Trabajo Fin de Grado

Desarrollo de un sistema de gestión de Recursos Humanos para pymes

Development of a Human Resources Management System for SMEs

Autora

Marina Ariño Armengol

Director

Fernando Cortés Franco

Ponente

Raquel Trillo Lado

Universidad de Zaragoza
Escuela de Ingeniería y Arquitectura

2018

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Marina Ariño Armengol,

con nº de DNI 73215107N en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Desarrollo de un sistema de gestión de Recursos Humanos para pymes.

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 30 de diciembre de 2017

Fdo: Marina Ariño Armengol

AGRADECIMIENTOS

En primer lugar, quisiera agradecer a mis compañeros y amigos de Endalia, en especial a Daniel Mur y a Nicolás Bailo por estar a mi lado durante estos meses y hacerme crecer profesionalmente, ayudándome en todo momento.
Gracias compañeros.

Mis agradecimientos también van para Fernando Cortés y Raquel Trillo por su excelente labor de dirección. No sólo por asesorarme y aconsejarme, sino también por la confianza depositada en todo momento. *Gracias Fernando y Raquel.*

Agradezco también el apoyo de mi familia y de mis amigos. En especial a mis padres Eusebio y Rosa, a mi hermana Marta, y a Adrián. *Gracias familia y amigos.*

Y por último, agradezco también a todos los docentes de la Universidad de Zaragoza de los que me siento afortunada de poder haber aprendido de ellos a lo largo de esos cuatro últimos años. *Gracias.*

¡Muchas gracias a todos!

endalia®
HR Consulting & Software

MEMORIA

DESARROLLO DE UN SISTEMA DE GESTIÓN DE RECURSOS HUMANOS PARA PYMES

VERSIÓN 1.4
PUBLICADO EL 29/12/2017

DESARROLLO DE UN SISTEMA DE GESTIÓN DE RECURSOS HUMANOS PARA PYMES

RESUMEN

Actualmente, se exige a los departamentos de recursos humanos que se adapten rápidamente a los cambios del mercado y la empresa, que aumenten la eficiencia de sus equipos de personas y que controlen y reduzcan los costes asociados a los recursos humanos. Las empresas afrontan estos desafíos consolidando una única plataforma integrada que les permite automatizar todos los procesos principales de recursos humanos, cumplir los requisitos legales y ofrecer unas prestaciones de servicio consistentes a una plantilla global.

Este documento presenta una descripción del desarrollo de un sistema para la gestión de recursos humanos para pymes (pequeña y mediana empresa). Este sistema servirá de ayuda a los responsables de recursos humanos para gestionar la información de las personas, los puestos de trabajo, la estructura organizativa y la gestión de vacaciones y ausencias de la organización. Está complementado con un portal web del empleado de Endalia, que permitirá la interacción por parte de los empleados de la organización.

Para implementar este sistema se han tenido en cuenta las posibles estructuras organizativas, que pueden tener en diversas empresas heterogéneas, con objeto de adecuar la gestión a las necesidades y requisitos de cada caso.

El sistema cuenta con tres módulos diferenciados:

- Organización: gestiona de forma dinámica las estructuras organizativas de la compañía. Describe sus puestos de trabajo, define su misión y automatiza sus procesos de revisión y valoración.
- Personas: gestiona la información personal y profesional de los empleados, desplegada a través del portal del empleado, y automatiza los procesos de actualización de datos.
- Vacaciones y ausencias: automatiza la gestión de vacaciones y ausencias, desde la solicitud inicial por parte del empleado hasta la aprobación final o rechazo, reduciendo la carga administrativa del proceso y facilitando su seguimiento.

El proyecto ha pasado por diferentes fases, siguiendo el ciclo de vida definido dentro de Endalia:

- Definición de alcance y captura de los requisitos técnicos y funcionales.
- Análisis del sistema a desarrollar.
- Diseño y prototipado del sistema.
- Implementación del sistema.
- Pruebas para comprobar el correcto funcionamiento del sistema.
- Implantación en cliente y puesta en producción para los usuarios.

El sistema desarrollado consta de una aplicación web que se conecta con una base de datos de forma segura, garantizando así la disponibilidad, integridad y confidencialidad de los datos; mientras que la lógica de negocio la provee una web API que permite compartir las clases de lógica al disponer el sistema de una arquitectura multicapa.

La plataforma y soporte tecnológico utilizado ha sido Microsoft .NET Framework 4.5 [W1] con el entorno de programación Microsoft Visual Studio [W2], el lenguaje de desarrollo C#, el *framework* Angular 4 las librerías de Kendo [W3] y NHibernate [W4], además de Git [W5] y Microsoft Team Fundation Server [W6] para el control de cambios en el software.

Por último, en cuanto a accesibilidad, se ha hecho especial hincapié en que el sistema sea compatible con diversos navegadores como, por ejemplo: Internet Explorer (en versiones superiores a la 8.0), Mozilla Firefox o Google Chrome, intentando mantener un comportamiento y apariencia similar en todos ellos.

HISTÓRICO DE REVISIONES

Fecha	Versión	Descripción	Autor
29/12/2017	1.0	Redacción inicial del documento	Marina Ariño Armengol
15/01/2018	1.1	Revisión del documento	Fernando Cortés Franco
17/01/2018	1.2	Correcciones sobre el documento inicial	Marina Ariño Armengol
23/01/2018	1.3	Revisión del documento	Raquel Trillo Lado
24/01/2018	1.4	Correcciones sobre el documento revisado	Marina Ariño Armengol

ÍNDICE

1. INTRODUCCIÓN

Este documento presenta la descripción del proyecto “Desarrollo de un sistema de gestión de Recursos Humanos para pymes”; en concreto, se han implementado tres módulos de este sistema: organización, personas y vacaciones/ausencias.

En esta primera sección de la memoria, se comienza describiendo el contexto en el que se enmarca el proyecto y, a continuación, se presentan los objetivos del mismo. En la segunda sección, se presenta la información general del proyecto y las tecnologías utilizadas para su desarrollo. Además, se ofrecen algunas métricas del proyecto como un cronograma, sus costes asociados y métricas relacionadas con el software. En la tercera sección, se describen las diferentes fases que se han realizado a lo largo de este trabajo. Comenzando por una fase de adaptación al entorno empresarial en el que se ha desarrollado y terminando por el desarrollo de este sistema de gestión de Recursos Humanos. La cuarta sección presenta los resultados del trabajo, ofreciendo imágenes de la aplicación final desarrollada. En la quinta sección, se presentan las conclusiones del trabajo, así como las ideas de mejora y algunas líneas de trabajo futuro. Además también se ofrece una valoración personal sobre este proyecto.

A lo largo del documento se presentan referencias bibliográficas a las principales fuentes de información documentales y tecnológicas empleadas a lo largo del desarrollo del proyecto y la elaboración de esta memoria de descripción del trabajo realizado.

1.1 Marco y justificación del proyecto

Endalia [W1] desde 2004 desarrolla un proceso de especialización funcional en el ámbito de los Recursos Humanos. Durante estos años se ha cubierto una necesidad importante en un conjunto de organizaciones de tamaño medio o grande, de una diversidad amplia de sectores, con unas determinadas características. Se están implantando en el mercado nuevos procesos y consolidando estos modos de hacer para la gestión del talento.

Se observa que existen organizaciones de diversos sectores con un tamaño de 30 a 100 empleados que también tiene la misma necesidad de aplicar políticas de dirección y desarrollo de personas y se encuentran ante la imposibilidad de poder acceder a estos modelos de gestión. Debido a esto nace la necesidad de crear y ofrecer a la pequeña o mediana empresa un sistema que permita gestionar en una única plataforma todos los principales procesos de Recursos Humanos, automatizándolos, permitiendo que se adapten rápidamente al cambio y que aumenten la eficiencia del personal, reduciendo los costes asociados a los recursos humanos gracias a esta herramienta intuitiva.

Se va a aprovechar la experiencia que se posee en el despliegue de estos modelos para construir un nuevo entorno de modelo aplicado a la pequeña empresa. En este nuevo modelo el producto deberá ser más sencillo, intuitivo, auto explicativo y cuya implantación no requiera la intervención de ningún técnico ni consultor. El usuario debe ser capaz por sí mismo con material de ayuda de poner en marcha los procesos de gestión de recursos humanos en su organización. El modelo de negocio cambia por completo. El marketing del producto, su comunicación y conceptualización exigen construcciones nuevas. En este proyecto nos centraremos en el diseño, implementación y despliegue del producto.

1.2 Objetivos

A continuación, se detallarán los objetivos que se pretenden alcanzar mediante la realización de este proyecto. Estos objetivos se dividen en dos secciones: objetivos de proyecto, referente al sistema a desarrollar, y desarrollo profesional, referente a la formación de la desarrolladora de este proyecto.

1.2.1 Objetivos de proyecto

El objetivo de este proyecto es el análisis, diseño y desarrollo de una aplicación web que permita la gestión de los empleados, la estructura organizativa de la empresa y las vacaciones y ausencias de los empleados. Este sistema permitirá al responsable de RRHH añadir, eliminar o editar puestos y empleados de la organización. También se podrán gestionar los periodos laborales y las solicitudes de vacaciones y ausencias que realicen.

El sistema deberá cumplir los estándares de calidad de la organización (Anexos I y II), adecuándose a su diseño. Además, se deberá crear un sistema parametrizable y configurable en función de las necesidades del cliente, ya que el mismo sistema debería poder ser utilizado por diversos clientes con diferentes necesidades. Por último, el sistema deberá asegurar la disponibilidad, integridad y confidencialidad de los datos.

1.2.2 Desarrollo profesional

Uno de los principales motivos a la hora de considerar el desarrollo de este proyecto ha sido el gran valor didáctico que suponía. Entre otras cosas, la desarrolladora de este proyecto tendrá la oportunidad de:

- Conocer, utilizar y adaptarse a las tecnologías utilizadas en la empresa, las cuales serán utilizadas en el desarrollo del proyecto.
- Comprender, analizar y justificar las metodologías, procesos, planificaciones y herramientas usadas en la empresa y cómo de adecuadas son para el trabajo a desarrollar.
- Participar en el proceso global de desarrollo de software, participando en todas y cada una de sus fases, con el objetivo de conocer de primera mano el ciclo de vida de un producto profesional.
- Formar parte de un equipo de trabajo en un entorno empresarial real, en el que obtener experiencia tanto del desarrollo de este proyecto como de las actividades derivadas de la actividad regular de la organización.
- Continuar con la formación de Ingeniera de Software, complementando la experiencia obtenida trabajando en una empresa con la formación recibida en la universidad.

1.3 Documentación

La documentación generada durante el proyecto, que se entrega en el momento del depósito, consta de dos partes: memoria y anexos. Se ha realizado el depósito electrónico en ZAGUAN, incluyendo ambos documentos en formato PDF.

1.3.1 Memoria

La primera parte de la documentación corresponde a este documento, la memoria del trabajo. El objetivo de este documento es el de presentar una visión global y resumida del trabajo, de forma que el lector obtenga una idea general de la totalidad del trabajo realizado sin necesidad de entrar en excesivo detalle. Esta memoria se apoya en el resto de la documentación, los anexos descritos a continuación, los cuales proporcionan al lector interesado información más detallada de los procesos llevados a cabo a lo largo del desarrollo del trabajo.

1.3.2 Anexos

La segunda parte de la documentación corresponde a los llamados anexos. Se compone de varios documentos diferentes:

- **Especificación de requisitos**: presenta los resultados obtenidos del estudio de las necesidades del cliente, identificando los diferentes requisitos que el sistema diseñado debe cumplir.
- **Análisis**: presenta un análisis de la arquitectura del sistema a desarrollar, identificando casos de uso y la estructura de paquetes, así como un estudio de los requisitos especiales del sistema.
- **Diseño**: describe el proceso de diseño del sistema de gestión de recursos humanos para pymes, presentando los requisitos, interfaces y clases obtenidos a partir de las especificaciones de requisitos y análisis previos.
- **Implementación**: presenta la fase de implementación del trabajo, en la que se desarrolla una versión funcional del sistema de gestión de Recursos Humanos para pymes.
- **Pruebas**: describe la fase de pruebas del sistema de gestión de recursos humanos para pymes desarrollado, en la que se asegura la calidad del software y el correcto funcionamiento de la implementación realizada en la fase anterior, mientras se garantiza que se cumplen los requisitos planteados al comienzo de la realización de este trabajo.
- **Plan de gestión de configuraciones**: define el plan de gestión de configuraciones utilizado. En él se presenta la configuración del software y se explica cómo se gestionan los cambios en él.
- **Estándar de documentación:** define el estándar de documentación utilizado. Contiene información sobre el formato y las plantillas utilizadas en la aplicación Microsoft Word.
- **Estándar de codificación:** define el estándar de codificación utilizado. Contiene información sobre las normas y convenciones de programación para el código fuente del trabajo, así como para las entidades relacionadas con la base de datos.

1.4 Acrónimos

API: *Application Programming Interface.*

DTO: *Data Transfer Object.*

HTML: *HyperText Markup Language*.

ORM: *Object-Relational Mapping.*

PDF: *Portable Document Format*.

SGBD*:* Sistema Gestor de Bases de Datos*.*

SPA: *Single-Page Application*.

WCAG: *Web Content Accessibility Guidelines*.

2. INFORMACIÓN GENERAL DEL PROYECTO

En esta sección se van a tratar temas relacionados con la información general asociada al proyecto software. Después de presentar el software creado, se realiza una descripción de las tecnologías utilizadas para llevarlo a cabo y se presentan unas métricas generales que permitirán obtener una idea del tamaño y el coste del mismo.

2.1 Descripción del software desarrollado

El software desarrollado consiste en un sistema de gestión de recursos humanos para pymes. Este proyecto aborda el desarrollo de la aplicación web de administración, la cual se integra con el portal web de Endalia, en el cual se han llevado a cabo varias mejoras como por ejemplo la centralización de la lógica de negocio de los módulos desarrollados.

Se han desarrollado los módulos de organización, personas y vacaciones/ausencias. En el módulo de organización se distinguen dos vistas: la estructura organizativa en la que se muestra una jerarquía de los puestos con estado activo y una vista de puestos históricos (Figura 1). Estos puestos poseen una información asociada (Figura 2) como puede ser el código, si es un departamento, el centro de coste, la función, la categoría interna, la fecha de creación o la misión de este. Los valores de estos desplegables son totalmente parametrizables en función de la empresa. Se permiten varias acciones en la estructura como por ejemplo la creación de un puesto en el mismo nivel que otro, o por debajo jerárquicamente y añadir empleados activos al puesto.

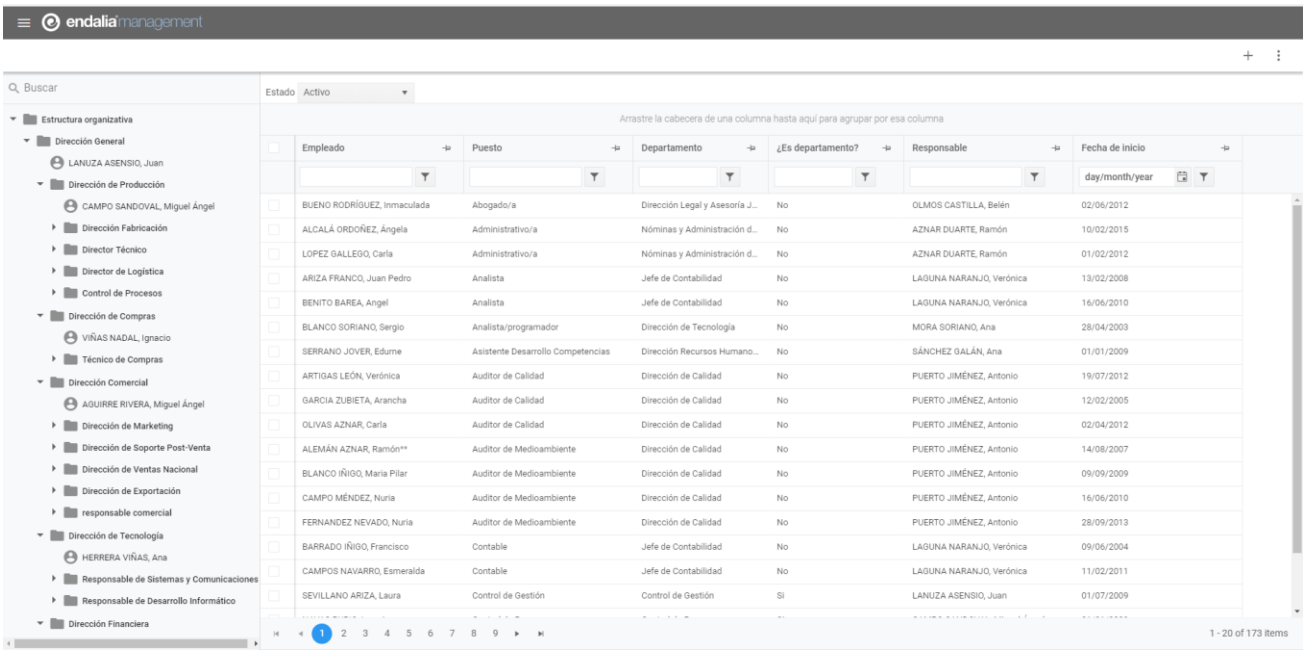


Figura 1. Estructura organizativa

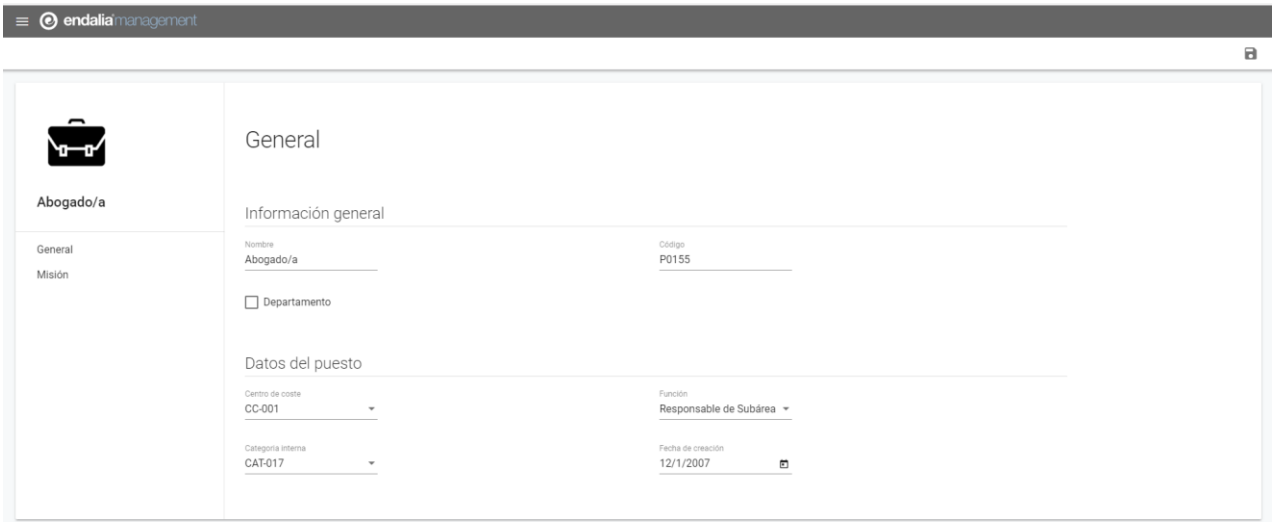


Figura 2. Información asociada al puesto

En el módulo de personas (Figura 3) también se distinguen dos vistas que se corresponden con el estado de los empleados (activos o históricos). En cada una de ellas se presenta la lista de empleados junto con la información más relevante de los mismos. Al acceder a cada empleado se presentará la información más detallada tanto personal como profesional (Figura 4).

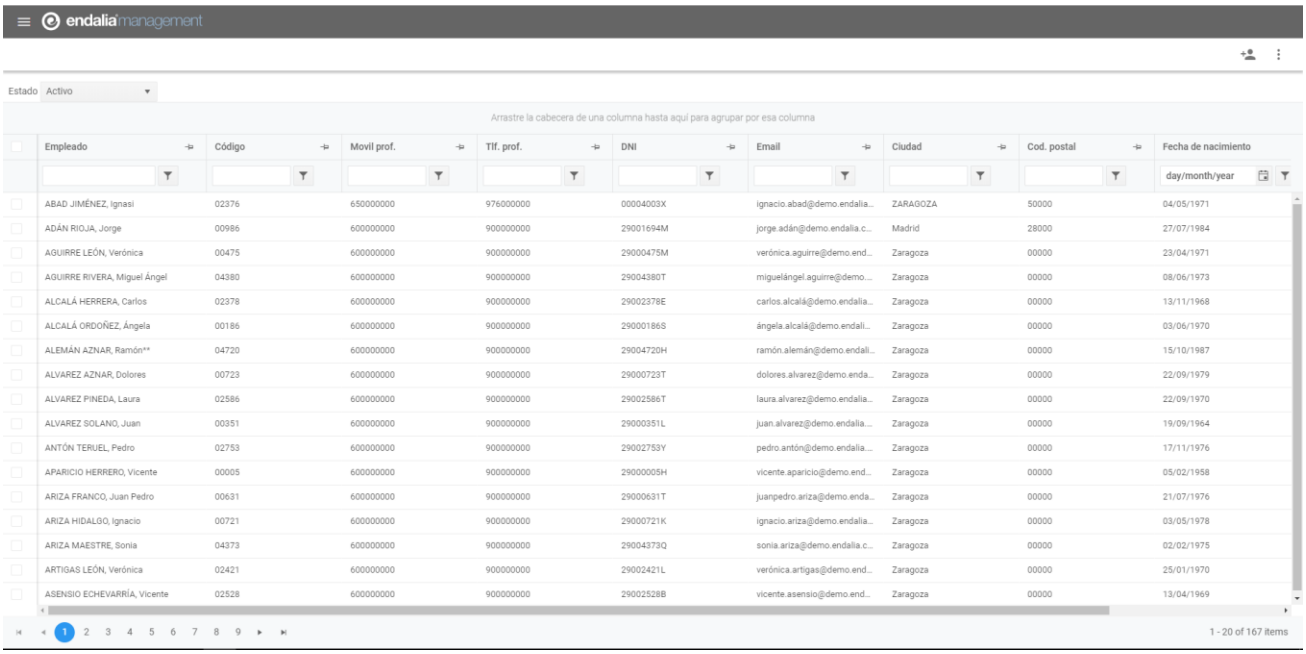


Figura 3. Listado de empleados

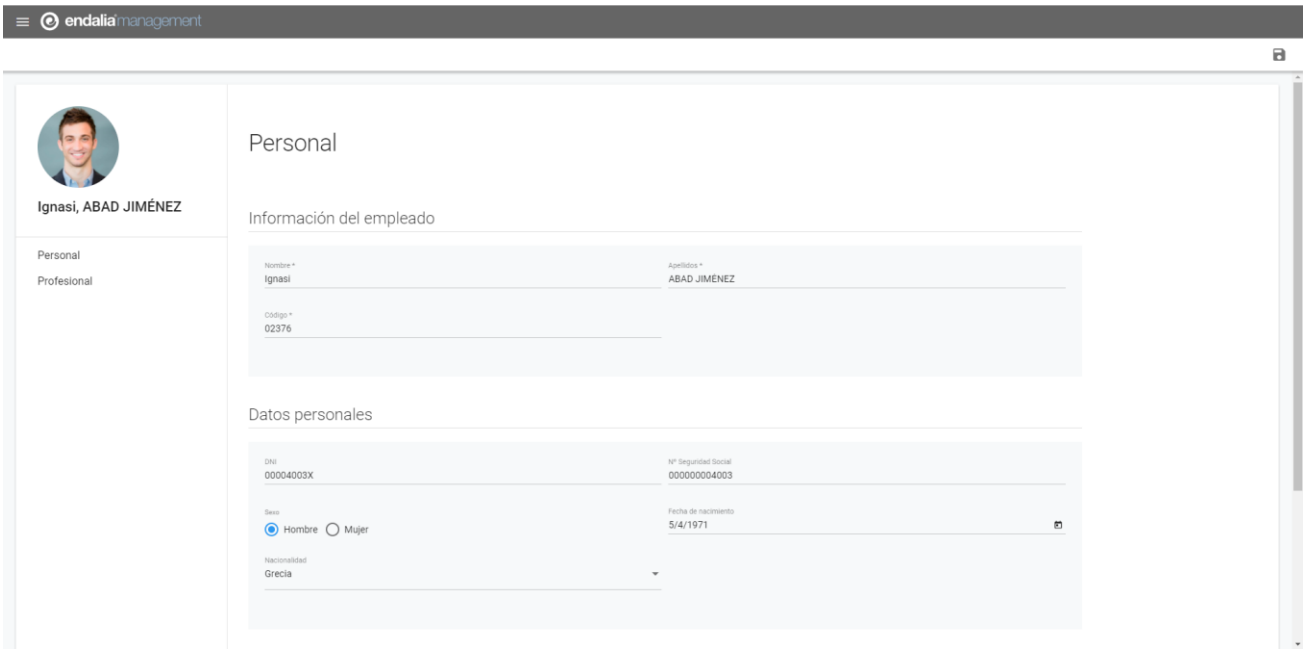


Figura 2. Información asociada al empleado

En el módulo de vacaciones se distinguen dos secciones: periodos y solicitudes. En primer lugar, se podrán administrar los distintos periodos de solicitud (Figura 5), identificando el rango de fechas para cada periodo. De este modo, cuando un empleado incluya una solicitud de ausencia, siempre lo hará vinculada a un periodo. Además, se permite determinar los períodos máximos de ausencia en cada concepto (por ejemplo, de vacaciones), de modo que podamos controlar las desviaciones respecto a los límites impuestos.

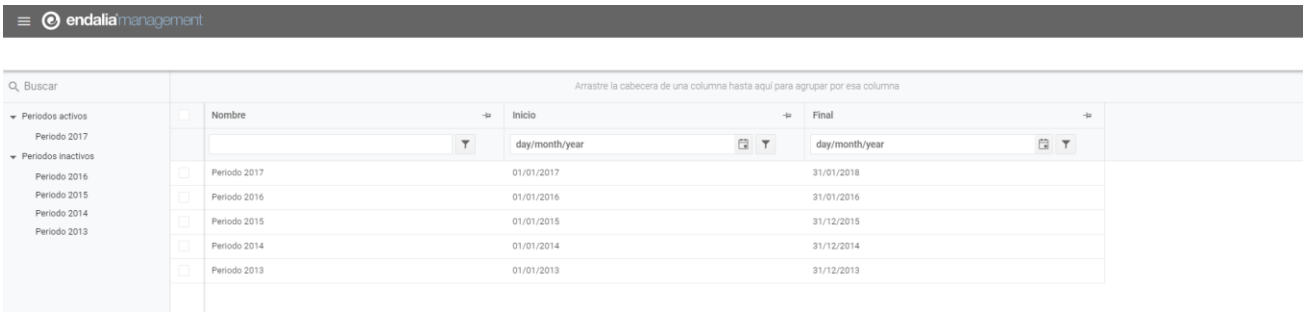


Figura 3. Periodos de solicitud

Por otro lado, se podrán visualizar todas las solicitudes realizadas (Figura 6) por los empleados de la organización. Cada usuario puede solicitar desde el portal web un nuevo periodo de vacación o ausencia. En el momento en que el empleado realiza la solicitud, se envía una notificación a su(s) responsable(s) directo(s), de modo que estos puede aprobar o rechazar la solicitud. En caso de aceptación, la solicitud pasa a Recursos Humanos, quién se encargará de dar la aceptación definitiva.

Desde esta misma ventana (Figura 6), Recursos Humanos puede también aprobar o rechazar las solicitudes que vengan de cada responsable, conociendo el número de días solicitados, cuántos días de ausencias le quedan, y en qué situación se encuentran todas las solicitudes. Por otro lado, podrá crear nuevas solicitudes tanto de vacaciones como de ausencias o modificar las ya existentes.

2.2 Tecnologías utilizadas

El trabajo que nos ocupa se lleva a cabo en el marco de la empresa Endalia. Este hecho condiciona algunas decisiones sobre las tecnologías a utilizar, ya que el software desarrollado debe ser compatible y fácilmente integrable. Cabe destacar, que en este proyecto se han introducido la tecnología Angular 5 para desarrollar el cliente y Git como herramienta de control de versiones. Estas dos tecnologías son novedosas y de reciente incorporación en la organización. Ello ha exigido un mayor nivel de análisis y de complejidad en el proceso de aprendizaje.

Las tecnologías utilizadas durante el desarrollo de este proyecto han sido las siguientes:

- Microsoft .NET Framework 4.5 como plataforma y soporte tecnológico.
- C# y HTML como lenguajes de programación.
- Angular 5 como *framework*.
- Microsoft Internet Information Services como servidor de aplicaciones.
- Microsoft Visual Studio como entorno de programación y de depuración de código.
- Microsoft Team Foundation Server como herramienta de control de versiones del proyecto sobre Visual Studio.
- Microsoft SQL Server como sistema gestor de bases de datos.
- Microsoft SQL Management Studio como herramienta de gestión de bases de datos SQL Server.
- T-SQL como lenguaje de acceso a datos basado en SQL.
- NHibernate. *Framework* de mapeo objeto-relacional que facilita la representación de entidades relacionales de base de datos en objetos accesibles desde la aplicación desarrollada. Simplifica la gestión de la persistencia de los objetos de datos.
- Log4Net como librería para la generación y el mantenimiento de los logs del sistema.
- Microsoft Windows 10 como sistema operativo.
- Microsoft Internet Explorer (versiones superiores a la 8.0), Mozilla Firefox y Google Chrome como navegadores con los que se ha desarrollado y testeado el sistema.
- Microsoft Office como herramienta ofimática para crear la documentación del proyecto.

2.3 Métricas del proyecto

Con el objetivo de dar una visión global del proyecto, se presentan a continuación diversas métricas correspondientes a la planificación del proyecto, el tiempo necesario para su desarrollo y el software creado.

2.3.1 Cronograma

En este apartado se detallará la planificación que se ha seguido a lo largo del proyecto en cuanto a fechas de entrega y sesiones de seguimiento.

Tras una fase previa de adaptación en la empresa, se comenzó a desarrollar el proyecto compaginándolo con otras funciones y tareas de aprendizaje. Se comenzó la fase de requisitos el 11 de julio de 2017, donde se estudiaron las necesidades que el sistema debía satisfacer.

Durante dos meses se realizaron diversas sesiones de seguimiento con los diferentes equipos de la organización en las que se estudió el análisis y diseño del sistema. Posteriormente, se comenzó con la implementación de la API y por último se desarrolló el *front-end* de la aplicación. Durante esta fase de implementación se refinó el análisis y diseño siguiendo una metodología basada en el ciclo de vida en espiral. Se realizó una sesión el día 13 de noviembre, en la que se mostraron las primeras pantallas del sistema junto con una funcionalidad básica.

Una vez realizada la implementación, paso el sistema a una fase de calidad en la que se comprobó el correcto funcionamiento a través de pruebas unitarias y de sistema.

El sistema se va a implantar durante el mes de febrero lo que permitirá la realimentación por parte de los usuarios y se continuará trabajando para realizar actualizaciones.

2.3.2 Coste (en horas)

Endalia posee un sistema de registro de esfuerzos que consiste en asignar las horas de trabajo realizadas a los diversos proyectos de la organización. De esta manera se ha podido obtener que el coste total del desarrollo de este proyecto ha sido de 360 horas, lo que corresponde a 45 días de trabajo a tiempo completo.

2.3.3 Métricas del software

En este apartado se muestran otras métricas relacionadas con el trabajo y con su código, las cuales pueden ser de utilidad para dar una idea de su magnitud.

Este proyecto software está compuesto por:

- 3 controladores ASP.NET WEB API compuestos de 53 endpoints.
- 5 lógicas ASP.NET.
- 15 clases de DTOs.
- 15 componentes Angular.
- 5 servicios Angular.

3. FASES DEL TRABAJO

En esta sección se detallan las diferentes etapas que se han llevado a cabo a lo largo de este trabajo. El objetivo es proporcionar una visión global del proyecto, por lo que no se explicarán los detalles y especificaciones de las distintas etapas. Esto último se hace en los distintos anexos asociados a esta memoria, a los que se hará referencia.

3.1 Fase de adaptación

Esta etapa, previa al desarrollo del proyecto, ha permitido la adaptación al entorno empresarial y al uso de las tecnologías usadas en la empresa, tanto internas como de terceros. Adaptarse a los procesos y mecánicas de trabajo en la organización, así como el aprendizaje de las tecnologías que después serían utilizadas en el desarrollo del sistema (.NET o Angular, por ejemplo), ha sido de vital importancia para el desarrollo del proyecto.

3.2 Fase de planificación

3.2.1 Planificación del proyecto

La etapa de planificación se realizó al comienzo del desarrollo del proyecto. En ella, se estimó el coste de cada una de las fases de trabajo y se realizó su respectivo diagrama de Gantt (Figura X), con el objetivo de exponer el tiempo de dedicación previsto para cada una de estas fases:

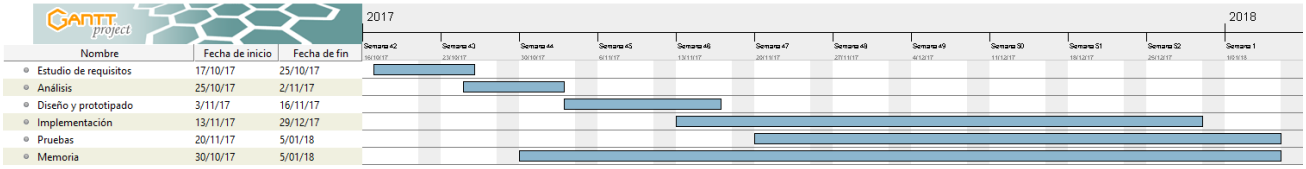


Figura 4. Cronograma de la planificación del proyecto

La primera etapa del proyecto, el estudio de requisitos, tuvo una duración una semana, así como la etapa de análisis. A continuación, se comenzará la etapa de diseño y prototipado, la cual se realizará durante las dos siguientes semanas. Además, se planificó una reunión el día 13 de noviembre para mostrar los primeros prototipos funcionales, por lo que se debía empezar con una primera implementación básica a lo largo de esa semana.

A partir de esa fecha se realizó toda la fase de implementación y pruebas del módulo, la que ha sido sin duda la fase más costosa del proyecto. Durante esta fase también se fue refinando el prototipo inicial.

3.3 Estudio de requisitos

El objetivo de esta fase es el de identificar los requisitos del cliente a partir de sus necesidades. A continuación, se crea un modelo del sistema a implementar teniendo en cuenta los requisitos obtenidos, que el sistema a desarrollar deberá cumplir.

En la especificación de requisitos se ha realizado un listado tanto de los requisitos funcionales como de los requisitos no funcionales. Además, también se plantean las funciones generales del sistema y se presentan los distintos tipos de usuarios que interactuarán con el sistema.

Dada la naturaleza de este documento, se presentan a continuación únicamente los requisitos funcionales más destacados:

#	Requisito funcional
RF-ORG-1	Creación de nuevo puesto
RF-ORG-2	Visualización de la información asociada a cada puesto
RF-ORG-3	Modificación de la información asociada a cada puesto
RF-ORG-4	Asignar empleado a puesto
RF-ORG-5	Visualización de un listado de empleados
RF-ORG-6	Creación de nuevo empleado
RF-ORG-7	Visualización de la información asociada a cada empleado
RF-ORG-8	Modificación de la información asociada a cada empleado
RF-ORG-9	Visualización de los periodos laborales
RF-ORG-10	Modificación de los periodos laborales
RF-ORG-11	Nueva solicitud de vacación o ausencia
RF-ORG-12	Visualización de solicitudes de vacación o ausencia
RF-ORG-13	Modificación de solicitudes de vacación o ausencia
RF-ORG-14	Eliminar solicitud de vacación o ausencia

El listado completo de requisitos funcionales y no funcionales, junto con sus respectivas descripciones, puede consultarse en la sección de anexos adjunta a esta memoria.

3.4 Análisis

La fase de análisis parte de los requisitos definidos mediante el estudio de requisitos para identificar los distintos casos de uso de la aplicación y los diferentes actores que interactúan con ella. Además, en esta fase, se presenta la estructura de paquetes de la aplicación y los requisitos especiales del sistema. Los resultados de este análisis han marcado las pautas a seguir en las siguientes fases del desarrollo.

En una primera instancia, se han identificado los diferentes actores del sistema. Estos serían: el Administrador/responsable RRHH que gestionaría la información de la organización mediante el sistema desarrollado en este proyecto y el resto de empleados de la organización que accederán al sistema mediante el portal del empleado. A continuación, se ha realizado el análisis de los casos de uso del sistema, identificando todos los procesos que tienen lugar en la aplicación. Cada caso de uso identificado se ha acompañado de un diagrama de caso de uso, un diagrama de secuencia y un diagrama de actividades, los cuales se presentan en el anexo correspondiente.

Se han dividido los casos de uso por módulo para obtener una visión más clara de las funcionalidades disponibles en cada uno de ellos, tal y como se puede observar en las siguientes figuras (Figura 8-9).

