Optinet: Software de Gestión web para centros ópticos.

José Ángel Parada Jiménez, [(1)]Iván Ruiz Rube

Calle Fray Gerónimo de la concepción 1 D, San Fernando, Cádiz

625038673 paradajimenez85@gmail.com

[(1)] Escuela Superior de ingeniería, C/ Chile 1, 11002 - Cádiz.

Extracto:

Se ha desarrollado una aplicación web, por petición del gerente de Salud Visión, para llevar la gestión del centro óptico de manera eficiente y eficaz. La aplicación usará como lenguaje de programación PHP ayudándonos con el framework symfony2. El sistema debe poder gestionar todas las tareas que se realizan diariamente en el centro óptico tales como ventas, reservas, apartados, devoluciones, citas e informes de una manera sencilla, cómoda y sin errores. Además la aplicación generará documentos para su posterior impresión.

Palabras clave: web, php, symfony2, óptica, gestión.

1. Introducción

La empresa Salud Visión es una empresa especialista en venta de gafas graduadas, gafas de sol, lentillas y demás artículos relacionados con un centro óptico. La empresa utilizaba para sus funciones diarias una aplicación de gestión que presentaba algunas carencias de funcionalidades, problemas de usabilidad y errores. Por lo tanto, se ha construido una aplicación para solucionar esos errores y que el cliente quedase satisfecho con el producto.

1.1. Objetivos

Se ha construido una aplicación muy sencilla de utilizar ya que los usuarios del sistema pueden tener conocimientos informáticos limitados. La aplicación ofrece la posibilidad de ser usada en varios puestos de trabajo simultáneamente. Además, la aplicación trabaja en cualquier navegador, es segura y dispone de unos tiempos de respuesta adecuados.

2. Planificación

Para la construcción del sistema hemos usado la metodología iterativa RUP por ser la metodología estándar más utilizada en el desarrollo de software orientado a objetos. Esta metodología divide el proceso en fases dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que hace un mayor o menor hincapié en las distintas actividades.

Realizamos una planificación de los tiempos para la distintas fases del proyecto que se detallan a continuación:

Tabla 1: Tiempo estimado - Tiempo real

Fase	Tiempo estimado	Tiempo real
Fase de iniciación	60 horas	75 horas
Fase de elaboración y construcción	400 horas	550 horas
Fase de documentación	30 horas	50 horas
Fase de transición	50 horas	70 horas

Donde se puede ver que no se cumplieron los tiempos previstos por problemas no esperados o dificultad añadida no prevista.

En la construcción de la aplicación llevamos a cabo una serie de iteraciones en las cuales se van añadiendo requerimientos dando como resultado un ejecutable. Las iteraciones efectuadas para la construcción del software son las siguientes:

- **Primera iteración: Edificación del software**

  Se construyó un prototipo de la aplicación de acuerdo con las necesidades del cliente. Estas necesidades se obtuvieron en las distintas reuniones iniciales.

- **Segunda iteración: Control de identificación**

  Se creó un sistema de control de usuarios para que cada tipo de usuario de la aplicación sólo pudiese realizar algunas acciones de acuerdo a sus funciones o responsabilidades.

- **Tercera iteración: Aumento de funcionalidades**

  Se aumentaron las funcionalidades del software conforme a las nuevas peticiones del cliente. Se construyó un sistema de citas e informes.

- **Cuarta iteración: Modificación de base de datos**

  Se modificó en la base de datos algunos atributos, tanto insertando como eliminando de las tablas. Se modificaron algunos tipos de relaciones.

- **Quinta iteración: Aumento de las funcionalidades**

  Se aumentaron las funcionalidades del sistema con la generación de gráficas, auditoría, conexiones de los usuarios al sistema y generación de documentos PDF.

- **Sexta iteración: Modificación de la interfaz**

  Se modificó la interfaz de usuario para que se visualizase correctamente en resoluciones de pantalla pequeñas. También se modificó para dotar al sistema de un aspecto más profesional.

