Integración de Sistemas de Información en Estaciones de Servicio

Integration of Information Systems at Service Stations

Jésica Carballo Yanes
Departamento de Ingeniería Informática
Escuela Técnica Superior de Ingeniería Informática

Trabajo de Fin de Grado

La Laguna, 08 de julio de 2014

D. **Pedro Antonio Toledo Delgado,** con N.I.F. 45.725.874-B profesor Ayudante adscrito al Departamento de Ingeniería Informática de la Universidad de La Laguna

C E R T I F I C A

Que la presente memoria titulada:

"Integración de Sistemas de Información en Estaciones de Servicio".

Ha sido realizada bajo su dirección por D. Jésica Carballo Yanes, con N.I.F. 78.633.820-V.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 8 de julio de 2014.

Agradecimientos

Familia gracias por esos ánimos
Andrés gracias por tu apoyo incondicional
María de los Ángeles por su colaboración con el inglés

Resumen

En este proyecto se han analizado los flujos de información en Estaciones de Servicio, detectando necesidades de mejora en algunos de ellos. En las estaciones de servicio de turnos rotativos, en particular, y en otras muchas empresas que trabajan con turnos rotativos en general se necesita un control de datos. Estos datos son generados a la entrada a un turno, que resume el estado en que se comienza, y que suele coincidir con el estado al finalizar el turno anterior, datos producidos durante el turno en sí mismo, y datos generados a la salida, definiendo el estado en el que se deja la instalación. Este control de datos se hace muchas veces de forma no informatizada, y por tanto lenta, con alta probabilidad de cometer errores y con menor retroalimentación hacia la empresa.

Partiendo de esta necesidad se ha realizado el análisis diseño e implementación de una solución. Dicha solución se ha centrado finalmente en una herramienta para la gestión de partes de turno.

Para el desarrollo de esta herramienta se ha utilizado un entorno denominado CakePHP que ha permitido la agilidad del desarrollo y mejora de funcionalidades como por ejemplo la seguridad.

Palabras clave

Estación de Servicio, Sistemas de Información, ERP, back-office, front-office, TPV, CakePHP.

Abstract

On this project we have analyzed information flows in service stations, identifying needs for improvement in some of them. At the service stations with rotating shifts in particular, and at many other companies working with rotating shifts in general, data management is needed. The data is generated at a shift start, summarizing the state in which it begins, which is usually the same state at the end of the previous one. Furthermore, data is also produced during the shift itself, and finally, again when the shift finishes, data is gathered to define the state in which the facilities are left. This data management is not often done in a computerized form, and therefore management is a slow error-prone process, which leaves poorer feedback to the company.

Based on this need the analysis, design and implementation of a solution has been made. This solution is finally focused on a tool for managing shift reports.

For the development of this tool a framework, called CakePHP, has been used which has improved the agility of the development and included some extra features such as security.

Keywords

Service Station, Information Systems, ERP, back office, front-office, TPV, CakePHP.

Índice General

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

La información, son datos procesados que permiten tomas decisiones cuando se dispone de ella de una manera correcta.

En las estaciones de servicio la recogida de datos se realiza para labores administrativas, y muchas veces la forma es lenta y con alta probabilidad de errores. No hay un control en tiempo real para poder decidir y los errores tardan en solventarse cuando ocurren. Situación que se complica cuando se realizan turnos rotativos, por ello se ha detectado la necesidad de un control de turnos.

Este proyecto presenta una solución a la necesidad detectada, ofreciendo no sólo la digitalización de la información, que actualmente no se realiza en muchos casos. También ofrece integrar esta información en un sistema para poder controlarla, ayudar en la toma de decisiones y minimizar los errores.

Al principio el proyecto se había destinado a las estaciones de servicio, pero esta necesidad de turnos y de control se dan en otras organizaciones como hospitales, tiendas 24 horas, etc. El proyecto se ha orientado de manera flexible para que se pueda integrar en cualquiera de estas situaciones.

1.1 Sistema de Información

La definición de Sistema de Información se encuentra como: *"Conjunto de elementos orientados al tratamiento y administración de datos e información, organizados y listos para su uso posterior, generados para cubrir una necesidad u objetivo."[1]*

Los elementos se pueden clasificar en personas, datos y actividades o técnicas de trabajo. Estos generarán información más elaborada para que el usuario pueda trabajar con ella.

No todos los Sistemas de Información utilizan material informático, un ejemplo de Sistemas de Información es la biblioteca de un colegio, que no dispone de catálogos digitales y dispone de escasos ejemplares. La búsqueda de un libro se basa en dirigirse a un bibliotecario para mostrar el carnet de socio y el título buscado. Al realizar el préstamo del libro, el bibliotecario se queda con la ficha física que identifica el libro y apunta en dicha ficha el nombre del usuario que lo sacó (para asegurarse la devolución del mismo). Estas fichas muchas veces se pueden llenar, eso significa que el libro tiene demanda, por lo tanto, genera un conocimiento que el bibliotecario utiliza para conocer que libros son más demandados.

En cambio, un Sistema de Información Informatizado, es aquel que usa las tecnologías de la información, para realizar las labores de recogida y tratamiento de los datos de esta manera generan información de forma más rápida y clara.

1.2 ¿Por qué un Sistema de Información Informatizado en una Estación de Servicio?

Una estación de servicio utiliza muchos datos a lo largo de la jornada laboral, esta cantidad de información crece y si se quisiera hacer un análisis avanzado de los mismos, se podría superar la capacidad del software habitual para poder ser capturada, procesada y gestionada en un tiempo razonable, es lo que se conoce como “Big Data”[2].

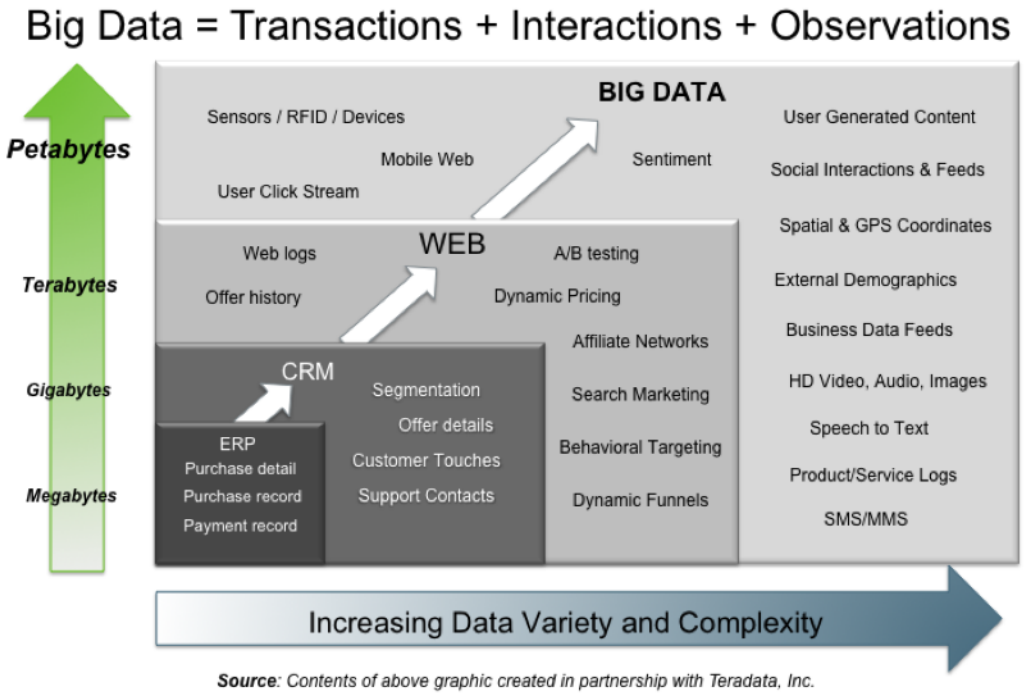

Figura 1.2.1 Big Data

Si los datos recogidos se procesan con un buen sistema de información podremos obtener una información valiosa para los usuarios y gerentes. Esta información nos permitirá tomar decisiones, obtener valoraciones de negocio, mejorar la competitividad, favorecer la comunicación entre empleados, etc.

En resumen, esta información puede ser analizada y utilizada para crear nuevo conocimiento.

Según las estadísticas un 98%[3] de las empresas ya usan las tecnologías de la información para mejorar su competitividad y productividad.

Un ejemplo de mejora de funcionamiento de la organización, se podría dar en una Estación de Servicio, donde el sistema de información nos permitiría saber si se debe realizar un pedido en cuanto los niveles de combustibles de los depósitos sean bajo, en este caso el sistema informaría al responsable. Una vez recibida la "alerta", el personal autorizado realizará la correspondiente orden de pedido

o incluso el sistema podría realizar una orden automática de pedido.

1.3 Solución aportada por este proyecto

Los sistemas normalmente presentes en una estación son los Terminales Punto de Venta o TPV, BackOffice o gestión, los cuáles se explicarán en el siguiente apartado. También se pueden encontrar: el concentrador de comunicaciones, surtidores electrónicos, sondas electrónicas de nivel en tanques, monolito electrónico, máquinas de lavado automático, gestión de stock, etc.

Este proyecto trata de mejorar la gestión de las estaciones de servicio, ayudando a informatizar y automatizar la recogida de datos que se realizan de forma manual e incluirlas en el sistema de información.

Los datos que se quieren almacenar son, los datos iniciales al abrir el turno, que coincidirán con los finales del turno anterior, los datos que se generen durante el turno y los datos al finalizar el turno, los que definirán el estado en el que se deja la instalación.

Algunos ejemplos de recogida de datos que han sido mencionados en el párrafo anterior son:

- Lectura de contadores de surtidores: Permite conocer los litros vendidos por cada una de las mangueras instaladas en la estación.
- Niveles de los tanques: Permite conocer el stock de combustible disponible.
- Recuento de caja: Debe anotarse la cantidad inicial de la caja, la final así como las entradas y salidas.
- Seguimiento de lavados: Llevar un control de los lavados realizados durante el turno.
- Otras operaciones de seguimiento: Cualquier variación de recursos cuyo control deba ser requerido, como por ejemplo determinados

artículos de mayor valor que recomiende llevar un mejor control, y que no se realice de forma automática.

El seguimiento de estas tareas en turnos rotativos complica su gestión. Los cambios de turno se realizan en poco tiempo, lo que podría dar lugar a errores.

Los vendedores deben rellenar el parte de turno que pasará al gestor. Este gestor puede no volver a coincidir con el vendedor durante su turno, por tanto en caso de error la notificación se retrasa. Si hay un error en los datos al comprobar varios partes, será más difícil reconocer donde se ha cometido. Y la comunicación del error podría retrasarse por días, cuando vuelvan a coincidir vendedor y gerente.

Un error en la toma de datos puede derivar en más errores, por ejemplo un mal seguimiento del combustible vendido puede afectar al cierre de caja ya que las ventas calculadas no corresponderán con los cobros.

Este enfoque va un paso más allá de la digitalización del documento, se trata de gestionar los datos, que se producen en todo el ciclo de trabajo de un parte de turno. Estos datos permiten varias tareas como control de los partes de turno partiendo de datos anteriores, ayuda a los vendedores en la gestión de partes, ayudar al gerente en la toma de decisiones y el control de la propia estación, etc.

Capítulo 2. Antecedentes y estado del arte

2.1 Soluciones que existen actualmente implantadas en estaciones de servicio

Los sistemas de información informatizados que se encuentran en las diversas estaciones de servicio suelen constar de back-office y Terminal Punto de Venta (TPV).

