# TITULO

Sistema de resolución automática de incidencias y generación de informes.

# NOTAS PARA EL DOCUMENTO

* Evitar títulos de nivel 3
* Revisar los tiempos utilizados a lo largo de todo el documentos para que sea consistente (pasado normal, pasado compuesto – haber .ado-, tercera persona etcetcetc)

# 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 es una pequeña empresa de Calahorra dedicada a proporcionar servicios de cartelería digital mediante la distribución y gestión de estaciones de publicidad. Dichas estaciones no son más que un dispositivo similar a un ordenador normal y corriente, pero más compacto y con una potencia menor pero suficiente para las tareas que desempeña la estación, entre las que se encuentran tanto mostrar publicidad a través de la pantalla conectada a la salida de vídeo como controlar que este proceso se lleve a cabo correctamente y avisar en caso contrario.

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 transparente para el cliente final.

## 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.

Los resultados obtenidos fueron los siguientes:

Entre las herramientas de pago se encontraron herramientas tales como [insertar nombres aqui] capaces de [insertar características aqui], pero ninguna se adaptaba del todo a las necesidades del proyecto, además de tener un coste bastante elevado para la empresa.

También se analizaron herramientas opensource como [insertar nombres aqui], las cuales proporcionaban [insertar características aqui] pero, al igual que las herramientas anteriores, no se adaptaban del todo bien.

Tras el estudio realizado se optó por realizar el proyecto desde cero, aunque visto el potencial de las herramientas opensource se optó por usar la librería de una de ellas: Open Hardware Monitor.

-Programas: guardados en favoritos pero porsia: Nagios, Cacti, Zabbix, OpenHardwareMonitor, OpenNMS creo, …

## Enumeración informal de objetivos (igual lo suprimo y dejo solo el alcance)

Estos mismos objetivos son los que he puesto en el alcance, pero en el alcance debo desarrollarlos un poco más.

* Desarrollar procedimientos de evaluación del sistema y detección de posibles problemas o acerías y generar informes a resolver por un técnico.
* Desarrollar un sistema de ejecución de scripts en estaciones cliente conectado a una base del conocimiento centralizada.
* Desarrollar la base del conocimiento que albergue todos los posibles problemas y soluciones a los mismos y las sirva a los clientes.

## Conceptos básicos

A lo largo del proyecto se van a utilizar repetidamente una serie de términos o conceptos que se describen a continuación:

* Estación:
* Registro:
* Contadores de rendimiento:

Estacion, OpenHardware, Registro/RegistryKey, PerformanceCounters/Contadores de rendimiento…

# 2. Gestión del proyecto

En el mensaje a Laureano decirle que las secciones están redactadas pero que no sé en qué orden ponerlas, exceptuando que metodología va inmediatamente antes de fases.

Tampoco se si poner la sección de actores o no.

En las fases: ponerlas en pasado o en futuro? Es decir: durante la fase x se realizó y || durante la fase x se realizará y?

## 2.1. Actores?

## 2.2. Alcance (esta un poco puto en sí)

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. Además, deberá ser capaz de generar informes para que un técnico de la empresa elabore soluciones a los fallos detectados. Las soluciones consistirán en scripts a ejecutar por la propia aplicación.

La aplicación deberá ser capaz ejecutar los scripts por sí misma y deberá encargarse de gestionar los scripts que hay en las distintas estaciones, [[de modo que compruebe si la solución o script a un error supuestamente nuevo en la estación ‘x’ está ya en otra estación ‘y’ antes de generar un informe para un técnico de la empresa – esto corresponde al funcionamiento de la app, véase, la parte de analisis]]. Para ello tendrá que haber una base de datos centralizada y conectada a todas las estaciones.

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, decida 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.
* Comprobar si existe una solución (script) al error detectado, ya sea en la estación en la que se detecta el fallo o en otra.
* Ejecutar scripts para solucionar los problemas detectados.

### Alcance del entregable de planificación anterior

Como he mencionado anteriormente, es necesario controlar que el producto que The New Ads proporciona funcione correctamente, al ser posible, evitando que sea el cliente final el que informe de fallos en la funcionalidad del producto. Es por ello que surge la necesidad de poder controlar el estado y funcionamiento de cada una de las estaciones proporcionadas a los clientes para poder prevenir posibles problemas y, en caso de que detectar algún problema intentar solucionarlo de inmediato.

De este modo, lo que se pretende con este proyecto es desarrollar el software necesario para realizar dicho control de las estaciones y, detectar y solucionar los problemas que tengan, todo ello sin la necesidad de que intervengan el cliente o algún técnico de The New Ads. Dicho de otra forma, la finalidad de este proyecto es: elaborar un sistema de detección y resolución de incidencias en estaciones, conectadas a una base del conocimiento centralizada, de forma automatizada.

### Objetivos del entregable de planificación anterior

El software a desarrollar deberá:

* Evaluar la estación cliente, detectar posibles errores y generar el informe correspondiente al error encontrado.
* 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. Para ello, un técnico se encargará de resolver los errores nuevos una vez y el sistema aprenderá dichas soluciones, de modo que si se repite el mismo error no haga falta un técnico para solucionarlo sino que el sistema detecte que ese error ya ha sido solucionado previamente (ya sea en la estación cliente en la que se detecta el error o en otra) y tome las medidas necesarias.

## 2.1. 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.

## 2.1. 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.

## Descomposición de tareas (descomposición, descripción y estimación de tiempos)

## EDT

## Gantt

# 3. Análisis

## Descripción de la aplicación (mezcla entre alcance y enumeración informal de objetivos)

QUE HACE LA APLICACIÓN – DESCRIBIR PROCESO DE LA APP CUANDO LA EJECUTAS

HABLAR DE LAS 3 COSAS QUE SE CONTROLAN: HARDWARE, SOFTWARE-REGISTROS, SOFTWARE-CONTADORES

La aplicación desarrollada para este proyecto se encargara de evaluar el sistema (tanto el hardware como el software), generar un informe con los resutlados 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.

Dos modos de ejecución:

* Modo 1: evaluación inicial
* Modo 2: evaluación en busca de fallos
* \* En algún lado hablar de que para que todo funcione bien es necesario supervisar que cuando se ejecute por primera vez lo haga correctamente

Scripts almacenados tanto en local como en servidor. Antes de buscar en el servidor y descargarlo mira a ver si esta en local para ahorrar el tiempo de descarga

## Casos de uso y otros diagramas

No hay casos de uso o a lo mucho podría poner que cada modo de ejecución es un caso de uso, y el segundo modo puede tener 3 caminos (todo OK, algo falla y no hay solución, algo falla y hay solución y se aplica la solucion) – De todos modos creo que es mejor hacer dos diagramas de actividad o interaccion, correspondientes a los dos modos de ejcucion que hay.

## Requisitos de la aplicación o proyecto

### Requisitos funcionales

### Requisitos no funcionales

# 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

## 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

## Seguimiento y control del tiempo

# 8. Futuras mejoras, mantenimiento del software y conclusión

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

# Conclusión

# Agradecimientos

# Bibliografía