# TITULO

Sistema de resolución automática de incidencias y generación de informes.

# Resumen

Hoy en día estamos acostumbrados a ver multitud de pantallas en las que solo hay publicidad, ya sea en el autobús, en algún centro comercial o en la panadería de nuestro barrio.

Cuando el número de dispositivos gestionados por una misma empresa aumenta significativamente puede resultar tediosa la tarea de monitorizar y dar soporte a las mismas, aumentando así el coste y el personal necesario para esta tarea.

Con este proyecto se pretende automatizar la tarea de monitorización y resolución de incidencias, con el objetivo de reducir los costes tanto de tiempo como de mano de obra a la hora de detectar y solucionar dichas incidencias.

Además, se pretende que todo este proceso de detección y solución de incidencias sea transparente para el cliente.

# Abstract

Contenido

[TITULO 1](#_Toc454303314)

[Resumen 1](#_Toc454303315)

[Abstract 1](#_Toc454303316)

[1. Introducción 4](#_Toc454303317)

[1.1. Contexto/Antecedentes 4](#_Toc454303318)

[1.2. Estudio del mercado 4](#_Toc454303319)

[1.3. Objetivos generales 5](#_Toc454303320)

[1.4. Posibles tecnologías y software a usar 5](#_Toc454303321)

[1.5. Conceptos básicos 5](#_Toc454303322)

[2. Gestión del proyecto 6](#_Toc454303323)

[2.1. Alcance 6](#_Toc454303324)

[2.2. Metodología de desarrollo 6](#_Toc454303325)

[2.3. Fases 7](#_Toc454303326)

[2.4. Descomposición de tareas 7](#_Toc454303327)

[2.5. EDT 8](#_Toc454303328)

[3. Análisis 10](#_Toc454303329)

[3.1. Descripción de la aplicación 10](#_Toc454303330)

[3.2. Diagramas de actividad (están en papel pero no en formato digital) 10](#_Toc454303331)

[3.3. Requisitos funcionales 11](#_Toc454303332)

[3.4. Requisitos no funcionales 11](#_Toc454303333)

[4. Diseño 12](#_Toc454303334)

[4.1. Arquitectura de la aplicación 12](#_Toc454303335)

[4.2. BBDD 13](#_Toc454303336)

[Diagrama de clases UML / Clases utilizadas a destacar / Librerías 14](#_Toc454303337)

[Diseño de plan de pruebas? 15](#_Toc454303338)

[5. Implementación 16](#_Toc454303339)

[Tecnologías y software empleado durante el desarrollo 16](#_Toc454303340)

[Librerías empleadas 16](#_Toc454303341)

[Aspectos generales de la implementación 16](#_Toc454303342)

[Puntos reseñables 16](#_Toc454303343)

[Problemas y soluciones destacables 16](#_Toc454303344)

[6. Plan de pruebas e integración/implantación 17](#_Toc454303345)

[Plan de pruebas / pruebas unitarias 17](#_Toc454303346)

[Plan de implantación / plan de integración 17](#_Toc454303347)

[7. Seguimiento y control (comparación estimado vs real) 18](#_Toc454303348)

[Seguimiento y control del alcance 18](#_Toc454303349)

[Seguimiento y control del tiempo 18](#_Toc454303350)

[8. Futuras mejoras, mantenimiento del software // Proyecto a largo plazo 19](#_Toc454303351)

[Futuras mejoras/Ampliaciones 19](#_Toc454303352)

[MANTENIMIENTO DEL SOFTWARE 19](#_Toc454303353)

[9. Conclusión 20](#_Toc454303354)

[Agradecimientos 20](#_Toc454303355)

[Bibliografía 21](#_Toc454303356)

# 1. Introducción

## 1.1. Contexto/Antecedentes

The New Ads[[1]](#footnote-1) es una pequeña empresa de Calahorra que se dedica a intentar hacer crecer un negocio y darlo a conocer a través de la cartelería digital. La cartelería digital es una plataforma multimedia en la que se puede distribuir información mediante vídeos o anuncios, todo ello gracias a la colocación de pantallas en establecimientos distribuidos en distintas localidades.

The New Ads proporciona todo lo necesario para que el servicio de la cartelería digital funcione. Esto incluye: el hardware (estaciones de ahora en adelante), el software (administración y control de la publicidad), los servicios de instalación y los servicios de mantenimiento.

Proporcionar el hardware, el software y el servicio de instalación puede resultar más o menos sencillo ya que es un proceso único. El problema surge a la hora de proporcionar el servicio de mantenimiento, ya que conforme aumenta el número de estaciones la tarea de mantenerlas todas y darles soporte se complica, y, llegados a este punto es cuando se plantea la necesidad de un proyecto como este.

Este proyecto surge ante la necesidad de controlar no solo que la publicidad se muestre bien y se vea bien sino controlar que la estación funcione correctamente y monitorizar el estado del hardware y del software para que, en caso de que algo falle, poder detectar dicho fallo y resolverlo, todo ello de forma automática y sin que afecte al rendimiento, de modo que el proceso sea transparente para el cliente final.

## 1.2. Estudio del mercado

Antes de empezar con el proyecto en sí se realizó un estudio del mercado en busca de posibles soluciones ya existentes que pudiesen cumplir los objetivos especificados para este proyecto o que, en su defecto, sirviesen de apoyo durante el desarrollo del proyecto.

Durante el estudio se encontraron herramientas de pago tales como Nagios y op5, capaces de ofrecer una monitorización del sistema casi perfecta. Ambas quedaron descartadas por su alto coste económico y de recursos, puesto que para este proyecto la eficiencia es algo vital y, estas dos herramientas requieren mucha potencia y recursos de los cuales no disponemos.

