Trabajo fin de grado
Grado en Ingeniería de las Tecnologías Industriales

Diseño e implementación de un sistema de información para la gestión del personal hospitalario

Autor: Jose María Gessa Aguilera
Tutor: Jose Miguel León Blanco

Dpto. de Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería

Sevilla, 2020

Trabajo fin de grado
Grado en Ingeniería de las Tecnologías Industriales

Diseño e implementación de un sistema de información para la gestión del personal hospitalario

Autor:
Jose María Gessa Aguilera

Tutor:
Jose Miguel León Blanco
Profesor contratado doctor

Dpto. de Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Trabajo fin de grado: Diseño e implementación de un sistema de información para la gestión del personal hospitalario

Autor: Jose María Gessa Aguilera

Tutor: Jose Miguel León Blanco

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

A mi familia, que siempre ha proporcionado paz y confianza.

A Raquel, que ha hecho esto posible.

A mi abuelo, por su ejemplo.

AGRADECIMIENTOS

Cuando comencé el desarrollo de este trabajo, a finales de 2019, jamás me habría imaginado que las personas que me estaban abriendo una ventana para enseñarme la forma en la que trabajan iban a ser puestas al límite en unos pocos meses. Es por ello por lo que me gustaría agradecer profusamente a todos los que han velado por nosotros durante la pandemia que ha azotado nuestro país en los últimos meses.

También quiero agradecer a mi tutor, Jose Miguel León Blanco, por su dedicación y paciencia, y por haberse sabido adaptar a la tutorización a pesar de las dificultades.

Por último, es importante para mí agradecer a mi familia, a mis amigos y a mi pareja, que han estado a mi lado contra viento y marea desde el momento en el que inicié la aventura de estudiar una Ingeniería. Sin ellos, hubiera sido mucho más difícil.

Jose María Gessa Aguilera

Sevilla, 2020

RESUMEN

El sistema sanitario español presenta de manera habitual algún tipo de turno rotatorio. La correcta asignación de turnos es vital para un funcionamiento correcto, que cubra las necesidades tanto de los pacientes como de los trabajadores. En este trabajo se estudian los métodos de resolución más comunes para los problemas de asignación de empleados, y posteriormente se diseña e implementa un sistema de gestión de la información que provea a los administradores de un hospital español, cuyos datos han sido tomados en base a la realidad, de una herramienta útil que les permita reducir la carga de trabajo que supone cuadrar los horarios de todos los sanitarios, tarea que actualmente se desarrollaba de forma manual, con la consiguiente dificultad añadida. Al fin del proyecto, se habrá desarrollado por completo el sistema de gestión, de forma que pueda utilizarse en un caso real.

ABSTRACT

The Spanish Health System usually organize it's staff following some sort of rotatory schedule. Producing a well structured schedule it's vital in order to cover not only the patient's need, but also the worker's ones. In this work, we study some of the ways to resolve the nurse rostering problem, and later we design and develop an information management system whose goal is to help the people in charge of hospital's staff organize and schedule their personnel, providing them with a powerfull tool that can reduce their work load and agilize some common issues. By the end of this proyect, a web application will have been completely developed, allowing real hospital's administrators to try it.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

NOTACIÓN

NRP	Nurse Rostering Problem
VNS	Variable Neighbourdhood Search
MVC	Modelo Vista Controlador
GUI	Graphical User Interface
CLI	Command-line interface
NUI	Natural user interface
PHP	Hypertext Preprocessor
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
WWW	Wordl Wide Web
SQL	Structured Query Language
RDBMS	Relational Database Management System
CU	Caso de Uso
ERD	Entity Relationship Diagram

1 INTRODUCCIÓN Y OBJETIVO DEL PROYECTO

1.1 Introducción

La asignación de turnos se ha convertido en un asunto relevante en el desarrollo de la actividad económica moderna. Según la VII Encuesta Nacional de Condiciones de Trabajo [1] un 7,2% rotaba entre mañana, tarde y noche. Además, un 0.4% declaraba realizar otro tipo de horario rotativo. La organización de estos horarios tiene un impacto elevado en el personal, donde una mala gestión puede provocar efectos psicológicos, concretados en trastornos de ansiedad o el conocido como síndrome de *Burnout,* además de un mayor absentismo laboral. Estas consecuencias son especialmente importantes en el campo sanitario [2].

Por ello, la correcta asignación de turnos es de vital importancia para generar mejores resultados, y sigue siendo objeto de estudio actualmente.

Por lo tanto, en este proyecto se desarrollará una base de datos, unida con una página web que automatice la organización de empleados a turnos de trabajo. Para conseguir un mayor ajuste con un caso real, se ha trabajado con los datos de una planta de un centro hospitalario. Su proceso administrativo en cuanto a la asignación de turnos sigue siendo manual, con la ayuda de hojas de cálculo. Esto obliga a los responsables de la asignación a dedicar gran parte de su jornada a esta tarea, en la que pueden producirse errores y dificultades para modificar los turnos.

El desarrollo de un sistema de información permitiría a los administradores agilizar las tareas relativas a la organización de sus trabajadores, además de proporcionar, de forma compacta y sencilla, un mejor control de su plantilla y de la información que se requiere para llevar a cabo su labor.

1.2 Objetivo y alcance del proyecto

El objetivo de este proyecto es, por tanto, desarrollar una aplicación utilizando distintas herramientas de programación que se detallarán posteriormente para la asignación de turnos del departamento de enfermería de un hospital. Esta asignación se desarrollará siguiendo los requisitos reales del centro sanitario, y se encargará de automatizar el desarrollo del calendario y su visualización, de forma que tanto el administrador como los empleados puedan obtener de forma sencilla la información sobre cuando deberán desarrollar su trabajo. De forma paralela, desde la aplicación se podrán realizar otras acciones relacionadas con la organización laboral, creando un entorno de trabajo cómodo y ágil para facilitar las labores organizativas. Los detalles sobre estas funcionalidades quedarán descritos en el apartado posterior.

2 PLANTEAMIENTO

Este capítulo versa sobre el problema matemático conocido como "Nurse Rostering Problem", de gran relevancia en la resolución de problemas de asignaciones de turnos. Por otra parte, quedarán expuestas las herramientas que se van a utilizar para el posterior desarrollo del sistema de información.

2.1 El Nurse Rostering Problem

Los problemas de asignación de empleados a turnos han sido estudiados durante más de 40 años, dada su complejidad. Dentro de estos, los que se refieren a la asignación de trabajadores del ámbito sanitario son interesantes por las condiciones específicas de trabajo que se dan en estos centros, donde los turnos son generalmente más complejos que en otros campos y suele haber muchos horarios rotatorios. En 1966, Howell describió por primera vez una serie de tareas cíclicas encaminadas a la resolución de un problema de asignación de turnos [3]. El término "Nurse Rostering Problem", en adelante referido como NRP, ha sido acuñado para el problema general de asignación de turnos. El NRP es clasificado como NP-completo debido a la inclusión de las restricciones que comúnmente aparecen al programar turnos de trabajo. La literatura existente repasa muchos métodos distintos de resolución, desde la asignación manual que realizan algunos responsables hasta el desarrollo de metaheurísticas y la aplicación de algoritmos para su automatización, como queda expuesto a continuación [3],[4].

2.1.1 Formas de resolución

En este apartado se comentarán los métodos de resolución que se han investigado. Hay que tener en cuenta que, dentro de estos acercamientos, la resolución puede variar mucho según las restricciones que se apliquen.

2.1.1.1 Programación lineal entera y mixta

Uno de los trabajos principales basándose en estos métodos de resolución es el desarrollado por Abernathy et al.[5] En este documento se propone la creación de una variable aleatoria, asumiendo la necesidad de que un trabajador sea asignado a un puesto de trabajo en un cierto periodo de tiempo sigue un proceso estocástico. Posteriormente unen estos factores en una función que denominan Utilidad e intentan maximizar su esperanza matemática. Sin embargo, consideran esta función como poco capaz de resolver problemas más realistas. Es, por tanto, un gran acercamiento al problema de organización de turnos, aunque otros trabajos más modernos han avanzado en las técnicas y conseguido adaptar mejor el modelo a casos reales.

En 2004, Isken [6] desarrolla un modelo en el que quedan definido los días trabajados y las horas de inicio de los turnos asignados (tipo tour). Su trabajo tiene como objetivo minimizar el coste de la mano de obra, e incluye una restricción que evalúa la flexibilidad en la hora de inicio de los turnos. Desarrolló este modelo para nueve hospitales estadounidenses, atendiendo a la necesidad de organizar los trabajadores en un entorno con horas y turnos flexibles. Fue usado para comprobar la diferencia entre necesidades de personal al programar horarios más fijos frente a otros más variables, encontrando una diferencia en el coste del 8% de media, siendo mayor en los horarios fijos.

Un enfoque más moderno fue estudiado por Della Croce y Salassa [7], siendo su objetivo el minimizar el número de personal de enfermería freelance que se asignan a los turnos. Esto provoca que el modelo intente asignar adecuadamente los turnos de forma que el gasto en contratar agentes externos, que es el más caro, sea mínimo. Además, añaden aproximadamente 2000 restricciones, atendiendo a las necesidades de un hospital turinés del que se obtuvieron los datos para probar el modelo. Cabe destacar que a su algoritmo de programación lineal entera se le añadieron cuatro restricciones más, que añadían una metaheurística de búsqueda de vecinos. Se desarrolló posteriormente un software usado por el hospital con buenos resultados.

Dado que los casos reales añaden muchas restricciones específicas, es muy habitual combinar la programación matemática con heurísticas que permiten encontrar una solución válida en un tiempo computacional aceptable.

2.1.1.2 Programación por objetivos

Esta forma de resolver problemas multiobjetivo ha sido muy prolífica en el campo de la asignación de turnos, dado que permite modelar distintos objetivos que son, muchas veces, contradictorios. Es una extensión más de la programación lineal entera, donde se usan algunas restricciones de ese problema como objetivos [8].

En este caso separamos restricciones en duras, es decir, aquellas que deben satisfacerse obligatoriamente, y blandas, que son objetivos adicionales que se pueden conseguir. De esta forma podemos afrontar problemas más complejos y que se ajustan más a la realidad. Normalmente los modelos son acompañados por heurísticas que permiten conseguir un resultado más realista y eficiente.

En 2005 Azaiez y Al Sharif [9], aplican la programación por objetivos para crear un software utilizado en un hospital en Arabia Saudí. Su modelo presenta restricciones duras como asegurar el nivel mínimo de personal, impedir turnos de 24 horas seguidos y 4 días consecutivos, asegurar un tiempo de trabajo mensual y otros requisitos de carácter legal. Por otro lado, las restricciones blandas intentan que todo el personal tenga asignado 15 días de trabajo, intentar que tengan más turnos de día que de noche o evitar patrones de un día libre uno trabajando. El programa generado ha sido bien recibido por el personal del hospital, aunque los autores siguen trabajando actualmente para mejorar e incluir nuevas características al software.

También se han desarrollado modelos en los que las preferencias de los trabajadores eran tenidas en cuenta, dado que este tipo de resolución permite asignar esos objetivos como restricciones blandas. Topaloglu y Ozkarahan [10] añaden algunas como el número de horas de trabajo deseadas, asignarle el turno de trabajo preferido o el número de días consecutivos y respetar sus fines de semanas libres pedidos. El modelo fue probado con buenos resultados, pero los autores apuntan el elevado tiempo de computación, que dificultaría encontrar una solución en casos reales con muchos empleados.

Al igual que en el apartado anterior, la combinación de este tipo de resolución con heurísticas y metaheurísticas es muy común.

2.1.1.3 Heurísticas

Las heurísticas permiten obtener soluciones válidas en un tiempo computacional razonable. En muchos casos, las heurísticas desarrolladas surgían directamente de la automatización de la organización manual que se hacía por parte de un encargado. [11]

Okada y Okada [12] desarrollan un programa informático a partir de un algoritmo heurístico que codifica un procedimiento similar al que se realizaba a mano, automatizándolo en el lenguaje Prolog. Aunque no fue utilizado en hospitales, los autores consideran que resultaría en una mejoría dada la experimentación

realizada.

Randhawa y Sitompul [13] también crearon un programa informático con el mismo fin, en el que utilizaban una heurística para generar patrones de turnos y conseguir satisfacer los objetivos tanto del hospital como del personal. A través de una interfaz de usuario es posible modificar datos y añadir nuevas restricciones, así como generar los distintos horarios.

2.1.2 Metaheurísticas

Las metaheurísticas han sido el método más utilizado recientemente para solucionar el NRP, permitiendo ajustarse más a la realidad. Mientras las heurísticas suelen servir a propósitos específicos, las metaheurísticas buscan adaptarse a esos problemas, lo que lleva menos trabajo que desarrollar una heurística particular. Esto, unido al aumento de la capacidad de computación, ha permitido resolver problemas con muchas restricciones y objetivos, con datos reales. [14]. Algunos de estos acercamientos serán expuestos a continuación,

2.1.2.1 Variable Neighbourhood Search

En adelante VNS, este algoritmo es ampliamente combinado con las demás formas de resolución ya que se basa en encontrar el mínimo local de cada "vecindario" de soluciones, y asume que estos óptimos no pueden estar muy alejados el uno del otro, por lo que el cambio y búsqueda de estas soluciones acabará aportando el óptimo global del problema [15]. El VNS combina hallar esos óptimos locales con escapar de ellos mediante el cambio de vecindario. [16] En este trabajo se estudia el NRP mediante este método y encuentran que los cambios de vecindario ayudan al sistema a encontrar soluciones que no se hallarían con una sola población global.

2.1.2.2 Búsqueda tabú

Los algoritmos de búsqueda tabú son usados para hallar el óptimo o un valor cercano eliminando las soluciones ya encontradas, lo cual permite explorar todo el rango de posibilidades existentes más rápidamente. Downsland [17] crea un algoritmo que separa la función de coste en dos partes, y oscila entre ellas para balancear el tiempo de búsqueda de vecinos en cada una. Así, permite reducir la lista tabú, de forma que no se generan pequeños ciclos. Genera dos listas tabúes, una para el intercambio de personal en turnos y otra para la división entre trabajadores de día y noche. Las pruebas demostraron que es un algoritmo fiable, pero no ha sido aplicado en la práctica.

Bester, Nieuwoudt y Van Vuuren [18] mantienen una lista de los movimientos realizados durante la búsqueda, de forma que esas soluciones no fueran consideradas a lo hora de buscar otras mejores. Estos movimientos son clasificados en simples y compuestos. Los simples se basan en cambiar el personal asignado a un turno o el turno asignado a un trabajador. El movimiento compuesto alterna entre estos dos, generando un ciclo horario o antihorario. Este último se probó mucho más eficaz que los otros dos por separado, y se desarrolló un software que usaba esta metodología, aplicado en un hospital en Sudáfrica. Los autores indican que ciertas correcciones manuales son necesarias tras ejecutar el programa, pero son menores.

Bellanti et al. [19] estudiaron un caso real de un hospital turinés, resolviendo el problema mediante este método, y desarrollando un software que se entregó a la administración para organizarse autónomamente. Este algoritmo utilizaba la búsqueda local de forma que intercambiaba mediante cuatro operaciones la asignación de turno de noche a personal. La lista tabú fue dimensionada experimentalmente en n=6, lo que proporcionaba el mejor resultado. En esta tabla se indicaba el trabajador que cambiaba su turno de noche, los días en los que lo hacían y la operación que se había realizado para llegar a ese resultado. La solución encontrada es fiable y tiene un buen tiempo computacional, además de mejorar sustanciosamente la organización del hospital, que se realizaba a mano.

2.1.2.3 Algoritmos genéticos

Los algoritmos genéticos se inspiran en la evolución biológica. Para ello, utilizan soluciones generadas aleatoriamente, y eligiendo las mejores se les realizan mutaciones y combinaciones, de forma que esa información es utilizada en las iteraciones para encontrar una solución óptima. En cada una de estas iteraciones algunas de las mejores se guardan y otras varían aleatoriamente, repitiéndose el proceso hasta llegar a los criterios de parada. [20]

En 2004 Aickelin y Downslad [21] mejoran mediante heurísticas su algoritmo genético previamente descrito en el año 2000, resolviendo primero el problema sin las restricciones para encontrar la mejor ordenación de enfermeras, y aplicando posteriormente una heurística que encuentra la mejor solución. Este trabajo presenta una considerable complejidad, dada las limitaciones del algoritmo genético para resolver problemas de optimización. Estos algoritmos convergen hacia una solución, pero tienen problemas en hacer pequeños cambios que mejoren la función objetivo. Es por ello por lo que en este trabajo añade pesos a ciertas características que se han identificado como posibles mejoras del problema.

Bai et al. [22] combinan un algoritmo genético con uno de recocido simulado para conseguir atacar un problema real que había sido probado mediante distintos acercamientos sin resultados satisfactorios. Su algoritmo mutaba cambiando el patrón de turnos de un trabajador elegido aleatoriamente a otro patrón aleatorio pero válido. El rango de soluciones válidas se estabilizaba en torno al 20-30%, mientras el coste se reducía gradualmente con el tiempo. La selección de las soluciones está basada en un proceso de selección estocástico, que compara la probabilidad de un individuo de mejorar la función objetivo y las penalizaciones frente a otro. La combinación de este método de elección junto con el algoritmo de recocido simulado probó ser altamente eficaz frente a otras soluciones.

2.1.2.4 Algoritmo de recocido simulado

Este tipo de metaheurística es una variación del método Metrópolis-Hasting, que propone una analogía entre el recocido de materiales físicos y la optimización de un objetivo en sistemas grandes y complejos. En estas heurísticas, valiéndose de un valor que ha mantenido el nombre de temperatura dada la analogía, los estados varían hacia un vecino siguiendo una función que depende de esa variable, que va decreciendo con el tiempo. En valores de temperatura altos los cambios en la función objetivo son significativos. Cuando va bajando esa temperatura, los cambios son más sutiles y permiten acercarse al óptimo global. [23]

En el ámbito del NRP, además del trabajo de Bai et al. [22] anteriormente mencionado en el apartado 2.1.4.2, cabe destacar el desarrollado por Geyer y Thompson [24] en el que organizan los turnos de enfermería mediante un recocido simulado, pero considerando a todo el personal con el mismo nivel de habilidad.

La siguiente tabla contiene las referencias en las que la resolución del NRP fue aplicada a casos reales, continuando la solución mediante el desarrollo de un software.

Tabla 2-1. Referencias en las que se desarrolla un software

Método de resolución	Referencia
Programación lineal entera y mixta	[7]
Programación por objetivos	[9]
Heurísticas	[12] [13]
Variable Neighbourhood Search	
Búsqueda Tabú	[18]
Algoritmo Genético	[22]
Algoritmo de Recocido Simulado	

2.1.3 Softwares disponibles

En este apartado serán expuestos algunos de los softwares que permiten organizar los turnos de trabajo. No todas se aplican al campo de la sanidad, y son herramientas comerciales que se desarrollan con el propósito de mejorar la eficiencia de las empresas cliente.

2.1.3.1 SERVICEMAX

Este software está orientado hacia asignar técnicos a trabajos concretos, pero entre sus funcionalidades se encuentra la de configurar reglas para asignar automáticamente tareas según cualificación, proximidad al trabajo o necesidad de mano de obra. Sin embargo, no es su única aplicación, ya que se presenta como un sistema de gestión más completo, pudiendo por ejemplo enviar nuevas órdenes de trabajo o encontrar el personal más cercano geográficamente a una tarea reciente.[25]

2.1.3.2 SYNCHROTEAM

Con este software, que tampoco trata directamente el NRP, se permite asignar automáticamente trabajos por hacer en la franja horaria que más conviene hacerlo, así como filtrar al personal por cualificación para poder asignar a la persona adecuada para ello. También permite muchas otras opciones de gestión: comunicación interna, geolocalización de transportes, base de datos de inventario, etc. [26]

2.1.3.3 KVP HIBERUS

Diseñada de forma modular, en su versión más básica esta aplicación permite asignar turnos según distintos factores (cualificación, disponibilidad, horas de trabajo…) y controlar el horario laboral. Se pueden añadir distintas funcionalidades, como control de almacén, contabilidad o facturación entre otras muchas. [27]

2.1.3.4 ATURNOS

Este software online permite organizar automáticamente los turnos, atendiendo a unos criterios previamente establecidos por el usuario. Se pueden introducir algunas normas, como días de descanso después de turnos nocturnos o vacaciones. Es interesante dado que se trata de una web, sin necesidad de instalar ningún programa. Según su página web, es usado por el hospital Nuestra Señora del Rosario de Madrid. [28]

Como se ha tratado anteriormente, el problema tratado en este trabajo se aleja de las condiciones necesarias para su optimización mediante métodos matemáticos, ya que al pertenecer a una institución pública no existe plena libertad de contrataciones, despidos u otras alteraciones de personal que generalmente conforman la resolución del NRP. Es por ello por lo que, aunque inicialmente se planteó la opción de realizar el modelado y desarrollo de algoritmos compatibles con el problema descrito, se ha optado por la creación de un sistema informático que aplique las normas ya existentes en el hospital para facilitar la asignación de empleados. Sin embargo, podría estudiarse una ampliación del trabajo en la que se desarrollase este problema atendiendo a unas condiciones más laxas.

A continuación, por tanto, quedan detalladas las herramientas que se han aplicado para la creación del sistema de gestión de empleados.

2.2 La arquitectura Modelo Vista Controlador

La arquitectura de Modelo Vista Controlador, en adelante MVC, fue descrita como una mejora de los programas codificados en la época, como un medio para facilitar la programación de sistemas que requerían la interacción con el usuario, además de compartimentar el código de forma que su modificación y escalabilidad fuera más sencilla .La unificación de la interfaz con las funcionalidades no permitía que los habituales cambios en las aplicaciones fueran resueltos de forma cómoda, ya que se requería un estudio extenso del código y su funcionamiento. Mediante esta arquitectura, se consigue mejorar no solo la programación física sino el modelo conceptual [29].

El MVC divide el software en tres componentes:

-El Modelo: Consiste en un conjunto de clases que modelan el problema a resolver. Estas clases están por tanto definidas para atender a las funcionalidades requeridas, y serán útiles mientras estas no cambien. El Modelo no interactúa directamente con el usuario, sino que recibe mensajes del Controlador y envían resultados a la Vista.

-La Vista: Es la encargada de mostrarle datos al usuario. Las vistas conforman por tanto la interfaz con la que interactuamos, y se comunica por el modelo para saber qué tiene que mostrar. Algunas interfaces que puede utilizar son la Interfaz Gráfica de Usuario o GUI, la Interfaz de Línea de Comandos o CLI, y la Interfaz Natural de Usuario o NUI. Estos tipos representan distintas formas de comunicación con el usuario.

-El Controlador: Coordina las relaciones entre Vistas y Modelos, recibiendo los datos necesarios e interactuando con estos dos, generando respuestas. Es una suerte de intermediario entre sus compañeros, y la clase más “consciente” del programa en general.

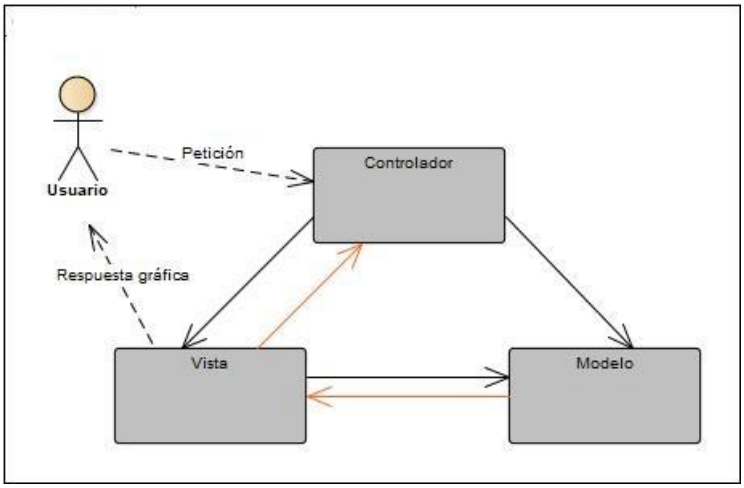


Figura 2-1. Arquitectura MVC

2.3 El entorno de desarrollo

2.3.1 Hypertext Preprocessor (PHP)

PHP es un lenguaje de programación de código abierto creado en 1995 por Rasmus Lerdof [30]. Los programas codificados en PHP son interpretados por el servidor, y el cliente obtiene la respuesta de este. Mucha de su sintaxis es heredada de lenguajes como C, Java y Perl, y permite interactuar con HTML muy estrechamente: PHP puede ser insertado en páginas de HTML, y este último puede interpretar código escrito en PHP. Actúa como un conector entre las páginas web y las bases de datos.

