo</b>	Entidad
<b>ModoDeTrabajo</b>	Entidad
<b>CanalDeTransmisión</b>	Entidad
<b>EstaciónUI</b>	Interfaz
<b>TramaUI</b>	Interfaz
<b>ProtocoloUI</b>	Interfaz
<b>ModoDeTrabajoUI</b>	Interfaz
<b>CanalDeTransmisiónUI</b>	Interfaz
<b>CanalDeTransmisión</b>	Control
<b>ControlEstación</b>	Control
<b>ControlTrama</b>	Control
<b>ControlProtocolo</b>	Control
<b>ControlModoDeTrabajo</b>	Control

Tabla 5: Resumen de las clases del análisis del proyecto

## Realización de casos de uso a nivel de análisis

La realización de casos de uso a nivel de análisis consiste en una descripción de cómo se ejecutan los casos de uso identificados en la disciplina de Requisitos en función de las clases del análisis previamente identificadas. De esta manera, esta realización se trata de una evolución de la funcionalidad del sistema ya que se incrementa el nivel de detalle.

En el caso particular del proyecto a desarrollar, la realización de los casos de uso a nivel de análisis se ha realizado utilizando las clases del análisis identificadas previamente. Dicha realización de casos de uso a nivel de análisis se realizará a través de los diagramas de secuencia, que es la herramienta que proporciona UML para realizar este cometido.

Las operaciones utilizadas para detallar la secuencia de interacción serán conceptuales y de carácter genérico ya que el objetivo es representar la secuencia de operaciones realizada para implementar una determinada funcionalidad. Para elaborar los diagramas de secuencia correspondientes, se tendrá en cuenta en todo momento, la secuencia de pasos especificada en la disciplina de Requisitos (trazabilidad).

En la siguiente imagen, se adjunta un ejemplo de la realización de un caso de uso a nivel de análisis:

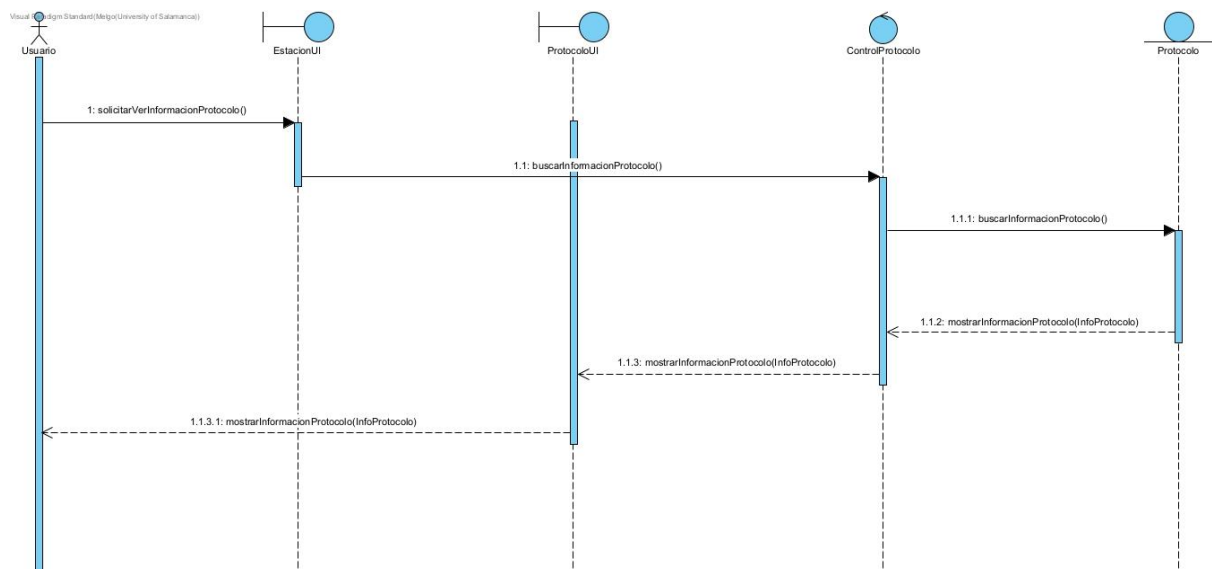


Figura 25: Ejemplo de realización de un caso de uso a nivel de análisis del proyecto

## Análisis de tareas

El análisis de tareas se encarga de analizar la secuencia de tareas o pasos que debe realizar un usuario para lograr un objetivo o conseguir un determinado resultado. Esto es importante, ya que nos va a proporcionar mucha información sobre la interacción del usuario con la interfaz gráfica del sistema

En el caso particular de este proyecto, para realizar este análisis de tareas, se utilizará una técnica llamada HTA (*Hierarchical Task Analysis*) en la cual se realiza una descripción de tareas en términos de operaciones y planes, donde las operaciones se descomponen de manera jerárquica y los planes indican cómo deben ejecutarse las operaciones.

Para realizar el análisis de tareas se emplearán 2 tipos de notaciones distintas. Por una parte, se utilizará una notación en texto donde se especificará de manera textual la descomposición del objetivo en las distintas tareas, así como los planes de ejecución de las tareas. Por otra parte, se utilizará una notación gráfica donde se especificará la misma información, pero utilizando diagramas de bloques.

A continuación, se adjunta un ejemplo del análisis de una tarea, tanto en la notación en texto como la notación gráfica:



## Notación en texto

- 0. Consultar información del protocolo
  - 1. Solicitar información del protocolo
  - 2. Recopilar información del protocolo
  - 3. Mostrar información del protocolo

Plan 0:

hacer 1, 2

cuando la información del protocolo esté lista, hacer 3

## Notación gráfica

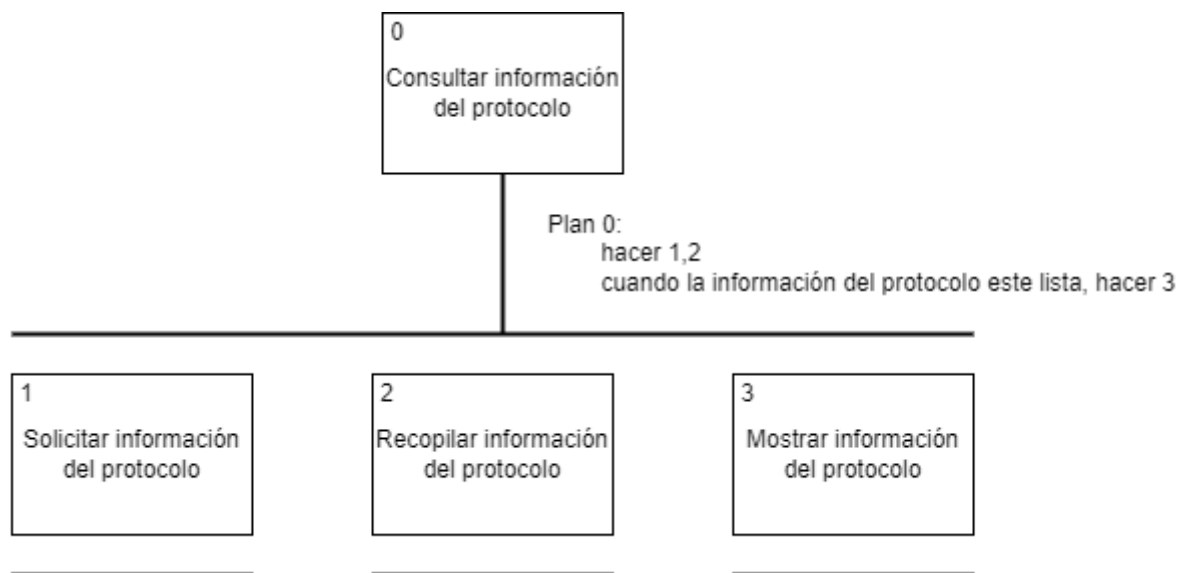


Figura 26: Ejemplo de análisis de una tarea del proyecto

## Elaboración de la propuesta de arquitectura

Una vez se ha evolucionado tanto la estructura como la funcionalidad del sistema a desarrollar, la última tarea a realizar en la disciplina de análisis es la elaboración de una propuesta de arquitectura. Esta propuesta de arquitectura incluirá los paquetes del análisis identificados anteriormente, donde dentro de estos paquetes se agruparán las clases del análisis involucradas en cada paquete del análisis. También se establecerán relaciones entre los distintos paquetes del análisis.

En el caso particular de este proyecto, la propuesta de arquitectura se habrá realizado de la siguiente manera:

En primer lugar, se identificarán las clases del análisis que deben ir dentro de cada paquete. Para ello, se revisarán las realizaciones de casos de usos a nivel de análisis de los casos de uso relacionados con cada paquete y se extraerán las distintas clases del análisis utilizadas en dichos diagramas de secuencia.

En segundo lugar, se establecerán las distintas relaciones existentes entre los paquetes del análisis. Para identificar dichas relaciones se evaluarán las dependencias existentes entre los distintos paquetes.

Diagrama de arquitectura de software que muestra la estructura de un sistema de gestión de recursos humanos. El diagrama está dividido en tres secciones principales:

- Gestión de la configuración de la estación:**
  - Gestión de la configuración de la estación:** Incluye componentes como 'Procesamiento', 'Control de acceso', 'Procesamiento', 'Modulo de trabajo', 'Relacionado', 'Control de acceso', 'Modulo de trabajo', 'Control de acceso', 'Modulo de trabajo', 'Modulo de trabajo'.
  - Gestión de la configuración de la estación:** Incluye componentes como 'Procesamiento', 'Control de acceso', 'Procesamiento', 'Modulo de trabajo', 'Relacionado', 'Control de acceso', 'Modulo de trabajo', 'Control de acceso', 'Modulo de trabajo', 'Modulo de trabajo'.
- Gestión del nivel de la estación:**
  - Gestión del nivel de la estación:** Incluye componentes como 'Procesamiento', 'Control de acceso', 'Procesamiento', 'Modulo de trabajo', 'Relacionado', 'Control de acceso', 'Modulo de trabajo', 'Control de acceso', 'Modulo de trabajo', 'Modulo de trabajo'.
  - Gestión del nivel de la estación:** Incluye componentes como 'Procesamiento', 'Control de acceso', 'Procesamiento', 'Modulo de trabajo', 'Relacionado', 'Control de acceso', 'Modulo de trabajo', 'Control de acceso', 'Modulo de trabajo', 'Modulo de trabajo'.
- Gestión del intercambio de datos:**
  - Gestión del intercambio de datos:** Incluye componentes como 'Procesamiento', 'Control de acceso', 'Procesamiento', 'Modulo de trabajo', 'Relacionado', 'Control de acceso', 'Modulo de trabajo', 'Control de acceso', 'Modulo de trabajo', 'Modulo de trabajo'.
  - Gestión del intercambio de datos:** Incluye componentes como 'Procesamiento', 'Control de acceso', 'Procesamiento', 'Modulo de trabajo', 'Relacionado', 'Control de acceso', 'Modulo de trabajo', 'Control de acceso', 'Modulo de trabajo', 'Modulo de trabajo'.

Las sub-secciones están conectadas por líneas de flujo que indican la dirección del intercambio de datos.

Como se puede observar, la propuesta de arquitectura del proyecto está formada por los distintos paquetes del análisis identificados anteriormente. Cada paquete del análisis incluye en su interior las clases del análisis que tenga asociadas. Entre los distintos paquetes del análisis se han establecido distintas relaciones de dependencia.

# Diseño

Las principales actividades o tareas realizadas en la disciplina de diseño son las siguientes:

El mapa del sitio tiene como objetivo definir las pantallas o mesas de trabajo que van a ser accesibles por los usuarios y definir cómo el usuario puede interaccionar entre estas pantallas estableciendo relaciones entre ellas. De esta manera, se establece una primera arquitectura basada en la representación gráfica de la interfaz.

En el caso particular de este proyecto, se va a identificar las distintas pantallas o mesas de trabajo con las que va a contar el simulador del protocolo HDLC y se establecerá una jerarquía con dichas pantallas. Para identificar las pantallas del simulador, se tendrá en cuenta las distintas pantallas o mesas de trabajo identificadas en los escenarios de uso.

Se plantearán 3 niveles de jerarquía diferentes en los que se agruparán las distintas pantallas identificadas.

- **Primer nivel de jerarquía:** En este nivel de jerarquía, se encontrará la mesa de trabajo de la estación, la cual se trata de la mesa de trabajo principal del simulador. En esta mesa de trabajo se implementan la gran mayoría de funcionalidades del simulador.
- **Segundo nivel de jerarquía:** En este nivel de jerarquía, se encontrará la gran mayoría de mesas de trabajo del simulador, las cuales pueden ser accedidas a través de la mesa de trabajo principal. En estas mesas de trabajo se implementan funcionalidades secundarias.
- **Tercer nivel de jerarquía:** En este nivel de jerarquía, se encontrarán las mesas de trabajo de ayuda y advertencia, las cuales pueden ser accedidas a través de una mesa de trabajo secundaria. En estas mesas de trabajo se implementan funcionalidades adicionales y de escasa importancia.

En la siguiente imagen, se adjunta el mapa del sitio del proyecto:

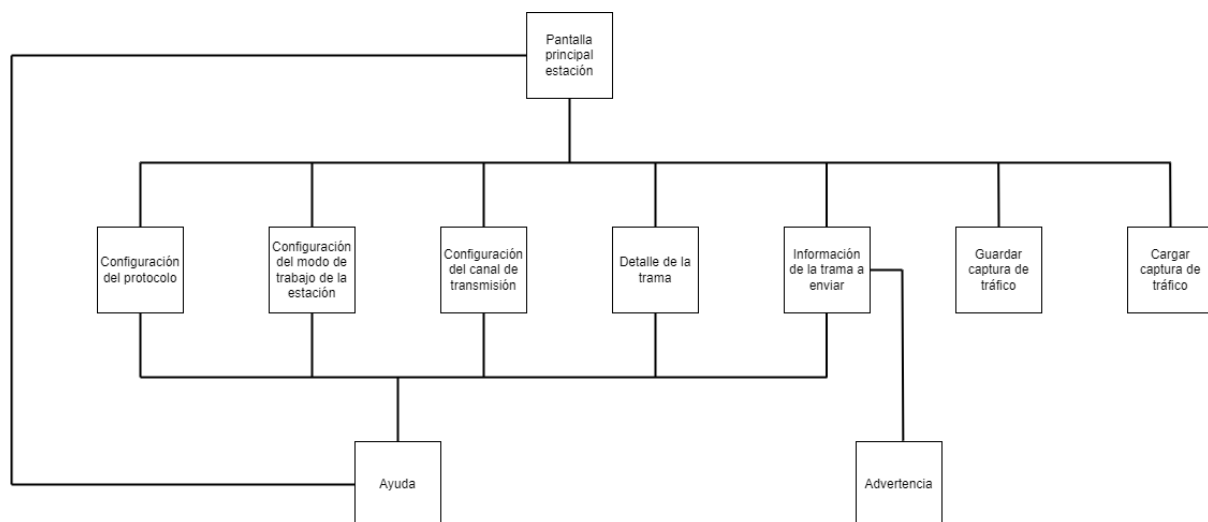


Figura 28: Mapa del sitio del proyecto

## Toma de decisiones de diseño sobre la interfaz gráfica

La toma de decisiones de diseño consiste en la elección de distintos colores para los diferentes componentes de la interfaz, la elección de la tipografía o fuente en la que se presentará la información en la interfaz, la utilización de negrita o cursiva para resaltar información importante, inclusión de imágenes y cualquier otro componente gráfico, etc.