Así mismo, también se analizaron herramientas opensource como Zabbix, Cacti y OpenHardwareMonitor, pero que por uno u otro motivo también quedaron descartadas.

Tras el estudio realizado se optó por realizar el proyecto partiendo de cero y, a pesar de haber descartado las herramientas anteriores, se optó por integrar la librería proporcionada por OpenHardwareMonitor para las tareas de monitorización de hardware.

## 1.3. Objetivos generales

A grandes rasgos, los objetivos a cumplir al realizar este proyecto son los siguientes:

* Desarrollar procedimientos de evaluación del sistema y detección de posibles problemas o averías.
* Generar informes con los resultados de la evaluación para facilitar la tarea de resolver las averías por parte de los técnicos.
* Desarrollar la base del conocimiento que albergue todos los posibles problemas y soluciones a los mismos y las sirva a los clientes.
* Desarrollar un sistema de ejecución de scripts en estaciones cliente conectado a la base del conocimiento centralizada.

## 1.4. Posibles tecnologías y software a usar

A la hora de plantear las tecnologías a usar se han elegido las siguientes:

* El framework a usar para desarrollar la aplicación será .NET ya que la aplicación está destinada a ser usada en equipos con Windows como sistema operativo y, además, The New Ads ha desarrollado su software bajo este framework.
* El lenguaje de programación a usar dentro del framework .NET será C#, lenguaje con el que se ha tenido un poco de contacto a lo largo del Grado y del que se tiene un conocimiento básico.
* Como entorno de desarrollo se usará Visual Studio, ya que integra tanto el framework como el lenguaje mencionado anteriormente y que, gracias a la universidad, los alumnos disponen de licencias de uso gratuitas.
* Para crear la base de datos se usará MySQL y, su correspondiente conector para C# a la hora de acceder a la misma desde la aplicación.
* Si fuera necesario desarrollar parte de la aplicación vía web se usará PHP.

## 1.5. Conceptos básicos

**Estación[[2]](#footnote-2)**: Consiste en una placa base con una tarjeta gráfica integrada, de tamaño reducido y una potencia suficiente para proporcionar los servicios requeridos.

**Registro**: Hace referencia al Registro de Windows, en el cual se almacenan datos relativos a configuraciones tanto del sistema como de los distintos usuarios y que, es una parte muy importante de las estaciones. El registro definido para cada modelo de estación puede ser diferente.

**Eventos:** Hace referencia a errores que Windows detecta durante la ejecución del sistema y que son almacenados en un registro, pero que el usuario no es consciente de su existencia ni los percibe cuando ocurren. Pueden estar relacionados con cualquier parte del sistema y su gravedad puede variar.

# 2. Gestión del proyecto

## 2.1. Alcance

El objetivo del proyecto es desarrollar una aplicación que sea capaz de evaluar el funcionamiento del hardware y el software de la estación y, detectar posibles fallos o incidencias y, si es posible, arreglar dichos fallos; todo ello sin la necesidad de que intervenga el cliente.

Dicho de otro modo, el objetivo del proyecto consiste en elaborar un sistema de detección y resolución de incidencias en estaciones cliente, conectadas a una base del conocimiento centralizada, de forma automatizada.

Concretamente, las acciones que debe llevar a cabo la aplicación son:

* Evaluar el sistema, comprobando que tanto el hardware como el software funcionan correctamente.
* Establecer los parámetros y umbrales adecuados para que, en base a la evaluación anterior, detecte si existe o no alguna incidencia.
* Generar informes de evaluación del hardware y el software para facilitar la tarea de elaborar una solución por parte del técnico.
* Conectarse a una base del conocimiento centralizada en la que compartir informes de error y soluciones a dichos errores.
* Ejecutar las acciones necesarias para corregir los errores detectados. Dichas soluciones pueden variar dependiendo del tipo de incidencia.
* Si un error ya ha sido resuelto en otra estación, la aplicación debe ‘aprender’ y aplicar la solución de forma automatizada.

## 2.2. Metodología de desarrollo

Dado que el funcionamiento del software desarrollado podía dividirse en distintas etapas (evaluación, análisis de los resultados, generación de informes y resolución de errores) y teniendo en cuenta que para empezar la siguiente etapa era necesario haber finalizado la anterior, parecía lógico planificar el desarrollo del proyecto a través de Sprints, de modo que en cada Sprint se fuese desarrollando cada una de las etapas mencionadas anteriormente.

Por otro lado, ante la necesidad de que para el desarrollo de algunas etapas fuese necesario estudiar el comportamiento de la siguiente así como los inputs que la primera etapa debía producir para que la siguiente etapa funcionase correctamente se optó por llevar a cabo un análisis, diseño y posterior implementación del proyecto, siguiendo el modelo de desarrollo en cascada.

El proyecto se desarrollará por las mañanas de los días laborales, mezclando días de trabajo autónomo en casa y días de asistencia a la oficina de la empresa en Calahorra. Además, habrá reuniones periódicas con los tutores por parte de la universidad y por parte de la empresa para revisar el avance del proyecto.

## 2.3. Fases

Como se menciona anteriormente, la metodología utilizada para desarrollar el proyecto se corresponde con el desarrollo en cascada, pero las fases reales del proyecto no son solo análisis, diseño e implementación.

A continuación se detallan las distintas fases del proyecto:

**Formación**: Fase destinada a la toma de contacto con el funcionamiento de la estación, con todo lo relacionado con rendimiento de hardware y software, con el entorno de desarrollo Visual Studio y el lenguaje de programación C# bajo el framework .NET.

