# 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 sólo 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

1­Se entiende por estación……..

Contenido

[TITULO 1](#_Toc452380622)

[NOTAS PARA EL DOCUMENTO 1](#_Toc452380623)

[Resumen 1](#_Toc452380624)

[Abstract 1](#_Toc452380625)

[1. Introducción 4](#_Toc452380626)

[1.1. Contexto/Antecedentes 4](#_Toc452380627)

[Estudio del mercado 4](#_Toc452380628)

[Enumeración informal de objetivos (igual lo suprimo y dejo solo el alcance) 4](#_Toc452380629)

[Conceptos básicos 5](#_Toc452380630)

[2. Gestión del proyecto 6](#_Toc452380631)

[2.1. Actores? 6](#_Toc452380632)

[2.2. Alcance (esta un poco puto en sí) 6](#_Toc452380633)

[2.1. Metodología de desarrollo 6](#_Toc452380634)

[2.1. Fases 7](#_Toc452380635)

[Descomposición de tareas (descomposición, descripción y estimación de tiempos) 7](#_Toc452380636)

[EDT 7](#_Toc452380637)

[Gantt 7](#_Toc452380638)

[3. Análisis 8](#_Toc452380639)

[Descripción de la aplicación (mezcla entre alcance y enumeración informal de objetivos) 8](#_Toc452380640)

[Casos de uso y otros diagramas 8](#_Toc452380641)

[Requisitos de la aplicación o proyecto 8](#_Toc452380642)

[Requisitos funcionales 8](#_Toc452380643)

[Requisitos no funcionales 8](#_Toc452380644)

[4. Diseño 9](#_Toc452380645)

[Arquitectura de la aplicación 9](#_Toc452380646)

[BBDD 10](#_Toc452380647)

[Diagrama de clases UML / Clases utilizadas a destacar / Librerías 11](#_Toc452380648)

[Diseño de plan de pruebas? 12](#_Toc452380649)

[5. Implementación 13](#_Toc452380650)

[Tecnologías y software empleado durante el desarrollo 13](#_Toc452380651)

[Librerías empleadas 13](#_Toc452380652)

[Aspectos generales de la implementación 13](#_Toc452380653)

[Puntos reseñables 13](#_Toc452380654)

[Problemas y soluciones destacables 13](#_Toc452380655)

[6. Plan de pruebas e integración/implantación 14](#_Toc452380656)

[Plan de pruebas / pruebas unitarias 14](#_Toc452380657)

[Plan de implantación / plan de integración 14](#_Toc452380658)

[7. Seguimiento y control (comparación estimado vs real) 15](#_Toc452380659)

[Seguimiento y control del alcance 15](#_Toc452380660)

[Seguimiento y control del tiempo 15](#_Toc452380661)

[8. Futuras mejoras, mantenimiento del software y conclusión 16](#_Toc452380662)

[Conclusión 16](#_Toc452380663)

[Agradecimientos 16](#_Toc452380664)

[Bibliografía 16](#_Toc452380665)

# 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 es llegados a este punto 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. Conceptos básicos

**Estación**: 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.

**Contadores de rendimiento**: Contadores gestionados por Windows en los que se almacena información relativa al rendimiento del Sistema Operativo y de distintas aplicaciones y servicios, como la carga del procesador, la memoria disponible, los bytes que han entrado y salido por una interfaz de red…

# 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 sólo análisis, diseño e implementación.

A continuación se detallan las distintas fases del proyecto:

**Formación**: Esta fase supuso una 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**: Durante esta fase se concretó el funcionamiento de la aplicación a través del análisis de los requisitos funcionales y no funcionales de la misma. También se realizaron una serie de diagramas para ayudar a entender el funcionamiento de la aplicación.

**Diseño**: Durante esta fase se llevó a cabo el diseño de las distintas partes de la aplicación, entre las que se incluyen la arquitectura de la solución, el diseño de la base de datos 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 no fue sólo de desarrollo sino que hubo un pequeño periodo de aprendizaje ya que las librerías empleadas en el desarrollo eran totalmente desconocidas. Entre estas librerías se encuntran: OpenHardwareMonitor, RegistryKey y PerformanceCounter (creo que esto será mejor dejarlo para la parte de implementación en sí misma).