En el caso particular de este proyecto, se clasificarán las decisiones de diseño tomadas en 3 grandes categorías:

- **Elección de la tipografía:** En esta categoría, se tomarán decisiones sobre la fuente o fuentes que se van a utilizar para representar la distinta información de la interfaz. Por ejemplo, en el simulador se escogerá el tipo de fuente Segoe UI debido a su facilidad de lectura y su separación en el trazo.
- **Elección de los colores:** En esta categoría, se tomarán decisiones sobre los distintos colores que se van a utilizar en los distintos botones, secciones, texto, bordes que incorpore la interfaz. Por ejemplo, los elementos básicos de la interfaz gráfica seguirán una gama de colores blancos y grises claros.
- **Elección de elementos gráficos:** En esta categoría, se tomarán decisiones sobre las distintas imágenes, animaciones o componentes gráficos que se incorporen en la interfaz gráfica. Por ejemplo, se elegirán los iconos para los distintos botones de la interfaz.

## Elaboración de un esquema de la visualización de la interfaz gráfica (*Wireframe*)

Los esquemas de visualización de la interfaz gráfica o wireframes se utilizan para organizar la información que se encontrará dentro de esta y definir de manera conceptual la colocación de los distintos elementos de la interfaz gráfica. De esta manera, se realizará un esquema o *wireframe* de cada una de las pantallas o mesas de trabajo identificadas anteriormente en el mapa del sitio.

En el caso particular de este proyecto, para la colocación de los elementos de la interfaz se tendrá bastante en cuenta la distribución de los elementos en la interfaz gráfica de la herramienta Visual\_HDLC ya que implementa una funcionalidad similar a la que se desea en este proyecto. Aunque la interfaz gráfica de Visual\_HDLC se encuentre algo anticuada en términos de visualización, se considera que la distribución de los elementos en la interfaz es bastante correcta, por lo que es totalmente lógico seguir una distribución similar de los elementos en el proyecto actual.

También se tendrá en cuenta la utilización de ciertos patrones de colocación comunes en muchas interfaces gráficas como la utilización de barras superiores o la utilización de ventanas con distintas secciones. También se ha buscado en todo momento agrupar los distintos elementos de la interfaz en distintas secciones o recuadros, sobre todo en aquellas pantallas donde se tiene mayor cantidad de información

En la siguiente imagen, se adjunta un ejemplo de *wireframe* sobre la pantalla principal de la estación del proyecto:

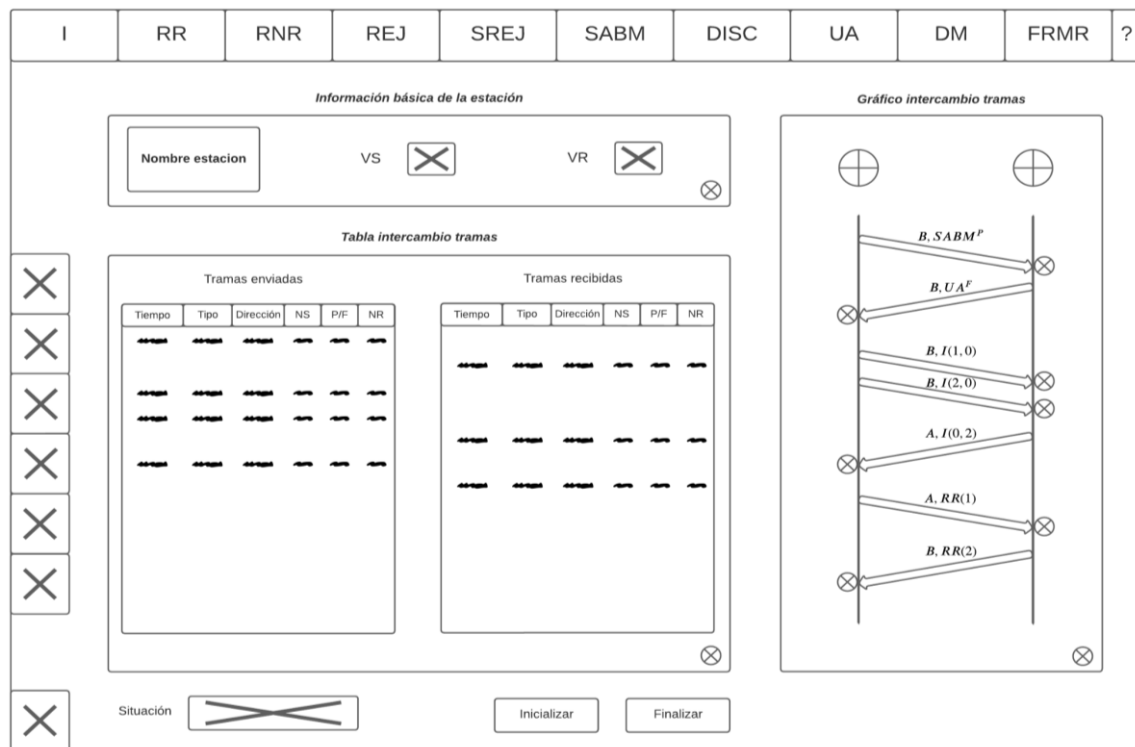


Figura 29: Wireframe de la pantalla principal de la estación del proyecto

## Identificación de subsistemas

Un subsistema de diseño se trata de una evaluación de los paquetes del análisis identificados en la disciplina de Análisis. De esta manera, se produce una nueva evaluación en la arquitectura del sistema a desarrollar. Esos subsistemas de diseño agruparán las distintas clases de diseño identificadas de la misma manera que sucedía en el análisis con los paquetes de servicio.

En el caso particular del proyecto a desarrollar, se identifican 4 subsistemas de diseño principales. Estos subsistemas de diseño son:

- **Gestión del intercambio de tramas**
- **Gestión de las capturas de tráfico**
- **Gestión de la configuración de la estación**
- **Gestión de la visualización de tramas**

El paquete de **Gestión del intercambio de tramas** incluye una gran parte de la funcionalidad del sistema por lo que se subdivide también en 4 subsistemas llamados **Gestión del envío de tramas**, **Gestión de la recepción de tramas**, **Gestión de los Timeouts** y **Gestión de la conexión física**.

El paquete de **Gestión de la configuración** se ha subdividido también en 3 subsistemas llamados **Gestión del protocolo de la estación**, **Gestión del modo de trabajo de la estación** y **Gestión del canal de transmisión de la estación**.

Como se puede observar, la evolución de los paquetes del análisis a subsistemas de diseño no ha sido directa ya que han aparecido nuevos subsistemas de diseño como el subsistema de Gestión de la visualización de tramas el cual no se tuvo en cuenta en la disciplina de Análisis.

## Identificación de clases de diseño

Las clases de diseño representan una evolución de las clases del análisis identificadas anteriormente. Sin embargo, en la disciplina de Diseño no se hace ningún tipo de distinción de tipos de clases. Las clases de diseño están orientadas a la programación por lo que incluirán todos los atributos, métodos, etc. que se utilicen en la implementación final del sistema.

En el caso particular del proyecto a desarrollar, se hará uso del patrón MVVM el cual permitirá identificar las distintas clases del diseño del sistema a desarrollar. En el patrón MVVM se definen 3 elementos principales (***vista***, ***modelo de la vista***, ***modelo***) por lo que se identificarán clases de la vista, clases del modelo de la vista y clases del modelo. De esta manera, siguiendo este patrón MVVM se identificarán todas las clases de diseño, tomando como referencia las clases del análisis identificadas anteriormente.

En la siguiente tabla, se muestra un resumen de todas las clases del diseño identificadas en el desarrollo del proyecto:

Nombre	Tipo
<i>VentanaEstación</i>	Vista
<i>VentanaProtocolo</i>	Vista
<i>VentanaModoDeTrabajo</i>	Vista
<i>VentanaCanalTranmisión</i>	Vista
<i>VentanaVisualizaciónTramaInformación</i>	Vista
<i>VentanaVisualizaciónTramaInformación</i>	Vista
<i>VentanaVisualizaciónDetalleTramaInformación</i>	Vista
<i>VentanaVisualizaciónDetalleTramaSupervisión</i>	Vista
<i>VentanaVisualizaciónDetalleTramaNoNumerada</i>	Vista
<i>VentanaAdvertencia</i>	Vista
<i>VentanaProgreso</i>	Vista
<i>ControlEstación</i>	Modelo de la vista
<i>ControlConfiguración</i>	Modelo de la vista
<i>ControlVisualizaciónTramaInformación</i>	Modelo de la vista
<i>ControlVisualizaciónTrama</i>	Modelo de la vista
<i>ControlVisualizaciónDetalleTramaInformación</i>	Modelo de la vista
<i>ControlVisualizaciónDetalleTramaNoNumerada</i>	Modelo de la vista
<i>ControlVisualizaciónDetalleTramaSupervisión</i>	Modelo de la vista
<i>Estación</i>	Modelo
<i>Trama</i>	Modelo
<i>Protocolo</i>	Modelo
<i>ModoDeTrabajo</i>	Modelo
<i>CanalDeTransmisión</i>	Modelo
<i>NamedPipeClient</i>	Modelo
<i>NamedPipeServer</i>	Modelo

Tabla 6: Resumen de las clases del diseño del proyecto

## Realización de casos de uso a nivel de diseño

La realización de casos de uso a nivel de análisis consiste en una descripción de cómo se ejecutan los casos de uso identificados en la disciplina de Requisitos en función de las clases del diseño previamente identificadas. De esta manera, esta realización se trata de una evolución de la funcionalidad del sistema ya que se incrementa el nivel de detalle.

En el caso particular del proyecto a desarrollar, la realización de los casos de uso a nivel de diseño se realizará utilizando las clases del diseño identificadas previamente. Para la relación de los casos de uso a nivel de diseño, UML proporciona 3 tipos de diagramas:

- **Diagramas de secuencia a nivel de diseño**
- **Diagramas de actividad**
- **Máquinas de estado**

Para el desarrollo de este proyecto, se utilizarán los diagramas de secuencia a nivel de diseño para la realización de los casos de uso a nivel de diseño ya que se cuenta con cierta experiencia y conocimiento sobre este tipo de diagramas además de seguir una mecánica similar a los diagramas de secuencia a nivel de análisis.

Las operaciones utilizadas para detallar la secuencia de interacción serán reales ya que el objetivo es representar la secuencia de operaciones real necesaria para implementar una determinada funcionalidad. Para elaborar los diagramas de secuencia correspondientes, se tendrá en cuenta en todo momento, la secuencia de pasos especificada en la disciplina de Requisitos (trazabilidad).

En la siguiente imagen, se adjunta un ejemplo de la realización de un caso de uso a nivel de diseño:

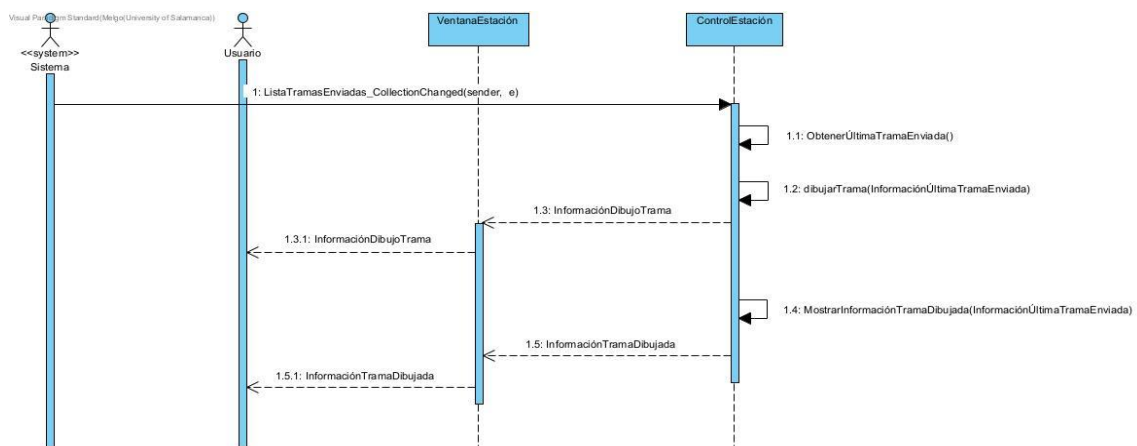


Figura 30: Ejemplo de realización de un caso de uso a nivel de diseño del proyecto

## Elaboración del diagrama de subsistemas

Un diagrama de subsistemas de diseño está formado por los subsistemas de diseño identificados anteriormente, donde dentro de estos subsistemas se agruparán las clases del diseño involucradas en cada subsistema de diseño. También se establecerán relaciones entre los distintos subsistemas de diseño.

En el caso particular de este proyecto, la elaboración del diagrama de subsistemas se habrá realizado de la siguiente manera:

En primer lugar, se identificarán las clases del diseño que deben ir dentro de cada subsistema. Para ello, se revisarán las realizaciones de casos de usos a nivel de diseño de las funcionalidades relacionadas con cada subsistema y se extraerán las distintas clases del diseño utilizadas en dichos diagramas de secuencia.

En segundo lugar, se organizarán dichas clases del diseño dentro de cada subsistema siguiendo el patrón MVVM donde se agruparán las clases vista, las clases modelo de la vista y las clases modelo en 3 subsistemas distintos.



En tercer lugar, se establecerán las distintas relaciones existentes entre los subsistemas de diseño. Para identificar dichas relaciones se evaluarán las dependencias existentes entre los distintos subsistemas.