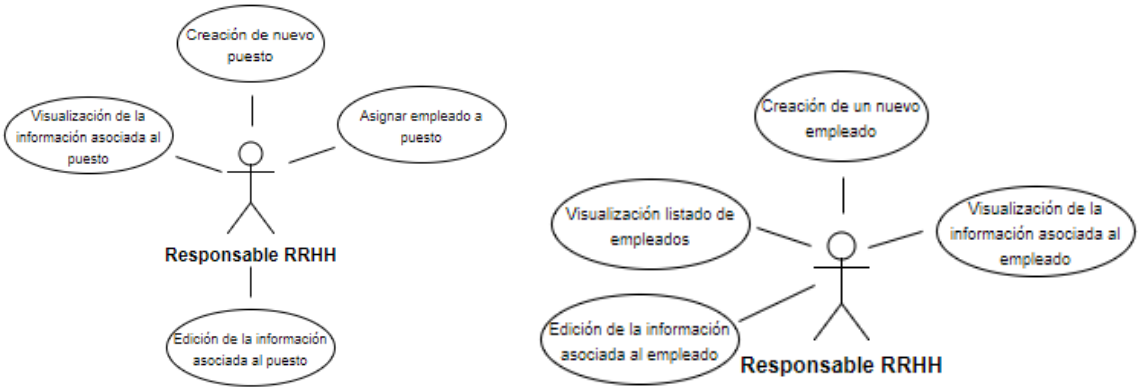


Figura 8. Casos de uso del módulo organización (izquierda) y personas (derecha)

Figura 9. Caso de uso del módulo vacaciones/ausencias

Tras analizar los casos de uso de la aplicación, se ha realizado un análisis de paquetes, en el que se identifican las entidades relevantes y sus relaciones. El objetivo de esta sección es la definición de paquetes de análisis que permitan organizar las diferentes entidades y funcionalidades del sistema en partes más manejables. En la Figura 10 se pueden observar los diferentes paquetes identificados:

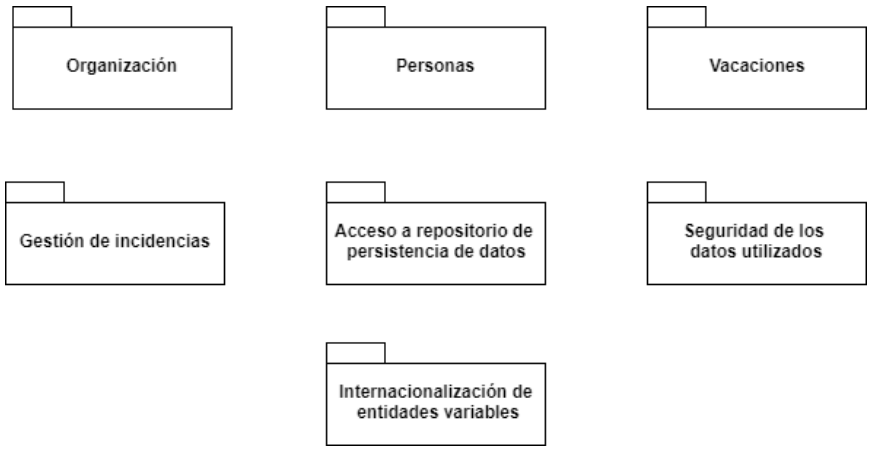


Figura 10. Paquetes de análisis

Por último, se han identificado ciertos requisitos especiales que, si bien no suelen hacer referencia a la funcionalidad final de cara al usuario, presentan restricciones y necesidades propias del sistema, ya sea por su naturaleza o por los estándares de calidad de la organización en la que se desarrolla. Los requisitos especiales identificados son los siguientes:

- Tolerancia a fallos: se deberá poder recuperarse de un error no controlado en el sistema y ofrecer información relativa a dicho error.
- Seguridad de los datos: al contener información confidencial no se deberá permitir el acceso a un usuario no registrado en la aplicación. Por otro lado, se deberá asegurar la integridad de los datos en las transacciones con la base de datos.
- Disponibilidad del sistema.

Se puede consultar en el anexo IV una explicación más detallada.

3.5 Diseño y prototipado

El objetivo de la fase de diseño y prototipado es el de obtener un punto de partida para las actividades de implementación, gracias a los resultados obtenidos en la fase de análisis. En esta fase se ha realizado, inicialmente, una descripción de la arquitectura del sistema tanto a nivel físico como a nivel de organización del mismo. Se detallan las clases identificadas así como el diseño de la base de datos. Por último, se presentan los diferentes prototipos de la interfaz del sistema, con la que interactuará el usuario. Este documento también ofrece un listado de especificaciones tanto tecnológicas como de diseño, en el que se presentan y describen las tecnologías a utilizar durante el desarrollo del mismo.

Uno de los principales requisitos al diseñar este proyecto debe ser la facilidad y comodidad con la que pueden los usuarios interactuar con la aplicación. Para ello se ha realizado un diseño centrado en el usuario, diseñado para y por el usuario. Se han realizado varias reuniones con futuros usuarios de este producto para obtener su retroalimentación (*feedback)* y recomendaciones con el objetivo de diseñar un sitio web usable. Se ha evaluado la usabilidad con el validador del World Wide Web Consortium [W13], obteniendo tiempos de respuesta en media de 4 segundos, una navegación simple y constante siguiendo la misma distribución en todas las páginas y compatible con la mayoría de navegadores.

A continuación se presentan algunos de los resultados obtenidos en esta fase. El documento íntegro se puede consultar en la sección de anexos adjunta a esta memoria.

3.5.1 Especificaciones tecnológicas

Al ser un trabajo realizado en la organización Endalia, varias de las tecnologías utilizadas en el proyecto han sido condicionadas, aunque se ha introducido el lenguaje Angular. La introducción de Angular se considera como una innovación en la organización ya que se trata de una tecnología novedosa a nivel de mercado y que, previamente, no había sido utilizada en la organización. Se ha utilizado la plataforma .NET de Microsoft para el desarrollo del *back-end*, en concreto ASP NET Web API con autentificación OAuth2 y Angular 5 para el *front-end*. Se ha implementado el patrón de diseño de inyección de dependencias en el que se suministran objetos a una clase en lugar de ser la propia clase la que cree el objeto.

Se realizó una comparativa de distintas bibliotecas de componentes de interfaz para facilitar la creación de la misma, se recopilaron los componentes y características que debían tener estas y la elección fue Kendo-UI de Telerik [W12]. No obstante, también nos hemos apoyado en Material Desing [W11] para determinados componentes. Para la gestión de los datos se ha utilizado el SGBD Microsoft SQL Server, así como NHibernate para el mapeo objeto-relacional (ORM) de las entidades en base de datos. Para obtener más detalles sobre las tecnologías utilizadas, consultar el documento de diseño en los anexos adjuntos a esta memoria.

3.5.2 Arquitectura física del sistema

La arquitectura física del sistema está compuesta por las siguientes partes:

- El sistema de gestión de recursos humanos que es accesible desde cualquier navegador introduciendo la URL correspondiente.
- La base de datos, la información relativa a la estructura organizativa, los empleados y las solicitudes de vacaciones que se gestionan en el sistema. Está implementada desde el SGBD SQL Server.

3.5.3 Arquitectura lógica del sistema

La decisión de adoptar una arquitectura en multicapa se debe al desacoplamiento que ofrece. Además de independizar cada uno de los componentes permite organizar el sistema y sustituir la implementación de una capa por otra de forma sencilla. El sistema está basado en la siguiente arquitectura multicapa:

- Capa de presentación: formada por los componentes que se ejecutan en el cliente. En nuestro caso es el navegador web desde el que se accede a la plataforma y el código TypeScript que se ejecuta en las páginas de la aplicación. En este sistema se corresponde con los componentes angular y su código subyacente. En esta capa se ha concentrado la mayor parte del trabajo desarrollado.
- Capa de lógica de negocio. Contiene los objetos (DTOs) que representan los datos almacenados en el repositorio de datos, así como la lógica necesaria para procesarlos.
- Capa de acceso a datos: Contiene objetos que automatizan el acceso a datos, realizada a través del *Framework* NHibernate.
- Capa de datos. Contiene los sistemas de información de la aplicación, habitualmente una base de datos. En nuestro sistema se corresponde con los archivos de recursos de la aplicación y la base de datos, que contiene la información sobre los empleados, la organización y sus solicitudes de vacaciones y ausencias. Así cómo funciones y procedimientos para acceder a ellos.

3.5.4 Estructura en subsistemas

Los subsistemas son un medio para organizar el modelo en partes más pequeñas y manejables que tienen como objetivo mejorar la escalabilidad del sistema. A continuación se presenta un listado con los diferentes subsistemas identificados para el sistema de gestión de recursos humanos, los cuales se detallan en profundidad en el documento de diseño adjunto en la sección de anexos:

- Subsistema de estructura organizativa.
- Subsistema de personas.
- Subsistema de periodos de solicitud.
- Subsistema de solicitudes.
- Subsistema de gestión de incidencias.
- Subsistema de acceso a base de datos.
- Subsistema de internacionalización de entidades visibles.
- Subsistema de seguridad de datos utilizados.

3.5.5 Clases del sistema

Las diferentes clases que se han desarrollado para el sistema de gestión de recursos humanos para pymes se dividen en cuatro grupos diferenciados:

- Clases de interfaz: son las encargadas de crear la GUI y gestionar las interacciones del usuario con el sistema.
- Clases de acceso a datos: son las encargadas de gestionar la persistencia de los datos del sistema y la interacción con la base de datos.
- Clases de mapeo objeto-relacional de datos: son las encargadas de crear una representación de los datos persistentes en las bases de datos en forma de objetos utilizables desde el código de la aplicación.
- Clases de objeto de transferencia de datos que se encargan de transportar datos entre procesos.
- Clases auxiliares.

Los detalles de estas clases se encuentran en el documento de diseño adjunto a esta memoria.

3.5.6 Diseño de la base de datos

Debido al tamaño de la base de datos, se ha separado en subconjuntos de menor tamaño, ya que no sería posible representarlo de otro modo en este formato de documento. El primer subconjunto se corresponde a los módulos de organización y personas, y el segundo con el módulo de vacaciones y ausencias.

A continuación se muestran los dos esquemas (Figuras 11 y 12):

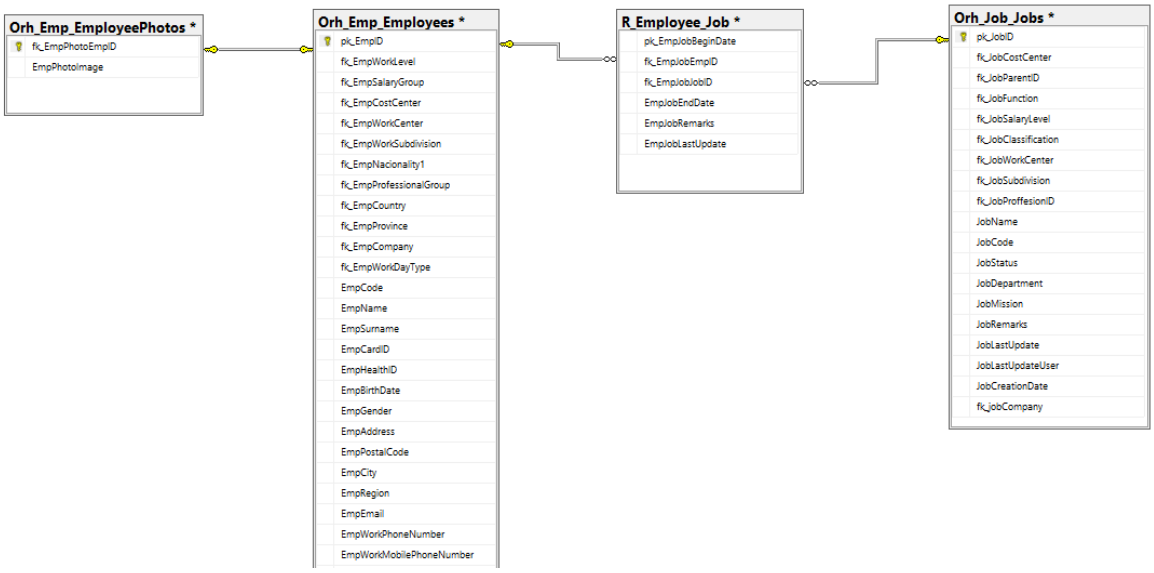


Figura 11. Esquema general de los datos implicados en los módulos de organización y personas

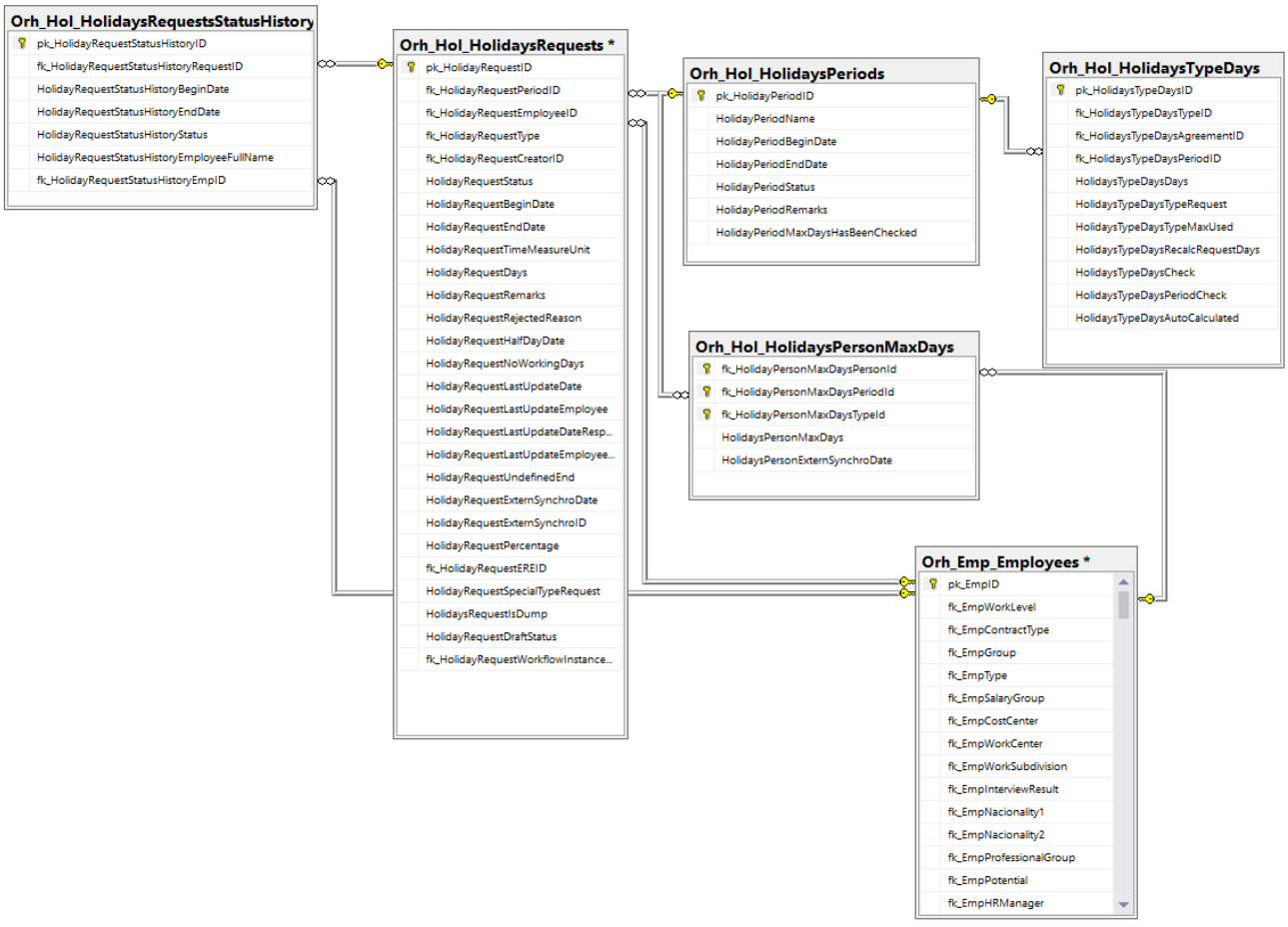


Figura 12. Esquema general de los datos implicados en el módulo de vacaciones

3.6 Implementación y pruebas

En la fase de implementación se parte del trabajo realizado en las fases de análisis y diseño para desarrollar una versión funcional del sistema de gestión de recursos humanos para pymes.

El documento correspondiente a dicha fase se encuentra en la sección de anexos adjunta a esta memoria. En él se detallan las tecnologías, herramientas y lenguajes empleados durante la implementación del sistema, mientras se describen los procesos seguidos para la implementación del sistema, la gestión de incidencias, el acceso a datos y la interfaz de usuario.

Dentro del proceso de implementación propiamente dicho podemos identificar dos fases:

- Implementación de la base de datos: creación de las tablas y las relaciones necesarias para el nuevo sistema en la base de datos, el conjunto de clases de acceso a datos correspondientes y sus respectivos mapeos a objetos.
- Implementación de los subsistemas identificados en el diseño, adecuándose a la interfaz definida previamente.

Cabe destacar que en este trabajo se ha realizado únicamente la segunda de estas fases, ya que el modelo relacional de tablas ya había sido creado previamente por la entidad para un proyecto similar. El trabajo realizado en la primera de las fases ha consistido únicamente en la validación de dicho modelo relacional para asegurar su validez en el sistema de gestión de recursos humanos para pymes.

Una vez finalizada la implementación del sistema se pasa a la realización de las pruebas del sistema. El objetivo de esta fase es el de detectar las partes del proceso que deben ser modificadas debido a que no alcanzan los estándares de calidad necesarios. Se han realizado ambas pruebas unitarias y pruebas de sistema. Esta fase está detallada en el anexo “Pruebas”, adjunto a esta memoria, que contiene una descripción de las pruebas realizadas así como los criterios de aceptación y el resultado de cada una.

3.7 Integración con el portal de Endalia e implantación en el entorno del cliente

La aplicación desarrollada permitiría al responsable de recursos humanos de la organización administrar todos los aspectos que se han presentado en este documento. Este sistema se complementa con el portal web donde podrán acceder todos los empleados y visualizar la estructura organizativa y un listado con el directorio de empleados junto con la información asociada a estos. Por otro lado, será desde este portal donde se realizarán las solicitudes de vacaciones y ausencias. También se ha integrado en esta aplicación la visualización del organigrama que está disponible en el portal web.

Por otro lado, durante el mes de febrero se va a realizar la implantación en el cliente y se capturará su retroalimentación (*feedback)* con el fin de realizar pequeñas mejoras y correcciones.

3.8 Plan de mantenimiento

Se realizan diariamente copias de seguridad de la base de datos de la aplicación. La política es:

1. Copia semanal completa, realizada todos los sábados.
2. Copia incremental diaria.
3. Copia diaria del registro de transacciones.

Las copias anteriores se mantienen con una antigüedad de 31 días. Además de la política anterior, todos los días 1 de cada mes se realiza una copia completa, de la cual se almacenan las correspondientes al último año.

De esta forma, se dispone, durante un año, de una copia de la base de datos del día 1 y, durante el último mes, se pude obtener el estado al final de cada uno de los días del mismo.

El resto de elementos que componen la aplicación no requieren copia de seguridad, ya que Endalia registra en cada actualización la versión que se instala, pudiendo realizar la reinstalación del producto en un breve plazo de tiempo una vez restaurado el servidor.

4. RESULTADOS OBTENIDOS

En esta sección se mostrarán los resultados de este proyecto, a través de capturas de pantalla de la aplicación que ilustren el comportamiento conseguido.

Dada la naturaleza de esta memoria no se ha incluido la totalidad de las capturas de pantalla de la aplicación. Éstas se encuentran disponibles en el documento de implementación, adjunto a esta memoria.

4.1 Página "Estructura organizativa"

A continuación (Figura 13), se muestra una captura de pantalla de la estructura organizativa. En esta sección, se muestra en la parte izquierda un árbol con la estructura organizativa de la empresa y en la parte derecha el listado de empleados junto con el puesto o puestos que ocupan y los datos asociados a estos:

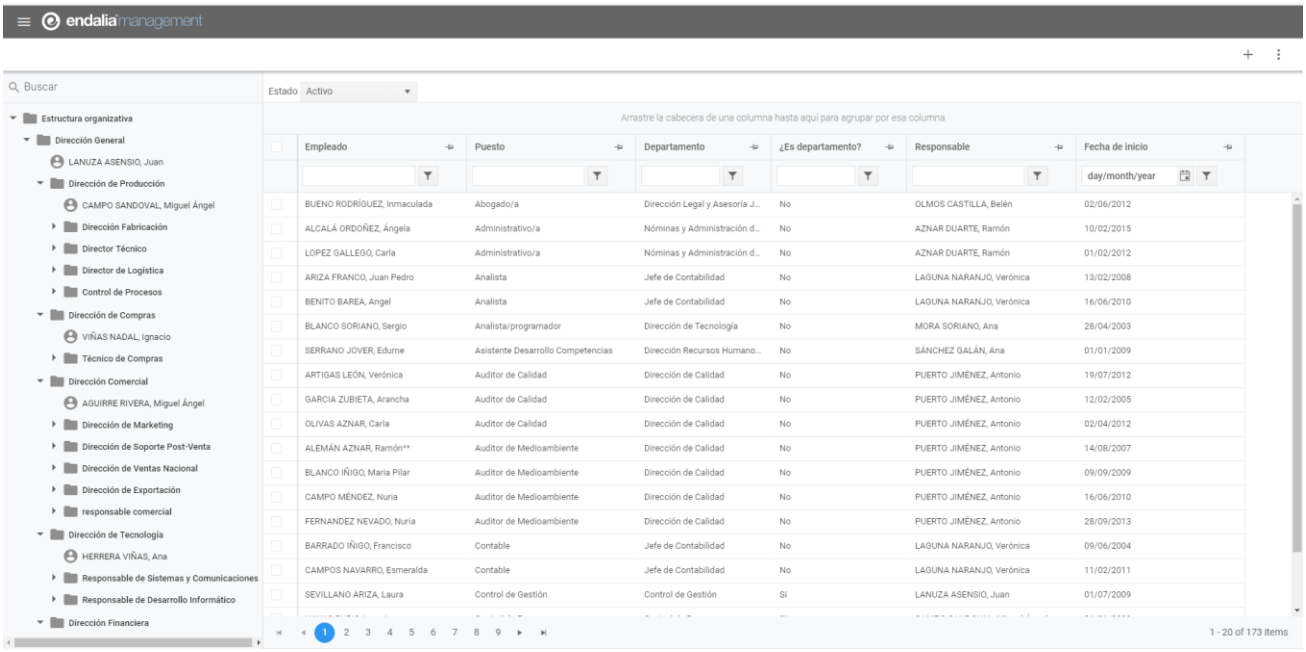


Figura 5. Página estructura organizativa

La página ofrece filtros que el usuario podrá utilizar para seleccionar el subconjunto de puestos que quiera ver. Estos puestos se pueden ver o editar haciendo clic en las filas. Seleccionando un nodo del árbol y mediante las acciones situadas en la parte superior derecha se podrán añadir puestos, asignar un empleado a un puesto o cambiar el estado de este.

Al seleccionar un puesto, la aplicación pasa a mostrar la información asociada al puesto y realizar modificaciones de esta.

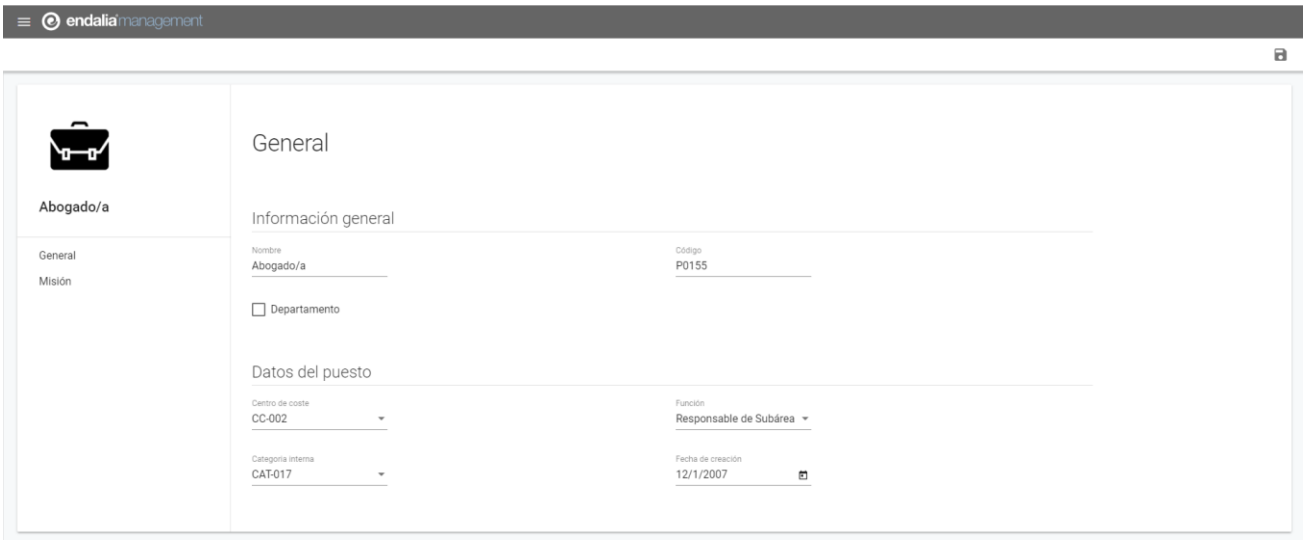


Figura 6. Página de información de puesto

Página “Empleados”

En la Figura 15 se muestra una captura de pantalla de la sección de “Empleados”. En esta sección, se visualiza un listado de los empleados activos o históricos.

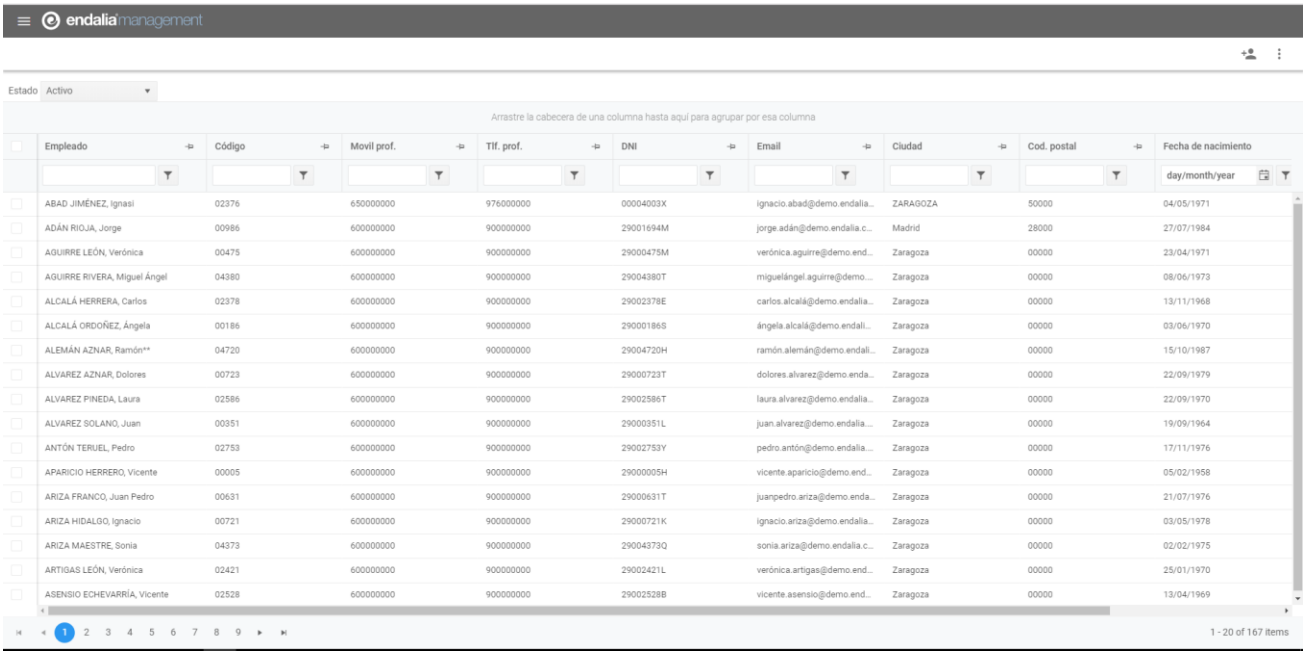


Figura 7. Página listado de empleados

La página ofrece filtros que el usuario podrá utilizar para seleccionar el subconjunto de empleados que quiera ver. Estos empleados se pueden ver o editar haciendo clic en las filas. Mediante las acciones situadas en la parte superior derecha se pueden añadir empleados o cambiar el estado de estos.

Al seleccionar un empleado, la aplicación muestra la información personal y profesional asociada al empleado y permite realizar modificaciones de esta.

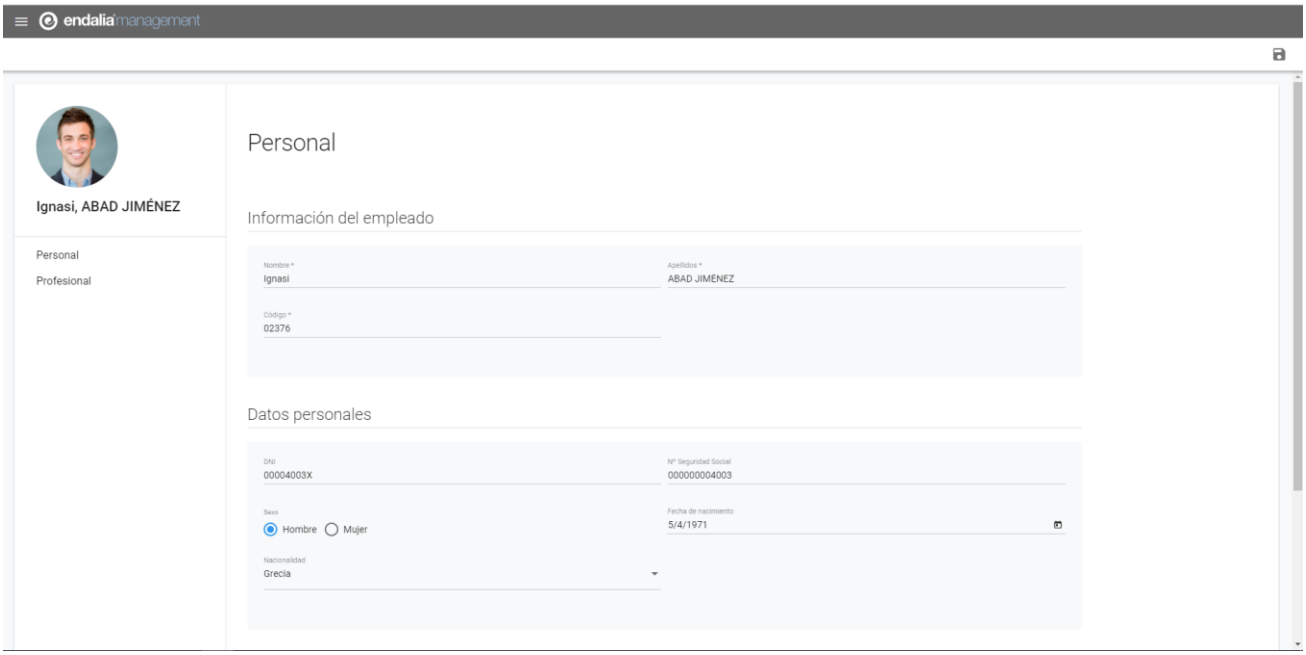


Figura 8. Página información empleado

4.2 Página “Periodos”

En la Figura 17 se muestra una captura de pantalla de la sección de “Periodos”. En esta sección, se visualiza un listado de los periodos activos o históricos.

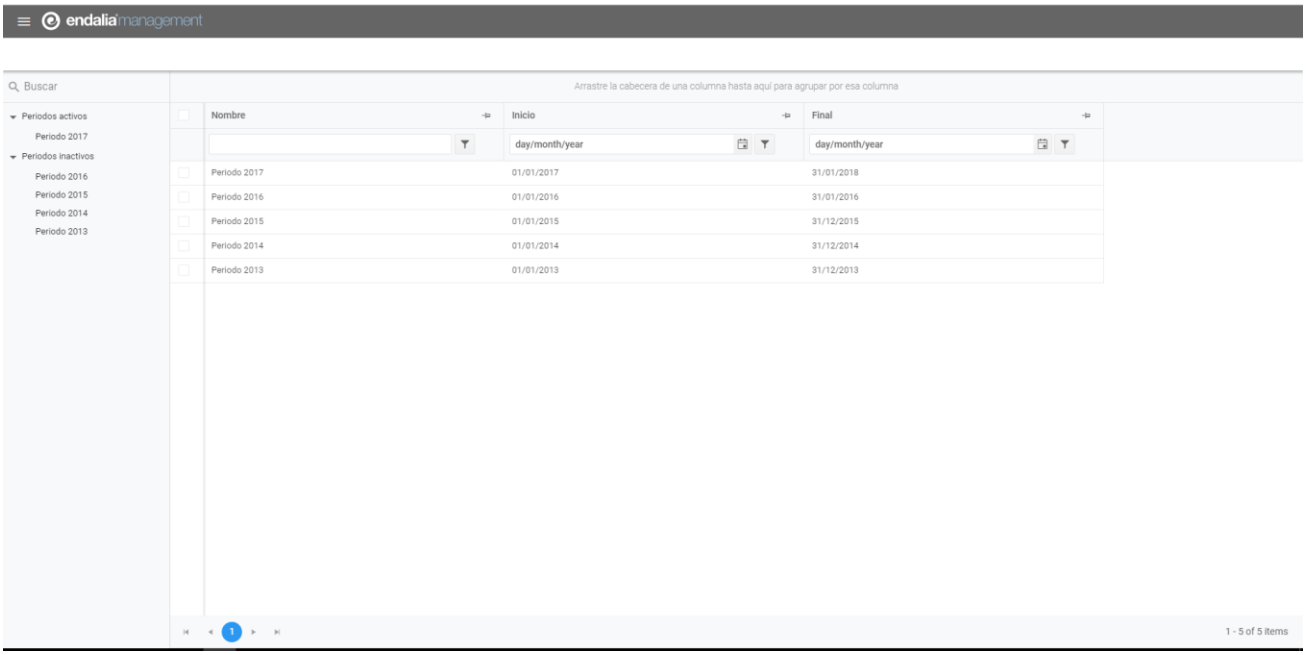


Figura 9. Página de periodos

Al seleccionar un periodo haciendo clic sobre la fila, la aplicación muestra un detalle de este (Figura 18) junto con tres pestañas: la primera de ellas muestra para cada tipo de convenio y ausencia cuantos días se pueden solicitar, la segunda un listado de empleados que tienen un máximo de días particular y, por último, una pestaña de observaciones acerca del periodo.

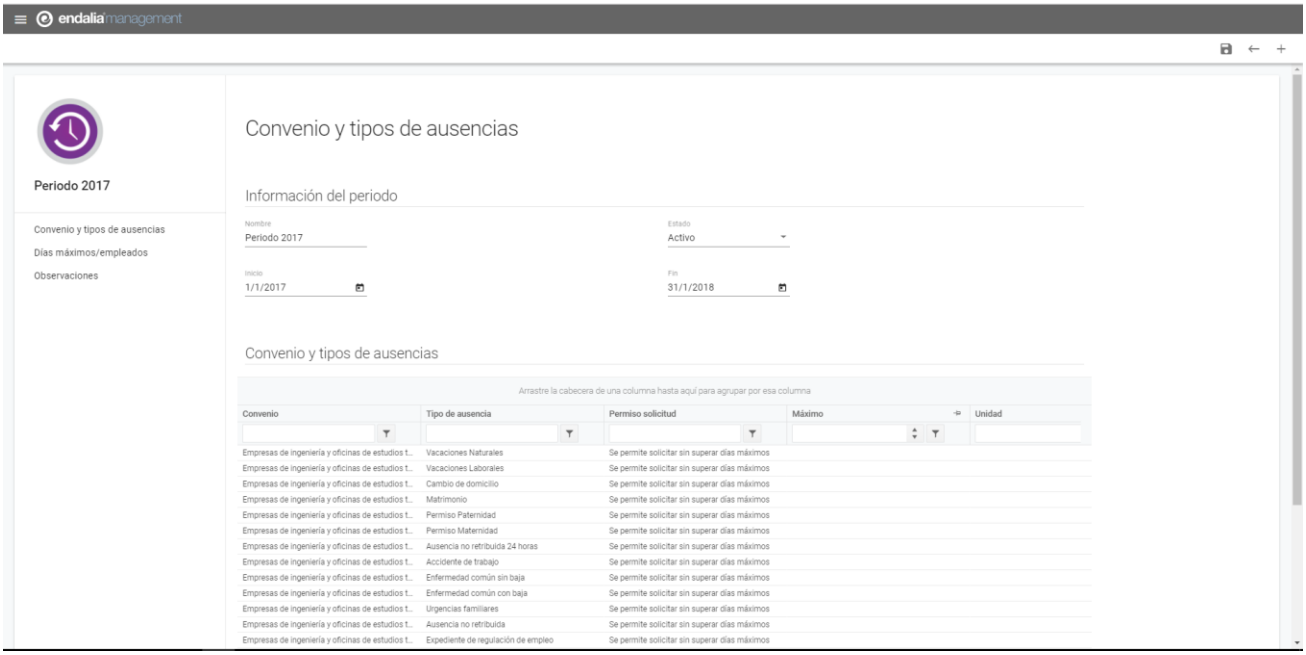


Figura 10. Página de información de periodo

4.3 Página “Solicitudes”

A continuación (Figura 19), se muestra una captura de pantalla de las solicitudes. En esta sección, se muestra en la parte izquierda un árbol con la estructura organizativa de la empresa y en la parte derecha el listado de solicitudes junto con información relevante de estas. Estas solicitudes se podrán filtrar por puestos mediante el árbol de estructura o mediante los filtros que se encuentran en la parte superior.