**Testeo**: Durante esta fase se puso a prueba la aplicación tal y como se había diseñado el plan de pruebas 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

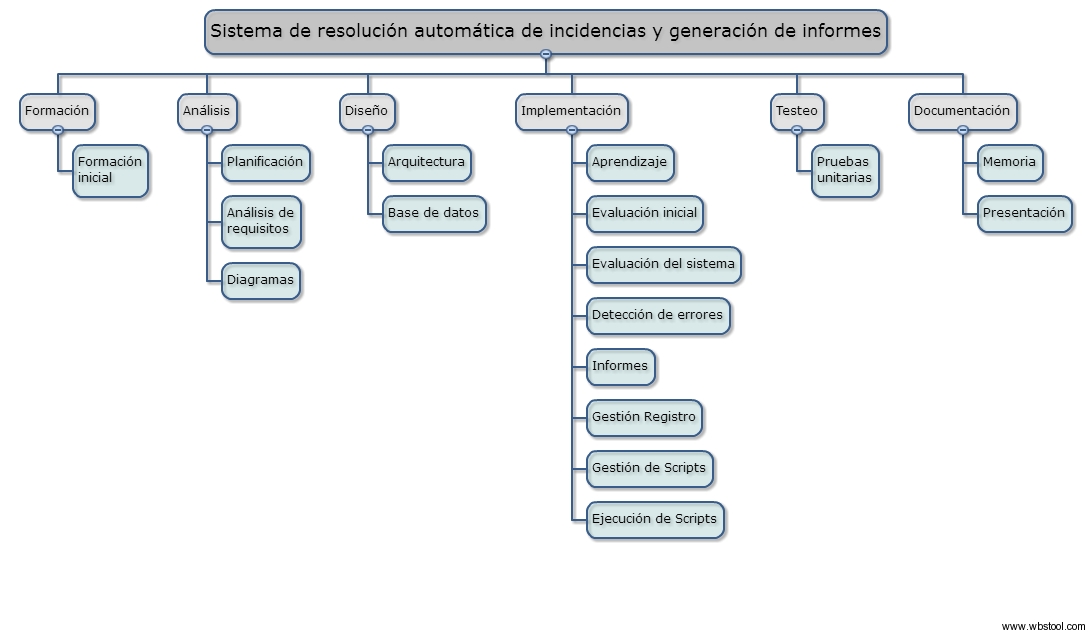
Una vez identificadas las fases de las que constará el proyecto queda concretar y definir las tareas a realizar durante cada fase, asignándoles a cada una la estimación temporal que corresponda. Dicha descomposición queda reflejada en la tabla x.x.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 Contadores de Rendimiento. | 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. |

Poner nota al pie de la tabla con: Tabla x.x: Descomposición de tareas.

## 2.5. EDT

En la figura x.x 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.

Para describir el funcionamiento de la aplicación de una forma gráfica y detallada se ha optado por usar diagramas de actividad, pues al no haber interacción con el usuario no existen casos de uso.

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.

LOS DOS SIGUIENTES DIAGRAMAS NO SE SI PONERLOS, SEGÚN REALICE LOS DOS ANTERIORES SALDRÉ DE DUDAS

En la figura 3.3. se detalla el proceso de comprobación y actualización del fichero del Registro de Windows (conexión a la BBDD y demas)

En la figura 3.4. se detalla el control y gestión de los scripts tanto en local como en otras estaciones (proceso de solución==ejecutar script)

## 3.3. Requisitos de la aplicación

Llegados a este punto queda establecer los requisitos que deberá satisfacer la aplicación.

### Requisitos funcionales

### Requisitos no funcionales

Eficiencia (carga reducida para que no afecte al rendimiento)

Conectividad (Base de datos centralizada, estaciones distribuidas pero conectadas)…

# 4. Diseño

## Arquitectura de la aplicación

Para llevar a cabo este proyecto se ha optado por una arquitectura Cliente – Servidor.