**Análisis**: Fase destinada a concretar el funcionamiento de la aplicación a través del análisis de los requisitos funcionales y no funcionales de la misma. También incluye la realización de una serie de diagramas para ayudar a entender el funcionamiento de la aplicación.

**Diseño**: Fase correspondiente al diseño de las distintas partes de la aplicación, entre las que se incluyen la arquitectura de la solución, la base de datos, el diagrama de clases y el diseño del plan de pruebas.

**Implementación**: Esta fase se corresponde con la fase de desarrollo en sí misma. Cabe destacar que esta fase, si se da el caso, podría incluir un pequeño período de formación ya que las librerías que se utilicen para el desarrollo pueden resultar totalmente desconocidas, en cuyo caso será necesario estudiarlas y entenderlas.

**Testeo**: Fase destinada a poner a prueba la aplicación, siguiendo el plan de pruebas diseñado anteriormente.

**Documentación**: Fase correspondiente a la elaboración de esta memoria y, la elaboración y preparación de la defensa del TFG ante el tribunal de la Universidad.

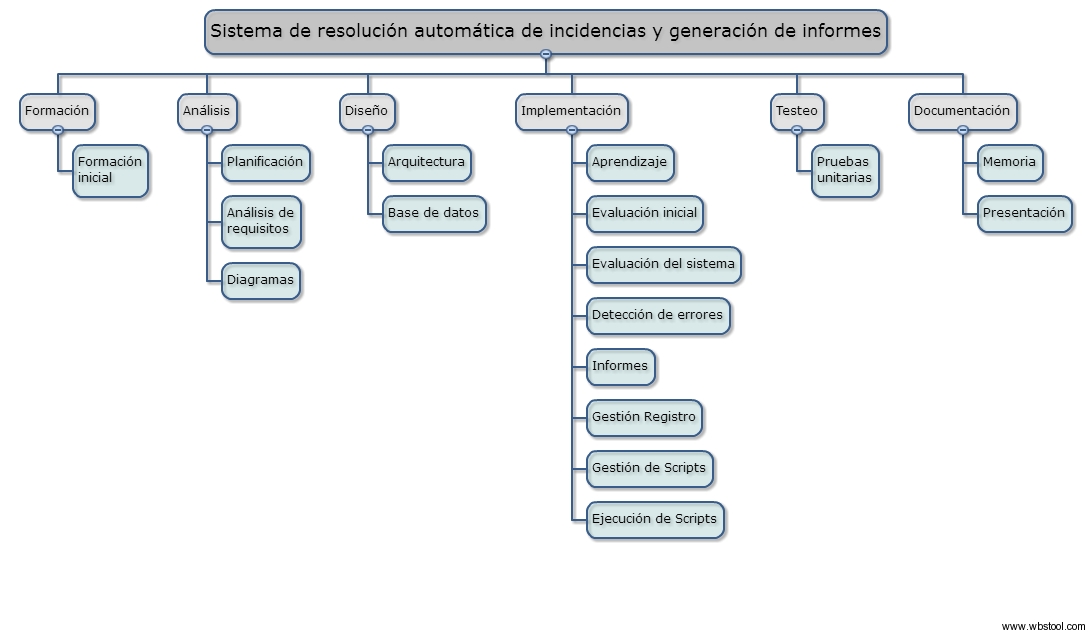
## 2.4. Descomposición de tareas

Una vez identificadas las fases de las que constará el proyecto queda concretar y definir las tareas a realizar durante cada fase, asignándoles a cada una la estimación temporal que corresponda. Dicha descomposición queda reflejada en la tabla 2.1.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Tarea | Descripción | Tiempo estimado |
| Fase de formación | | | |
| 01 | Formación inicial | Período de formación inicial en el que se entrará en contacto con el IDE y el lenguaje que se usarán a lo largo del proyecto | 20 h. |
| Fase de análisis | | | |
| 02 | Planificación | Definición del alcance, las fases y las tareas a realizar durante el proyecto, y de todo lo relacionado con la gestión del mismo | 15 h. |
| 03 | Análisis de requisitos | Captación de requisitos funcionales y no funcionales que deberá cumplir la aplicación | 5 h. |
| 04 | Diagramas | Elaboración de los diagramas pertinentes para el correcto entendimiento del funcionamiento de la aplicación | 5 h. |
| Fase de diseño | | | |
| 05 | Arquitectura | Diseño de la arquitectura que tendrá la aplicación | 10 h. |
| 06 | Base de datos | Diseño de la base de datos que tendrá la aplicación | 10 h. |
| Fase de implementación | | | |
| 07 | Aprendizaje | Período de aprendizaje y de toma de contacto con las librerías que se utilizarán en la implementación del proyecto | 10 h. |
| 08 | Evaluación inicial | Realización de una evaluación inicial, en un entorno óptimo y supervisado con la intención de obtener valores ‘de fábrica’ | 20 h. |
| 09 | Evaluación del sistema | Evaluación del sistema durante su funcionamiento normal, con la intención de detectar errores en dicho funcionamiento | 20 h. |
| 10 | Detección de errores | Análisis de los resultados obtenidos en la tarea 09 con el fin de determinar si existe algún error en el funcionamiento de la estación | 10 h. |
| 11 | Informes | Generación de informes en caso de que exista algún error | 15 h. |
| 12 | Resolución de errores | Desarrollo de técnicas para arreglar los errores que se puedan producir, ya sean errores de Hardware, del Registro o de Eventos. | 30 h. |
| 13 | Gestión Registro | Desarrollar todo lo relacionado con la gestión de ficheros del Registro de Windows | 20 h. |
| 14 | Gestión de Scripts | Desarrollar todo lo relacionado con la gestión de ficheros con los Scripts a ejecutar para resolver errores conocidos | 20 h. |
| 15 | Ejecución de scripts | Integrar en la aplicación la ejecución de Scripts almacenados en ficheros externos a la aplicación | 10 h. |
| Fase de testeo | | | |
| 16 | Pruebas unitarias | Elaboración de pruebas unitarias para ver la correcta resolución de los errores de Hardware, Registro y Contadores de Rendimiento. | 30 h. |
| Fase de documentación | | | |
| 17 | Memoria | Elaboración de la memoria del TFG, correspondiente a este documento | 40 h. |
| 18 | Presentación | Elaboración de la presentación de este TFG y preparación de la defensa ante el tribunal de dicha presentación | 10 h. |
| Total | | | 300 h. |