Este lenguaje es un módulo oficial del servidor HTTP de Apache, lo que permite desarrollar el código directamente en el servidor Web, permitiendo una mayor velocidad de procesamiento, y un mantenimiento más sencillo.

2.3.2 HyperText Markup Language (HTML)

El lenguaje HTML fue descrito por primera vez por Tim Berners Lee en 1991 [31], y es considerado uno de los lenguajes web más importante, participando activamente en el desarrollo de la World Wide Web (en adelante WWW). Creado para el intercambio de documentos científicos, fue estandarizado por la WC3 en octubre de 1994, cuyo objetivo principal fue el de encabezar el desarrollo de la WWW. Desde entonces ha sido acogido por todos los navegadores.

Divide sus categorías en cómo se generan las páginas en el servidor, pudiendo ser de forma dinámica, donde usuarios distintos pueden recibir informaciones diferentes, o estáticas, que sin un procesamiento previo muestran la misma información a todos los usuarios.

Consta de una serie de etiquetas, mayoritariamente pares de apertura y cierre, que definen el código. Es muy sencillo trabajar con él, dado que solo se necesita un editor de texto plano para escribir el código y un navegador para interpretarlo y visualizarlo. La mayoría de los entornos de programación desarrollados para facilitar su uso se basan simplemente en generar plantillas o automatizar las etiquetas.

Como se ha comentado anteriormente, su relación con PHP es muy estrecha, y la combinación de estos dos lenguajes está presente en muchos desarrollos actuales de páginas web.

2.3.3 MySQL

El Structured Query Language (SQL) es un lenguaje de programación dedicado al desarrollo de bases de datos. Está estandarizado por el Instituto Nacional Estadounidense de Estándares y por la Organización Internacional de Normalización. Esto permite trabajar con todos los entornos de base de datos mayoritarios, ya que a pesar de que hay algunas diferencias, el estándar es SQL [32]. Permite, con una serie de funciones predefinidas, administrar los datos almacenados.

MySQL es un sistema gestor de bases de datos desarrollado por la empresa MySQL AB, transferido posteriormente a Sun Microsystems y por último a Oracle. Es de tipo RDBMS (Relational Database Management System), diseñado específicamente para bases de datos relacionales, donde la información queda recogida en tablas, que tienen filas y columnas, y que se relaciona con otras tablas mediante claves primarias y externas. Es el modelado más común, y el término base de datos presupone hoy en día un modelo relacional [33]. MySQL, por tanto, se encarga de las operaciones con estas tablas y con sus relaciones.

Es Open Source, y sus laxos términos de licencia y uso han permitido que se haga con una gran base de usuarios, aunque en sus inicios no tuvo mucho éxito. Hoy en día es el segundo motor de bases de datos más popular, solo por detrás de Oracle, según el ranking de DB-engines [34].

2.3.4 Enterprise Architect

Enterprise Architect es un software desarrollado por SPARX systems para el modelado de proyectos multidisciplinares. Este programa permite diseñar un sistema de información completo, desde la captura de requisitos hasta la simulación de su funcionamiento. En este proyecto, ha sido utilizado para todos los pasos del diseño de la base de datos, y la generación de los diagramas necesarios. [35]

2.3.5 Aptana Studio

Aptana Studio es un entorno de desarrollo integrado desarrollado por la compañía homónima. Es de licencia libre y permite el desarrollo de aplicaciones web en una amplia variedad de lenguajes de programación [36]. Ha sido el entorno escogido para el desarrollo del código perteneciente a este proyecto.

2.3.6 Javascript

Javascript es un lenguaje de programación desarrollado en la década de los noventa por Brendan Eich, en respuesta a la necesidad de crear un lenguaje que pudiera ejecutarse directamente en el navegador, con el fin de acortar los tiempos de procesamiento. Javascript es incluido fácilmente en las páginas HTML, y el código puede ser interpretado sin necesidad de compilarlo [37]. Su estandarización comenzó en 1997 a través de EMAScript, que continúa actualizando las versiones anualmente, siendo la más reciente la ES10 de 2019. [38]

2.3.7 phpMyAdmin

Creado para administrar bases de datos en la web, este software libre permite manejar el código SQL y los sistemas de información desarrollados en MySQL. Contempla diversas funcionalidades para la exportación e importación de datos y una interfaz gráfica sencilla para la administración de la base de datos. [39]

2.3.8 EasyPHP Devserver

Mediante esta aplicación se puede establecer un servidor local para realizar las pruebas necesarias de la aplicación desarrollada, además de incluir otras funcionalidades que permiten, por ejemplo, probar código PHP. Permite varios lenguajes de programación y gestores de bases de datos distintos. [40]

3 ANÁLISIS DE REQUISITOS

Este capítulo versa sobre las necesidades que se han identificado en el hospital para poder mejorar la gestión de los empleados, la asignación de turno y modernizar el sistema actual.

3.1 Requisitos de la organización

Como ha quedado expuesto anteriormente, la turnificación es algo común en el sistema sanitario. En este caso, existen distintos tipos de empleados según su horario y tipo de contrato. En concreto, por tipo de contrato, se dividen en 3 categorías: Fijos, Interinos y Eventuales. Estos profesionales desarrollan distintos tipos de turnos con sus correspondientes secuencias, que son las siguientes:

- Fijo de mañana: Trabajan un total de 8 horas, entre las 7:00 y las 15:00. Su secuencia es la más común en el mundo laboral, desarrollan su labor de lunes a viernes y descansan fines de semanas y festivos según el calendario laboral.
- Fijo de tarde: De 15:00 a 20:00, por un total de 5 horas. Trabajan de lunes a viernes, pero este último día desarrollan un turno de 8:00 a 20:00. Tanto en este tipo como en el anterior se suelen asignar trabajadores eventuales.
- Diurno de 12 horas: El más habitual actualmente, se divide semanalmente entre dos empleados, de forma que una semana uno trabaja 3 días y la siguiente 2, complementado por su compañero. Este turno es deficitario, puesto que la semana que trabaja dos días faltan 11 horas para alcanzar las 35 que tiene la jornada. Además, cuando le tocan 3 turnos trabaja una hora de más. Estos desajustes se compensan mediante un sistema de deuda horaria, aplicando turnos extra cuando el servicio tenga necesidades, y de esta forma completar la jornada laboral.
- Rotatorio: Consta de 12 horas diarias, de 8 de la mañana a 8 de la tarde o viceversa, realizando su labor a lo largo del día o de la noche. Sigue la siguiente secuencia: Día, noche, tres descansos, día, noche, 4 descansos. Es necesario que haya dos personas de día y dos de noche durante los 365 días del año, requisito que se suple con 11 trabajadores. Esta secuencia puede ser iniciada en cualquier momento. Suele asignarse a personal interino o fijo.

Como detalle, existía un turno fijo nocturno que está en extinción, habiendo sido sustituido por el rotatorio.

Las vacaciones se calculan según el cómputo horario y la antigüedad. Sin embargo, los eventuales no tienen mes de vacaciones, y se les incluye en el cómputo mensual. En el hospital se calculó que asignando a dos eventuales a turnos de 12 horas se obtiene el cómputo exacto ya que al darles las vacaciones no se producen desajustes.

Las bajas laborales no afectan al cómputo horario, y son suplidas por cualquier profesional disponible con deuda horaria, que copia el turno del empleado de baja.

Actualmente trabajan 13 personas en los turnos de enfermería, de forma que se cubren las necesidades de empleados necesarias. Por turnos, se divide en un empleado fijo de mañana, 2 empleados que realizan turnos diurnos de 12 horas, y 11 en el rotatorio.

El hospital requiere, por tanto, de un sistema que automatice el proceso de asignación, que como se puede ver es complejo e implica muchos cálculos y variaciones prácticamente diarias. El objetivo es simplificar y unificar la tarea de administrar los empleados.

Aun así, existen más requisitos que no quedarán cubiertos por la aplicación, dado que es posible solicitar reducciones de jornada, que se suplen mediante contrataciones adicionales. Tampoco se ha incluido el turno de tarde en el algoritmo, dado que es asignado a trabajadores eventuales y se hace manualmente según la experiencia de los administradores.

3.2 Requisitos Funcionales

El software desarrollado, por tanto, tendrá como objetivo último la gestión de los empleados. Para ello, necesita una serie de requisitos funcionales requeridos para el correcto funcionamiento del proceso:

- Iniciar y cerrar sesión: El acceso al sistema debe quedar restringido a los usuarios que el administrador decida, y por lo tanto es necesario poder Ingresar en la web mediante unas credenciales seguras.
- Crear usuarios: El administrador necesita dar de alta nuevos usuarios para que los empleados puedan acceder al sistema
- Modificar usuarios: Tanto la persona responsable como los empleados tendrán disponible la opción de modificar sus credenciales de acceso.
- Ingresar empleados: Es un requisito indispensable la posibilidad de crear nuevos registros cuando se produzcan altas en el hospital.
- Eliminar empleados: Si fuera necesario, el administrador podrá eliminar empleados de la base de datos.
- Modificar datos: La persona a cargo debe ser capaz de modificar los datos de empleado existentes, para modificar errores o actualizar información.
- Visualizar datos: Será posible la visualización de la información referente a los empleados que existe en un momento concreto.
- Generar un nuevo cuadrante: Esta funcionalidad es uno de los motivos más importantes para el desarrollo del sistema de información, y permitirá al administrador generar un cuadrante de trabajo para todos los empleados automáticamente.
- Visualizar cuadrante: Tanto los empleados como el administrador podrán visualizar los turnos de trabajo que han sido asignados.
- Solicitar bajas: Todas las personas que accedan al sistema deben poder solicitar días de baja, para cubrir posibles incidencias.
- Cubrir bajas: El administrador podrá modificar los turnos de alguno de los empleados para suplir las bajas que sean necesarias.

La recogida de estos requisitos funcionales da lugar al siguiente diagrama:

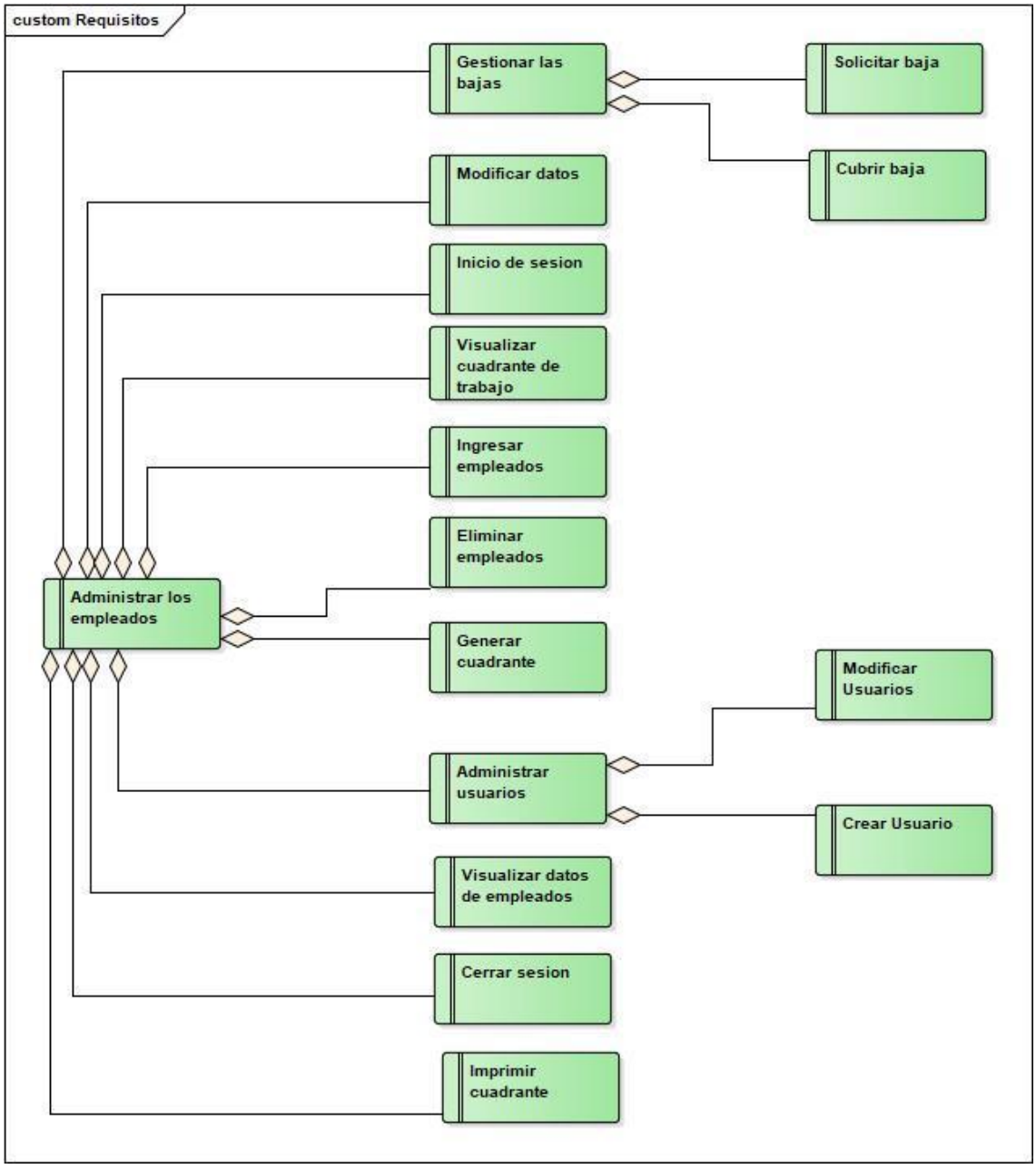


Figura 3-1. Diagrama de Requisitos

La escalabilidad del proyecto permitirá incluir nuevos usos si se prueban necesarios para mejorar o adaptar el trabajo. Estas acciones quedarán descritas detalladamente a posteriori.

3.3 Requisitos del sistema

3.3.1 Actores

Los actores refieren distintos roles que se pueden asumir al interactuar con el sistema. Pueden ser tanto personas físicas como hardware, aunque en nuestro caso todos serán personas. Tendremos 2:

- Administrador: Persona encargada de la gestión del personal, y responsable de la organización. Tendrá acceso a funcionalidades organizativas propias de su trabajo.
- Empleado: Los trabajadores del hospital, que podrán interactuar con el sistema para enviar o recibir información, pero no se autogestionan.

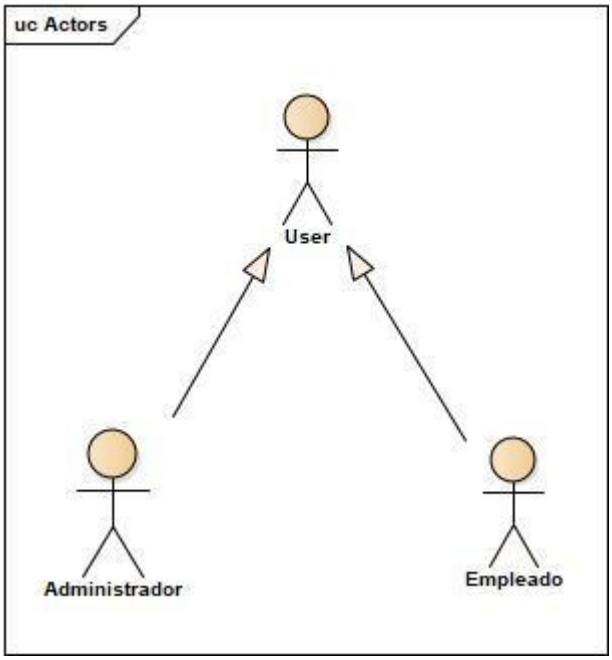


Figura 3-2.Diagrama de Actores

3.3.2 Casos de uso

Los casos de uso nos proporcionan una descripción detallada de la interacción entre los actores y el sistema, de forma que sirvan como una guía para el diseño de las funcionalidades. En la siguiente imagen se puede observar un esquema de los casos de uso que se han identificado como fruto del estudio de los requisitos, donde se aprecia la relación que presenta cada uno de ellos con los actores del sistema.

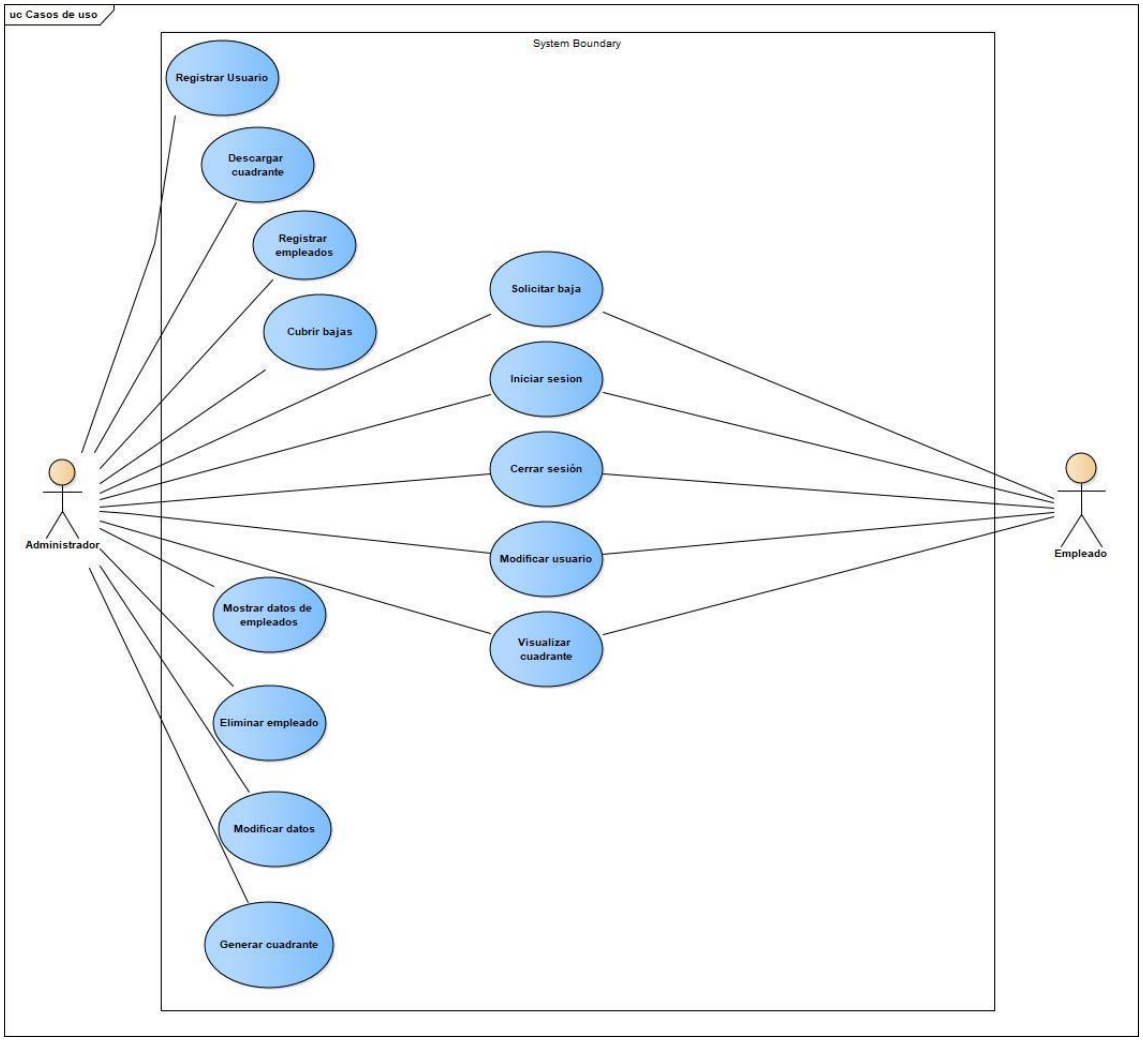


Figura 3-3. Diagrama de Casos de Uso

A continuación, se exponen los casos de uso mediante la tabla ejemplo disponible en la Guía para la redacción de casos de uso de la Junta de Andalucía [41].

CU-XXX	Nombre del caso de uso	
Actores	Actores involucrados	
Precondiciones	En el caso de que existan, las condiciones que han de darse para acceder al caso de uso correspondiente	
Descripción	Breve descripción de la funcionalidad	
Secuencia Normal	**Paso**	**Acción**
	1	
	2	
	3	

	4	
	5	
	6	
	7	
Postcondición	Condiciones que se dan al finalizar el caso de uso	
Excepciones	Paso	Acción
Comentarios		

Tabla 3-1. Ejemplo de Caso de Uso

Los casos de uso también quedan definidos por su diagrama de actividad, en el que se muestran de forma gráfica las interacciones que se generan entre los actores y el sistema

3.3.2.1 Caso de uso: Inicio de sesión

Mediante esta funcionalidad, tanto los administradores como los empleados iniciarán sesión en el sistema, y posteriormente serán redirigidos hacia la página de inicio correspondiente según sean un actor u otro. Para ello, el actor introducirá su usuario y contraseña en el formulario correspondiente, que serán validados por el sistema. En el caso en el que los datos no fueran correctos, se redirige de nuevo hacia el formulario.

CU-001	Inicio de sesión	
Actores	Administrador – Empleado	
Precondiciones	El actor se encuentra en la página de inicio, cargada previamente, y está dado de alta en el sistema.	
Descripción	El sistema permitirá el acceso a más funcionalidades una vez se ejecuten los pasos descritos a continuación	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario solicita el ingreso en la web
	2	El sistema muestra el formulario de inicio de sesión
	3	El usuario introduce datos de acceso, tanto usuario como contraseña
	4	El sistema comprueba los datos de acceso

	6	Si son correctos, el sistema comprueba el tipo de usuario
	7	El sistema redirige al actor a la pantalla correspondiente
Postcondición	El sistema inicializa la sesión y almacena las variables necesarias para el manejo de la información.	
Excepciones	Paso	Acción
	4	Si son incorrectos, el sistema muestra un mensaje y redirige al formulario de inicio de sesión
Comentarios		

Tabla 3-2. CU-001:Inicio de sesión

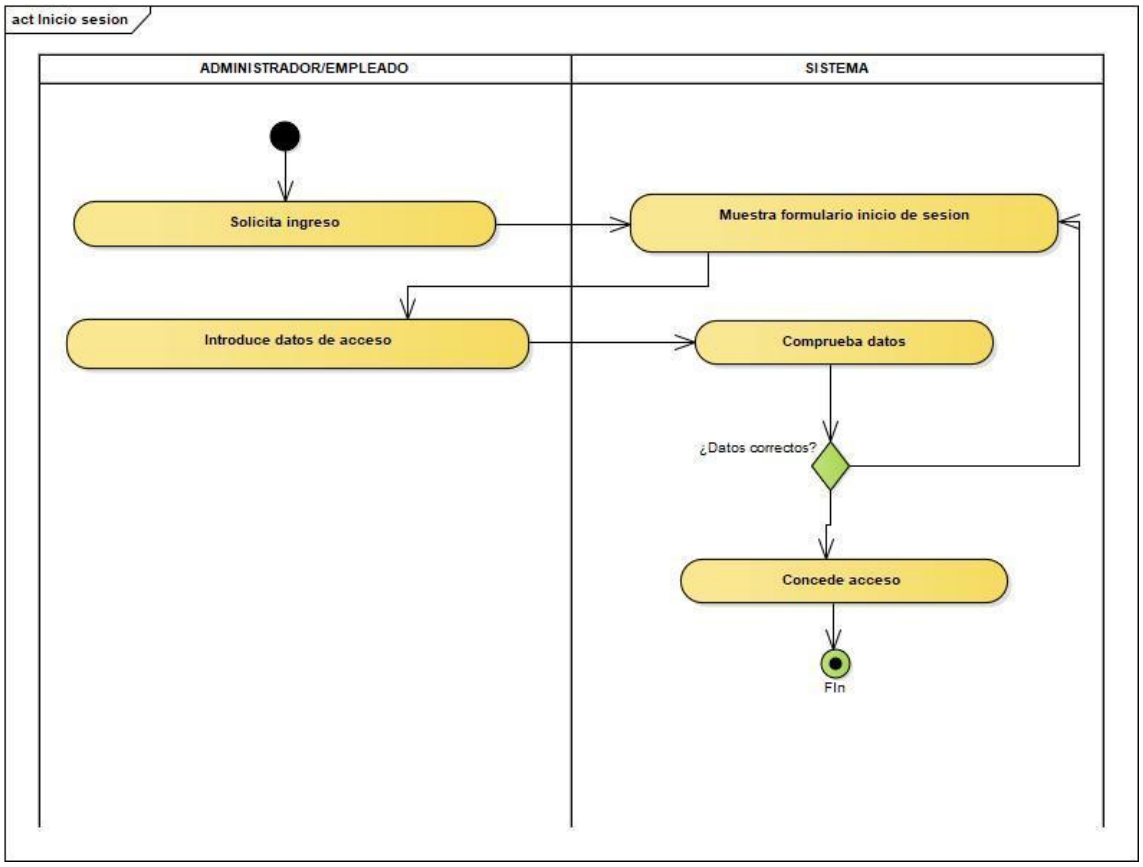


Figura 3-4. CU-001: Diagrama de Actividad

3.3.2.2 Caso de uso: Registrar empleado

Mediante este caso de uso el administrador puede almacenar en la base de datos un nuevo empleado, de forma que quede registrado en la organización.