En la siguiente imagen, se adjunta el diagrama de subsistemas de diseño del proyecto:

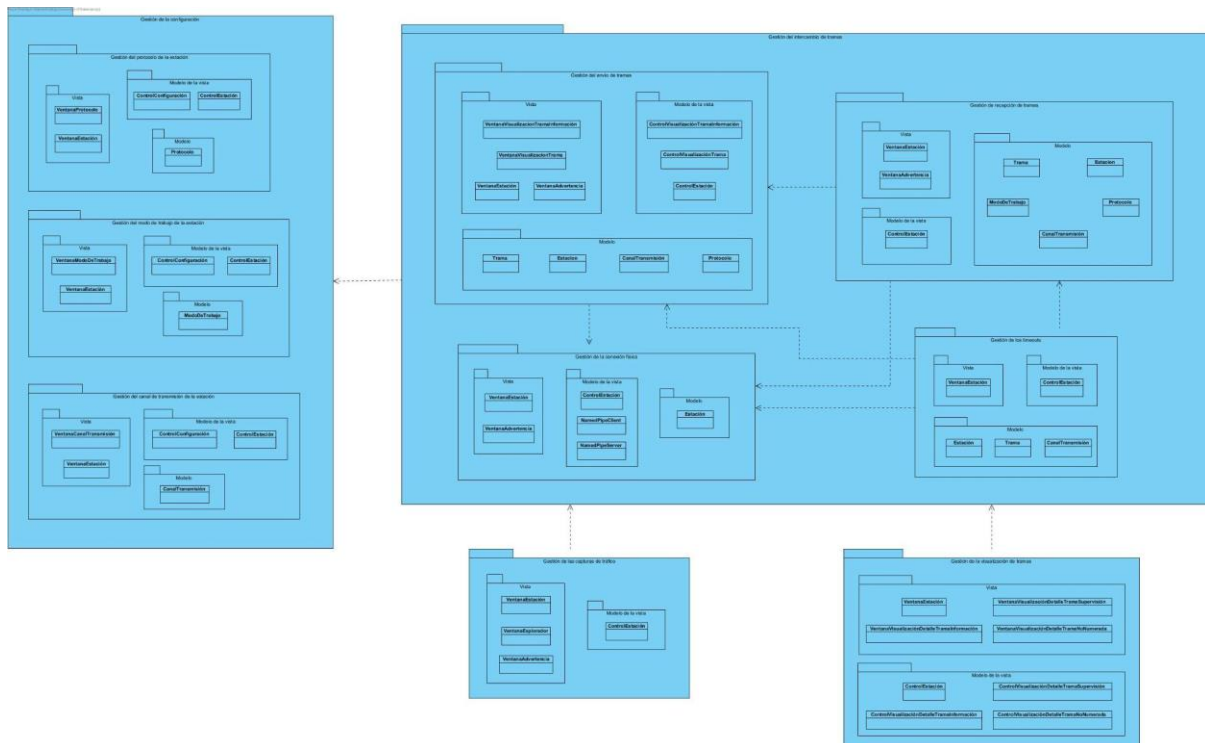


Figura 31: Diagrama de subsistemas de diseño del proyecto

## Elaboración del diagrama de despliegue

El diagrama de despliegue describe la distribución física del sistema en función de cómo se distribuye la funcionalidad de la aplicación en distintos recursos o nodos de cómputo. El principal objetivo es definir en qué entorno físico se va a llevar a cabo las distintas funcionalidades de la aplicación y definir cómo es la relación entre dichos entornos físicos.

En el caso particular del proyecto a desarrollar, el entorno de WPF permitirá la creación de un fichero ejecutable el cual será ejecutado de manera local en el equipo correspondiente. De esta manera, no será necesario el uso de más nodos físicos ya que la conexión física entre las estaciones se realizará de manera local y nunca de forma remota y tampoco se hace uso de una base de datos para almacenar información.

De esta manera, el despliegue del sistema completo se realizará desde un único nodo donde dicho nodo ejecutará el fichero ejecutable 2 veces, una por cada estación, por lo que el nodo contará con 2 entornos de ejecución diferenciados.

En la siguiente imagen, se adjunta el diagrama de despliegue del proyecto:

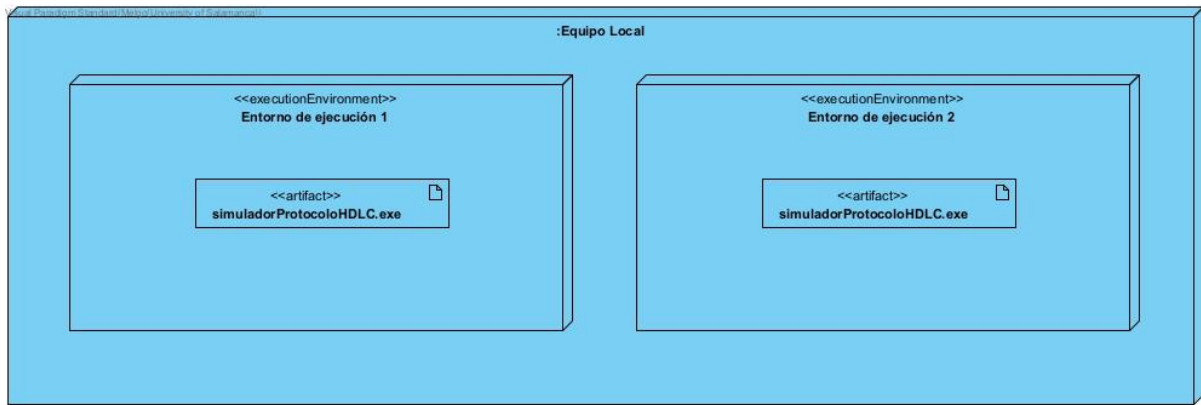


Figura 32: Diagrama de despliegue del proyecto

## Implementación

Esta disciplina, se centra fundamentalmente en tareas relacionadas con la codificación y elaboración de software. Dentro de esta disciplina también se agrupa la realización de prototipos interactivos.

Las principales actividades o tareas realizadas en la disciplina de implementación son las siguientes:

### Elaboración de un prototipo interactivo

Un prototipo se trata de una representación del sistema final el cual carece de cualquier tipo de funcionalidad. El prototipo interactivo incluirá todos los elementos de la interfaz gráfica definidos anteriormente, así como los distintos elementos gráficos, colores, tipografía, etc. También contará con una interacción completa entre las distintas pantallas del simulador que se han definido anteriormente en el mapa del sitio.

En el caso particular de este proyecto, se ha utilizado la herramienta Adobe XD para la implementación del prototipo interactivo debido a su sencillez, a su facilidad para compartir el resultado con otros usuarios y debido a la posibilidad de definir una interacción entre las distintas pantallas del prototipo.

Para la elaboración del prototipo interactivo se habrá tomado como base los esquemas o *wireframes* realizados anteriormente sobre las distintas pantallas del simulador. También se tendrán en cuenta las decisiones de diseño tomadas sobre el tipo de fuente, los colores de los elementos, las imágenes, etc.

En la siguiente imagen, se adjunta un ejemplo de prototipo de la pantalla principal de la estación del proyecto:

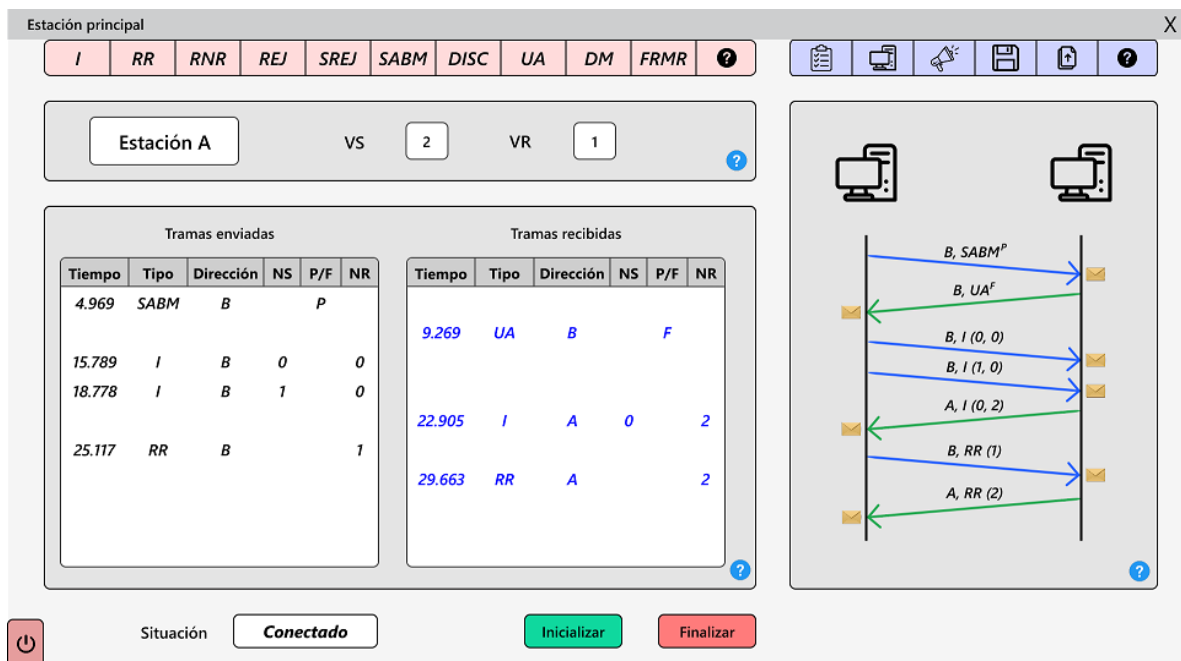


Figura 33: Prototipo de la pantalla principal de la estación del proyecto

## Implementación del simulador del protocolo HDLC

En esta actividad, se realizará la codificación del simulador del protocolo HDLC en el entorno de desarrollo de WPF. Dicha codificación abarcará tanto la representación gráfica del simulador (XAML) como la lógica de negocio del simulador (C#).

### Implementación de la representación gráfica del simulador del protocolo HDLC

Para la implementación de la representación gráfica del simulador del protocolo HDLC, se utilizará el lenguaje de programación XAML, el cual se trata de un lenguaje de etiquetas similar a HTML. A continuación, se describen algunos aspectos de la implementación de la representación gráfica del simulador que se quieren destacar:

#### Diseño de las *TextBox* y *ComboBox* con bordes redondeados

Los objetos *TextBox* y *ComboBox* en XAML no cuentan con una propiedad que permita ajustar el radio de los bordes redondeados. Para solventar esta limitación, se creará un objeto de tipo *Border* que contendrá el *TextBox/ComboBox* al cual se desean aplicar los bordes redondeados. De esta manera, para ajustar el radio del borde redondeado se utilizará la propiedad *CornerRadius* que incluye por defecto los objetos de tipo *Border*.

Para que el mecanismo funcione correctamente, es necesario aplicar un margen al *TextBox/ComboBox* que se encuentra dentro del borde para que no sobresalga por las esquinas del *Border*. También es necesario quitar el borde del propio *TextBox/ComboBox* ya que el borde se encuentra desacoplado.

#### Uso de estilos para las cabeceras de las tablas de tramas enviadas y recibidas

En la implementación del simulador, se desea que el estilo de la cabecera de las tablas de tramas enviadas y recibidas sea distinto al del cuerpo de dichas tablas. Para ello se definirá un estilo llamado *HeaderStyle* el cual tiene como objetivo o *TargetType* las cabeceras de las tablas.

También se utilizará un estilo para representar la información de las tramas enviadas y recibidas de manera centrada. También se utilizará un *DataTrigger* para cambiar el color de texto a rojo cuando se envíe/reciba una trama errónea.

## **Diseño de los mecanismos de ayuda**

Para realizar el diseño de los mecanismos de ayuda, se utilizarán distintos *Canvas* donde en cada *Canvas* se dibujará un bocadillo que contendrá en su interior un texto de ayuda. Por defecto, la visibilidad de los *Canvas* estará oculta, de manera que solo se hará visible el *Canvas* cuando el usuario pulse el botón de ayuda asociado.

Para elaborar el texto de ayuda, se utilizará un *RichTextBox* el cual permite organizar la información en párrafos y utilizar negrita/cursiva para resaltar ciertos conceptos importantes de la explicación.

## **Implementación de la lógica de negocio del simulador del protocolo HDLC**

Para la implementación de la lógica de negocio del simulador del protocolo HDLC, se utilizará el lenguaje de programación C#, el cual se trata de un lenguaje orientado a objetos de características similares a Java. A continuación, se destacan algunos aspectos de la implementación de la lógica de negocio:

### **Establecimiento de la conexión física entre las estaciones**

Una de las funcionalidades más importantes del simulador es el establecimiento de la conexión. Para realizar el establecimiento de la conexión deben existir dos estaciones que la soliciten y que dichas estaciones tengan nombres distintos.

El establecimiento de la conexión entre las estaciones realmente consistirá en establecer un mecanismo de comunicación entre los dos procesos que las estaciones representan. Como ya se mencionó anteriormente en la disciplina de modelado de negocio, se utilizará las tuberías o pipes como mecanismo de comunicación. De esta manera, se deberá establecer una conexión bidireccional entre los dos procesos a través de tuberías.

### **Funcionamiento de las tuberías en C#**

Para establecer una conexión bidireccional entre 2 procesos a través de tuberías, es necesario conocer de antemano cómo se realiza el uso de tuberías en el lenguaje de programación correspondiente.

Las tuberías en C# siguen un esquema cliente-servidor de manera que un extremo de la comunicación se encargará de crear la tubería cliente (*NamedPipeClientStream*) la cual solicitará una conexión a una tubería servidor (*NamedPipeServerStream*) situada en el otro extremo de la comunicación que estará esperando conexión por parte de algún cliente.

### ***Establecimiento de una conexión bidireccional entre dos procesos***

Las tuberías son un mecanismo de comunicación entre proceso la cual permite una comunicación unidireccional, por lo que será necesario el uso de dos tuberías para establecer una conexión bidireccional. También es importante tener en cuenta que los dos procesos solicitarán el establecimiento de la conexión en instantes diferentes.

El proceso que solicite el establecimiento de la conexión creará en primer lugar una tubería cliente utilizando el nombre de la estación contraria e intentará conectarse a la tubería servidora del otro proceso.

Si dicha conexión tiene lugar, esto significa que el proceso situado en el otro extremo ha solicitado previamente el establecimiento de la conexión. De esta manera, se creará una tubería servidor con el nombre de la estación actual con el objetivo de que la estación situada en el otro extremo se conecte a dicha tubería servidor a través de una tubería cliente.

Si dicha conexión no tiene lugar, se creará una tubería servidor con el nombre de la estación actual. Si la creación de dicha tubería falla significa que la estación situada en el otro extremo ha solicitado previamente el establecimiento de la conexión, pero ambas estaciones tienen el mismo nombre. Si la creación de la tubería es correcta, significa que la estación situada en el otro extremo no ha solicitado previamente el establecimiento de la conexión.

Si se detecta que el nombre de las dos estaciones que intentan conectarse coinciden, la segunda estación creará una tubería cliente para sacar a la tubería servidor de la primera estación de la espera y poder liberar todas las tuberías creadas.

En caso contrario, la estación se mantendrá a la espera de que otra estación se conecte a su tubería servidora. Cuando esto ocurra, la estación creará una tubería cliente con el nombre de la estación contraria para conectarse con la estación situada en el otro extremo la cual ha solicitado el establecimiento de la conexión porque se ha conectado a la tubería servidor de la estación.