## 2.5. EDT

En la figura 2.1 se muestra la estructura de descomposición del trabajo, agrupando las tareas por fases.



# 3. Análisis

## 3.1. Descripción de la aplicación

De forma resumida, la aplicación desarrollada para este proyecto se encargará de evaluar el sistema, generar un informe con los resultados obtenidos y, en caso de que se hubiese identificado un fallo, corregir dicho fallo o informar a un técnico para que él se encargue de resolverlo.

La aplicación consistirá en un ejecutable con un comportamiento variable o, por decirlo de un modo más claro, con dos modos de ejecución, descritos a continuación.

**Primer modo de ejecución**: este modo se corresponde con la evaluación inicial del sistema, realizada antes de distribuir la estación al cliente. Durante el proceso de evaluación inicial se obtendrán los valores de hardware y software que la estación tiene en un entorno óptimo y supervisado, sin que exista fallo alguno, suponiendo estos valores como óptimos o ideales.

**Segundo modo de ejecución**: este modo de ejecución será el habitual una vez haya sido distribuida la estación y sus valores ideales hayan sido obtenidos. Este modo se corresponde con la evaluación y detección de incidencias y, será la empresa la que establezca el intervalo de tiempo que pasará entre una ejecución y la siguiente, lo cual no descarta que se pueda ejecutar manualmente.

Durante el segundo modo de ejecución la aplicación realizara una evaluación del sistema similar a la evaluación inicial y, una vez se hayan obtenido los valores de hardware y software, se comparan con los resultados de la evaluación inicial y, en base a las métricas establecidas por el alumno y la empresa, se decide si existe una incidencia o no.

En el caso de que no exista una incidencia se actualizarán los valores de la BBDD puesto que los últimos valores leídos se consideran aceptables y han sido obtenidos en un entorno real y no supervisado. En el caso de que se detecte una incidencia se procedería a, en función del tipo de incidencia, aplicar las medidas correspondientes.

En la descripción de ambos modos de ejecución se han obviado detalles más técnicos del funcionamiento de la aplicación, pero que aparecen en los diagramas utilizados para representar el funcionamiento de la aplicación.

## 3.2. Diagramas de actividad (están en papel pero no en formato digital)

Para describir el funcionamiento de la aplicación de una forma gráfica y detallada se ha optado por usar diagramas de actividad, pues la aplicación no interacciona de ninguna forma con el usuario.

En la figura 3.1. se muestra el proceso inicial de ejecución, en el cuál se establece el modo de ejecución correspondiente a la ejecución actual. Además, se detalla el primer modo de ejecución mientras que el segundo se verá en profundidad en la figura 3.2.

## 3.3. Requisitos funcionales

Aunque en el alcance de la aplicación ya se ha comentado por encima la funcionalidad que debe aportar la aplicación, en esta sección se van a detallar las acciones que va a llevar a cabo la aplicación, correspondientes a las actividades de los diagramas de actividad:

* Utilizar el conector adecuado para poder acceder a la BBDD, realizar consultas y, guardar y actualizar datos.
* Realizar conexiones FTP con el servidor para poder descargar y subir ficheros.
* Leer datos de los componentes hardware, utilizando la librería que corresponda, para poder comprobar el funcionamiento de dichos componentes.
* Comparar datos de hardware y establecer límites en los valores, de modo que se puedan identificar fallos o incidencias en el funcionamiento.
* Acceder al registro de Windows y, leer y modificar los valores de las distintas claves.
* Leer datos de los contadores de rendimiento para que, al igual que con el hardware, se pueda detectar un fallo o incidencia.
* Escribir ficheros de texto para crear informes.
* Crear ficheros XML, escribir en ellos y obtener información de ellos.
* Acceder a la información guardada en ficheros de propiedades y manipularla.
* Programar la ejecución de Scripts externos a la aplicación.