CU-002	Registrar empleado	
Actores	Administrador	
Precondiciones	El administrador ha iniciado sesión en el sistema	
Descripción	El administrador registrará un empleado en el sistema.	
Secuencia Normal	**Paso**	**Acción**
	1	El administrador solicitar generar un cuadrante
	2	El sistema muestra el formulario
	3	El actor introduce los datos
	4	El sistema comprueba si los datos son válidos
	5	El sistema graba los datos
	6	El sistema muestra un mensaje de confirmación
Postcondición		
Excepciones	Paso	Acción
	4	Si los datos no son válidos, el sistema vuelve al formulario
Comentarios		

Tabla 3-3. CU-002: Registrar Empleado

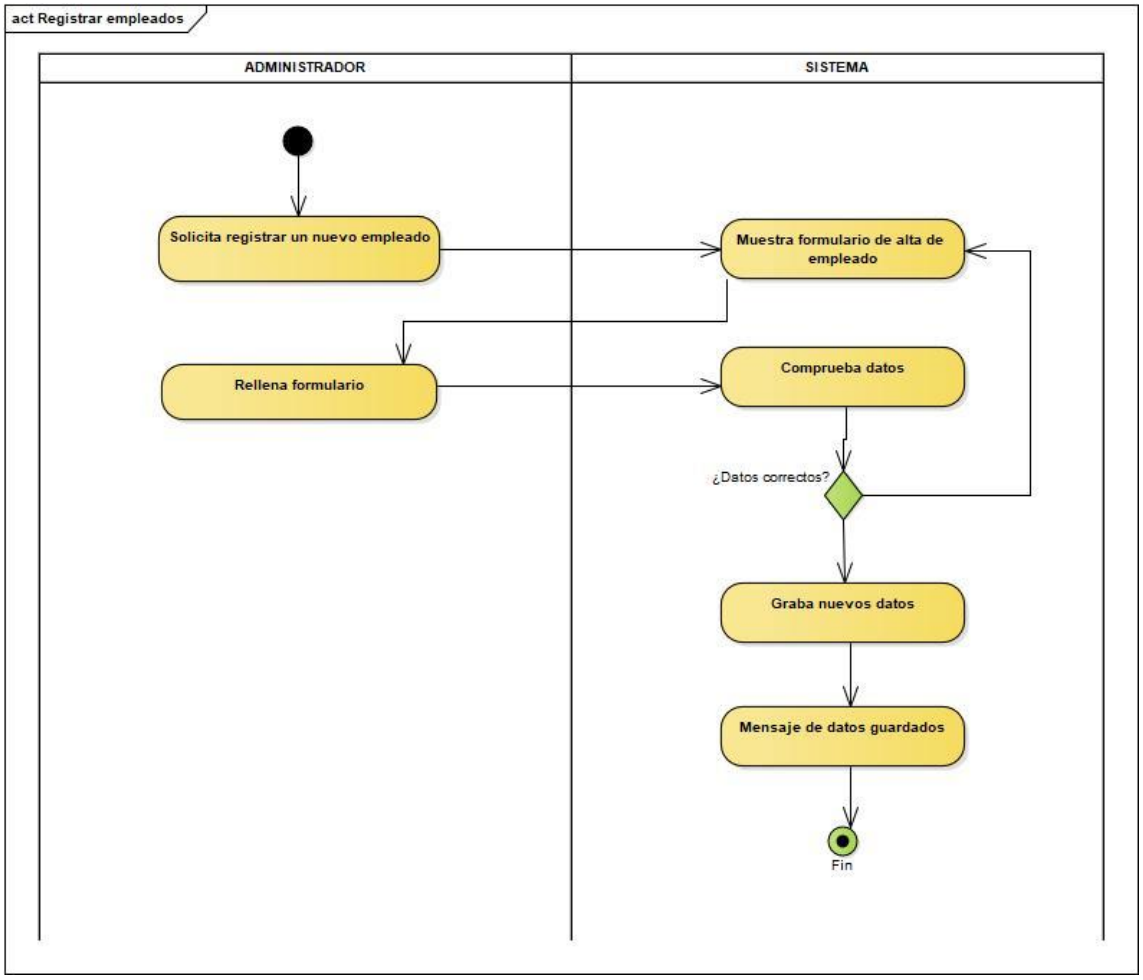


Figura 3-5. CU-002: Diagrama de actividad

3.3.2.3 Caso de uso: Registrar usuario

Mediante esta funcionalidad el administrador podrá generar un nuevo usuario y contraseña para un empleado cualquiera. De esta forma, cada empleado contará con un inicio de sesión predeterminado que posteriormente podrá modificar.

CU-003	Registrar usuario
Actores	Administrador
Precondiciones	El empleado correspondiente ha sido registrado en la base de datos.
Descripción	El administrador registrará un usuario y contraseña temporal en el sistema para un empleado, de forma que pueda iniciar sesión.

Secuencia Normal	**Paso**	**Acción**
	1	El administrador solicitar registrar un nuevo usuario
	2	El sistema muestra el formulario de registro
	3	El usuario introduce los datos correspondientes
	4	El sistema comprueba si los datos de acceso son válidos y no hay repeticiones
	5	Si son correctos, el sistema registra el usuario
	6	El sistema redirige al administrador a la página principal
Postcondición		
Excepciones	Paso	Acción
	4	Si son incorrectos, el sistema muestra un mensaje y redirige al formulario inicial
Comentarios	Tras el registro inicial, el empleado podrá modificar su usuario y contraseña a su elección.	

Tabla 3-4. CU-003: Registrar Usuario

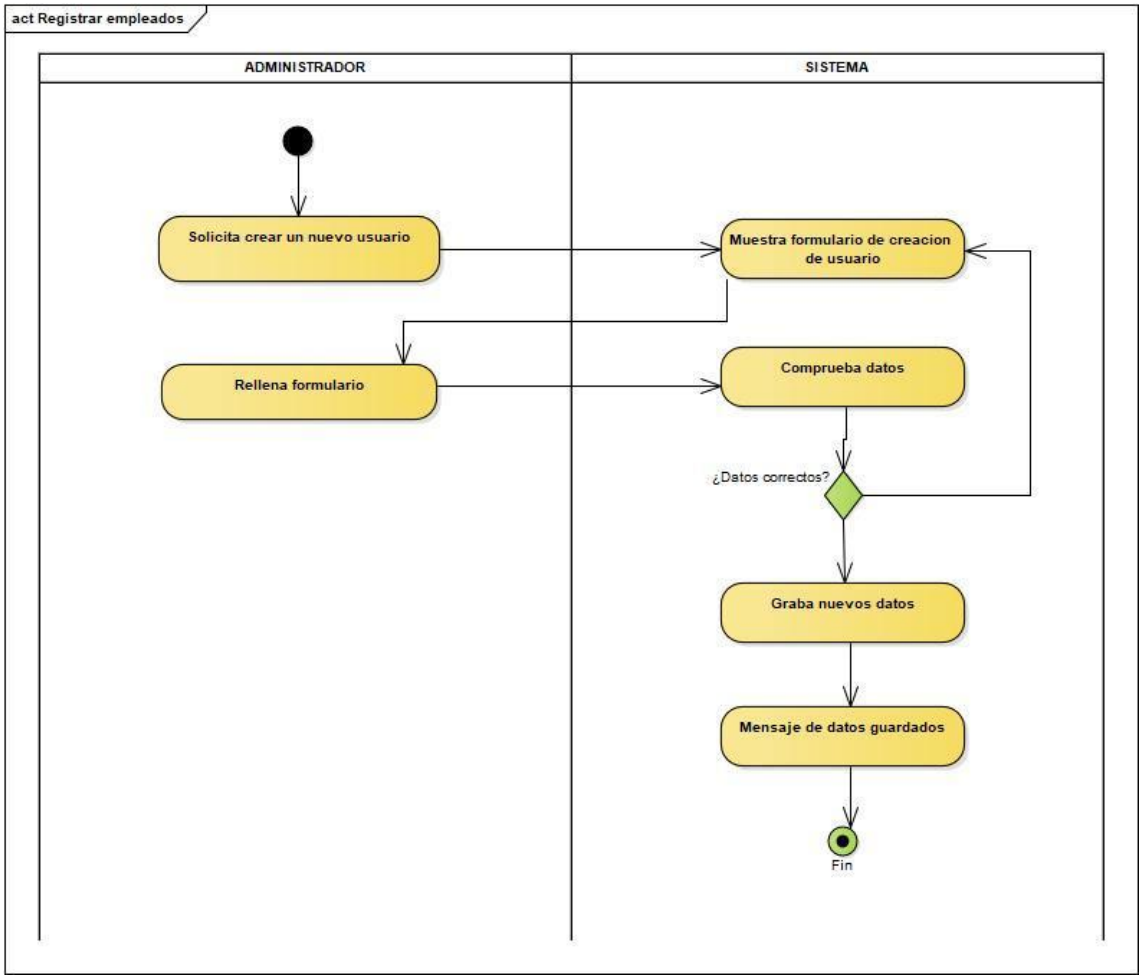


Figura 3-6. CU-003: Diagrama de actividad

3.3.2.4 Caso de uso: Mostrar empleados

El administrador deberá ser capaz de visualizar los datos que hay almacenados sobre los empleados, tanto la información general sobre la persona correspondiente como sus características laborales. De esta forma, el sistema le mostrará datos como la dirección o el teléfono y el tipo de turno o de contrato. Además, la modificación de datos será más sencilla si el administrador puede ver la información actual previamente.

CU-004	Mostrar empleados
Actores	Administrador
Precondiciones	Los empleados están registrados en la base de datos El administrador ha iniciado sesión
Descripción	El administrador recibe por pantalla la información existente sobre los empleados

Secuencia Normal	**Paso**	**Acción**
	1	El administrador solicitar ver los datos
	2	El sistema muestra la información sobre los empleados
Postcondición		
Excepciones	Paso	Acción
Comentarios		

Tabla 3-5. CU-004: Mostrar empleados

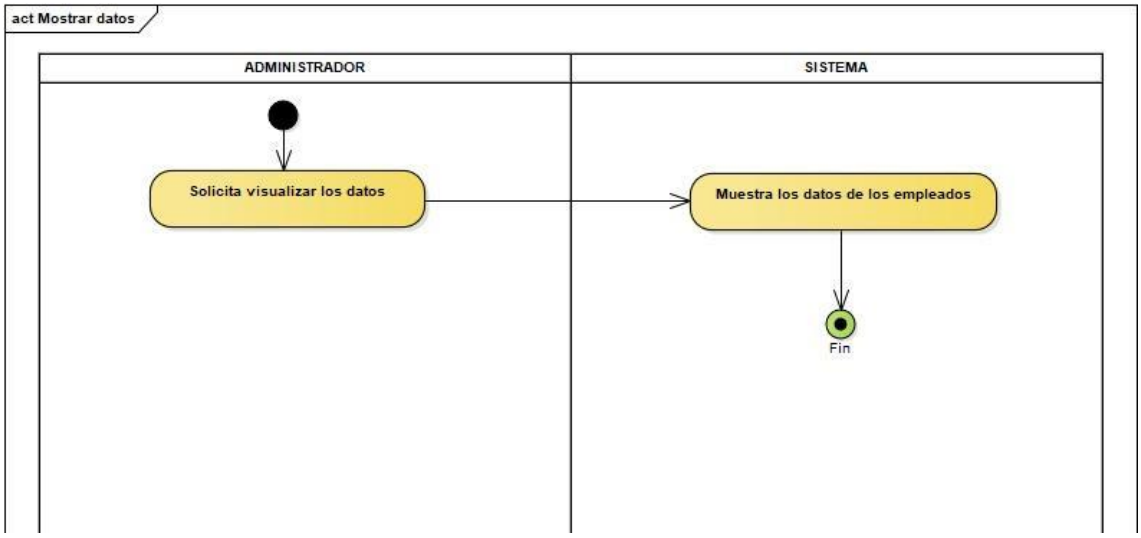


Figura 3-7. CU-004: Diagrama de actividad

3.3.2.5 Caso de uso: Generar cuadrante

Mediante este caso de uso el administrador puede generar un nuevo cuadrante de trabajo para todos los empleados de forma automática, y que atiende a las condiciones que se han estudiado anteriormente. El cuadrante asignará los turnos de los trabajadores entre las fechas que se proporcionan en el calendario.

CU-005	Generar cuadrante	
Actores	Administrador	
Precondiciones	Los empleados están registrados en la base de datos El administrador ha iniciado sesión	
Descripción	El administrador genera un calendario laboral que cumple con las condiciones necesarias, indicando los empleados y los días.	
Secuencia Normal	**Paso**	**Acción**
	1	El administrador solicitar generar un cuadrante
	2	El sistema muestra el formulario
	3	El usuario introduce el mes para el que quiere generar el cuadrante
	4	El sistema asigna los empleados a los días
	5	El sistema muestra el calendario generado
Postcondición		
Excepciones	Paso	Acción
Comentarios		

Tabla 3-6. CU-005: Generar cuadrante

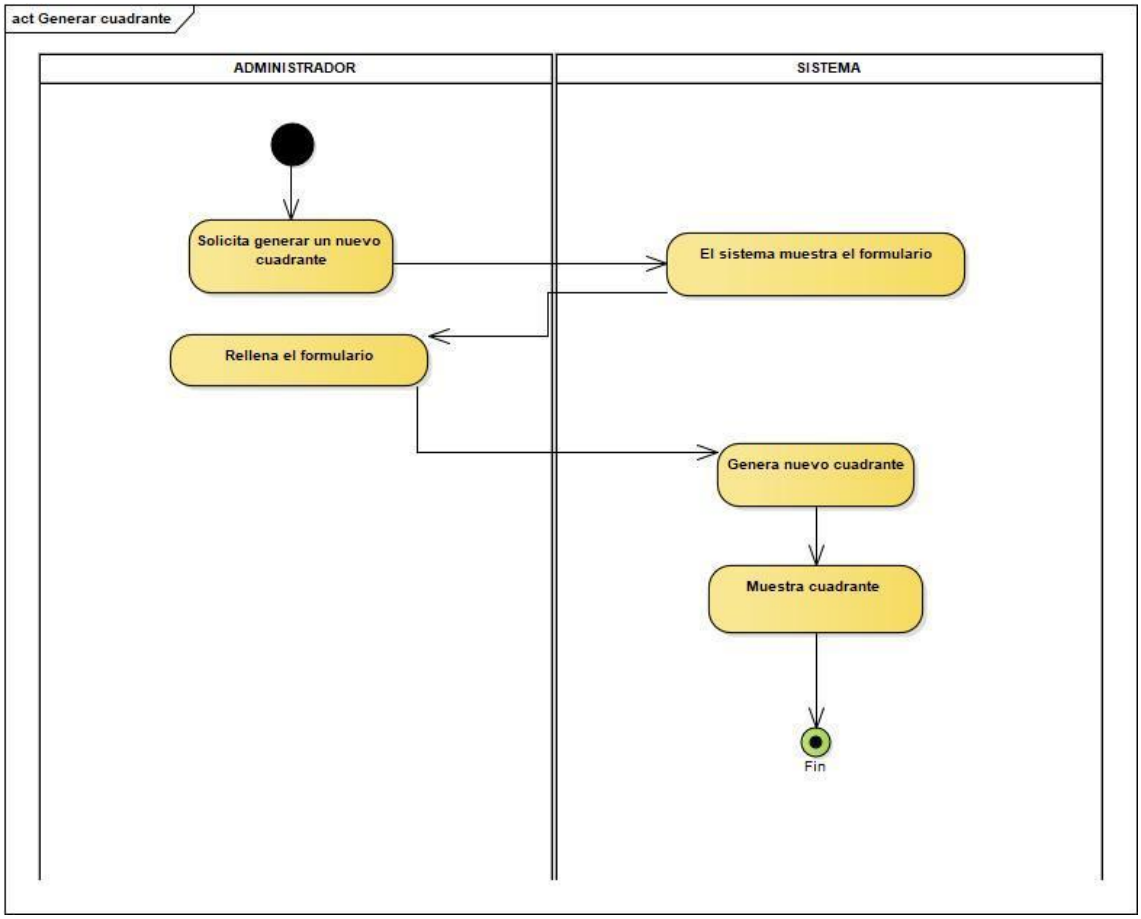


Figura 3-8. CU-005: Diagrama de actividad

3.3.2.6 Caso de uso: Descargar cuadrante

El modelado de este requisito permitirá al administrador descargar el cuadrante mensual en formato hoja de cálculo, permitiendo introducir cambios de una forma sencilla, o manipularlo por encima de las posibilidades del sistema de información.

CU-006	Descargar cuadrante	
Actores	Administrador	
Precondiciones	El administrador ha iniciado sesión	
Descripción	Se produce la descarga del horario en formato hoja de cálculo	
Secuencia Normal	**Paso**	**Acción**
	1	El administrador solicitar descargar el cuadrante
	2	El sistema descarga el cuadrante
Postcondición		
Excepciones	Paso	Acción
Comentarios		

Tabla 3-7. CU-006: Descargar cuadrante

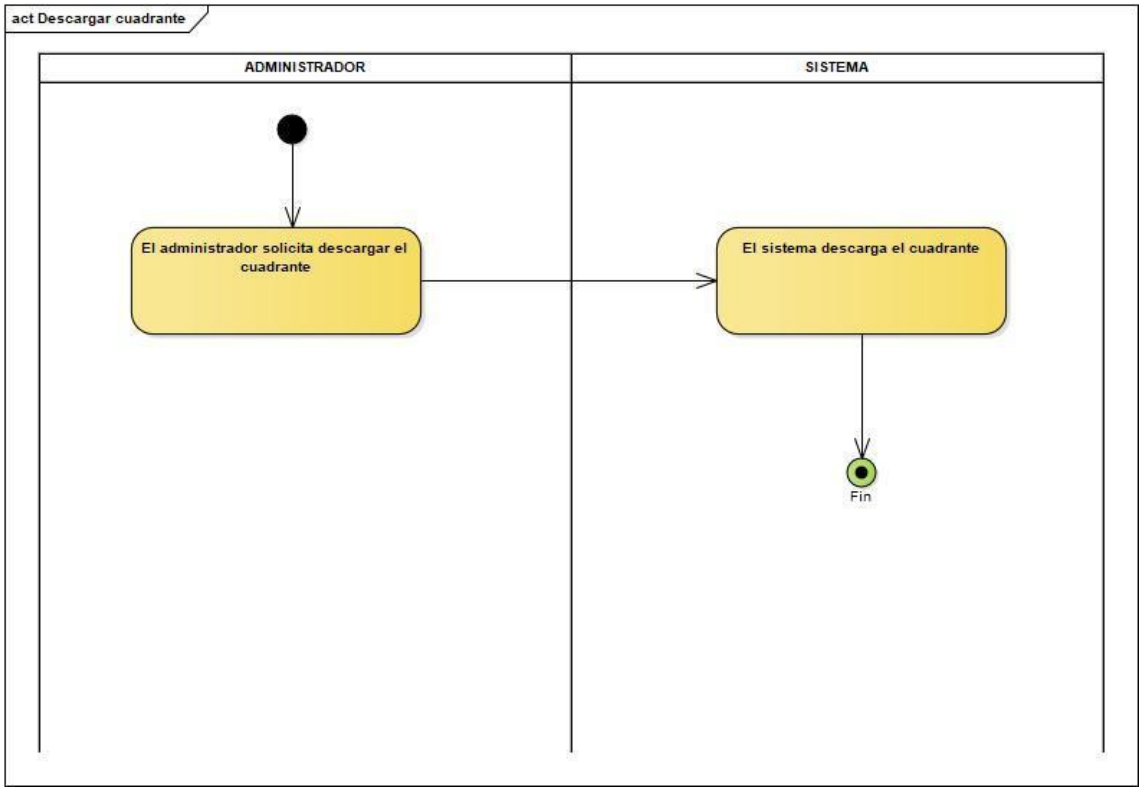


Figura 3-9. CU-006: Diagrama de actividad

3.3.2.7 Caso de uso: Visualizar cuadrante

Este caso de uso puede ser ejecutado por ambos actores, y se encarga de mostrar el cuadrante de trabajo actual. Al ejecutarse, por defecto se mostrará el cuadrante correspondiente al mes actual.

CU-007	Visualizar cuadrante	
Actores	Administrador - Empleado	
Precondiciones	El usuario ha iniciado sesión y existe un cuadrante en la base de datos	
Descripción	El usuario podrá ver en pantalla el cuadrante actual	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario solicita visualizar el cuadrante
	2	El sistema muestra el cuadrante

Postcondición		
Excepciones	Paso	Acción
Comentarios	La pantalla que muestra el calendario será distinta para el Administrador y para el Usuario.	

Tabla 3-8. CU-007: Visualizar cuadrante

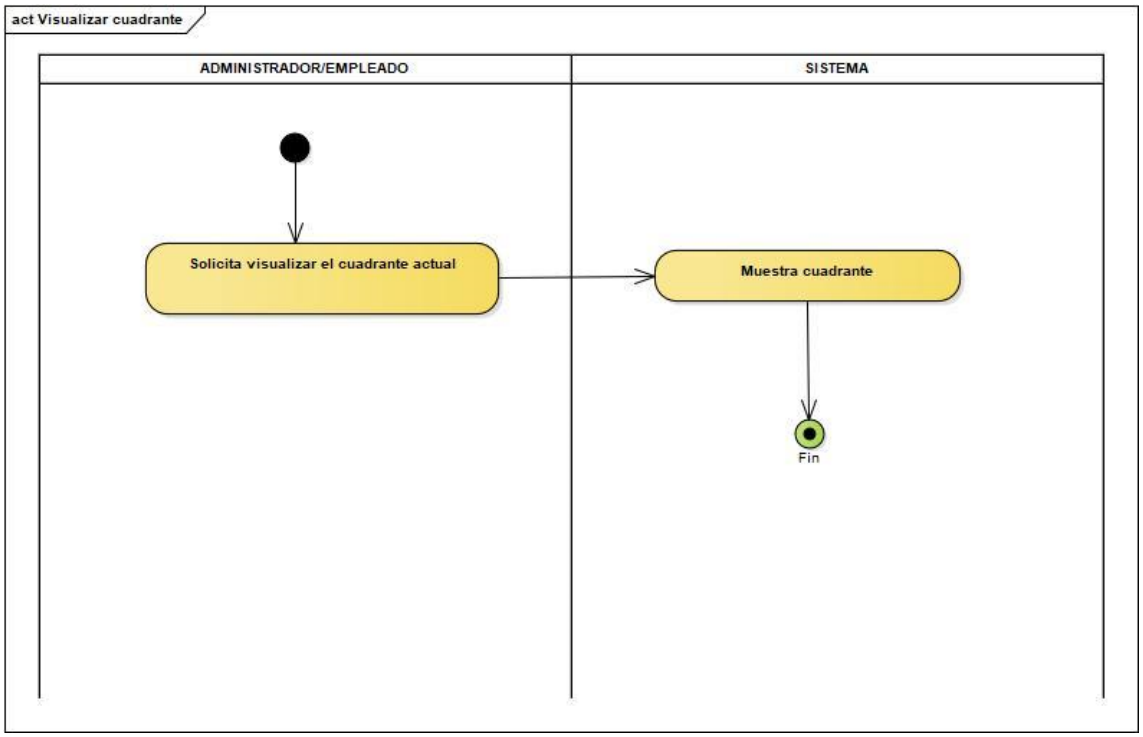


Figura 3-10. CU-007: Diagrama de actividad

3.3.2.8 Caso de uso: Solicitar baja

Ambos actores podrán solicitar una baja por cualquier motivo. De esta forma, si el empleado no puede registrarla lo podrá hacer el administrador.