Si esta última conexión tiene lugar, el establecimiento de la conexión se habrá realizado correctamente. En el caso de que esta última conexión no tenga, significará que las 2 estaciones que intentan conectarse coinciden.

En el proceso de establecimiento de la conexión se pueden dar las siguientes casuísticas:

- La estación A solicita el establecimiento de la conexión, pero no hay ninguna estación situada en el otro extremo que haya solicitado el establecimiento de la conexión (Figura 34).

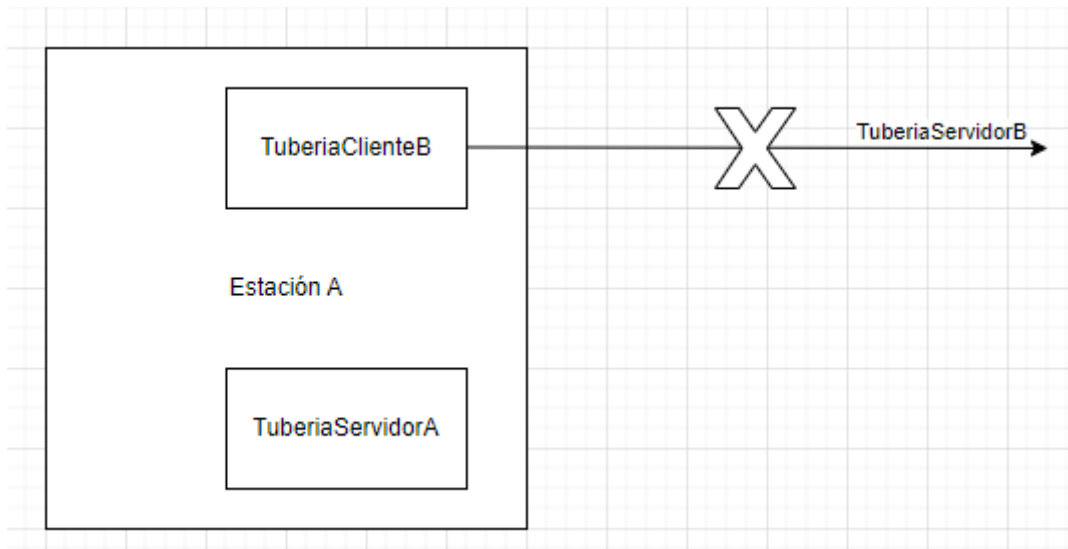


Figura 34: Primera situación en el establecimiento de la conexión

- La estación B solicita el establecimiento de la conexión y la estación A situada en el otro extremo ha solicitado previamente el establecimiento de la conexión (Figura 35).

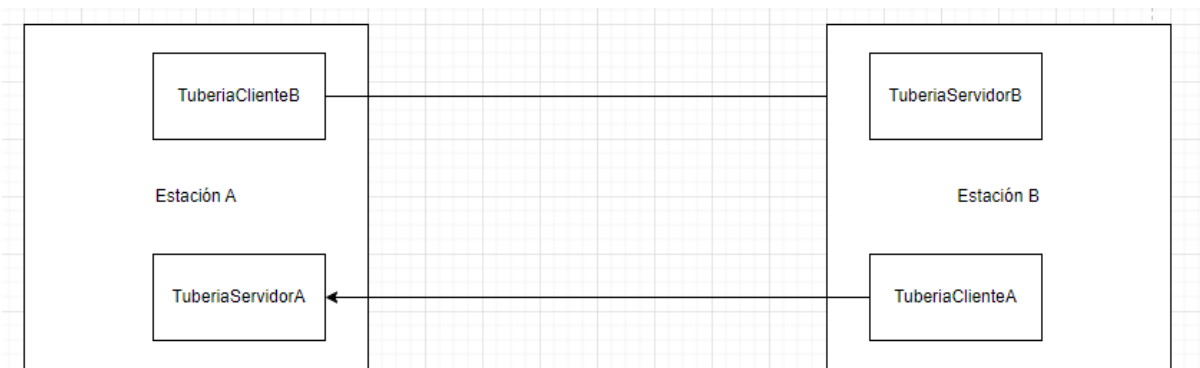


Figura 35: Segunda situación en el establecimiento de la conexión

- La estación A solicita el establecimiento de la conexión y la estación A situada en el otro extremo ha solicitado previamente el establecimiento de la conexión (Figura 36).

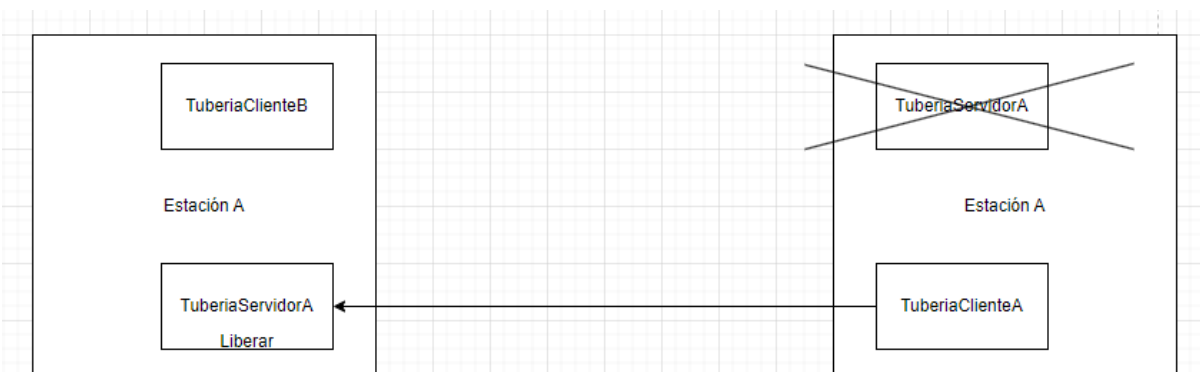


Figura 36: Tercera situación en el establecimiento de la conexión

Si en mitad del establecimiento de la conexión, el usuario decide cancelar el establecimiento del proceso, se realizará un procedimiento similar al realizado cuando se detecta que ambas estaciones tienen nombres iguales. Básicamente el procedimiento consistirá en crear una tubería cliente para sacar a la tubería servidor de la estación que inició el establecimiento de la conexión y así liberar las tuberías creadas.

## **Fin del establecimiento de la conexión entre dos estaciones**

El fin del establecimiento de la conexión consistirá en la liberación del enlace y de las tuberías creadas para establecer dicho enlace. Sin embargo, se desea que ambas estaciones liberen todos los recursos que han creado. Para conseguir esto, la estación que desea terminar con el establecimiento de la conexión física enviará un mensaje de finalización a la estación situada al otro extremo antes de liberar los recursos correspondientes. La estación situada en el otro extremo, al recibir el mensaje de finalización también liberará los recursos correspondientes.

Este mecanismo funciona debido a que es posible leer mensajes de una tubería cerrada siempre y cuando estos mensajes se hayan enviado antes de que la tubería haya sido cerrada. Si esto no fuera así, el mecanismo utilizado no sería válido.

## **Envío de tramas**

Una de las funcionalidades más importantes del simulador es el envío de tramas a la estación físicamente conectada situada en el otro extremo.

Para realizar el envío de una trama cualquiera, el usuario pulsará sobre uno de los botones de la botonera superior en función del tipo de trama que desee enviar. Al realizar esto, se desplegará una nueva ventana en el que se le presentará la información básica de la trama a enviar. También se permitirá al usuario la edición de ciertos parámetros.

De esta manera, la secuencia de acciones realizadas en el envío de cualquier tipo de trama es la siguiente:

1. Se definen los valores por defecto que tendrá la trama a enviar (información del bit del bit C/R, información del bit P/F, nombre de la estación, dirección de la estación, número de secuencia (NS), número de trama esperada (NR)).
2. Si la trama a enviar no se trata de una trama de información se creará una nueva instancia de *VentanaVisualizaciónTrama* a la cual se le pasará como parámetros el tipo de trama a enviar, los valores por defecto que tendrá la trama a enviar, información sobre el protocolo y sobre la lista de tramas enviadas y recibidas por la estación.
3. Si la trama a enviar se trata de una trama de información se creará una nueva instancia de *VentanaVisualizaciónTramaInformación* a la cual se le pasará como parámetros el tipo de trama a enviar, los valores por defecto que tendrá la trama a enviar, información sobre el protocolo y booleano que indica si la trama a enviar se trata o no de una retransmisión de una trama anterior.
4. El usuario en ambos casos modifica a su gusto los valores de la trama a enviar.

5. Cuando el usuario confirme su deseo de enviar la trama correctamente, se realizarán una serie de comprobaciones sobre la trama configurada para verificar que dicha trama está correctamente configurada. Dichas comprobaciones variarán en función del tipo de trama a enviar.
6. Se generará la trama correspondiente a partir de la información especificada en la ventana de visualización de la trama, calculado el CRC de dicha trama y obteniendo el instante temporal en el que se generó dicha trama.
7. Se aplicará la tasa de error definida para la estación, generando tramas erróneas con la probabilidad especificada. También se activarán los *timeouts* correspondientes en el caso de que el envío de la trama implique activar alguno de los *timeouts* (siempre que la trama enviada no sea errónea).
8. Se añade la trama generada en la lista de tramas enviadas y se convierte dicha trama en un objeto JSON para que pueda ser enviada a través de la tubería correspondiente a la estación situada en el otro extremo de la conexión. Se aplica el retardo asociado al canal de transmisión antes de enviar la trama por la tubería correspondiente.

En ocasiones, la propia estación generará y enviará tramas de manera automática ya bien sea por una respuesta automática o bien por el agotamiento de algún *timeout*. En estos casos, la secuencia de acciones será similar con la diferencia de que la información de la trama enviada se establece de manera automática y no es configurada por el usuario.

También es posible enviar tramas de manera directa, sin realizar la configuración de la trama por parte del usuario. En este caso, la secuencia de acciones será similar con la diferencia de que la ventana de visualización de la trama a enviar se cerrará inmediatamente después de ser cargada (se indicará el cierre inmediato de la ventana a través de un booleano). De esta manera, se podrá generar la trama correspondiente a partir de la información colocada por defecto en la ventana de visualización de la trama a enviar.

Para aplicar el retardo en la transmisión, se utilizará un hilo asíncrono que ejecute una tarea en la que el hilo duerme durante el tiempo especificado para el retardo de la transmisión. Mientras el hilo se encuentra realizando dicha tarea, el proceso principal se ocupará de otras tareas con lo que evitamos que el simulador se quede bloqueado. Cuando el hilo finalice la tarea, el proceso principal retomará la ejecución por el punto en el que lo había dejado y realizará el envío de la trama a través de la tubería correspondiente.

## **Recepción de tramas**

Una de las funcionalidades más importantes del simulador es la recepción de tramas a la estación físicamente conectada situada en el otro extremo.

Para recibir tramas por parte de la estación situada en el otro extremo, cada estación creará un hilo asíncrono el cual se encargará de esperar bloqueado a que lleguen datos por la tubería correspondiente. De esta manera, el proceso principal se puede encargar de otras labores mientras se está escuchando por la tubería a la espera de recibir algún tipo de información.



Cuando se reciba una trama o cualquier otro tipo de información por la tubería correspondiente, la estación saldrá del bloqueo y disparará el evento *MessageReceived* al cual se le pasará como parámetro el contenido del mensaje recibido (a través de *MessageReceivedEventArgs* la cual se trata de una clase que hereda de *EventArgs* y que tiene una propiedad llamada *Message* que contendrá el mensaje recibido). Para proteger la recepción de mensajes y garantizar que sea atómica, se utilizará un *mutex* que protegerá la recepción del mensaje.

Al disparar el evento *MessageReceived*, se ejecutará un método llamado *Server\_MessageReceived* en el que se realizarán las acciones necesarias en función del mensaje recibido. El mensaje recibido puede ser una trama, un mensaje de terminación de la conexión física, un mensaje para realizar el guardado de la captura de tráfico y un mensaje para realizar el cargado de la captura de tráfico.

Si el mensaje recibido es una trama se realizarán las siguientes acciones:

1. Se convierte el objeto JSON recibido en un objeto de tipo trama (deserialización).
2. Se obtiene la referencia temporal en la que se ha producido la recepción de la trama y se asigna dicha referencia temporal a la trama recibida.
3. Se añade la trama recibida a la lista de tramas recibidas por la estación.
4. Se calcula el CRC de la trama recibida y se compara con el CRC que incluye la trama recibida. Si estos 2 valores no coinciden, la recepción de la trama será errónea.
5. Si el CRC de la trama es correcto, en función del tipo de trama recibida se realizarán una serie de acciones particulares (actualizar la situación de la conexión, cancelar un *timeout*, generar una respuesta automática, confirmar tramas recibidas, etc.).
6. Una vez se haya terminado de procesar la recepción del mensaje, la estación volverá a escuchar por la tubería correspondiente a la espera de recibir nuevos mensajes.

## Gestión de *timeouts*

Una de las funcionalidades más importantes del simulador, así como una de las funcionalidades más complejas de implementar es la gestión de los distintos tipos de *timeouts*.

La gestión de los *timeouts* se realiza de la siguiente manera:

1. En el envío de cualquier tipo de trama (ya bien se trate de un envío manual, un envío rápido o una trama generada automáticamente), se comprueba si debe activarse algún *timeout*.  
Si la trama enviada tiene el bit P/F activado y se trata de un comando, se activará el *timeout ante command*. Si la trama enviada se trata de una trama de información se activará el *timeout ante trama I*. Si la trama enviada se trata de una trama de rechazo de trama se activará el *timeout ante request*.
2. Al producirse la activación de un *timeout*, se ejecutará un hilo asíncrono que ejecutará una tarea en la que dormirá al hilo tanto tiempo como se especifica en el protocolo para el *timeout* correspondiente.
3. Si el hilo asíncrono sale de la espera antes de que expire el *timeout*, se desactivará el *timeout* correspondiente.

4. Si el hilo asíncrono no sale de la espera antes de que expire el *timeout*, entonces se generará y enviará una trama automática al otro extremo. El tipo de trama generada dependerá del tipo de *timeout* expirado.  
En el *timeout ante command*, se reenviará la trama que activó el *timeout* o una trama RR con el bit P/F activado si la trama que activó el *timeout* fue una trama de información. En el *timeout ante trama I*, se reenviará una trama RR con el bit P/F activado. En el *timeout ante request*, se reenviará la trama que activó el *timeout*.
5. Tras realizar el envío de la trama automática, se creará otro hilo asíncrono en la que se vuelve a ejecutar una tarea en la que dormirá al hilo tanto tiempo como se especifica en el protocolo para el *timeout* correspondiente.

La gestión de *timeouts* cuenta por otra parte con una serie de particularidades:

En primer lugar, es posible que en el mismo momento en el que se expire un *timeout* y se genere una trama automática, la estación haya recibido una trama. Si esto ocurre es posible que se produzca un efecto no deseado.