## 3.4. Requisitos no funcionales

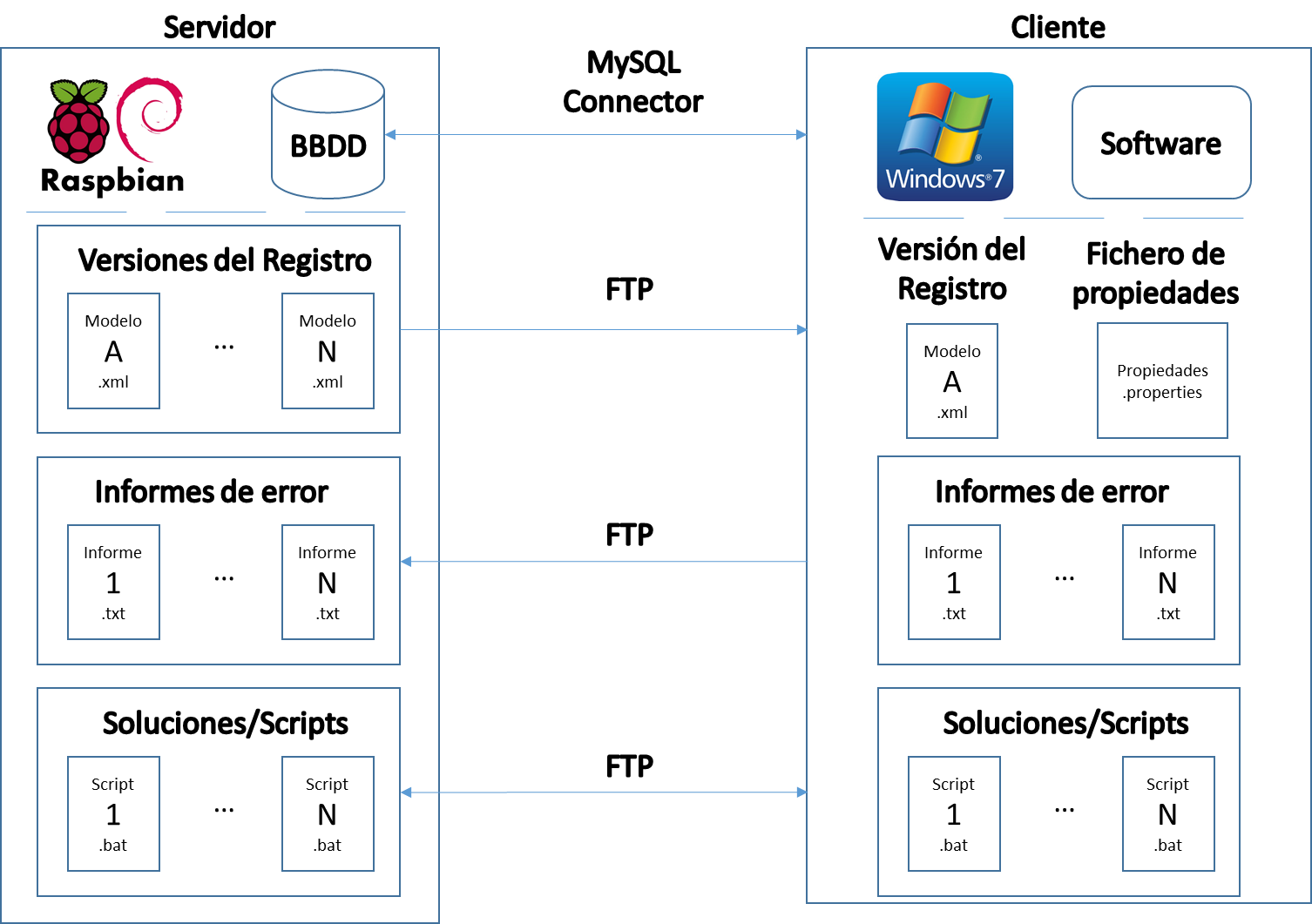
A parte de proporcionar toda la funcionalidad detallada en el apartado anterior, la aplicación debe respetar una serie de requisitos relacionados con otros aspectos como son:

* **Tecnología**: la aplicación se desarrollará para ser ejecutada en Windows 7, ya que es el Sistema Operativo bajo el que corren las estaciones en este momento, pero, al ser posible, deberá ser compatible con versiones superiores de Windows.
* **Eficiencia**: la aplicación debe consumir pocos recursos para que se ejecución no afecte al rendimiento del resto de aplicaciones que se estén ejecutando en la estación cliente, y más aun teniendo en cuenta que los recursos de los que se dispone son bajos de por sí.
* **Conectividad**: al tratarse de una aplicación con una base de datos centralizada en un servidor, la estación en la que se ejecute aplicación debe disponer[[3]](#footnote-3) de conexión a internet en todo momento.

# 4. Diseño

## 4.1. Arquitectura de la aplicación

Para llevar a cabo este proyecto se ha optado por una arquitectura Cliente – Servidor. En la figura 4.1 se muestra de forma gráfica la arquitectura de la aplicación.



Como cliente se utilizará una estación proporcionada por The New Ads, diseñada y configurada como si de un cliente real se tratase, con el fin de hacer el proyecto lo más realista posible.

Como servidor se utilizará una Raspberry Pi 2 Model B propiedad del alumno, con el fin de poder simular conexiones reales a un servidor, ya que trabajar en local sobre la estación cliente supone una carga excesiva para la estación. Este servidor está configurado para aceptar peticiones a la base de datos a través del puerto 3306 y también admite conexiones vía FTP, que es el modo en el que el cliente obtendrá tanto la versión del registro más reciente como los ficheros con scripts para solucionar las incidencias. El envío de informes al servidor por parte de las estaciones cliente se realizará también mediante FTP.