CU-008	Solicitar baja	
Actores	Administrador - Empleado	
Precondiciones	El empleado correspondiente ha sido registrado en la base de datos.	
Descripción	El empleado o el administrador solicitarán una baja al sistema, ya sea baja médica o vacaciones. El sistema lo registrará y lo usará para asignar el calendario correctamente	
Secuencia Normal	**Paso**	**Acción**
	1	El actor solicita dar de baja
	2	El sistema muestra el formulario de registro
	3	El usuario introduce los datos correspondientes
	4	El sistema comprueba si los datos son válidos.
	5	Si son correctos, el sistema registra la baja, con fecha de inicio y fin
	6	El sistema redirige al usuario a la página principal correspondiente
Postcondición	La baja queda registrada e influirá directamente en la generación de cuadrante.	
Excepciones	Paso	Acción
	5	Si son incorrectos, el sistema muestra un mensaje y redirige al formulario inicial
Comentarios	La unión de baja y vacaciones responde a la optimización de la base de datos, ya que la información de ambas se almacena de la misma forma, y las condiciones de suplencia son las mismas.	

Tabla 3-9. CU-008: Solicitar baja

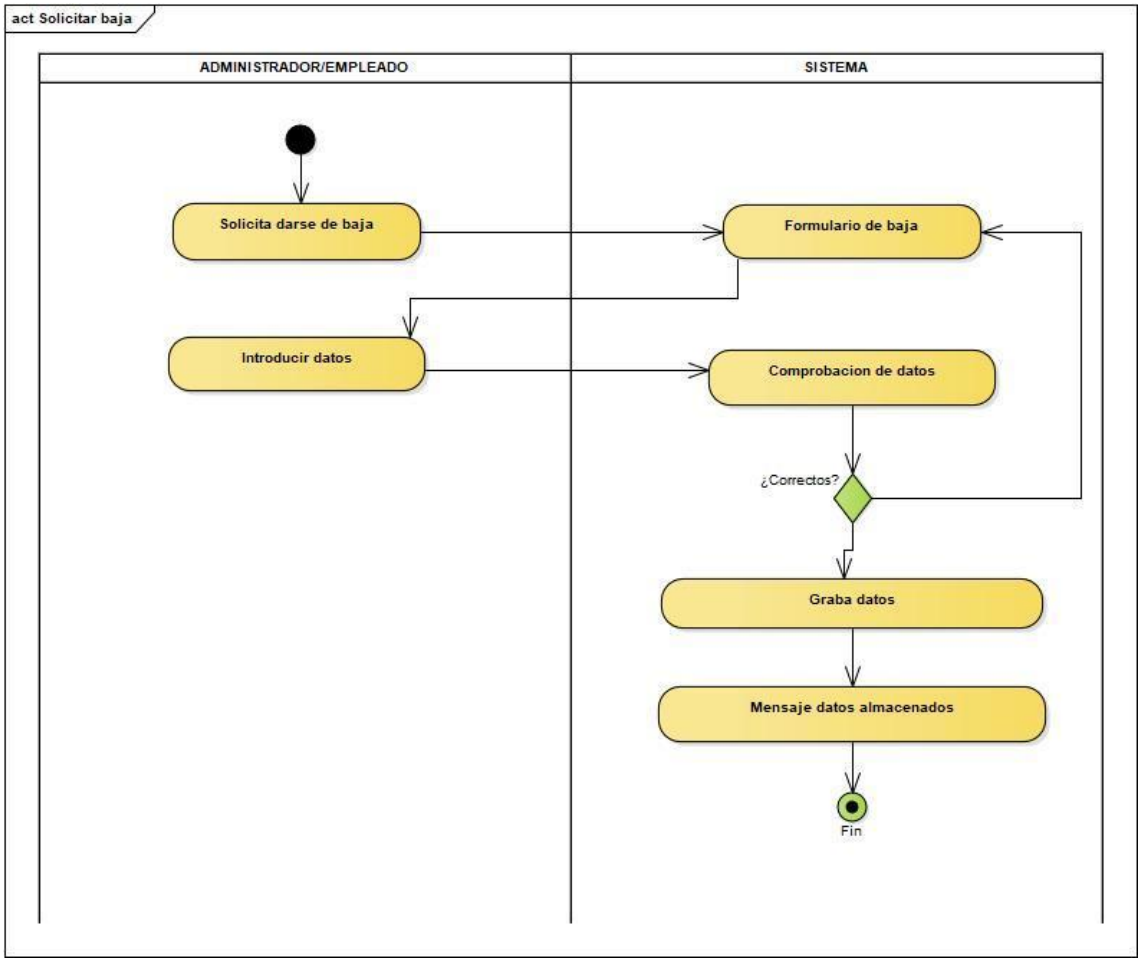


Figura 3-11. CU-008: Diagrama de actividad

3.3.2.9 Caso de uso: Modificar datos

Ambos actores podrán modificar algunos datos ya introducidos en el servidor, con el propósito de corregir posibles errores o actualizar la información cuando sea necesario.

CU-009	Modificar datos	
Actores	Administrador	
Precondiciones	El empleado correspondiente ha sido registrado en la base de datos.	
Descripción	El usuario podrá modificar ciertos datos específicos que no influyen con la asignación de turnos.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario solicita realizar modificaciones

	2	El sistema muestra el formulario de modificación
	3	El usuario introduce los datos correspondientes
	4	El sistema comprueba si los datos son válidos.
	5	Si son correctos, el sistema registra las modificaciones
	6	El sistema redirige al administrador a la página principal
Postcondición		
Excepciones	Paso	Acción
	5	Si son incorrectos, el sistema muestra un mensaje y redirige al formulario inicial
Comentarios	Este caso de uso se modela para posibles cambios de domicilio, nombre u otro tipo de datos de los que no depende la organización	

Tabla 3-10. CU-009: Modificar datos

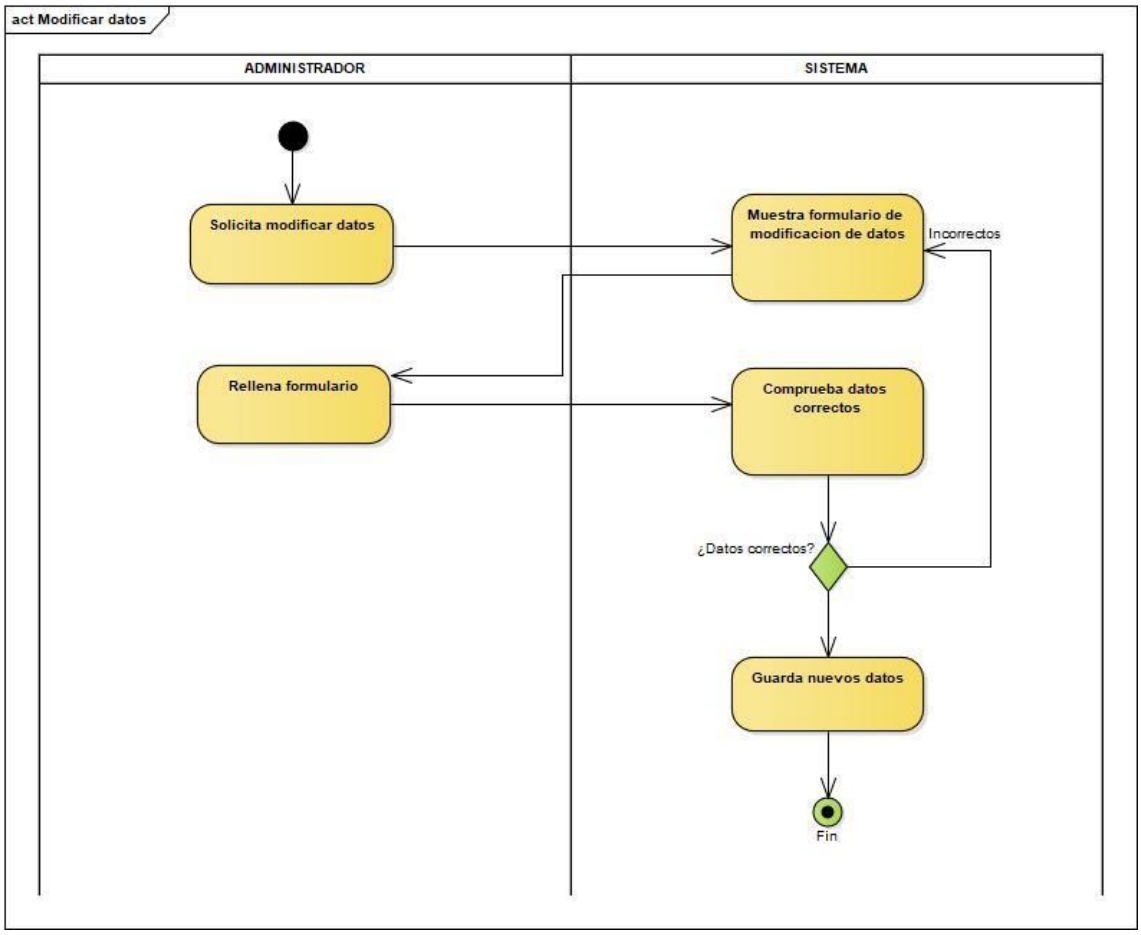


Figura 3-12. CU-009: Diagrama de actividad

3.3.2.10 Caso de uso: Modificar Usuario

Ambos actores tendrán a su disposición la posibilidad de modificar sus credenciales de acceso al sistema, de forma que puedan cambiarlas por alguna que le resulte más cómoda.

CU-010	Modificar Usuario	
Actores	Administrador - Empleado	
Precondiciones	El usuario está registrado en la base de datos.	
Descripción	El usuario podrá modificar ciertos datos específicos que no influyen con la asignación de turnos.	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario solicita realizar modificaciones
	2	El sistema muestra el formulario de modificación
	3	El usuario introduce los datos correspondientes
	4	El sistema comprueba si los datos son válidos.
	5	Si son correctos, el sistema registra las modificaciones
	7	El sistema redirige al actor a la página principal
Postcondición		
Excepciones	Paso	Acción
	4	Si son incorrectos, el sistema muestra un mensaje y redirige al formulario inicial
Comentarios		

Tabla 3-11. CU-010: Modificar Usuario

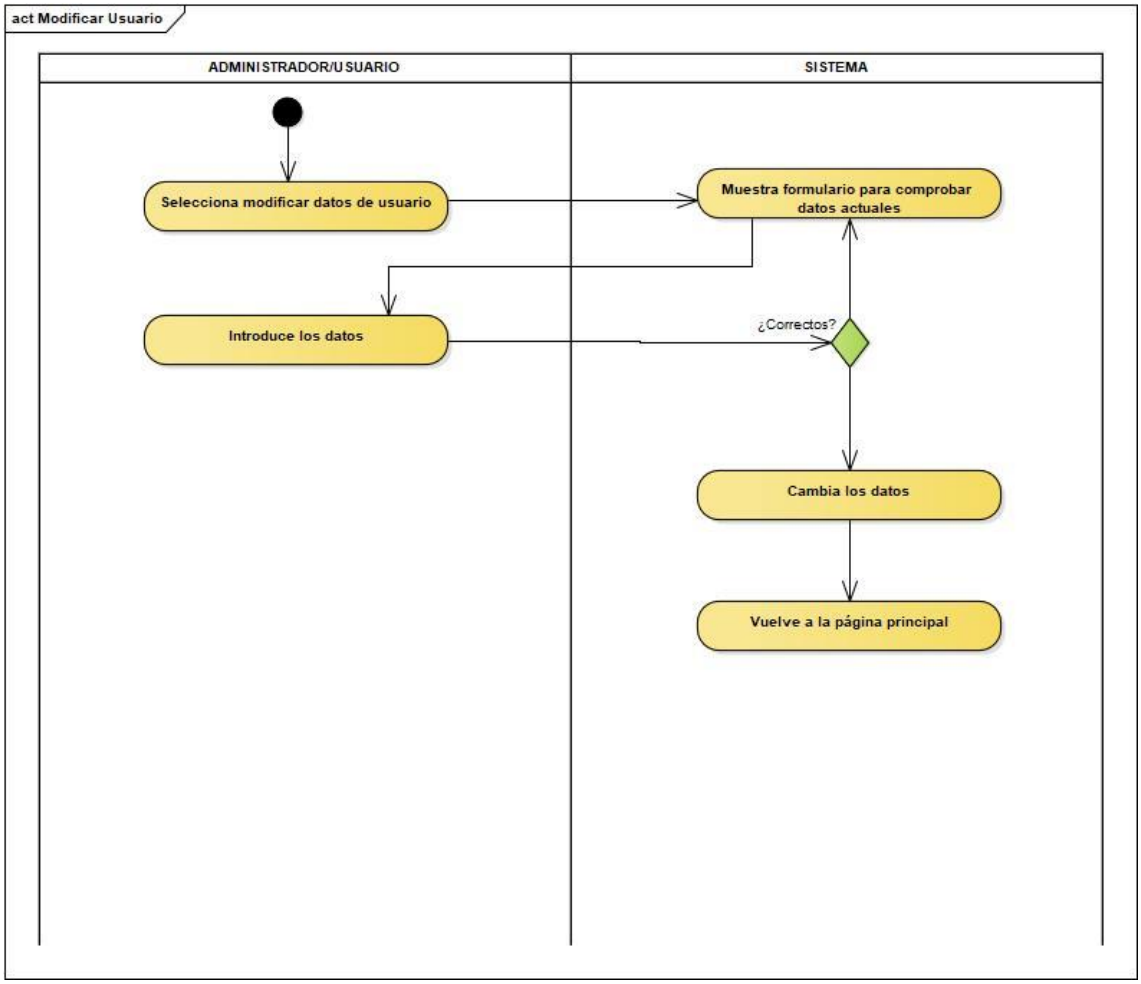


Figura 3-13. CU-010: Diagrama de actividad

3.3.2.11 Caso de uso: Cubrir bajas

El administrador podrá seleccionar un empleado para suplir a un compañero de baja, siempre que se cumplan las condiciones que marca la organización. En este caso serían:

- Si el turno a cubrir es de tipo rotatorio de mañana, lo podrán cubrir los empleados que tengan ese mismo turno si no trabajan ese día, y los que tengan rotatorio siempre que no trabajen ese día y el día anterior no hiciesen turno de noche, para evitar que salgan a las 8 de la mañana y vuelvan a entrar a esa misma hora.
- Si el turno a cubrir es turno rotatorio, se dan dos posibilidades. Si el empleado de baja hacia turno de noche ese día, lo pueden cubrir los que tengan turno de mañana pero no trabajen ni ese mismo día ni el siguiente, o los que tengan turno rotatorio, pero no trabajen ese día. Si por el contrario el turno a suplir es diurno, lo podrán hacer los que tengan turno de mañana y no trabajen o los que tengan rotatorio y ni trabajen ni hiciesen noche el día anterior.

El empleado seleccionado "copiará" el turno que cubre, y las horas trabajadas le serán asignadas como propias. La selección de estos empleados no es automática, dado que se basa en la comunicación entre jefe de planta y personal, y en criterios de deuda horaria y voluntariedad. Así, si un empleado debe horas a la organización por estar en un mes con déficit horario, puede ofrecerse a suplir a algún compañero, regularizando su calendario laboral. Es por ello por lo que el sistema de información mostrará al

administrador los empleados que pueden cubrir el turno y las horas que tienen asignadas en ese mes.

El sistema no cubre la casuística de que no existan candidatos que puedan cubrir el turno sin incumplir las normas, por lo que el administrador deberá tomar una decisión sin el apoyo del sistema, dado que esta incluiría modificaciones del calendario generado automáticamente.

CU-011	Cubrir bajas	
Actores	Administrador	
Precondiciones	El calendario ha sido generado y algún empleado está de baja en las fechas incluidas.	
Descripción	El administrador podrá suplir las bajas de los empleados con un compañero a su elección.	
Secuencia Normal	**Paso**	**Acción**
	1	El actor solicita cubrir las bajas
	2	El sistema muestra los empleados de baja
	3	El administrador selecciona el empleado a cubrir
	4	El sistema muestra los empleados que pueden cubrir ese turno
	5	El administrador selecciona al trabajador que va a suplir la baja
	6	El sistema asigna al empleado al turno correspondiente
	7	El sistema muestra al resto de empleados de baja
Postcondición		
Excepciones	Paso	Acción
	2	Si no existen empleados de baja, el sistema muestra un mensaje desde el que se puede volver al calendario.
Comentarios		

Tabla 3-12. CU-011: Cubrir bajas

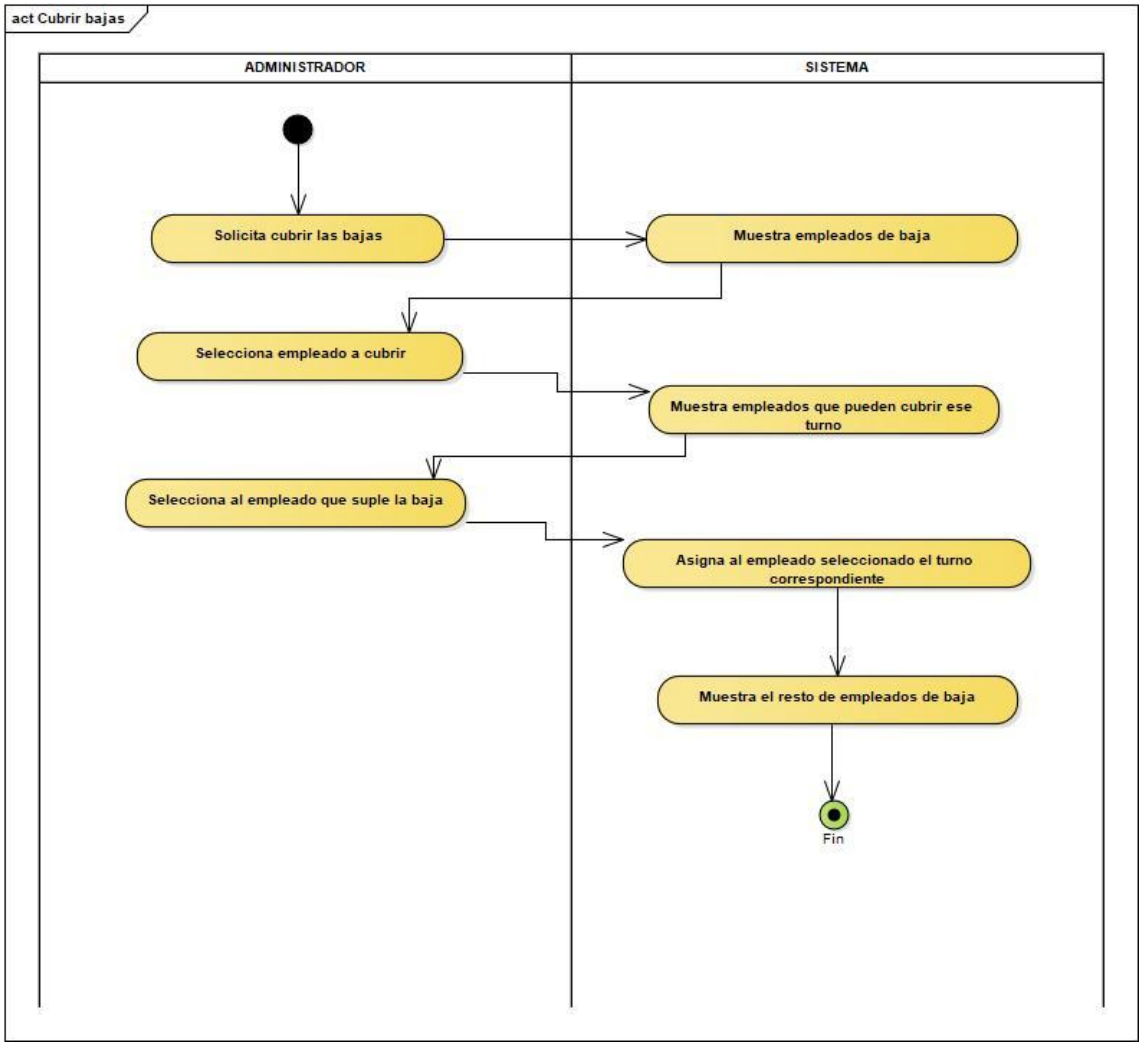


Figura 3-14. CU-011: Diagrama de actividad

3.3.2.12 Caso de uso: Eliminar empleado

Cuando sea necesario eliminar a un empleado de la base de datos, el administrador podrá realizarlo.

CU-012	Eliminar empleado	
Actores	Administrador	
Precondiciones	El empleado a eliminar está registrado en la base de datos	
Descripción	Se eliminará el registro de un empleado concreto	
Secuencia Normal	**Paso**	**Acción**

	1	El administrador solicita eliminar un empleado
	2	El sistema muestra un mensaje de confirmación
	3	El administrador confirma o rechaza la decisión
	4	El sistema comprueba la decisión
	5	Si se confirma, elimina al empleado
	6	El sistema muestra un mensaje
Postcondición	El registro del empleado correspondiente desaparece de la base de datos	
Excepciones	Paso	Acción
	3	Si la decisión ha sido rechazada, vuelve al formulario de selección
Comentarios		

Tabla 3-13. CU-012: Eliminar empleado

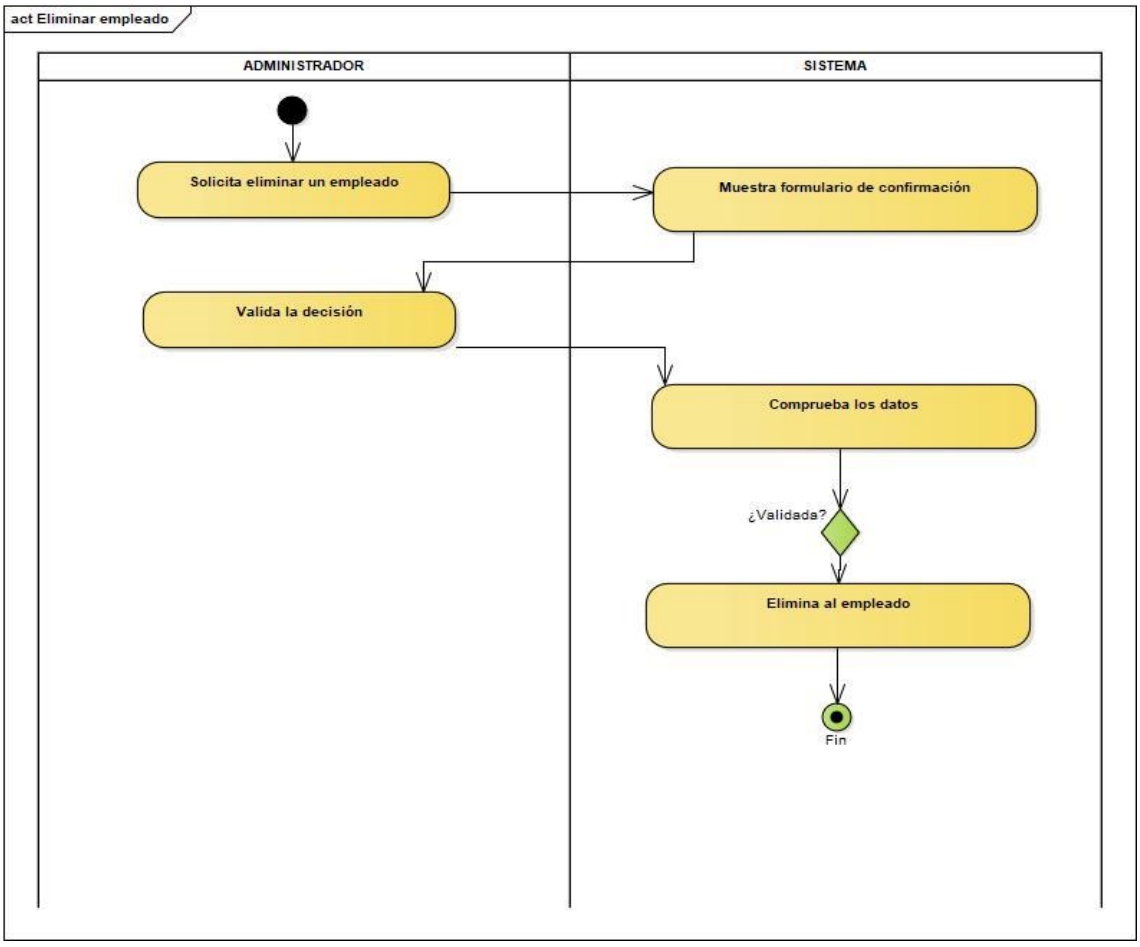


Figura 3-15. CU-012: Diagrama de actividad

3.3.2.13 Caso de uso: Cerrar sesión

Este caso de uso es relevante no solo por poder salir de la página cuando se haya finalizado la tarea que se quería realizar, si no como herramienta de seguridad para garantizar que sólo los usuarios acreditados pueden acceder a la página y a sus acciones permitidas.