De esta manera, se ha diseñado un pequeño esquema que garantiza que la recepción de tramas no afectará la generación y envío de tramas. Este esquema consiste en activar una variable booleana que indique que se está produciendo el envío de una trama. Si en el momento en el que se recibe una trama, se está produciendo el envío de otra trama, dicha trama recibida se almacenará en una cola. Cuando el envío de la trama finalice, se revisará la cola de mensajes recibidos y si existe algún mensaje pendiente de recibir, se realizará su recepción.

En segundo lugar, para implementar el *timeout ante command* se utilizarán varios hilos asíncronos, es decir, por cada comando con el bit P/F se tendrá un hilo asíncrono que controle el *timeout* asociado a dicho comando. Cuando se reciba la respuesta a un comando se cancelarán automáticamente todos los *timeouts ante command* activos.

En tercer lugar, para implementar el tiempo ante trama I solo se permitirá la creación de un hilo asíncrono ya que no se desea tener un hilo asíncrono controlando cada trama de información enviada. De esta manera, se buscará en todo momento gestionar el *timeout* de la trama de información más antigua sin confirmar. Es decir, si se envían 3 tramas de información y se confirma la primera, se desea aplicar el *timeout* desde la segunda trama, que se trata de la trama de información más antigua sin confirmar.

Para implementar este funcionamiento, se implementará en primera instancia el *timeout* ante trama I sobre la primera trama de información enviada. Si el *timeout* de dicha trama expira, se revisará la diferencia de tiempo entre la trama de información más antigua sin confirmar y el instante actual. Si dicha diferencia de tiempo es inferior al valor del *timeout* ante trama I, se esperará el tiempo suficiente hasta que dicha diferencia supere el valor del *timeout* ante trama I. Si durante esta segunda espera se recibe confirmación a todas las tramas de información el *timeout* finaliza. Si la segunda espera también expira, entonces se procederá a generar y enviar la trama automática correspondiente.

## **Guardado de capturas de tráfico**

Otra funcionalidad importante del simulador se trata del guardado de capturas de tráfico.

Cuando el usuario solicite el guardado de la captura de tráfico, la estación realizará los siguientes pasos:

1. Se desplegará la ventana del explorador donde el usuario determinará el nombre y ubicación en la que se guarda la captura de tráfico. Con esta información, se generará la ruta en la cual se encontrará almacenada la captura de tráfico.
2. Se almacenará cierta información de estado de la estación como puede ser el nombre de la estación, la situación, el número de secuencia, (NS) o el número de trama esperada (NR). También se almacenará el tamaño de la ventana, el número de tramas erróneas consecutivas recibidas y el turno.
3. Se almacenarán las tramas enviadas y recibidas por la estación en formato JSON (serialización) donde se intercalarán las tramas enviadas con las tramas recibidas.
4. Se enviará un mensaje a la estación situada en el otro extremo de la conexión, para que esta también almacene la captura de tráfico. En el mensaje se incluirá la ruta donde debe almacenarse dicha captura de tráfico. Se almacenará en un fichero con el mismo nombre pero que contiene un (2) al final.
5. La estación situada en el otro extremo realizará el guardado de la captura de tráfico en la ruta indicada como parámetro.

Es posible realizar el guardado de la captura de tráfico sin que las estaciones se encuentren físicamente conectadas, con el inconveniente de que solo la estación que recibió la petición de guardado es la que realizará el guardado de la captura de tráfico. Esto afectará negativamente al proceso de cargado ya que se necesitan las capturas de tráfico de las 2 estaciones para realizar la correcta carga de la captura de tráfico.

De esta manera, el guardado de la captura de tráfico cuando las estaciones se encuentren físicamente conectadas solo tiene utilidad para almacenar el intercambio de tramas realizado desde la perspectiva de la estación en la que se solicita el guardado. Puede verse como una solución de emergencia para los casos en los que el usuario olvide guardar la captura de tráfico antes de finalizar la conexión física.

## **Cargado de capturas de tráfico**

Otra funcionalidad importante del simulador se trata del cargado de capturas de tráfico.

Cuando el usuario solicite el cargado de la captura de tráfico, la estación realizará los siguientes pasos:

1. Se desplegará la ventana del explorador donde el usuario determinará el nombre y ubicación en la que se encuentra almacenada la captura de tráfico. Con esta información, se generará la ruta en la cual se encontrará almacenada la captura de tráfico.
2. Se limpia la lista de tramas enviadas y recibidas.

3. Se extrae la primera línea de la captura, la cual contiene la información de estado de la estación y de otras variables relevantes como el tamaño de la ventana, el número de tramas erróneas consecutivas recibidas y el turno.
4. Se comprueba el nombre de la estación que realizó la captura de tráfico, si el nombre de la estación que realizó la captura no coincide con el nombre de la estación que está realizando el cargado de la captura de tráfico, se cancela la operación.
5. Se extraen las tramas enviadas y recibidas del fichero las cuales se encuentran almacenadas de manera intercalada y se convierten en objetos de tipo trama (deserialización) antes de ser almacenadas en las listas de tramas enviadas y tramas recibidas.
6. Se obtiene la referencia temporal en la que se ha producido la carga de la captura de tráfico y se activan los *timeouts* correspondientes en función del estado de la captura de tráfico.
7. Se enviará un mensaje a la estación situada en el otro extremo de la conexión, para que esta también cargue la captura de tráfico. En el mensaje se incluirá la ruta donde se encuentra almacenada dicha captura de tráfico (incluyendo un (2) al final de la ruta). El mensaje también incluirá la referencia temporal en la que se produjo la carga de la captura de tráfico.
8. La estación situada en el otro extremo realizará la carga de la captura de tráfico correspondiente y recibirá la referencia temporal en la que se produjo la carga de la captura de tráfico. También, se activan los *timeouts* correspondientes en función del estado de la captura de tráfico.

Para realizar el cargado de una captura de tráfico, es necesario que ambas estaciones se encuentren físicamente conectadas, para así poder notificar a la otra estación para realizar también el cargado de la captura de tráfico. Si el usuario solicita el cargado de la captura de tráfico y la estación no se encuentra físicamente conectada, se cancela la operación y se mostrará un mensaje de error.

## **Representación de las tramas enviadas y recibidas**

Una funcionalidad de carácter más secundario pero que tiene cierta relevancia en el funcionamiento del simulador es la representación de las tramas enviadas y recibidas y en especial la representación gráfica de las tramas enviadas y recibidas.

La representación de las tramas enviadas y recibidas en la tabla de tramas enviadas y recibidas es bastante sencilla. En la inicialización de la estación se establece la lista de tramas enviadas como el origen de *items* (*ItemsSource*) de la tabla de tramas enviadas. Lo mismo sucede con la tabla de tramas recibidas. De esta manera, cuando se añade una nueva trama a la lista de tramas enviadas/recibidas, automáticamente la tabla de tramas enviadas/recibidas representa la información de dicha trama.

Para representar en la tabla ciertos campos de la trama, se utilizarán enlaces (*Binding*) a distintas propiedades de la clase Trama. Se han diseñado propiedades especiales en la clase Trama para que en el caso de que un determinado campo tenga un valor incoherente, dicho campo se represente vacío.

La representación gráfica de las tramas enviadas y recibidas siguen un proceso algo más complejo. La secuencia de pasos realizados para representar gráficamente una trama enviada/recibida es la siguiente:

1. Cuando se incluye una nueva trama a la lista de tramas enviadas/recibidas se disparará el evento *CollectionChanged* de la lista correspondiente. De esta manera, se solicitará el dibujo de las nuevas tramas añadidas a la lista de tramas enviadas/recibidas.
2. Se creará y dibujará la flecha asociada a la trama enviada/recibida. Si la trama dibujada es una trama enviada se dibuja de color azul y si la trama dibujada es una trama recibida se dibuja de color verde.
3. Se aplicará una animación sobre la flecha dibujada. Esta animación tendrá como propiedad objetivo la opacidad de la flecha ya que se desea aplicar un efecto de fundido de entrada a las fechas dibujadas.
4. Se creará y dibujará un sobre asociado a la trama enviada/recibida. Se aplicará una animación sobre el sobre dibujado. Esta animación tendrá como propiedades objetivo las coordenadas X e Y del sobre ya que se desea desplazar el sobre a través de una ruta que siga el recorrido de la flecha.
5. Se creará una etiqueta en la que se mostrará información básica de la trama enviada/recibida representada. Esta información será el campo de dirección de la trama, el tipo de trama y los números de secuencia (NS) y trama esperada (NR) en las tramas en las que estos números tengan sentido.
6. Se creará una subetiqueta en la que se mostrará el estado del bit P/F. Esta subetiqueta solo se creará cuando el bit de sondeo de la trama enviada/recibida esté activado.

Si la trama a representar gráficamente se trata de una trama errónea se dibujará de color rojo independientemente de si se trata de una trama enviada o recibida. En este caso, tampoco se tendrá en cuenta el valor del bit P/F en la representación gráfica.

## Pruebas

Esta disciplina, se centra fundamentalmente en tareas relacionadas con la realización de distintas pruebas sobre el sistema ya desarrollado e implementado. Estas pruebas tienen como objetivo validar que se cumplen con los requisitos y objetivos definidos anteriormente para el proyecto a desarrollar.

Las principales actividades o tareas realizadas en la disciplina de pruebas son las siguientes:

### Realización de pruebas en entornos similares

Una actividad muy interesante en el desarrollo de cualquier proyecto software es la relación de pruebas sobre sistemas similares al sistema que se desea desarrollar. La realización de pruebas en entornos similares permite al desarrollador obtener una gran cantidad de información sobre cómo puede ser el comportamiento del sistema en ciertas situaciones.

En el caso particular del proyecto desarrollado, se realizarán distintas pruebas en el entorno de Visual\_HDLC que también implementa el protocolo HDLC en el modo balanceado asíncrono (ABM). Se realizarán distintos intercambios de tramas y se probarán distintas configuraciones del protocolo, modo de trabajo y canal de transmisión con el objetivo de descubrir cómo se comporta el sistema en cada una de las situaciones y tener en cuenta dicho compartimiento en el desarrollo del simulador.

En la realización de pruebas habrá que tener en cuenta los errores de funcionamiento que tiene Visual\_HDLC para no propagar estos errores al desarrollo del simulador. Para cada prueba realizada habrá que contrastar los resultados obtenidos con la teoría de funcionamiento del protocolo HDLC.

En la siguiente imagen, se adjunta un ejemplo de la realización de pruebas en el entorno de Visual\_HDLC:

Time	Tipo	Ad	NS	P/F	NR
2.068	SABM	B		P	
5.972	I	B	0		0
8.291	I	B	1		0
57.856	RR	A		F	1
67.465	DM	A		F	

Time	Tipo	Ad	NS	P/F	NR
4.080	UA	B		F	
14.285	RR	A			2
57.854	I	A	0	P	2
67.462	DISC	A		P	

Figura 37: Ejemplo de prueba en el entorno de Visual\_HDLC

## Definición de casos de prueba

Un caso de prueba es una descripción detallada de las condiciones iniciales que se deben cumplir, las acciones a realizar y los resultados esperados con el objetivo de verificar el correcto funcionamiento de una determinada funcionalidad del sistema. Es importante definir estos casos de prueba antes de realizar las pruebas funcionales sobre el sistema ya implementado.

En el caso particular de este proyecto, los casos de prueba estarán orientados fundamentalmente a validar intercambios de tramas realizados entre las 2 estaciones. Para cada caso de prueba se definirá las condiciones iniciales, la secuencia de acciones a realizar y el resultado final esperado.

A continuación, se muestra un ejemplo de caso de prueba de un intercambio de tramas en el que una estación envía 2 tramas de información y la otra estación confirma la recepción de dichas tramas:

- **Condiciones iniciales:** Las 2 estaciones se encuentran físicamente conectadas y en el modo de trabajo semiautomático.
- **Secuencia de acciones a realizar:** En primer lugar, se establecerá la conexión a través de una trama SABM y su correspondiente asentimiento UA.  
En segundo lugar, la estación A enviará 2 tramas de información. En último lugar, la estación B confirmará las tramas recibidas a través de una trama RR(2).
- **Resultado esperado:** El número de secuencia de la estación A será de 2 y el número de trama esperada de la estación A será de 0. El número de secuencia de la estación B será de 0 y el número de trama esperada de la estación A será de 2.  
Las tramas enviadas por la estación A estarán confirmadas por lo que el *timeout* ante trama I estará desactivado.

## Definición de pruebas de integración

Un aspecto muy importante en cualquier sistema software es el correcto funcionamiento del sistema en su conjunto. Las pruebas de integración permiten verificar el correcto funcionamiento de distintos módulos o componentes del sistema cuando interactúan entre sí.

En el caso particular de este proyecto, se han definido varios tipos de pruebas de integración.

- **Pruebas de integración a nivel de métodos:** En este tipo de pruebas, se busca verificar la correcta cooperación entre varios métodos o funciones para lograr un objetivo común.
- **Pruebas de integración a nivel de clases:** En este tipo de pruebas, se busca verificar la correcta cooperación entre los métodos de distintas clases para lograr un objetivo común.
- **Pruebas de integración de interfaces:** En este tipo de pruebas, se busca verificar el correcto intercambio de información entre los componentes visuales y la lógica de negocio.

## Realización de pruebas de usabilidad sobre el prototipo interactivo

En esta actividad, se realizarán distintas pruebas de usuario con diferentes personas para medir la usabilidad del prototipo interactivo elaborado anteriormente. Estas pruebas tienen como objetivo obtener una retroalimentación por parte de los usuarios y así conocer qué cosas están bien y qué cosas se pueden mejorar de cara al resultado final.

En el caso particular del proyecto a desarrollar, se han elegido usuarios que se encuentran dentro del público objetivo. Se utilizará la técnica del conductor para realizar las pruebas del prototipo. En esta técnica, se plantearán distintas tareas simples que los usuarios deben realizar. Una vez el usuario haya realizado todas las tareas, se utilizará la técnica de entrevista en la que se preguntará al usuario una opinión sobre el prototipo con el objetivo de saber qué aspectos de la interfaz le gustan a los usuarios y qué aspectos de la interfaz piensan los usuarios que se pueden mejorar.

En el caso particular de este proyecto, las tareas que se plantearán a los usuarios sean las siguientes:

Tarea	Descripción
Tarea 1	Cambiar el tamaño de la ventana deslizante.
Tarea 2	Cambiar el modo de trabajo de la estación de semiautomático a manual.
Tarea 3	Bajar a 0 la tasa de error del canal de transmisión.
Tarea 4	Visualizar la información detallada de la trama de receptor preparado (RR) enviada por la estación.
Tarea 5	Obtener información adicional sobre el significado de los distintos campos que doran el campo de control de la trama.
Tarea 6	Enviar una trama de información con el bit P/F desactivado.
Tarea 7	Enviar una trama de petición de desconexión (DISC) con el bit C/R configurado como respuesta.

