# 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 estaciones1 gestionadas 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……..

# Introducción

## Contexto/Antecedentes

### La Empresa y su Negocio + Necesidad para la empresa de mi proyecto

The New Ads es una pequeña empresa de Calahorra que, dentro del sector de la informática, se dedica a proporcionar servicios de cartelería digital. En concreto, The New Ads proporciona tanto el hardware como el software necesario para poder reproducir publicidad en las pantallas de los clientes.

Cuando un cliente contrata los servicios de The New Ads lo que realmente hace es conseguir que sus anuncios se publiquen en las pantallas de otros clientes de The New Ads. De este modo, según aumenta el número de clientes, se crea una red de clientes o anunciantes

Este proyecto surge ante la necesidad de la empresa de dar soporte a toda la red de estaciones cliente de una forma rápida…

El producto que esta empresa proporciona consiste tanto en el hardware como el software necesario para poder realizar dicha publicidad. Por hardware se entiende el ordenador (a partir de ahora estación) con potencia suficiente y coste mínimo capaz de desempeñar las tareas requeridas, y por software se entiende tanto el sistema operativo personalizado y optimizado para las funciones requeridas como el software desarrollado por la propia empresa para gestionar los anuncios publicitados, mostrarlos por pantalla de la forma adecuada y controlar que todo funcione correctamente.

## Posibles soluciones de pago

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

## Conceptos básicos?

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

# Gestión del proyecto (planificación y metodología)

## Planificación

### Alcance

Redactar esto de forma adecuada pero básicamente es:

* 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.

### Fases - Formación Análisis Diseño Implementación Pruebas Documentacion

* Formación: Toma de contacto con la estación, con .NET c#, con OpenHardwareMonitor, con RegistryKey, con PerformanceCounter
* Análisis: requisitos funcionales y no funcionales, descomposición de tareas, realización de diagramas
* Diseño:
* Implementación:
* Pruebas:
* Documentación:

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

### EDT

### Gantt

### Actores/Participantes del TFG (igual sobra)

## Metodología de desarrollo (cual, descripción, por qué esa y no otra)

# Análisis

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

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

## Requisitos de la aplicación o proyecto

### Requisitos funcionales

### Requisitos no funcionales

# 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 simular una conexión real 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)

# 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

# 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

## Plan de implantación / plan de integración

# Seguimiento y control (comparación estimado vs real)

## Seguimiento y control del alcance

## Seguimiento y control del tiempo

# 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

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