CU-013	Cerrar sesión	
Actores	Administrador – Empleado	
Precondiciones	La sesión ha sido iniciada previamente	
Descripción	El sistema se encargará de cerrar la sesión que se inicializó anteriormente.	
Secuencia Normal	**Paso**	**Acción**
	1	El actor solicita cerrar sesión.
	2	El sistema muestra mensaje de confirmación.
	3	Si el actor confirma su decisión, el sistema cierra la sesión.
	4	El sistema devuelve a la página de inicio.
Postcondición	La sesión queda cerrada.	
Excepciones	Paso	Acción
	3	Si el actor descarta cerrar sesión, el sistema redirige a la página principal correspondiente.
Comentarios		

Tabla 3-14. CU-013: Cerrar sesión

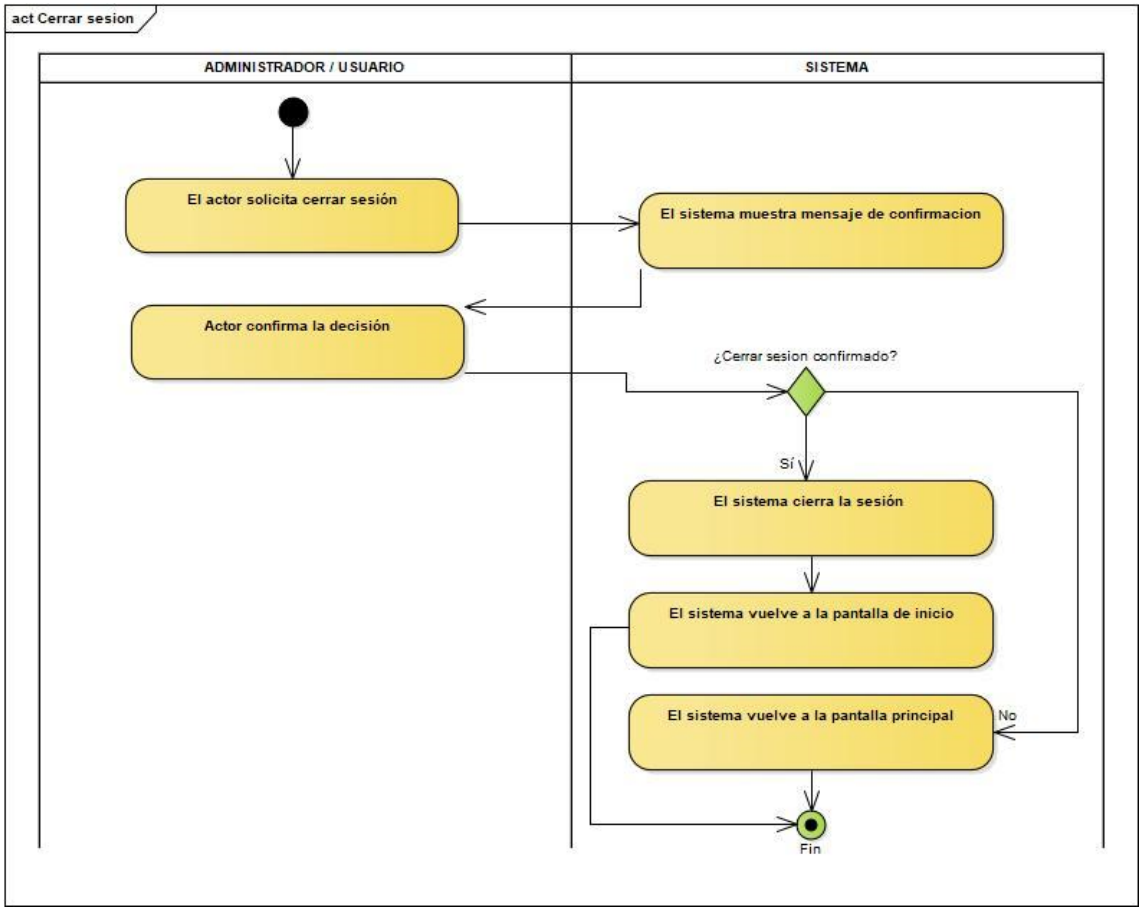


Figura 3-16. CU-013: Diagrama de actividad

4 MODELO

El modelado de la base de datos comprende tanto la información almacenada como las relaciones que se establecen entre la distinta información recogida. Para conocer el funcionamiento del sistema no es suficiente saber qué contiene, si no como se relaciona con su exterior. Para cubrir ambos aspectos, se exponen los distintos tipos de modelos que se han generado.

4.1 Modelo Entidad-Relación

El Modelo Entidad-Relación es una forma de visualización de datos descrito por primera vez en 1976 por Peter Cheng [42]. Conocido más habitualmente como ERD, está diseñado específicamente para tratar con bases de datos relacionales. El ERD permite obtener un análisis de un sistema de información desde su base, atendiendo a la diferencia entre Entidades, que como su nombre indica son personas, organizaciones, u entes similares que pueden definirse inequívocamente y Relaciones, que representan una asociación entre entidades. Las Entidades poseen atributos, que son cualidades que las definen o características específicas que poseen.

Las cardinalidades refieren el tipo de relación que se establece entre Entidades, como hemos visto en el apartado anterior. La información obtenida mediante este modelo es utilizada posteriormente en el desarrollo de la base de datos. El modelo ERD para el problema planteado en este trabajo es el siguiente:

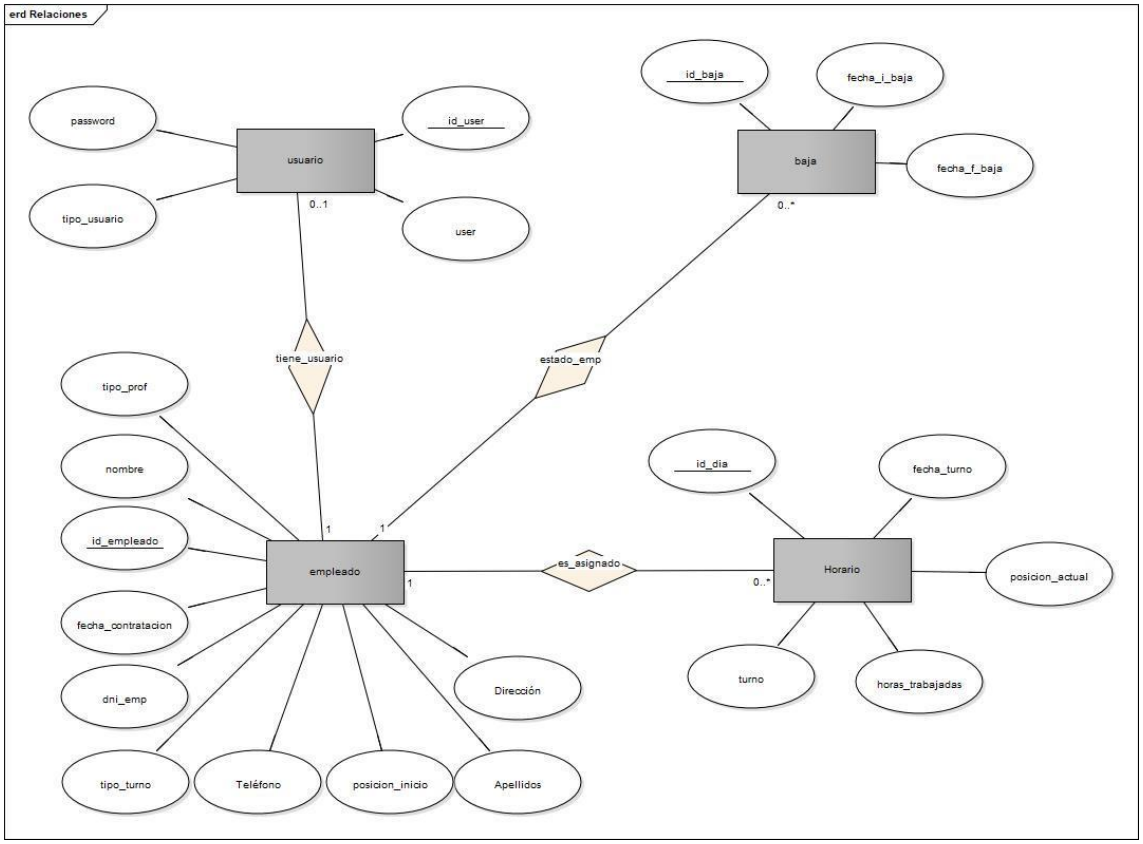


Figura 4-1. Diagrama ERD

Posteriormente quedarán expuestas las tablas y las relaciones entre ellas, donde se puede observar la fidelidad con el modelo ERD desarrollado.

4.2 Modelo Relacional

El modelo relacional fue descrito por Codd [43] como una forma de solucionar los posibles errores que se daban al escalar la cantidad de datos o modificar la estructura. Codd creó un modelo matemático a través de lógica de predicados, que posteriormente sería adaptado al mundo de la programación, donde se ha convertido en uno de los modelos más utilizado a la hora de diseñar bases de datos.

El modelo relacional divide la estructura de almacenamiento en:

- Atributo: Cada una de las columnas, es decir, un elemento susceptible de tomar un valor
- Dominio: Conjunto de valores que pueden adoptar los atributos
- Tupla: Las filas, cada elemento que contiene un atributo.

Al crear las relaciones, surge el concepto de Clave Primaria, que impide la duplicación de las tuplas y las identifica. También la Clave Externa, que permitirá identificar una tupla desde otra tabla distinta. Además, dependiendo de la cardinalidad, se creará un tipo de relación u otra.

La cardinalidad es un concepto de las relaciones entre tablas que representa el número de registros cruzados que se dan. De esta forma, una relación puede ser de 1 a 1, de 1 a varios, de varios a varios, etc. Por ejemplo, un escritor puede haber escrito muchos libros, pero un libro generalmente no tiene más de un escritor. Por lo tanto, la relación escritor-libro sería de 1 a varios. Si cambiásemos la analogía por

actores y películas, nos damos cuenta de que la cardinalidad cambia.

La base de datos desarrollada para este caso tiene distintas cardinalidades entre sus tablas. En la siguiente imagen podemos ver las relaciones existentes entre tablas

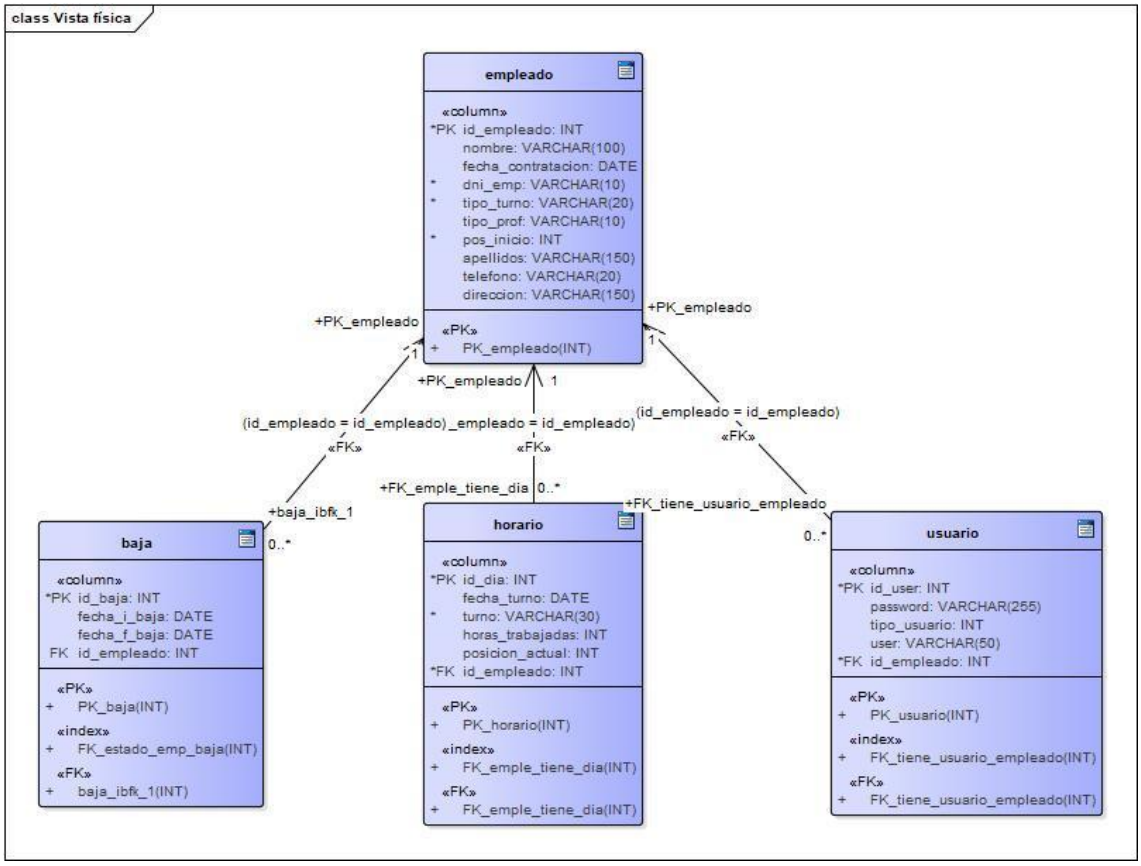


Figura 4-2. Vista física

Estas relaciones, y las propias tablas, quedan más detalladas en la tabla expuesta a continuación:

Tabla 4-1. Relaciones entre tablas

Relación	Cardinalidad	Comentarios
Empleado - Usuario	1 a 0 ó 1	El empleado deberá tener un usuario, pero inicialmente no es así hasta que el administrador genera los datos, por lo que puede no tener en un momento dado
Empleado - Baja	1 a 0…*	Cada empleado podrá tener asignado un número indefinido de bajas o vacaciones, o no tener ningún registro.
Empleado – Horario	1 a 0…*	Cada empleado tendrá muchos registros asignados en la tabla horario, concretamente uno por cada día que trabaje o descanse. La clave externa en horario es el identificador del empleado, que definirá inequívocamente cada uno de los días registrados en ella.

4.2.1 Descripción de las tablas

En las tablas encontramos toda la información que el desarrollador ha requerido para la resolución del problema planteado. Estas tablas se dividen en filas, donde se encuentra el dato, y columnas, que agrupan esos datos en distintos tipos. Los campos almacenados tienen ciertas características según el tipo de información que contienen o la forma en la que se guarda. Para poder detallar mejor las tablas, se expone a continuación información sobre estas características.

- Nombre: El nombre del campo en la tabla
- Tipo: Referido al tipo de dato que está almacenado. Existen muchos tipos de dato a almacenar, aunque los más comunes, y los usados en la base de datos desarrollada, quedan recogidos en la siguiente tabla

Tabla 4-2. Tipo de datos

Nombre	Descripción
INT	Numero entero, de longitud variable
VARCHAR	Cadena de caracteres, incluyendo letras, números y símbolos. Tiene una longitud máxima de 255.
DATE	Almacena una fecha en formato AAAA-MM-DD, sin longitud variable.
BOOLEAN	Datos binarios, representados por 0 o por 1

- Longitud: Número de caracteres almacenados. En la descripción de las tablas queda recogido entre paréntesis junto al tipo.

4.2.1.1 Empleado

La tabla “empleado” contiene toda la información necesaria referente a los trabajadores de la organización, incluyendo los administradores. Es un registro administrativo de toda la información útil sobre las personas que están contratados. En la siguiente tabla queda registrado tanto el nombre de los campos de la tabla, como una breve descripción y el tipo de datos que almacena. La información que se almacena actualmente en la tabla “empleado” puede ser extendida, añadiendo los campos que se consideren necesarios.

Tabla 4-3. Empleado

Nombre	Descripción	Tipo
Id_empleado	Identificador del empleado, autoasignado.	INT
Nombre	El nombre propio del empleado	VARCHAR (50)
Apellidos	Ambos apellidos del empleado	VARCHAR (150)
Fecha_contratación	La fecha de inicio del contrato	DATE
Dni_emp	Documento Nacional de Identidad del empleado	VARCHAR (10)
Tipo_turno	Clasifica mediante una cadena de caracteres predeterminada el tipo de turno del empleado	VARCHAR (20)
Tipo_prof	Clasifica a los empleados según pertenezcan al tipo enfermería o al tipo auxiliar.	VARCHAR (20)
Telefono	Número de teléfono del empleado, que podrá introducirse con el código internacional si es necesario	VARCHAR (20)
Direccion	Dirección del empleado	VARCHAR (150)

4.2.1.2 Usuario

En la tabla "usuario" se almacenan los datos necesarios para el inicio de sesión en el sistema.

Tabla 4-4. Usuario

Nombre	Descripción	Tipo
Id_user	Identificador del usuario almacenado, autoasignado. Clave Primaria	INT (11)
Password	Contraseña de inicio de sesión, almacenada encriptada mediante la función de encriptación predefinida en PHP	VARCHAR (255)
Tipo_usuario	El tipo de usuario, administrador o empleado, almacenado de forma binaria (0 empleado, 1 administrador)	INT (11)
User	Nombre de usuario para efectuar el inicio de sesión	VARCHAR (50)
Id_empleado	Identificador del empleado que tiene asignado el usuario concreto. Clave Externa.	INT (11)

4.2.1.3 Baja

En la tabla "baja" quedan recogidas las bajas laborales y las vacaciones que los empleados solicitan, que será información de gran relevancia para la generación del calendario laboral anual.

Tabla 4-5. Baja

Nombre	Descripción	Tipo
Id_baja	Identificador de la tabla baja, autoasignado. Clave Primaria	INT (11)
Fecha_i_baja	Fecha de inicio de la baja o vacación	DATE
Fecha_f_baja	Fecha de fin de baja o vacaciones	DATE
Id_empleado	Identificador del empleado. Clave Externa.	INT (11)

4.2.1.4 Horario

La tabla "horario" contiene los días de trabajo, y será la información mostrada al ejecutar la asignación automática de los empleados.

Tabla 4-6. Horario

Nombre	Descripción	Tipo
Id_dia	Identificador de la tabla horario, autoasignado. Clave Primaria	INT (11)
Fecha_turno	Fecha del día asignado	DATE
Turno	Almacena el tipo de turno que está desarrollando un empleado concreto en la fecha correspondiente.	VARCHAR (50)
Horas_trabajadas	Representa el número de horas que se trabajan en un día concreto.	INT (5)
Id_empleado	Identificador del empleado. Clave Externa.	INT (11)
Posicion_actual	Representa la posición en la secuencia rotatoria que el empleado está realizando ese día concreto.	INT (5)

4.3 Diagrama de clases

Las clases permiten agrupar objetos relacionados entre ellos, y que usan los mismos métodos y variables. Estas clases están estrechamente relacionadas con las entidades que se han generado en el modelo ERD.

La declaración de las clases y las funciones que contienen definen la parte "Modelo" del MVC.

La implementación desarrollada divide la funcionalidad en cuatro clases, expuestas en la siguiente imagen:

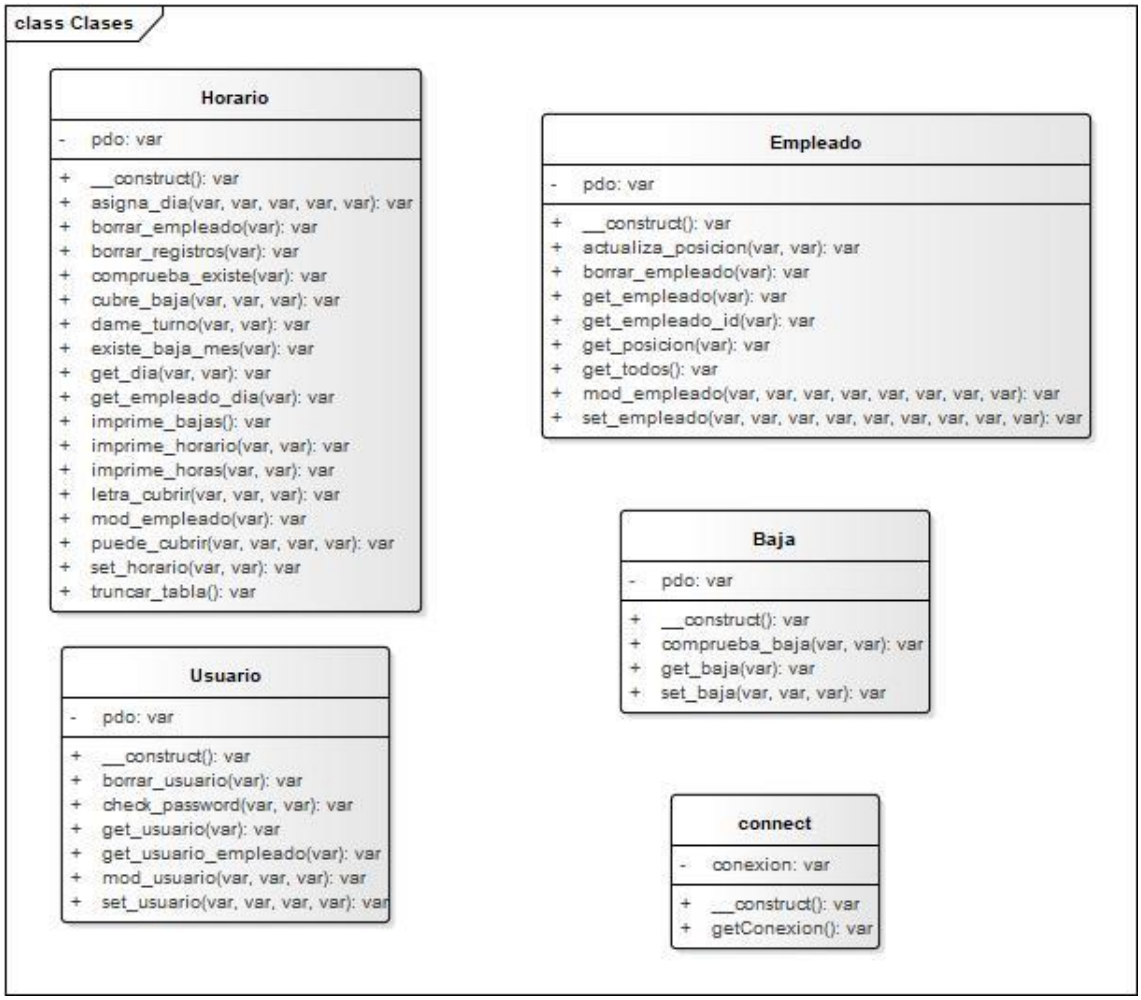


Figura 4-3. Diagrama de clases

Estas clases, salvo la llamada “connect”, se corresponden a las tablas de la base de datos, y sus métodos permiten manipular los datos contenidos en ella, haciendo posible la implementación del sistema.