El back-office es lo conocido vulgarmente como "la trastienda", y es la parte que no está a la vista de los clientes, donde se llevan las gestiones de logística, etc. En este caso, son los sistemas de planificación de recursos empresariales conocidos por sus siglas ERP (Enterprise Resource Planning), un ejemplo de estos programas son SAP y OPENERP.

La definición de Terminal Punto de Venta (TPV) se encuentra registrada como: *"dispositivo y tecnologías que ayudan en la tarea de gestión de un establecimiento comercial de venta al público que puede contar con sistemas informáticos especializados mediante una interfaz accesible para los vendedores" [4].*

Inicialmente, antes de focalizar el proyecto en una solución para la gestión de partes de turno mencionada en el apartado anterior, se realizó un análisis de la situación de los sistemas de información en Estaciones de servicio. Para ello, se hicieron pruebas de campo, recorriendo varias estaciones en las que se realizó una encuesta para conocer de primera mano la opinión sobre sus sistemas de gestión.

Las encuestas que se pueden consultar en el **APENDICE A**, se ha obtenido como resultado que se usan programas de gestión administrativa y software para el control de surtidores, así como programas para la manipulación de los monolitos. Pero se aprecia un notable hueco en programas que controlen los turnos y

no sólo de forma digital sino integren esta información en un sistema para el control de la información.

Con la información recabada en las encuestas se concluye que el software propietario de la cadena se utiliza en menor escala, por los problemas de redes, y en algunos casos únicamente para el control de puntos en promociones. Este último caso se da cuando en las estaciones disponen de tarjeta de puntos, las estaciones se conectan a la plataforma de la cadena sólo para acumular o canjear dichos puntos, realizando las labores de gestión con otro software.

Los propietarios prefieren las terceras empresas para proveer software de gestión en las estaciones, estas les ofrecen un trato más personalizado y una atención inmediata en caso de problemas.

Los ejemplos de sistema de gestión que se han citado en las encuestas han sido los siguientes:

TECNIREG empresa dedicada al desarrollo del software que desarrolla e implanta el programa multifuncional "TEIDE", el cual incluye lo siguiente:

- Control de mermas de combustible
- Lecturas de Contadores de Surtidores y Mangueras
- Control de Pérdidas de Combustible
- Recepción de Tickets en tiempo real
- Posibilidad de modificación de tickets emitidos
- Posibilidad de modificación de forma de cobro de tickets emitidos
- Posibilidad de modificación de transacciones realizadas en el TPV
- Control de operadores (Entrada/Salida, Horas trabajadas)
- Emisión de facturación vía correo electrónico.
- Emisión de adeudos domiciliados de clientes en formato Q19

- Importación de compras vía email desde la central de compras.

FULLGAS empresa dedicada a dar soluciones tecnológicas a las estaciones de servicio, centrándose en el control de los tanques con sensores inteligentes, pérdidas de combustible. La mayoría de estaciones hacen uso de este software para labores de mantenimiento y uso de los surtidores, no lo utilizan para ningún otro fin.

COPERMÁTICA empresa de software que desarrolla el programa Alf@ ERP Hidrocarburos, ofreciendo:

- Stocks y Facturación
- T.P.V. Táctil integrado con pista
- Pedidos de compras
- Medios de pago
- Supervisión cámaras IP integradas
- Perfecta trazabilidad de los artículos en el almacén
- Facturación inmediata de cualquier albarán
- Personalización de precios por grupos de clientes
- Asistente gráfico del parte del turno
- Créditos locales / tarjetas de crédito
- Control de Stocks en línea
- Cobros y pagos por caja
- Prepago, postpago y desatendido
- Gestión de empleados
- Rápida parametrización

Capítulo 3. Especificación de requisitos y análisis de la solución

3.1 Requisitos

Pensando en la generalización de la solución diseñada para el proyecto del proyecto se han estipulado unos requisitos que puedan ser utilizados en cualquier estación de servicio.

3.1.1 Requisitos funcionales

Se desea almacenar datos para poder llevar un control y seguimiento de los mismos, como por ejemplo el seguimiento de liquidez en caja, las ventas realizadas y quien es el usuario que ha realizado la introducción de datos o modificación, para ello se han fijado los siguientes requisitos.

Control del acceso de usuarios al sistema. Se deberá implementar una funcionalidad de control, que garantice el acceso al sistema del personal autorizado. Esta medida será realizada mediante la identificación con su nombre de usuario y contraseña. Existirán tres tipos de usuario o roles en el sistema (empleado, gerente y administrador).

Gestión del parte de turno. Los empleados al empezar su turno abren un nuevo parte que se queda en modo borrador, en el que se importarán los datos de cierre del turno anterior como punto de partida. Una vez editado y firmado por el empleado (acción firmar), pasará a ser validado por un gestor que se identifique en la aplicación.
En el parte de turno el empleado podrá:

- Crear un nuevo parte.
- Editar un parte, previamente creado.

- Firmar un parte, previamente creado

En el parte de turno el gerente podrá:

- Validar un parte de turno.

Para garantizar el almacenamiento y tratamiento de datos, se debe realizar un seguimiento del flujo de trabajo del parte, desde que se crea y edita, hasta que se valida. Lo que conlleva una mantener una copia de los datos de cada etapa del flujo de gestión definida en el workflow.

En la figura 3.1.1.1 se muestra una imagen con un ejemplo de una plantilla de un parte de turno.

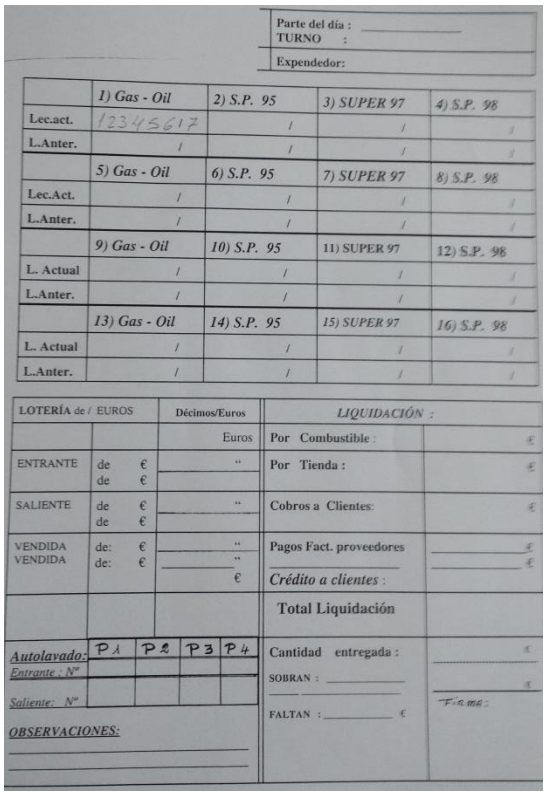

Figura 3.1.1.1 Ejemplo parte de turno en papel

Panel de administración de usuarios. La aplicación contará con la opción de agregar, borrar o modificar los usuarios de la misma, dicha acción será realizada por los roles de administrador y gerente.

Gestión de incidencias. La aplicación contará con un sistema para la gestión de incidencias. Dichas incidencias se crearán en la elaboración del parte, para permitir la notificación de errores en caso de producirse. También pueden crearse por medio de los gestores y vendedores, para realizar comunicaciones de errores o averías que puedan ocurrir.

3.1.2 Requisitos no funcionales

Desarrollo web. Se deberá desarrollar como una aplicación web. Esta aplicación se desarrollará en entorno web para facilitar la portabilidad de la misma.

Seguridad. Medidas en la base de datos que garantice la seguridad y confidencialidad de los datos. Para evitar accesos al sistema de personal no autorizado y garantizar la Ley Orgánica de Protección de Datos (LOPD)[5].

3.2 Alcance del proyecto

Se ha realizado el análisis de los requisitos y este proyecto abarcará las siguientes partes:

- Gestión del parte de turno: listado, creación, edición, firmado y validación.
- Gestión de usuarios: listado, creación, borrada y edición.
- Control de acceso: login y logout.

Las incidencias aunque puedan ligarse a la creación de parte se posponen a desarrollos posteriores de la herramienta quedando fuera su implementación del ámbito del presente proyecto.

3.3 Casos de uso

Especificamos los casos de uso que se han utilizado para poder explicar la comunicación y el comportamiento del sistema con la iteración de los usuarios.

A continuación vemos los diferentes diagramas de caso de uso.

- Control de acceso

Explica cómo acceder al sistema: los roles de vendedor, administrador y gerente mediante identificación pueden acceder.

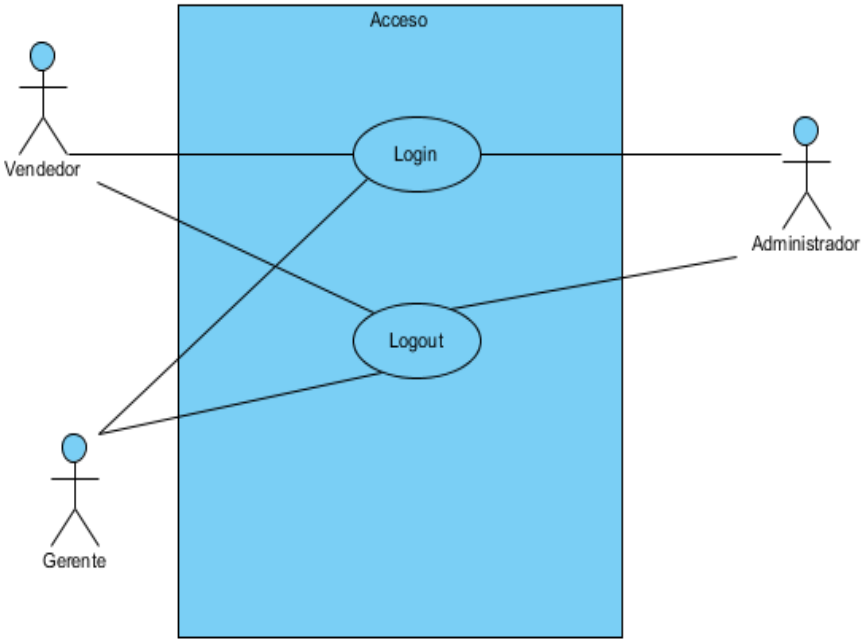

Figura 3.2.1 Caso de uso. Acceso

- Partes

Es funcionalidad principal de la aplicación, el vendedor accede a listar los partes desde ahí puede firmarlos y editarlos. También puede crear un nuevo parte.

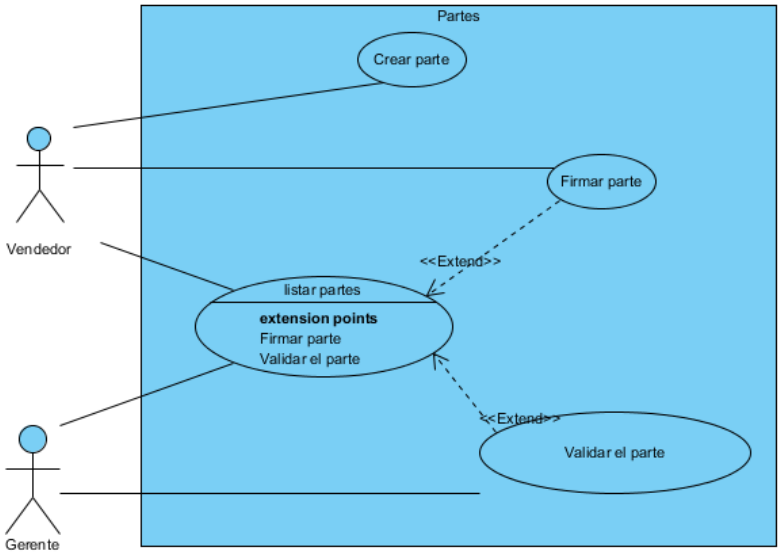

Figura 3.2.2 Caso de uso. Partes

- Gestión de usuarios

Estas funcionalidades estarán destinadas a los gerentes y administradores, los cuales serán los únicos que puedan crear usuarios, editarlos y borrarlos.