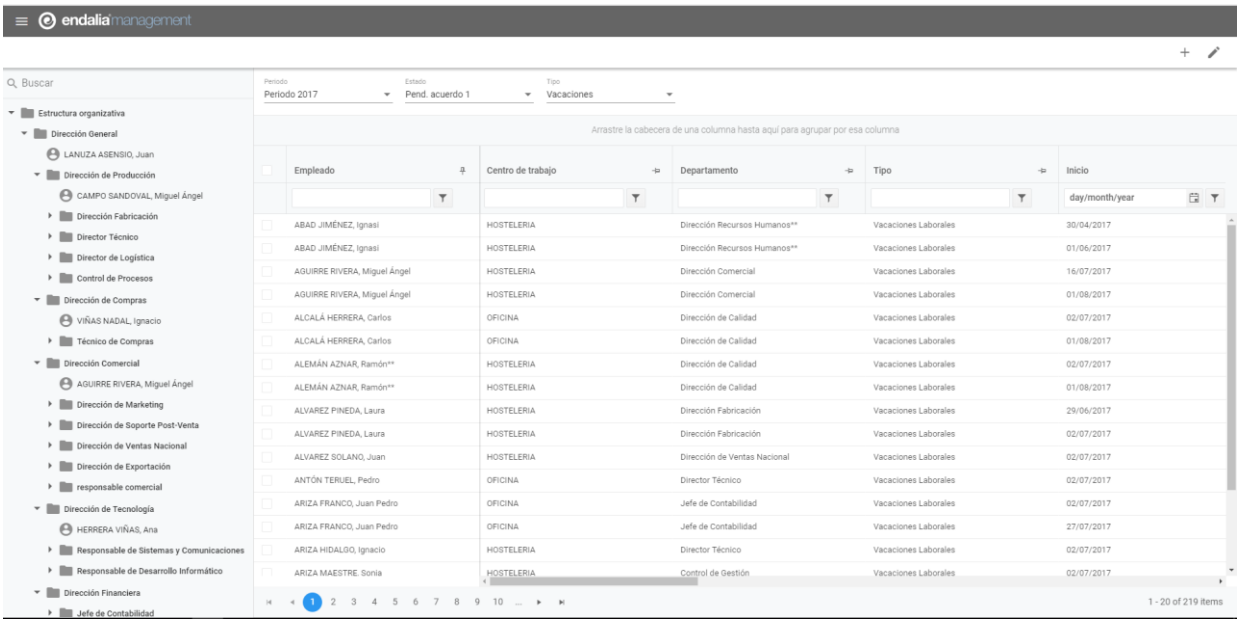


Figura 11. Página listado de solicitudes

Al seleccionar una solicitud, haciendo clic sobre la fila, la aplicación muestra un detalle de esta (Figura 14) informando del tipo de ausencia, su fecha de inicio y fin, el periodo al que pertenece, el estado en el que se encuentra y unas observaciones. Mediante las acciones situadas en la parte superior derecha se pueden añadir nueva solicitud para lo que habrá que rellenar una ficha análoga a la de la Figura 19.

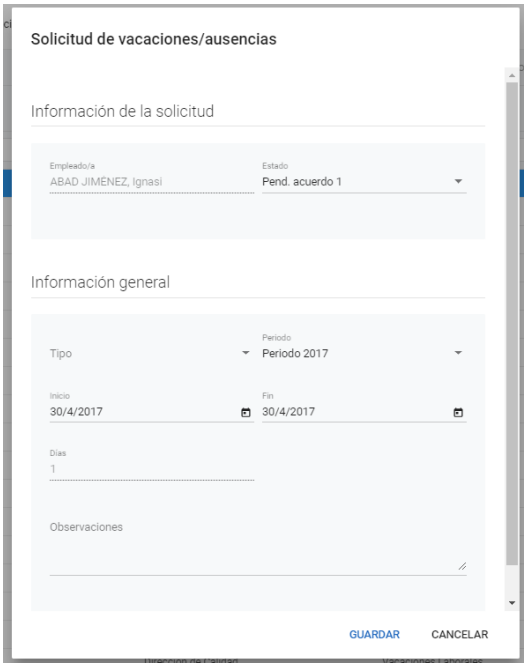


Figura 12. Pop up solicitud de vacaciones/ausencias

5. CONCLUSIONES

En los siguientes apartados se van a presentar las conclusiones de este trabajo. Primero se estudiará el grado de cumplimiento de los diferentes objetivos definidos en el apartado 1.2 para poder obtener conclusiones objetivas del proyecto. A continuación, se presentarán tanto las líneas de trabajo futuro en relación con este proyecto como unas ideas de mejora que, si bien no habían sido requeridas hasta el momento, de implementarlas darían mayor valor al sistema. Por último, se realizará una pequeña valoración personal sobre el trabajo y su desarrollo.

5.1 Conclusiones del proyecto

Para valorar el desarrollo del trabajo y sacar conclusiones sobre éste se va a proceder a analizar los resultados obtenidos en función de los objetivos definidos en la sección 1.2 y de los requisitos definidos en el anexo “”Especificación de requisitos”.

El objetivo principal de este proyecto era crear una aplicación web que permitiera la gestión de recursos humanos para pymes, desarrollando los módulos de organización, personas y vacaciones/ausencias. El sistema desarrollado cumple este objetivo ofreciendo administración por parte del responsable de Recursos Humanos.

Se han cumplido los objetivos relacionados con los requisitos funcionales, que englobaban las distintas acciones que la aplicación podría realizar. Se han desarrollado todas las secciones detalladas en los requisitos y se ha implementado el sistema adecuándose a la calidad general del software establecida en Endalia.

Otro objetivo, esta vez relacionado con el desarrollo profesional, era el de conocer, utilizar y adaptarse a las tecnologías usadas en la empresa, y a las nuevas, como Angular, que serían utilizadas en el desarrollo de este proyecto. Se puede decir que el objetivo de conocer, utilizar y adaptarse a las tecnologías, se ha cumplido con éxito. Sin embargo, aún queda mucho margen de mejora en este ámbito.

Lo mismo se puede decir del objetivo de comprender, analizar y justificar las metodologías, procesos, planificaciones y herramientas utilizadas en la organización, y cómo de adecuadas son para el trabajo a desarrollar. Si bien este proyecto ha bastado para tener una comprensión general de todos estos aspectos, se requerirá de trabajo en el futuro para llegar a dominar este ámbito.

En cuanto a los últimos objetivos, referentes a formar parte de un equipo de desarrollo de software en un entorno empresarial, de forma que complemente la formación recibida en la Universidad, también se puede decir que se han cumplido. En los meses en los que ha tenido lugar este proyecto, se ha tenido la oportunidad de poner a prueba todo lo aprendido y de complementarlo de la mano del resto de profesionales del equipo, lo que tiene un gran valor en la formación de un profesional de la ingeniería.

5.2 Líneas de mejora y trabajo futuro

Como se ha ido explicando a lo largo de esta memoria, solo se han desarrollado tres módulos. Por esto, el trabajo futuro a nivel funcional consistiría en desarrollar todos los módulos que puedan ser útiles para las pequeñas y medianas empresas: formación, selección, dirección por objetivos, competencias y desarrollo, viajes y gastos, prevención de riesgos laborales y planificación de turnos. Otro aspecto sería la mejora y completitud de los módulos desarrollados en este proyecto, ya que para la primera iteración se ha seleccionado un conjunto básico de funcionalidad de dichos módulos. Esto se complementará con un servicio SaaS de gestión laboral y nómina. A nivel tecnológico se va a desarrollar una aplicación móvil que permita agilizar ciertos procesos por parte del empleado. A modo de ejemplo, la comunicación al servicio de asesoramiento laboral de una baja por enfermedad que suponga para el empleado dos clics en un dispositivo móvil agiliza y facilita estos procesos administrativos.

Por otro lado, se considera que el sistema tiene margen de mejora en el aspecto de la accesibilidad. Se espera que en el futuro sea compatible con el estándar internacional “Pautas de Accesibilidad para el Contenido Web (WCAG) 2.0”, en el que se detallan una serie de pautas que debe cumplir una aplicación web para tener un alto grado de accesibilidad. Entre esas pautas destaca, por ejemplo, el hecho de facilitar el uso de la aplicación a personas con algún tipo de discapacidad visual mediante la escucha.

5.3 Valoración personal

Estoy muy satisfecha con el proyecto que he tenido la oportunidad de desarrollar. Principalmente porque ha supuesto una experiencia muy importante en mi formación como ingeniera, complementándose con lo aprendido en la universidad. Integrarme en un equipo profesional de desarrolladores de software y poder participar en la totalidad de un proyecto empresarial han sido, sin duda, algunos de los aspectos más enriquecedores, no sólo de este trabajo, sino de toda mi formación.

El trabajo en Endalia ha sido muy satisfactorio y gratificante, ya que he tenido ocasión de conocer las dinámicas de la organización y de trabajar con sus tecnologías y herramientas. Además, la relación con mis compañeros de equipo ha sido excelente, de forma que fue muy sencillo sentirme integrada en la organización.

Como conclusión, me siento muy orgullosa del trabajo realizado y satisfecha por haberlo sacado adelante cumpliendo con los planes y objetivos de la organización e incorporando una nueva tecnología. Realizar este Trabajo de Fin de Grado en una empresa como Endalia ha sido una experiencia muy positiva que sin duda recomendaría a aquellos que busquen una manera de comenzar en el vasto mundo del desarrollo de software.

6. BIBLIOGRAFÍA

[W1] Microsoft.com. (2018). *Microsoft .NET Framework 4.5 from Official Microsoft Download Center*. Disponible en: https://www.microsoft.com/es-es/download/details.aspx?id=30653 [Accedido 25 enero 2018].

[W2] Es.wikipedia.org. (2018). *Microsoft Visual Studio*. Disponible en: https://es.wikipedia.org/wiki/Microsoft_Visual_Studio [Accedido 25 enero 2018].

[W3] Telerik.com. (2018). Kendo UI Web Components - Build Better Apps Faster. Disponible en: https://www.telerik.com/kendo-ui [Accedido 25 enero 2018].

[W4]: Nhibernate.info. (2018). Home - NHibernate. Disponible en: http://nhibernate.info/ [Accedido 25 enero 2018].

[W5] Msdn.microsoft.com. (2018). Usar Visual Studio con Git. Disponible en: https://msdn.microsoft.com/es-es/library/hh850437(v=vs.120).aspx [Accedido 25 enero 2018].

[W6] En.wikipedia.org. (2018). *Team Foundation Server*. Disponible en: https://en.wikipedia.org/wiki/Team_Foundation_Server [Accedido 25 enero 2018].

[W7] Endalia HR. (2018). Software de Recursos Humanos y Outsourcing de nómina. Disponible en: https://www.endalia.com/ [Accedido 25 enero 2018].

[W8] Microsoft. (2018). .NET. Disponible en: http://www.microsoft.com/net [Accedido 25 enero 2018].

[W9] Uml.org. (2018). Welcome To UML Web Site!. Disponible en: http://www.uml.org [Accedido 25 enero 2018].

[W10] Material.angular.io. (2018). Angular Material. Disponible en: https://material.angular.io/ [Accedido 25 enero 2018].

[W11] Telerik.com. (2018). Kendo UI Web Components - Build Better Apps Faster. Disponible en: https://www.telerik.com/kendo-ui [Accedido 25 enero 2018].

[W12] Validator.w3.org. (2018). The W3C Markup Validation Service. Disponible en: https://validator.w3.org/ [Accedido 25 enero 2018].

Universidad Zaragoza
1542

Trabajo Fin de Grado

Desarrollo de un sistema de gestión de Recursos Humanos para pymes

Development of a Human Resources Management System for SMEs

ANEXOS

Autora

Marina Ariño Armengol

Director

Fernando Cortés Franco

Ponente

Raquel Trillo Lado

Universidad de Zaragoza
Escuela de Ingeniería y Arquitectura

2018

endalia®
HR Consulting & Software

ESPECIFICACIÓN DE REQUISITOS

DESARROLLO DE UN SISTEMA DE GESTIÓN DE RECURSOS HUMANOS PARA PYMES

VERSIÓN 1.0
PUBLICADO EL 30/12/2017

HISTÓRICO DE REVISIONES

Fecha	Versión	Descripción	Autor
30/12/2017	1.0	Redacción inicial del documento	Marina Ariño Armengol

ÍNDICE

1. INTRODUCCIÓN

1.1 Propósito del documento

El presente documento pretende presentar y describir los requisitos del sistema a desarrollar. Estos requisitos se dividen en requisitos funcionales, que pretenden identificar la funcionalidad objetivo del sistema, y en requisitos no funcionales, que son relativos a la seguridad, interfaz, rendimiento o escalabilidad del sistema a desarrollar. Además, es esta especificación lo que guiará el resto del desarrollo software del proyecto.

Una buena especificación de requisitos debe ser [R1]:

- Correcta
- Inequívoca
- Completa
- Consistente
- Categorizada según su importancia
- Comprobable
- Modificable
- Identificable

1.2 Alcance del documento

Este documento describe el resultado de la fase de estudio de los requerimientos. A lo largo del mismo se describirán los distintos requisitos funcionales y no funcionales que el sistema desarrollado debe cumplir.

1.3 Acrónimos

- RF: Requisito funcional
- RNF: Requisito no funcional
- PDF: Portable Document Format
- GUI: Graphical User Interface

1.4 Definiciones

- Graphical User Interface (GUI): es un tipo de interfaz de usuario que permite a los usuarios interactuar con un sistema mediante iconos visuales e indicadores gráficos en lugar de usar una interfaz basada en texto o comandos por teclado. [W1]

1.5 Referencias

Este documento no contiene referencias a otros documentos del trabajo.

1.6 Resumen

El objetivo de este documento es el de presentar los resultados obtenidos del estudio de las necesidades, identificando los diferentes requisitos que el sistema diseñado debe cumplir. Se compone de los siguientes apartados:

1. Una introducción en la que se presenta el documento, su propósito y su alcance, así como una lista de acrónimos, definiciones y referencias que puedan ser de utilidad.
2. La descripción general, en la que se presenta la funcionalidad del sistema a desarrollar, así como el tipo de usuarios que lo utilizará
3. Un listado detallado con los requisitos funcionales del sistema, su nivel de criticidad y sus dependencias.
4. Un listado detallado con los requisitos no funcionales del sistema, su nivel de criticidad y sus dependencias.
5. La bibliografía utilizada en esta fase del proyecto.

2. DESCRIPCIÓN GENERAL

2.1 Funciones generales del sistema

Este sistema de gestión de recursos humanos para pymes está compuesto de tres módulos. El módulo Organización permitirá visualizar la estructura organizativa de la empresa y gestionar la creación de nuevos puestos y la edición de los mismos, así como la asignación de empleados a los puestos. En el módulo personas se mostrará un listado de empleados de la organización permitiendo la creación de nuevos y la modificación de los ya existentes. Por último, el módulo Vacaciones y ausencias se visualiza los periodos de solicitud y las solicitudes de vacaciones/ausencias, junto con la creación, modificación y borrado.

2.2 Características de los usuarios

Los usuarios que interactuarán con el sistema serán los responsables de RRHH de la empresa ya que serán los encargados de gestionar todo aquello con la organización de la empresa. Se deberá considerar que los usuarios no poseen una formación específica, por lo que es posible que no estén familiarizados con el uso de aplicaciones web.

3. REQUISITOS FUNCIONALES DEL SISTEMA

Se van a presentar los requisitos agrupados por módulo.

3.1 Organización

Nombre	**RF – 01: Visualización de puestos activos en estructura jerárquica**
Descripción	El módulo organización contará con una pantalla donde se muestre la estructura organizativa de la empresa. Además, ésta deberá permitir algún tipo de interacción que permita alterar los puestos mostradas (filtrado, ordenación).
Nivel de criticidad	Alto
Dependencias	-

Nombre	**RF – 02: Visualización de puestos históricos**
Descripción	El módulo organización contará con una pantalla donde se muestre un listado de los puestos históricos. Además, ésta deberá permitir algún tipo de interacción que permita alterar los puestos mostradas (filtrado, ordenación).
Nivel de criticidad	Alto
Dependencias	-

Nombre	**RF – 03: Creación de un nuevo puesto en estructura**
Descripción	Los usuarios podrán crear uno o varios puestos al mismo nivel que otro o por debajo de este. El proceso de creación de puestos deberá ser intuitivo y lo más breve posible.
Nivel de criticidad	Alto
Dependencias	-

Nombre	**RF – 04: Modificación de un puesto**
Descripción	Los usuarios podrán modificar la información asociada a los puestos.
Nivel de criticidad	Alto
Dependencias	Creación de un nuevo puesto en estructura

Nombre	RF – 05: Asignar empleado a puesto
Descripción	Los usuarios podrán añadir a uno o varios empleados activos a un puesto.
Nivel de criticidad	Alto
Dependencias	-

Nombre	RF – 07: Cambiar estado de un puesto
Descripción	Se podrá cambiar el estado de los puestos activos a histórico y viceversa, especificando en qué posición de la estructura organizativa se debe situar.
Nivel de criticidad	Alto
Dependencias	-

Nombre	RF – 08: Visualización organigrama
Descripción	El módulo organización contará con una pantalla donde se muestre el organigrama. Además, ésta deberá permitir algún tipo de interacción que permita alterar los puestos mostradas (filtrado, ordenación).
Nivel de criticidad	Alto
Dependencias	-

3.2 Personas

Nombre	RF – 09: Visualización de empleados
Descripción	El módulo personas contará con una pantalla donde se muestre un listado de los empleados de la organización. Además, ésta deberá permitir algún tipo de interacción que permita alterar los empleados mostrados (filtrado, ordenación).
Nivel de criticidad	Alto
Dependencias	-

Nombre	RF – 10: Creación de un nuevo empleado
Descripción	Los usuarios podrán crear uno o varios empleados. El proceso de creación de empleados deberá ser intuitivo y lo más breve posible.
Nivel de criticidad	Alto
Dependencias	-

Nombre	RF – 11: Modificación de un empleado
Descripción	Los usuarios podrán modificar la información asociada a los empleados.
Nivel de criticidad	Alto
Dependencias	Creación de un nuevo empleado

Nombre	RF – 12: Cambio de estado de un empleado
Descripción	Se podrá cambiar el estado de un empleado de activo a histórico y viceversa.
Nivel de criticidad	Alto
Dependencias	-

3.3 Vacaciones/ausencias

Nombre	RF – 13: Visualización de periodos
Descripción	El módulo vacaciones/ausencias contará con una pantalla donde se muestre un listado de los periodos de solicitud. Además, ésta deberá permitir algún tipo de interacción que permita alterar los periodos mostradas (filtrado, ordenación).
Nivel de criticidad	Alto
Dependencias	-

Nombre	RF – 14: Creación de un nuevo periodo
Descripción	Los usuarios podrán crear uno nuevo periodo. El proceso de creación de periodos deberá ser intuitivo y lo más breve posible.
Nivel de criticidad	Alto
Dependencias	-

Nombre	RF – 15: Modificación de un periodo
Descripción	Los usuarios podrán modificar la información asociada a los periodos.
Nivel de criticidad	Alto
Dependencias	Creación de un nuevo periodo

Nombre	RF – 16: Visualización de solicitudes
Descripción	El módulo personas contará con una pantalla donde se muestre un listado de las solicitudes realizadas por los empleados. Además, ésta deberá permitir algún tipo de interacción que permita alterar las solicitudes mostradas (filtrado, ordenación).
Nivel de criticidad	Alto
Dependencias	-

Nombre	RF – 17: Creación de una nueva solicitud
Descripción	Los usuarios podrán crear una nueva solicitud de vacación o ausencia. El proceso de creación de solicitudes deberá ser intuitivo y lo más breve posible.
Nivel de criticidad	Alto
Dependencias	-

Nombre	RF – 18: Modificación de solicitudes
Descripción	Los usuarios podrán modificar la información asociada a las solicitudes de ausencias o vacaciones.
Nivel de criticidad	Alto
Dependencias	Creación de una nueva solicitud

4. REQUISITOS NO FUNCIONALES DEL SISTEMA

Nombre	**RNF – 01: Eficiencia del sistema**
Descripción	Toda funcionalidad del sistema y navegación entre las diferentes secciones del sistema debe responder al usuario en menos de 3 segundos.
Nivel de criticidad	Alto
Dependencias	-

Nombre	**RNF – 02: Aplicación parametrizable**
Descripción	El sistema será fácilmente parametrizable en función de las necesidades del cliente.
Nivel de criticidad	Medio
Dependencias	-

Nombre	**RNF – 03: Interacción con el usuario mediante GUI consistente**
Descripción	La interacción con el usuario deberá realizarse a través de una GUI consistente. Es decir, deberá respetar el diseño de elementos como tablas, botones o colores, así como la distribución de las pantallas.
Nivel de criticidad	Alto
Dependencias	-

Nombre	**RNF – 04: Aplicación intuitiva**
Descripción	El sistema deberá ser intuitivo y fácil de usar, incluso para usuarios poco familiarizados con este tipo de aplicaciones o tecnologías.
Nivel de criticidad	Medio
Dependencias	-

Nombre	RNF – 05: Tolerancia a fallos
Descripción	El sistema deberá ser tolerante a fallos, por lo que deberá poder recuperarse de un error no controlado en el sistema y ofrecer información relativa a dicho error.
Nivel de criticidad	Alto
Dependencias	-

Nombre	RNF – 06: Disponibilidad del sistema
Descripción	El sistema deberá ser accesible las 24 horas del día y los 7 días de la semana, adecuándose a la disponibilidad del sistema global de la entidad. Además, deberá permitir el acceso concurrente a múltiples usuarios sin que esto conlleve una pérdida de rendimiento o inconsistencias en los datos.
Nivel de criticidad	Alto
Dependencias	-

Nombre	RNF – 07: Resolución
Descripción	El módulo deberá ser correctamente visualizado en pantallas con resoluciones iguales o superiores a 1024x768, adecuándose a los estándares del resto de sistemas de la organización.
Nivel de criticidad	Alto
Dependencias	-

Nombre	RNF – 08: Seguridad e integridad de los datos
Descripción	El sistema contiene información potencialmente confidencial por lo que no se deberá permitir el acceso a un usuario no registrado en la aplicación. Por otro lado, se deberá asegurar la integridad de los datos en las transacciones con la base de datos.
Nivel de criticidad	Alto
Dependencias	-

Nombre	**RNF – 09: Compatibilidad con distintos navegadores**
Descripción	El sistema deberá funcionar como mínimo en los siguientes navegadores: Internet Explorer (versiones superiores a la 8.0), Mozilla Firefox y Google Chrome, adecuándose a los estándares de la aplicación de Endalia.
Nivel de criticidad	Alto
Dependencias	-

5. BIBLIOGRAFÍA

5.1 Referencias

[R1] IEEE Recommended Practice for Software Requirements Specifications, Std 830-1998.

5.2 Referencias web

[W1] http://www.wikipedia.org

endalia®
HR Consulting & Software

ANÁLISIS

DESARROLLO DE UN SISTEMA DE GESTIÓN DE RECURSOS HUMANOS PARA PYMES

VERSIÓN 1.0
PUBLICADO EL 30/12/2017

HISTÓRICO DE REVISIONES

Fecha	Versión	Descripción	Autor
30/12/2017	1.0	Redacción inicial del documento	Marina Ariño Armengol

ÍNDICE

1. INTRODUCCIÓN

1.1 Propósito del documento

En este documento se presenta la fase de análisis de este proyecto software. Partiendo de los requisitos descritos en el documento de especificación de requisitos se va a realizar un análisis de la arquitectura del sistema a desarrollar, identificando casos de uso y la estructura de paquetes. También se realizará un estudio de los requisitos especiales del sistema. Los resultados de este análisis marcarán las pautas a seguir en las siguientes fases del desarrollo.

1.2 Alcance del documento

Este documento muestra los resultados de la fase de análisis del proyecto. En él, se describirá el proceso de análisis para a continuación realizar un estudio de los casos de uso y sus diferentes actores, así como un análisis de paquetes.

1.3 Acrónimos

- GUI: Graphic User Interface.
- UML: Unified Modeling Language.

1.4 Definiciones

- Caso de uso: especificación de las secuencias de acciones que pueden ser efectuadas por un sistema, subsistema o clase por interacción con actores externos.
- Diagrama de actividades: notación gráfica que representa los flujos de trabajo de los componentes en un sistema, paso a paso. Muestra el flujo de control general del sistema.
- Diagrama de casos de uso: notación gráfica que describe un caso de uso.
- Diagrama de secuencia: notación gráfica que muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo.
- UML: lenguaje de modelado de sistemas de software desarrollado por *Rational* y de uso extendido.

1.5 Referencias

En este documento se referencian los siguientes documentos del proyecto:

- Especificación de requisitos: documento en el que se especifican los requisitos del sistema.

1.6 Resumen

En este documento se presenta la fase de análisis del trabajo. Se compone de los siguientes apartados:

1. Introducción, propósito y alcance del documento.
2. Descripción del proceso de análisis utilizado en este proyecto.
3. Identificación y descripción de los casos de uso.
4. Realización de un análisis de paquetes.
5. Identificación de los requisitos especiales del sistema.
6. Presentación de la bibliografía utilizada para la realización de este documento.

2. DESCRIPCIÓN DEL PROCESO

El flujo de trabajo de análisis se compone de distintas actividades, como son: análisis de los casos de uso, análisis de paquetes y análisis de requisitos especiales comunes.

El análisis de paquetes pretende identificar los componentes esenciales del sistema, garantizando que mediante los mismos se cumplen los objetivos definidos en el documento de requisitos, así como la descripción de los casos de uso, y describir las dependencias entre paquetes, de forma que pueda estimarse el efecto de cambios futuros.

El análisis de los requisitos especiales comunes tiene el objetivo de identificar las peculiaridades del sistema que determinarán restricciones y especificaciones para fases futuras del proyecto.

3. ANÁLISIS DE LOS CASOS DE USO

3.1 Introducción

En este apartado se va a realizar el análisis de los casos de uso del sistema, identificando todos los procesos que tienen lugar en el sistema.

El análisis de un caso de uso se realiza con el objetivo de profundizar en la funcionalidad desarrollada por el mismo. Para formalizar el detalle de los casos de uso, se van a utilizar las siguientes técnicas:

- Diagramas de caso de uso: describen la relación entre los usuarios del sistema y los casos de uso.
- Diagramas de secuencia: muestran la vista dinámica del caso de uso. Es decir, describen las interacciones existentes a lo largo del tiempo.
- Diagramas de actividades: describen el flujo de trabajo del caso de uso.

3.2 Actores de los casos de uso

En el ámbito de este proyecto, se denominan actores a los distintos usuarios que interaccionan con la aplicación, los cuales han sido caracterizados en el documento de especificación de requisitos. Para realizar el presente análisis, se distinguirán dos tipos de usuarios diferenciados:

- Responsables RRHH: son usuarios que acceden al sistema de gestión de recursos humanos con el objetivo de gestionar la información de los módulos presentados en este trabajo.
- Usuario portal del empleado: usuarios que corresponden con el resto de empleados de la organización que acceden al portal web y visualizan la información gestionada por el responsable de RRHH. Dentro de este conjunto se encuentran dos subconjuntos que influyen al módulo de vacaciones/ausencias:
    - Solicitantes: son usuarios que acceden al portal web con el objetivo de manipular sus propias solicitudes. Pueden crear una nueva solicitud, editar una ya existente, enviarla para aprobación, o simplemente visualizar sus solicitudes a modo de consulta.
    - Aprobadores: son usuarios de tipo solicitante, pero con privilegios añadidos. Además de realizar las acciones propias de los usuarios de tipo solicitante, pueden visualizar las solicitudes de los empleados situados por debajo en la estructura organizativa y aprobar las solicitudes que estén en una fase que dependa de estos.

3.3 Casos de uso

En esta sección se especifican diversos casos de uso del sistema de gestión de Recursos Humanos para pymes. Cada caso de uso de presentará mediante una breve descripción y un diagrama explicativo. Para comenzar, se mostrará un esquema general de los casos de uso del sistema, lo que ofrecerá una visión global del funcionamiento del sistema desde el punto de vista del usuario. A continuación se detallarán los distintos casos de uso.

3.3.1 Esquema general de los casos de uso

El esquema general de los casos de uso de la aplicación se muestra en la Figura 1. En él se presenta la base sobre la que se van a plantear los casos de uso explicados más adelante. Para comenzar, se mostrará un esquema general de los casos de uso del sistema divididos por módulo, lo que ofrecerá una visión global del funcionamiento del sistema desde el punto de vista del usuario responsable de RRHH. A continuación, se detallarán los casos de uso de los distintos actores.

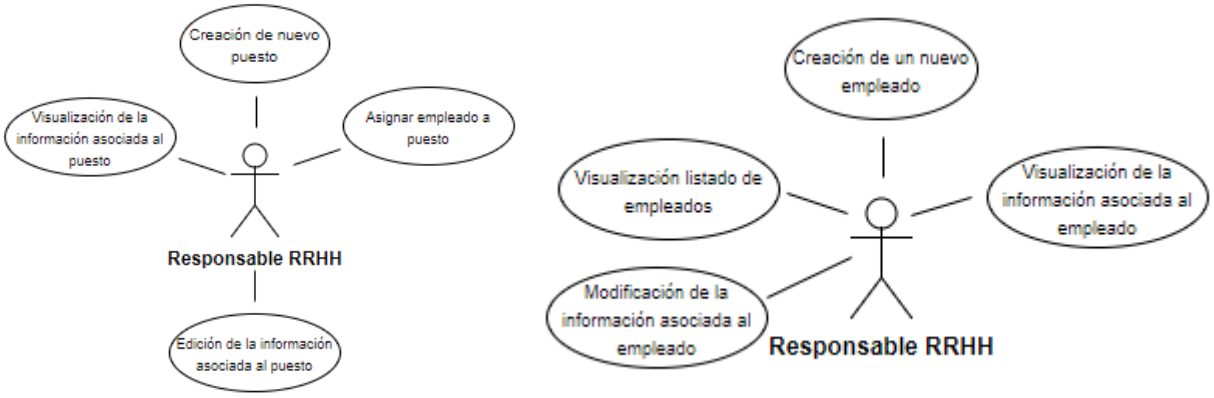


Figura 1. Casos de uso del módulo organización (izquierda) y personas (derecha)

Figura 2. Caso de uso del módulo vacaciones/ausencias

3.3.2 Módulo organización

A continuación se van a describir los casos de uso relacionados con el módulo organización. En total se dispone de cuatro casos de uso diferenciados, tal y como se muestra en la Figura 3. Como se ha explicado anteriormente, todos los casos de uso detallados a continuación requieren un acceso previo al módulo de organización, mediante autentificación previa introduciendo un nombre de usuario y una contraseña.

Figura 3. Casos de uso organización

Así, los casos de uso mostrados en la figura son los siguientes:

- Creación de un nuevo puesto.
- Visualización de la información asociada al puesto.
- Modificación de la información asociada al puesto.
- Asignar empleado a puesto.

A continuación se procede a detallar los diferentes casos de uso en orden, mostrando un diagrama de secuencia así como un diagrama de actividades para cada uno:

- Creación de un nuevo puesto. El usuario interactúa con la GUI del sistema y mediante un asistente crea un nuevo puesto con la información básica. Este puesto se guarda para la posterior modificación si se desea.
    - Diagrama de secuencia (Figura 4):

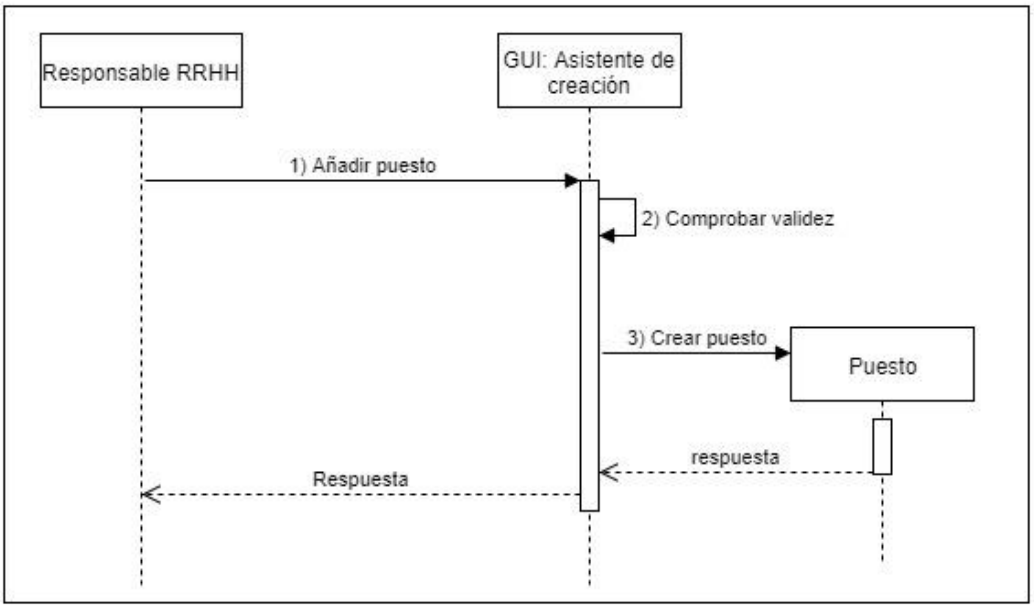


Figura 4. Diagrama de secuencia del caso de uso "Creación de un nuevo puesto"

    - Diagrama de actividades (Figura 5):

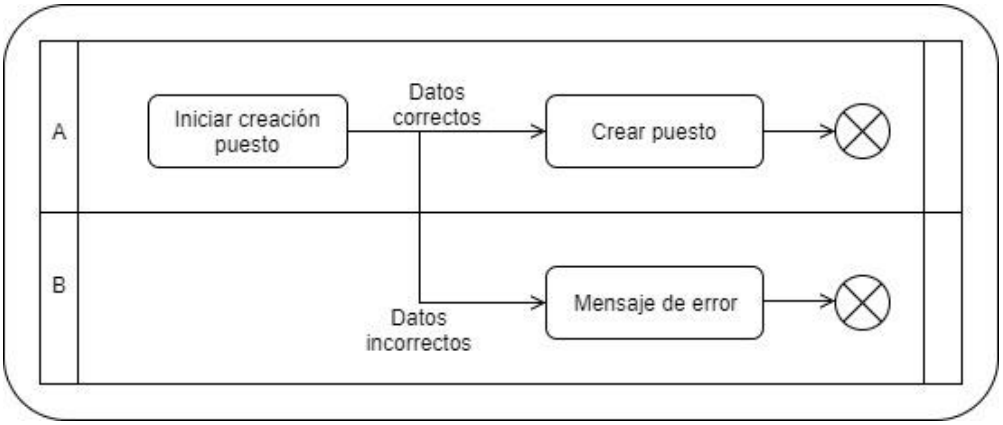


Figura 5. Diagrama de actividades del caso de uso "Creación de un nuevo puesto"

- Visualización de la información asociada al puesto. El usuario interactúa con la GUI seleccionando un puesto que haya sido creado. El sistema muestra a continuación la información correspondiente al puesto.
    - Diagrama de secuencia (Figura 6):

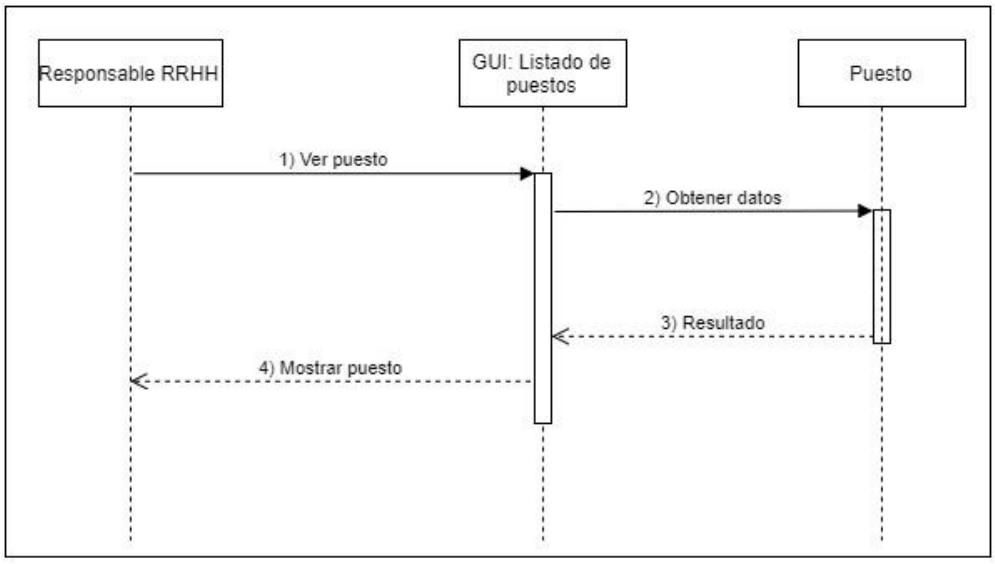


Figura 6. Diagrama de secuencia del caso de uso "Visualización de la información asociada al puesto"

    - Diagrama de actividades (Figura 7):

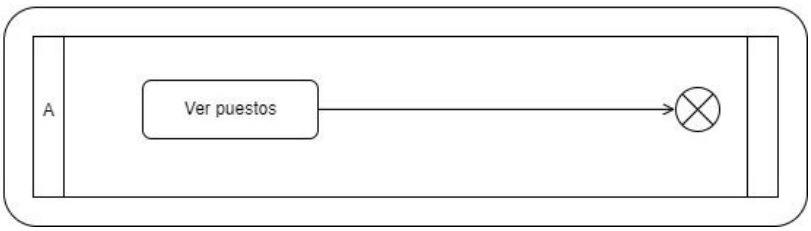


Figura 7. Diagrama de actividades del caso de uso "Visualización de la información asociada al puesto"

- Modificación de la información asociada al puesto. El usuario deberá seleccionar un puesto de la lista de puestos para a continuación modificar cualquiera de sus campos y validar los cambios. El sistema notificará del resultado de la actualización.
    - Diagrama de secuencia (Figura 8):

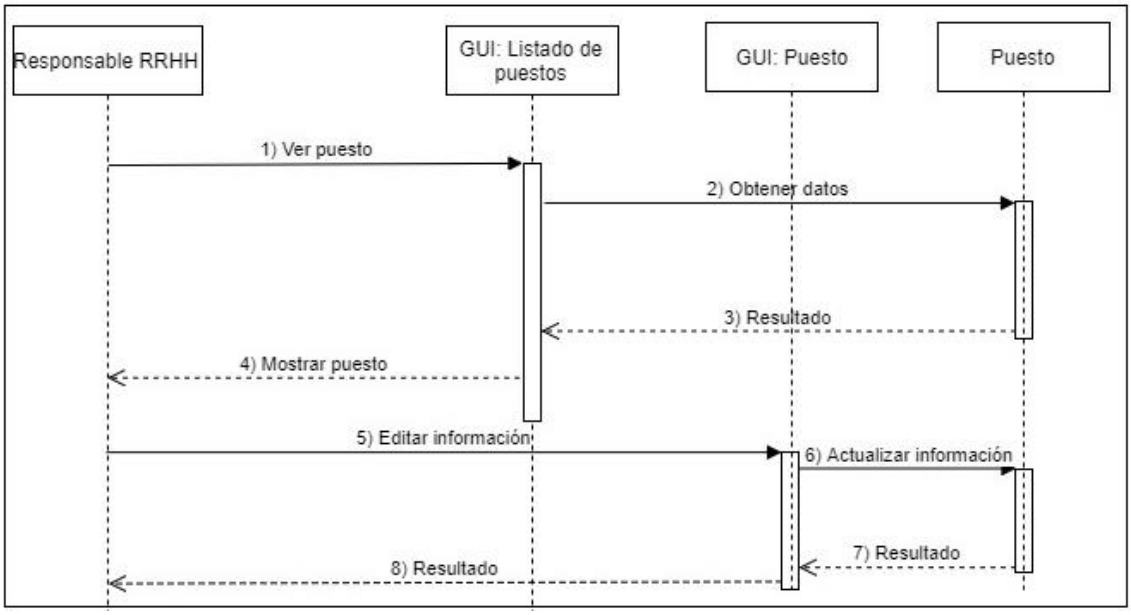


Figura 8. Diagrama de secuencia del caso de uso " Modificación de la información asociada al puesto".