A continuación, se expone un desglose de cada una de ellas y una breve explicación de los métodos que contienen.

4.3.1 Clase connect

Esta clase se encarga de la conexión con el servidor MySQL. Para ello hace uso de una variable, llamada conexión, y de dos métodos, que son los siguientes:

- __construct(): Es la función predeterminada de inicialización de PHP. Se invoca cada vez que se crea un nuevo objeto perteneciente a esta clase, por lo que al crearlo estamos también inicializando. En este caso no recibe variables, pero contiene los datos necesarios para establecer la conexión con la base de datos correspondiente. Almacena la conexión en la variable

"conexion".

- getConexion(): Cuando es invocada devuelve la variable conexion. Será utilizada en las funciones constructoras de las demás clases.

4.3.2 Clase baja

La clase "baja" nos permitirá operar con los contenidos de su tabla homónima. Solo tiene una variable definida, llamada pdo, utilizada para la ejecución de métodos PDO. Entre sus métodos encontramos:

- __construct(): Crea un objeto de tipo connect y ejecuta el método getConexion() de esa clase, que posteriormente almacena en la variable conn.
- get_baja(): desuso de momento
- set_baja(): Recibe las variables necesarias para insertar un nuevo registro en la tabla baja (fecha de inicio, fecha de fin, identificador del empleado) y ejecuta la orden de inserción.
- Comprueba_baja(): Recibe una fecha y el identificador de un empleado, y devuelve una variable binaria de valor TRUE si la fecha dada corresponde a una baja para ese empleado o FALSE en caso contrario.

4.3.3 Clase usuario

Esta clase contiene los métodos necesarios para operar con su tabla homónima. Al igual que la anterior, solo tiene una variable definida, con el mismo nombre.

- __construct(): Funciona exactamente igual que el constructor de la clase "baja".
- Get_usuario(): Recibe un nombre de usuario y devuelve toda la información recogida en la tabla sobre ese usuario concreto, en forma de array asociativo.
- Get_usuario_empleado(): Es igual que el método anteriormente expuesto, pero este recibe el identificador del empleado del que quiero obtener la información.
- Mod_usuario(): Recibe contraseña, nombre de usuario e identificador. Posteriormente encripta la contraseña recibida y actualiza la información del empleado que corresponde con el identificador con los nuevos datos recibidos.
- Set_usuario(): Recibe la contraseña, el tipo de usuario, el nombre y el identificador y los introduce en la tabla, habiendo encriptado previamente la contraseña que el actor introduzca en el formulario.

4.3.4 Clase empleado

Mediante esta clase, que al igual que las anteriores solo tiene una variable definida, se maneja la información contenida en la tabla empleado. Los métodos definidos son los siguientes:

- __construct(): De nuevo, coincide con los anteriores métodos constructores.
- Get_empleado(): Recibe el DNI de un empleado y devuelve mediante un array asociativo todos los registros de la tabla empleado cuyo campo DNI coincida con la variable obtenida.
- Get_empleado_id(): Funciona de forma parecida al método anterior, pero recibiendo el identificador.
- Get_todos(): Devuelve todos los registros de la tabla empleado, en forma tanto de array asociativo como numérico.

- Get_posicion(): Recibe el identificador de un empleado y devuelve la posición de la secuencia que tiene registrada.
- Set_empleado(): Recibe variables referidas a todos los campos de la tabla empleado e inserta un nuevo registro en la base de datos.
- Borrar_empleado(): Elimina un registro de la base de datos cuando el campo DNI coincide con la variable recibida.
- Mod_empleado(): Actualiza los campos de la tabla empleado con la información que se ha recibido, salvo el identificador, que se recibe como variable.
- Actualiza_posicion(): Recibe el identificador de un empleado y una nueva posición de la secuencia, y procede a actualizar la posición que había registrada con la recibida.

4.3.5 Clase horario

Es la clase más extensa, dado que tiene la tarea más compleja: la generación del cuadrante. Al igual que las demás, solo define una variable, que es necesaria para la conexión. Sus métodos quedan definidos a continuación.

- __construct(): Idéntica a las anteriores.
- Get_dia(): Recibe el identificador de un empleado y una fecha y devuelve todos los registros encontrados donde los campos correspondientes coinciden.
- Set_horario(): Recibe los mismos datos que el método anterior, e inserta un nuevo turno para ese día. Este método es creado para actualizar el campo de un empleado de baja que ya ha sido suplido, y sustituye la palabra "Baja" que hay almacenada en el campo "turno" de ese día por la palabra "---", de forma que el sistema pueda interpretar que la baja ha sido cubierta.
- Asigna_dia(): Inserta en la tabla horario toda la nueva información que es recibida como variables, en este caso una para cada campo existente.
- Borrar_registros(): Recibe el mes para el cual se quiere generar un nuevo cuadrante y procede a eliminar los registros existentes de la tabla horario para ese mes.
- Comprueba_existe(): Recibe el mes a generar y devuelve verdadero o falso según se encuentre al menos un registro para ese mes.
- Imprime_horario(): Devuelve mediante un array asociativo y numérico todos los registros de la tabla horario que se tienen para un empleado cuyo identificador recibe el método, y para el mes de la fecha que recibe, ordenados ascendentemente según el día.
- Imprime_bajas(): Devuelve todos los registros que se tienen de la tabla horario donde el turno asignado es igual a la palabra "Baja".
- Imprime_horas(): Recibe el identificador de un empleado y una variable que representa un mes, y devuelve la suma de horas trabajadas para ese empleado en ese mes.
- Cubre_baja(): Recibe una fecha, una letra que representa el turno a cubrir y un identificador de empleado. Este método es llamado cuando el administrador ha decidido el empleado que va a cubrir una baja, y asigna el turno y horas correspondientes para ese empleado en ese día concreto.
- Truncar_tabla(): Elimina todos los registros de la tabla horario. Sirve para inicializar el calendario laboral.
- Puede_cubrir(): Este método recibe la letra a cubrir, la posición del empleado seleccionado y el tipo de turno a cubrir, y evalúa si el empleado seleccionado puede cubrir la baja que se está intentando cubrir cumpliendo las condiciones establecidas anteriormente.
- Letra_cubrir(): Recibe una fecha, la posición a cubrir y el tipo de turno. Devuelve la letra a cubrir

que se utilizaba en el método anterior. Esta función es necesaria debido a que en la tabla horario cuando un empleado está de baja se le asigna la palabra “Baja”, por lo que no se puede obtener información del turno que debía hacer y no está haciendo. Mediante este método se obtiene ese turno, representado por una letra, que se utilizará para evaluar si un empleado puede o no cubrir una baja y posteriormente para asignárselo a quien la cubra.

4.4 Trazabilidad del diseño

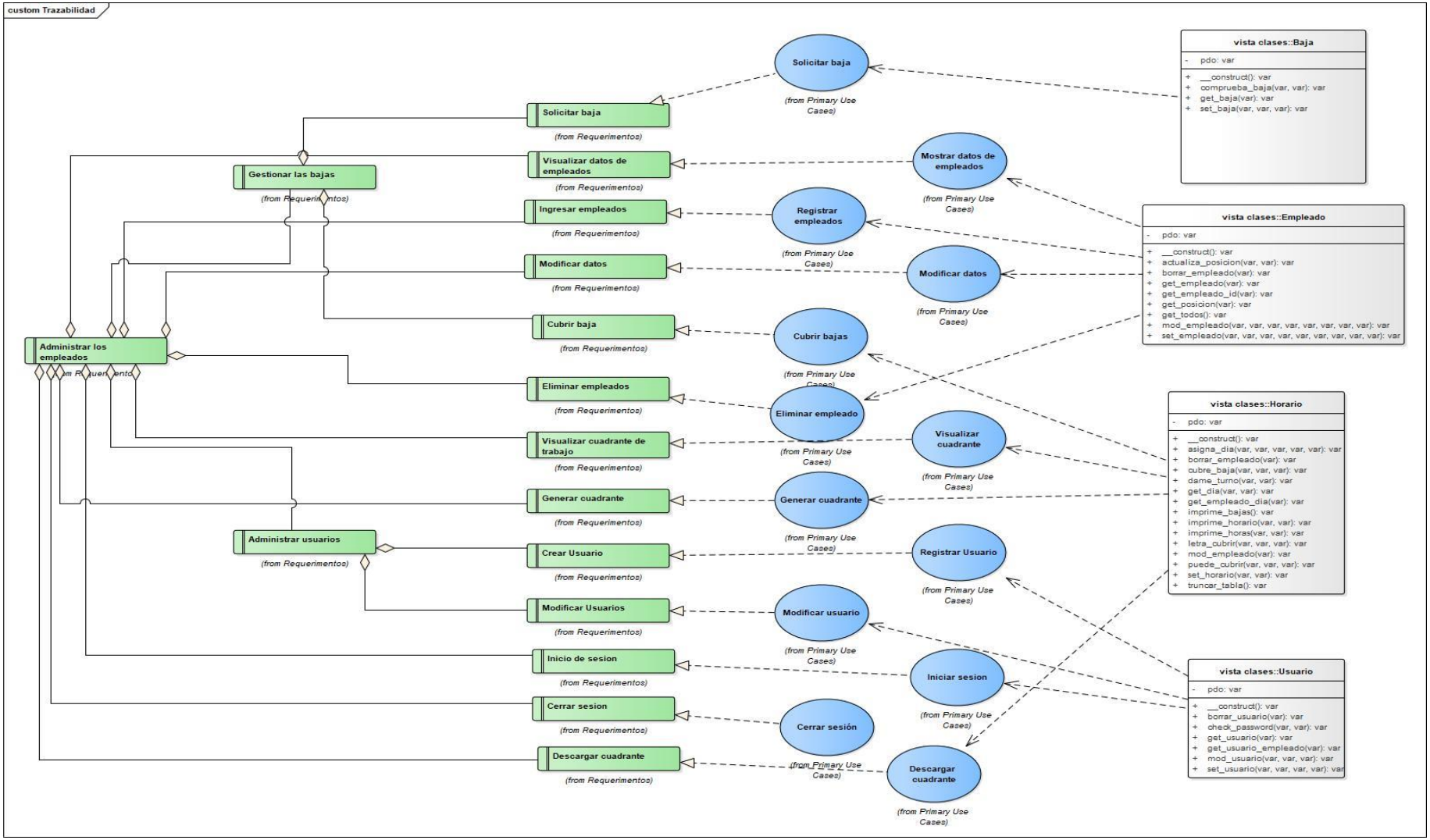
Una vez quedan definidas las clases, se puede observar la trazabilidad completa de los requisitos, casos de uso y clases, que en su conjunto modelan el problema y permiten comprobar que todo lo que se desarrolla no queda “huérfano” ni sobra.

Se puede comprobar como todas las clases participan activamente en el desarrollo de los casos de uso, que a su vez satisfacen los requisitos funcionales establecidos.

El diseño real suele presentar más de una clase asociada a un caso de uso, pero para mayor comprensibilidad se ha optado por representar la unión con la más relevante. Se puede apreciar como el CU-012 (Cerrar sesión) no está asociado. Esto se debe a que su implementación ha sido desarrollada mediante las funciones predefinidas de manejo de sesión que PHP incluye en sus librerías.

En la siguiente página puede observarse el diagrama de trazabilidad.

Figura 4-4. Diagrama de trazabilidad



4.5 Mapa de navegación

Los mapas de navegación aportan información visual sobre la implementación del sistema, permitiendo modelar el funcionamiento de la página web diseñada.

El mapa desarrollado para el sistema propuesto es relativamente extenso, por lo que será dividido para una mejor comprensión.

4.5.1 Inicio de sesión

A partir de la página de inicio, el usuario podrá acceder al inicio de sesión, y posteriormente a la página principal de cada tipo de actor según corresponda. En esta sección, por tanto, se aplica el caso de uso CU-001 (Inicio de sesión).

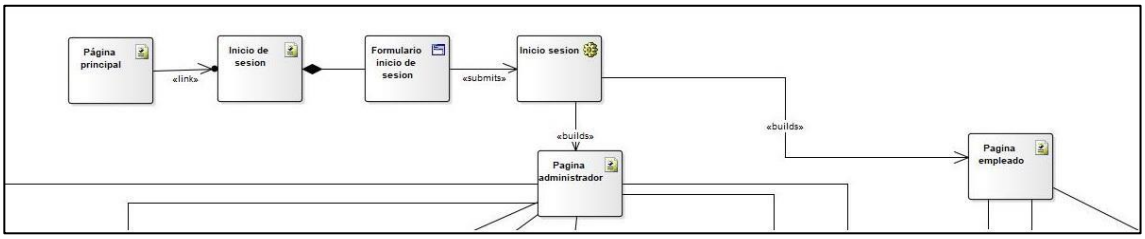


Figura 4-6. Mapa de Inicio de Sesión

4.5.2 Página de administrador

Una vez ha accedido a la página principal, el administrador encontrará disponibles las páginas que corresponden a sus funcionalidades. Esta sección contiene los casos de uso referentes a dar de alta un usuario (CU-003), generar un nuevo cuadrante (CU-005), dar de alta un empleado (CU-002), mostrar los datos de los empleados, modificarlos y eliminarlos (CU-004, CU-009 y CU-012), solicitar bajas (CU-008), modificar su usuario (CU-010) y mostrar el cuadrante (CU-007). A pesar de que este último caso de uso es igual para los dos actores, en el caso del administrador la visualización es diferente, dado que desde ella se accede a las pantallas necesarias para cubrir las bajas y descargar el cuadrante, que implementan el caso de uso CU-011 y CU-006, respectivamente. Por último, puede cerrar sesión (CU-013).

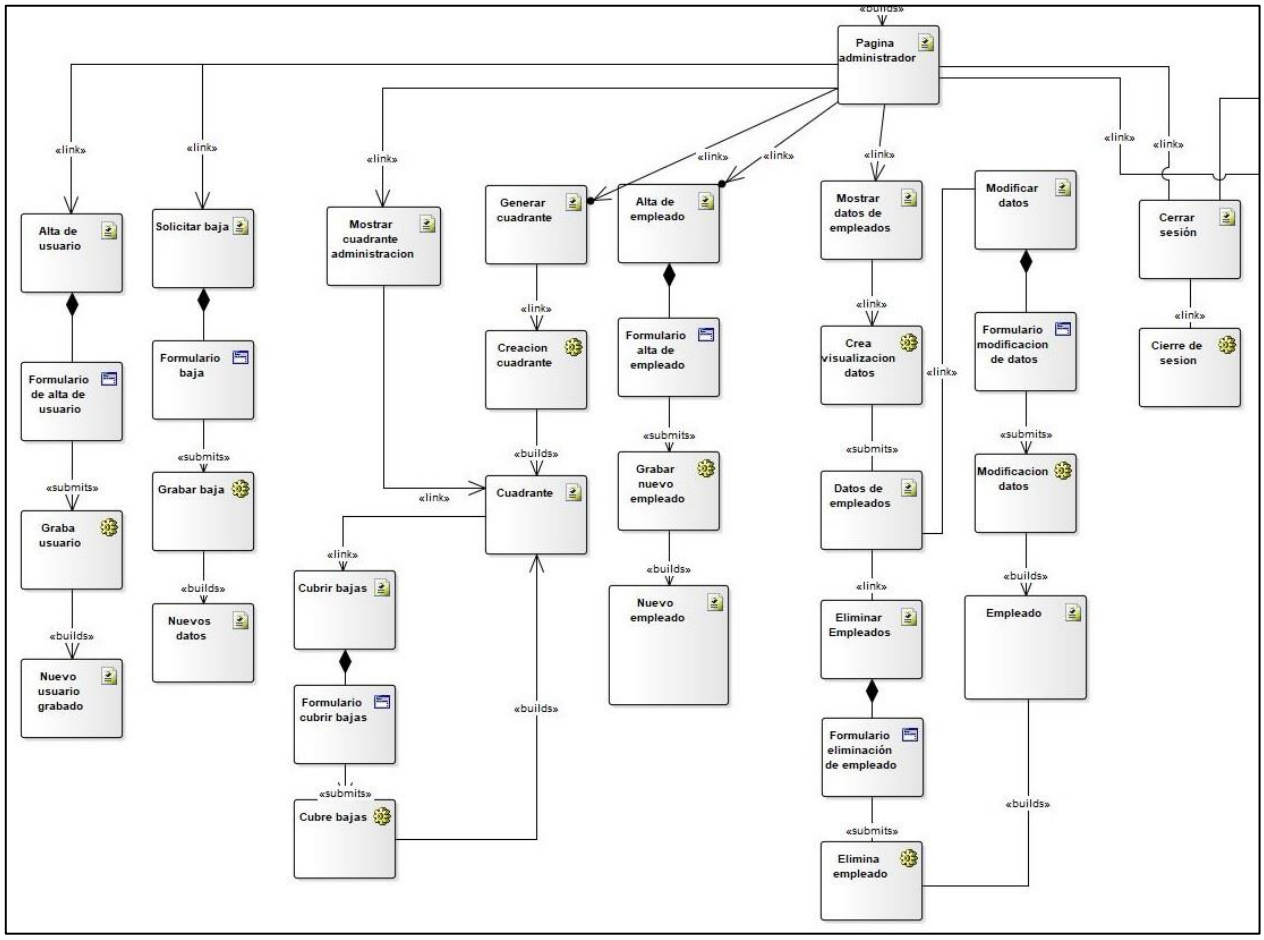


Figura 4-7. Mapa de administración

4.5.3 Página de empleado

Desde la página principal del empleado, el actor correspondiente podrá hacer uso las funcionalidades de las que dispone. Para ello se le mostrarán una serie de enlaces que le redirigen a las visualizaciones de modificar su usuario (CU-010), mostrar el cuadrante y descargarlo (CU-004 y CU-006), solicitar la baja (CU-008) y cerrar sesión (CU-013).

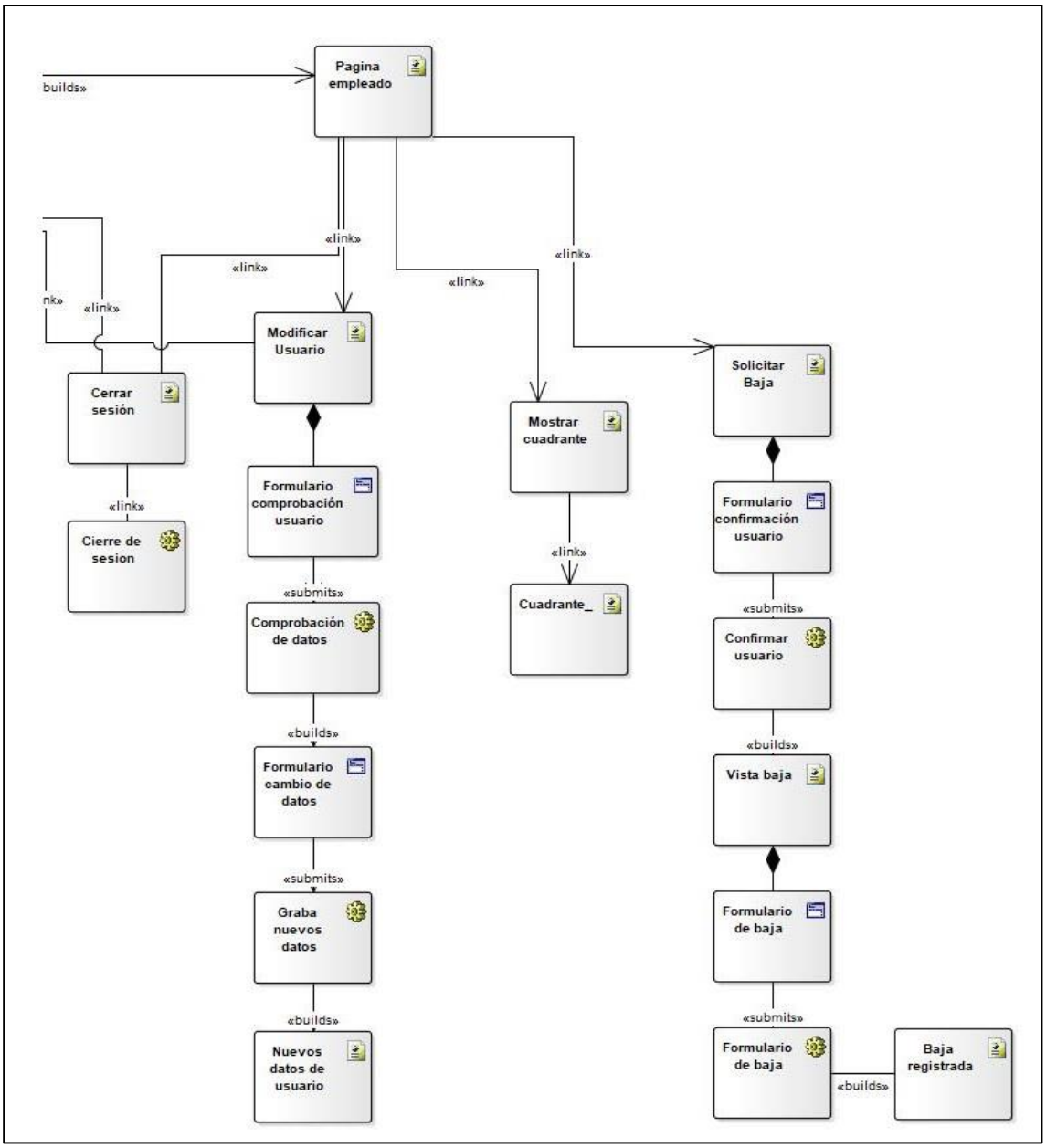


Figura 4-8. Mapa de empleado

5 IMPLEMENTACIÓN

Una vez realizado el modelo que permitirá resolver el problema planteado, se realiza la implementación de la solución mediante las herramientas que se comentaron en capítulos anteriores. En este apartado se desarrollará la solución encontrada. Para ello se expone la estructura del código y un manual de usuario que ofrezca a quién se enfrente al sistema de información el aprendizaje necesario para manejarlo de forma eficaz.

5.1 Estructura del código

Como se ha comentado anteriormente, se ha seguido el patrón MVC para el desarrollo web. El código, por tanto, se ha dividido en Vistas, Modelos y Controladores, cada uno almacenado en su propio directorio. El usuario navegará por las Vistas, ejecutando por el camino el resto de código. A continuación, se desglosan los distintos archivos según su clasificación en MVC. Además de esta distribución, en el directorio padre se encuentra el archivo Index.php, que es el primero que se ejecuta.

5.1.1 Controladores

Los controladores quedan definidos por una coletilla al final de su nombre de archivo, la palabra "Controller". Se encuentran en un subdirectorio llamado "controladores".