El servidor se encargará de almacenar una BBDD centralizada a la que tendrán acceso todos los clientes. Además, almacenará tres tipos de ficheros en directorios distintos:

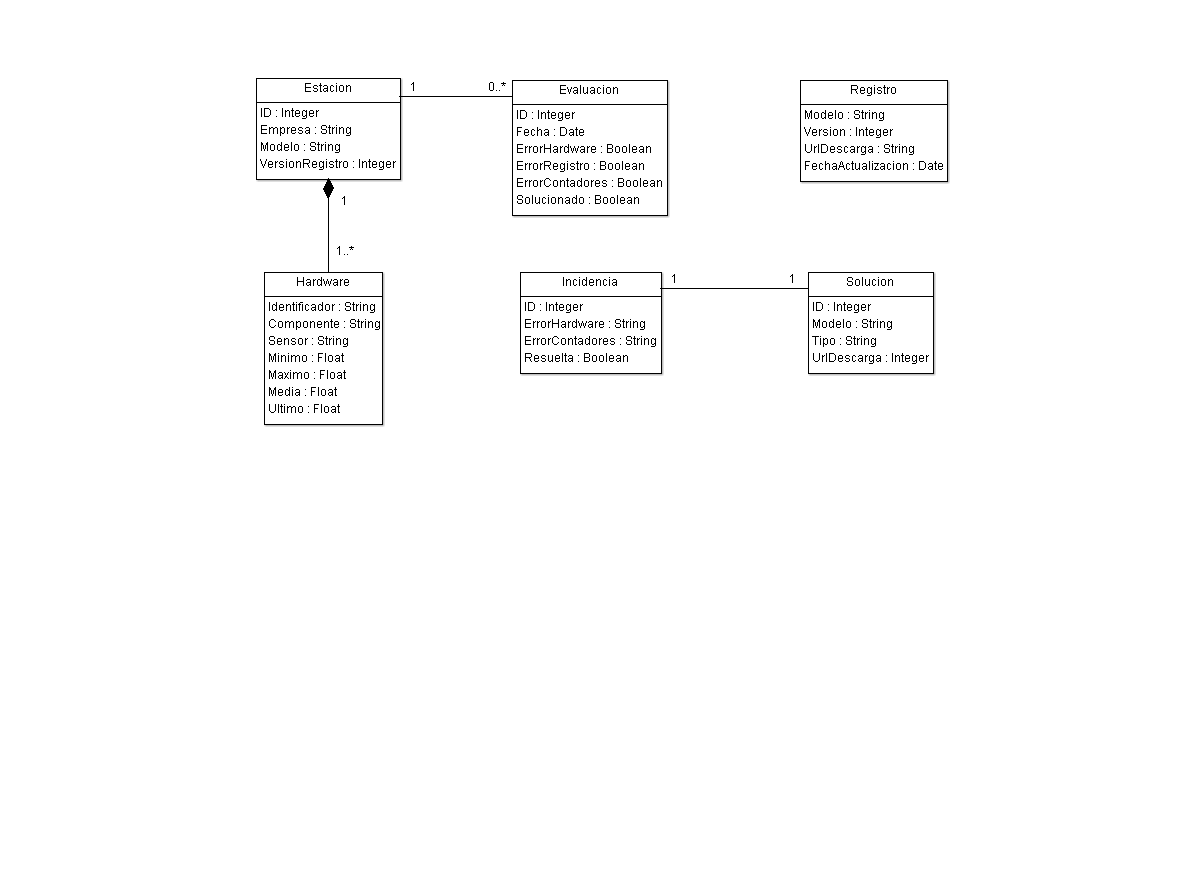
* Ficheros XML con las últimas versiones del registro de windows que debería tener cada modelo de estación.
* Ficheros de texto con los informes de error generados tras la evaluación de los sistemas.
* Ficheros ejecutables con los scripts necesarios para arreglar las incidencias cuya existencia y solución hayan sido detectados.

Por su parte, el cliente o estación contendrá:

* El software o aplicación encargada de realizar todas las tareas descritas para este proyecto.
* Un fichero de propiedades con la información básica de la estación y, como bien dice, alguna que otra propiedad importante para el funcionamiento de la aplicación.
* Un fichero XML con la última versión del registro asociada al modelo de estación correspondiente en el momento de distribución de la estación. Este fichero estará sujeto a cambios según se vaya actualizando la versión en el servidor
* Los ficheros con los scripts que ya hayan sido ejecutados en el cliente.
* Los ficheros de texto con los informes de error de forma temporal, hasta que se hayan enviado al servidor.

## 4.2. BBDD

En la figura 4.2 se puede observar el diseño UML de la base de datos.



**Estacion**: Esta tabla contendrá los datos que identifican a cada una de las estaciones y el tipo de estación de que se trata. El campo ‘VersionRegistro’ que aparece en esta tabla hace referencia a la versión local[[4]](#footnote-4) del registro.

**Hardware**: Esta tabla contendrá los valores de cada uno de los componentes de las estaciones, tomados durante la evaluación inicial. El campo ‘Ultimo’ hace referencia al valor del componente correspondiente durante la última ejecución de la aplicación, por lo que este campo se irá actualizando con cada ejecución de la aplicación y no solo con la evaluación inicial.

**Evaluacion**: Esta tabla hará las veces de un historial de evaluación, guardando las fechas y resultados de las evaluaciones realizadas. Si durante la evaluación no hay ningún error los campos de error y el campo ‘Solucionado’ estarán a ‘False’; si por el contrario hubiese algún error, el campo de error correspondiente estaría a ‘True’ y el campo ‘Solucionado’ estaría a ‘False’ hasta que se solucionase la incidencia, momento en que pasaría a estar a ‘True’.

**Incidencia**: Esta tabla contendrá todas las incidencias que se vayan generando durante las evaluaciones, de modo que queden identificadas por los errores de hardware y de contadores correspondientes (los errores del registro se pueden arreglar sin necesidad de un técnico). Estas incidencias deberán ser estudiadas por un técnico que se encargará de elaborar la solución correspondiente. El campo ‘Resuelta’ se inicializará siempre a ‘False’, hasta que un técnico elabore la solución y entonces se actualice el campo a ‘True’.