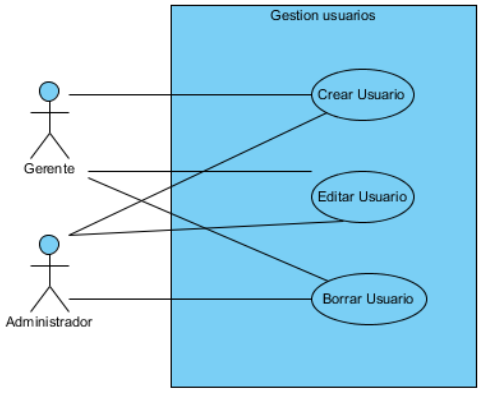

Figura 3.2.3 Caso de uso. Gestión de Usuarios

Capítulo 4. Herramienta para el desarrollo, cakephp

4.1 Conociendo a CakePHP

CakePHP es un framework basado en el paradigma de la arquitectura "modelo vista controlador (MVC)[6]", es de código abierto y se distribuye mediante la licencia MIT[7].

4.2 Recomendaciones para su uso

CakePHP es un framework que brinda muchas ventajas, entre ellas:

- Agilidad para el desarrollo. Permite centrarse en las características de la aplicación.
- Generación de código. Permite generar la estructura de la aplicación de una forma rápida.
- Limpieza de código.
- Compatible con PHP5.
- Agilidad en la validación de datos.
- Fácil y rápida integración con la base de datos, contiene CRUD (Create, read, update and delete), lo que permite agilidad en el desarrollo. Esta característica es aplicable siempre que el proyecto se adapte al estándar de CakePHP, lo que significa seguir las pautas establecidas a la hora de definir las tablas y los campos que contienen.
- Ayuda para definir la seguridad de la aplicación, se puede limitar el acceso al contenido de forma rápida y ágil, mejorando notablemente la seguridad.
- Licencia que permite reutilizar el software tanto para ser software libre como para ser software no libre, permitiendo no liberar los cambios realizados al programa original.

Existen más frameworks como por ejemplo Zend FW[8] y Symfony[9], que sirven para el desarrollo web.

Cakephp y Symfony contienen operaciones CRUD (Create, Retrieve, Update y Delete) asociadas a patrón Active Record y mapeado de objetos a bases de datos relacionales (ORM), mientras que Zend FW no dispone de ellas. Las herramientas de prueba y depuración que ayudan al desarrollo están en Cakephp y Zend FW pero no en Symfony. Además Cakephp tiene almacenamiento en caché de configuración de las aplicaciones, funcionalidad no disponible en los otros.

Finalmente por las características mencionadas se decidió que CakePHP, era una opción válida.

Capítulo 5. Diseño e implementación

Se ha realizado el proyecto siguiendo el paradigma antes explicado de modelo, vista y controlador, proporcionada por CakePHP.

A continuación explicamos las principales características que han influido en el proyecto al aplicar el MVC bajo las directrices del framework CakePHP.

5.1 Diseño e implementación de la Base de Datos

La base de datos ha sido la parte fundamental de este proyecto, se ha realizado pensando en la escalabilidad y ampliación de la base de datos. Si la aplicación cambia tanto sus contenidos como su formato se puede adaptar a este diseño.

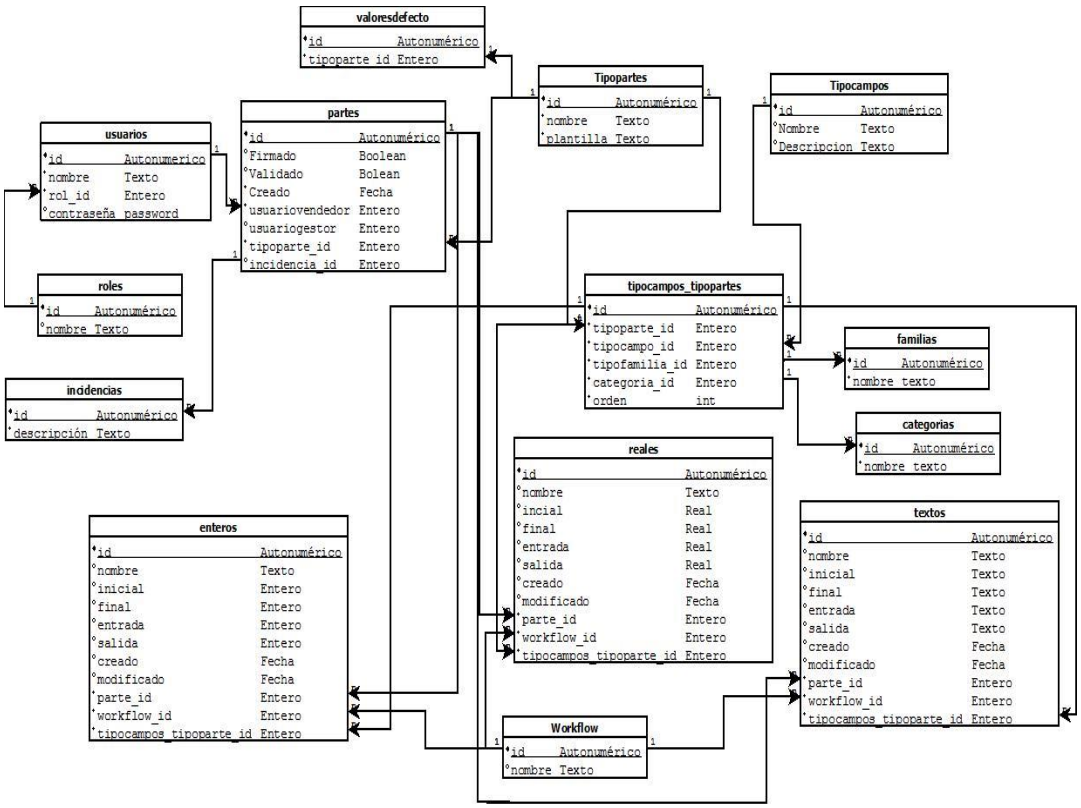

Figura 5.1.1. Esquema de las relaciones de la base de datos

A continuación se explican las tablas involucradas en la aplicación y las funciones principales de cada una, se debe tener en cuenta que sus nombres siguen el estándar de CakePHP.

Usuarios: Serán los que usen la aplicación desarrollada.
Roles: Determinará el rol de cada usuario, útil para la gestión de permisos.
Partes: Tabla para gestionar el estado del parte, así como los usuarios que interactúan y las fechas.
Incidencias: Esta tabla se crea con la posibilidad de vincularla al flujo de trabajo del parte, en ella si ocurre algo durante el workflow, puede quedar registrado, enviarse un mensaje, etc. Estas funcionalidades quedan fuera del ámbito del problema a tratar, como se mencionó en el alcance del proyecto.
Tipopartes: Permite fijar un formato de parte y un nombre de plantilla de datos, cuando el usuario use ese tipo de parte se le vinculará a la vista que apunta "plantilla".
Valoresdefecto: Fija el valor del tipo de parte a utilizar cuando se cree un nuevo parte.
Tipocampos: Sirve para definir los tipos de campos que necesitemos. De esta manera dependiendo del tipo en la vista se visualizarán unos campos u otros. Por ejemplo si tenemos tipo campo 2, sólo nos mostrará el campo entrada-salida.
Categorías: Almacena los tipos de datos que existen para guardar (enteros, reales, etc). Necesario para la creación de nuevas entradas, sabiendo la categoría a utilizar se inicializa la tabla correspondiente.
Familias: Sirve para agrupar los datos cuando se muestren. Por ejemplo familia surtidor, lavado, etc.
Tipocampos_tipoparte: Tabla fundamental en la aplicación, permite que la misma sea dinámica. En ella se fijan las relaciones que definirán los campos que

contendrá un tipo de parte en específico, también como serán dichos campos (categorías) y en qué familia se agruparán. En este parte se define el orden de los campos, que se deben asignar dentro de la familia.

Workflow: Es la encargada de guardar todos los pasos del ciclo de trabajo.

Las tablas encargadas de guardar los datos son **enteros**, **reales** y **textos**.

Estas tablas contienen las siguientes columnas: inicio, fin, entrada, salida. Esto es porque dependiendo del tipo de dato puede ser necesario el uso de los cuatro campos, dos o uno. Por ejemplo la contabilización de la liquidez necesita contar con un dato de inicio (cantidad al abrir el turno), un dato final (cantidad final) y los datos de entrada salida (Ingresos y pagos) que ha sido realizados, en este caso se usarán los cuatro campos de categoría real (son números reales los que se almacenan). Otro ejemplo es el contador de los surtidores, se trata también de un dato de categoría real, pero en este caso sólo necesitaremos rellenar los campos inicio y final.

Cuando el administrador encargado de la creación del formato de parte define el tipo de campo también crea la vista, mostrando tanto para la visualización como el guardado sólo los campos necesarios, quedando el resto como nulos en la tabla.

Esto puede suponer un gasto de memoria, pero con los avances en capacidad de almacenamiento este gasto se considera mínimo y esta opción permite la agilidad y rapidez de la aplicación.

Todas ellas tienen el mismo funcionamiento, guardar el dato vinculado al parte y el paso dentro del workflow, cada una del formato que corresponda (entero, real o texto).

5.1.1 Base de datos y CakePhP

La base de datos se comunica con la aplicación a través del modelo, cada tabla corresponde a un modelo de la aplicación.

En cada modelo se definen las reglas de validación de los datos, lo que ayuda a la integridad de los mismos, por ejemplo no dejar datos en blanco.

Uno de los casos en los que se ha utilizado esta validación es al realizar el login, ya que los campos usuario y contraseña no pueden estar vacíos.

En este ejemplo, vemos como en el modelo usuarios se configuran las reglas de validación para que el campo usuario y pasword no puedan estar vacíos cuando se creen.

```
/**
* Validation
*/
     public $validate = array(
           'username' => array(
            'required' => array(
                'rule' => array('notEmpty'),
                'message' => 'Se require un nombre'
            )
           ),
           'password' => array(
            'required' => array(
                'rule' => array('notEmpty'),
                'message' => 'Se require una contraseña'
            )
           )
     );
```

Si se intenta acceder con algún apartado obligatorio sin rellenar, el sistema emite un mensaje avisando que se ha dejado un campo en blanco.

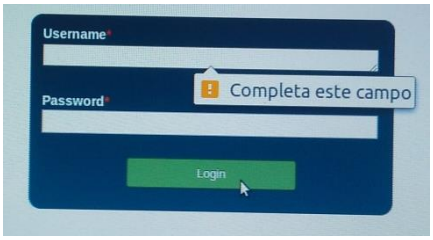

Figura 5.1.1.1. Intento de acceso con datos en blanco

También se definen los tipos de relaciones que existen entre los distintos modelos, para que la aplicación funcione de forma correcta. En esta aplicación una tabla que contiene múltiples relaciones es "tipocampostipoparte" (ver **Apendice B**).