    - Diagrama de actividades (Figura 8):

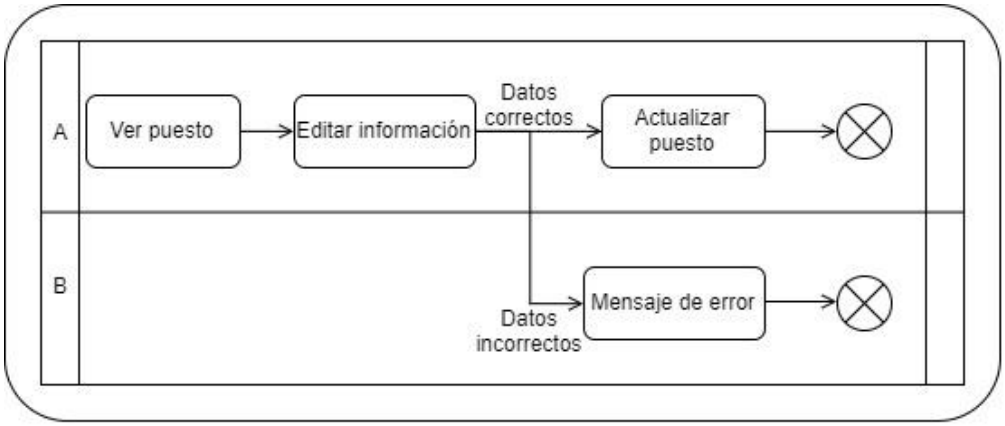


Figura 9. Diagrama de actividades del caso de uso "Edición de la información asociada al puesto"

- Asignar empleado a puesto. El usuario deberá seleccionar un para a continuación mediante un asistente asignar un nuevo empleado a este puesto.
    - Diagrama de secuencia (Figura 10):

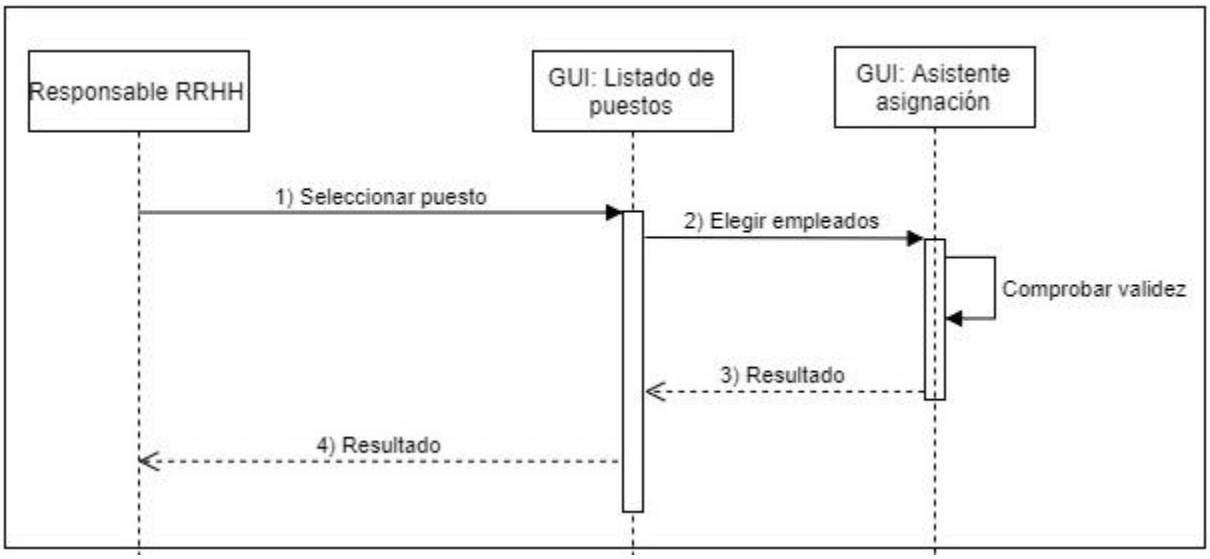


Figura 10. Diagrama de secuencia del caso de uso "Asignar empleado a puesto".

    - Diagrama de actividades (Figura 11):

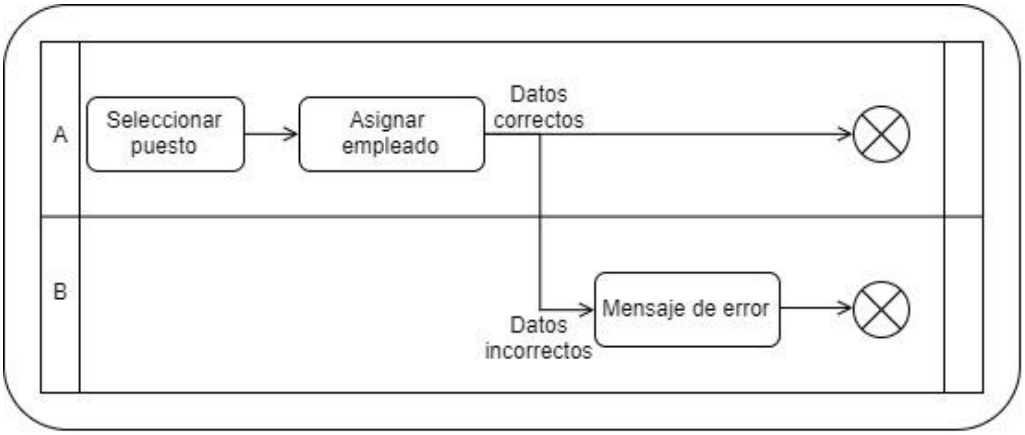


Figura 11. Diagrama de actividades del caso de uso "Asignar empleado a puesto".

3.3.3 Módulo personas

A continuación se van a describir los casos de uso relacionados con el módulo personas. En total se dispone de cuatro casos de uso diferenciados, tal y como se muestra en la Figura 12. Como se ha explicado anteriormente, todos los casos de uso detallados a continuación requieren un acceso previo al módulo de personas, mediante autentificación previa introduciendo un nombre de usuario y una contraseña.

Figura 12. Casos de uso módulo personas

Así, los casos de uso mostrados en la figura son los siguientes:

- Creación de un nuevo empleado.
- Visualización de la información asociada al empleado.
- Edición de la información asociada al puesto.
- Visualización listado de empleados.

A continuación se procede a detallar los diferentes casos de uso en orden, mostrando un diagrama de secuencia así como un diagrama de actividades para cada uno:

- Creación de un nuevo empleado. El usuario interactúa con la GUI del sistema y mediante un asistente crea un nuevo empleado con la información básica. Este empleado se guarda para la posterior modificación si se desea.
    - Diagrama de secuencia (Figura 13):

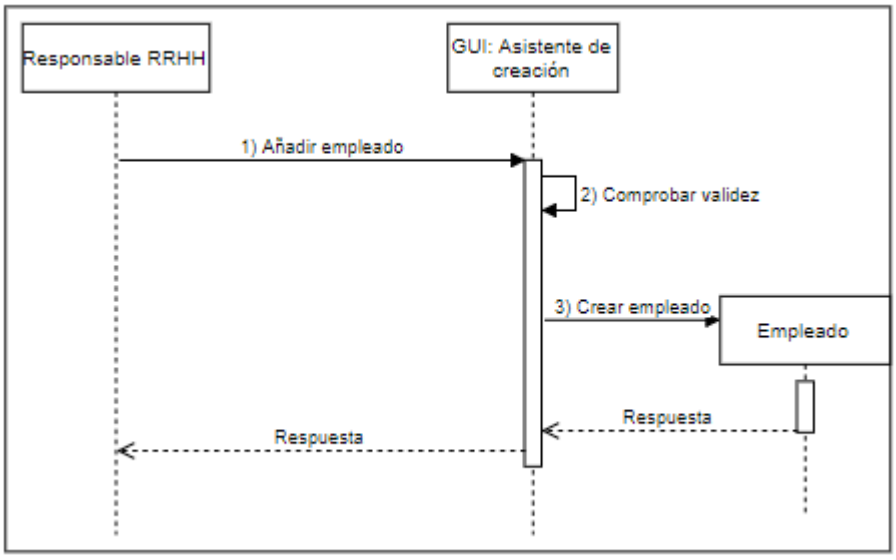


Figura 13. Diagrama de secuencia del caso de uso "Creación de un nuevo empleado".

- Diagrama de actividades (Figura 14):

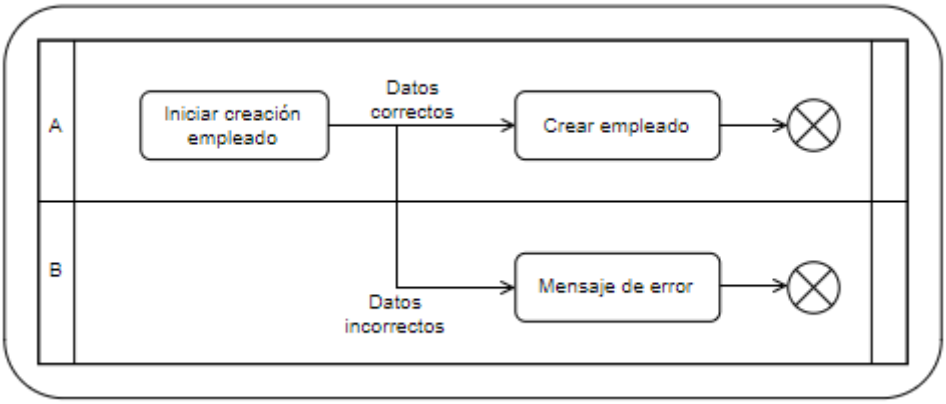


Figura 14. Diagrama de actividades del caso de uso "Creación de un empleado".

- Visualización de la información asociada al empleado. El usuario interactúa con la GUI seleccionando un empleado que haya sido creado. El sistema muestra a continuación la información correspondiente al empleado.
  - Diagrama de secuencia (Figura 15):

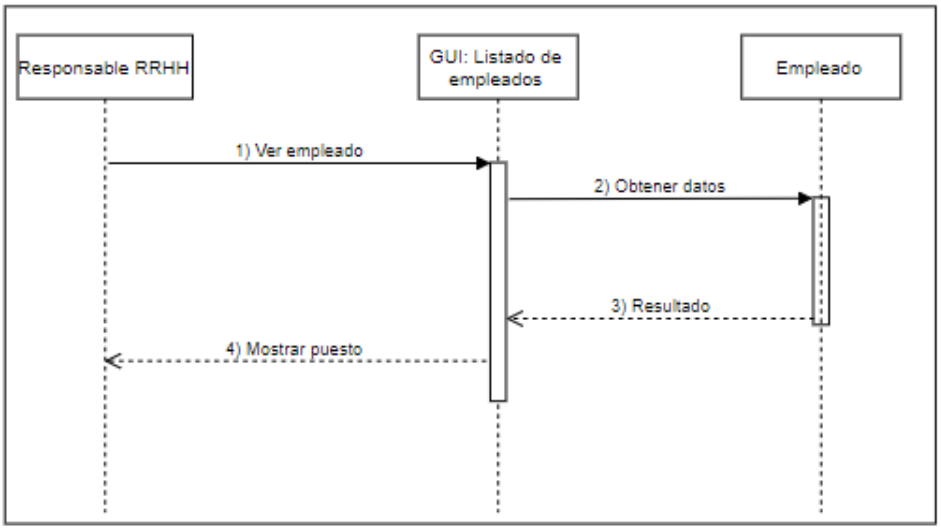


Figura 15. Diagrama de secuencia del caso de uso "Visualización de la información asociada al empleado".

  - Diagrama de actividades (Figura 16):

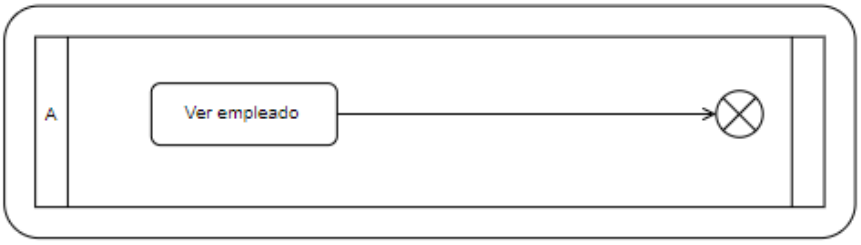


Figura 16. Diagrama de actividades del caso de uso "Visualización de la información asociada al empleado".

- Edición de la información asociada al empleado. El usuario deberá seleccionar un empleado de la lista de empleados para a continuación modificar cualquiera de sus campos y validar los cambios. El sistema notificará del resultado de la actualización.
    - Diagrama de secuencia (Figura 17):

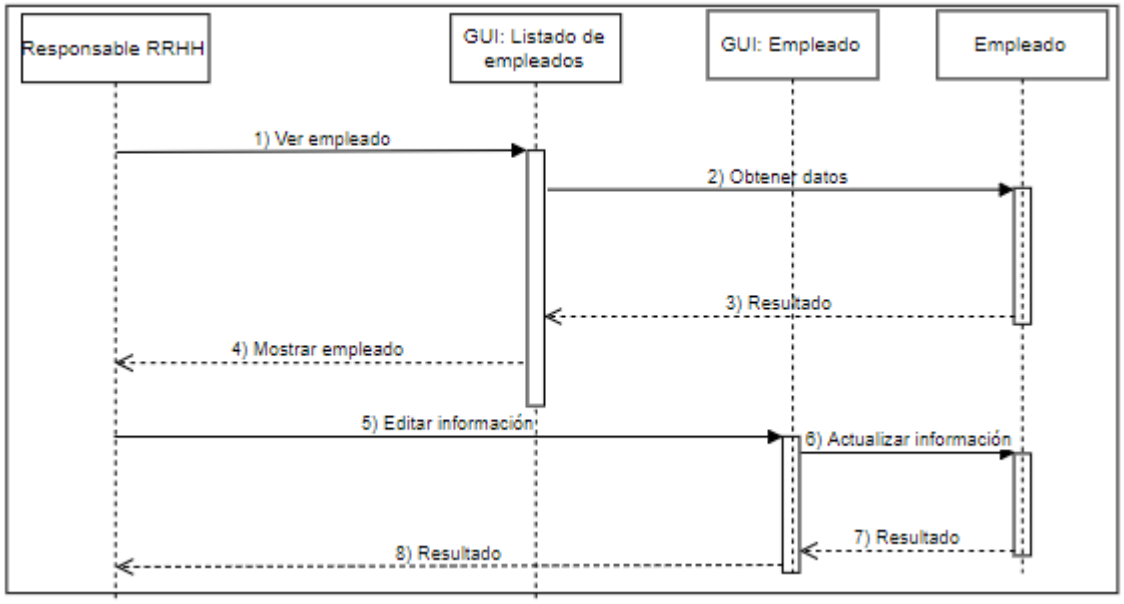


Figura 17. Diagrama de secuencias del caso de uso "Edición de la información asociada al empleado"

    - Diagrama de actividades (Figura 18)

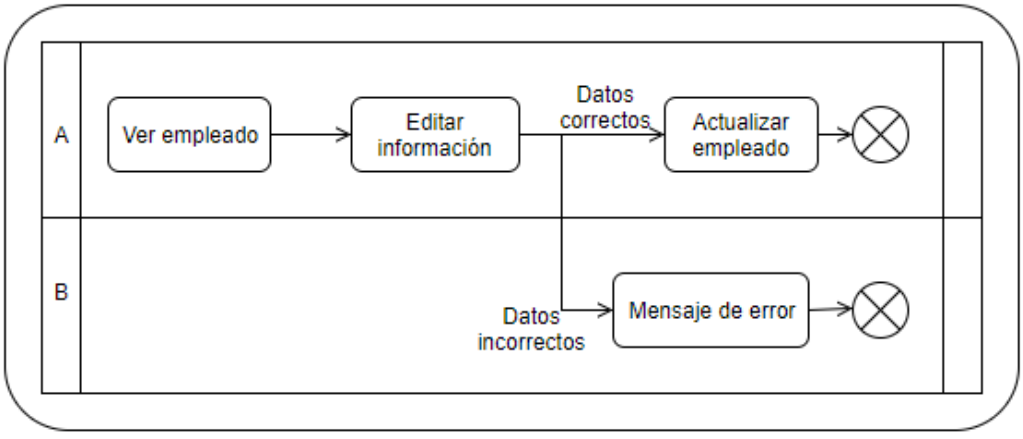


Figura 18. Diagrama de actividades del caso de uso "Edición de la información asociada al empleado".

3.3.4 Módulo vacaciones

A continuación se van a describir los casos de uso relacionados con el módulo vacaciones/ausencias. En total se dispone de seis casos de uso diferenciados, tal y como se muestra en la Figura 19. Como se ha explicado anteriormente, todos los casos de uso detallados a continuación requieren un acceso previo al módulo de vacaciones/ausencias, mediante autentificación previa introduciendo un nombre de usuario y una contraseña.

Figura 19. Casos de uso módulo vacaciones/ausencias

Así, los casos de uso mostrados en la figura son los siguientes:

- Creación de una nueva solicitud.
- Visualización de las solicitudes.
- Visualización de periodos laborales.
- Modificación de solicitudes.
- Modificación de periodos laborales.
- Eliminar solicitud.

A continuación se procede a detallar los diferentes casos de uso en orden, mostrando un diagrama de secuencia así como un diagrama de actividades para cada uno:

- Creación de una nueva solicitud. El usuario interactúa con la GUI del sistema y mediante un asistente crea un nuevo empleado con la información básica. Esta solicitud se guarda para la posterior modificación si se desea. Los diagramas resultan análogos a los de creaciones anteriores por lo tanto no se presentan.

- Visualización de solicitudes. El usuario interactúa con la GUI seleccionando una solicitud que haya sido creada. Los diagramas resultan análogos a los de visualizaciones anteriores por lo tanto no se presentan

- Visualización de periodos laborales. El usuario interactúa con la GUI seleccionando un periodo que haya sido creado. Este caso de uso es análogo del anterior pero sustituyendo solicitudes por periodos, por lo tanto no se presentan los diagramas.
- Modificación de solicitudes. El usuario deberá seleccionar una solicitud de la lista de solicitudes para a continuación modificar cualquiera de sus campos y validar los cambios. El sistema notificará del resultado de la actualización. Los diagramas resultan análogos a los de modificaciones anteriores por lo tanto no se presentan

- Modificación de periodos. El usuario deberá seleccionar un periodo de la lista de periodos para a continuación modificar cualquiera de sus campos y validar los cambios. El sistema notificará del resultado de la actualización. Este caso de uso es análogo del anterior pero sustituyendo solicitudes por periodos, por lo tanto no se presentan los diagramas.

4. ANÁLISIS DE PAQUETES

4.1 Introducción

Una vez descritos los casos de uso, se procede a identificar las entidades relevantes y sus relaciones. Se van a definir paquetes de análisis para organizar las diferentes entidades y funcionalidades del sistema en partes más manejables.

4.2 Identificación de paquetes de análisis

Figura 20. Paquetes de análisis del sistema

4.2.1 Gestión de incidencias

Este paquete de análisis engloba la funcionalidad necesaria para dejar constancia de las incidencias que hubiere en el sistema para su correcta identificación y posterior resolución.

4.2.2 Acceso a repositorio de persistencia de datos

El paquete de análisis “Acceso a repositorio de persistencia de datos” engloba la funcionalidad necesaria para asegurar la persistencia de las entidades del sistema que lo precisen. Este paquete de análisis no surge a partir de la fase de captura de requisitos, sino que proviene de las necesidades identificadas en el propio análisis del sistema, por lo que no contiene ningún caso de uso identificado.

4.2.3 Seguridad de los datos utilizados

Este paquete de análisis engloba la funcionalidad necesaria para asegurar la seguridad e integridad de los datos

5. REQUERIMIENTOS ESPECIALES

En esta sección se describen requerimientos especiales identificados durante la fase de análisis y que son importantes para el sistema. Normalmente, no están referidos a funcionalidad final de cara al usuario, sino que son restricciones o necesidades propias del sistema.

5.1 Persistencia

El sistema a desarrollar debe garantizar la persistencia de algunos objetos de los identificados en la fase de análisis. La definición y especificación de las clases que necesitarán de esta propiedad de persistencia se hará en fases posteriores al análisis. Asimismo, el medio sobre el cual se hará efectiva la capacidad persistente del sistema será concretado y definido en la fase del diseño del sistema.

En esta fase de análisis, se ha definido el paquete 'Acceso a repositorio de persistencia de datos', el cual contendrá la funcionalidad necesaria para garantizar dicha perspectiva.

5.2 Tolerancia a fallos

El sistema debe ser capaz de recuperarse de una acción no permitida, y volver a un estado estable y válido. El paquete de análisis "Gestión de incidencias" será el que contenga la funcionalidad necesaria para que el sistema sea tolerante a fallos.

5.3 Seguridad

En cualquier proyecto software es un requisito fundamental que el sistema garantice la seguridad de las comunicaciones entre la GUI y el repositorio de almacenamiento, así como la privacidad de los datos. En el caso de este trabajo, este requerimiento se va a llevar a cabo con el uso de un Framework de mapeo de objetos relacionales que garantiza la seguridad, integridad y privacidad de los datos, lo que está incluido en el paquete de análisis "Seguridad de los datos utilizados".

5.4 Disponibilidad del sistema

El sistema debe ser multiusuario, y además, debe acceder a un repositorio de información, por lo que se debe controlar el acceso de cada usuario a la información almacenada, evitando escrituras simultáneas en el repositorio de datos. También se debe garantizar la concurrencia en las interacciones de los usuarios con la aplicación, sin que empeore el rendimiento.

6. BIBLIOGRAFÍA

6.1 Referencias

[R1] I. Jacobson, G. Booch, J. Rambaugh. 2000. “El Proceso Unificado de Desarrollo de Software”. Pearson Education.

[R2] I. Jacobson, G. Booch, J. Rambaugh. 1999. “El lenguaje unificado de modelado. Manual de referencia”. Ed. Addison Wesley.

[R3] Martin Fowler. 1999. “UML Destilled”. Addison-Wesley 1999 (2nd Ed.).

6.2 Referencias web

[W1] http://www.uml.org

[W2] http://www.wikipedia.org

[W3] http://www.rational.com

[W4] http://www.cs.ualberta.ca/~pfiguero/soo/uml/

[W5] http://www.endalia.com

endalia®
HR Consulting & Software

DISEÑO

DESARROLLO DE SISTEMA DE GESTIÓN DE RECURSOS HUMANOS PARA PYMES

VERSIÓN 1.0
PUBLICADO EL 29/12/2017

HISTÓRICO DE REVISIONES

Fecha	Versión	Descripción	Autor
29/12/2017	1.0	Redacción inicial del documento	Marina Ariño Armengol

ÍNDICE

1. INTRODUCCIÓN

1.1 Propósito del documento

El objetivo del diseño es obtener, a partir del análisis, un punto de partida para actividades de implementación, capturando los requisitos, interfaces y clases a partir de las especificaciones de requisitos y análisis previos.

1.2 Alcance del documento

El alcance del documento comprende toda la fase de diseño del sistema de gestión de Recursos humanos para pymes.

1.3 Acrónimos

- ACID: Atomicity-Consistency-Isolation-Durability.
- API: Application Programming Interface.
- BCL: Base Class Library.
- CLI: Common Language Infrastructure.
- CLR: Common Language Runtime.
- CLS: Common Language Specification.
- CTS: Common Type System.
- DDL: Data Definition Language.
- DML: Data Manipulation Language.
- ECMA: European Computer Manufacturer Association.
- FTP: File Transfer Protocol.
- FTPS: File Transfer Protocol Segure.
- GDI: Graphics Device Interface.
- GUI: Graphics User Interface.
- HQL: Hibernate Query Language.
- HTTP: HyperText Transfer Protocol.
- HTTPS: HyperText Transfer Protocol Secure.
- IEC: International Electrotechnical Commission.
- IIS: Internet Information Services.
- ISO: International Organization for Standardization.
- JIT: Just-in-Time.
- MSIL: Microsoft Intermediate Language.
- NNTP: Network News Transfer Protocol.

- OBRM/ORM: Object-Relational Mapping.
- SGBD: Sistema Gestor de Bases de Datos.
- SMTP: Simple Mail Transfer Protocol.
- TCP/IP: Transmission Control Protocol / Internet Protocol.
- T-SQL: Transact – Structured Query Language.
- WSDL: Web Services Descriptor Language.
- XML: eXtensive Markup Language.

1.4 Definiciones

- Principio ACID: es el conjunto de propiedades de una base de datos que aseguran la realización de transacciones seguras. En concreto, ACID es un acrónimo de *Atomicity, Consistency, Isolation and Durability* (Indivisibilidad, Consistencia, Aislamiento y Durabilidad en castellano).
    - Indivisibilidad: es la propiedad que asegura que todas las tareas incluidas en la transacción han sido realizadas, o bien que ninguna de ellas lo ha sido.
    - Consistencia: es la propiedad que asegura que la base de datos está en un estado coherente antes del comienzo de la transacción, y que queda en otro estado coherente (sea el mismo u otro) después de la finalización de la transacción.
    - Aislamiento: es la propiedad que asegura que una operación externa a la transacción no puede acceder a un estado intermedio de los producidos durante la misma.
    - Durabilidad: es la propiedad que asegura que una vez realizada la transacción con éxito, ésta persistirá y no se podrá deshacer aunque falle el sistema.
- *Framework*: es una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado.
- Mapeo objeto-relacional (ORM): es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos.
- *Tooltip*: es una herramienta de ayuda que funciona al situar o pulsar con el ratón sobre algún elemento gráfico, mostrando una ayuda adicional para informar al usuario de la finalidad del elemento sobre el que se encuentra. Los *tooltip* son una variación de los globos de ayuda y es un complemento muy usado en programación, dado que proporcionan información adicional sin necesidad de que el usuario la solicite.

1.5 Referencias

Este documento contiene referencias a los siguientes documentos del trabajo, contenidos en la sección “Anexos”:

- Especificación de requisitos: documento que describe la especificación de requisitos realizada a partir de las necesidades del cliente.
- Análisis: documento que describe la arquitectura del sistema desarrollado, junto con un listado de casos de uso y su estructura de paquetes.

1.6 Resumen

En este documento se describe el proceso de diseño del sistema de gestión de Recursos humanos para pymes. Se compone de ocho apartados:

1. Introducción del documento, definición del propósito y alcance del mismo.
2. Se describe el proceso de diseño seguido para la confección de este documento.
3. Se describen las decisiones y restricciones iniciales del diseño, y se comparan y describen diferentes alternativas.
4. Descripción de la arquitectura del sistema tanto a nivel físico como a nivel de organización del mismo.
5. Detalle de las clases identificadas.
6. Detalle del diseño de la base de datos.
7. Prototipado de la interfaz de las diferentes pantallas con la que ha de interactuar el usuario del sistema.
8. Bibliografía y referencias Web utilizadas durante esta fase del proyecto.

2. DESCRIPCIÓN DEL PROCESO

Partiendo de las entidades, casos de uso y paquetes identificados en la fase de análisis y de los requisitos del sistema identificados en la especificación de requisitos, se procede a la descripción de la arquitectura del sistema, para su posterior implementación. A partir de los casos de uso, se identifican las necesidades de interacción entre el usuario y el sistema, y se define la estructura de la interfaz, así como los prototipos del mismo. A partir de las entidades se describen las tablas necesarias en base de datos, así como sus clases de acceso a datos.

3. CONSIDERACIONES INICIALES

3.1 Introducción

En esta sección se describen las primeras decisiones y especificaciones de diseño del sistema de gestión de Recursos humanos para pymes, que sirven como base para el diseño del resto del sistema. Asimismo, se describen las principales características de las distintas tecnologías utilizadas.

3.2 Especificaciones tecnológicas

El trabajo que nos ocupa se lleva a cabo en el marco de la empresa Endalia. Este hecho condiciona la tecnología a utilizar que, evidentemente, debe ser la usada en el resto de aplicaciones llevadas a cabo en dicha empresa, con el objeto de hacerlas compatibles y fácilmente integrables.

Es por esto que el sistema de gestión de Recursos humanos para pymes se va a desarrollar, en parte, en la plataforma .NET de Microsoft, con el framework ASP .NET Web API 2 para la API, usando el lenguaje de programación C# y el gestor de base de datos Microsoft SQL Server. Por otro lado, se va a desarrollar el front-end con Angular 5, tecnología de reciente incorporación en la empresa, y por lo tanto, sin experiencia.

Se analizaron y recopilaron los componentes y características que debían tener los diseños de la aplicación, entre ellos, debía ser un diseño responsive, con modales, tabs, árboles, datepickers, etc. Se estudiaron varias herramientas que se encuentran en el mercado teniendo en cuenta su cobertura en los componentes que se necesitan, características interesantes y la compatibilidad con las últimas versiones de Angular, Bootstrap y con los diferentes navegadores. Finalmente, se decidió adquirir Kendo de Telerik por su variedad de componentes, sus características y su soporte comunitario.

3.3 Especificaciones de diseño

A partir de los requisitos identificados en el documento de análisis de requisitos, podemos definir las siguientes características necesarias del sistema:

- Acceso a base de datos de forma transaccional, cumpliendo el principio ACID.
- Escalabilidad: el sistema debe soportar más carga de trabajo sin necesidad de modificar el software.
- Extensibilidad: el sistema debe soportar la adición de nuevos componentes y funcionalidades sin que ello afecte al resto de componentes.
- Usabilidad: el sistema debe poder ser manejado de forma intuitiva.
- Seguridad: el sistema debe ser fiable, tanto a nivel de autenticación, como de autorización, y debe mantener la privacidad de la información confidencial.
- Rendimiento: el sistema debe soportar un incremento en la carga de trabajo sin que ello repercuta notablemente en el usuario.

3.4 Plataforma .NET

Microsoft .NET es, de acuerdo con la definición de Microsoft, una plataforma que comprende servidores, clientes y servicios. Consiste en un conjunto de aplicaciones como Visual Studio .NET, los servicios .NET, etc. Esta plataforma es una implementación basada en estándares abiertos como SOAP, WSDL o C#. Desde el punto de vista del programador, el entorno .NET ofrece un solo entorno de desarrollo para todos los lenguajes que soporta (por ejemplo, Visual Basic, C#, C++, Visual J#, Fortran, Cobol…).

3.4.1 .NET Framework

El "*framework*" o marco de trabajo constituye la base de la plataforma .NET (Figura 1), y denota la infraestructura sobre el cual se reúnen un conjunto de lenguajes, herramientas y servicios que simplifican el desarrollo de aplicaciones en un entorno de ejecución distribuido.

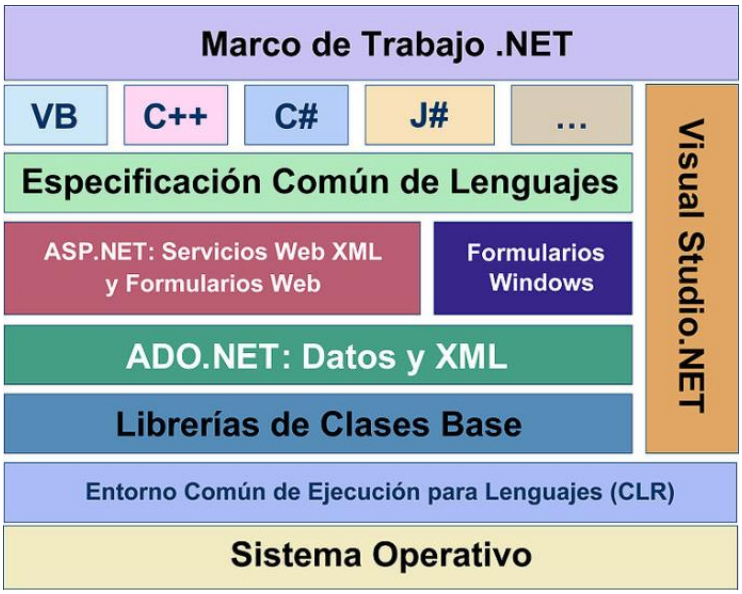


Figura 1. .NET Framework

Bajo el nombre .NET Framework, o Marco de Trabajo .NET, se encuentran reunidas una serie de normas, entre las cuales se encuentran:

- La norma que define las reglas que debe seguir un lenguaje de programación para ser considerado compatible con el marco de trabajo .NET (ECMA-335, ISO/IEC 23271). Por medio de esta norma se garantiza que todos los lenguajes desarrollados para la plataforma ofrezcan al programador un conjunto mínimo de funcionalidad y compatibilidad con todos los demás lenguajes de la plataforma.
- La norma que define el lenguaje C# (ECMA-334, ISO/IEC 23270). Éste es el lenguaje insignia del marco de trabajo .NET y pretende reunir las ventajas de lenguajes como C/C++ y Visual Basic en un solo lenguaje.
- La norma que define el conjunto de funciones que debe implementar la librería de clases base (BCL, siglas en inglés) (incluido en ECMA-335, ISO/IEC 23271). Tal vez el más importante de los componentes de la plataforma, esta norma define un conjunto funcional mínimo que debe implementarse para que el marco de trabajo sea soportado por un sistema operativo. Aunque Microsoft implementó esta norma para su sistema operativo Windows, la publicación de la norma abre la posibilidad de que sea implementada para cualquier otro sistema operativo existente o futuro, permitiendo que las aplicaciones corran sobre la plataforma, independientemente del sistema operativo para el cual hayan sido implementadas.

Los principales componentes del marco de trabajo son:

- El conjunto de lenguajes de programación.
- La Biblioteca de Clases Base o BCL.
- En entorno Común de Ejecución para Lenguajes o CLR.

Debido a la publicación de la norma para la infraestructura común de lenguajes (CLI por sus siglas en inglés), el desarrollo de lenguajes se facilita, por lo que el marco de trabajo .NET soporta ya más de 20 lenguajes de programación (C#, Visual Basic, C++, Perl, Python, Fortran…) y es posible desarrollar cualquiera de los tipos de aplicaciones soportados en la plataforma con cualquiera de ellos.

3.4.2 Common Language Runtime

El CLR (Figura 2) es el verdadero núcleo del Framework de .NET, entorno de ejecución en el que se cargan las aplicaciones desarrolladas en los distintos lenguajes, ampliando el conjunto de servicios del sistema operativo.

Figura 2. Common Language Runtime (CLR)

La herramienta de desarrollo compila el código fuente de cualquiera de los lenguajes soportados por .NET en un código intermedio (MSIL, *Microsoft Intermediate Language*), similar al *BYTECODE* de Java. Para generar dicho código, el compilador se basa en el *Common Language Specification* (CLS), que determina las reglas necesarias para crear ese código MSIL compatible con el CLR.