En la figura 1 se muestra un diagrama de Gantt donde se puede ver el tiempo dedicado a cada una de las tareas del proyecto junto con las iteraciones efectuadas:

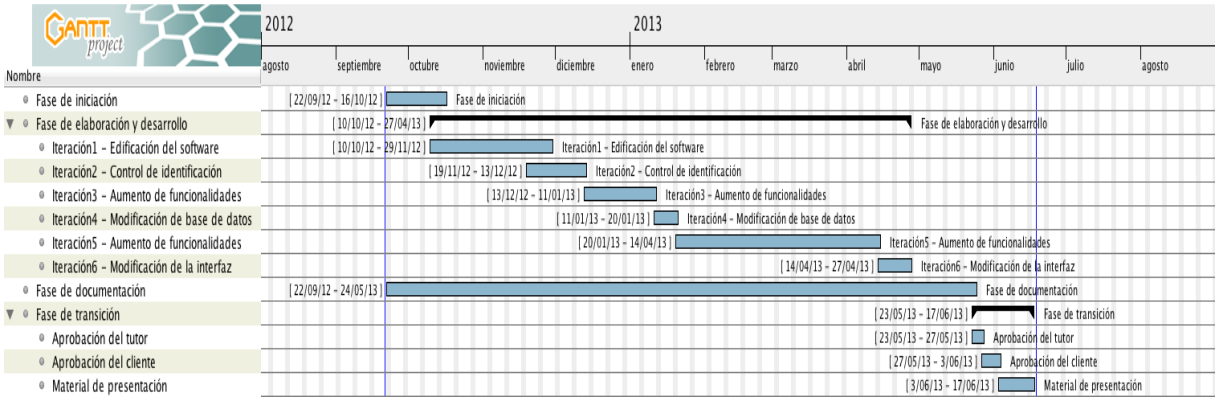

Figura 1: Diagrama de Gantt

3. Análisis de requisitos

En esta sección de análisis, nos centramos en lo que tiene que hacer el sistema olvidándonos de la tecnología. Para realizar el análisis de requisitos se concertaron entrevistas con el cliente, en las cuales recogimos un informe de necesidades que posteriormente formulamos como requisitos del sistema. En estas entrevistas, el cliente nos formuló cuales eran sus necesidades principales y, más tarde, en las siguientes entrevistas, llegamos a un acuerdo para que el sistema tuviese un aumento de funcionalidades y quedase más profesional. Durante estas entrevistas se estudiaron:

- ◇ Las responsabilidades de cada tipo de actor del sistema.
- ◇ Los requisitos funcionales que debería de tener la aplicación tales como gestión de usuarios, clientes, proveedores, productos, ventas, reservas, apartados, devoluciones, gráficas, citas e informes.
- ◇ Los requisitos no funcionales que la aplicación debería cumplir como portabilidad, rendimiento, seguridad, interfaz de usuario y auditoría.

En esta fase, hacemos uso de los diagramas de casos de uso para describir los pasos necesarios para cumplir un objetivo. Estudiamos los requisitos de información realizando su diagrama y, también, estudiamos el modelo de comportamiento del sistema. En la figura 2 se muestra una imagen en la que podemos apreciar el diagrama conceptual de datos del sistema.