Cake nos permite varios tipos de relaciones, estas se resumen en la tabla siguiente:

Relación	Tipo de Asociación	Ejemplo
one to one	hasOne	A user has one profile.
one to many	hasMany	A user can have multiple recipes.
many to one	belongsTo	Many recipes belong to a user.
many to many	hasAndBelongsToMany	Recipes have, and belong to, many ingredients.

Tabla 5.1.1.1. Relaciones existentes en CakePHP

En CakePHP se debe tener en cuenta los estándares en el diseño de las tablas, en sus nombres como en los campos, para facilitar las relaciones y que puedan identificarlas.

5.2 Diseño e implementación del controlador

La funcionalidad se corresponde a la parte de controlador de la aplicación, encargado de enviar los datos a las vistas y recoger datos. Es el supervisor de todas las actividades.

En la aplicación se han desarrollado varios controladores. Estos son:

a) Usuarios

Permite la gestión de los usuarios. En este controlador se definen los métodos de crear, editar y borrar usuario. Está vinculada con Roles, ya que cada usuario definido debe tener un rol asociado.

b) Roles

Permiten la creación, edición y borrado de roles.

c) Partes

Controlador principal de la aplicación que permite la creación, edición, firmado y validado.

- Creación: Cuando un usuario con rol vendedor crea un parte, la aplicación genera la estructura para poder almacenarlo tanto a los datos del parte como sus datos relacionados. Esto se consigue por medio de consultas diseñadas específicamente para esta aplicación.
- Edición: Un vendedor después de crear un parte puede editarlo, para ello el sistema actualizará la estructura de datos creada con anterioridad.
- Firmar: El vendedor puede firmar el parte, el sistema actualizará el estado del parte. No interfieren los datos relacionados.
- Validar: Este método precisa de una copia previa del parte, por tanto el gerente tendrá primero la opción de crear copia para validar y luego podrá editar esa copia y realizar la validación.

d) Controladores Enteros, Textos y Reales.

Permite la creación y edición de los datos, se vinculan a partes, como se ha explicado en el apartado anterior. Esto quiere decir que cuando se inicialice un parte se creará una estructura para almacenar los distintos campos, que se editarán cuando el vendedor edite dicho parte. También para poder realizar una validación y que no se sobrescriban los datos anteriores se crean copias de los.

El controlador hace posible la comunicación entre varios modelos, pasando los parámetros si fuera necesario. De esta forma se realizan las consultas de edición, selección y actualización de datos relacionados.

5.2.1 Uso del Controlador en el proyecto con las herramientas CakePHP.

Mientras se desarrolla la aplicación en CakePHP v2 se puede optar por instalar un pluggin denominado DebugKit[10]. Este es una barra de herramientas que se instala en el HTML representado y proporciona herramientas de depuración como aviso de errores y contenidos de variables.

En la imagen siguiente, que corresponde a una captura de la aplicación mientras se desarrolla, se muestra los contenidos de las variables del parte.

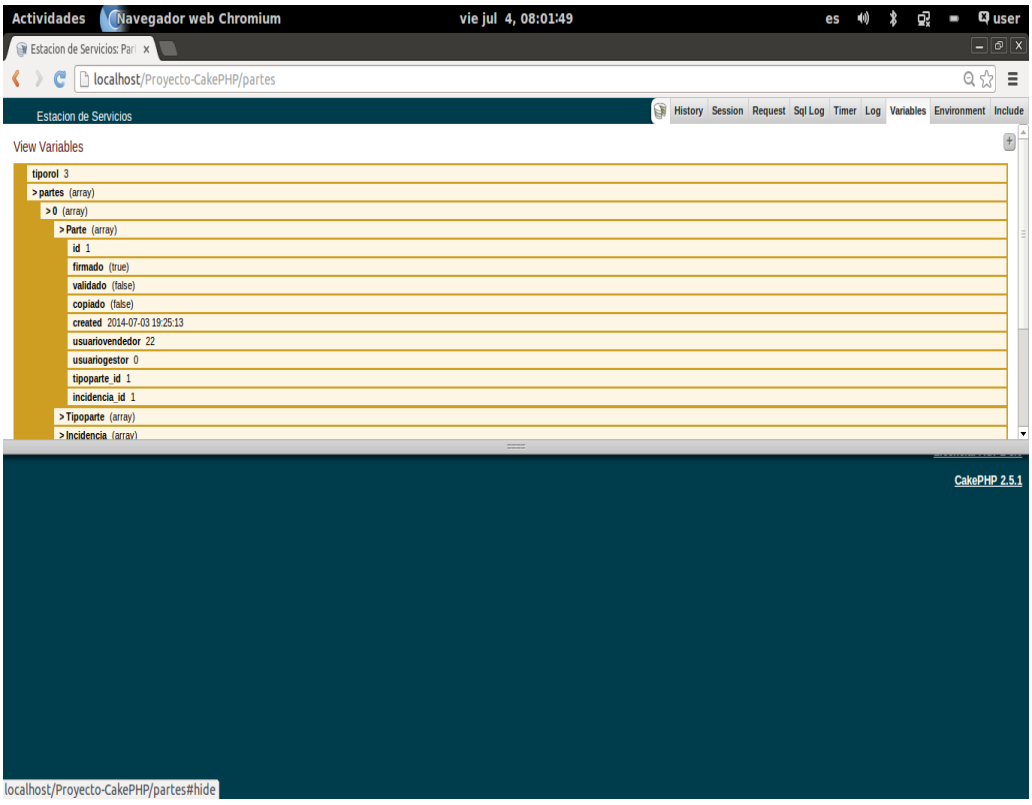

Figura 5.2.1 Uso del DebugKit de Cakephp en la aplicación.

El controlador que permite asignar los permisos para acceder a los diversos apartados dependiendo del usuario, es el encargado de la seguridad.

En la aplicación se implementaron varias limitaciones de acceso, una es el caso de que si un usuario no ha iniciado sesión no puede acceder a través

de la barra de direcciones al apartado partes, para ver los detalles de cómo se definen estos permisos se puede consultas el **Apendice B.**

En la imagen siguiente muestra el mensaje de error si intenta acceder sin identificarse.

Figura 5.2.2 Intento de acceso al apartado partes sin identificarse.

En Cakephp los métodos que se encuentran en los controladores se denominan acciones. Cada acción que encontramos en el controlador si seguimos el estándar de CakePHP debe corresponderse con una vista, esto se debe a que se utiliza el método render del controlador que es llamado automáticamente al final de cada acción del controlador pedida. Este método lleva a cabo toda la lógica de la vista, coloca la vista (view) dentro de su diseño (layout) y lo sirve de vuelta al usuario final.

Por otro lado tenemos funcionalidades como por ejemplo "crear nuevo parte" cuya vista asignada por defecto no es necesaria y podemos modificarlo para que solamente se realice la lógica que necesitamos y se envíen los resultados a la acción correspondiente. (Ver **Apendice B**).

5.3 Diseño e implementación de interfaces

Este proyecto consta de varias interfaces, a continuación se explican cada una de ellas:

- Login: una interfaz común de acceso para todos los usuarios.
- Índice de usuarios: Para la gestión de parte se muestra una interfaz común con opciones distintas según el rol de cada uno.
- Edición de partes: La edición de partes es una interfaz dedicada al vendedor, el cuál es el único que puede editarlos.
- Validación de parte: Es una interfaz única para el gestor que valide el parte. Estas interfaces son las creadas por el administrador dependiendo del formato de parte que haya establecido.
- Ver detalles del parte: Esta interfaz permite ver los detalles principales de cada parte.
- Administrar Usuarios: En esta interfaz se puede listar los usuarios del sistema, editarlos, crear nuevos y borrarlos.

5.3.1 Características de CakePhP aplicadas a las interfaces del proyecto.

En CakePHP las interfaces se agrupan por carpetas que se denominan con el nombre de cada controlador .En cada una de ellas encontramos distintas vistas que se corresponden con acciones de ese controlador.

Estas se llaman directamente con el método render, mencionado en el apartado anterior, pero después de cada lógica de las acciones, puedes utilizar "render" para especificar un fichero de vista alternativo indicando un nombre de acción en el controlador usando "$action". De esta manera la acción editar parte se puede utilizar para distintas vistas, ya que cada una

de ellas será un formato distinto de datos con un diseño personalizado si cambiamos el modelo de parte inicial a otro.

Para realizar una llamada a otra vista simplemente tenemos que incluir el siguiente código dentro de la acción del controlador:

```
// Render the element in
/views/elements/ajaxreturn.ctp
$this->render('/elements/ajaxreturn');
```

Para asignar un diseño al proyecto, CakePHP nos permite personalizar los entornos que deseemos, para ello disponen dentro de las vistas una carpeta denominada “layout”, donde podemos almacenar el modelo de estructura que queremos (podemos disponer de varios si fuese necesario). Esta estructura se vincula a un archivo css, que deseemos con el formato deseado.

En este proyecto se han creado varios layouts personalizados para que el usuario pueda ver los datos de una forma clara como es el login, edición y validación de datos, etc.

Capítulo 6. Interactuando con la aplicación

En el capítulo anterior se ha hablado del desarrollo e implementación de la herramienta, ahora se explicará en el funcionamiento de la aplicación.

Todos los usuarios tienen la pantalla de login en común, esta pantalla estará disponible al acceder a la aplicación, de esta manera el usuario se identifica y podrá acceder a la pantalla de partes. En la misma se le dará la bienvenida mostrarán las opciones correspondientes en el listado de partes (si lo hubiese) y las acciones según su rol.

Figura 6.1 Login de la aplicación.

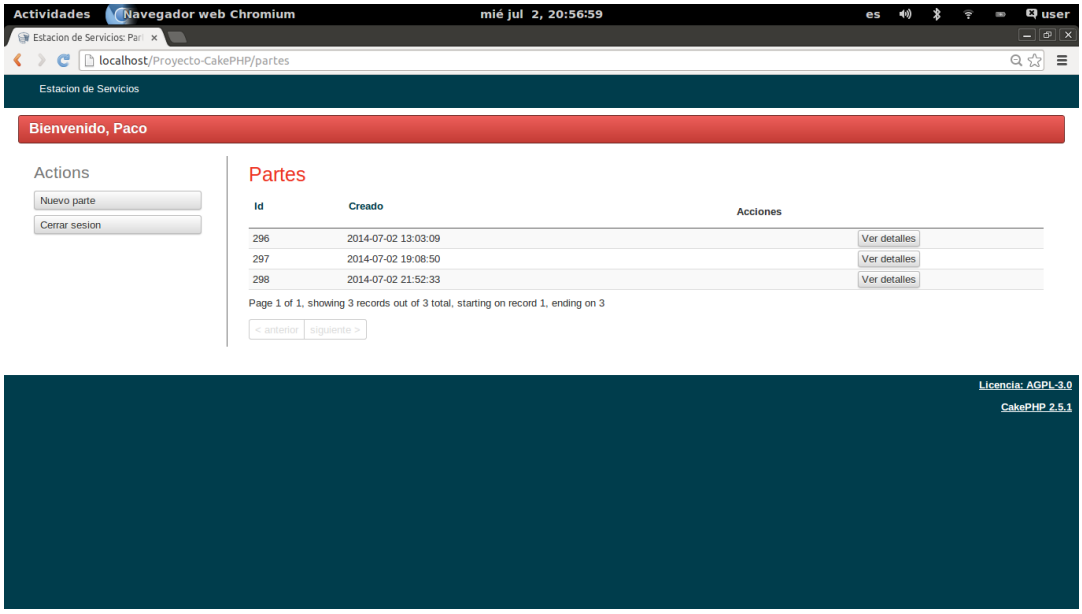

Figura 6.2 Bienvenida de la aplicación e inicio.