Tabla 7: Tareas planteadas a los usuarios en la evaluación del prototipo interactivo

Tras la realización de las pruebas, se pueden extraer 2 importantes conclusiones sobre el prototipo interactivo planteado:

- Los usuarios han tenido ciertas dificultades para realizar las 3 primeras tareas. Esto se debe fundamentalmente a la mala elección de los iconos y la falta de claridad para acceder a las distintas configuraciones.
- Los usuarios han podido realizar sin problemas el resto de las tareas por lo que el grueso de la interacción con el usuario se considera correcta, así como el diseño de la interfaz gráfica.

## Realización de pruebas funcionales

Las pruebas funcionales se centran en verificar si el sistema desarrollado cumple con la funcionalidad especificada anteriormente en los requisitos funcionales. Con estas pruebas, se busca verificar que el sistema se comporta de la manera esperada.

En el caso particular del sistema a desarrollar, se plantearán 3 tipos de pruebas funcionales. Estos tipos de pruebas son:

### ***Pruebas de funcionalidad general***

En este tipo de pruebas, se verificará el correcto funcionamiento de las funcionalidades generales del sistema.



En el proyecto desarrollado, las funcionalidades generales son:

- Gestión de la configuración
- Establecimiento de la conexión física
- Envío de tramas
- Recepción de tramas
- Gestión de *timeouts*
- Guardado/cargado de capturas de tráfico
- Representación gráfica de las tramas intercambiadas

De esta manera, en estas pruebas se verificará el correcto funcionamiento de cada una de estas funcionalidades generales

### ***Pruebas de funcionalidad específica***

En este tipo de pruebas, se verificará el correcto funcionamiento de las funcionalidades específicas del sistema.

Un posible ejemplo de una funcionalidad específica del simulador puede ser visualizar el detalle de una trama enviada/recibida. Otro ejemplo de funcionalidad específica del simulador puede ser cambiar el nombre de la estación actual.

De esta manera, en estas pruebas se verificará el correcto funcionamiento de cada una de las funcionalidades específicas que se encuentren en el sistema.

### **Pruebas unitarias**

En este tipo de pruebas, se verificará el correcto funcionamiento de los distintos métodos que conforman el sistema.

Este tipo de pruebas no se consideran estrictamente pruebas funcionales ya que no todos los métodos implementan una funcionalidad específica del simulador. Sin embargo, se ha decidido agrupar este tipo de pruebas dentro de las pruebas funcionales ya que el objetivo de dichas pruebas es verificar el correcto funcionamiento de los distintos métodos por separado, los cuales cooperarán para implementar las distintas funcionalidades del sistema.

## **Realización de pruebas de integración**

Esta actividad consistirá en la realización de las pruebas de integración definidas previamente. Estas pruebas de integración tienen como objetivo verificar el correcto funcionamiento de los distintos componentes del sistema cuando interactúan entre sí.

En el caso particular de este proyecto, se realizarán varios tipos de pruebas de integración.

### **Pruebas de integración a nivel de métodos**

En estas pruebas se busca verificar la correcta coordinación de distintos métodos para implementar una funcionalidad específica.

Por ejemplo, se realizarán pruebas enviando tramas erróneas con el objetivo de que la estación receptora detecte que las tramas recibidas están corruptas. Otro ejemplo de prueba puede ser el desbordamiento del tamaño de la ventana, es decir, se enviarán desde una estación muchas tramas de información consecutivas hasta agotar el tamaño de la ventana con el objetivo de observar el comportamiento del simulador en este caso.

### **Pruebas de integración a nivel de clases**

En estas pruebas, se busca verificar la correcta coordinación de distintos métodos de distintas clases para implementar una funcionalidad específica.

Un ejemplo de prueba de este tipo puede ser el envío manual de una trama cualquiera. En el envío manual de una trama participan las siguientes clases:

La clase *VentanaVisualizaciónTrama.xaml.cs* la cual muestra la información básica de la trama al usuario y permite configurar algunos de sus parámetros y la clase *VentanaEstación.xaml.cs* la cual se encarga de generar y enviar la trama a la estación situada en el otro extremo. También intervienen las clases *NamedPipeClient* y *NamedPipeServer* ya que la trama será enviada a través de una tubería.

Otro ejemplo de prueba puede ser la visualización del detalle de una trama de información. En la representación del detalle de una trama de información intervienen 2 clases:

La clase *VentanaEstación.xaml.cs* la cual recibe la petición de visualización del detalle de una trama específica y la clase *VentanaVisualizaciónDetalleTramaInformación.xaml.cs* la cual recibe la información de la trama específica y se encarga de mostrar la información detallada al usuario.

### **Pruebas de integración de interfaces**

En estas pruebas, se busca verificar el correcto intercambio de información entre los componentes visuales y la lógica de negocio.

Un ejemplo de prueba de este tipo es la representación de las tramas enviadas/recibidas. En la representación de tramas enviadas/recibidas, se presenta cierta información sobre la trama enviada/recibida tanto en las tablas de tramas enviadas y recibidas como en la sección de representación gráfica.

Aunque se haya establecido una clasificación para las pruebas de integración, existen pruebas que pueden clasificarse en varias categorías al mismo tiempo. Sin embargo, lo importante en la realización de las pruebas de integración no es la categorización de las pruebas si no la realización de la mayor cantidad de pruebas posibles para asegurar que el funcionamiento del sistema es el esperado.

## Realización de pruebas de usuario del sistema completo

Para concluir con el apartado de pruebas, se realizarán una serie de pruebas sobre el sistema completo, ya desarrollado. Estas pruebas son tan importantes como otro tipo de pruebas, ya que permiten obtener una realimentación sobre el trabajo general realizado en el desarrollo del proyecto.

En el caso particular de este proyecto, para la realización de las pruebas de usuario del sistema completo, se elegirán usuarios que se encuentren dentro del público objetivo al que va dirigido el desarrollo del simulador. En el público objetivo de este proyecto, se encuentran usuarios con conocimientos en Informática y sobre todo conocimiento del ámbito de las redes de computadores.

De esta manera, se distribuirá el sistema final a los distintos usuarios elegidos con el objetivo de que prueben las distintas funcionalidades del simulador. Una vez, los usuarios hayan realizado las distintas pruebas con el sistema final y hayan probado todas las funcionalidades del simulador, se les pedirá una opinión general sobre el sistema. También se les pedirá que comenten los aspectos que más les hayan gustado o que les hayan parecido más interesantes.

Tras realizar distintas pruebas con los usuarios sobre el sistema final, se pueden obtener las siguientes conclusiones:

- **El diseño de la interfaz:** Los usuarios en general están muy satisfechos con el diseño de la interfaz. La elección de los distintos colores, así como la colocación de los distintos elementos se considera acertada.
- **Mecanismos de ayuda:** Uno de los puntos más repetidos en las opiniones de los usuarios es el gran trabajo realizado con los sistemas de ayuda. Algunos usuarios han comentado que han tenido algunos problemas con el uso de algunas funcionalidades, pero gracias a los sistemas de ayuda han podido resolver el problema y seguir adelante. también han permitido refrescar ciertos conceptos del funcionamiento de HDLC a los usuarios.
- **Representación gráfica de las tramas intercambiadas:** En general, a los usuarios les ha gustado bastante la inclusión de una sección donde se representa de manera gráfica las tramas intercambiadas por la estación. Sobre todo, consideran que la inclusión de animaciones en la representación gráfica de la trama es bastante acertada.
- **Facilidad de comprensión:** A pesar de tratar un tema bastante específico y técnico como es la simulación del protocolo HDLC, los usuarios en general han podido entender el significado de las distintas secciones, campos y elementos del simulador. En algunos casos, algún usuario ha tenido dudas sobre el significado de algún elemento, pero gracias a los mecanismos de ayuda han podido comprender finalmente el significado de dichos elementos.

## Sistema final

Una vez se han documentado y explicado los aspectos más relevantes del desarrollo del proyecto, desde su planteamiento inicial hasta la implementación del sistema completo, se mostrarán los resultados del sistema final.

Para mostrar los resultados del sistema final se hará un recorrido de las pantallas y funcionalidades más importantes del simulador.

## Ventana principal

Cuando el usuario, ejecute el simulador, automáticamente se le presentará la ventana principal de la estación, la cual tiene el siguiente aspecto:

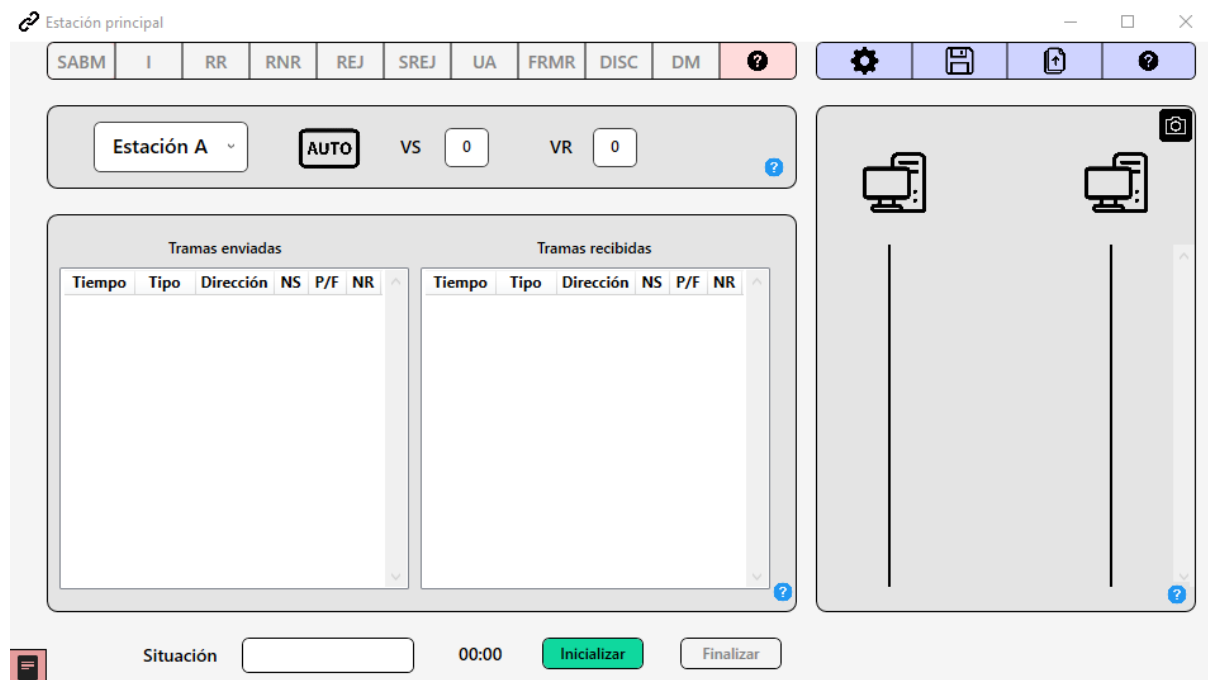


Figura 38: Ventana principal de la estación de simulador

Como se puede observar, existen 3 secciones bien diferenciadas. En primer lugar, existe una pequeña sección con información específica de la estación (Nombre, NR y NS). En segundo lugar, existe una sección en la que se representan las tramas enviadas y recibidas por la estación. En tercer lugar, existe una sección en la que se representa de manera gráfica las tramas intercambiadas por la estación.

## Configuración

El usuario a través de la ventana principal puede acceder a las distintas configuraciones de la estación. Para acceder a las distintas configuraciones simplemente tendrá que pulsar en el botón con la rueda dentada.

La ventana de configuración cuenta con 3 pestañas (Protocolo, Modo de Trabajo y Canal). Por defecto, se muestra la pestaña relativa a la configuración del protocolo.

La pestaña de Protocolo de la ventana de configuración tendrá el siguiente aspecto:



Figura 39: Ventana de configuración en la pestaña de Protocolo de simulador

En esta pestaña, el usuario puede modificar y guardar los distintos parámetros de la configuración del protocolo. Como se puede observar se han agrupado los parámetros del protocolo en 2 secciones. En la primera sección, se han agrupado los distintos *timeouts* y en la segunda sección, se han agrupado los parámetros relacionados con el control de flujo.

La pestaña de Modo de trabajo de la ventana de configuración tendrá el siguiente aspecto:



Figura 40: Ventana de configuración en la pestaña de Modo de trabajo de simulador

En esta pestaña, el usuario puede modificar y guardar el tipo de modo de trabajo de la estación. Existen 2 modos de trabajo para la estación: Modo Manual y Modo Semiautomático.

La pestaña de Canal de la ventana de configuración tendrá el siguiente aspecto:

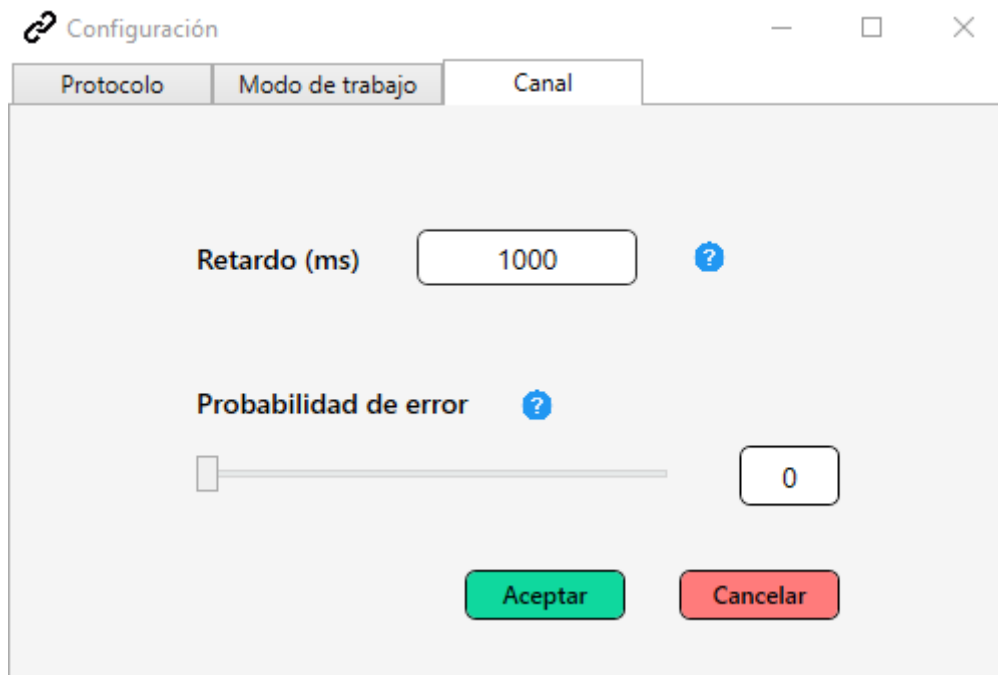


Figura 41: Ventana de configuración en la pestaña de Canal de simulador

En esta pestaña, el usuario puede modificar y guardar los distintos parámetros de la configuración del canal de transmisión.