Para ejecutarse se necesita un segundo paso: un compilador JIT (*Just-in-Time*), que es el que genera el código máquina real que se ejecuta en la plataforma del cliente. De esta forma se consigue con .NET independencia de la plataforma hardware. La compilación JIT la realiza el CLR a medida que el programa invoca métodos. El código ejecutable obtenido se almacena en la memoria caché del ordenador, siendo recompilado de nuevo solo en el caso de producirse algún cambio en el código fuente.

3.4.3 Biblioteca de Clases Base (BCL)

La Biblioteca de Clases Base (*Base Class Library*, BCL) proporciona las clases básicas predefinidas que manejan la mayoría de las operaciones que se encuentran involucradas en el desarrollo de aplicaciones, incluyendo entre otras:

- Interacción con los dispositivos periféricos.
- Manejo de datos (ADO.NET).
- Administración de memoria.
- Cifrado de datos.
- Transmisión y recepción de datos por distintos medios (XML, TCP/IP).
- Administración de componentes Web que corren tanto en el servidor como en el cliente (ASP.NET).
- Manejo y administración de excepciones.
- Manejo del sistema de ventanas.
- Herramientas de despliegue de gráficos GDI.
- Herramientas de seguridad e integración con la seguridad del sistema operativo.
- Manejo de tipos de datos unificado.
- Interacción con otras aplicaciones.
- Manejo de cadenas de caracteres y expresiones regulares.
- Operaciones aritméticas.
- Manipulación de archivos de imágenes.
- Aleatoriedad.
- Generación de código.
- Manejo de idiomas.
- Auto descripción de código.
- Interacción con el API Win32 o Windows API.
- Compilación de código.

Esta funcionalidad se encuentra organizada por medio de espacios de nombre jerárquicos en lo que se denomina *Namespace*, como se puede apreciar en la siguiente figura (Figura 3):

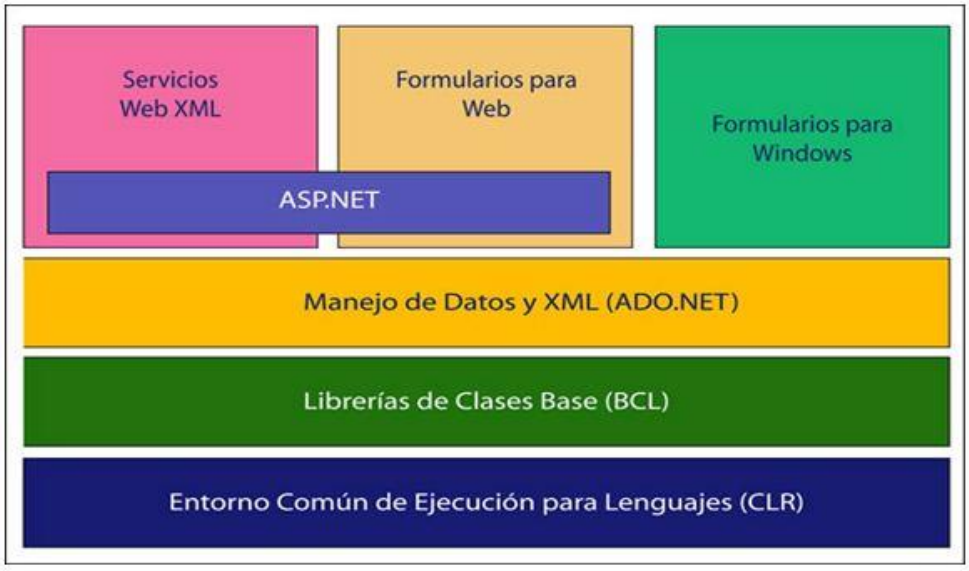


Figura 3. Esquema de distribución jerárquica de Namespaces

3.5 Angular 5

Angular es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

La biblioteca lee el HTML que contiene atributos de las etiquetas personalizadas adicionales, entonces obedece a las directivas de los atributos personalizados, y une las piezas de entrada o salida de la página a un modelo representado por las variables estándar de JavaScript. Los valores de las variables de JavaScript se pueden configurar manualmente, o ser recuperados de recursos JSON estáticos o dinámicos.

Angular es la evolución de AngularJS aunque incompatible con su predecesor.

3.5.1 Principales características

Angular permite desarrollar aplicaciones multiplataforma:

- **Aplicaciones web progresivas**: Permite utilizar las capacidades modernas de las aplicaciones sobre plataforma web para ofrecer experiencias similares en aplicaciones de escritorio. Permitiendo una instalación rápida, en pocos pasos y sin necesitar una conexión a internet.
- **Aplicaciones nativas**: Es posible construir aplicaciones móviles nativas combinando Angular con Ionic Framework, NativeScript y React Native.
- **Escritorio**: Permite crear aplicaciones instalables para escritorios de Mac, Windowsy Linux utilizando los mismos métodos de Angular utilizados para desarrollar sobre plataformas web, con la capacidad de acceder a las APIs nativas del Sistema Operativo.

Angular destaca por su velocidad y rendimiento, debido a lo siguiente:

- **Generación de código**: Angular convierte tus plantillas en código altamente optimizado para las máquinas virtuales de JavaScript de hoy en día, ofreciéndote todas las ventajas del código escrito a mano con la productividad de un framework.
- **Universal**: Ejecuta la primera vista de tu aplicación en node.js, .NET, PHP, y otros servidores para renderizado de forma casi instantánea obteniendo solo HTML y CSS. También abre posibilidades para la optimización del SEO del sitio.
- **División del código**: Las aplicaciones de Angular se cargan rápidamente gracias al nuevo enrutador de componentes. Éste ofrece una división automática de códigos para que los usuarios sólo carguen el código necesario para procesar la vista que solicitan.

Ha sido optimizado para aumentar la productividad de desarrollo mediante:

- **Plantillas**: Permite crear rápidamente vistas de interfaz de usuario con una sintaxis de plantilla simple y potente.
- **Angular CLI**: Las herramientas de línea de comandos permiten empezar a desarrollar rápidamente, añadir componentes y realizar test, así como previsualizar de forma instantánea la aplicación.
- **IDEs**: Obtén sugerencias de código inteligente, detección de errores y otros comentarios en la mayoría de los editores populares e IDEs.

3.5.2 Arquitectura

Angular es un framework para crear aplicaciones cliente en HTML y JavaScript o en un lenguaje como TypeScript que se compila en JavaScript. El framework consta de varias bibliotecas, algunas de ellas centrales y otras opcionales.

Para desarrollar una aplicación con Angular se componen plantillas HTML, se implementan los componentes para administrar estas plantillas, añadiendo la lógica en los servicios. Estos servicios y componentes se agrupan en módulos.

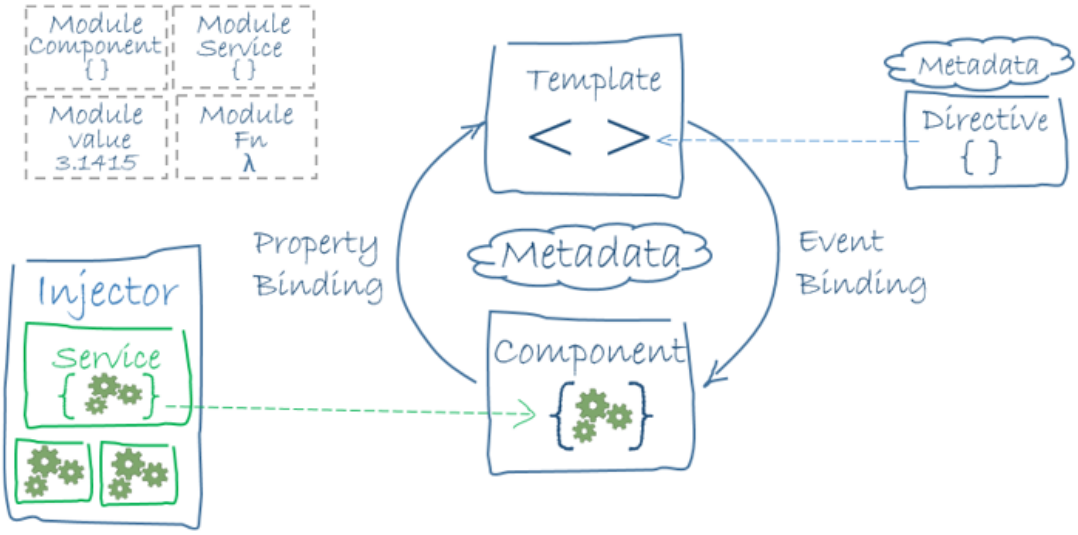


Figura 4. Esquema general arquitectura Angular

A continuación se van a detallar las distintas partes de la arquitectura (Figura 4). Las aplicaciones de Angular son modulares y este tiene su propio sistema de modularidad llamado *NgModules*. Cada aplicación Angular debe tener al menos un módulo raíz, aunque la mayoría de aplicaciones tienen muchos más. Por otro lado, posee una colección de módulos de JavaScript que se pueden ver como módulos de biblioteca. Los componentes controlan las vistas, en ellos se define la lógica de aplicación dentro de una clase. La clase interactúa con la vista a través de una API de propiedades y métodos. Cuando se define un componente, normalmente, se definirá la plantilla complementaria. Una plantilla es un formulario HTML en el que se le indica a Angular como renderizar el componente. Aunque en estas plantillas se suelen usar elementos típicos de HTML también se incluyen algunas directivas como ngFor, ngIf o etiquetas personalizadas que representan a componentes, estos se pueden mezclar sin problema. Los metadatos informan a Angular cómo procesar una clase, estos se pueden indicar mediante decoradores, para registrar un componente deberemos añadir el decorador *@Component* para que éste sea reconocido por otras partes de la aplicación.

Sin un framework, se debería introducir los valores de los datos en los controles HTML y convertir las respuestas de los usuarios en acciones y actualizaciones sobre los valores. Angular soporta enlace de datos, conocido como *data binding*, un mecanismo para coordinar la plantilla con el componente. Como se muestra en la figura 5 hay cuatro formas de sintaxis de data binding. Cada formulario tiene una dirección: al DOM, desde el DOM o en ambas direcciones.

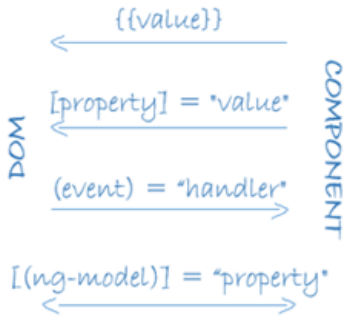


Figura 5. Data binding

Las plantillas de Angular son dinámicas. Cuando este renderiza, transforma el DOM de acuerdo con las instrucciones dadas por las directivas. Una directiva es una clase con un decorador *@Directive.* El servicio es un categoría amplia que abarca cualquier valor, función o característica que se necesite, normalmente es una clase con un propósito estrecho y bien definido. Por último, Angular usa la inyección de dependencia para proporcionar nuevos componentes con los servicios que necesitan. Angular puede decir qué servicios necesita un componente mirando los tipos de sus parámetros de constructor.

3.6 SQL Server

Es un SGBD para bases de datos relacionales desarrollado por Microsoft. Sus principales características son:

- Soporte para transacciones (bajo el principio ACID).
- Escalabilidad.
- Estabilidad.
- Seguridad.
- Soporta procedimientos almacenados.
- Entorno gráfico que permite ejecutar comandos DDL y DML.
- T-SQL como lenguaje de consultas nativo.

Una base de datos de SQL Server es una colección de tablas con columnas de un tipo definido, más otros objetivos como restricciones, vistas, procedimientos almacenados o índices. El espacio de almacenamiento está dividido en páginas de 8KB, que es la unidad básica de entrada-salida para una operación de SQL Server. Las filas de cada tabla se almacenan físicamente en fichero o bien en un montículo (*heap*) o bien en un árbol-B. Los índices (que son estructuras para acelerar el acceso a datos en las consultas) definidos son almacenados siempre en árboles-B. Hay dos tipos de índices en SQL Server:

- Índices agregados, en los que se almacenan los datos de la fila indexada en las hojas del árbol.
- Índices no agregados, que en las hojas del árbol-B almacenan una referencia a la hoja del índice agregado correspondiente, o bien una referencia a la página correspondiente.

Sólo puede haber un índice agregado por tabla, y éste habitualmente es la clave primaria de ésta.

3.7 NHibernate

NHibernate es un Framework de Mapeo objeto-relacional (ORM de sus siglas en inglés) para la plataforma .NET y el lenguaje C#. Facilita el mapeo de atributos entre una base de datos relacional y el modelo de objetos de una aplicación mediante archivos declarativos (XML). NHibernate es software libre y soporta los motores de base de datos más habituales del mercado: MySQL, PostgreSQL, Oracle, MS SQL Server, etc.

NHibernate basa su configuración en un fichero XML en el que se le indica información sobre los recursos a mapear así como diversas opciones de configuración:

```
<?xml version="1.0" encoding="utf-8"?>
<hibernate-configuration xmlns="urn:nhibernate-configuration-2.2">
    <session-factory>
        <property name="dialect">NHibernate.Dialect.MsSql2005Dialect</property>
        <property name="connection.provider">
NHibernate.Connection.DriverConnectionProvider</property>
        <property name="connection.isolation">ReadUncommitted</property>
        <property name="connection.driver_class"> NHibernate.Driver.SqlClientDriver</property>
        <property name="connection.connection_string_name">conn</property>
        <property name="default_schema">dbo</property>
        <property name="adonet.batch_size">10</property>
        <property name="show_sql">false</property>
        <property name="proxyfactory.factory_class">
NHibernate.ByteCode.Castle.ProxyFactoryFactory, NHibernate.ByteCode.Castle</property>
        <mapping assembly="IntegraDBAccess"/>
    </session-factory>
</hibernate-configuration>
```

La utilización de NHibernate nos permite la creación de dos *Namespaces* diferentes para gestionar el acceso a los datos. El primero, llamado *Domain*, contendrá los ficheros de mapeo de NHibernate así como los ficheros de representación de las diferentes entidades mapeadas en forma de objetos. El segundo *Namespace*, llamado *DataAccess*, contiene las clases donde se encuentran los métodos para acceder a los datos.

El mapeo de una entidad guardada en la base de datos se realiza mediante un fichero de mapeo (en formato XML), dónde se definen los atributos y las relaciones de dicha entidad, tal y como se muestra en el siguiente ejemplo:

```
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2" assembly="InterfacesDBAccess"
namespace="InterfacesDBAccess.Domain">
    <class name="Orh_Hol_HolidaysPersonMaxDay" table="Orh_Hol_HolidaysPersonMaxDays"  dynamic-update="true"
>
    <composite-id name="ID">
        <key-property type="int" name="EmpId" column="fk_HolidayPersonMaxDaysPersonId" />
        <key-property type="int" name="PeriodId" column="fk_HolidayPersonMaxDaysPeriodId" />
        <key-property type="int" name="TypeId" column="fk_HolidayPersonMaxDaysTypeId" />
    </composite-id>
    <property name="HolidaysPersonMaxDays" column="HolidaysPersonMaxDays" not-null="false" />
    <property name="HolidaysPersonExternSynchroDate" column="HolidaysPersonExternSynchroDate" not-
null="false" />

      <many-to-one name="Employee" class="Orh_Emp_Employee" insert="false" update="false" >
        <column name="fk_HolidayPersonMaxDaysPersonId" />
      </many-to-one>

      <many-to-one name="Period" class="Orh_Hol_HolidaysPeriod" insert="false" update="false" >
        <column name="fk_HolidayPersonMaxDaysPeriodId" />
      </many-to-one>

      <many-to-one name="Type" class="Gen_AdministrationItem" insert="false" update="false" >
        <column name="fk_HolidayPersonMaxDaysTypeId" />
      </many-to-one>

   </class>
</hibernate-mapping>
```

Por otro lado, NHibernate ofrece su propio lenguaje de consultas, el *Hibernate Query Language* (HQL). Permite al programador escribir consultas similares a SQL directamente contra los objetos de datos mapeados. Permite modificar los objetos y añadir restricciones sobre éstos. Además, se caracterizan por ser consultas independientes de la base de datos utilizada.

3.8 IIS

Internet Information Services, desarrollado por Microsoft, es un conjunto de servicios sobre TCP/IP que permiten a la máquina donde está alojado proporcionar funcionalidad de servidor de aplicaciones. Actualmente soporta los siguientes protocolos: HTTP, HTTPS, FTP, FTPS, SMTP y NNTP.

Como servidor web, permite definir una serie de directorios virtuales, donde se encuentra cada una de las aplicaciones indexadas por él, resolviendo las peticiones de páginas estáticas, y gestionando las necesidades de código dinámico de las mismas. Asimismo, proporciona mecanismos de seguridad y autenticación de usuarios, donde permite una total integración con el *Active Directory* de Microsoft Windows.

4. DISEÑO DE LA ARQUITECTURA

4.1 Introducción

En esta sección se describe la estructura del sistema, tanto desde el punto de visto físico como lógico, detallando las decisiones que se han tomado en dichos ámbitos, así como las restricciones que estaban impuestas por la organización en la que se desarrollaba el sistema de gestión de recursos humanos.

4.2 Estructura general del sistema

La estructura general del sistema está definida por la plataforma de otros servicios web de Endalia. Por lo tanto, la estructura del proyecto debe adaptarse a la de estos servicios. A continuación se detallan las restricciones que deben mantenerse, referente a la estructura:

- Servidor IIS: servidor de aplicaciones donde se aloja la plataforma de servicios web. Resuelve peticiones (http en este caso, pero también de otros tipos) y las asigna a su aplicación correspondiente, en función de los permisos previamente establecidos.
- Base de datos: repositorio de datos de la aplicación, implementado desde el sistema gestor de bases de datos SQL Server 2012.

4.3 Estructura de capas del sistema

Una posible forma de organizar un sistema de un tamaño considerable es agrupando funcionalidades que comparten la misma naturaleza, funcionalidad y estructura en capas. De esta manera, se consigue que cambios en algún componente (por ejemplo, presentación) no afecten al resto de elementos del sistema; lo que proporciona una óptima escalabilidad y ayuda a mejorar el rendimiento.

El sistema está basado en la siguiente arquitectura multicapa:

- Capa de presentación: formada por los componentes que se ejecutan en el cliente. En nuestro caso es el navegador web desde el que se accede a la plataforma y el código TypeScript que se ejecuta en las páginas de la aplicación. En este sistema se corresponde con los componentes angular y su código subyacente. En esta capa se ha concentrado la mayor parte del trabajo desarrollado.
- Capa de lógica de negocio. Contiene los objetos (DTOs) que representan los datos almacenados en el repositorio de datos, así como la lógica necesaria para procesarlos.
- Capa de acceso a datos: Contiene objetos que automatizan el acceso a datos, realizada a través del *Framework* NHibernate.
- Capa de datos. Contiene los sistemas de información de la aplicación, habitualmente una base de datos. En nuestro sistema se corresponde con los archivos de recursos de la aplicación y la base de datos, que contiene la información sobre los empleados, la organización y sus solicitudes de vacaciones y ausencias. Así cómo funciones y procedimientos para acceder a ellos.

4.4 Estructura de subsistemas

Los subsistemas son un medio para organizar el modelo en partes más pequeñas y manejables. Una de las opciones para realizar esta actividad se basa en la identificación de subsistemas de diseño a partir de los paquetes definidos en la fase de análisis. La correspondencia no siempre debe ser uno a uno, ya que intervienen ciertos condicionantes que la limitan, pero sí constituye un punto de partida para iniciar la identificación.

En los siguientes apartados se enumeran y explican los subsistemas de diseño que forman la aplicación.

4.4.1 Subsistema de estructura organizativa

En este subsistema se realizan las funcionalidades propias para la gestión de la estructura organizativa de la empresa junto con la visualización de la información de los puestos que la forman, su creación y modificación.

4.4.2 Subsistema de personas

En este subsistema se realizan las funcionalidades propias de los empleados de la organización junto con la creación, modificación y visualización de la información de estos.

4.4.3 Subsistema de periodos de solicitud

Subsistema en el cual se desarrolla las funcionalidades propias de los periodos de las solicitudes, creación, modificación y eliminación de estas.

4.4.4 Subsistema de solicitudes

En este subsistema se realizan las funcionalidades propias de las solicitudes de vacaciones y ausencias, como la creación de solicitudes, su modificación y eliminación.

4.4.5 Subsistema de gestión de incidencias

Subsistema en el cual se desarrolla la funcionalidad identificada en el paquete “Gestión de incidencias”, la relacionada con el control de los eventos y errores producidos por el sistema.

4.4.6 Subsistema de acceso a base de datos

En este subsistema se desarrollan las funcionalidades necesarias para gestionar el acceso a base de datos de manera transparente y eficiente.

4.4.7 Subsistema de internacionalización de entidades visibles

En este subsistema se desarrollan las funcionalidades necesarias para que la aplicación pueda adaptar sus contenidos de manera automática a la cultura en la que se ejecute.

4.4.8 Subsistema de seguridad de datos utilizados

En este subsistema se desarrollan las funcionalidades necesarias para que la aplicación sea segura y consiga una integridad completa de los datos.

5. CLASES DEL SISTEMA

5.1 Introducción

En este apartado se detallan las clases del sistema de gestión de Recursos Humanos para pymes. Estas clases se dividen en cuatro grupos:

- Clases de interfaz: son las encargadas de crear la GUI y gestionar las interacciones del usuario con el sistema.
- Clases de acceso a datos: son las encargadas de gestionar la persistencia de los datos del sistema y la interacción con la base de datos.
- Clases de transferencia de datos: son objetos que transportan datos entre la api y la vista.
- Clases de mapeo objeto-relacional de datos: son las encargadas de crear una representación de los datos persistentes en la base de datos en forma de objetos utilizables desde el código de la aplicación.
- Clases auxiliares.

5.2 Clases de interfaz

En esta sección se van a analizar las clases de interfaz de la aplicación. Como ya se ha comentado, el front-end se desarrolla en Angular. La estructura básica de la página web que contiene cabecera y menús está contenida en la página "*master*". Esto permite la creación del resto de las páginas del sistema como un contenido de los master, y por lo tanto la reutilización de la estructura base, permitiendo la realización de modificaciones en la estructura básica de forma rápida y sencilla.

A continuación se comentan las distintas clases de interfaz agrupándolas por subsistemas.

5.2.1 Clases del subsistema de personas

- employee-list.component: componente que contiene el listado de empleados de la organización.
- employee-form.component: componente que contiene la información detallada de los empleados.
- add-employee-form.component: componente que permite añadir nuevos empleados en el sistema.

5.2.2 Clases del subsistema de estructura organizativa

- job-list.component: componente que contiene el árbol de la estructura jerárquica de puestos junto con un listado de estos.
- job-form.component: componente que contiene la información detallada de los puestos.
- new-job-form.component: componente que permite añadir nuevos puestos en el sistema.
- job-employee-form.component: componente que permite asignar empleados activos a un puesto de la estructura organizativa.
- change-status-form.component: componente que permite cambiar un puesto histórico a activo colocándolo en la estructura organizativa.
- org-chart.component: componente que presenta el organigrama de la organización.

5.2.3 Clases del subsistema de periodos

- period-list.component: componente que contiene un listado de periodos de solicitud de vacaciones/ausencias
- period-form.component: componente que contiene información detallada sobre los periodos definidos.
- add-period-list.component: componente que permite definir un nuevo periodo de solicitud.

5.2.4 Clases del subsistema de solicitudes

- request-list.component: componente que contiene un listado de solicitudes realizadas por los empleados de la organización.
- request-form.component: componente que contiene información detallada sobre las solicitudes creadas en el sistema y permite añadir nuevas solicitudes.

5.3 Clases de acceso a datos

En esta sección se van a analizar las clases de acceso a datos del sistema de gestión de Recursos Humanos para pymes. A continuación se detallarán las clases más destacadas:

- Orh_Emp_EmployeeDao.cs: es la clase de acceso a los empleados.
- Orh_Emp_EmployeePhotoDao: clase de acceso a las fotos de los empleados.
- Orh_Job_JobDao.cs: es la clase de acceso a datos de los puestos.
- R_Employee_JobDao: es la clase de acceso a datos que representa la relación entre los empleados y los puestos.
- Gen_AdministrationItemDao: es la clase de acceso a datos de las parametrizaciones.
- Orh_Hol_HolidaysPeriodDao: es la clase de acceso a datos de los periodos de solicitud.
- Orh_Hol_HolidaysRequestDao: es la clase de acceso a datos de las solicitudes.

5.4 Clases de transferencia de datos

En esta sección se van a analizar las clases de transferencia de datos del sistema de gestión de Recursos Humanos para pymes. A continuación se detallarán las clases más destacadas:

- DTO_AdministrationItem.cs: clase que contiene las propiedades que son necesarias que se transfieran del objeto de base de datos Gen_AdministrationItem.
- DTO_EmpJob.cs: clase que contiene las propiedades que son necesarias que se transfieran del objeto de base de datos R_EmployeeJob.
- DTO_Job: clase que contiene las propiedades que son necesarias que se transfieran del objeto de base de datos Orh_Job_Jobs.
- DTO_PersonalData: clase que contiene las propiedades que son necesarias para transferir la información personal de los empleados del objeto de base de datos Orh_Emp_Employees.
- DTO_ProfessionalData: clase que contiene las propiedades que son necesarias para transferir la información profesional de los empleados del objeto de base de datos Orh_Emp_Employees.
- DTO_HolidaysPeriod: clase que contiene las propiedades que son necesarias para transferir la información necesaria de los periodos del objeto de base de datos Orh_Hol_HolidaysPeriod.
- DTO_HolidaysRequest: clase que contiene las propiedades que son necesarias para transferir la información necesaria de las solicitudes del objeto de base de datos Orh_Hol_HolidaysRequest.

5.5 Clases de mapeo objeto-relacional de datos

En esta sección se van a analizar las clases de mapeo objeto-relacional del sistema de gestión de Recursos Humanos para pymes. En la siguiente figura (Figura 6), se puede observar el listado completo, aunque únicamente se detallarán las más destacadas:

- Orh_Emp_Employee.cs y Orh_Emp_Employee.hbm.xml: son las clases que representan los empleados.
- Orh_Emp_EmployeePhoto.cs y Orh_Emp_EmployeePhoto.hbm.xml: son las clases que representan las fotos de los empleados.
- Orh_Job_Job.cs y Orh_Job_Job.hbm.xml: son las clases que representan los puestos de la organización.
- R_Employee_Job.cs y R_Employee_Job.hbm.xml: son las clases que representan las relaciones entre los empleados y los puestos.
- Gen_AdministrationItem.cs y Gen_AdministrationItem.hbm.xml: son las clases que representan las parametrizaciones del sistema.
- Orh_Hol_HolidaysPeriod.cs y Orh_Hol_HolidaysPeriod.hbm.xml: son las clases que representan los periodos de solicitud de vacaciones o ausencias.
- Orh_Hol_HolidaysRequest.cs y Orh_Hol_HolidaysRequest.hbm.xml: son las clases que representan las solicitudes realizadas de vacaciones y ausencias.

5.6 Clases auxiliares

Las clases auxiliares son un tipo de clases con diferentes utilidades sobre los datos del sistema. En esta tipología se engloban desde las clases genéricas de los *Frameworks* utilizados hasta clases para encriptación de datos que, por motivos de seguridad y confidencialidad, no se detallarán.

6. DISEÑO DE LA BASE DE DATOS

6.1 Introducción

En este apartado se presenta una vista global del diseño de la base de datos, resultante de las entidades identificadas en el análisis, adecuando al repositorio de datos que utiliza la aplicación. Como ya se ha comentado anteriormente, el repositorio de datos a utilizar es una base de datos relacional gestionada desde el SGBD MSSQL Server 2012.

También se describen las tablas de la base de datos junto con sus atributos y las relaciones existentes entre ellas.

6.2 Diseño general de la base de datos

Debido al tamaño de la base de datos, se va a separar en subconjuntos de menor tamaño, ya que no sería posible representarlo de otro modo en este formato de documento. Se van a agrupar en dos subconjuntos: el primero de ellos engloba la estructura organizativa y personas y la relación entre ambos y el segundo corresponde a los periodos y solicitudes de vacaciones.

6.2.1 Estructura organizativa y personas

Todos los empleados del sistema tienen una entrada en la tabla *Orh_Emp_Employees,* ya que es la que contiene los datos de acceso al este. Por otro lado, todos los puestos están registrados en la tabla *Orh_Job_Jobs.* Ambos, se relacionan a través de la tabla *R_Employee_Job*, la cual recoge todos los empleados que están asignados en algún puesto junto a qué puesto se trata.

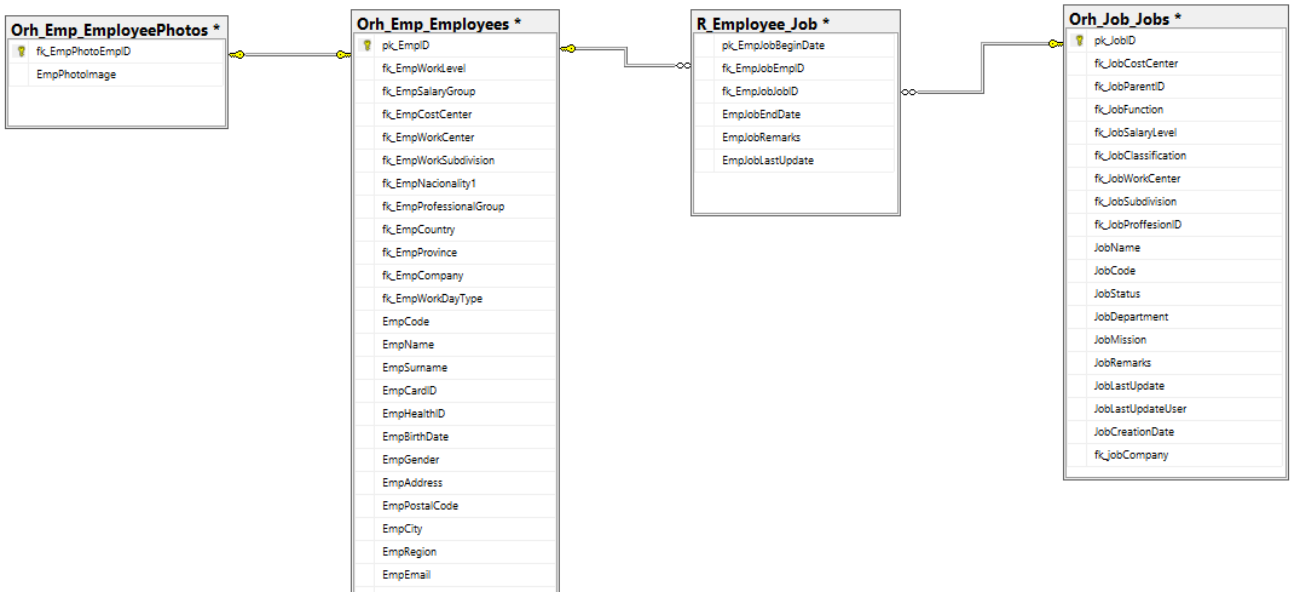


Figura 6. Esquema general de los datos implicados en la estructura organizativa y personas.

6.2.2 Vacaciones/Ausencias

En este módulo la clase principal corresponde con las solicitudes de vacaciones y ausencias (*Orh_Hol_HolidayRequests)*, estas solicitudes están relacionadas con los periodos (*Orh_Hol_HolidaysPeriods)* ya que una solicitud se tiene que realizar en un periodo. Las solicitudes están relacionadas con la tabla (*Orh_Emp_Employees)* debido al solicitante y ambas se relacionan con la tabla (*Orh_Hol_HolidaysPersonMaxDays)* que indica el máximo de días que puede solicitar un empleado concreto.

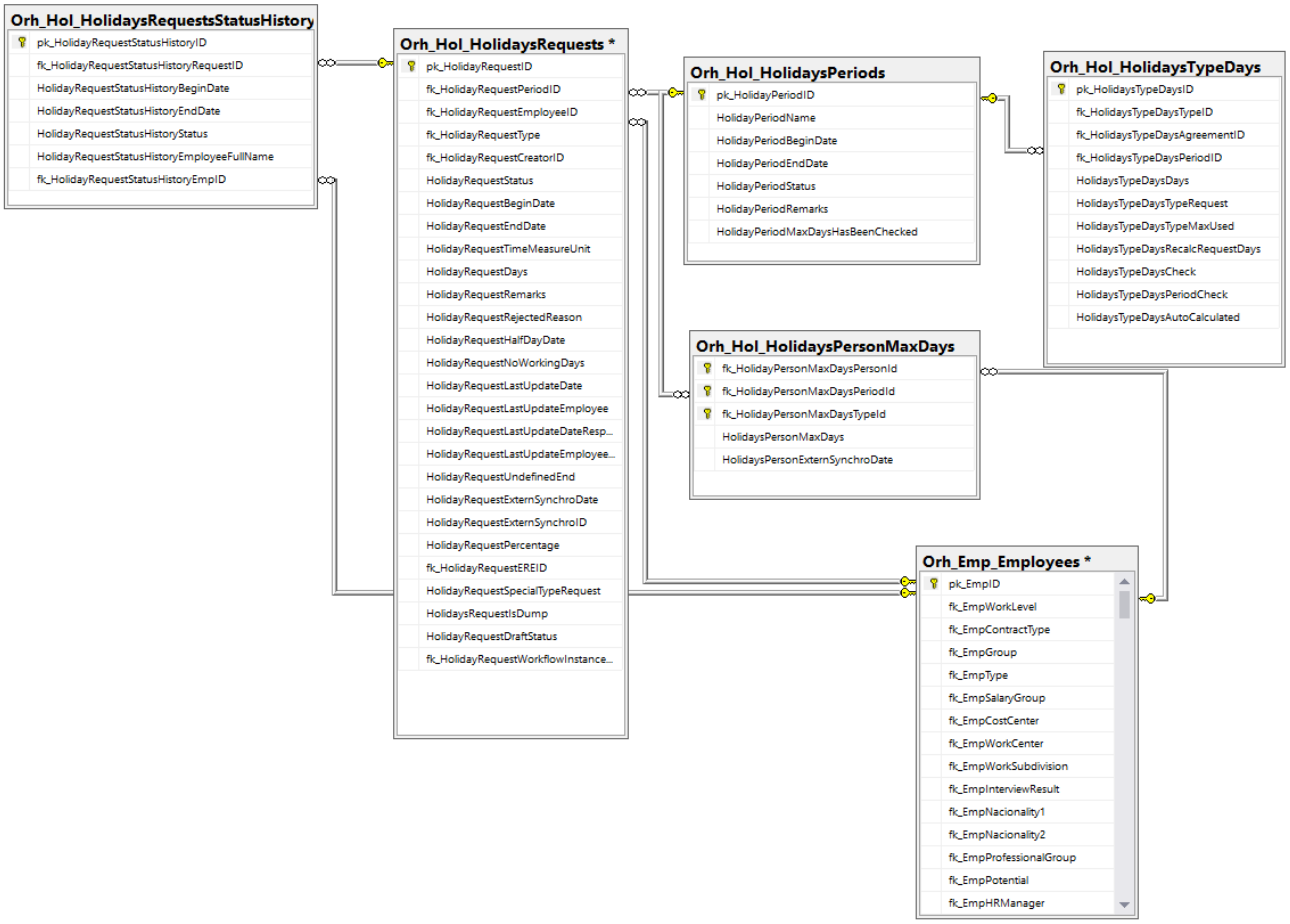


Figura 7. Esquema general de los datos implicados en el módulo de vacaciones

7. PROTOTIPADO DE LA INTERFAZ

7.1 Introducción

En este apartado se muestran los prototipos de interfaz de pantalla diseñados para el sistema, teniendo en cuenta el interfaz de la web corporativa de la organización a la que va destinado este proyecto realizado dentro de Endalia.

Los objetivos y decisiones tomadas en este punto son:

- La resolución mínima para la que se diseñará la aplicación es 1024x768.
- Primar la claridad y la usabilidad por encima de otros aspectos, proporcionando un entorno claro e intuitivo para los usuarios.
- Evitar la aparición de barras de desplazamiento horizontal, y limitar dentro de lo posible el uso de la barra de desplazamiento vertical, ocupando a poder ser únicamente la pantalla visible.

7.2 Interfaz básico

A continuación se muestra la base del diseño del interfaz gráfico, enmarcando las diferentes secciones que lo componen (Figura 9).

Los aspectos más significativos son los siguientes:

- Cabecera: debe ocupar la parte superior de la pantalla.
- Menú: se trata de un menú desplegable, situado en el lateral izquierdo.
- Contenido: la parte central de la página.

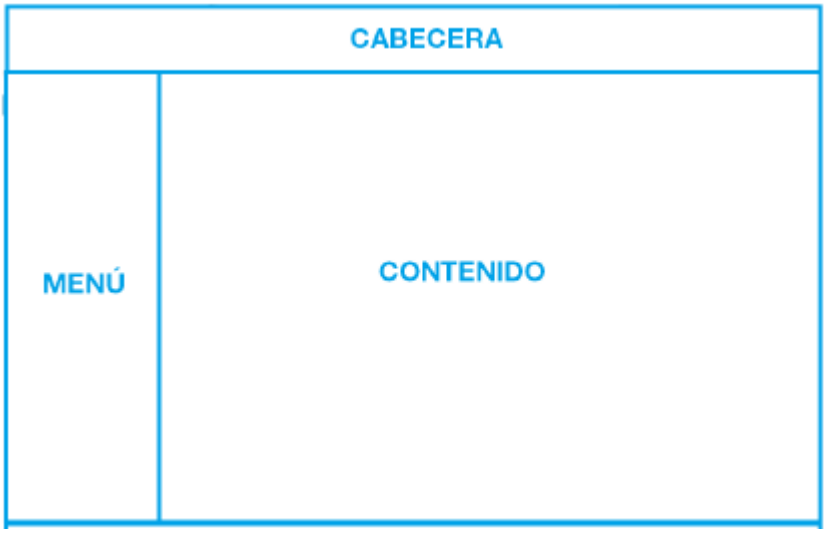


Figura 8. Prototipo de interfaz básico

7.3 Interfaz del contenido de la página de estructura organizativa

A continuación, se muestra el prototipo de la página de estructura organizativa, que contiene un árbol con la estructura jerárquica de puestos y un listado de estos. El prototipo de personas sería un listado análogo con información de los empleados en el que se eliminaría el árbol.