A continuación se muestra la aplicación a través de los perfiles definidos en los capítulos anteriores: administrador, vendedor y gestor.

6.1 Administrador en la aplicación

El administrador en la aplicación tiene una labor fundamental de implantación de la herramienta, actualización y mantenimiento.

Estas labores son derivadas de la flexibilidad de la aplicación, ya que una vez implantada, las necesidades pueden cambiar. Un ejemplo es una estación con un nuevo túnel de lavado sobre el que se quisiera llevar un control manual del uso, este cambio implicaría únicamente un nuevo formato de parte, con el contenido y vista correspondiente. Por esta razón las labores del administrador será la de crear los campos necesarios en un nuevo formato de parte de turno, crear la vista para la representación de dicho formato y

tener acceso a la base de datos, para realizar los cambios.

A su vez el administrador tiene acceso a la gestión de los usuarios, por lo que realizará labores de alta, baja y modificación si fuera necesario.

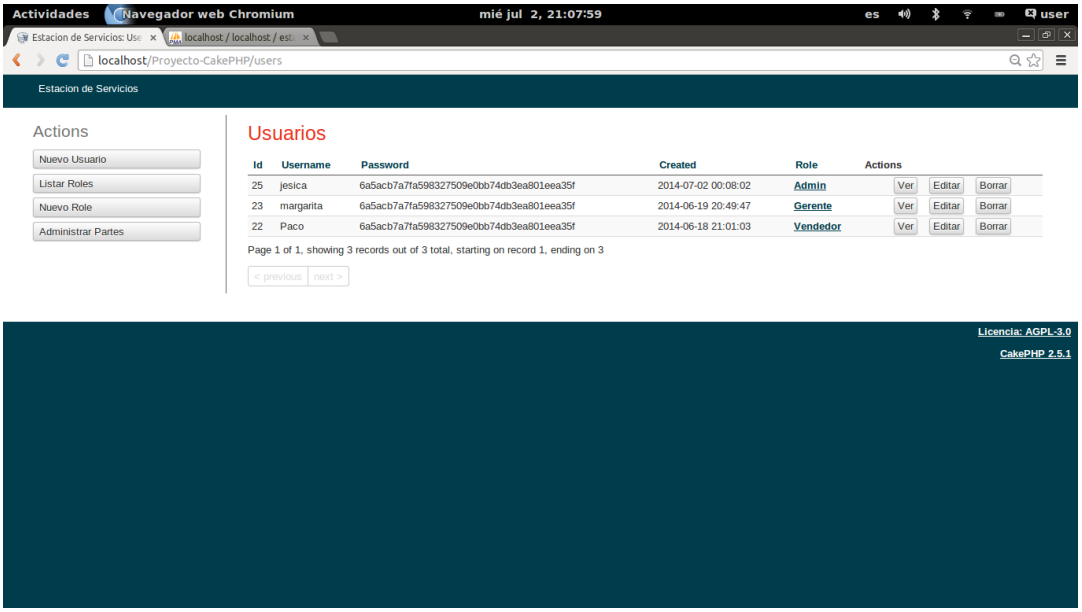

Figura 6.1.1. Ventana de gestión de usuarios.

6.2 Vendedor en la aplicación

El vendedor será el encargado de crear los partes editarlos y firmarlos para que el gerente pueda realizar la validación del mismo.

Cuando empiece el turno de trabajo el vendedor tendrá que inicializar el parte, de esta manera se creará la estructura definida por el administrador.

Este proceso se realiza de forma transparente al usuario, este sólo verá un mensaje de parte creado.

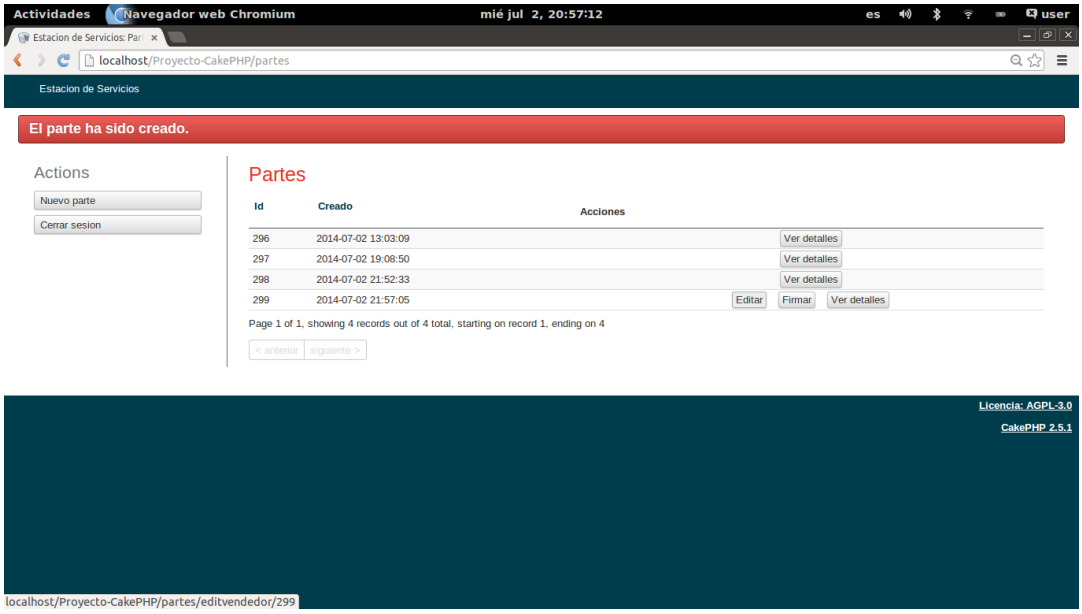

Figura 6.2.1. Listado de partes y mensaje al crear uno nuevo.

Una vez creado el parte, el usuario podrá editarlo, para ello pulsa sobre editar y se le mostrará una nueva página con el formato del parte y los campos necesarios.

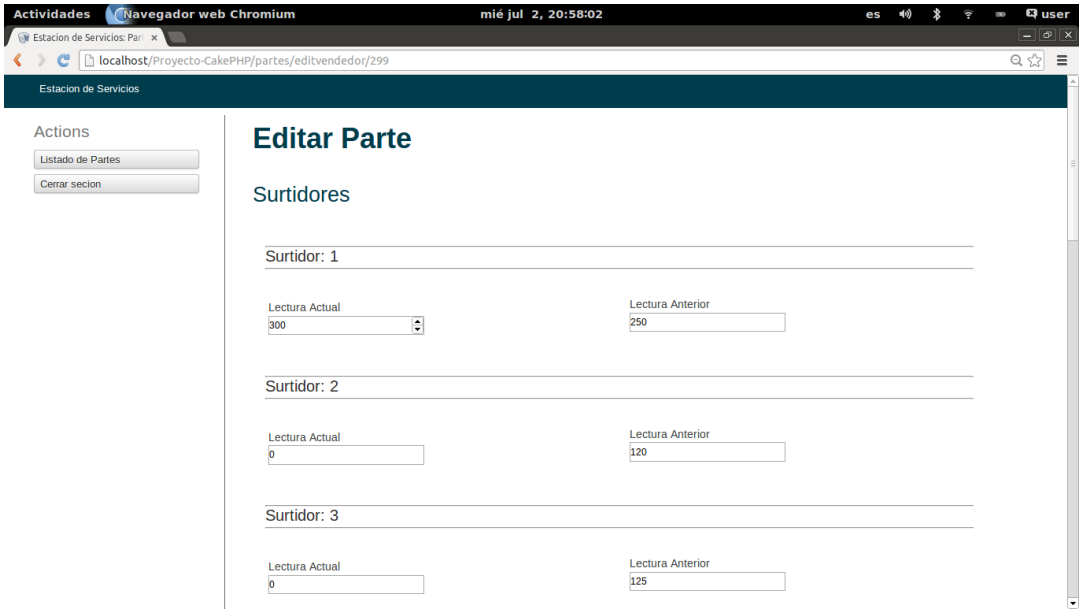

Figura 6.2.2.Edición de parte, sección superior.

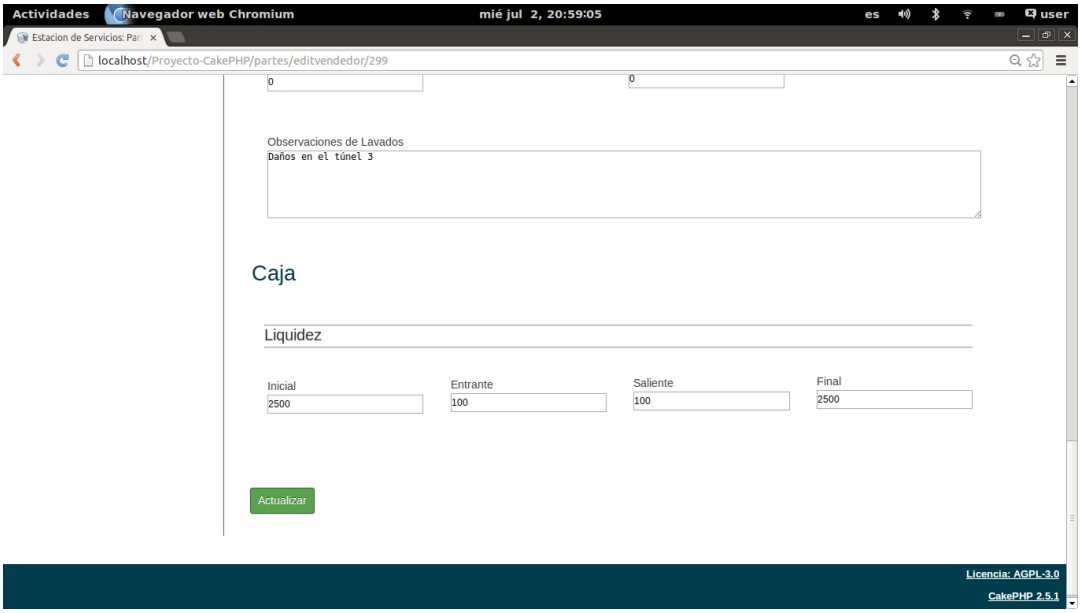

Figura 6.2.3.Edición de parte, sección inferior.

En el listado de partes también le aparece la opción de firmar, esta opción debe realizarse una vez editado, por lo tanto se le avisará al usuario de que no se puede anular.