**Solucion**: Esta tabla identificará la solución a la incidencia correspondiente. El campo ‘Tipo’ indicará el tipo de solución[[5]](#footnote-5) y, si corresponde, la dirección de descarga del Script correspondiente.

**Registro**: Esta tabla contendrá las últimas versiones del registro correspondientes a cada modelo de estación así como los enlaces a los ficheros con dichas versiones.

Además, se han coloreado las clases de distintos colores en función de qué o quién puede escribir y actualizar los campos de las mismas, como se explica a continuación:

**Azul**: Estas clases serán pobladas únicamente cuando se ejecute la evaluación inicial y sus valores no se modificarán en ningún momento.

**Azul-Verde**: Estos campos se inicializarán con valores tomados durante la evaluación inicial pero con las sucesivas evaluaciones podrán actualizarse siempre que corresponda.

**Verde**: Con cada nueva evaluación se añadirá un registro a la tabla correspondiente, además de generar una incidencia si se da el caso, pues puede que tras la evaluación todo funcione correctamente y no haya necesidad de añadir incidencias.

**Rojo**: Las tablas ‘Registro’ y ‘Solucion’ serán gestionadas por un técnico en todo momento, ya que será él quien se encargue de actualizar las versiones de los registros asociadas a cada modelo y quien descubra y elabore las soluciones a las incidencias encontradas durante las evaluaciones.

## Diagrama de clases UML / Clases utilizadas a destacar / Librerías

Como mucho hablar de las clases de la app

PONER DIAGRAMA DE CLASES CON PROPIEDADES Y METODOS EN BASE A LO YA DESARROLLADO

**EvaluacionHardware:**

* Implementa/Usa OpenHardwareMonitor
* Funcionalidad: se encarga de evaluar y generar informes correspondientes al hardware

**EvaluacionSoftware:**

* Utiliza RegistryKey y PerformanceCounter
* Funcionalidad: lo mismo que la de arriba pero con el software

**Properties.cs:**

* Emulacion de la clase Properties de Java ya que en .NET c# no existe
* Funcionalidad: tratar ficheros de propiedades (NombrePropiedad = Valor)

**EvaluaciónInicial:**

* Correspondiente al modo de ejecución 1, se encarga de realizar las primeras labores de configuración y evaluación y conexión a la BBDD para realizar las inserciones de las lecturas de fabrica y cosas asi (inspeccionar código para ver que hace realmente)

QUIZAS HAGA FALTA UNA CLASE PARA GESTIONAR FICHEROS XML (REGISTROS) Y PARA GESTIONAR SUBIDAS/BAJADAS DE FICHEROS POR FTP.

QUIZAS TAMBIEN HAGA FALTA UNA CLASE INCIDENCIA CON YO QUE SE PROPIEDADES Y METODOS

## Diseño de plan de pruebas?

Hacer aquí como sería el diseño de las pruebas a realizar y en la sección de más debajo de plan de pruebas e integración poner el resultado de las pruebas que se hayan podido llevar a cabo, y los motivos por los que no se han podido llevar a cabo las que no se hayan realizado.

1. Conectividad a internet -> Descartada pues de esto se va a encargar el servidor

2. Actualizar ficheros del registro y su version en la BBDD y que afecte a las estaciones1 -> Vease, actualizar fichero del server, ejecutar programa y que descargue el nuevo fichero

Ligado con lo anterior, pruebas de descarga y subida de ficheros.

3. Corregir errores del registro

4. Detectar errores de Hardware y enviar informe

5. Detectar errores de Contadores y yo que se

6. Ejecución de scripts

# 5. Implementación

## Tecnologías y software empleado durante el desarrollo

Software base desarrollado en .NET, utilizando el lenguaje C#

Base de datos: MySQL

Servidor montado bajo Raspbian y que soporta el protocolo SFTP

PHP finalmente no ha sido usado pues no ha habido parte de programación web.

## Librerías empleadas

OpenHardwareMonitor

Renci.ssh.net creo (la de transferencia sftp)

La de taskscheduler (sacada de codeplex….)

## Aspectos generales de la implementación

0. Como se estructura la evaluación del sistema y que parámetros se supervisan (HARDWARE, SOFTWARE-REGISTRO, SOFTWARE-eventos)

1. Toma de decisiones para saber si se trata de la evaluación inicial o no

2. Parametros para decidir si se trata de una incidencia o no

3. Soluciones aplicadas a cada tipo de incidencia

3.1. HARDWARE🡪INFORME

3.2. SOFTWARE-REGISTRO 🡪CORREGIR EL REGISTRO DEFECTUOSO COMPROBANDO EL FICHERO DEL REGISTRO CON LA VERSIÓN ULTIMA

3.3. SOFTWARE-CONTADORES RENDIMIENTO 🡪 GENERAR INFORME Y COMPROBAR SI EXISTE SOLUCION PARA ESE INFORME, APLICAR SCRIPT SI EXISTE EN LOCAL O EN OTRA ESTACION, NO HACER NADA MAS QUE ENVIAR EL INFORME SI AUN NO HAY SOLUCION

HABLAR DE COMO SE HA IMPLEMENTADO LA BBDD Y QUE SIGNIFICA CADA COSA?

HABLAR DE QUE PARA RELLENAR EL CAMPO MEDIA DE LA TABLA HARDWARE SE HAN HECHO MEDIDAS CADA SEGUNDO DURANTE UN MINUTO Y SE HA HECHO LA MEDIA ETC ETC…

## Puntos reseñables

## Problemas y soluciones destacables

Registros protegidos por el fabricante

# 6. Plan de pruebas e integración/implantación

¿Hacer una clase en el programa llamada Pruebas que incluya métodos booleanos de prueba, que comprueben si funcionan las descargas y subidas via sftp, que simule errores de hardware, que haga pruebas con el registro….?