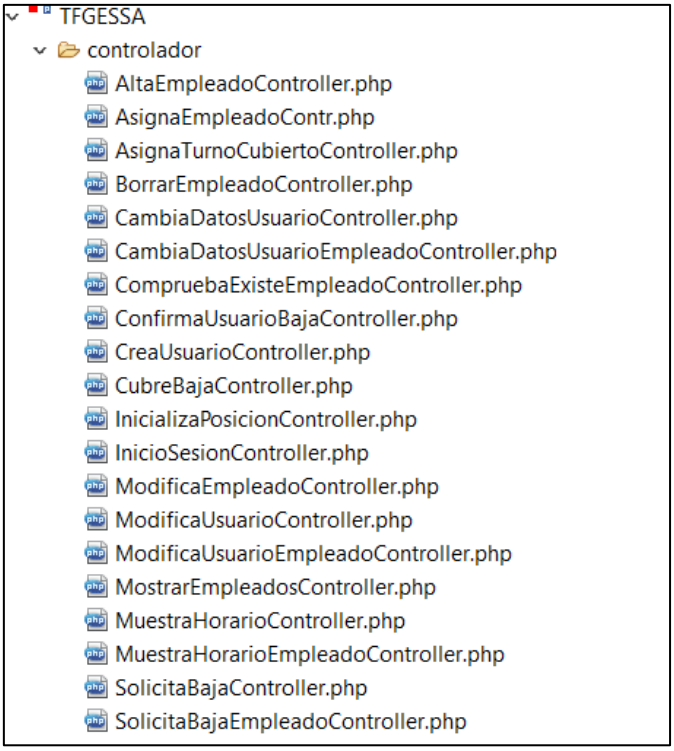


Figura 5-1. Controladores

5.1.2 Modelos

Los modelos han sido expuestos anteriormente, ya que codifican las clases implementadas. Se encuentran en dos directorios distintos, uno para la conexión con la base de datos y otro para los demás. Los modelos se identifican con la palabra “model”.

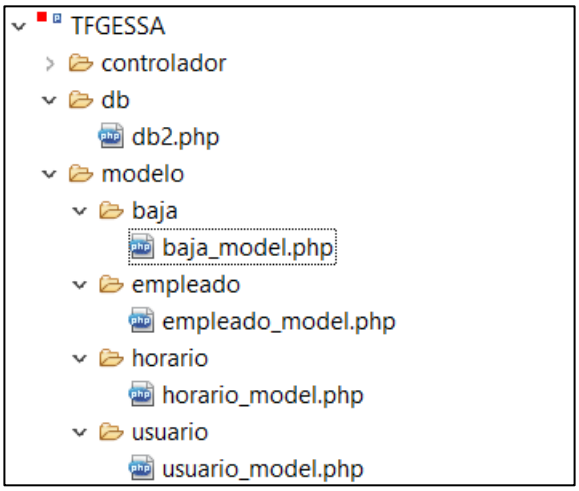


Figura 5-2. Modelos

5.1.3 Vistas

Las vistas se identifican con esa misma palabra, y corresponden con el mapa de navegación.

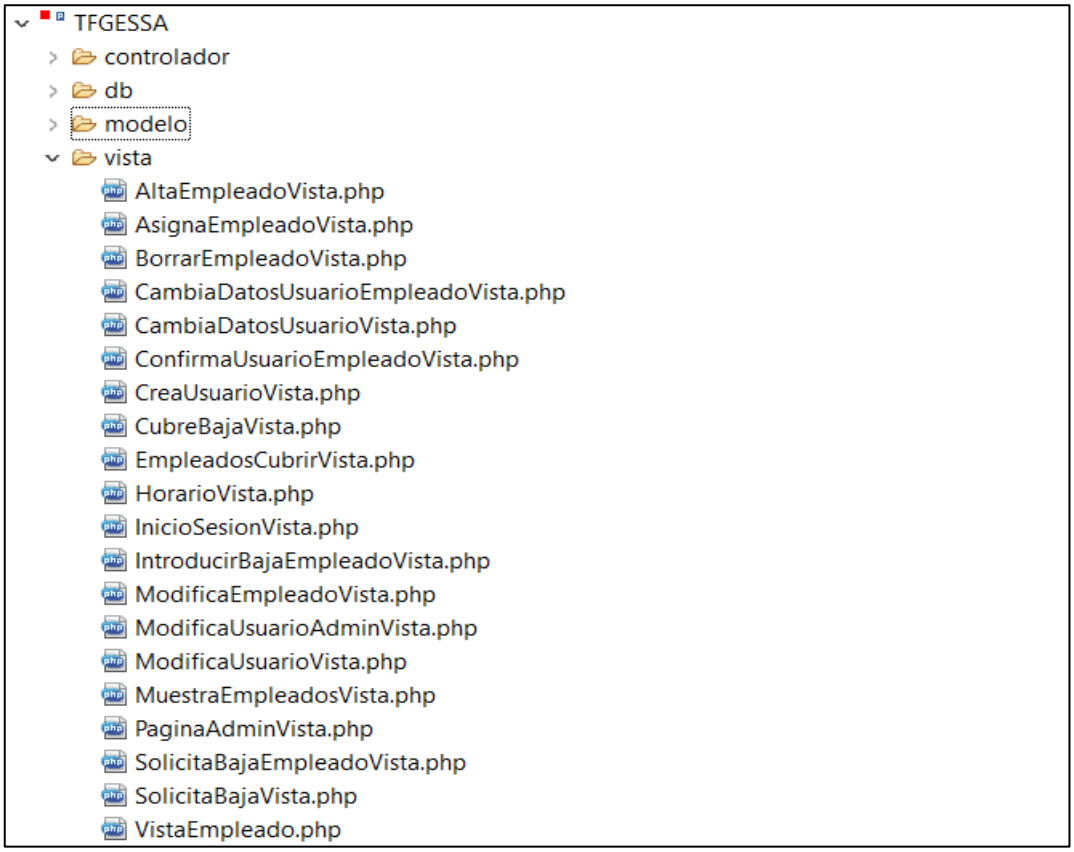


Figura 5-3. Vistas

5.2 Estructura de la base de datos

Mediante phpMyAdmin se puede acceder directamente a la base de datos. En esta página se pueden ver los privilegios de acceso y demás datos que permiten conecta el código con el sistema gestor. La base de datos creada está nombrada como "tfgessa5", y mediante el usuario creado como "Root" se tienen privilegios de administración

Figura 5-5. Usuarios con acceso a la base de datos

También se puede ver en detalle la estructura de una tabla, como se muestra en la siguiente imagen:

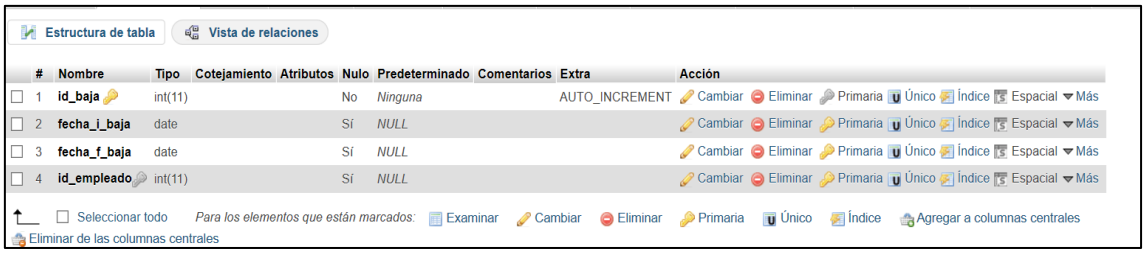


Figura 5-4. Detalle de tabla

Desde esta página podemos acceder a diversas funcionalidades, como modificar la estructura de las tablas o de los datos, exportar e importar el código SQL, probar instrucciones en línea u observar el diagrama de relación entre las tablas, como se puede observar a continuación:

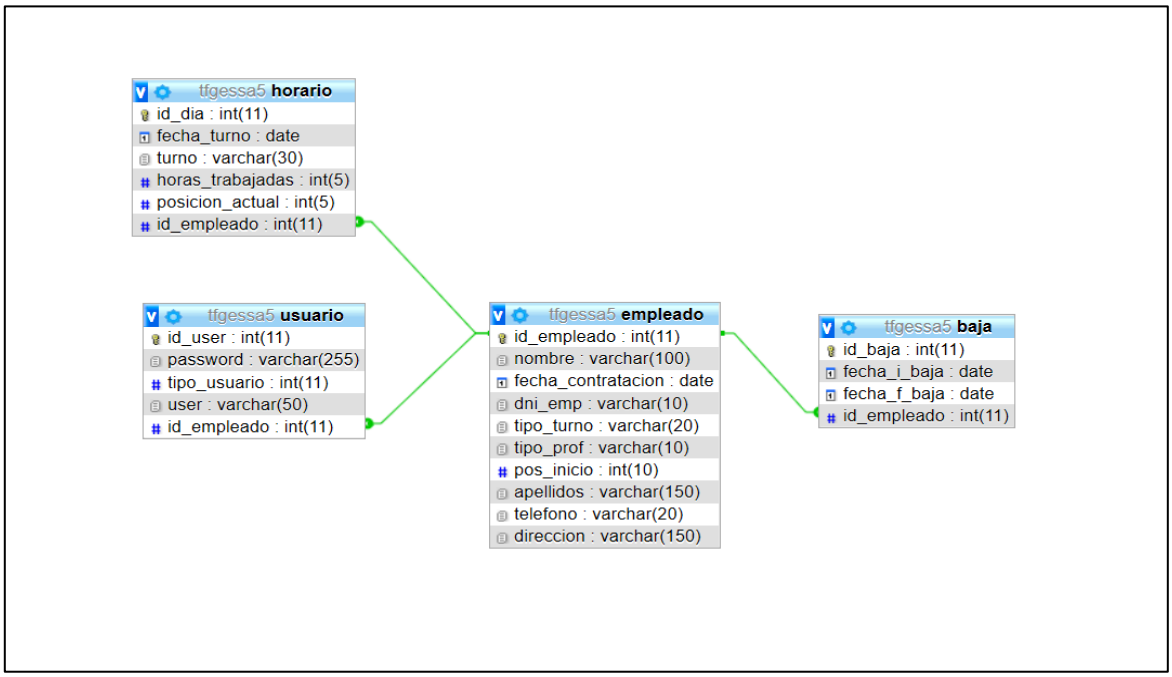


Figura 5-6. Relaciones entre tablas

5.3 Manual de Usuario

En este apartado quedará expuesto un recorrido a través de todas las pantallas que ambos actores tienen disponibles, comentando brevemente la forma de codificarlas. Este recorrido se realizará atendiendo a las divisiones del mapa de navegación.

5.3.1 Inicio de sesión

Esta parte es común para ambos actores, y permite acceder a la pantalla principal de cada uno de ellos. Comienza al ejecutar el archivo Index.php, que redirige automáticamente a la Vista de Inicio de sesión. El usuario introducirá sus credenciales de acceso, y el sistema podrá o bien permitirle avanzar hacia su página correspondiente o mostrar un mensaje de error y devolverle al formulario.

Figura 5-7. Inicio de sesión

Cuando el formulario es enviado, se ejecuta el controlador correspondiente, que evalúa si las credenciales introducidas coinciden con algunas que se encuentran almacenadas en la base de datos. Como se ha expuesto anteriormente, la contraseña permanece encriptada. Para permitir el acceso, se comparan la contraseña encriptada almacenada con una encriptación realizada in situ de la contraseña introducida en el formulario. Si no coinciden, se muestra la siguiente pantalla, desde la que se puede volver a intentar acceder.

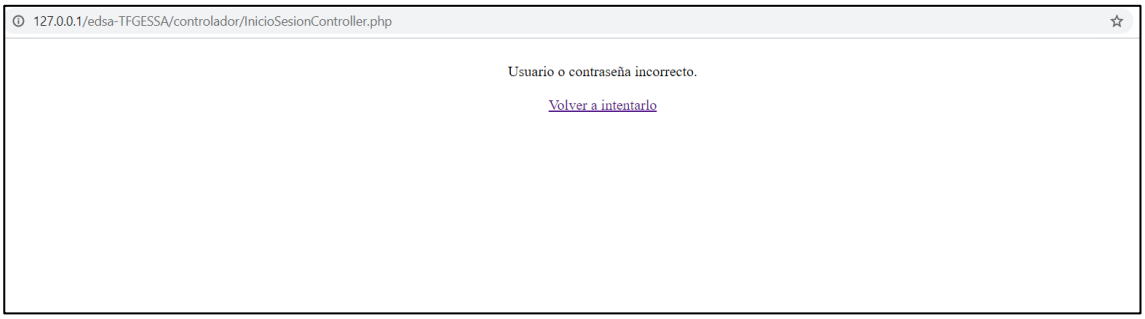


Figura 5-8. Error de inicio de sesión

Si son correctas, el sistema redirige hacia la página de inicio correspondiente.

5.3.2 Página de Administrador.

Figura 5-9. Página principal de administrador

Si el tipo de usuario que ha ingresado en el sistema es un Administrador, el enlace redirige hacia la vista de la página principal de este actor. En ella se encuentran distintos enlaces que refieren a las páginas a las que tiene acceso.

5.3.2.1 Visualizar datos y modificar empleados.

Si el administrador escoge ver datos y modificar los empleados, será redirigido a la siguiente pantalla, donde se exponen los datos de los empleados.

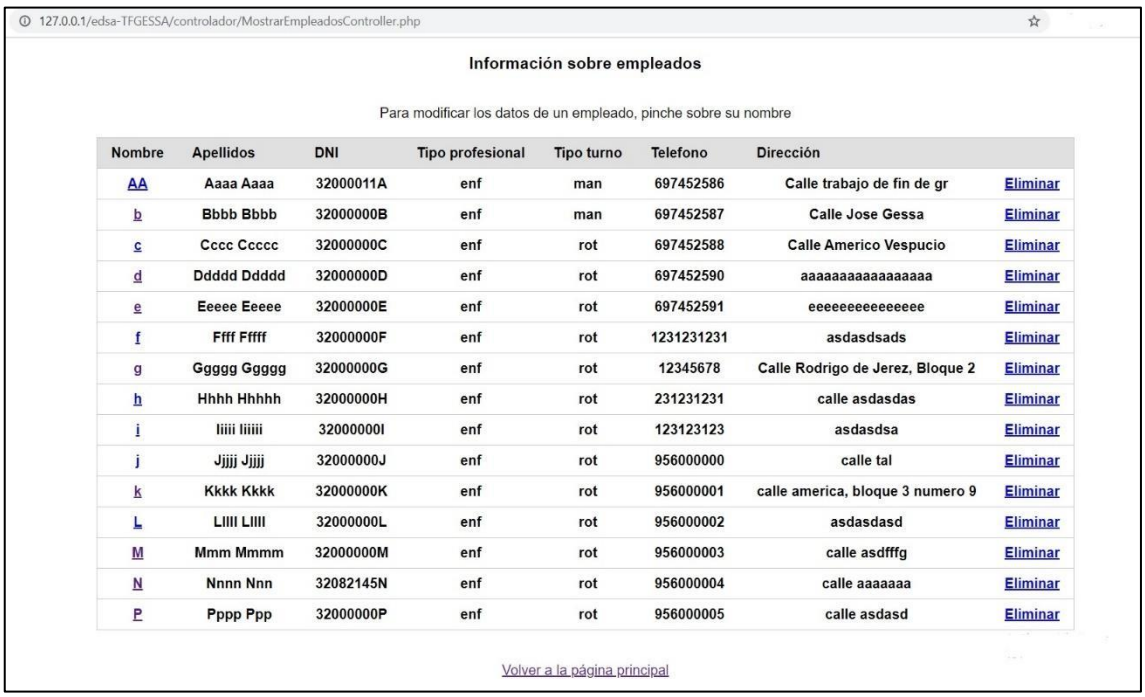


Figura 5-10. Vista de empleados

Desde esta pantalla el usuario puede volver a la página principal mediante el enlace situado en la parte inferior. Si desea modificar los empleados, puede hacerlo pinchando sobre el nombre del empleado en concreto, que redirige hacia la siguiente página:

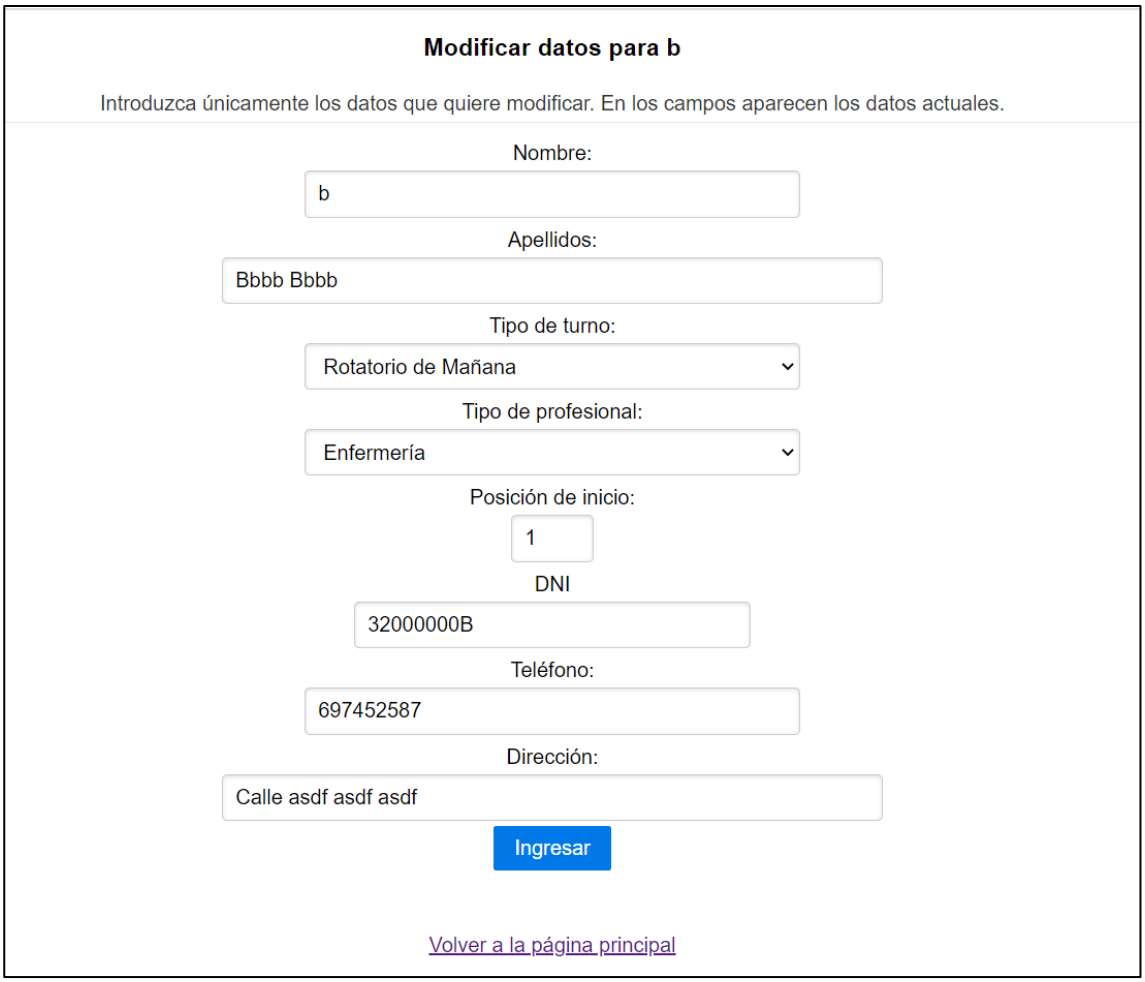


Figura 5-11. Modificación de empleado

Esta página contiene algo de información para recordar el proceso a ejecutar cuando se están modificando los datos. En el título aparece el nombre del empleado que va a ser modificado, y en cada uno de los campos del formulario aparecen los registros que existen actualmente en la base de datos. Si el administrador no modifica algún campo, no será actualizado. Desde esta pantalla puede ejecutar la orden de modificación o volver a la página principal. Si finalmente decide modificar algún dato, el sistema ejecutará el controlador, los cambios quedarán registrados en la base de datos, y el sistema mostrará la siguiente pantalla, desde donde se puede volver a la página de visualización de datos o la principal:

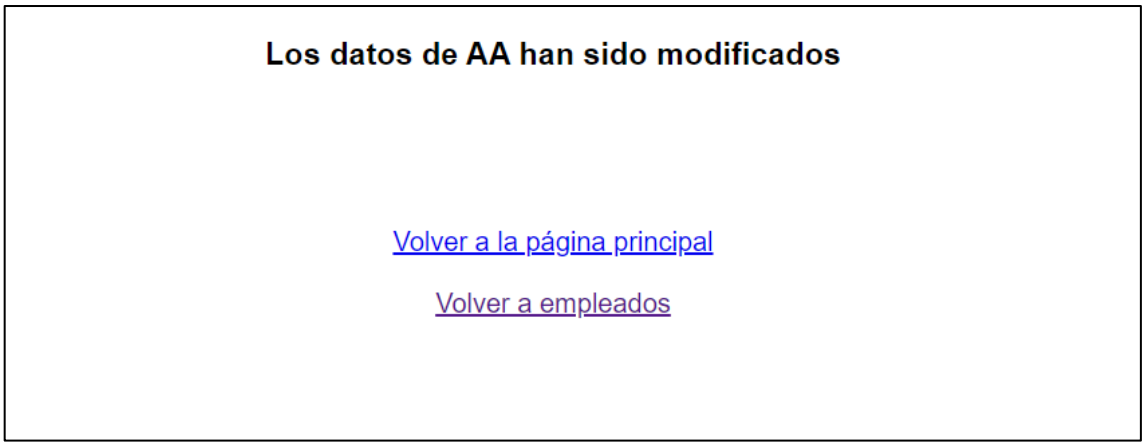


Figura 5-12. Confirmación de modificación de datos

Si el administrador pincha sobre la palabra "Eliminar", que corresponde con el empleado expuesto en la misma fila, el sistema mostrará un mensaje de confirmación. Si se valida la decisión, el sistema eliminará el registro de la base de datos y mostrará un mensaje similar al anterior. En caso contrario, no se ejecuta ninguna acción.

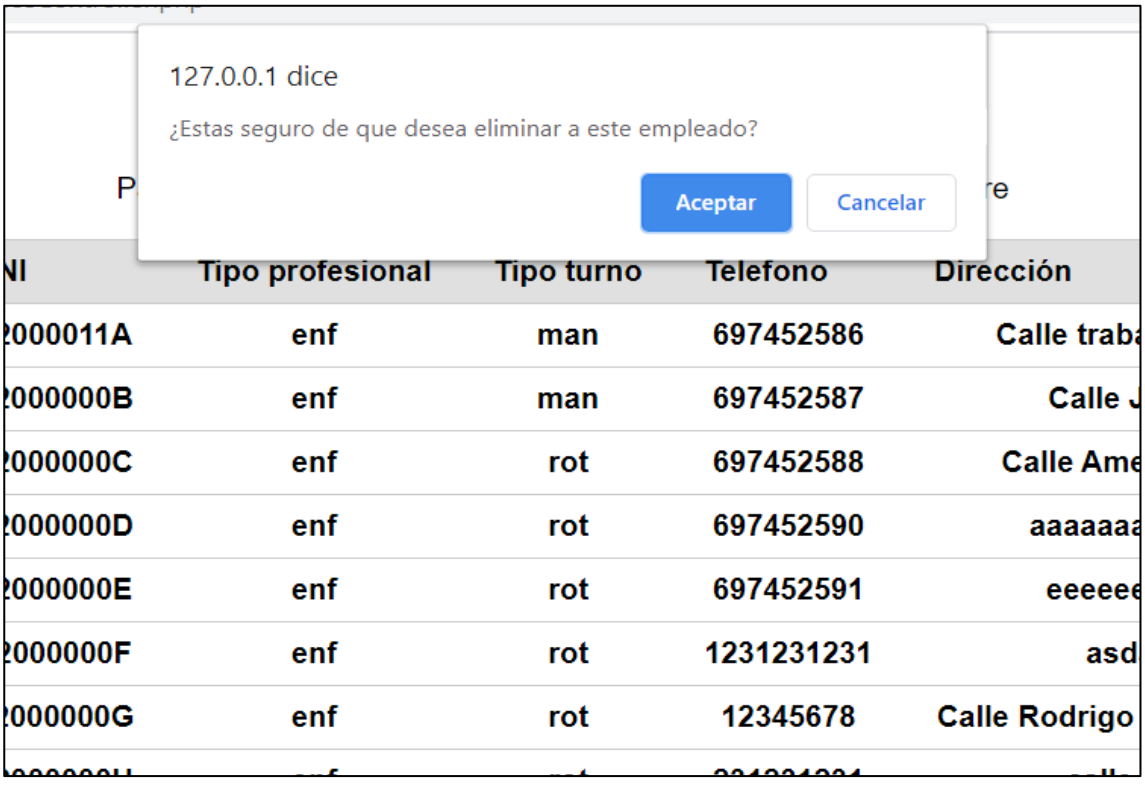


Figura 5-13. Alerta de eliminación de empleado

5.3.2.2 Ingresar Empleado

Cuando el administrador pincha sobre el enlace “Ingresar Empleado”, el sistema muestra el formulario de alta, donde el actor introducirá todos los datos. Es obligatorio rellenar todos los campos.