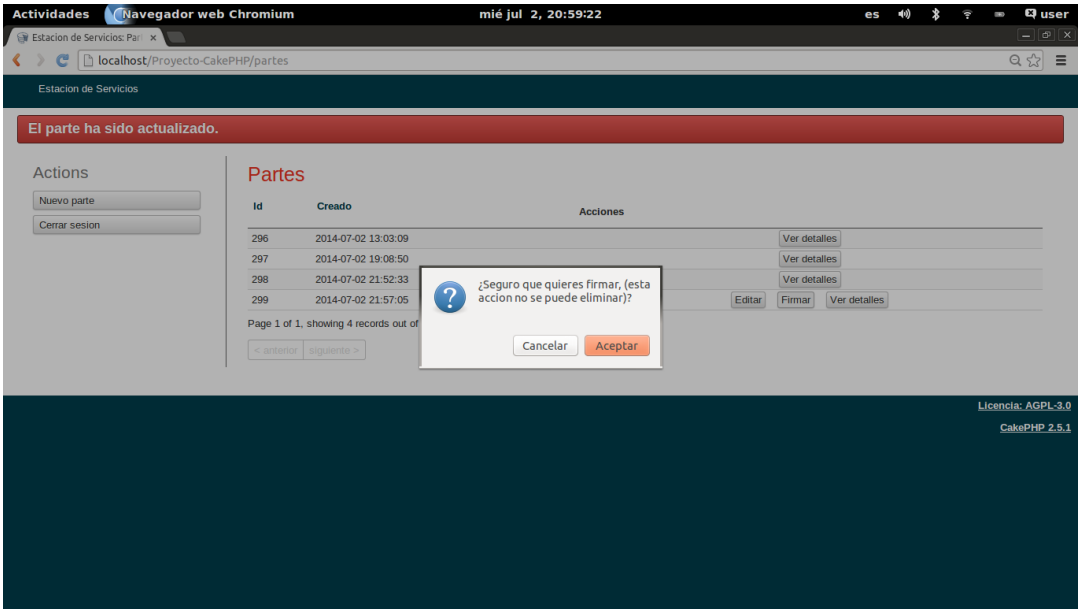

Figura 6.2.4.Confirmación de firma de parte.

De esta manera el usuario ha creado, editado y firmado un parte. Una vez firmado sólo se podrá acceder a ver detalles del parte, esta visualización sólo es informativa de los datos de creación del parte, no de su contenido. A continuación se muestra un ejemplo de dicha información (el gerente se asigna en el siguiente paso del workflow).

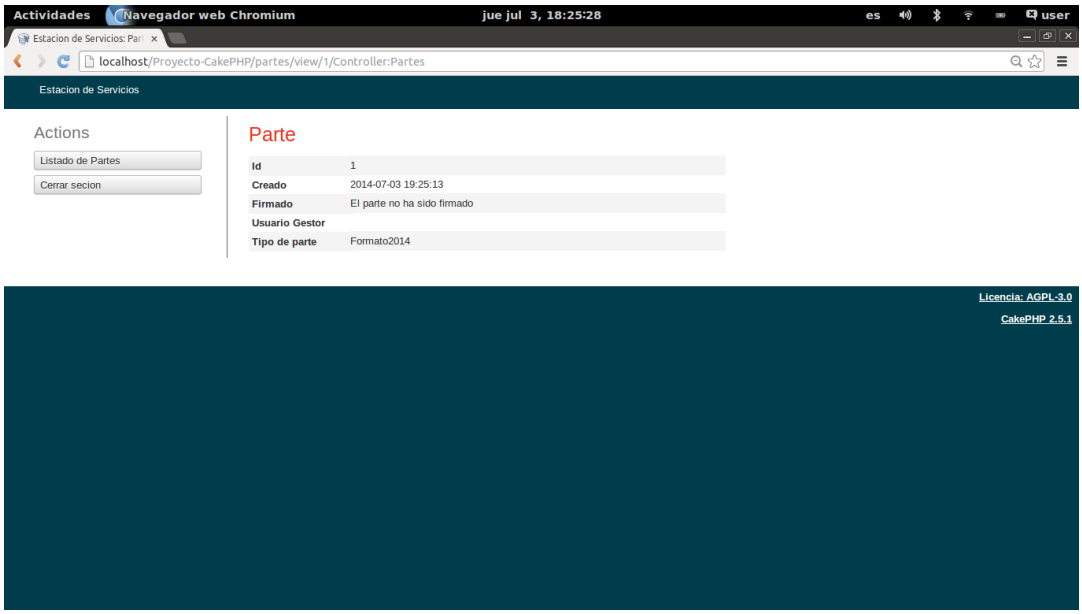

Figura 6.2.5.Visualización de detalles del parte

Algo importante de esta aplicación es tener un seguimiento del flujo de trabajo o workflow del parte de turno, por lo que se han definido dos pasos (aunque pueden ampliarse tal como indica la base de datos). Las acciones que se han mencionado anteriormente del vendedor son consideradas el primer paso dentro del workflow, por tanto los datos serán guardados con ese parámetro.

6.3 Gerente en la aplicación

El gerente tendrá un listado de todos los partes que están en la aplicación, dependiendo del estado se le mostrarán una u otras acciones.

En caso de que el parte no esté firmado o ya haya validado el parte se habilitará únicamente las opciones de ver detalles del parte (ver figura 6.2.5.), en el caso de que este validado ya nos aparecerá el gestor que lo ha realizado.

Si el parte ha sido firmado, el gerente podrá acceder a "crear una copia para validar". Esta opción es incluida ya que necesitamos tener una réplica de los datos, y no modificar los datos del vendedor. Lo que posteriormente nos permite editar. Esta acción se realiza de forma transparente al usuario emitiendo un aviso y habilitando las acciones de "visualizar y actualizar" y "validar".

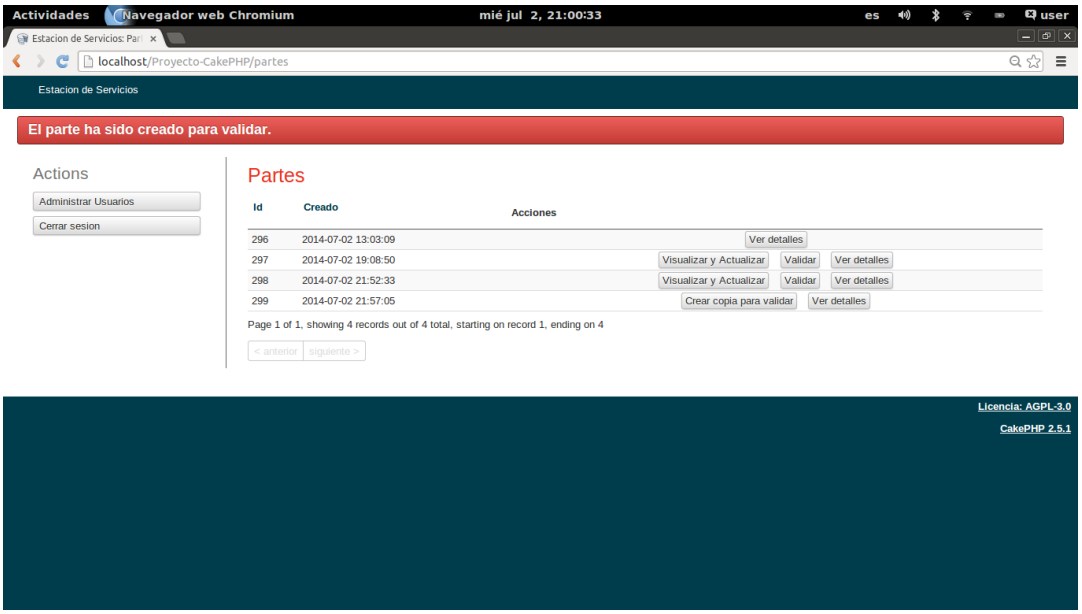

Figura 6.3.1.Visualización de lista de partes y mensaje al crear copia.

Si el gerente decide visualizar y actualizar se le mostrará una ventana nueva con la copia de los datos

introducidos por el vendedor. Esta actualización de datos usa una copia, para mantener ambas versiones y poder hacer un seguimiento del flujo de trabajo en todo momento.

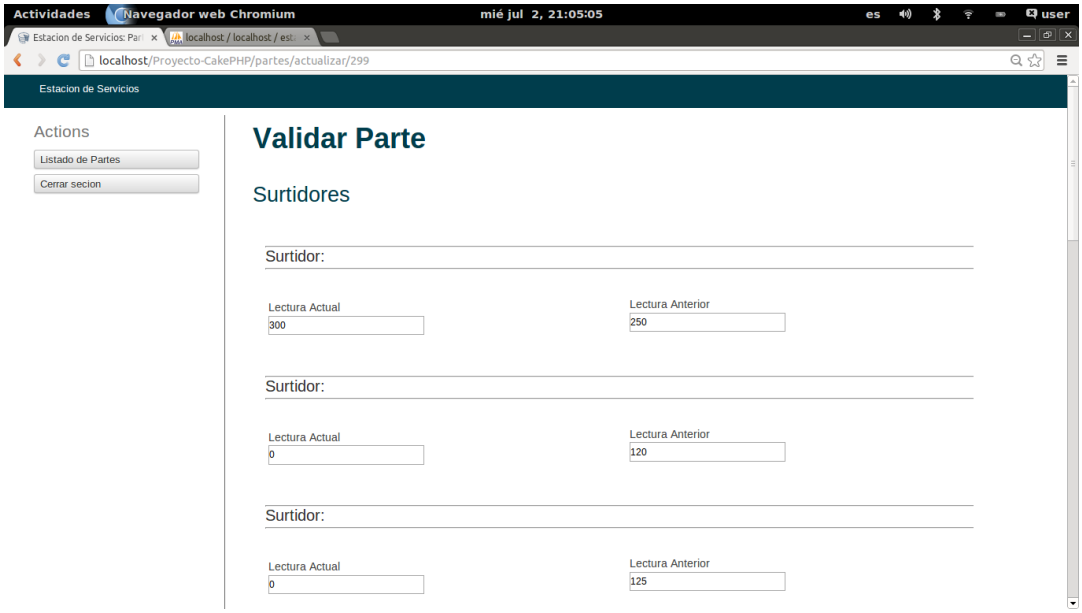

Figura 6.3.2.Visualización del parte para realizar cambios, muestra los datos del original.

Finalmente el gerente valida el parte, con aviso de que no puede modificar datos después de esta acción y terminamos con el segundo paso del workflow.

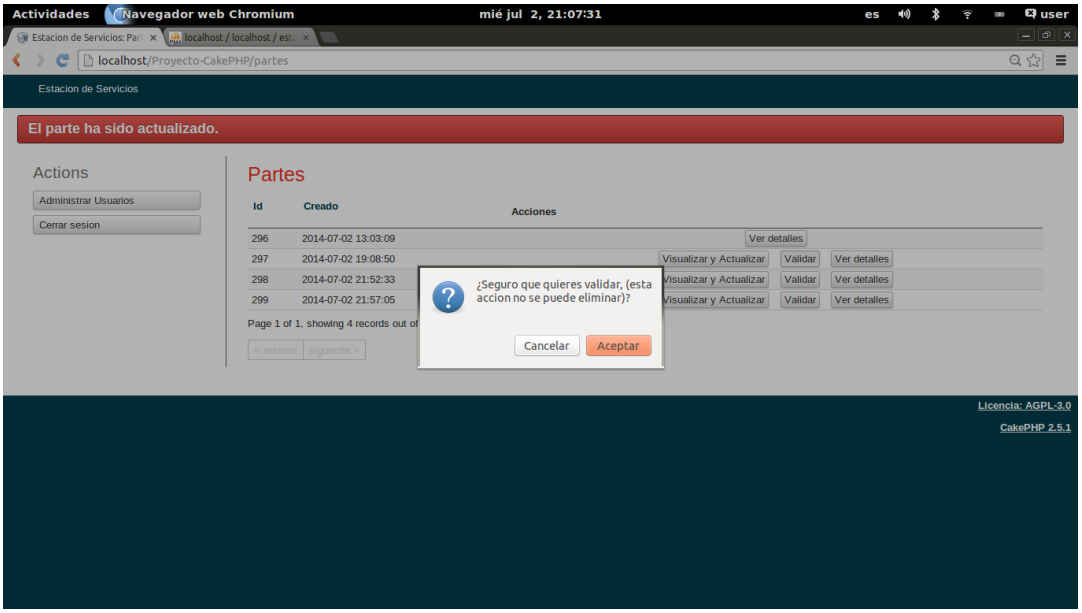

Figura 6.3.3.Confirmación de validación de partes

Por otro lado el gerente también debe poder administrar a los usuarios, por ello a la derecha se le muestra la acción "Administrar Usuarios", tal como indicamos con el administrador.