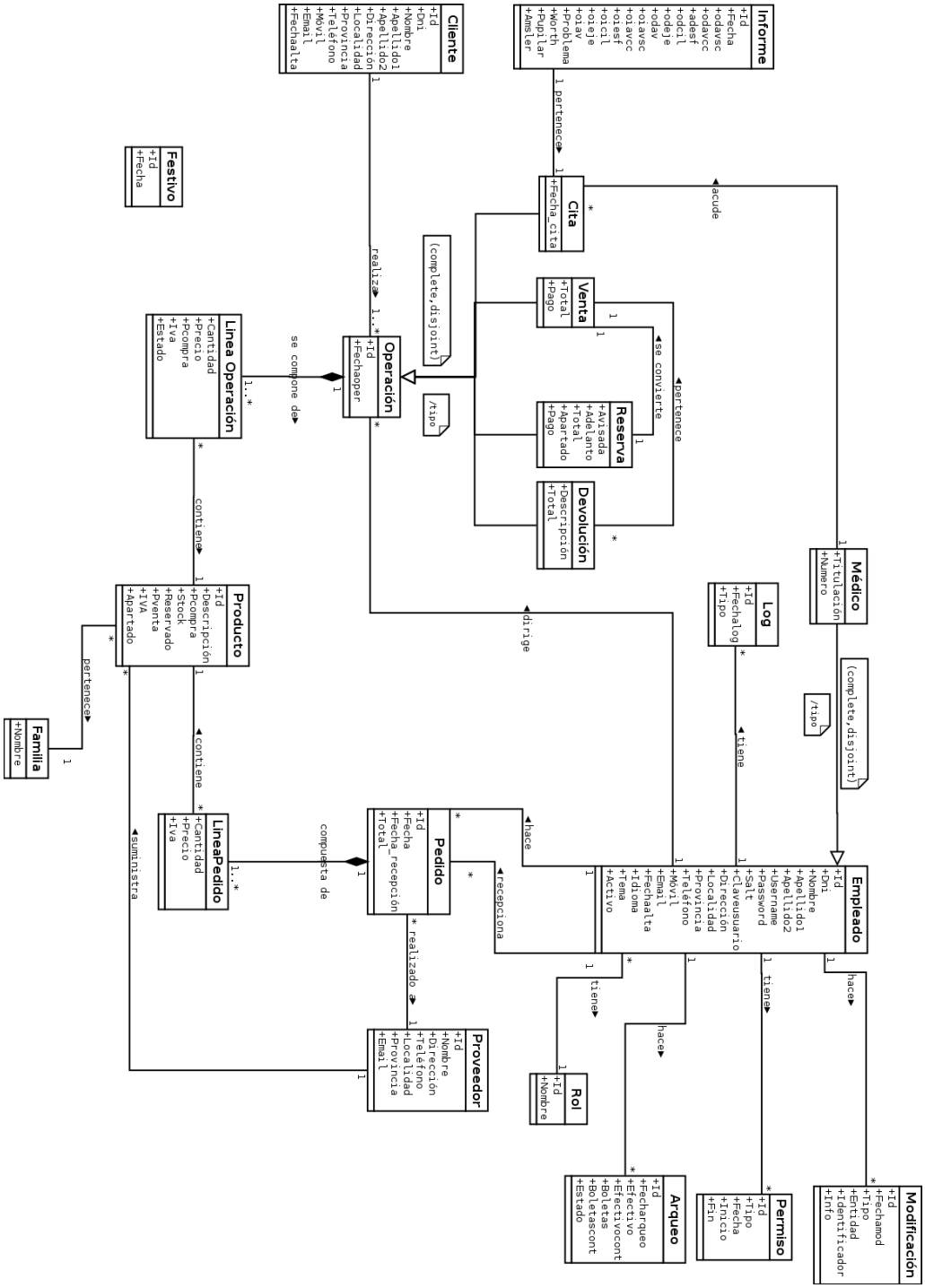

Figura 2: Modelo conceptual de datos

4. Diseño del sistema

En el diseño del sistema nos centramos en cómo hace el sistema para cumplir los objetivos teniendo en cuenta la tecnología usada. En el diseño del sistema nos centramos en el estudio de la arquitectura general del sistema, el diseño de la interfaz de usuario y el diseño físico de componentes.

Para la arquitectura de la aplicación se escogió la arquitectura MVC ya que tendremos el código más organizado, sencillez para distintas representaciones y podremos reutilizar código. Esta arquitectura MVC se separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.

La aplicación trabaja en un servidor web Apache con el lenguaje de programación PHP5.3+ instalado y MySQL para el banco de datos. Para que la interfaz de usuario fuese la adecuada se hicieron prototipos que se le enseñaron al cliente el cual los aceptó. También, a medida que se iba construyendo el sistema, se le fue enseñando para que la interfaz fuese plenamente de su agrado.

5. Implementación

Para realizar la aplicación se ha utilizado numerosas herramientas, bibliotecas y lenguajes de programación que se describen a continuación:

- **Lenguajes de programación utilizados:**
  Se ha elegido PHP como lenguaje base ya que es muy parecido al lenguaje C con una gran documentación. Para facilitarnos la tarea se ha utilizado el framework Symfony2 el cual nos aporta funciones, librerías, seguridad y velocidad entre otras cosas. Cuando un usuario solicita una página escrita en PHP, el motor de PHP ejecuta el código y envía al usuario una página en HTML a la cual daremos un aspecto visual utilizando hojas de estilo CSS. Para dotar al sistema de interactividad se ha hecho uso del lenguaje Javascript. Para la ayuda con este lenguaje se ha usado el

framework jQuery el cual nos facilita el uso del lenguaje JavaScript.

◇ **Bibliotecas utilizadas:**

Hacemos uso de la librería de componentes jQueryUI para el framework jQuery la cual nos permite un conjunto muy amplio de widgets, plugins y efectos para usar en nuestra aplicación web.

Para la creación de calendarios nos ayudamos de la librería Fullcalendar, que nos proporciona poder implementar de manera muy sencilla calendarios visuales muy atractivos y fáciles de personalizar. Además, ésta permite el uso de otra librería llamada qTip para generar tooltips a los elementos del calendario.

Para la generación de gráficos estadísticos usamos la librería HighCharts, ya que nos permite generar distintas estadísticas interactivas dependiendo de nuestra necesidad con muy buen aspecto.