## Establecimiento de la conexión física

Para establecer una conexión física entre 2 estaciones es necesario configurar las estaciones de manera que las estaciones tengan nombres distintos. Una vez hecho esto, se podrá restablecer la conexión física, se pulsará el botón “Inicializar” en ambas estaciones.

En las siguientes imágenes, se muestra el aspecto visual de la estación, durante el establecimiento de la conexión física y una vez se ha establecido la conexión física.

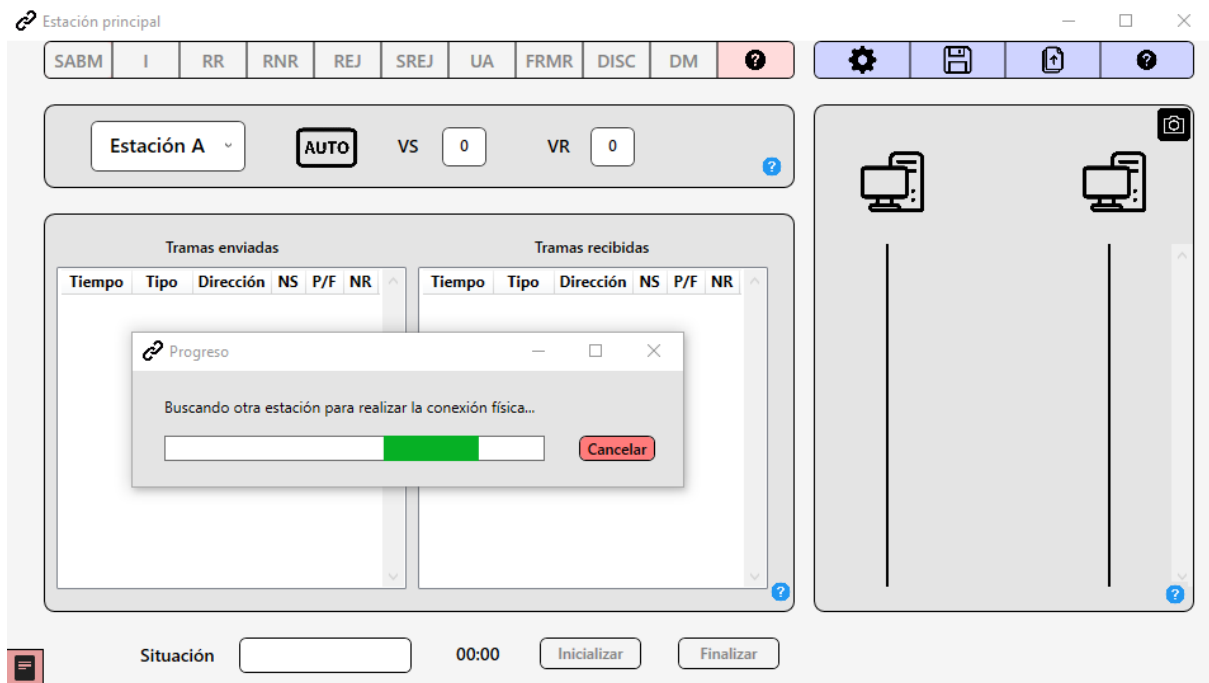


Figura 42: Ventana de la estación durante el establecimiento de la conexión física en el simulador

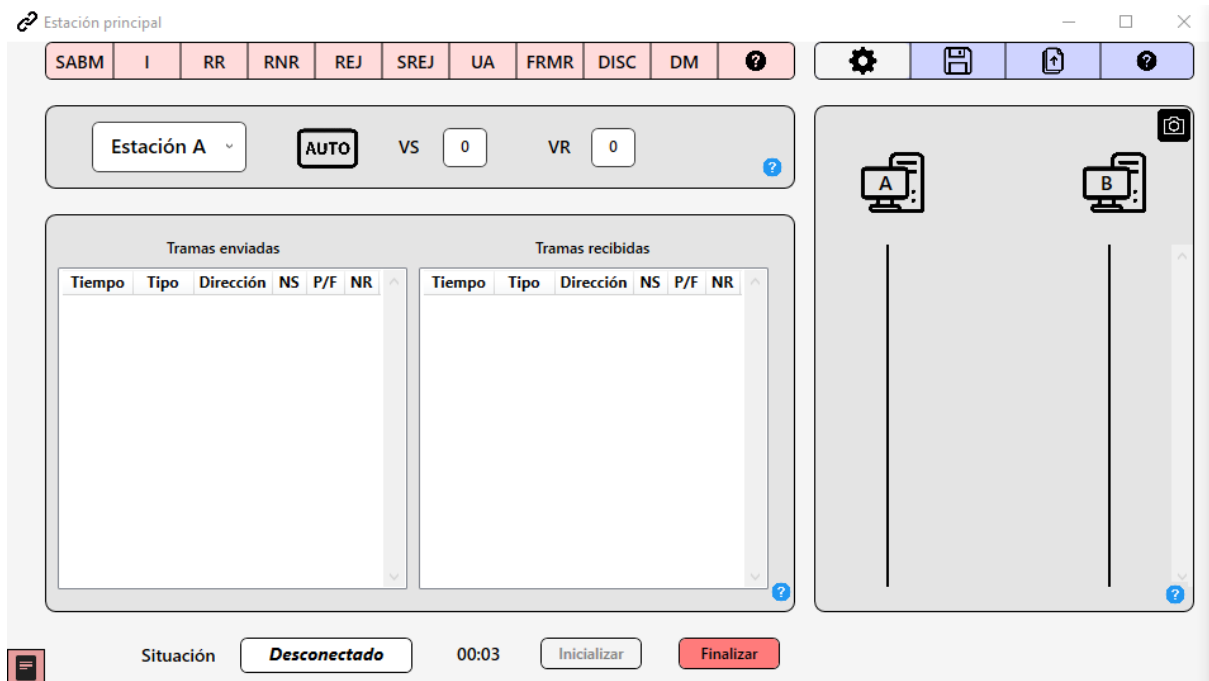


Figura 43: Ventana de la estación después del establecimiento de la conexión física en el simulador

## Envío de tramas

Para enviar una trama a la estación situada en el otro extremo, el usuario utilizará los botones de la botonera superior. De esta manera, en función del tipo de trama a enviar se pulsará un botón u otro.

Cuando el usuario pulse uno de los botones para realizar el envío de una trama, se presentará una ventana en la que se mostrará información básica sobre la trama que se desea enviar. Esta ventana tendrá el siguiente aspecto:

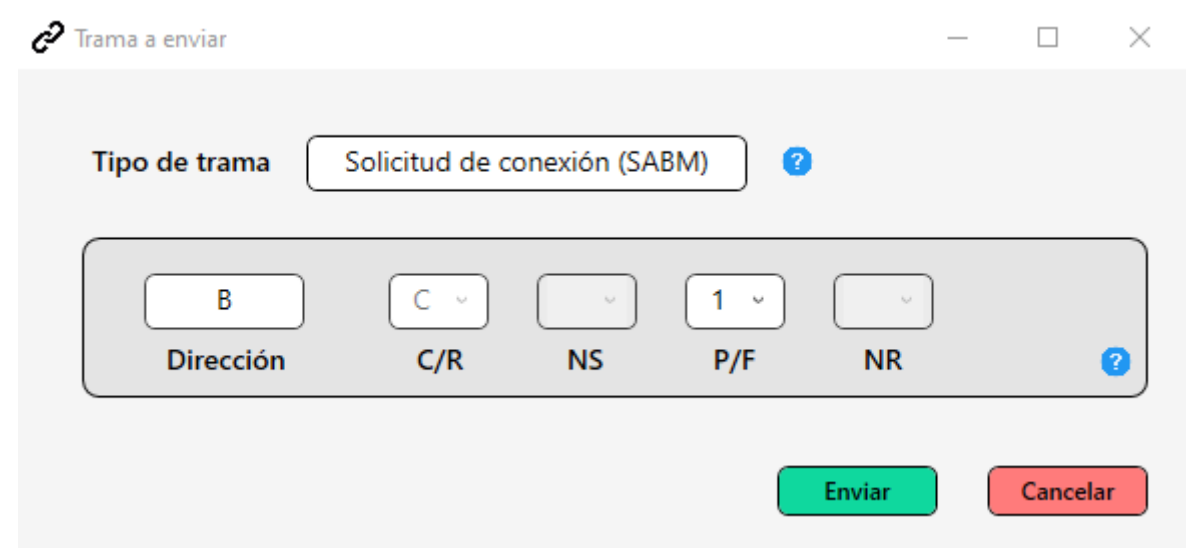


Figura 44: Ventana de visualización de la trama a enviar en el simulador

En esta ventana, el usuario puede modificar ciertos parámetros de la trama a enviar. En función del tipo de trama enviar se podrán modificar una serie de parámetros diferentes.

Una vez el usuario confirme el deseo de enviar la trama correspondiente, se enviará dicha trama y se actualizará la ventana de la estación con la información de la trama enviada. El aspecto visual de la estación será la siguiente:

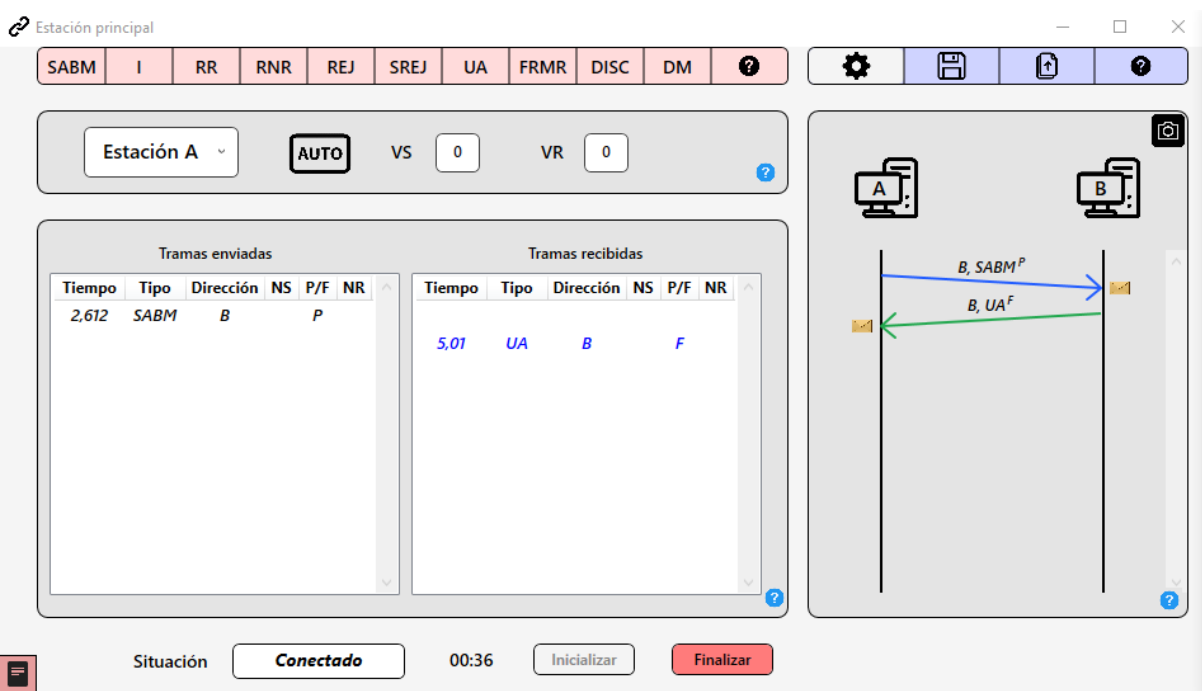


Figura 45: Ventana de la estación tras realizar el envío de una trama en el simulador



Como se puede observar, se ha representado la información de la trama enviada tanto en la tabla de tramas enviadas como en la sección gráfica. También se ha recibido una trama UA de vuelta ya que la estación situada en el otro extremo se encuentra en modo semiautomático y ha generado automáticamente dicha respuesta.

## Visualización del detalle de una trama

Para acceder a la información detallada de una trama enviada o recibida, el usuario deberá hacer clic o bien en el ítem de la tabla correspondiente asociado a la trama que se desea visualizar o bien en el sobre asociado a la trama que se desea visualizar.

Cuando el usuario pulse el ítem o el sobre, se presentará una ventana en la que se mostrará información detallada sobre la trama seleccionada. Esta ventana tendrá el siguiente aspecto:

Composición detallada de la trama

Tipo de trama: Solicitud de conexión (SABM) ?

C/R NS P/F NR ?

01111110 00000010 11111100 Información 0101010111111110 01111110 ?

Flag Dirección Control Información CRC Flag

↓

1 1 11 1 100 Trama no numerada ?

Bit 1 Bit 2 M3M4 P/F M6M7M8

Aceptar

Figura 46: Ventana de visualización del detalle de una trama en el simulador

Como se puede observar se han agrupado la información de la trama en 3 secciones. En la primera sección, se ha agrupado la información básica de la trama. En la segunda sección, se ha agrupado la información general de los campos de la trama. En la tercera sección, se ha agrupado la información específica del campo de control de la trama.

## Timeouts

En el modo de trabajo semiautomático, la estación gestionará una serie de *timeouts*. De esta manera, cuando transcurra un periodo de tiempo determinado y la estación no reciba respuesta en un contexto particular, expirará el *timeout* y se producirá un envío automático de una trama a modo de sondeo.