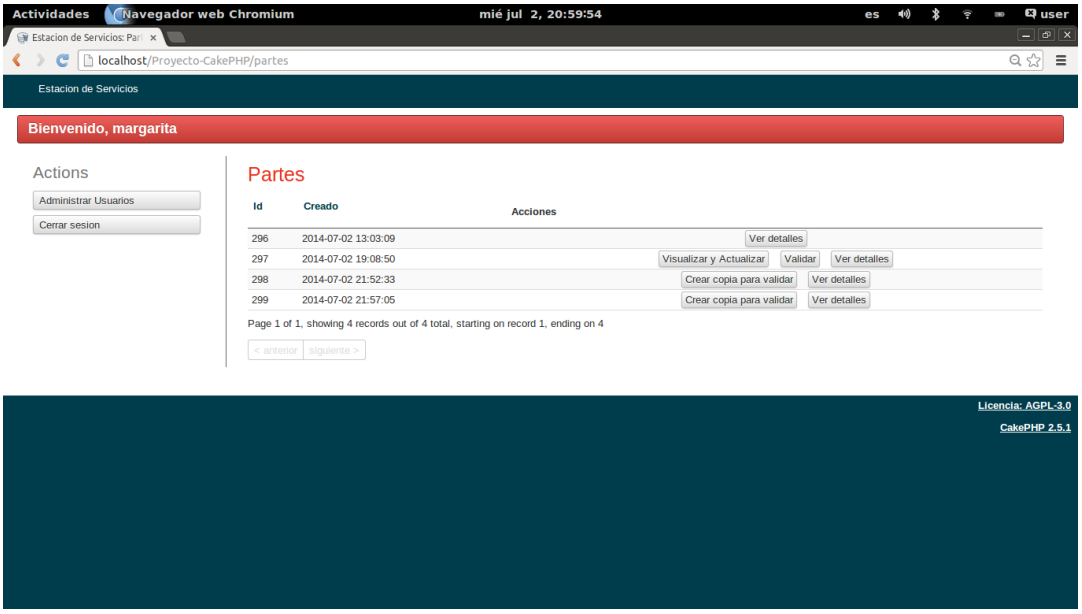

Figura 6.3.4.Imagen de listado de partes con bienvenida y a la izquierda acciones "Administrar Usuarios"

Futuras líneas de desarrollo

Como se mencionó en apartados anteriores, el proyecto puede seguir ampliándose con varias mejoras:

- Incluir un apartado de incidencias. Este tendría varias funcionalidades.
  - Se enviarían notificaciones en el caso de error en los partes de turno, de esta manera se evitarían réplicas de errores.
  - Enviaría mensajes de aviso o informativos a los usuarios de la aplicación. Dichos mensajes se podrían enviar desde la directiva de la organización o entre empleados, favoreciendo las comunicaciones tanto horizontales como verticales en la empresa.
  - Etc.
- Incluir un apartado de calendario que permitiría, varias acciones:
  - Agilizar y administrar el turno de vacaciones.
  - Administrar las vacaciones y horarios laborales.
  - Uso como recordatorio de tareas por realizar.
  - Reservas de compras de clientes.
  - Etc.

Todas estas medidas mejorarán la comunicación entre trabajadores y gestores, agilizando los trámites rutinarios y permitiendo mejoras en la organización.

Todo el código de esta aplicación está bajo la licencia GNU Affero General Public License (AGPL)[11] si desean consultar el código está disponible en la plataforma Github, en el repositorio del autor[12].

Capítulo 7. Conclusiones

En primer lugar, se ha conseguido una herramienta para poder integrar el parte de turno en el sistema de información.

Por una parte, los usuarios pueden realizar sus partes de turno de forma sencilla y segura. Esto se consigue con la limitación en acceso a la aplicación como en gestión de contenido al aplicarse las validaciones de datos.

Por otra parte los gerentes pueden gestionar estos partes de forma rápida y ágil. También se crean datos e información que ayudarán a la toma de decisiones.

Este proyecto puede aplicarse a una organización y modificarse para futuras necesidades dentro de la misma, únicamente cambiando el formato de vista al usuario e incluyendo los campos necesarios.

La flexibilidad de la aplicación permite la integración de la misma en cualquier organización que necesite un control de partes de turno por sus horarios rotativos continuos.

Capítulo 8. Conclusions

Firstly, it can be stated that it has been analyzed, designed and developed a tool to integrate the shift report management into the information system of service stations.

Additionally, users can do their shift reports simply and securely. Security is achieved with the limitation on access to the application.

Furthermore, managers can manage these reports quickly. Data and information that will help decision making are also managed.

The application developed in this project can be deployed into an organization and modified for future needs within it, just changing the format of the user view and including some additional fields.

The flexibility of the application allows its integration in any organization that needs shift reports management due to their continuous rotating shift schedules.

Capítulo 9. Presupuesto

Este proyecto se basa en la investigación, estudio y realización de un software. Para la realización del proyecto se ha optado por software libre y gratuito. También debe tenerse en cuenta el tiempo de desarrollo del mismo que ha sido de 16 semanas.

Licencias utilizadas:

Nombre	Licencia	Precio
XAMPP	GNU General Public License[13]	0€
CakePHP versión: 2.5	MIT[7]	0€
Navegador Mozilla Firefox	Mozilla Public License[14]	0€
Navegador Chromium	BSB Licence[15]	0€
Dia	GNU General Public Licence	0€
Geany	GNU General Public Licence	0€

Tabla 9.1. Presupuesto de licencias

El presupuesto para la realización del proyecto es:

Objeto	Coste	
Amortización de equipos*	41.60 €	
Coste de desarrollo**	4000 €	
Licencias	0€	
	Total	4041.60***

Tabla 9.2. Presupuesto

* Precio del equipo 500€.Duración del equipo 4 años->48 meses. Total: (4*500)/48
**10€/h.5h diarias, durante 16 semanas (5 días semanales).
***Impuestos no incluidos

Bibliografía

[1] Wikipedia lel. Sistemas de Información. 2014;2014(02/15):http://es.wikipedia.org/wiki/Sistemas_de_informaci%C3%B3n.

[2] Wikipedia lel. Big Data. 2014;2014(05/24):http://es.wikipedia.org/wiki/Big_data.

[3] Instituto Nacional dE. Encuesta sobre el uso de Tecnologías de la Información y las Comunicaciones (TIC) y del comercio electrónico en las empresas. Notas de prensa 2013.

[4] Wikipedia lel. Terminal Punto de venta (TPV). 2014;2014(02/15):http://es.wikipedia.org/wiki/Terminal_punto_de_venta.

[5] Gobierno de España. LOPD. http://www.boe.es 1999;298:43088-43099.

[6] Wikipedia lel. Modelo–vista–controlador. 2014;2014(05/07):http://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador.

[7] Opensource.org. Licencia MIT. 2014;2014(05/24):http://opensource.org/licenses/MIT.

[8] Technologies Ltd Z. Zend Framework. 2014;2014(06/28):http://framework.zend.com/.

[9] Potencier F. Symfony framework. 2014;2014(05/25):http://symfony.com/.

[10] Story M. Cakephp/debug_kit. 2013;2.2.x:https://github.com/cakephp/debug_kit.

[11] Free Software Foundation. GNU Affero General Public License. 2008;2014(06/06):http://www.gnu.org/licenses/agpl-3.0.html.

[12] Carballo Yanes J. Integración de sistemas de información en estaciones de servicio. 2014;1:https://github.com/JCYanes/Proyecto-CakePHP.

[13] Free Software Foundation. GNU General Public License.
2014;2014(06/2014):http://www.gnu.org/copyleft/gpl.html

[14] mozilla.org c. Mozilla Public License.
2012;2014(06/25):http://www.mozilla.org/MPL/.

[15] Wikipedia lel. BSD, Licencia.
2013;2014(07/1):http://es.wikipedia.org/wiki/BSD.

[16] Acevedo Solis J. Modelo Vista Controlador.
2014;2014(04/25):http://www.slideshare.net/jjsolis1/modelo-mvc-31699424.

[17] Conde J. Curso de CakePHP.
2011;Web:https://www.youtube.com/watch?v=YfIWFfEJGl0&list=PL5C6050FF4BBACAF5.

[18] Galindo F. Relacion muchos a muchos en cakephp.
2013:https://groups.google.com/forum/#!topic/cakephp-esp/HgxLTjOhqwc-
https://groups.google.com/forum/#!forum/cakephp-esp.

[19] Garcerant I. Ejemplo de caso de uso.
2008;2014(04/25):http://synergix.wordpress.com/.

[20] Lage ÓM. Ejecuntando un query a la antigua.
2008;2014(06/25):http://www.userlinux.net/1281_cakephp_ejecutando_un_query_a_la_antigua.html.

[21] O' zambrano A. Tablas relacionadas en cakephp y Recuperar Datos de ellas.
2011;2014(05/16):http://aaacini.blogspot.com.es/2011/06/tablas-relacionadas-en-cakephp-y.html.

[22] Olwest. CakePHP clarification on using set() and compact()together.2011:http://stackoverflow.com/questions/5477696/cakephp-clarification-on-using-set-and-compact-together-will-only-work-w-c.

[23] Otero D.Cakephp, consultas complejas a base de datos utilizando model.
2013:http://developando.com/blog/cakephp-consultas-complejas-base-datos-model 2014(06/02):1.

[24] Pablo. [CakePHP-esp] mostrar campos de diferentes tablas. 2013:http://grokbase.com/t/gg/cakephp-esp/133mbpss52/mostrar-campos-de-diferentes-tablas.

[25]Racó tècnic. CakePHP Debug Kit. 2014;2014(05/24):http://www.racotecnic.com/2010/11/cakephp-debug-kit/.

[26] Rafael Mercado R. Novato y Layaout. 2014:https://groups.google.com/forum/#!msg/cakephp-esp/LyT7POZHkxI/RK_p0MPWWuIJ.

[27] Reborn. Guadar datos en tablas relacionadas en CakePHP. 2014:http://www.forosdelweb.com/f68/guardar-datos-tablas-relacionadas-1089515/.

[28] Silva Orrego J. Problema con insert en Cakephp.2014:http://grokbase.com/t/gg/cakephp-esp/12aj1q315r/problema-con-insert.

Apendice A: Encuestas

A.1. Encuesta gasolinera "Cepsa, La Cañada".

ULL | Universidad de La Laguna

CEPSA LA CAÑADA
Ctra. Las Cañadas, Km. 5
38300 - LA OROTAVA
(Sta. Cruz de Tenerife)
Tfno.: 922 33 36 08
Fax: 922 33 04 69

LABRADOR LA CAÑADA S.L.
C.I.F.: B-38819520

Estación de servicio:

¿Qué software dispone esta estación de servicio? ¿De qué compañía?