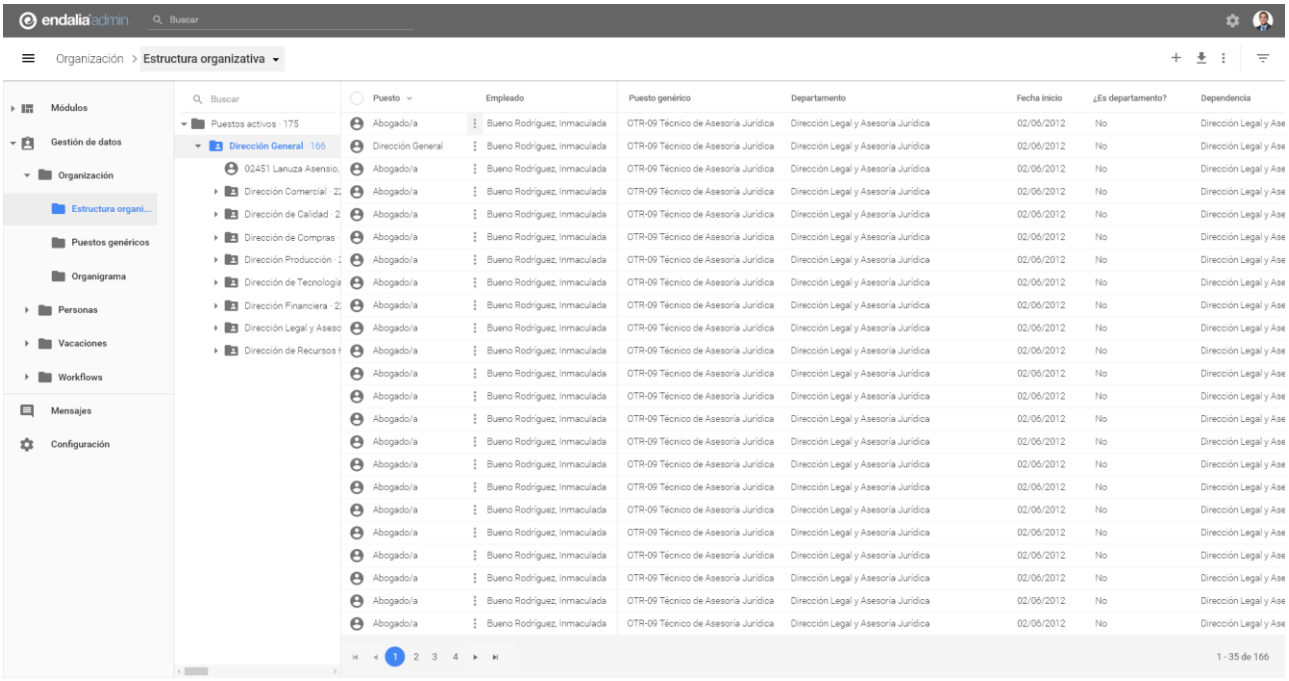


Figura 9. Prototipo de la página estructura organizativa

7.4 Interfaz del contenido de la página ficha del empleado

A continuación, se muestra el prototipo de la página de la ficha del empleado. La ficha del puesto tendría un diseño análogo a esta.

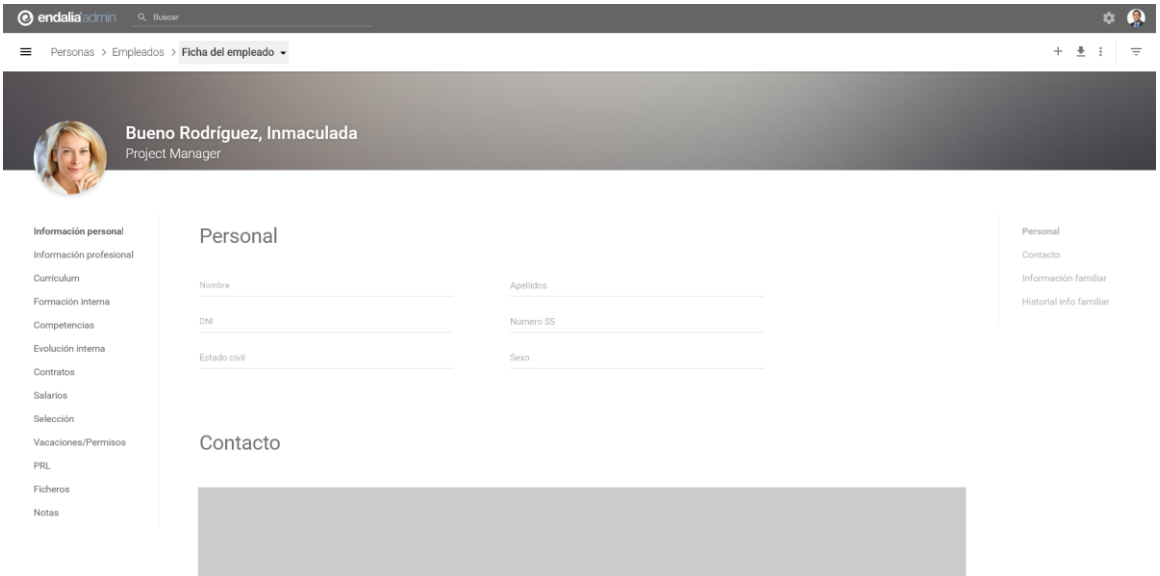


Figura 10. Prototipo de la página ficha del empleado

8. BIBLIOGRAFÍA

[W1] http://www.microsoft.com/net

[W2] http://www.wikipedia.org

[W3] http://www.uml.org

[W4] http://webdocs.cs.ualberta.ca/~pfiguero/soo/uml

[W5] http://www.rational.com

[W6] http://www.webstyleguide.com/index.html

[W7] http://www.endalia.com

[W8] https://balsamiq.com/

endalia®
HR Consulting & Software

IMPLEMENTACIÓN

DESARROLLO DE UN SISTEMA DE GESTIÓN DE RECURSOS HUMANOS PARA PYMES

VERSIÓN 1.0
PUBLICADO EL 29/12/2017

HISTÓRICO DE REVISIONES

Fecha	Versión	Descripción	Autor
29/12/2017	1.0	Redacción inicial del documento	Marina Ariño Armengol

ÍNDICE

1. INTRODUCCIÓN

1.1 Propósito del documento

Este documento presenta la fase de implementación del trabajo de desarrollo del sistema de gestión de Recursos Humanos para pymes. Partiendo del trabajo realizado en las fases de análisis y diseño se desarrolla una versión funcional de los módulos de organización, personas y vacaciones/ausencias.

1.2 Alcance del documento

Este documento describe la fase de implementación del sistema de gestión de Recursos Humanos para pymes, que corresponde a una de las últimas fases del desarrollo del trabajo.

1.3 Acrónimos

- DDL: Data Definition Language.
- DML: Data Manipulation Language.
- DTO: Data Transfer Object.
- FTP: File Transfer Protocol.
- GUI: Graphical User Interface.
- HQL: Hibernate Query Language.
- HTML: HyperText Markup Language.
- HTTP: HyperText Transfer Protocol.
- HTTPS: HyperText Transfer Protocol Secure.
- IIS: Internet Information Services.
- SDK: Software Development Kit.
- SGBD: Sistema Gestor de Base de Datos.
- SMTP: Simple Mail Transfer Protocol.
- SQL: Structured Query Language.
- T-SQL: Transact-Structured Query Language.
- XML: eXtensible Markup Language.

1.4 Definiciones

- Archivo de recursos: archivo en el que se almacenan datos que se corresponden con cierta información que maneja el sistema, pero que no dependen específicamente de las clases que contienen lógica.
- Control de versiones: gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo.
- Hoja de estilo: lenguaje formal usado para definir la presentación de un documento descrito en HTML o XML.
- Transacción: una transacción en un Sistema de Gestión de Bases de Datos es un conjunto de órdenes que se ejecutan formando una unidad de trabajo, es decir, en forma indivisible o atómica.
- SGBD transaccional: SGBD capaz de mantener la integridad de los datos, haciendo que las transacciones no puedan finalizar en un estado intermedio. Cuando por alguna causa el sistema debe cancelar la transacción, deshace las órdenes ejecutadas hasta dejar la base de datos en su estado inicial (llamado punto de integridad), como si la orden de la transacción nunca se hubiese realizado.

1.5 Referencias

Este documento contiene referencias a los siguientes documentos del trabajo, contenidos en la sección “Anexos”:

- Especificación de requisitos: documento que describe la especificación de requisitos realizada a partir de las necesidades del cliente.
- Análisis del sistema: documento que describe la arquitectura del sistema desarrollado, junto con un listado de casos de uso y su estructura de paquetes.
- Diseño del sistema: documento que describe el modelo por el cual se va a realizar la implementación del sistema.

1.6 Resumen

Este documento describe el proceso de implementación de un sistema de gestión de Recursos Humanos para pymes. Se compone de ocho apartados:

1. Introducción del documento, definición del propósito y alcance del mismo.
2. Se describe el proceso de implementación seguido.
3. Se describen las tecnologías, herramientas y lenguajes empleados durante la implementación del sistema.
4. Se describe el proceso seguido para la implementación de la internacionalización de la aplicación.
5. Se describe el proceso seguido para la gestión de incidencias ocurridas en el sistema.
6. Detalle de las consideraciones necesarias para la implementación del acceso a datos.
7. Se describe el proceso seguido para la implementación del interfaz de usuario.
8. Se describe el proceso seguido para la implementación de las notificaciones por correo electrónico.
9. Se describe el proceso seguido para la implementación de la generación de informes.
10. Bibliografía y referencias web utilizadas para la realización de este documento.

2. DESCRIPCIÓN DEL PROCESO

A partir del trabajo realizado en la fase de diseño, se procede a realizar el proceso de implementación del sistema. Dicho proceso permitirá obtener un sistema totalmente funcional y perfectamente integrado en el sistema de Endalia, que resuelva las necesidades del cliente establecidas mediante los requisitos funcionales del sistema así como siguiendo las pautas establecidas en la fase de análisis del trabajo. Durante la realización de esta fase se realizarán pequeñas pruebas para comprobar el correcto avance del desarrollo, acompañadas de un conjunto de pruebas finales realizadas cuando la implementación se complete. Estas últimas pruebas son las documentadas en el anexo correspondiente.

Dentro del proceso de implementación propiamente dicho podemos identificar dos fases:

- Implementación de la base de datos: creación de las tablas y las relaciones necesarias para el nuevo sistema en la base de datos, el conjunto de clases de acceso a datos correspondientes y sus respectivos mapeos a objetos.
- Implementación de los subsistemas identificados en el diseño, adecuándolos a la interfaz definida previamente.

En este trabajo se ha realizado únicamente la segunda de estas fases, ya que el modelo relacional de tablas ya había sido creado previamente por la entidad para un proyecto similar. El trabajo realizado en la primera de las fases ha consistido únicamente en la validación de dicho modelo relacional para asegurar su validez en el sistema de gestión de Recursos Humanos para pymes.

3. TECNOLOGÍAS, HERRAMIENTAS Y LENGUAJES

Para el desarrollo del presente proyecto se han utilizado las siguientes tecnologías, lenguajes y herramientas:

- Microsoft .NET Framework. Plataforma de desarrollo descrito en el apartado 3.4 del Documento de Diseño.
- Microsoft SQL Server 2016. SGBD utilizado para la gestión de la base de datos del sistema y descrito en el apartado 3.5 del Documento de Diseño.
- C#. Lenguaje de programación de propósito general nativo de la plataforma .NET.
- TypeScript es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, que esencialmente añade tipado estático y objetos basados en clases.
- Microsoft Visual Studio. Entorno de programación y depuración de código de la plataforma .NET.
- Microsoft SQL Management Studio 2012. Herramienta gráfica que permite realizar tareas de mantenimiento de la base de datos sobre SQL Server 2012.
- Microsoft Team Foundation Server. Herramienta que permite, entre otras funcionalidades, la gestión del control de versiones de un proyecto sobre Visual Studio.
- IIS. Servidor de aplicaciones de Microsoft que permite gestionar servidores HTTP, HTTPS, FTP o SMTP, entre otros. Se describe en el Documento de Diseño, en el apartado 3.7.
- T-SQL. Lenguaje de acceso a datos basado en SQL. Está descrito en el apartado 6.2.1 del presente documento.
- NHibernate. Framework de mapeo objeto-relacional que facilita la representación de entidades relacionales de base de datos en objetos accesibles desde la aplicación desarrollada.
- Log4Net. Herramienta para ayudar en la generación de ficheros de registro.
- Navegadores: Microsoft Internet Explorer (versiones superiores a la 8.0), Mozilla Firefox y Google Chrome.

4. IMPLEMENTACIÓN DE LA GESTIÓN DE INCIDENCIAS

El sistema de gestión de incidencias, o Log de la aplicación, se encarga de registrar secuencialmente los eventos del sistema en un fichero, guardando información acerca del tipo de evento, de cuándo ocurrió y qué lo causó. Dicho fichero puede posteriormente ser consultado, auditado y analizado para conocer el uso, funcionamiento y posibles problemas del sistema.

La siguiente información es registrada siempre, independientemente del resto del contenido del registro de log:

- Fecha y hora de registro del evento.
- Sistema gestor del log (clase que provoca la llamada).
- Etiqueta o tag con el tipo de mensaje según la siguiente clasificación:
    - DEBUG: mensaje informativo para depuración de la ejecución del sistema.
    - INFO: mensaje informativo sobre algún suceso acaecido en el sistema.
    - WARM: mensaje de aviso sobre intento de alguna ejecución peligrosa para el sistema.
    - ERROR: mensaje acerca de una ejecución fallida pero controlada del sistema.
    - FATAL: mensaje de interrupción total de la ejecución del sistema.

La siguiente información variará dependiendo del tipo de registro:

- Mensaje de error generado en caso de error de ejecución (por ejemplo en caso de fallo de ejecución de una instrucción SQL escribiría el mensaje devuelto por SQL Server).
- Cualquier mensaje informativo que necesite ser registrado por el sistema en un momento dado.

Para registrar en el fichero de Log las interacciones del usuario con la aplicación, se emplea la librería *log4net*, pero es necesario que en cada evento de la aplicación se escriba explícitamente en él, por ejemplo:

```
Log.Write("DEBUG: Index.GridRequestList_RowCommand command=" + e.CommandName);

Log.Write("DEBUG: Employee.Remove EmployeeID=" + EmpID.ToString());

Log.Write("ERROR: Employee.Remove EmployeeID=" + EmpID.ToString() + " " + e.ToString());
```

5. IMPLEMENTACIÓN DEL ACCESO A BASE DE DATOS

5.1 Introducción

Como se ha especificado en el documento de diseño, el Sistema Gestor de Base de Datos utilizado ha sido Microsoft SQL Server, al que se accede a través del *Framework* NHibernate.

En este apartado se van a describir las principales características e implicaciones del uso conjunto de ambos sistemas en la implementación propiamente dicha del sistema de gestión de Recursos Humanos para pymes.

5.2 SQL Server

Es un sistema de gestión de bases de datos relacionales (SGBD) basado en lenguaje SQL, con las características principales:

- Soporte de transacciones.
- Gran estabilidad.
- Seguridad.
- Escalabilidad.
- Soporta procedimientos almacenados.
- Potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información.
- Permite administrar información de otros servidores de datos.
- No es multiplataforma, ya que sólo está disponible en sistemas operativos de Microsoft.

5.2.1 T-SQL

Transact-SQL o T-SQL es una extensión a SQL con las siguientes características:

- Lenguaje de control de flujo. Instrucciones para permitir la ejecución de bloques condicionados e iterativos-

  ```
  IF-THEN-ELSE, CASE-WHEN, WHILE, ...
  ```

- Variables locales.

  ```
  DECLARE @EmployeeID INT
  ```

- Funciones nativas para procesado de cadenas, gestión de fechas, funciones matemáticas, etcétera.

  ```
  REPLACE(@foo, '-', '')
  ```

- Mejoras en sentencias *DELETE* y *UPDATE*, para permitir uniones en la cláusula *FROM*.

5.2.2 Transacciones

Una transacción es un conjunto de operaciones que debe ejecutarse como una sola unidad, según el principio ACID expuesto en el documento de Diseño.

En SQL Server, las sentencias SQL son tratadas como transacciones, y si ocurre cualquier tipo de incidencia durante su ejecución, el sistema es capaz de volver al estado anterior a su ejecución.

En ocasiones se requiere ejecutar un conjunto de sentencias SQL de forma transaccional, para lo que el lenguaje proporciona las sentencias de gestión de transacciones:

- BEGIN TRAN. Indica el comienzo de una transacción.
- ROLLBACK TRAN. Deshace todos los cambios realizados por la transacción activa.
- COMMIT TRAN. Una vez que las operaciones de la transacción se han completado con éxito, se indica a la base de datos que vuelva a un estado consistente.

```
DECLARE @Error INT
BEGIN TRANSACTION
INSERT INTO
NombreTabla
(
        Campo1,
        Campo2
)
VALUES
(
        @Valor1,
        @Valor2
)
SET @Error = @@ERROR
IF @Error != 0 GOTO ERROR_HANDLER

SELECT SCOPE_IDENTITY() AS 'Identity'
SET @Error = @@ERROR
IF @Error != 0 GOTO ERROR_HANDLER

COMMIT TRANSACTION
RETURN SCOPE_IDENTITY()
ERROR_HANDLER:
IF @@TRANCOUNT != 0 ROLLBACK TRANSACTION
RETURN @Error
```

5.2.3 Procedimientos almacenados

Un procedimiento almacenado (*Stored Procedure*) de Microsoft SQL Server es un conjunto de instrucciones T-SQL que residen físicamente en la base de datos. Las principales características de los procedimientos almacenados son las siguientes:

- Ejecución directamente en el motor de base de datos que suele estar en un servidor separado, evitando tráfico entre el SGBD y la aplicación cliente.
- Posibilidad de almacenamiento del plan de ejecución por parte del SGBD, lo que repercute positivamente en la eficiencia de la consulta.
- Encapsula la lógica de negocio, de tal modo que permiten que varios programas cliente puedan usarla, reduciendo el coste de mantenimiento de los mismos.
- Versatilidad. Al estar desarrollados en T-SQL, permiten realizar funciones del mismo nivel de complejidad que las que se pueden realizar en el código de la aplicación.

```
CREATE PROCEDURE (NombreProcedimiento)
(
	@parametro1 INT,
	@parametro2 INT
)
AS SELECT
	SUM(EnvPollLevelValue) AS IndicatorSUM
FROM
	NombreTabla1
WHERE
	<condición booleana>
```

5.2.4 Desencadenadores

Los desencadenadores (o *triggers*) son, al igual que los procedimientos almacenados, un conjunto de sentencias T-SQL almacenadas en base de datos, pero con la particularidad de ser ejecutadas automáticamente cuando un usuario realiza una acción en la tabla de la base de datos que lleva asociado el desencadenador.

Su principal utilidad es el control de actualizaciones y borrados en cascada, que no pueden ser controlados por el SGBD debido a ciclos múltiples.

Se pueden crear desencadenadores para los siguientes tipos de instrucciones:

- *INSERT*: Inserción de información en la base de datos.
- *UPDATE*: Actualización de información de la base de datos.
- *DELETE*: Eliminación de información de la base de datos.

A continuación se muestra un ejemplo de un *trigger* que controla que al eliminar una entrada de la tabla 1 se elimine una entrada asociada a esa tabla, en la tabla 2:

```
CREATE TRIGGER [NombreTrigger] ON [NombreTabla1] FOR DELETE AS DELETE FROM NombreTabla2
WHERE <condición booleana>
```

5.3 Acceso a datos

Las clases de acceso a datos están asociadas a cada una de las tablas de la base de datos, y contienen diversos métodos que pueden ser de actualización, consulta, eliminación o inserción de datos.

En este trabajo se han reutilizado algunas clases de acceso a datos ya implementadas anteriormente en el software de Endalia, pero realizando modificaciones en los métodos utilizados para el módulo de gestión de viajes y gastos para adaptarlos a las páginas implementadas.

5.3.1 Mapeo objeto-relacional de las entidades en base de datos

Gracias al uso del *Framework* NHibernate, se ha realizado un mapeo de las entidades presentes en la base de datos en forma de objetos accesibles desde el código de la aplicación. Así, cada entidad tiene su fichero de configuración (en formato XML) en el que se detallan sus diferentes atributos y relaciones, así como un fichero (en formato .cs) que constituye la representación del objeto en código, y que permite al desarrollador acceder a la entidad desde cualquier parte del proyecto.

5.3.2 Clases de acceso a datos

Además de las clases de representación de objetos, se han creado unas clases de acceso a datos con métodos comunes que permiten traer una gran cantidad de información de la base de datos. Los métodos incluidos en estas clases permiten obtener datos a medida, según las necesidades de las páginas desarrolladas.

Por otro lado, estas clases ofrecen una capa extra en la arquitectura software de la aplicación, permitiendo separar la aplicación de la propia gestión de la información en la base de datos. Esto se consigue en parte con el lenguaje de consultas de NHibernate, el *Hibernate Query Language* (HQL).

6. IMPLEMENTACIÓN DE LA INTERFAZ DE USUARIO

6.1 Introducción

El diseño definitivo de la interfaz viene definido por los esquemas de diseño de pantalla especificados en el documento de diseño.

A continuación, se especifica la manera en que se han utilizado los diferentes elementos que componen la interfaz finalmente diseñada, y se muestran capturas de pantalla del mismo.

6.2 Elementos de interfaz

6.2.1 Iconos e imágenes

La utilización de imágenes e iconos ha sido destinada siempre a facilitar al usuario la utilidad y funcionamiento de los diferentes elementos y partes del sistema, acompañándolos siempre preferiblemente de una etiqueta textual que especifica la funcionalidad o el destino del elemento representado por la imagen, llamada *tooltip*.

Los iconos de acceso adecuadamente escogidos facilitan a los usuarios encontrar las diferentes secciones del sistema con mayor rapidez.

6.2.2 Tablas

Se han utilizado tablas siempre que ha sido necesario para mostrar listados de datos. El texto de la cabecera destaca sobre el resto haciéndolo en negrita.

6.2.3 Editores de fechas

Se han utilizado, siempre que ha sido posible, editores de fechas que permiten una navegación gráfica a través de diferentes fechas mediante la visualización de un calendario.

6.2.4 Colores y fuentes

Los colores a utilizar vienen determinados por los colores corporativos de la organización a la que va destinado el proyecto del sistema de gestión de Recursos Humanos para pymes; del mismo modo que ocurre con las fuentes y estilos.

6.2.5 Estilos

Todos los estilos CSS del sistema deben estar registrados en archivos CSS u hojas de estilo, facilitando al máximo las futuras modificaciones de estilo y permitiendo su adaptación a otras organizaciones. En esta hoja de estilos también deben estar incluidos los colores y fuentes comentados en el punto anterior.

6.3 Pantallas del sistema

A continuación se presenta el diseño definitivo de las principales interfaces del sistema; ya que por el tamaño del mismo, entrar en detalle en cada una de las pantallas existentes sería excesivo para este documento.

6.3.1 Página “Estructura organizativa”

En la figura a continuación (Figura 1) se muestra una captura de pantalla de la sección “Estructura organizativa”, donde el usuario puede ver un listado de puestos junto con la estructura jerárquica.

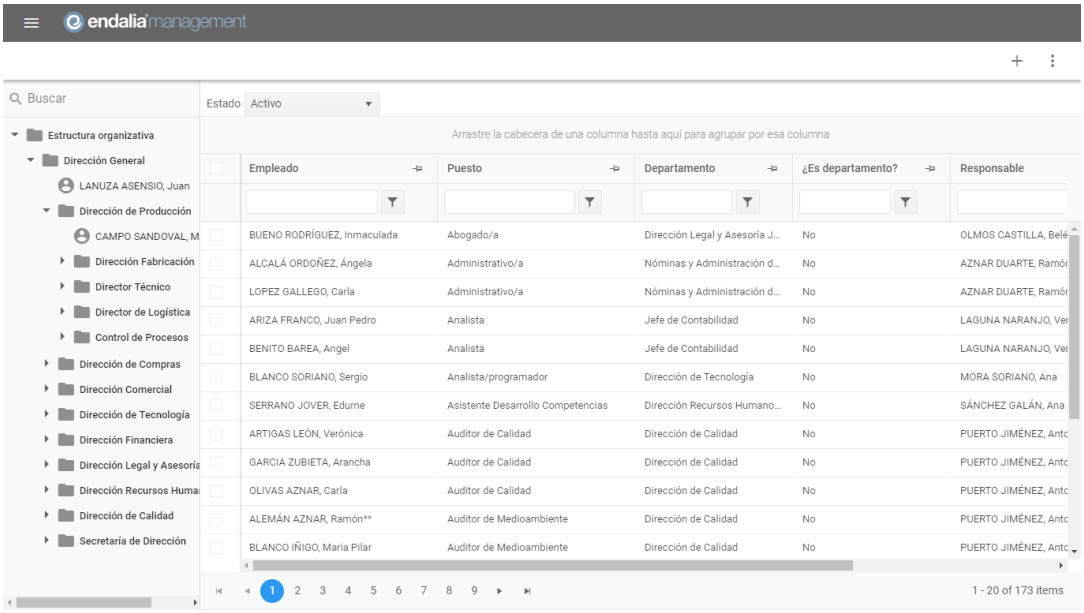


Figura 1. Página de Estructura organizativa.

La página ofrece un pequeño número de filtros que el usuario podrá utilizar para seleccionar el subconjunto de puestos que quiera ver. Estos puestos se pueden ver o editar gracias a la navegación superior derecha. Además en esta barra de navegación se sitúa el botón para abrir un asistente para la creación de un puesto, tal y como se muestra a continuación:

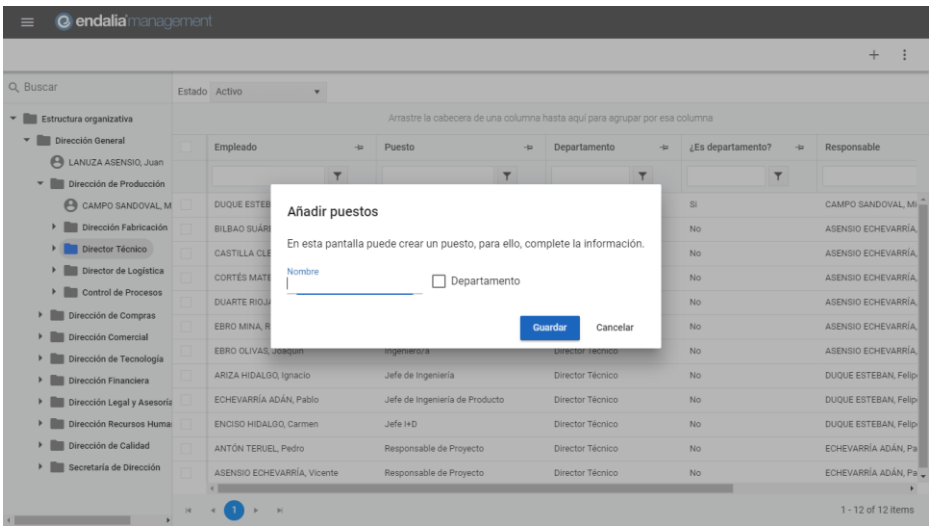


Figura 2. Asistente de creación de un puesto.

Los puestos creados se pueden visualizar en detalle (Figura 3) mediante el menú de navegación superior derecho y pulsando doble clic en la fila correspondiente.

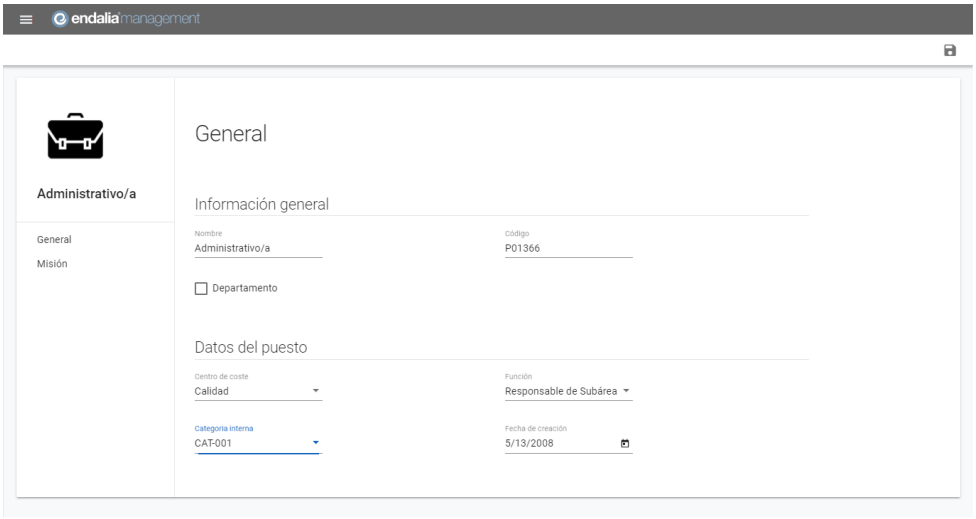


Figura 3. Detalle del puesto

Por último, se permite con el menú de la barra de navegación asignar empleados a un puesto, tal y como se muestra a continuación:

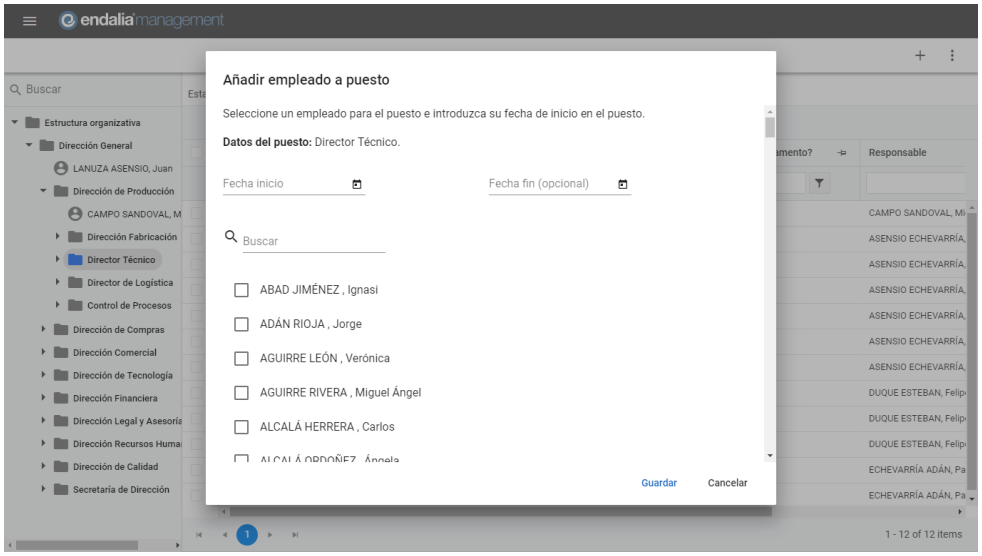


Figura 4. Asistente asignación empleado a puesto

6.3.2 Página “Empleados”

En la figura a continuación (Figura 5) se muestra una captura de pantalla de la sección “Empleados”.

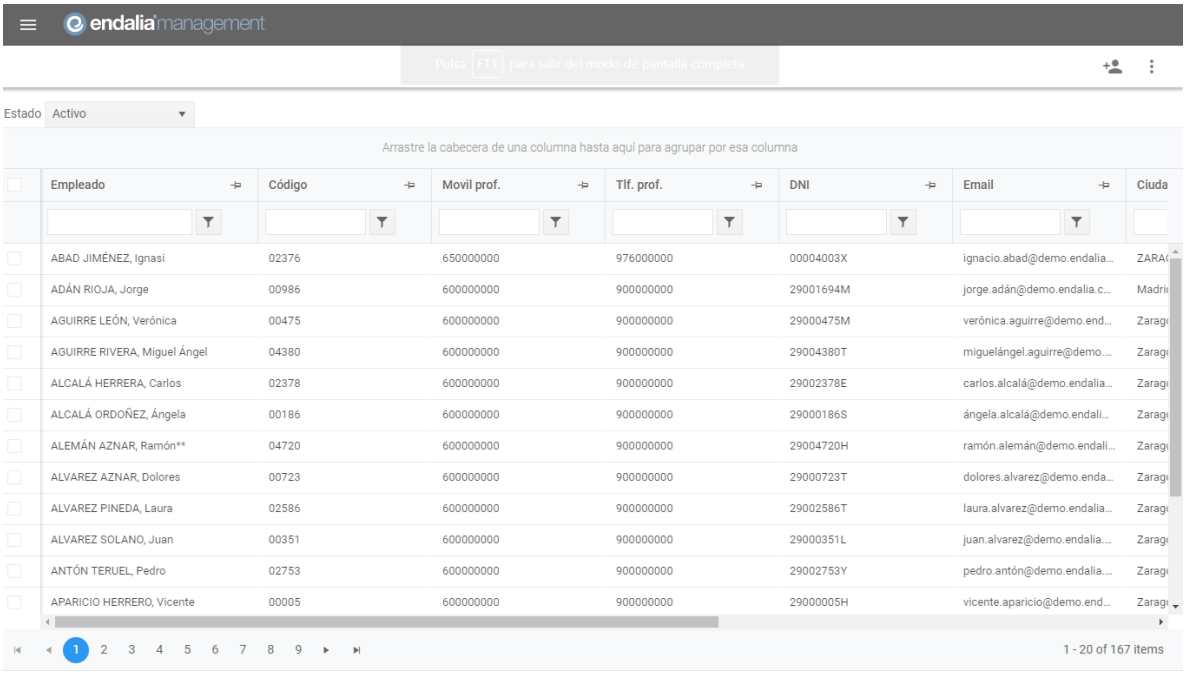


Figura 5. Página de Empleados

Se puede visualizar la información asociada (Figura 6) al empleado con el menú de navegación superior derecho o haciendo doble clic sobre la fila. Por otro lado, mediante el menú situado en la barra de navegación se puede abrir el asistente para creación de empleados (Figura 7), con el que se podrán añadir al sistema varios empleados de manera sencilla.

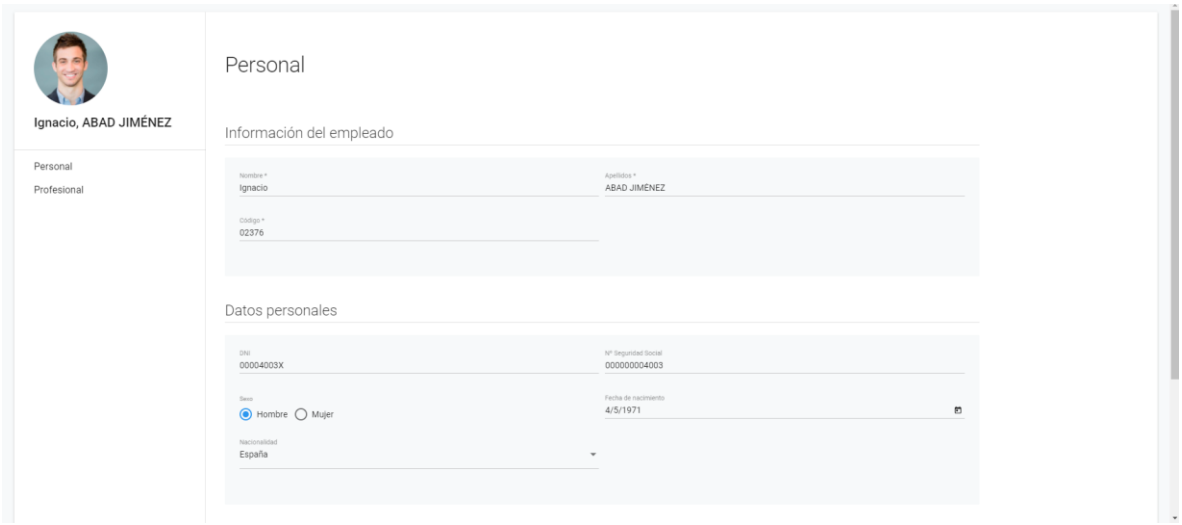


Figura 6. Información asociada al empleado

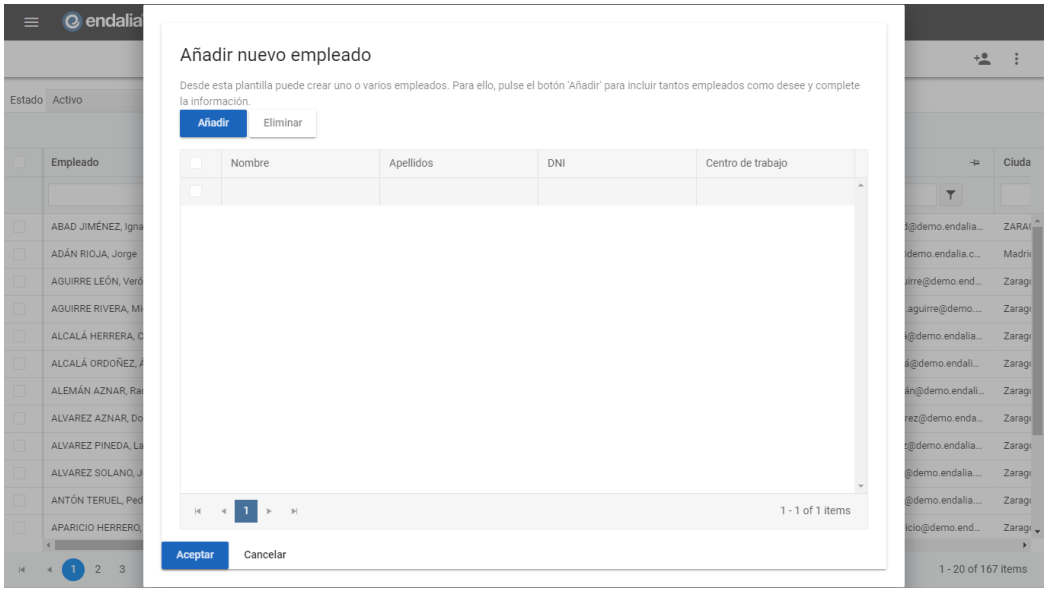


Figura 7. Asistente creación empleados

6.3.3 Página "Periodos"

En la figura a continuación (Figura 8) se muestra una captura de pantalla de la sección "Periodos", donde el usuario puede ver un listado de periodos junto con un árbol de activos e históricos.