## Plan de pruebas / pruebas unitarias

* Hardware: Suponer que algo falla y mandar aviso por correo al servicio técnico informando de la estación que falla etc.
* Software-Registro: Corregir el registro defectuoso
* Software-Registro: Actualizar la version del registro y que actualice la version en la bbdd y descargue el nuevo fichero del servidor
* Software-Contadores: Señalar cuando los contadores de error sean mayores que cero
* Ejecucion de scripts: probar que funciona, por ejemplo, iniciando una app que no debería estar en ejecución, y diseñar el script que cierre la app con el id especifico o como sea para que se cierre esa app y deje de gastar recursos.

Hay una especie de prueba o cosa que habría que controlar de alguna forma. La aplicación debe disponer de acceso a internet en todo momento, pero internet podría fallar por un mal funcionamiento de la estación, y en este caso la estación no podría informar de dicha incidencia ni resolverla pues no tiene acceso a la bbdd donde están los scripts, por lo que esa situación de fallo-solucion debería diseñarse de fabrica para que se pudiese resolver desde el primer momento de ejecución de la app.

\*\*\*Como se supone que debo describir los resultados de las pruebas tengo que decir lo siguiente:

Las pruebas relacionadas con el hardware no se han podido realizar de forma real sino que se han simulado, pues para poder tratar este tipo de errores sería necesario manipular el hardware y provocar un funcionamiento erróneo de forma intencionada, y esto supondría un coste para la empresa tanto a la hora de reponer el componente a dañar como en el tiempo destinado a dañarlo de forma controlada. Por ello, se ha simulado una lectura de datos (ficticia) en la que se detecta un fallo y se actúa en consecuencia.

## Plan de implantación / plan de integración

# 7. Seguimiento y control (comparación estimado vs real)

## Seguimiento y control del alcance

Que no he podido hacer y por que

## Seguimiento y control del tiempo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Tarea | Tiempo estimado | Tiempo dedicado | Desviación |
| 01 | Formación inicial | 20 h. |  |  |
| 02 | Planificación | 15 h. |  |  |
| 03 | Análisis de requisitos | 5 h. |  |  |
| 04 | Diagramas | 5 h. |  |  |
| 05 | Arquitectura | 10 h. |  |  |
| 06 | Base de datos | 10 h. |  |  |
| 07 | Aprendizaje | 10 h. |  |  |
| 08 | Evaluación inicial | 20 h. |  |  |
| 09 | Evaluación del sistema | 20 h. |  |  |
| 10 | Detección de errores | 10 h. |  |  |
| 11 | Informes | 15 h. |  |  |
| 12 | Resolución de errores | 30 h. |  |  |
| 13 | Gestión Registro | 20 h. |  |  |
| 14 | Gestión de Scripts | 20 h. |  |  |
| 15 | Ejecución de scripts | 10 h. | 15-20 |  |
| 16 | Pruebas unitarias | 30 h. |  |  |
| 17 | Memoria | 40 h. |  |  |
| 18 | Presentación | 10 h. |  |  |
| Total | | 300 h. |  |  |

# 8. Futuras mejoras, mantenimiento del software // Proyecto a largo plazo

## Futuras mejoras/Ampliaciones

Crear una aplicación web que integrase el resultado de las evaluaciones de todas las estaciones en tiempo real para poder ver de forma grafica y simple cuales están fallando: a pesar de que el proyecto desarrollado debería resolver las incidencias por sí solo siempre se dará el caso de que aparezcan nuevas incidencias cuya solución sea desconocida en el momento, y en este caso sería útil poder saber que estación falla etc etc

App web desde la que gestionar los informes de error recibidos y además gestionar los documentos con las ultimas versiones del registro y los valores de la bbdd correspondientes

App web que compruebe que las estaciones tienen conexión al servidor (mediante un ping o lo que sea)

Realizar un estudio real/en mayor profundidad de los valores y umbrales permitidos a la hora de considerar que una estación falla o no

## MANTENIMIENTO DEL SOFTWARE

# 9. Conclusión

# Agradecimientos

Familia

Tutor universidad (Laureano)

Tutor empresa (Borja) y demás compañeros de la empresa por su trato y ayuda

# Bibliografía

1. Más datos acerca de la empresa y su negocio en <http://thenewads.com/> [↑](#footnote-ref-1)
2. Hay distintos modelos de estación cuyas diferencias residen tanto en el hardware como en el software, pero este proyecto está destinado a cualquier modelo de estación. [↑](#footnote-ref-2)
3. El requisito es que la estación **disponga** de conexión a internet, pero podría darse el caso de que la conexión no funcionase debido a una incidencia o error en el funcionamiento de la estación. Este caso se contempla como un error o incidencia a resolver y no como un incumplimiento de los requisitos. [↑](#footnote-ref-3)
4. La versión local se actualizará, si es necesario, cada vez que se ejecute la aplicación. Por ello, puede darse el caso de que esta versión esté por debajo de la última versión del registro asociada al modelo de estación correspondiente en la tabla ‘Registro’. [↑](#footnote-ref-4)
5. Los posibles tipos de solución son: ‘Script’, ‘Manual’ y ‘Ninguna’. En los casos en los que la solución sea ‘Manual’ o ‘Ninguna’ el valor del campo ‘UrlDescarga’ será **null**. [↑](#footnote-ref-5)