"Tecnireg" = Tepe, contrata surtidor
Cepsa tarjeta fidelización. Software completo pero hay problemas de conexión.

¿Qué acciones de las siguientes desarrollan de manera informatizada?

1. BackOffice
   - 1.1. Control de empleados (nominas, asistencias, partes de baja...) ✓
   - 1.2. Control de inventario ✓
   - 1.3. Control de pedidos (solicitud, envío, factura...) ✓
   - 1.4. Contabilidad de la empresa (datos programas)
   - 1.5. Otras_____
2. Partes de Turno (diurnos/nocturnos) Programa (el mismo)
3. Aviso de incidencias (averías, control, etc.) Telefónica
4. Planificación de cambios de turno/Modificación de cambios de turno Manual. Suelen repetirse. Sin uso de programas
5. Control de surtidores (cuando existen turnos diurnos/nocturnos) Cierra en la noche.
6. Control de la tienda (existencias, ventas, etc.) Programa de control de existencias
7. Automatización del tren de lavado:
   - 7.1. lavado manual Compra ficha y pones el código.
   - 7.2. lavado máquina
8. Monolito de precios.
   Electronico separado.

Cepsa lo tiene todo, pero da problemas.
Por comodidad empresa local, llevar incidencia, trato personal.
9. Mejoras
Esta contento, sin añadidos

A.2. Encuesta gasolinera "BP, El Ramal"

ULL | Universidad de La Laguna

Estación de servicio: Estación el Ramal BP

¿Qué software dispone esta estación de servicio? ¿De qué compañía?

fullgas programas (independientes)
~~Dos~~ 2i".

¿Qué acciones de las siguientes desarrollan de manera informatizada?

1. BackOffice
   - 1.1. Control de empleados (nominas, asistencias, partes de baja...) X (gestoría)
   - 1.2. Control de inventario ✓
   - 1.3. Control de pedidos (solicitud, envío, factura...) X
   - 1.4. Contabilidad de la empresa ✓
   - 1.5. Otras (parecido a contaplus)
2. Partes de Turno (diurnos/nocturnos) ✓ programa
3. Aviso de incidencias (averías, control, etc.) teamviewer (varios)
4. Planificación de cambios de turno/Modificación de cambios de turno Manual
5. Control de surtidores (cuando existen turnos diurnos/nocturnos) No hay 24. Avería BP aparte. Pero si controla ventas.
6. Control de la tienda (existencias, ventas, etc.) ✓
7. Automatización del tren de lavado: No hay
   - 7.1. lavado manual
   - 7.2. lavado máquina
8. Monolito de precios. Digital BP, con ellos.

9. Mejora: Control de pago con pistola.

A.3. Encuesta gasolinera "Shell"

ULL | Universidad de La Laguna

Estación de servicio: Shell las arenas

¿Qué software dispone esta estación de servicio? ¿De qué compañía?

El propio de la estación

¿Qué acciones de las siguientes desarrollan de manera informatizada?

1. BackOffice
   - 1.1. Control de empleados (nominas, asistencias, partes de baja...) X
   - 1.2. Control de inventario ✓
   - 1.3. Control de pedidos (solicitud, envío, factura...) ✓
   - 1.4. Contabilidad de la empresa completa
   - 1.5. Otras_____
2. Partes de Turno (diurnos/nocturnos) Manual
3. Aviso de incidencias (averías, control, etc.) Wasap
4. Planificación de cambios de turno/Modificación de cambios de turno Manual
5. Control de surtidores (cuando existen turnos diurnos/nocturnos) 24 control de se
6. Control de la tienda (existencias, ventas, etc.) TPV
7. Automatización del tren de lavado:
   - 7.1. lavado manual X Solo manual.
   - 7.2. lavado máquina X
8. Monolito de precios.

Digital a través de un mando.

9. Mejoras
Esta contento, sin añadidos

A.4. Encuesta gasolinera "Disa, El Calvario"

ULL | Universidad de La Laguna

Estación de servicio: Disa el calvario

¿Qué software dispone esta estación de servicio? ¿De qué compañía?

Teide V.3.0. Tecnireg ⇒ sólo la gasolinera

¿Qué acciones de las siguientes desarrollan de manera informatizada?

1. BackOffice otro programa disponible
   - 1.1. Control de empleados (nominas, asistencias, partes de baja...) (exterior)
   - 1.2. Control de inventario (físicamente)
   - 1.3. Control de pedidos (solicitud, envío, factura...) propio
   - 1.4. Contabilidad de la empresa propio
   - 1.5. Otras_____
2. Partes de Turno (diurnos/nocturnos) el sistema y cuadrar a contabilidad
3. Aviso de incidencias (averías, control, etc.) telefónicamente, sobre papel avisan a la empresa por parte.
4. Planificación de cambios de turno/Modificación de cambios de turno Manual.
5. Control de surtidores (cuando existen turnos diurnos/nocturnos) 24 horas con el sistema. Durante el día se puede
6. Control de la tienda (existencias, ventas, etc.) La aplicación lo permite
7. Automatización del tren de lavado:
   - 7.1. lavado manual aparte del sistema el cliente
   - 7.2. lavado máquina paga y da tiquet (exterior).
8. Monolito de precios. Manda digital externo.

1. Mejoras:
Medios de pago integrado, no aparte.

Apendice B:Ejemplos de código

Se han puesto un anexo el código de la aplicación para explicar mejor los conceptos anteriormente explicados de relaciones definidas en las tablas y controlador sin vista asociada.

B.1 Ejemplo de relaciones en la tabla "tipocampostipoparte"

Esta es una de las tablas que contiene más relaciones, se definen de la siguiente manera en Cakephp, en el modelo tipocampostipoparte.

```
/**
 * Relaciones belongsTo
 *
 */
     public $belongsTo = array(
          'Tipoparte' => array(
               'className' => 'Tipoparte',
               'foreignKey' => 'tipoparte_id',
               'conditions' => '',
               'fields' => '',
               'order' => ''
          ),
          'Tipocampo' => array(
               'className' => 'Tipocampo',
               'foreignKey' => 'tipocampo_id',
               'conditions' => '',
               'fields' => '',
               'order' => ''
          ),
          'Tipofamilia' => array(
               'className' => 'Tipofamilia',
               'foreignKey' => 'tipofamilia_id',
               'conditions' => '',
```

```
                'fields' => '',
                'order' => ''
            ),
            'Categoria' => array(
                'classNameName' => 'Categoria',
                'foreignKey' => 'categoria_id',
                'conditions' => '',
                'fields' => '',
                'order' => ''
            )
        );

/**
 *  Relaciones hasMany
 *
 */
        public $hasMany = array(
            'Entero' => array(
                'classNameName' => 'Entero',
                'foreignKey' => 'tipocampos_tipoparte_id',
                'dependent' => false,
                'conditions' => '',
                'fields' => '',
                'order' => '',
                'limit' => '',
                'offset' => '',
                'exclusive' => '',
                'finderQuery' => '',
                'counterQuery' => ''
            ),
            'Reale' => array(
                'classNameName' => 'Reale',
                'foreignKey' => 'tipocampos_tipoparte_id',
                'dependent' => false,
                'conditions' => '',
                'fields' => '',
```

```
            'order' => '',
            'limit' => '',
            'offset' => '',
            'exclusive' => '',
            'finderQuery' => '',
            'counterQuery' => ''
        ),
        'Texto' => array(
            'className' => 'Texto',
            'foreignKey' => 'tipocampos_tipoparte_id',
            'dependent' => false,
            'conditions' => '',
            'fields' => '',
            'order' => '',
            'limit' => '',
            'offset' => '',
            'exclusive' => '',
            'finderQuery' => '',
            'counterQuery' => ''
        )
    );
```

B.2 Ejemplo de permisos

En esta función sólo los usuarios registrados pueden acceder a la aplicación y también definimos que sólo el creador de un parte puede editarlo y firmarlo.

```
/**
* isAuthorized
* Es la encargada de verificar si esta identificado puede
* acceder, y sólo si es el propietario puede editarlo y
* firmarlo
**/

public function isAuthorized($user) {
```

```
      // Todos los registrados pueden acceder al index y los
detalles de los partes
      if(in_array($this->action,array('index','view'))){
        return true;
      }
      // Sólo el vendedor puede editar y firmar el parte
if (in_array($this->action, array('editvendedor',
'firmar'))){
            $postId = (int) $this->request
>params['pass'][0];
            if ($this->Parte->isOwnedBy($postId,
$user['id'])) { //Indica si es el propietario.
               return true;
            }
            else{
               return false;
            }
      }
      return parent::isAuthorized($user);
}
```

B.3 Ejemplo de código de controlador sin vista asociado

Este es un ejemplo de código del controlador sin vistas asociada donde ser crea un nuevo parte y se comunica mediante un mensaje al usuario. Podemos ver que hace uso de una función auxiliar, se ha añadido para que se vea más claro el funcionamiento de "nuevo parte".

```
/**
* Inicializar partes
* Es la encargada de inicializar las tablas que almacenaran
los valores
```

```
* Recibe el tipo dentro de la tabla de tipocampos_tipoparte, la categoria a la que pertenece, el paso dentro del workflow,y el tipo de campo que pertenece para obtener su nombre, este servira como "name" del valor para visualizar mejor los datos en la BBDD
**/

     public function inicializar($idtipo, $cat_id, $newParteId, $workFlow,$tipocampo){

          $categoria = $this->Categoria->obtenervalor($cat_id);
          $campo = $this->Tipocampo->obtenername($tipocampo);
          $this->requestAction('/'.$categoria.'/add/'.$newParteId.'/'.$workFlow.'/'.$idtipo.'/'.$campo);//LLama a la acción del controlador correspondiente
     }

     /**
     * Creacion de un nuevo parte
     * Función encargada de inicializar el parte con los valores correspondientes definidos en el tipo de de formato a utilizar
     * Así como de inicializar las tablas para el almacenamiento de datos.
     **/

     public function nuevoparte(){
       $tipoparteid = $this->Valoresdefecto->obtenervalor(); //Obtiene el valor guardado por defecto del formulario a usar.
       $usuario=$this->Auth->User('id');
       $workflow = 1; //Primera fase inicializamos el workflow a 1
       $this->Parte->create();
```

```
        $this->Parte->save();//guardamos un parte en blanco
        $newParteId = $this->Parte->id;

        //Crear todo el esqueleto para almacenar datos
        $data=  array('id'  =>  $newParteId,'tipoparte_id'=>
$tipoparteid,'usuariovendedor'=>$usuario,'incidencia_id'=>'1'
);

        // Por cada elemento que contenga el formato de parte
creamos los campos
        $camposparte = $this->TipocamposTipoparte
>obtenercamposformato($tipoparteid);

        //Recorrer cada uno de estos campos y por cada uno
obtener la categoria y crear un nuevo
        foreach($camposparte as $dato){
          foreach($dato as $d){
           $this->inicializar($d['id'],
$d['categoria_id'],$newParteId,$workflow,$d['tipocampo_id']);
//recorremos por categoria e id y creamos
          }
        }

        if ($this->Parte->saveAll($data)) {
          $this->Session->setFlash(__('El  parte  ha  sido
creado.'));

        } else {
               $this->Session->setFlash(__('El  parte  no  ha
podido ser creado intentelo de nuevo.'));
        }
        return $this->redirect(array('action' => 'index'));
        $this->autoRender = false;
    }
```