Con respecto a las tablas que genera la aplicación, nos ayudamos de la librería Datatables la cual nos permite darle dinamismo a nuestras tablas además de conseguir la ordenación por campos, paginados, filtros, etc.

Por último, para la generación de documentos, usamos la librería PDFBundle que nos ayuda a usar la librería FPDF dentro de un proyecto Symfony2. Los documentos se abren en una ventana modal con la ayuda de la librería de jQuery FancyBox.

◇ **Herramientas utilizadas:**

Para poder llevar a cabo la aplicación debemos instalar LAMP que nos proporciona los elementos necesarios para un servidor web(Apache, MySQL, PHP).

Como editor usamos SublimeText2 ya que es muy ligero, ofrece de forma nativa infinidad de lenguajes de programación, además de poder instalar plugins como por ejemplo, LiveReload el cual nos permite ahorrarnos el trabajo de recargar la página web cuando modificamos código.

Para llevar un control de los archivos del proyecto usamos un software de control de versiones como GIT, ya que es ampliamente usado, gratuito y de código abierto.

Para la generación de los distintos documentos del proyecto hemos usado Día para la generación y edición de diagramas, Gimp para la edición de imágenes y LaTeX como lenguaje de programación para la creación de documentos. Como editor de texto en LaTeX hemos usado TextWorks.

Por último, para la generación de prototipos usamos PencilApp y GanttProject para realizar el diagrama de tiempo empleado en el proyecto.

6. Pruebas

Para comprobar que el sistema funcionase correctamente de acuerdo a los objetivos previamente establecidos se realizaron todo tipo de pruebas. A medida que se iba desarrollando el sistema se realizaron pruebas unitarias y de integración manuales, continuando cuando ya estaba realizada la aplicación con pruebas de sistema y de aceptación. En las pruebas de sistema comprobamos que el sistema cumple con todos los requisitos previamente establecidos tanto funcionales como no funcionales.

Las pruebas funcionales se realizaron manualmente comprobando que todos los escenarios, tanto principales como secundarios, funcionasen correctamente. En estas pruebas se iban comprobando los datos haciendo cálculos manuales.

En las pruebas no funcionales comprobamos requisitos como portabilidad, rendimiento, seguridad e interfaz de usuario. Para comprobar la seguridad se ha consultado el documento de los diez riesgos más importantes realizado por la fundación OWASP[1] y para la usabilidad se ha consultado la lista de los 25 puntos clave de usabilidad de UserEffect[2] . Para comprobar la portabilidad y el rendimiento se hicieron uso de herramientas online. En la figura 3 y 4 se puede ver el resultado de las pruebas de rendimiento mientras que en la figura 5 se puede ver el resultado de las pruebas de portabilidad.

[1] `https://www.owasp.org/images/2/2d/OWASP_Top_10_-_2010_FINAL_(spanish).pdf`. *OWASP*