En la siguiente imagen, se muestra un ejemplo de una situación en la que se ha activado un *timeout* en la estación:

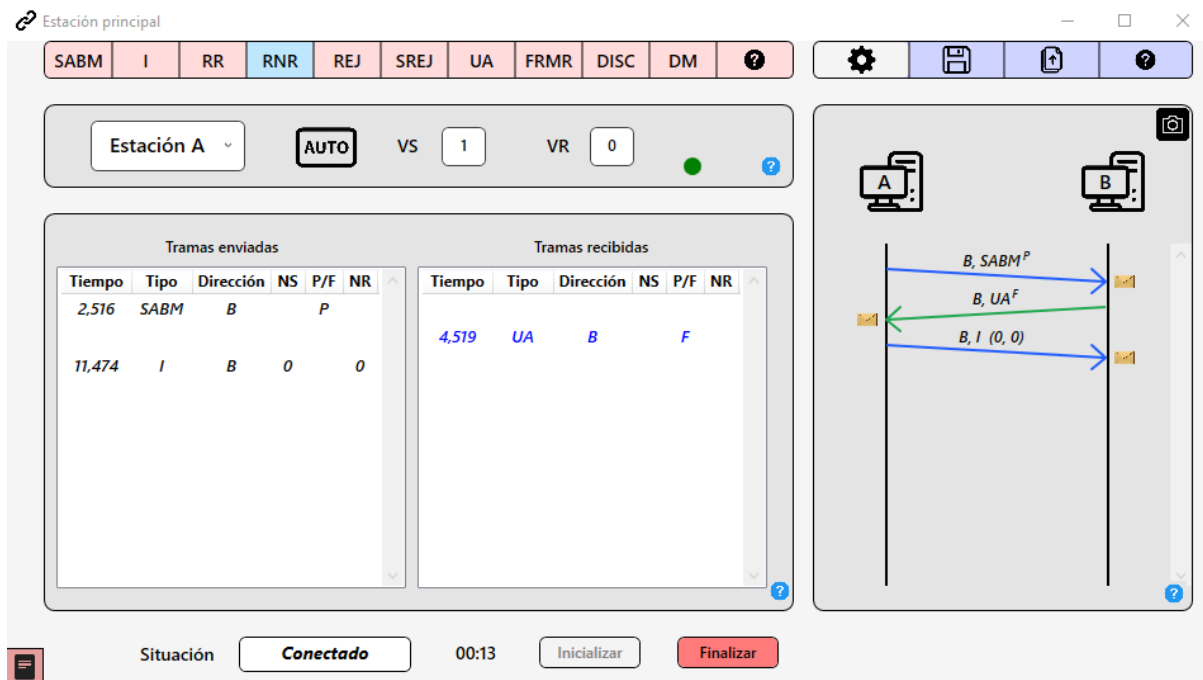


Figura 47: Ventana de la estación tras la activación de un *timeout* ante Trama I en el simulador

Tras realizar el envío de una trama de información, la estación activará el *timeout* ante trama I donde el círculo verde representa la activación del *timeout* ante trama I.

De esta manera, la situación se mantendrá a la espera de una confirmación para dicha trama de información. Si transcurre el tiempo especificado en el *timeout* y no se recibe una confirmación de la trama de información enviada, la estación enviará automáticamente una trama RR con el bit P activado solicitando una confirmación para dicha trama.

En la siguiente imagen, se muestra la ventana de la estación después de que se agote el *timeout* ante trama I:

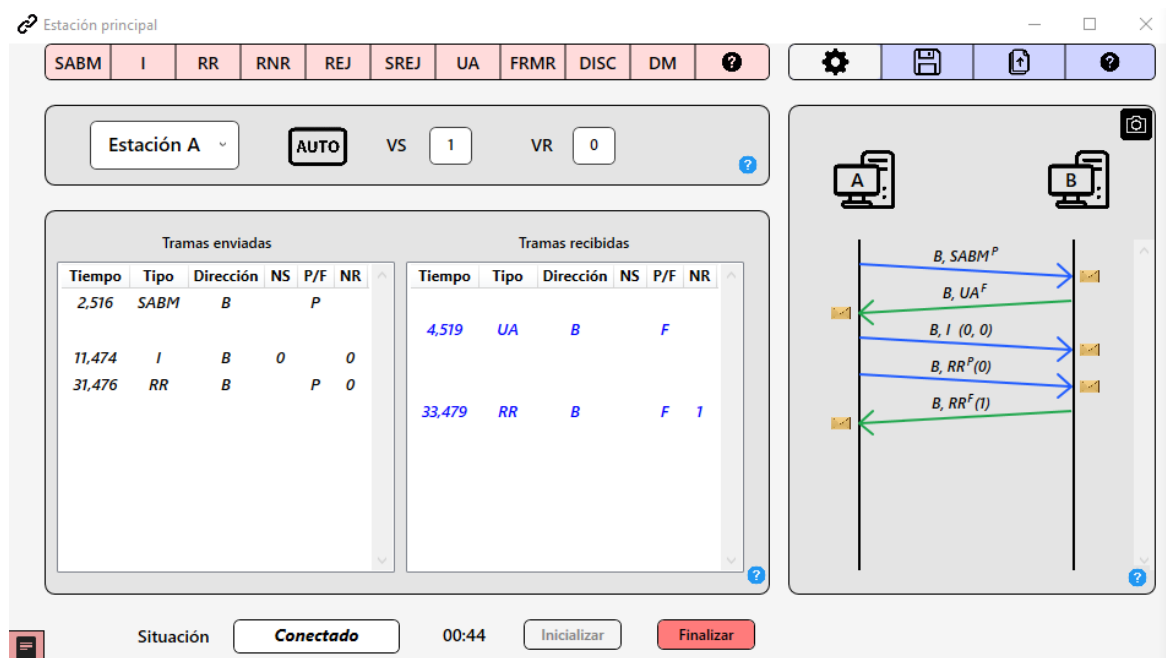


Figura 48: Ventana de la estación tras la expiración de un *timeout* ante trama I en el simulador

## Envío de tramas erróneas

En el intercambio de tramas a través de un medio físico es posible que se produzcan errores en las tramas erróneas. Debido a esto, el usuario tiene la capacidad de enviar tramas erróneas o tramas a través de un canal con fallos.

Cuando el usuario por ejemplo envíe una trama de información errónea, el aspecto visual de la estación será la siguiente:

Estación principal

SABM I RR RNR REJ SREJ UA FRMR DISC DM ?

Estación A AUTO VS 1 VR 0 ?

Tramas enviadas

Tiempo	Tipo	Dirección	NS	P/F	NR
2,608	SABM	B		P	
8,983	ERR	B		ERR	

Tramas recibidas

Tiempo	Tipo	Dirección	NS	P/F	NR
4,611	UA	B		F	

Situación: Conectado 00:18 Iniciar Finalizar

Figura 49: Ventana de la estación tras el envío de una trama errónea en el simulador

Como se puede observar, la trama de información enviada se habrá marcado de color rojo tanto en la tabla de tramas enviadas como en la representación gráfica de la trama.

# Conclusiones y líneas de trabajos futuras

Tras realizar la implementación completa de un simulador del protocolo HDLC en su modo balanceado asíncrono (ABM), se pueden obtener las siguientes conclusiones técnicas:

- Se simula el funcionamiento del protocolo HDLC en su modo balanceado asíncrono.
- Se pueden configurar diferentes parámetros en cada estación.
- Se muestra visualmente el envío y recepción de tramas.
- Se gestionan adecuadamente los *timeouts* y los reintentos de retransmisión.
- Se permite la generación de tramas erróneas y se muestra cómo el protocolo se recupera de estos errores.
- Se permiten dos modos de trabajo. El automático donde la estación responde automáticamente según las especificaciones del protocolo y el manual donde el alumno ha de responder por sí mismo.
- Se incorporan una serie de mecanismos de ayuda que facilitan el aprendizaje del protocolo y de la herramienta.
- Es posible guardar y cargar capturas de tráfico que pueden ayudar a recuperar el trabajo realizado en un determinado punto y a recordar todo lo realizado.
- Se han corregido todos los errores de funcionamiento con los cuales contaba la herramienta de simulación Visual\_HDLC.
- Se ha diseñado una interfaz de usuario moderna que facilita al usuario el uso adecuado de la herramienta. Se han obtenido buenos resultados en cuanto a la facilidad de uso.
- Se han incluido dentro del simulador nuevas funcionalidades que mejoran en gran medida la experiencia del usuario. Por ejemplo, se ha introducido la posibilidad de guardar y cargar capturas de tráfico. También se ha incluido animaciones sobre la representación de las tramas intercambiadas. Además, se ha añadido cierta información de estado en la ventana de la estación como puede ser el tipo de modo de trabajo de la estación, los *timeouts* activados o la referencia temporal de la estación.
- El simulador se entrega en un fichero ejecutable con un tamaño de 1 MB.

En cuanto a las conclusiones personales, este proyecto ha sido de gran utilidad ya que por primera vez se ha hecho frente al desarrollo completo de un sistema software. De esta manera, se han podido adquirir nuevos conocimientos sobre ámbitos sobre los cuales se tenía una base pero que no se comprende con profundidad.

En este proyecto, se ha aprendido a utilizar de principio a fin las directrices de la ingeniería del software. También se ha aprendido a realizar un diseño completo de la interfaz desde el punto de vista del usuario.

En último lugar, este proyecto ha sido de gran ayuda para adquirir una disciplina de trabajo y poder realizar la transición al mundo laboral de manera mucho más sencilla. También ha permitido obtener una visión general sobre el desarrollo de un proyecto y tener claro cada uno de los componentes que conforman dicho proyecto.

Aunque se ha implementado un simulador que implementa de manera completa el protocolo HDLC en su modo balanceado asíncrono (ABM), existen ciertas líneas de trabajo por las que puede seguir evolucionando este proyecto.

Las líneas de trabajo futuras que se plantean para este proyecto son las siguientes:

- Implementación de los modos de respuesta normal (NRM) y modo de respuesta asíncrono (ARM): El desarrollo de este proyecto se ha centrado en la implementación de protocolo HDLC en el modo balanceado asíncrono (ABM). De esta manera, sería interesante incluir la simulación del protocolo HDLC en los modos de respuesta normal (NRM) y de respuesta asíncrona (ARM), permitiendo así el aprendizaje del protocolo HDLC en dichos modos de funcionamiento.
- Implementación de otros protocolos de enlace: De la misma manera que se ha implementado el protocolo HDLC, se pueden desarrollar simuladores que implementen otro tipo de protocolo a nivel de enlace como puede ser Ethernet.
- Posibilidad de incluir un modo de conexión remoto: El simulador del protocolo HDLC está diseñado para simular el intercambio de tramas entre dos estaciones situadas en la misma máquina. Podría ser interesante diseñar un mecanismo de establecimiento de la conexión remoto (a través de sockets, por ejemplo) que permita el intercambio de tramas entre estaciones situadas en máquinas distintas.



# Bibliografía

- "High-Level Data Link Control" Wikipedia, 2021, [https://es.wikipedia.org/wiki/High-Level\\_Data\\_Link\\_Control](https://es.wikipedia.org/wiki/High-Level_Data_Link_Control).
- "Tema 3: El nivel de enlace", Ángeles M<sup>a</sup> Moreno Montero.
- "Visual\_HDLC", Departamento de Ingeniería Telemática de la ETS de Ingeniería de Telecomunicación de Barcelona. Universidad Politécnica de Cataluña.
- "Manual de usuario de Visual\_HDLC", Departamento de Ingeniería Telemática de la ETS de Ingeniería de Telecomunicación de Barcelona. Universidad Politécnica de Cataluña.
- "GNS3: Graphical Network Simulator." GNS3, <https://www.gns3.com>.
- "Cisco Packet Tracer." Cisco Networking Academy, <https://www.netacad.com/en-us/about/cisco-packet-tracer>.
- "El Proceso Unificado de Desarrollo de Software", Wordpress, <https://engenhariossoftwareisutic.files.wordpress.com/2016/04/rup.pdf>.
- "Tema 1: Introducción a la Ingeniería del Software", Jesús F. Rodríguez-Aragón, Carolina Zato Domínguez, [https://studium20.usal.es/pluginfile.php/95857/mod\\_resource/content/5/Transparencias/IS\\_I%20Tema%201%20-%20Introducci%C3%B3n%20a%20la%20IS\\_2021%20-%20completo.pdf](https://studium20.usal.es/pluginfile.php/95857/mod_resource/content/5/Transparencias/IS_I%20Tema%201%20-%20Introducci%C3%B3n%20a%20la%20IS_2021%20-%20completo.pdf).
- "Tema 2: Ingeniería de Requisitos", Jesús F. Rodríguez-Aragón, Carolina Zato Domínguez, [https://studium20.usal.es/pluginfile.php/95862/mod\\_resource/content/4/Transparencias/IS\\_I%20Tema%202%20-%20Ingenier%C3%ADa%20de%20Requisitos\\_2021%20-%20completo.pdf](https://studium20.usal.es/pluginfile.php/95862/mod_resource/content/4/Transparencias/IS_I%20Tema%202%20-%20Ingenier%C3%ADa%20de%20Requisitos_2021%20-%20completo.pdf).
- "Tema 5: Proceso Unificado", Jesús F. Rodríguez-Aragón, Carolina Zato Domínguez, [https://studium20.usal.es/pluginfile.php/95881/mod\\_resource/content/3/Transparencias/IS\\_I%20Tema%205%20-%20Proceso%20Unificado\\_2021%20-%20completo.pdf](https://studium20.usal.es/pluginfile.php/95881/mod_resource/content/3/Transparencias/IS_I%20Tema%205%20-%20Proceso%20Unificado_2021%20-%20completo.pdf).
- "Tema 6: Flujos de trabajo en el Proceso Unificado", Jesús F. Rodríguez-Aragón, Carolina Zato Domínguez, [https://studium20.usal.es/pluginfile.php/95887/mod\\_resource/content/4/Transparencias/IS\\_I%20Tema%206%20-%20Flujos%20de%20trabajo%20en%20el%20Proceso%20Unificado\\_2021%20-%20completo.pdf](https://studium20.usal.es/pluginfile.php/95887/mod_resource/content/4/Transparencias/IS_I%20Tema%206%20-%20Flujos%20de%20trabajo%20en%20el%20Proceso%20Unificado_2021%20-%20completo.pdf).

- "Tema 4: Diseño Centrado en el Usuario", Roberto Therón Sánchez, [https://studium21.usal.es/pluginfile.php/846254/mod\\_resource/content/4/IPO\\_04\\_Disen%C3o%20Centrado%20Usuario.pdf](https://studium21.usal.es/pluginfile.php/846254/mod_resource/content/4/IPO_04_Disen%C3o%20Centrado%20Usuario.pdf).
- "Modelo MVVM", Campus iOS Online, <https://campusiosonline.com/el-patron-de-arquitectura-mvvm-en-ios/>.
- "Metodología para la Elicitación de Requisitos de Sistemas Software", Amador Durán Toro, Beatriz Bernárdez Jiménez, <http://www.lsi.us.es/docs/informes/lsi-2000-10.pdf>.
- "Tema 3: Diseño Centrado en el Usuario", [https://lsi2.ugr.es/~mgea/docencia/diu/Temario/Diu\\_Tema3.pdf](https://lsi2.ugr.es/~mgea/docencia/diu/Temario/Diu_Tema3.pdf).
- "Escenarios de casos de uso", Toni Granollers, <https://mpiua.invid.udl.cat/escenarios/>.
- "Google Forms", Google, <https://docs.google.com/forms/u/0/>.
- "Visual Studio", Microsoft, <https://visualstudio.microsoft.com/es/>.
- "Visual Paradigm", Visual Paradigm International, <https://www.visual-paradigm.com/>.
- "Adobe XD", Adobe, <https://helpx.adobe.com/support/xd.html>.
- "Adobe Color", Adobe, <https://color.adobe.com/>.
- "EZ Estimate".
- "Microsoft Project" Microsoft, <https://www.microsoft.com/es-es/microsoft-365/project/project-management-software>.
- "Lucidchart" Lucidchart, <https://www.lucidchart.com/>.
- "Draw.io" Draw.io, <https://www.draw.io/>.