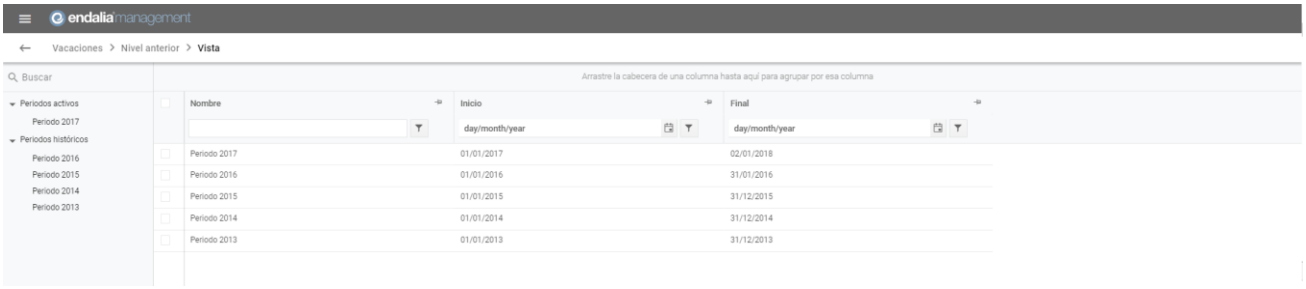


Figura 8. Página de periodos

Se puede visualizar la información asociada (Figura 9) al periodo con el menú de navegación superior derecho o haciendo doble clic sobre la fila. Junto con la información general se encuentran dos tablas, la primera presentan los convenios asociados a ese periodo junto con las solicitudes que pueden hacer para cada ausencia y la segunda muestra los empleados que tienen una restricción de máximo de días distinta a la establecida en el convenio. Por otro lado, mediante el menú situado en la barra de navegación se puede abrir el asistente para creación de periodos (Figura 10), con el que se podrán añadir al sistema varios periodos de manera sencilla.

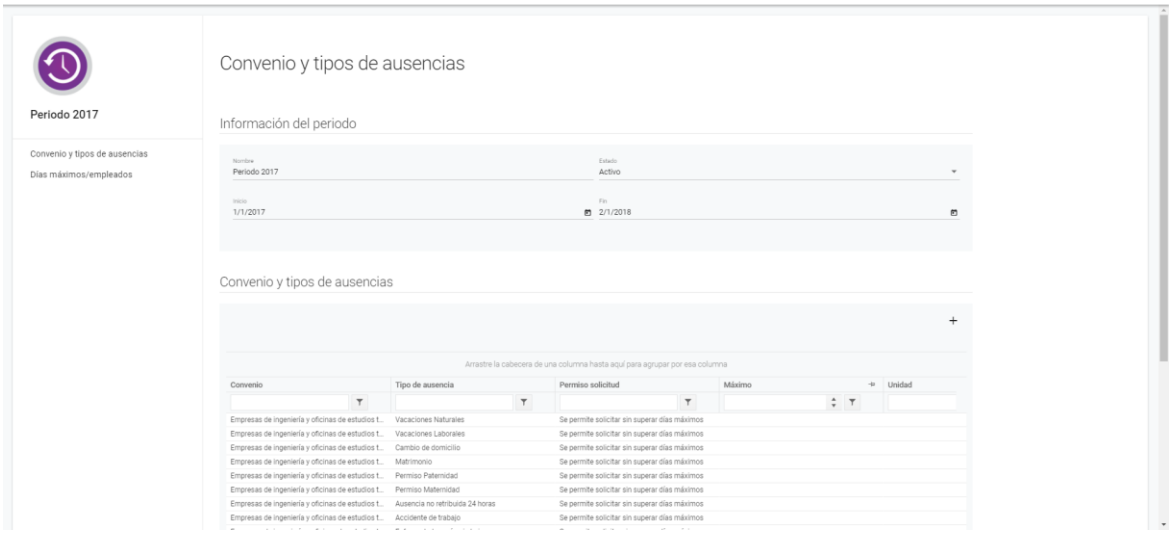


Figura 9. Información asociada al periodo

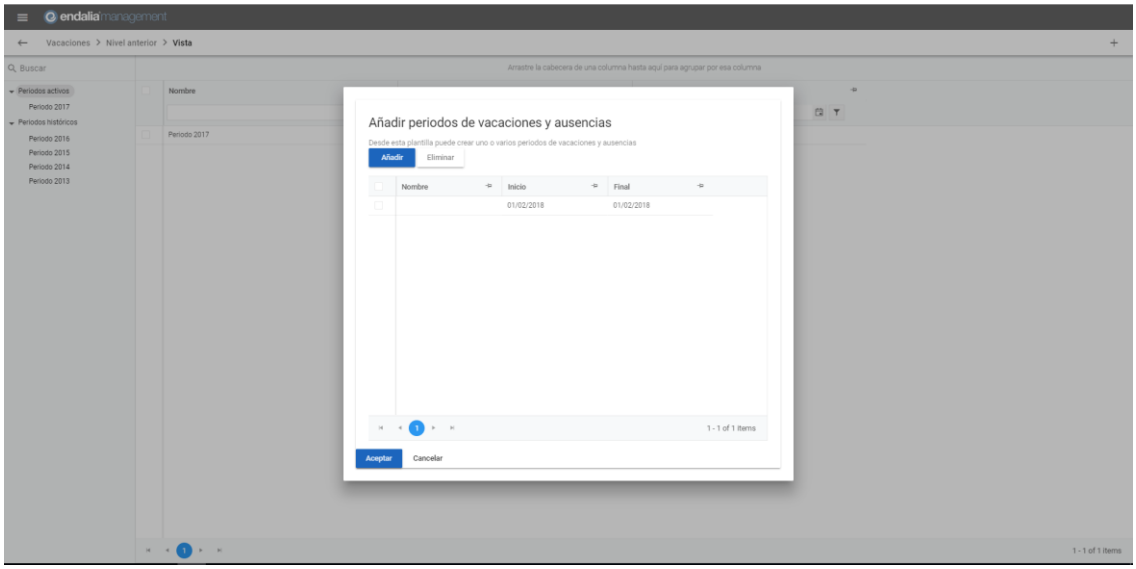


Figura 10. Asistente creación periodos

6.3.4 Página “Solicitudes”

En la figura a continuación (Figura 11) se muestra una captura de pantalla de la sección “Solicitudes”, donde el usuario puede ver un listado de solicitudes junto con la estructura organizativa que permitirá el filtrado de solicitudes por persona o puesto.

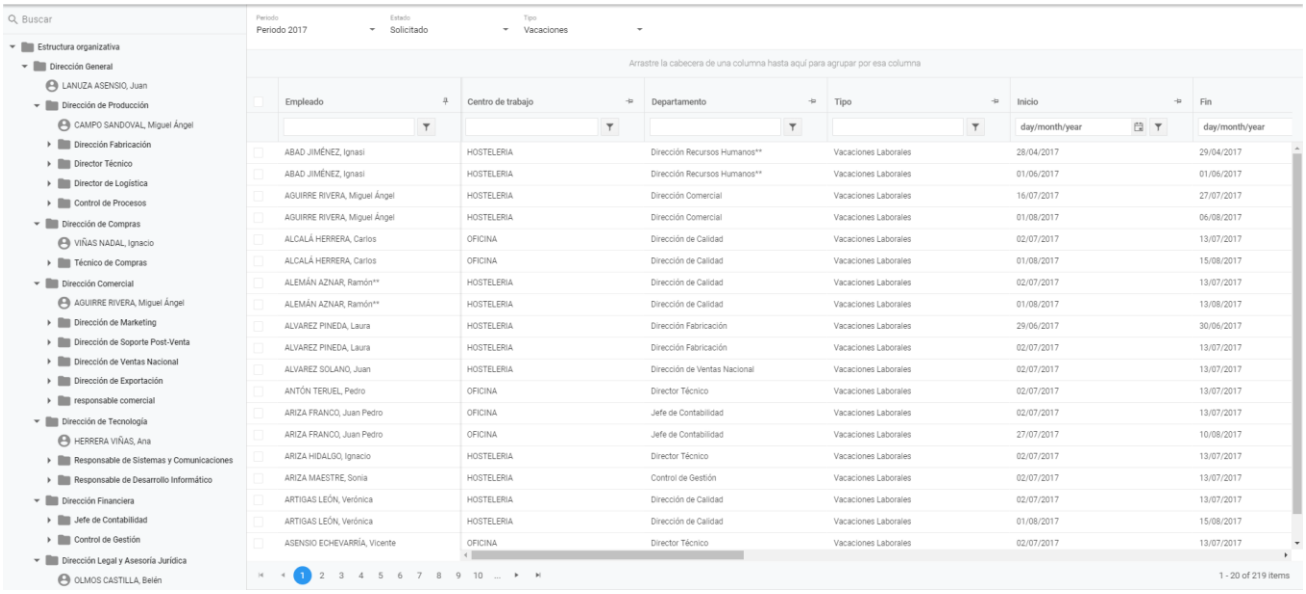


Figura 11. Página solicitudes

Se puede visualizar la información asociada a la solicitud con el menú de navegación superior derecho o haciendo doble clic sobre la fila. Por otro lado, mediante el menú situado en la barra de navegación se puede abrir el asistente para creación de solicitudes (Figura 12). Para la creación se debe introducir primero el empleado y el periodo para que se calcule que tipos de solicitudes puede solicitar y los días máximos disponibles para este.

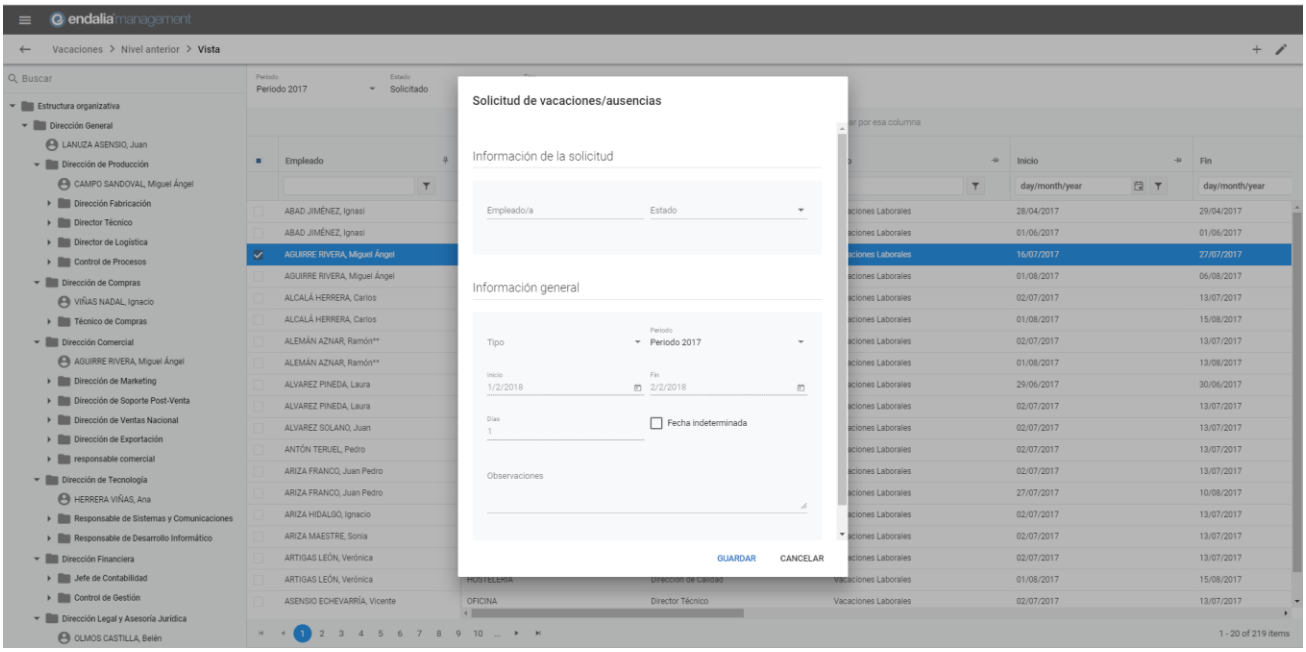


Figura 12. Asistente creación periodo

7. IMPLEMENTACIÓN AUTENTICACIÓN

El proceso de autenticación se basa en el uso de Oauth, un estándar abierto que permite flujos simples de autorización para sitios web o aplicaciones informáticas. El usuario inicia sesión en la aplicación (Figura 7) y en este momento se le asigna un token a partir de sus credenciales de usuario.

El formato de este token está definido por el estándar abierto JSON Web Tokens basado en JSON para crear un token que sirva para enviar datos entre aplicaciones o servicios y garantizar que sean válidos y seguros.

Se envían sus credenciales al servidor que será quien garantice que es un usuario correcto; realizada esta comprobación se le asigna un token que contiene información características del usuario y codificado con una clave temporal, de forma que al cabo de cierto tiempo dicho token caducará.

Para poder realizar peticiones se tendrá que adjuntar el token asignado por el servidor como parte de la cabecera de cada petición. Para esto se ha implementado un servicio que extiende el comportamiento de las peticiones http. Este servicio se encarga de incluir el token asignado así como de volver a solicitarlo en caso de que haya caducado con las credenciales antes introducidas.

El token se almacenará en las cookies lo cual permite además saber si la sesión del usuario ya ha sido iniciada y poder ofrecer contenido personalizado.

Figura 13. Inicio sesión aplicación

8. BIBLIOGRAFÍA

8.1 Referencias

[R1] I. Jacobson, G. Booch, J. Rumbaugh. 2000. “El Proceso Unificado de Desarrollo de Software”. Pearson Education.

8.2 Referencias web

[W1] http://www.microsoft.com/net

[W2] http://www.wikipedia.org

[W3] http://www.uml.org

[W4] http://www.endalia.com

endalia®
HR Consulting & Software

ESTÁNDAR DE DOCUMENTACIÓN

DESARROLLO DE UN SISTEMA DE GESTIÓN DE RECURSOS HUMANOS PARA PYMES

VERSIÓN 2.0
PUBLICADO EL 30/12/2017

HISTÓRICO DE REVISIONES

Fecha	Versión	Descripción	Autor
03/09/2007	1.0	Redacción del Estándar de Documentación de Endalia	Endalia
30/12/2017	2.0	Modificaciones del documento para adaptarlo al TFG	Marina Ariño Armengol

ÍNDICE

1. INTRODUCCIÓN

1.1 Propósito del documento

En el presente documento se define el estándar de documentación del trabajo de desarrollo de un sistema de gestión de recursos humanos para pymes. Se definen los formatos, diseños, tipología de fuentes y plantillas a utilizar en la elaboración de esta documentación.

1.2 Alcance del documento

Las especificaciones de este documento alcanzan a toda la documentación generada durante el proyecto.

1.3 Definiciones

Fuente: un miembro de una familia de tipo de letra.

1.4 Referencias

En este documento no se realizan referencias a otros documentos del trabajo.

1.5 Resumen

El presente documento es el estándar de documentación de Endalia. Se compone de cuatro apartados:

1. Se muestra el propósito del documento y se define su alcance. Se proporciona una lista de acrónimos y definiciones útiles para la comprensión del documento, así como una lista de los documentos del proyecto referenciados y el presente resumen.
2. Se especifica el formato de documentación del proyecto.
3. Plantillas de documentación.
4. Bibliografía y referencias Web utilizadas en la confección de este documento.

2. FORMATO DE DOCUMENTACIÓN

2.1 Fuentes y estilos

A continuación se definen los tipos de fuente utilizados para los diferentes formatos de texto del documento:

- Título 1: Helvética Neue 14, Negrita.
- Título 2: Helvética Neue 12, Negrita.
- Titulo 3: Helvética Neue 10.
- Texto normal: Helvética Neue 9.
- Texto en pie de imágenes: Helvética Neue 9, Cursiva.

Texto de Código fuente o acciones de línea de comandos: Courier New 9.

2.2 Interlineado y formato de página

El interlineado utilizado en la documentación será sencillo. El texto se justificará por ambos márgenes.

Se empezará página nueva entre apartados de primer nivel (Título 1).

Se evitará dejar títulos de segundo y tercer nivel como última línea de una página, siendo ubicados en la página siguiente.

Entre todos los títulos independientemente del nivel, habrá dos saltos de línea de separación.

Las relaciones de elementos se separarán mediante un salto de línea y se indicarán con un punto al principio de la línea, sin tabulado. En el caso de subrelaciones, se indicarán mediante tabulados sin guion de la siguiente manera:

- Elemento 1
    - Elemento de Nivel 2
        - Elemento de Nivel 3
        - Elemento de Nivel 3
        - Elemento de Nivel 3
    - Elemento de Nivel 2
    - Elemento de Nivel 2
    - Elemento de Nivel 2
    - Elemento de Nivel 2
- Elemento 2

2.3 Imágenes y diagramas

Las imágenes y diagramas se colocarán centrados y ajustando en lo posible su tamaño a los márgenes habituales de la página, excepto en el caso de que su tamaño sea excesivo para visualizarlos de manera óptima dentro de esos márgenes, en cuyo caso se colocarán en posición apaisada en una página nueva. Todas las imágenes estarán numeradas y se les referenciará en el texto al que acompañan y mediante una definición centrada debajo de las mismas, de la siguiente manera (Figura 1):

Figura 1. Ejemplo de formato de imagen

3. PLANTILLAS DE DOCUMENTACIÓN

Las plantillas explicadas en este apartado se encuentran guardadas en formato electrónico. Por razones de espacio no se pueden mostrar aquí a tamaño real.

El objetivo de este apartado es especificar el formato de los documentos para cualquiera que desarrolle algún texto para el proyecto. Por ello se muestra una captura a tamaño reducido y una explicación de las partes que las componen y la manera de utilizarlas. Como muestra del aspecto final sirve el presente documento.

3.1 Plantilla de documento

3.1.1 Hoja 1. Portada

El formato de la portada de todos los documentos es el siguiente (Figura 2).

Figura 2. Hoja 1 – Portada

En él se indica el título del documento con fuente Helvética Neue 60 y se incluye información de copyright y de control de distribución y autorización.

Todas las demás páginas del documento poseen un pie de página común, en el que aparece el logotipo de Endalia e incluye el título del documento, así como información del número de página.

3.1.2 Hoja 2. Información de copyright

El formato de la segunda hoja de todos los documentos es el siguiente (Figura 3).

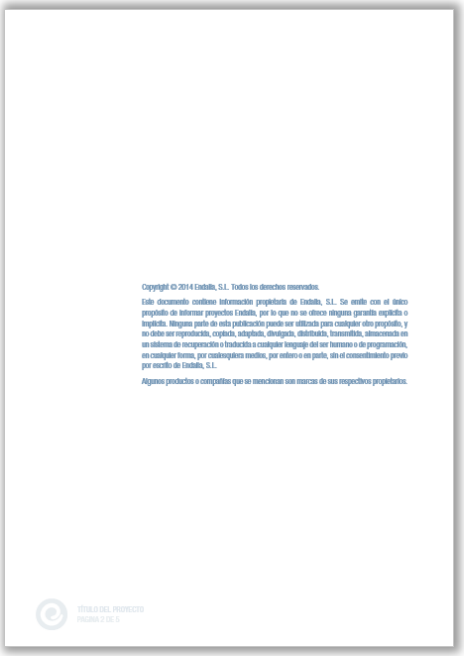


Figura 3. Hoja 2 - Información de copyright

En esta hoja, aparte del pie de página, aparece información relativa al copyright del documento.

3.1.3 Hoja 3. Histórico de revisiones

El formato de la tercera hoja de todos los documentos es el siguiente (Figura 4).

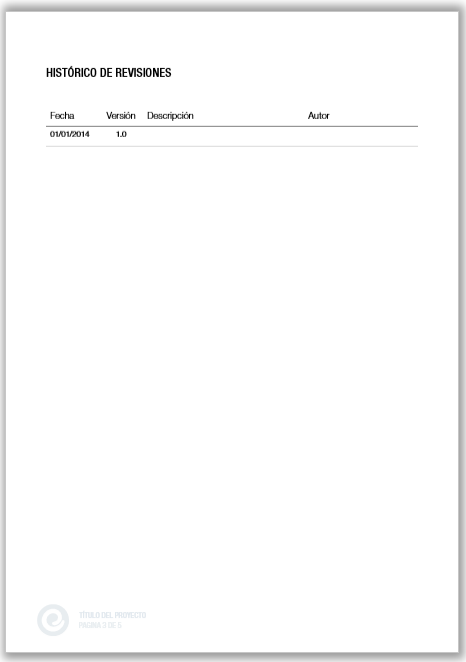


Figura 4. Hoja 3 - Histórico de revisiones

En esta hoja, aparte del pie de página, aparece un cuadro con información acerca de las sucesivas revisiones realizadas sobre el documento.

3.1.4 Hoja 4 y siguientes. Índice

El formato del índice se indica en la siguiente imagen (Figura 5). Se genera automáticamente con Word, estableciendo el tipo de letra de los apartados de nivel 1 a Helvética Neue 9.

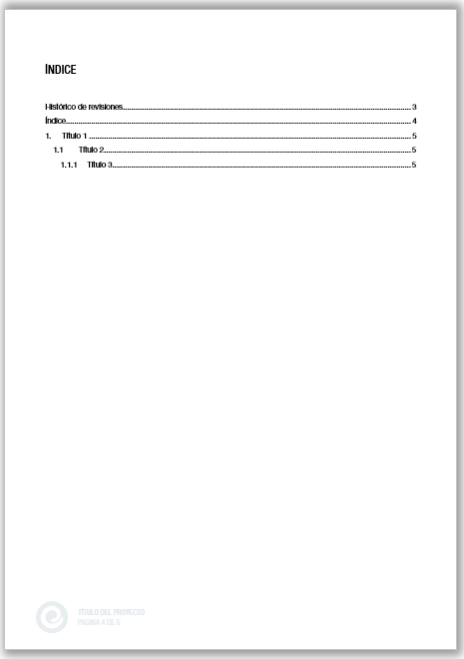
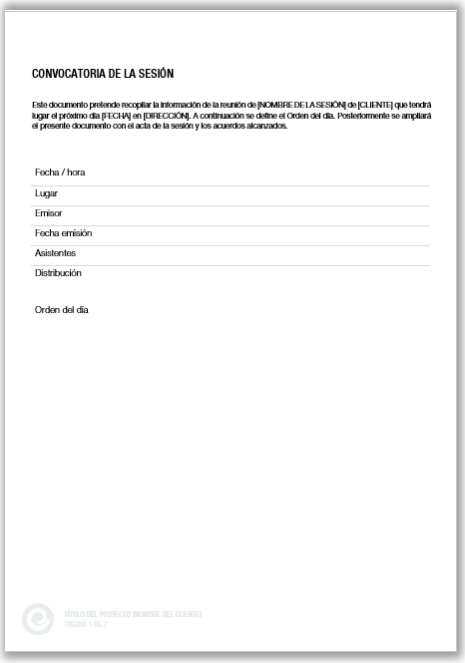


Figura 5. Índice

3.2 Plantilla de acta de reunión

El formato de plantilla de acta de reunión es el siguiente (Figura 6 y Figura 7). En la primera hoja se incluye la fecha, hora y lugar de la reunión, la persona emisora del documento, la fecha de emisión del mismo, los asistentes, la distribución y el orden del día. En la segunda hoja se incluye el acta de la reunión, donde aparecen, divididos por secciones, los distintos acuerdos alcanzados, la persona responsable de los mismos, la fecha de compromiso y el estado, expresado en cifra porcentual.



CONVOCATORIA DE LA SESIÓN

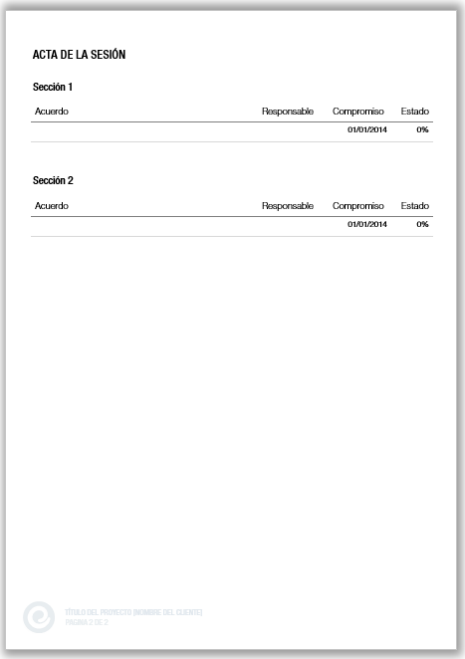
Este documento pretende recopilar la información de la reunión de [NOMBRE DE LA SESIÓN] de [CLIENTE] que tendrá lugar el próximo día [FECHA] en [DIRECCIÓN]. A continuación se detine el Orden del día. Posteriormente se ampliará el presente documento con el acta de la sesión y los acuerdos alcanzados.

Fecha / hora	
Lugar	
Emisor	
Fecha emisión	
Asistentes	
Distribución	

Orden del día

TÍTULO DEL PROYECTO [NOMBRE DEL CLIENTE]
PÁGINA 1 DE 2

Figura 6. Acta de reunión, hoja 1



ACTA DE LA SESIÓN

Sección 1

Acuerdo	Responsable	Compromiso	Estado
		01/01/2014	0%

Sección 2

Acuerdo	Responsable	Compromiso	Estado
		01/01/2014	0%

TÍTULO DEL PROYECTO [NOMBRE DEL CLIENTE]
PÁGINA 2 DE 2

Figura 7. Acta de reunión, hoja 2

4. BIBLIOGRAFÍA

4.1 Referencias

[R1] Edward J Huth *Scientific Style and Format: The CBE Manual for Authors, Editors, and Publishers* Cambridge University Press 1994

[R2] Estándar de documentación de Endalia S.L.

4.2 Referencias web

[W1] http://www.wikipedia.org

[W2] http://www.monografias.com/trabajos6/dosi/dosi.shtml

[W3] http://www.apa.org/journals/webref.html

endalia®
HR Consulting & Software

PRUEBAS

DESARROLLO DE UN SISTEMA DE GESTIÓN DE RECURSOS HUMANOS PARA PYMES

VERSIÓN 1.0
PUBLICADO EL 03/01/2018

HISTÓRICO DE REVISIONES

Fecha	Versión	Descripción	Autor
15/12/2017	1.0	Redacción inicial del documento	Marina Ariño Armengol

ÍNDICE

1. INTRODUCCIÓN

1.1 Propósito del documento

El presente documento describe la fase de pruebas del sistema de gestión de Recursos Humanos para pymes desarrollado. El objetivo de esta fase es el de asegurar la calidad del software y el correcto funcionamiento de la implementación realizada en la fase anterior, así como garantizar que se cumplen los requisitos planteados al comienzo de la realización de este trabajo.

1.2 Alcance del documento

Este documento contiene los resultados obtenidos en la última de las fases del trabajo: la realización de pruebas del sistema de gestión de Recursos Humanos para pymes.

1.3 Acrónimos

- PU: Prueba Unitaria
- PS: Prueba de Sistema

1.4 Definiciones

- Caso de prueba: conjunto de condiciones o variables bajo las cuáles un analista determinará si una aplicación, un sistema software, o una característica de éstos es parcial o completamente satisfactoria.
- Prueba de sistema: conjunto de verificaciones de la interconexión de varios subsistemas dentro de una aplicación o sistema software.
- Prueba unitaria: conjunto de verificaciones de un subsistema de una aplicación o sistema software.

1.5 Referencias

Este documento contiene referencias a los siguientes documentos del trabajo, contenidos en la sección “Anexos”:

- Especificación de requisitos: documento que describe la especificación de requisitos realizada a partir de las necesidades del cliente.
- Análisis del sistema: documento que describe la arquitectura del sistema desarrollado, junto con un listado de casos de uso y su estructura de paquetes.

1.6 Resumen

En este documento se presentan el proceso y los resultados de la fase de pruebas del sistema de gestión de Recursos Humanos para pymes. Se compone de los cinco apartados siguientes:

1. Introducción, propósito y alcance del documento.
2. Descripción del proceso de pruebas realizado en este proyecto.
3. Descripción y resultados de las pruebas unitarias.
4. Descripción y resultados de las pruebas de sistema.
5. Presentación de la bibliografía utilizada para la realización de este documento.

2. DESCRIPCIÓN DEL PROCESO

Se han realizado una serie de pruebas sobre el sistema desarrollado con el objetivo de asegurar tanto el correcto funcionamiento como la calidad del software realizado. Se puede introducir esta fase en cualquier punto del desarrollo de un software, dependiendo del tipo de pruebas que se vayan a realizar. En este caso, y dado que el objetivo final del trabajo es la entrega de un software funcional a un cliente, se ha decidido colocar estas pruebas como la última de las fases del desarrollo software, para obtener la máxima información posible sobre la calidad del sistema entregado al cliente. Por otro lado, cabe destacar que el hecho de que esta fase sea la última no significa que el desarrollo termine necesariamente aquí. Si el istema no superara alguna de las pruebas realizadas se deberá volver a una fase anterior para realizar los cambios necesarios en el software. Se trata por lo tanto de un proceso circular.

Para este desarrollo en concreto se han planteado dos tipos de pruebas bien diferenciadas:

1. Pruebas unitarias.
2. Pruebas de sistema.

Las pruebas unitarias responden a la necesidad de testear pequeños componentes del módulo que por sus implicaciones en el conjunto del sistema se consideran más críticas o importantes. Se trata de casos fácilmente aislables y, por lo tanto, sencillos de comprobar de forma semiautomática.

Por otro lado, las pruebas de sistema permiten comprobar la calidad del software desarrollado mediante la validación de la comunicación entre los distintos subsistemas que lo componen. Se tratan en este de casos de pruebas más amplios, difícilmente automatizables, pero que aseguran el correcto funcionamiento de la totalidad del sistema en su conjunto. Idealmente, y en el seno de la organización en la que se realiza este trabajo, estas pruebas siempre se realizan por un equipo independiente al que ha desarrollado el sistema. No obstante, dada la naturaleza académica de este proyecto, dichas pruebas fueron realizadas con anterioridad por el responsable del trabajo, para luego ser nuevamente validadas por el equipo independiente.

Por último, cabe destacar que el conjunto de pruebas seleccionado responde al último de los propósitos de este trabajo: realizar un software que responda a las necesidades del cliente, las cuales han sido definidas en las fases de requisitos y análisis del proyecto. Por lo tanto, un sistema que supere con éxito la colección de pruebas detalladas a continuación será un sistema que solvente correctamente las problemáticas planteadas por el cliente al comienzo de este trabajo.

3. PRUEBAS UNITARIAS

Una prueba unitaria tiene como objetivo el comprobar el correcto funcionamiento de una parte de código, pudiendo ser un método o un conjunto de éstos. Para realizar correctamente las pruebas unitarias, se han de crear casos de prueba que sean independientes entre ellos y del resto de la aplicación. Así, se podrá comprobar el correcto funcionamiento de dicha parte de código.

Una prueba unitaria debe ser, idealmente:

- Automatizable: debe ejecutarse en su totalidad sin intervención manual.
- Completa: debe abarcar la mayor cantidad de código posible.
- Repetible: debe poder ser ejecutada en múltiples ocasiones.
- Independiente: la ejecución de una prueba no puede afectar a la ejecución de otra.
- Profesional: debe estar documentada y formar parte de la documentación del proyecto.

Dada la gran cantidad de casos unitarios que existen en un proyecto de estas características se presentan únicamente pruebas de algunas partes del código, consideradas críticas o complejas.

Identificador	**PU - 01**
Nombre	Modificación información empleado.
Descripción	Edición de los datos que se deseen modificar. Deberán realizarse comprobaciones de validez en el DNI, email y número de la seguridad social así como que el código del empleado sea único.
Resultados esperados	La información asociada al empleado se modifica si no existen datos erróneos.
¿Prueba superada?	Sí.

Identificador	**PU – 02**
Nombre	Cálculo días vacaciones.
Descripción	El cálculo de los días de vacaciones que puede solicitar un empleado tiene que tener en cuenta varios factores como por ejemplo el tipo de solicitud realizada, el convenio asociado al empleado o algunos campos de la propia solicitud. En esta prueba se realiza una solicitud y se comprueba que los días son calculados correctamente.
Resultados esperados	Creación de la solicitud.
¿Prueba superada?	Sí.

Identificador	**PU – 03**
Nombre	Añadir nuevo convenio y tipo de ausencia.
Descripción	Creación de una nueva relación de convenio y tipo de ausencia para un periodo indicando el permiso asignado junto con el tiempo máximo a solicitar. Debe comprobarse que no exista ya la relación.
Resultados esperados	Creación de la nueva relación convenio/tipo de ausencia.
¿Prueba superada?	Sí.

4. PRUEBAS DE SISTEMA

Las pruebas de sistema tienen como objetivo validar que se cumplen correctamente los casos de uso definidos anteriormente en el desarrollo del software. Por esto, se han realizado diferentes pruebas de sistema en función de los casos de uso planteados en la fase de análisis.

Estas pruebas permiten, a diferencia de las pruebas unitarias, comprobar el correcto funcionamiento de los diferentes subsistemas que componen este sistema y verificar que éstos se comunican correctamente entre ellos.

A continuación se presentan un subconjunto de las que han sido realizadas.

Identificador	**PS – 01**
Nombre	Visualización de una solicitud.
Descripción	Se ha de comprobar la correcta visualización de una solicitud.
Resultados esperados	La aplicación debe mostrar la información relativa a la solicitud, con todos los campos que contengan información rellenos.
¿Prueba superada?	Sí.

Identificador	**PS – 02**
Nombre	Creación de una solicitud.
Descripción	Se ha de comprobar el proceso de creación de una solicitud mediante el asistente de creación de solicitudes.
Resultados esperados	La aplicación debe en primer lugar mostrar el asistente de creación de solicitudes. Una vez introducida la información en el asistente debe comprobar que todos los campos obligatorios se han rellenado. Si falta alguna información, se informará al usuario permitiéndole modificar los valores introducidos, sino se creará la solicitud.
¿Prueba superada?	Sí.

Identificador	**PS – 03**
Nombre	Modificación de una solicitud.
Descripción	Se ha de comprobar la correcta edición de una solicitud ya creada.
Resultados esperados	La aplicación debe comprobar que todos los campos obligatorios siguen rellenados de forma correcta tras la edición. Si no es así, deberá informar al usuario. Si todo está correcto, deberá actualizar la base de datos con los cambios introducidos por el usuario.
¿Prueba superada?	Sí.

Identificador	**PS – 04**
Nombre	Asignación de empleados a puestos.
Descripción	Se ha de comprobar la correcta asignación de los empleados a los puestos ya existentes.
Resultados esperados	La aplicación debe en primer lugar mostrar el asistente de asignación de empleados a puestos. Una vez introducida la información en el asistente debe comprobar que todos los campos obligatorios se han rellenado. Si falta alguna información, se informará al usuario permitiéndole modificar los valores introducidos, sino se asignarán los empleados al puesto.
¿Prueba superada?	Sí.

Identificador	**PS – 05**
Nombre	Creación de un puesto.
Descripción	Se ha de comprobar el proceso de creación de un puesto mediante el asistente de creación de puestos.
Resultados esperados	La aplicación debe en primer lugar mostrar el asistente de creación de puestos. Una vez introducida la información en el asistente debe comprobar que todos los campos obligatorios se han rellenado. Si falta alguna información, se informará al usuario permitiéndole modificar los valores introducidos, sino se creará el puesto en estructura.
¿Prueba superada?	Sí.

5. BIBLIOGRAFÍA

5.1 Referencias

Este documento no contiene referencias a textos impresos.

5.2 Referencias web

[W1] http://www.wikipedia.org

[W2] http://www.endalia.com

endalia®
HR Consulting & Software

ESTÁNDAR DE CODIFICACIÓN

DESARROLLO DE UN SISTEMA DE GESTIÓN DE RECURSOS HUMANOS

VERSIÓN 2.0
PUBLICADO EL 30/12/2017

HISTÓRICO DE REVISIONES

Fecha	Versión	Descripción	Autor
03/09/2007	1.0	Redacción del Estándar de Documentación de Endalia	Endalia
23/08/2016	2.0	Modificaciones del documento para adaptarlo al TFG	Marina Ariño Armengol

ÍNDICE

1. INTRODUCCIÓN

1.1 Propósito del documento

El presente documento describe las normas que deben seguirse en el desarrollo de cualquier tipo de elemento de codificación realizado en este trabajo. El objetivo primordial de la confección de un estándar de codificación se basa en homogeneizar el proceso de implementación y codificación, para lograr obtener beneficios en la comprensión del código, en la verificación y validación del mismo, y en posibles modificaciones posteriores.