La figura x.x muestra de forma gráfica la arquitectura de la aplicación.

* Figura x.x: Dibujito de un Servidor/Raspberry con el logo de raspbian
* Figura x.x: Dibujito de varias estaciones cliente conectadas a la Raspberry
* Figura x.x: Tanto dentro del cliente como del servidor poner que hay dentro con recuadros o logos o lo que sea.
* Figura x.x: La flechita de conexión al servidor para obtener ficheros es por FTP mientras que para acceder a la BBDD es la de siempre por el puerto 3306 o el que sea 3036…

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.

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 o .txt) con las últimas versiones del registro de windows que debería tener cada modelo de estación. (en algún lado tendre que hablar de que puede haber distintos modelos de estación – quizás en los conceptos básicos?). (Decir que estos fichero están pensados más bien para realizar futuras actualizaciones, de modo que actualizando el fichero del servidor se actualicen todos los clientes por sí mismos)
* Ficheros .txt con los informes de error generados tras la evaluación de los sistemas (Aún tengo que determinar si la agrupación de los informes será por estación o por categoría (Hardware, Software - Registro o Software - Contadores).)
* Ficheros (.cmd o .vb o como sea) 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 o de configuración con la información básica de la estación. (Este fichero se debería configurar antes de ejecutar el software por primera vez, de modo que se establezcan una serie de parámetros únicos para cada estación.)
* Un fichero (.xml o .txt) con la última versión del registro asociada al modelo de estación correspondiente en el momento de distribución de la estación. (Este fichero se irá actualizando según se actualice el servidor)
* Los ficheros con los scripts que ya hayan sido ejecutados en el cliente (cada vez que se detecte un fallo y se aplique un script para solucionarlo ese script quedará guardado en el cliente para usos futuros si hiciese falta).

## BBDD

La figura x.x muestra el diseño de la BBDD:

* Aquí diagrama UML de la BBDD según la siguiente estructura

Estación

* **ID**: int
* Estación
* Modelo
* VersiónRegistro
* Otros
* --------------------------------
* ID
* IP
* Empresa
* Otros datos (pantalla, lugar…)
* Modelo
* Version
* Otros datos de tipo software sobre los programas de la empresa

Hardware

* **IDEstacion**: int
* Componente
* Sensor
* **Identificador**: string (/…/…/…)
* Minimo
* Maximo
* Media
* Ultimo

Registro

* **Modelo**
* Version (ultima version asociada al modelo correspondiente)
* UrlDescarga

Futuras tablas (aún no se si las incluiré o no)

Contadores

* ID
* IDEstacion
* Categoria
* Contador (nombre)
* Valor
* ¿Instancia?
* ------------------------------
* ID
* Categoria
* Instancia
* Contador (nombre)
* Valor

Soluciones

* Error (nombre o conjunto de características/propiedades que definen un error)
* Solucion (Script|Manual|Ninguna)
* UrlDescarga (url de descargad el script a ejecutar)
* DescripcionSolucion (ya sea aplicar el script, cambiar la localización de la estación, resetear, instrucciones a ejecutar por el técnico, dejar reposar la estación…)

\*\* Puede que se identifiquen errores cuya solución no sea ejecutar un Script sino que haya que delegar en manos de un técnico para resolverlo de forma manual o que no tenga solucion

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

Como mucho hablar de las clases de la app

**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)

## 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.

# 5. Implementación

## Tecnologías y software empleado durante el desarrollo

Software base desarrollado en .NET

Programación WEB: PHP bajo Symfony 2.

## Librerías empleadas

OpenHardwareMonitor

## Aspectos generales de la implementación

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

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

## Puntos reseñables

## Problemas y soluciones destacables

Registros protegidos por el fabricante

# 6. Plan de pruebas e integración/implantación

## 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.

## 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. |  |  |
| 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

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)