[2] `http://www.usereffect.com/download/checklist.pdf`. *UserEffect*

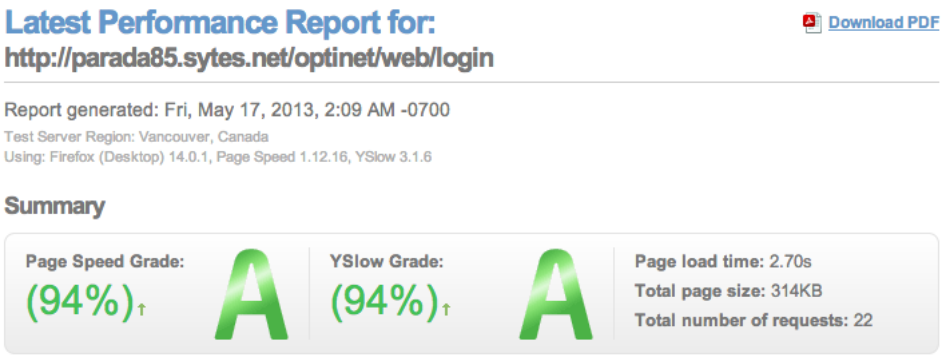

Figura 3: GTmetrix

Figura 4: Pigdom

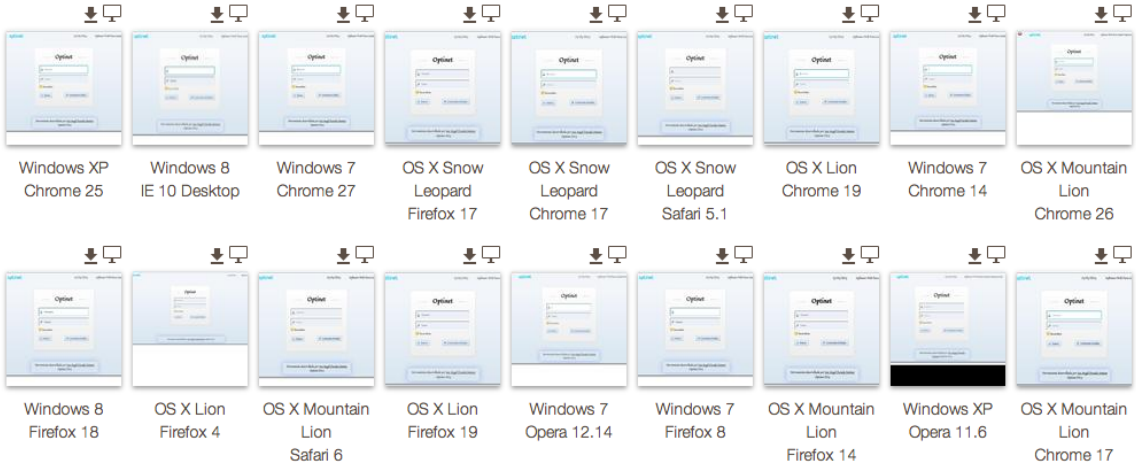

Figura 5: Browserstack

En las pruebas de aceptación diferenciamos a dos tipos de usuarios, por un lado usuarios con un alto nivel de conocimientos informáticos como amigos de la facultad con acceso al código y por otro lado personas con nivel bajo de conocimientos informáticos como cliente final. Los dos tipos de personas detectaron errores que fueron corregidos.

7. Conclusiones

La aplicación actualmente está implantada en el centro óptico y se han logrado los objetivos marcados al inicio del proyecto obteniendo así la satisfacción del cliente. A medida que se iba desarrollando el proyecto se repasaban conocimientos aprendidos durante mi formación universitaria además de ampliar y poner en práctica lenguajes de programación que serán muy valiosos de cara al futuro en el ámbito laboral. También, he aprendido a trabajar en solitario poniéndome fechas, leyendo manuales y participando en foros. A todo este trabajo en solitario se le unía la dificultad del idioma, por lo que he tenido que aprender obligatoriamente inglés técnico.
La aplicación se encuentra alojada en GitHub para que cualquier persona tenga acceso al código y pueda usarlo. De esta forma, aportamos nuestro granito de arena a la comunidad de software libre.

`http://github.com/parada85/optinet.git`

8. Bibliografía

◇ Página oficial de Symfony.
`http://www.symfony.com`

◇ Manual de PHP.
`http://es.php.net/manuel/es`

◇ Documentación de Symfony.
`http://gitnacho.github.io/symfony-docs-es`

◇ Api Symfony2.

`http://api.symfony.com/2.0/index.html`

◇ Conferencias de Symfony2.

`http://desymfony.com`

◇ Libro desarrollo ágil Symfony2.

`http://www.symfony.es/libro`

◇ Página oficial Doctrine.

`http://www.doctrine-project.org`

◇ Página oficial jQuery.

`http://www.jquery.com`

◇ Manual jQuery.

`http://librojquery.com`

◇ Página oficial de jQuery Datatables.

`http://www.datatables.net`

◇ Página oficial de jQuery Fullcalendar.

`http://arshaw.com/fullcalendar`

◇ Página oficial jQuery fancybox.

`http://fancybox.net`

◇ Página oficial Wijmo UI.

`http://wijmo.com`

◇ JsRoutingBundle.

`https://github.com/FriendsOfSymfony/FOSJsRoutingBundle`

◇ FPDF.

`http://www.fpdf.org`

◇ PdfBundle.
`https://github.com/psliwa/PdfBundle`

◇ DoctrineFixturesBundle.
`https://github.com/doctrine/DoctrineFixturesBundle`

◇ Foro google Symfony2.
`groups.google.es/group/symfony-es`

◇ Foro programación.
`http://stackoverflow.com`

◇ Wikipedia RUP.
`http://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational`

◇ El Lenguaje Unificado de Modelado. Guía de Usuario
G. Booch, J. Rumbaugh, I. Jacobson 1999

◇ UML y Patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado
C. Larman 2003

9. Agradecimientos

Para terminar el documento solo me queda agradecer a mis padres el apoyo recibido y a mi tutor del proyecto Iván Ruiz Rube por su dedicación e implicación en este proyecto.