Existen un gran número de razones por las que definir las convenciones de código es importante para los programadores:

- El 80% del coste del código de un programa se invierte en su mantenimiento.
- Casi ningún software lo mantiene toda su vida el autor original.
- Las convenciones de código mejoran la lectura del software, permitiendo entender código nuevo mucho más rápidamente y más a fondo.
- La distribución de código fuente como producto exige su presentación de manera adecuada.

De este modo, el presente documento pretende ser una colección de reglas que deben aplicarse a todo el código generado, con el propósito de que sea homogéneo. Esta homogeneidad permitirá una comprensión más efectiva del código tanto para su autor como para otros programadores, facilitando su distribución y mantenimiento.

1.2 Alcance del documento

Este documento se ubica dentro de la fase inicial de desarrollo del sistema de gestión de recursos humanos para pymes, como elemento necesario previo a la realización de cualquier tipo de código fuente del proyecto, y será utilizado como guía durante toda la fase de implementación.

1.3 Acrónimos

- ANSI: American National Standards Institute.
- HTML: Hypertext Markup Language.
- ID: Identificador.
- URL: Uniform Resource Locator.

1.4 Definiciones

- Pascal-Casing: Notación en la que un identificador está compuesto por múltiples palabras juntas comenzando cada una de ellas por una letra mayúscula.
- Camel-Casing: Notación similar a Pascal-Casing con la excepción de que la letra inicial del identificador debe ser minúscula.

1.5 Referencias

En este documento no se realizan referencias a otros documentos del trabajo.

1.6 Resumen

El presente documento describe las normas que deben seguirse en el desarrollo de cualquier tipo de elemento de codificación realizada en este proyecto. Se compone de 5 apartados:

1. Se muestra el propósito del documento y se define su alcance. Se proporciona una lista de acrónimos y definiciones útiles para la comprensión del documento, así como una lista de los documentos del proyecto referenciados y el presente resumen.
2. Se muestra la estructura de codificación de los diferentes tipos de archivos de código fuente desarrollados en el trabajo.
3. Se muestran reglas, recomendaciones y buenas prácticas para el desarrollo del código fuente del trabajo.
4. Se muestra el estándar de nombrado de los elementos de base de datos.
5. Bibliografía y referencias Web utilizadas en la confección de este documento.

2. ESTRUCTURA DE LOS FICHEROS

2.1 Introducción

En este apartado se define la estructura y organización de los archivos de código fuente del proyecto. Se especifica la codificación tanto de los archivos de clases, acceso a datos y servicios web (.cs) como de los archivos de interfaz de usuario (.aspx) y los archivos de código subyacente (.aspx.cs).

2.2 Codificación de archivos de interfaz de usuario .aspx

A continuación se muestra la estructura de codificación de los archivos de interfaz de usuario. Para una comprensión más sencilla se muestra mediante una tabla con dos columnas. En la columna de la izquierda aparece un índice para facilitar la posterior descripción de la región de la estructura definida en la columna de la derecha.

<1>Declaración de página	`<%@ Page Language="C#"`
	`AutoEventWireup="true"`
	`Inherits="myNamespace"`
	`MasterPageFile="myMaster"`
	`CodeBehind="myClass" %>`

<2>Importación *de controles*	`<%@ Register TagPrefix="myControlPrefix"`
	`TagName="myControl"`
	`Src="myControlSrc" %>`

<3>Definición de elementos de la página	`<Código de definición de elementos HTML>`

2.2.1 Declaración de página

En esta región se define la página o en su caso el control que representa el archivo. Es en esta región donde se enlaza la página con el código subyacente y con un fichero Master (en caso de definir una página y no un control).

2.2.2 Importación de controles

En esta región se definen los controles que se utilizarán a lo largo de la página. Es necesario referenciarlos mediante un prefijo y un nombre, así como indicar en qué ruta se encuentra su código fuente.

2.2.3 Definición de elementos de la página

En esta región se define la estructura de la página o del control que representa el archivo, mediante el uso de elementos tanto HTML como del framework de ASP.NET.

2.3 Codificación de archivos de código subyacente .aspx.cs

A continuación se muestra la estructura de codificación de los archivos de código subyacente que será comentada posteriormente. De la misma manera que en el apartado anterior, se muestra mediante una tabla con dos columnas. En la columna de la izquierda aparece un índice para facilitar la posterior descripción de la región de la estructura definida en la columna de la derecha.

<1>Directivas *using*	`using System;` `<directivas using>`

<2>Declaración *namespace* y *class*	`namespace MyNamespace1`
	`{`
	`public class MyClass`
	`{`

<3> Región variables	`#region variables`
	`#region Variables I18N`
	`<declaraciones de variables de internacionalización>`
	`#endregion Variables I18N`
	`#region Variables globales`
	`<declaraciones de variables globales>`
	`#endregion Variables globales`
	`#endregion variables`

<4> Región *Page_Load*	`#region Page_Load`
	`<Código Page_Load>`
	`#endregion Page_load`

<5> Región *I18N*	`#region I18N`
	`<Código LoadI18N>`
	`<Código class_Init>`
	`#endregion I18N`

<6> Región *DataBind*	`#region DataBind`
	`<Código DataBind>`
	`#endregion DataBind`

<7> Región Eventos	`#region Eventos`
	`<Código Eventos provenientes de controles>`
	`#endregion Eventos`

<8> Región Eventos de botones	`#region Eventos de botones`
	`<Código Eventos clic botones>`
	`#endregion Eventos de botones`

<9> Región Métodos de ordenación	#region Métodos de ordenación <Código Métodos de ordenación> #endregion Métodos de ordenación
<10> Región Código generado *Web Forms*	#region Código generado por el diseñador de Web Forms #endregion Código generado por el diseñador de Web Forms
	} }

2.3.1 Consideraciones generales sobre las regiones

Como se ha visto en el apartado 2.3 y para facilitar la organización y estructuración del código fuente se utilizan las directivas #region y #endregion. Las regiones no aportan funcionalidad como tal, se utilizan para marcar y agrupar una sección del código. Las regiones que se especifican en el apartado 2.3 son las obligatorias en el caso de que aparezcan los elementos para los que han sido definidas. En caso de que aparezcan elementos no especificados en las regiones del apartado anterior podrán definirse nuevas regiones para especificar la sección de código referida a los eventos y métodos del elemento o control. En cualquier caso, no se permitirán eventos o métodos que no estén incluidos dentro de alguna región.

2.3.2 Región directivas *using*

En esta región se colocarán, por orden alfabético creciente, las directivas que especifiquen las clases utilizadas en el código fuente definido en la clase actual.

2.3.3 Región declaración *namespace* y *class*

En esta región se colocarán las cabeceras que especifican el espacio de nombres en los que se integra el código y el nombre de la clase. Esta última irá precedida por una cabecera en la que se especificarán los siguientes datos:

```
/// <summary>
/// Nombre del fichero : Nombre del fichero
/// Autor              : Nombre del autor
/// Descripción        : Descripción de la funcionalidad y objetivo del fichero
/// </summary>
```

2.3.4 Región declaración de variables

Esta región estará integrada por tres subregiones que se especifican a continuación:

- Región variables *I18N*: en esta región aparecen las declaraciones de cadenas de internacionalización que son obtenidas del archivo de recursos y que se utilizan para definir todos los textos que son presentados al usuario.
- Región variables globales: en esta región aparecen las declaraciones de variables globales utilizadas acompañadas de una descripción de su funcionalidad.

2.3.5 Región *Page Load*

En esta región se coloca el código del evento *Page_Load* que se lanza cada vez que la página es lanzada o recargada.

2.3.6 Región *I18N*

Esta región está integrada por dos métodos:

- Método *Init()*: realiza la lectura del fichero de recursos en el que se almacenan las cadenas de internacionalización, cargando estas en variables de cadena.
- Método *LoadI18N()*: carga en los campos de texto del aspx las variables de cadenas obtenidas en el método *Init*.

2.3.7 Región *DataBind*

En esta región se colocan los métodos que enlazan orígenes de datos a controles de servidor como *DataGrids* o árboles.

2.3.8 Región Eventos

Se utiliza está región para definir para cada uno de los métodos que capturan los diversos eventos de diversos controles y que se especifican en la tabla del apartado 2.2.

2.3.9 Región Eventos de botones

En esta región se colocan los métodos que capturan los eventos clic de los distintos botones ubicados en el archivo aspx.

2.3.10 Región Métodos de ordenación

Esta región engloba los métodos que ordenan los diferentes orígenes de datos que son utilizados en el código.

2.3.11 Región Código generado por el diseñador de *Web Forms*

Esta región se genera automáticamente gracias a la herramienta de desarrollo, y consta de dos métodos:

- *InitializeComponent()*: contiene las declaraciones de los métodos del código que capturan los diferentes eventos producidos por los controles ubicados en el código aspx.
- *OnInit()*: este método se lanza al cargar la página y realiza la llamada al método *InitializeComponent* para comenzar la ejecución del código del servidor.

2.4 Codificación de archivos de clase, acceso a datos y servicios web .cs

A continuación se muestra la estructura de codificación de los archivos de definición de clases, acceso a datos y servicios web. De la misma manera que en los apartados anteriores, se muestra mediante una tabla con dos columnas. En la columna de la izquierda aparece un índice para facilitar la posterior descripción de la región de la estructura definida en la columna de la derecha.

<1>Directivas *using*	`using System;` `<directivas using>`

<2>Declaración *namespace* y *class*	`namespace MyNamespace1`
	`{`
	`public class MyClass`
	`{`

<3> Región atributos	`#region atributos`
	`<Declaración atributos>`
	`#endregion atributos`

<4> Región propiedades	`#region propiedades`
	`<declaración propiedades>`
	`#endregion propiedades`

<5> Región constructores	`#region constructores`
	`<Métodos constructores clase>`
	`#endregion constructores`

<6> Región Métodos	`#region métodos`
	`<Código Métodos>`
	`#endregion métodos`

	`}` `}`

2.4.1 Región directivas *using*

En esta región se colocarán, por orden alfabético creciente, las directivas que especifiquen las clases utilizadas en el código fuente definido en la clase actual.

2.4.2 Región declaración *namespace* y *class*

En esta región se colocarán las cabeceras que especifican el espacio de nombres en los que se integra el código y el nombre de la clase. Esta última irá precedida por una cabecera en la que se especificarán los siguientes datos:

```
/// <summary>
/// Nombre del fichero : Nombre del fichero
/// Autor              : Nombre del autor
/// Descripción        : Descripción de la funcionalidad y objetivo del fichero
/// </summary>
```

2.4.3 Región atributos

En esta región se colocará la declaración de los atributos de una clase. Los atributos se nombrarán mediante Pascal-Casing exceptuando la primera letra, que será en minúscula y precedida por un guion bajo de este modo:

```
private int _requestID;
```

Asimismo en esta región se declararán las constantes de la clase, que se nombrarán con mayúsculas.

2.4.4 Región propiedades

En esta región se coloca la declaración de las propiedades públicas de la clase. Las propiedades se nombran con Pascal-Casing y, en el caso de que representen el acceso al valor de un atributo de la clase, su nombre es el mismo del atributo sin el guión bajo y con la primera letra en mayúscula, especificando el acceso a los métodos *get* y *set* de este modo:

```
public int RequestID
{
       get{ return _requestID; }
       set{ _requestID = value; }
}
```

2.4.5 Región constructores

En esta región se colocará la declaración de los métodos constructores de la clase.

2.4.6 Región métodos

En esta región se colocará la declaración de los métodos de la clase.

2.5 Codificación de archivos de recursos de internacionalización

Los archivos .txt a partir de los cuales se generan los archivos de recursos de internacionalización se construirán del siguiente modo:

Para cada una de las secciones del programa que tengan una entidad lo suficientemente importante como para ser diferenciada, se colocará un comentario y a continuación la relación de las etiquetas. El nombrado de las etiquetas se hace del siguiente modo:

- La parte inicial del nombre de la etiqueta será la misma que el nombre del archivo .aspx en el que se utilizará la etiqueta. A continuación se colocará un guion bajo seguido de un prefijo que indicará la utilización de la etiqueta seguida de un guion bajo, siguiendo la siguiente convención:
    - etiqueta o campo de texto: `_lbl_`
    - etiqueta de *hyperlink*: `_lnk_`
    - texto de botón: `_btn_`
- En el caso de que la etiqueta sea un *tooltip,* se colocará a continuación el sufijo `_ToolTip`
- A continuación se colocará un nombre descriptivo de la función de la etiqueta que utilizará Pascal-Casing. No se especifica una norma rígida para este nombrado pero a continuación se muestran unos ejemplos que muestran buenas prácticas del mismo.

```
MyTrips_lbl_lblCost
MyTrips_btn_btnSend
MyTrips_btn_btnSend_ToolTip
```

3. REGLAS DE CODIFICACIÓN

3.1 Identación

Dada la actual uniformidad y estandarización de los editores utilizados para el desarrollo de código C# en .NET, se utilizará el tabulador como unidad de identación estándar. Los comentarios se identarán al mismo nivel de identación que el código que se esté documentando.

3.1.1 Longitud de línea

Se recomienda no escribir líneas con más de 80 caracteres, ya que no son bien manejadas por muchos terminales y herramientas.

3.1.2 Ruptura de líneas

Cuando una expresión no entre en una sola línea, se debe romper de acuerdo a estos principios generales:

- Romper después de una coma.
- Romper antes de un operador.
- Preferir las rupturas de alto nivel a las de bajo nivel.
- Alinear la nueva línea con el principio de la expresión al mismo nivel de la línea anterior.

3.2 Comentarios

En C# hay tres formas de escribir comentarios:

- La primera consiste en encerrar todo el texto que se desee comentar entre caracteres /* y */ siguiendo la siguiente sintaxis:

```
/*<texto>*/
```

    - Estos comentarios pueden abarcar tantas líneas como sea necesario. No es posible anidar comentarios de este tipo.

- En la segunda se considera como indicador del comienzo del comentario la pareja de caracteres // y como indicador de su final el fin de línea. Por tanto, la sintaxis que siguen estos comentarios es:

```
// <texto>
```

- La tercera manera es utilizando el trío de caracteres ///. Este tipo de comentario tiene la particularidad de ser reconocido y utilizado por las herramientas de generación automática de documentación y será el utilizado para la descripción de métodos y clases, ya que en el caso de los primeros genera la estructura de etiquetas o *tags* de documentación de nombres, parámetros y valores de retorno utilizados para la documentación automatizada.

```
/// <texto>
```

3.2.1 Aplicación de los comentarios

Como se ha comentado en el punto anterior, el tercer tipo de comentario (el que va precedido de los caracteres ///) se utiliza para crear de manera automática las etiquetas que permiten la generación automática de documentación.

Aparte de este punto, los comentarios deberían usarse para una introducción del código y proporcionar información adicional que no está disponible en el propio código. Los comentarios sólo deberían tener información que sea relevante para leer y entender el programa. Por ejemplo, información sobre cómo está construida la clase correspondiente o en qué directorio reside no debería ser incluida como comentarios.

Las discusiones no triviales o decisiones de diseño no obvias son apropiadas, pero debemos evitar la duplicidad de información que esté presente en el código. Es demasiado fácil que los comentarios redundantes se queden anticuados. En general, debemos evitar cualquier comentario que se pueda quedar anticuado cuando el código evolucione.

La frecuencia en los comentarios algunas veces refleja una pobre calidad de código. Cuando nos sintamos obligados a llenarlo de comentarios, debemos considerar la reescritura del código para hacerlo más claro. Los comentarios no deben encerrarse en grandes cajas dibujadas con asteriscos u otros caracteres. Los comentarios nunca deberían incluir caracteres especiales como saltos de página, etc.

Los siguientes puntos son técnicas de comentarios recomendadas.

- Cuando se modifica el código, se mantienen siempre actualizados los comentarios circundantes.
- Evitar los comentarios recargados, como las líneas enteras de asteriscos. En su lugar se utilizan espacios para separar los comentarios y el código.
- Evitar rodear un bloque de comentarios con un marco tipográfico. Puede resultar agradable, pero es difícil de mantener.
- Antes de la implementación, quitar todos los comentarios temporales o innecesarios, para evitar cualquier confusión en la futura fase de mantenimiento.
- Si se necesita realizar comentarios para explicar una sección de código compleja, examinar el código para decidir si se debería volver a escribir. Siempre que sea posible, no documentar un código malo, sino volver a escribirlo. Aunque, por regla general, no debe sacrificarse el rendimiento para hacer un código más simple para el usuario, es indispensable un equilibrio entre rendimiento y mantenibilidad.
- Usar frases completas al escribir comentarios. Los comentarios deben aclarar el código, no añadirle ambigüedad.
- Ir comentando al mismo tiempo que se programa, porque probablemente no habrá tiempo de hacerlo más tarde. Por otro lado, aunque se tuviera oportunidad de revisar el código que se ha escrito, lo que parece obvio hoy es posible que seis semanas después no lo sea.
- Evitar comentarios superfluos o inapropiados, como comentarios divertidos al margen.
- Usar los comentarios para explicar el propósito del código como si fueran traducciones interlineales.
- Comentar cualquier cosa que no sea legible de forma obvia en el código.
- Para evitar problemas recurrentes, hacer siempre comentarios al depurar errores y solucionar problemas de codificación, especialmente cuando se trabaje en equipo.
- Hacer comentarios en el código que esté formado por bucles o bifurcaciones lógicas. Se trata en estos casos de áreas clave que ayudarán a los lectores del código fuente.
- Realizar los comentarios en un estilo uniforme, respetando una puntuación y estructura coherentes a lo largo de toda la aplicación.
- Separar los comentarios de sus delimitadores mediante espacios. Si se respeta esta norma, los comentarios

serán más claros y fáciles de localizar si trabaja sin indicaciones de color.

3.2.2 Formatos de implementación de comentarios

Los programas pueden tener cuatro estilos de implementación de comentarios:

- Bloque de comentarios: Los bloques de comentarios se usan para proporcionar descripciones de ficheros, métodos, estructuras de datos y algoritmos. Los bloques de comentarios podrían usarse al principio de cada fichero y antes de cada método. También pueden usarse en otros lugares, como dentro de los métodos. Para este tipo de comentario se preferirá la estructura /* - */. Un bloque de comentario debería ir precedido por una línea en blanco para configurar un apartado del resto del código:

```
/*
 * Esto es un bloque de comentarios.
 */
```

- Comentarios de una línea: Los comentarios cortos pueden aparecer como una sola línea identada al nivel del código que la sigue. Si un comentario no se puede escribir en una sola línea, debería seguir el formato de los bloques de comentario. Un comentario de una sola línea debería ir precedido de una sola línea en blanco. Para este tipo de comentario se preferirá utilizar los caracteres '//'. A continuación se muestra un ejemplo:

```
if (condition)
{
        // Código de la condición.
        ...
}
```

- Comentarios finales: Los comentarios muy cortos pueden aparecer en la misma línea que el código que describen, pero deberían separarse lo suficiente de las sentencias. Si aparece más de un comentario en el mismo trozo de código, deberían estar identados a la misma altura. Para este tipo de comentario se preferirá utilizar los caracteres '//'. Aquí tenemos un ejemplo utilizando estos caracteres y la estructura /* - */:

```
if (a == 2)
{
        return TRUE;                    // caso especial
}
else
{
         return isPrime(a);             /* otro comentario */
}
```

3.3 Declaraciones

3.3.1 Número de declaraciones por línea

Se recomienda una declaración por línea ya que mejora los comentarios. En otras palabras:

```
int level;      // nivel de identación
int size;       // tamaño
```

se prefiere sobre:

```
int level, size;
```

No debemos poner diferentes tipos en la misma línea. Por ejemplo:

```
int foo, fooarray[]; //Evitar
```

3.3.2 Inicialización

Debemos intentar inicializar las variables locales donde son declaradas. La única razón para no inicializar una variable donde es declarada es si el valor inicial depende de algún cálculo que tiene que ocurrir antes.

3.3.3 Situación

Ponemos las declaraciones sólo al principio de los bloques. No debemos esperar a declarar variables hasta que son usadas por primera vez; puede confundir al programador y estorbar la portabilidad del código dentro del ámbito.

```
void myMethod()
{
  int int1 = 0;                    // comienzo de bloque

  if (condition)
  {
        int int2 = 0;              // comienzo de bloque if
        ...
  }
}
```

La única excepción a esta regla son los indexados para los bucles, que en C# pueden ser declarados en la sentencia for:

```
for (int i = 0; i < maxLoops; i++) { ... }
```

Debemos evitar las declaraciones locales que oculten las declaraciones de nivel superior. Por ejemplo, no debemos declarar el mismo nombre de variable en un bloque interno:

```
int count;

myMethod() {

  if (condition) {

    int count; // Evitar

  }

}
```

3.4 Sentencias

3.4.1 Sentencias simples

Cada línea debe contener, como máximo, una sentencia. Por ejemplo:

```
argv++;          // Correcto
argc++;          // Correcto
argv++; argc--;  // Evitar
```

3.4.2 Sentencias compuestas

Las sentencias compuestas son sentencias que contienen listas de sentencias encerradas entre llaves (“{sentencias}”). Se ilustrarán ejemplos en las siguientes secciones.

- Las sentencias encerradas deben identarse uno o más niveles que la sentencia compuesta.
- La llave (‘{‘) de apertura debe empezar una nueva línea a continuación de la que empieza la sentencia compuesta; la llave de cierre (‘}’) debe empezar una nueva línea y estar identada con el principio de la sentencia compuesta.

- Las llaves se usan alrededor de todas las sentencias, incluso para sentencias simples, cuando éstas forman parte de una estructura de control como una sentencia if-else o for. Esto hace más fácil la adición de sentencias sin introducir errores debido al olvido de las llaves.

Las únicas excepciones a esta regla serán para los métodos get y set de las clases de acceso a datos descritos en el apartado 2.4.4.

3.4.3 Sentencias de retorno

Una sentencia de retorno no deberá usar paréntesis a menos que el valor de retorno sea más obvio de esta forma. Por ejemplo:

```
return;
return myDisk.size();
return (size ? size : defaultSize);
```

3.4.4 Sentencias *if, if-else, if else- if else*

Las sentencias de tipo *if-else* deberán tener la siguiente forma:

```
if (condition)
{
  statements;
}

if (condition)
{
  statements;
}
else
{
  statements;
}


if (condition)
{
  statements;
}
else if (condition)
{
  statements;
}
else
{
  statements;
}
```

Las sentencias *if* siempre usan llaves. Debemos evitar el siguiente caso:

```
if (condition) //Evitar, se han omitido las llaves {}!
  statement;
```

Aunque es perfectamente válido a nivel de código, puede llevar a confusión y a la introducción de errores.

3.4.5 Sentencias *for*

Una sentencia *for* deberá tener la siguiente forma:

```
for (initialization; condition; update)
{
  statements;
}
```

Una sentencia *for* vacía, en la cual todo el trabajo se hace en las cláusulas de inicialización, condición y actualización, deberá tener la siguiente forma:

```
for (initialization; condition; update);
```

Cuando usamos el operador como en las cláusulas de inicialización o actualización de una sentencia *for*, debemos evitar la complejidad de usar más de tres variables. Si es necesario, debemos usar sentencias separadas antes del bucle *for*, para la cláusula de inicialización, o al final del bucle, para la cláusula de actualización.

3.4.6 Sentencias *while*

Una sentencia *while* deberá tener la siguiente forma:

```
while (condition)
{
    statements;
}
```

Una sentencia *while* vacía deberá tener la siguiente forma:

```
while (condition);
```

3.4.7 Sentencias *do-while*

Una sentencia *do-while* deberá tener la siguiente forma:

```
do
{
  statements;
}
while (condition);
```

3.4.8 Sentencias *switch*

Una sentencia *switch* deberá tener la siguiente forma:

```
switch (expression)
{
      case constant-expression:
            statement
            break;

      case constant-expression:
           statement
           /* continúa sin salto */

            [default:
                  statement
                   jump-statement]
}
```

Cada vez que un *case* no incluye una sentencia *break*, debemos añadir un comentario donde normalmente iría la sentencia *break*. Esto se ve en el ejemplo de código anterior con el comentario "/* continúa sin salto */". En cualquier caso se deberá evitar este tipo de construcción.

Toda sentencia *switch* deberá incluir un valor *default*. El *break* en el *case* por defecto es redundante, pero evita un error de caída si añadimos después otro *case*.

3.4.9 Sentencias *try-catch*

Una sentencia *try-catch* deberá tener la siguiente forma:

```
try
{
        statements;
}
catch (Exception e)
{
        statements;
}
```

Una sentencia *try-catch* también puede ir seguida de un bloque *finally*, que se ejecuta sin importar si se ha completado con éxito o no el bloque try:

```
try
{
        statements;
}
catch (Exception e)
{
        statements;
}
finally
{
        statements;
}
```

3.5 Espacios en blanco

3.5.1 Líneas en blanco

Las líneas en blanco mejoran la lectura separando secciones de código que están relacionadas lógicamente.

Siempre se deberán usar dos líneas en blanco en las siguientes circunstancias:

- Entre secciones de un fichero fuente.
- Entre definiciones de clases e interfaces.

Siempre se deberá usar una línea en blanco en las siguientes circunstancias:

- Entre métodos.
- Entre las variables locales de un método y su primera sentencia.
- Antes de un bloque de comentarios o un comentario simple.
- Entre secciones lógicas dentro de un método para mejorar su lectura.

3.5.2 Espacios en blanco

Los espacios en blanco deberán usarse en las siguientes circunstancias:

- Una palabra clave seguida por un paréntesis deberían estar separados por un espacio en blanco:

```
while (true)
{
```

```
        ...
    }
```

- No se deberá usar un espacio en blanco entre un nombre de método y su paréntesis de apertura. Esto ayuda a distinguir las palabras clave de las llamadas a métodos.
- Después de las comas en una lista de argumentos debe aparecer un espacio en blanco.
- Todos los operadores binarios excepto "." Deberían estar separados de sus operandos por espacios. Los espacios en blanco nunca deben separar los operadores unarios como incremento ("++"), y decremento ("—") de sus operadores. Por ejemplo:

```
a += c + d;
a = (a + b) / (c * d);
while (d++ = s++)
{
    n++;
}
```

- Las expresiones de una sentencia deberán estar separadas por espacio. Ejemplo:

```
for (expr1; expr2; expr3)
```

3.6 Convenciones de nombres

3.6.1 Clases

Se deberá usar Pascal-Casing para el nombrado de clases. Debemos intentar mantener los nombres de clases simples y descriptivos. Debemos usar palabras completas y evitar acrónimos y abreviaturas (a menos que la abreviatura se use muy ampliamente como URL o HTML). Ejemplo:

```
public class HelloWorld
{
    ...
}
```

3.6.2 Métodos

Los métodos deberán ser verbos. Se deberá usar Pascal-Casing para su nombrado. Ejemplo:

```
public class HelloWorld
{
    void SayHello(string name)
    {
        ...
    }
}
```

3.6.3 Variables y parámetros

Se usará Camel-Casing para el nombrado de variables y parámetros. Los nombres de variables deberán ser cortos y llenos de significado. La elección de una variable debería ser mnemónica, es decir, diseñada para indicar al observador casual su utilización. Se deben evitar los nombres de variable de un sólo carácter, excepto para variables temporales. Algunos nombres comunes de este tipo de variables son: i, j, k, m, y n para enteros. Ejemplo:

```
public class HelloWorld
{
```

```
        int totalCount = 0;
        void SayHello(string name)
        {
               string fullMessage = "Hello " + name;
        }
}
```

3.6.4 Constantes

Los nombres de variables constantes de clases y las constantes ANSI deberán escribirse todo en mayúsculas con las palabras separadas por subrayados ("_"). Se deberían evitar las constantes ANSI para facilitar la depuración. Ejemplo:

```
public static int MIN_WIDTH = 4;
public static int MAX_WIDTH = 999;
public static int GET_THE_CPU = 1;
```

3.7 Hábitos de programación

No debemos hacer pública ninguna variable de instancia o de clase sin una buena razón. A menudo las variables de instancia no necesitan ser asignadas/consultadas explícitamente. Normalmente, esto sucede como efecto lateral de llamadas a métodos. Un ejemplo apropiado de una variable de instancia pública es el caso en que la clase es esencialmente una estructura de datos, sin comportamiento.

3.7.1 Referencias a variables y métodos de clase

Evitar usar un objeto para acceder a una variable o método de clase (*static*). Usar el nombre de la clase en su lugar. Por ejemplo:

```
classMethod();             //correcto
AClass.classMethod();      //correcto
anObject.classMethod();    //Evitar
```

3.7.2 Constantes

Las constantes numéricas (literales) no se deben codificar directamente, excepto -1, 0 y 1, que pueden aparecer en un bucle *for* como contadores.

3.7.3 Asignaciones de variables

Evitar asignar el mismo valor a varias variables en la misma sentencia. Es difícil de leer. Ejemplo:

```
fooBar.fChar = barFoo.lchar = 'c'; // ¡EVITAR!
```

No usar el operador de asignación en un lugar donde se pueda confundir con el de igualdad. Ejemplo:

```
if (c++ = d++) // ¡EVITAR!
{
  ...
}
```

se debe escribir:

```
if ((c++ = d++) != 0)
```

```
{
  ...
}
```

No debemos usar asignaciones embebidas como un intento de mejorar el rendimiento en tiempo de ejecución. Ese es el trabajo del compilador. Ejemplo:

```
d = (a = b + c) + r; // ¡EVITAR!
```

Debería escribirse como:

```
a = b + c;
d = a + r;
```

3.7.4 Paréntesis

En general es una buena idea usar paréntesis en expresiones que implican distintos operadores para evitar problemas con el orden de precedencia de los operadores. Incluso si parece claro el orden de precedencia de los operadores, podría no ser así para otros. No se debe asumir que otros programadores conozcan el orden de precedencia.

```
if (a == b && c == d)      // Evitar
if ((a == b) && (c == d)) // Correcto
```

3.7.5 Variables de retorno

Debemos intentar hacer que la estructura de nuestro programa se corresponda con nuestra intención. Por ejemplo:

```
if ( booleanExpression)
{
      return true;
}
else
{
      return false;
}
```

debería escribirse:

```
return booleanExpression;
```

De forma similar,

```
if (condition)
{
      return x;
}
return y;
```

debería escribirse como:

```
  return (condition ? x : y);
```

3.7.6 Expresiones antes de '?' en el operador condicional

Si una expresión contiene un operador binario antes de ? en el operador ternario ?:, se debe colocar entre paréntesis. Ejemplo:

```
(x >= 0) ? x : -x;
```

3.7.7 Comentarios especiales

Debemos usar XXX, en un comentario para indicar que algo tiene algún error pero funciona. Usar la etiqueta *FIXME* para marcar algo que tiene algún error y no funciona.

4. ESTÁNDAR DE NOMBRADO DE BASE DE DATOS

4.1 Idioma a utilizar

Para el nombrado de todos los elementos de la base de datos, el idioma a utilizar será el inglés**,** salvo que se especifique de manera explícita lo contrario**.**

4.2 Convenciones de nombrado de tablas

4.2.1 Nombrado de tablas de entidad

El nombrado de las tablas de entidad dentro de la base de datos seguirá la siguiente codificación:

- Todos los nombres comenzarán por tres letras descriptivas del módulo o ámbito de aplicación de la tabla. La primera de estas tres letras será mayúscula y las otras dos restantes serán minúsculas. Los prefijos generales predefinidos con una descripción de los mismos se encuentran en el apartado 4.2.3 de este documento. En el caso de que el ámbito de aplicación de la tabla a nombrar difiera de los que aparecen en este apartado se definirá un nuevo prefijo y se añadirá a los existentes.
- En el caso de que la tabla forme parte del desarrollo de una sección en concreto dentro de un módulo, o forme parte de un conjunto de tablas relacionadas entre sí dentro de un ámbito de aplicación de la base de datos, se colocarán después del prefijo anterior y separadas por un guion bajo '_' tres letras descriptivas de la sección. La primera de estas letras será mayúscula y las otras dos restantes serán minúsculas. Si la tabla no tuviese relación con otras dentro del ámbito definido por el prefijo inicial estas tres letras no son necesarias.
- A continuación y separado por un guion bajo '_' se colocará un nombre descriptivo en plural del contenido de la tabla. Se deberá usar Pascal-Casing para este el nombrado. Se deberá intentar mantener los nombres simples y descriptivos. Se deberán usar palabras completas y evitar acrónimos y abreviaturas (a menos que la abreviatura se use muy ampliamente como ID, URL o HTML, en cuyo caso puede escribirse en mayúsculas).

  Ejemplos:

```
Gen_AdministrationItems
Orh_Emp_Employees
Orh_Trip_TripExpenses
```

4.2.2 Nombrado de tablas de relación

Las tablas de relación comenzarán con un prefijo 'R_'.

A continuación, y en singular, se colocarán utilizando Pascal-Casing el nombre descriptivo de las tablas de entidad que relaciona, pudiendo reducirse si es muy largo alguno de ellos.

De este modo, para relacionar las tablas 'Orh_Trip_TripRequests' y 'Orh_Emp_Employees' se creará la tabla 'R_TripRequest_Employee'.

4.2.3 Nombres de tablas predefinidos

- R: Relación. Tablas de relación.
- Gen: Generic. Tablas relativas a la organización general de la base de datos.

- Orh: Organización y recursos humanos.
- Holidays: Holidays. Tablas relativas a vacaciones.

4.3 Convenciones de nombrado de campos

Una norma establecida es que no puede haber dos campos dentro de una misma base de datos con el mismo nombre. Asimismo, el orden de los campos de una tabla debe seguir un orden lógico, (no tiene sentido colocar, <nombre, teléfono, apellidos, fax...>, sino <nombre, apellidos, teléfono, fax…>) Los nombres de los campos, dependiendo de si pertenecen a una tabla de entidad o de relación, se construyen siguiendo las normas de los siguientes apartados:

4.3.1 Nombrado de Campos de Tablas de Entidad

- Nombrado de claves primarias: Las claves primarias comenzaran con el prefijo 'pk_'. Como finalización del nombre tendrán la terminación 'ID'.
- Nombrado de claves ajenas: Las claves ajenas comenzaran con el prefijo 'fk_'. Si la clave ajena apunta a una clave primaria de otra tabla como finalización del nombre tendrá la terminación 'ID'.
- Nombrado de campos: Todos los campos de una misma tabla tendrán un mismo prefijo que resumirá el nombre descriptivo de la tabla. La primera de estas letras será mayúscula y las restantes serán minúsculas. Por ejemplo, si la tabla es 'Orh_Trip_TripRequest' todos los campos de la tabla en este punto empezarían por 'TripRequest'. Otro ejemplo: Si la tabla es 'Orh_Emp_Employees' los campos de la tabla empezarían por 'Employee' o 'Emp'.

  A continuación se colocará el nombre del campo. Para nombrar los campos de la tabla se utilizarán nombres descriptivos de su significado. Se deberá usar Pascal-Casing para este nombrado. Se deberá intentar mantener los nombres simples y descriptivos. Se deberán usar palabras completas y evitar acrónimos y abreviaturas (a menos que la abreviatura se use muy ampliamente como ID, URL o HTML, en cuyo caso puede escribirse en mayúsculas).

  Ejemplos:

```
fk_EmpCountry
pk_EmpID
EmpName
EmpGender
```

4.3.2 Nombrado de campos de tablas de relación

- Nombrado de claves ajenas: Las claves ajenas comenzarán por 'fk_', a continuación aparecerán, utilizando Pascal-Casing, los nombres descriptivos de la tablas de entidad que se relacionan (pudiendo resumirse si son muy largos), y se finalizarán con el nombre descriptivo de la tabla a la que apuntan seguido de ID.

  Ejemplo:

  En la tabla 'R_Employee_Language' que relaciona las tablas 'Orh_Emp_Employees' y 'Orh_Emp_Languages' se nombrarán de esta manera las claves ajenas:

```
fk_EmpLanEmployeeID
fk_EmpLanLanguageID
```

- Nombrado de campos: Los campos en las tablas de relación se nombrarán mediante Pascal-Casing. Comenzaran con los nombres descriptivos de la tablas de entidad que se relacionan (pudiendo resumirse si son

muy largos). A continuación se colocará el nombre del campo. Para nombrar los campos de la tabla se utilizarán nombres descriptivos de su significado. Se deberá intentar mantener los nombres simples y descriptivos. Se deberán usar palabras completas y evitar acrónimos y abreviaturas (a menos que la abreviatura se use muy ampliamente como ID, URL o HTML, en cuyo caso puede escribirse en mayúsculas).

Ejemplos:

En la tabla 'R_Employee_Language' que relaciona las tablas 'Orh_Emp_Employees' y 'Orh_Emp_Languages' los nombres de los campos podrían ser:

```
EmpLanQualification
EmpLanIsDeleted
EmpLanLastUpdate
```

5. BIBLIOGRAFÍA

5.1 Referencias

[R1] Juval Lowy 2003. *C# Coding Standard*. Idesign Inc 2004.

5.2 Referencias web

[W1] http://www.dotnetspider.com /tutorials/bestpractices.aspx

[W2] http://msdn.microsoft.com/

[W3] http://vyaskn.tripod.com/object_naming.htm