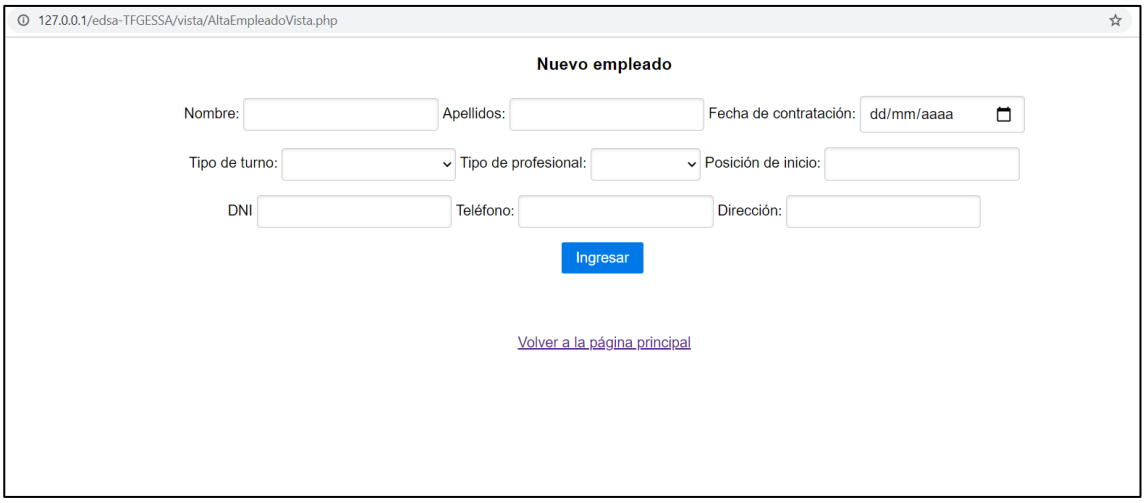


Figura 5-14. Ingresar empleado

La posición de inicio es un parámetro necesario para los administradores, ya que permite variar la secuencia que será asignada al empleado. Dependiendo del tipo de turno y profesional que sea el empleado, esta posición de inicio deberá tomar un valor u otro:

- Si el turno es rotatorio general, el número debe estar comprendido entre 0 y 10. Esto es así puesto que la secuencia rotatoria tiene 11 posiciones, siguiendo el patrón previamente descrito (día, noche, 3 descansos, día, noche, 4 descansos). La posición de inicio 0 corresponde al primer día y la 10 al último descanso.
- Si es rotatorio de mañanas, la posición de inicio debe ser 1 o 0. Así, si la posición es 1 corresponderá a empezar haciendo una semana en la que se trabaja lunes, miércoles y viernes, y si es 0 realizará la rotación complementaria.
- Si es auxiliar, la posición de inicio es indiferente

Este parámetro puede ser modificado en el formulario de modificación de datos. Si a la hora de generar un cuadrante la posición de inicio de un empleado no es válida, el sistema asigna una automáticamente (0 en ambos casos).

El campo relativo al DNI debe rellenarse con 8 dígitos y una letra, consecutivos. Si no coincide con este formato, el formulario no podrá ser enviado. Se muestra un recordatorio en el propio campo.

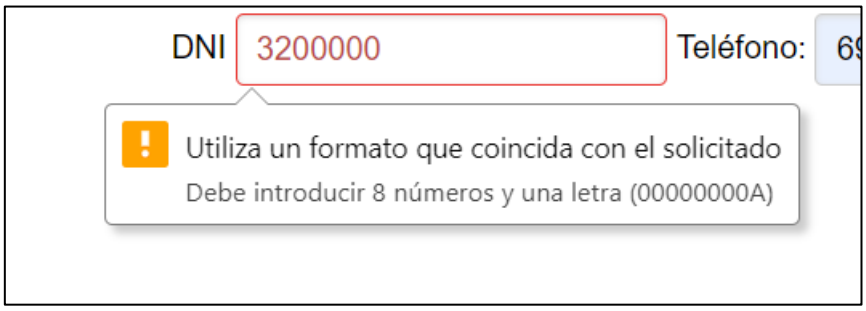


Figura 5-15. Comprobación DNI

Si se pulsa el botón de "Ingresar", se envía el formulario al controlador que procederá a registrar el empleado en la base de datos. Una vez ejecutado el código, se mostrará la siguiente pantalla, donde se dispone de la opción de seguir registrando empleados o de volver al menú principal.

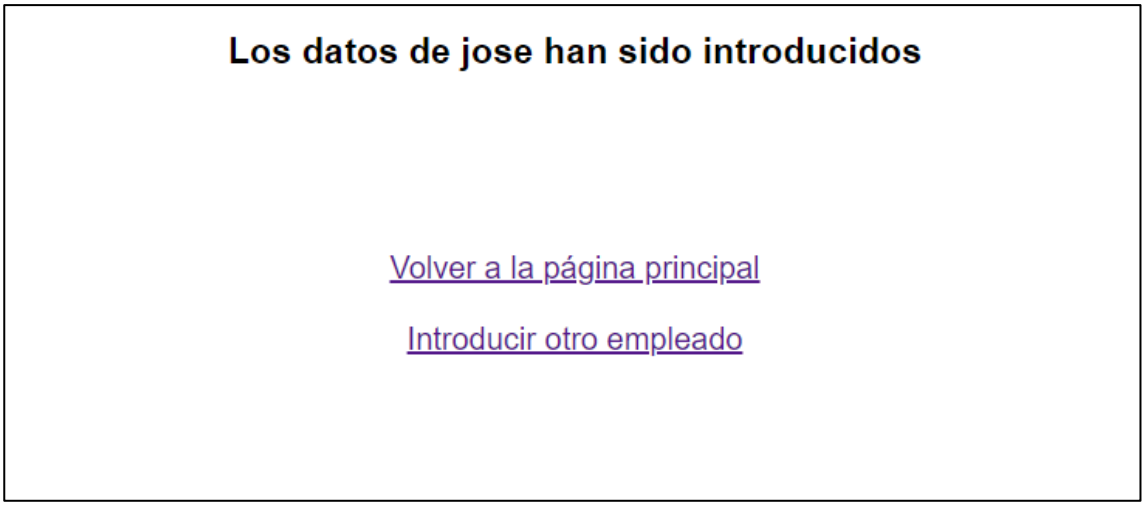


Figura 5-16. Datos introducidos

5.3.2.3 Crear nuevo usuario

Este enlace permite al administrador generar credenciales de acceso para un empleado, que posteriormente podrá modificarlas para mayor seguridad. Al escoger esta opción, el usuario deberá rellenar el siguiente formulario:

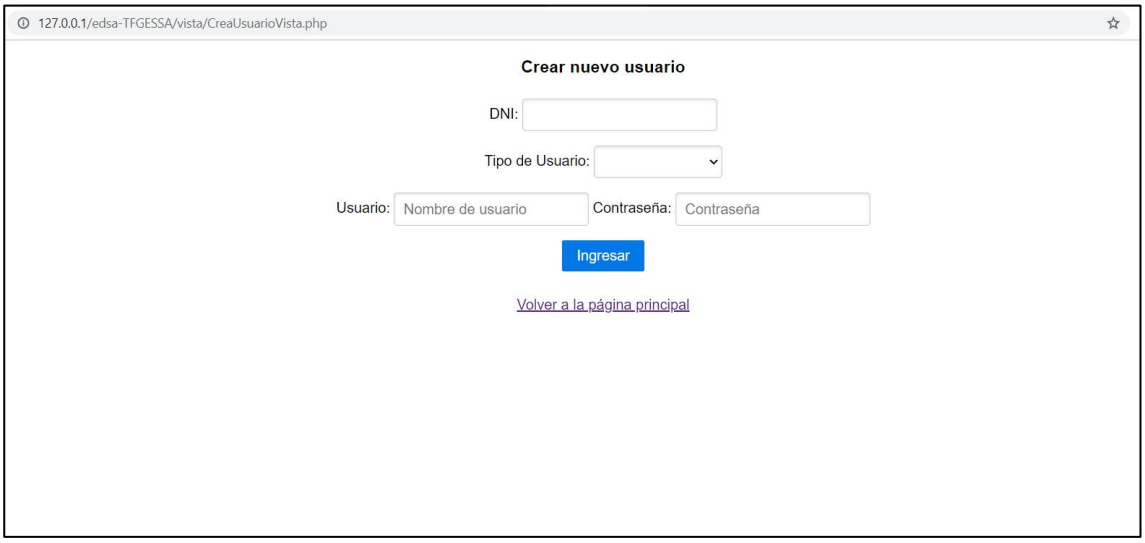


Figura 5-17. Creación de nuevo usuario

Donde el tipo de usuario se rellena mediante una lista desplegable con las opciones correspondientes (Administrador o Empleado). Una vez introducidos los datos, el formulario enviará los datos al controlador correspondiente, que comprobará y ejecutará distintas órdenes según se de uno de estos casos:

- Si el empleado ya tiene un usuario registrado, el sistema muestra un mensaje y presenta un enlace que redirige hacia el formulario.

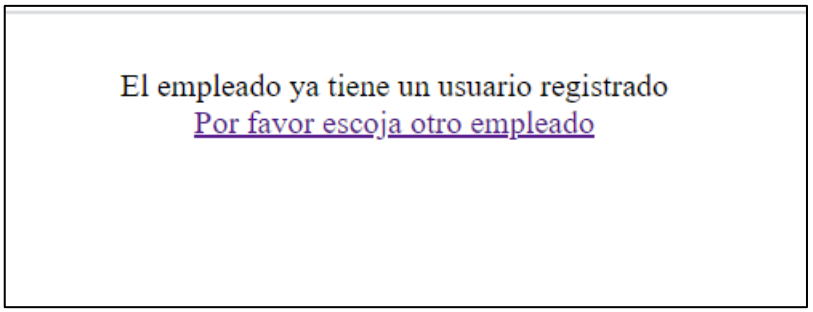


Figura 5-18.Usuario ya asignado

- Si el nombre de usuario ya existe en la base de datos, se muestra un mensaje y se da la opción de volver al formulario.

-

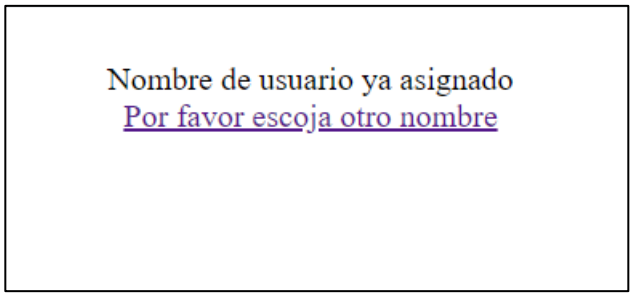


Figura 5-19. Nombre de usuario existente

- Si todas las comprobaciones son correctas, el sistema almacena el nuevo usuario y muestra un mensaje de confirmación

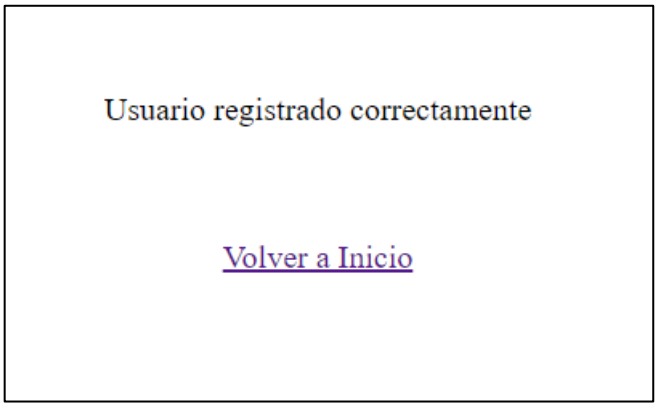


Figura 5-20. Usuario registrado

5.3.2.4 Solicitar baja

El administrador puede solicitar una baja, tanto para el mismo como para otros empleados. Cuando se requiere acceder a esta funcionalidad, el sistema muestra la siguiente pantalla (en el caso en el que la sesión iniciada tenga privilegios de administración):

Figura 5-21. Solicitar baja

El administrador rellenará los campos del formulario y lo ejecutará. Los empleados son redirigidos hacia un formulario similar, pero donde no tienen que introducir el DNI, ya que la sesión iniciada les reconoce. Es posible solicitar una baja de un solo día, introduciendo en los campos fecha de inicio y fecha de fin el mismo día. Cuando se pulse sobre el botón, puede suceder uno de estos dos casos:

- Tras comprobar el DNI, no se encuentra ningún empleado. El sistema muestra el siguiente mensaje, desde donde se puede volver:

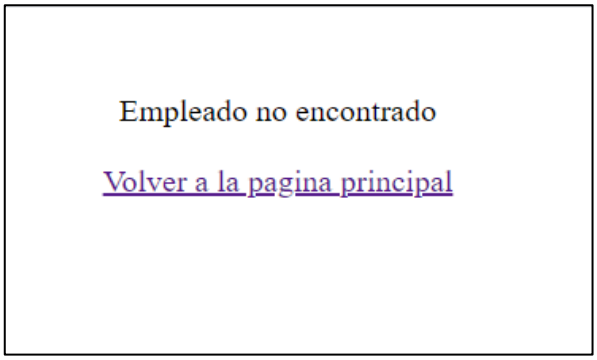


Figura 5-22. Empleado no encontrado

- Si el DNI es correcto, el sistema inserta la baja en la tabla e imprime un mensaje de confirmación

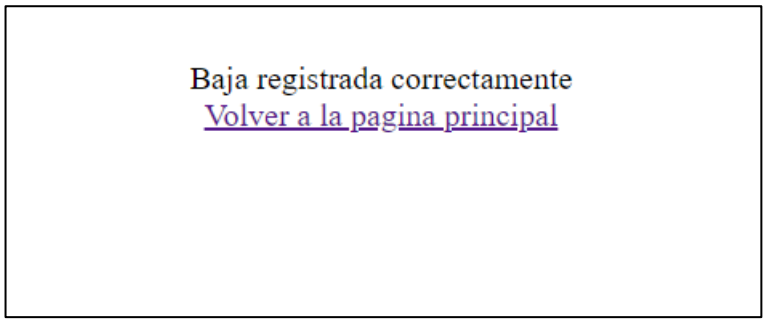


Figura 5-23. Baja registrada

5.3.2.5 Generar nuevo cuadrante

Mediante este enlace el administrador accede al formulario de generación de cuadrante, expuesto a continuación:

Figura 5-24. Generar nuevo cuadrante

En esta página se seleccionará el mes para el que se quiere generar un nuevo cuadrante. Al ingresar los datos, el sistema mostrará un mensaje para confirmar la decisión, dado que la generación de un calendario mensual sobrescribe todos los datos que hubiera anteriormente.

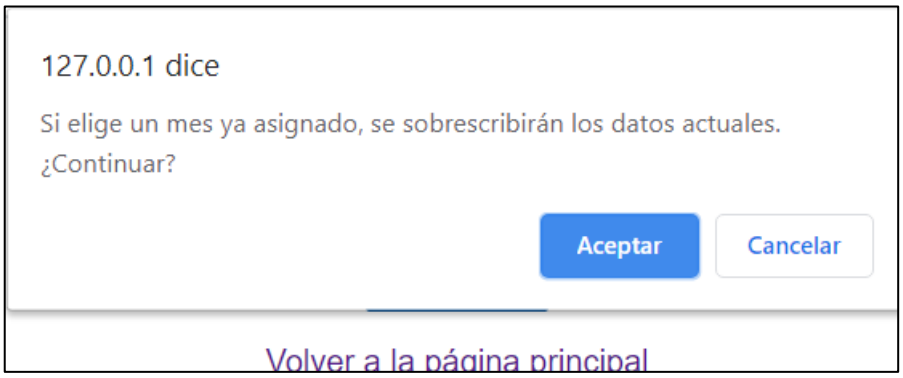


Figura 5-25. Alerta de sobreescritura

Si se decide eliminar todos los registros existentes, el sistema borrará de la base de datos cualquier horario que exista, de todos los meses de los que se tenga información.

Una vez se han introducido los datos, al ejecutar el formulario el sistema asignará los empleados existentes según sus secuencias y mostrará el siguiente mensaje:

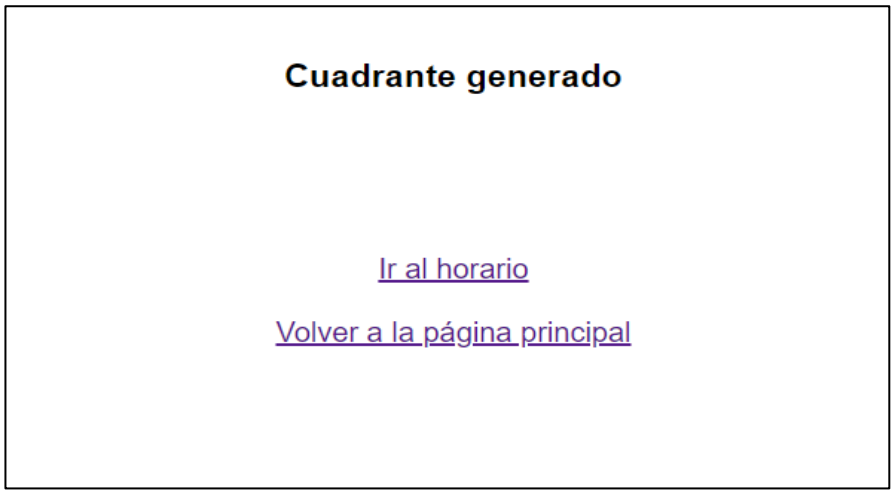
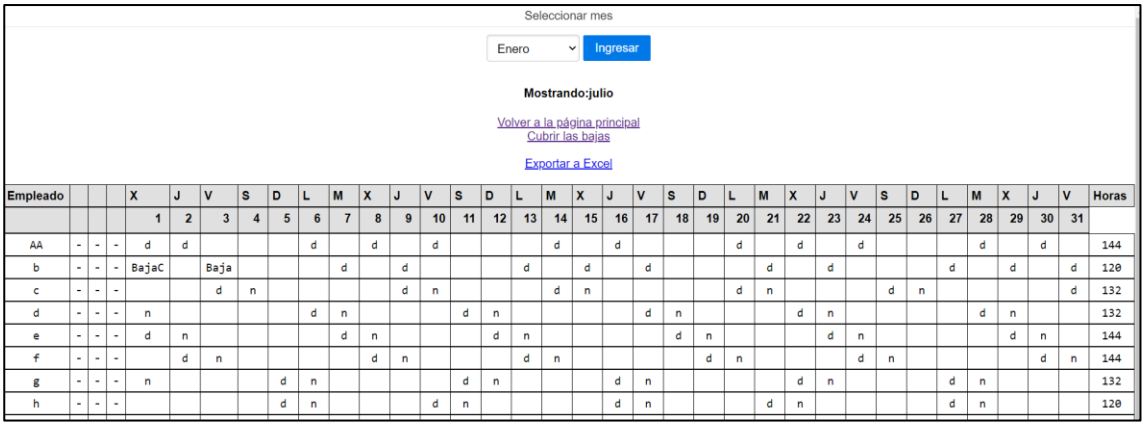


Figura 5-26. Cuadrante generado

Desde donde se puede enlazar directamente con el cuadrante.

5.3.2.6 Visualizar cuadrante

Esta opción permite visualizar el cuadrante actual, desde donde se puede cubrir las bajas si fuera necesario. Al acceder al enlace, el sistema muestra automáticamente la siguiente pantalla:



Empleado				X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	Horas
				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
AA	-	-	-	d	d				d		d		d				d		d				d		d		d				d		d		144
b	-	-	-	BajaC		Baja				d		d				d		d		d				d		d				d		d		d	120
c	-	-	-			d	n					d	n				d	n					d	n				d	n					d	132
d	-	-	-	n					d	n				d	n					d	n				d	n				d	n			132
e	-	-	-	d	n					d	n				d	n					d	n				d	n					d	n		144
f	-	-	-		d	n					d	n				d	n					d	n				d	n					d	n	144
g	-	-	-	n				d	n					d	n				d	n					d	n				d	n				132
h	-	-	-					d	n				d	n					d	n				d	n					d	n				120

Figura 5-27. Cuadrante

Por defecto, el primer mes que se muestra es el actual según la fecha del dispositivo.

Este cuadrante se divide en los siguientes apartados:

- En la parte superior encontramos un pequeño formulario para seleccionar el mes mediante una lista desplegable, que al enviar la información cambiará el mes que se muestra. Justo debajo se puede ver el mes que se está mostrando actualmente. También aparece un enlace para volver a la página principal y otro para cubrir las bajas que existan.
- Las dos filas superiores del cuadrante contienen una letra que representa el día de la semana y el día del mes correspondiente. Estos campos son calculados por el controlador.
- La parte inferior de la tabla representa los turnos de los empleados mediante una letra. La primera columna contiene los nombres de los empleados, seguido por un número de guiones correspondiente a los días que no proceden, por ser de un mes anterior. Los turnos están representados de la misma forma en la que funcionaba el sistema anterior de gestión, con una letra "d" minúscula para representar un turno diurno de 12 horas, "n" minúscula para los turnos de noche de 12 horas, "fm" para los turnos fijos de mañana, "Baja" para los empleados de baja y "BajaC" para las bajas que están cubiertas. La última columna representa las horas trabajadas en ese mes por cada empleado.

Si el administrador necesita cubrir las bajas, accederá a la siguiente página:

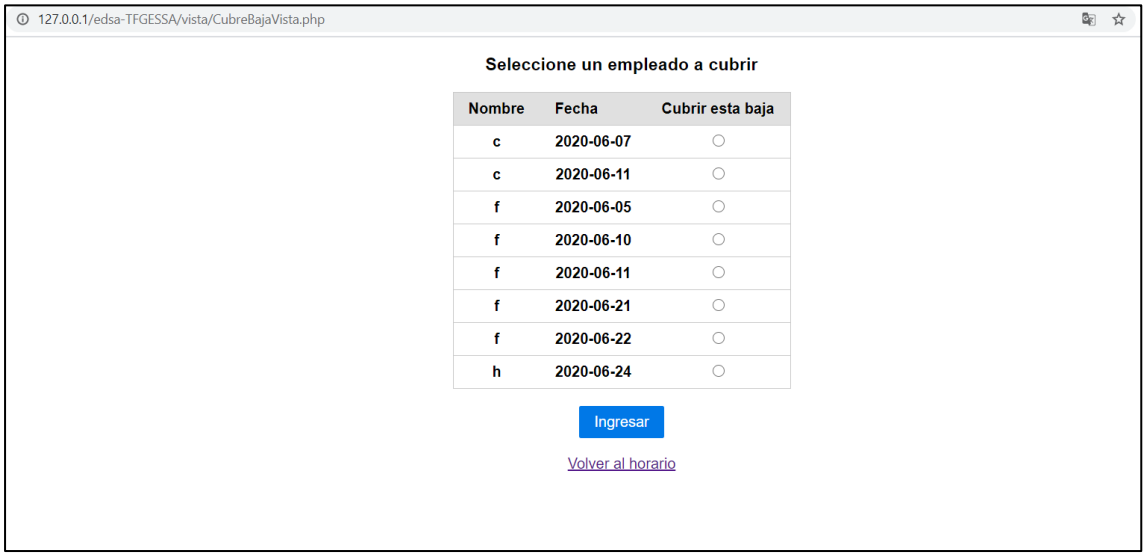


Figura 5-28. Cubrir las bajas

En esta vista se muestra una tabla en la que se expone el nombre de los empleados que está de baja en algún momento, la fecha y un selector. De esta forma, el administrador puede visualizar rápidamente las bajas existentes y seleccionar la que quiere suplir. Al pinchar en Ingresar, el sistema redirige hacia otra pantalla donde se podrá decidir quién cubre la baja:

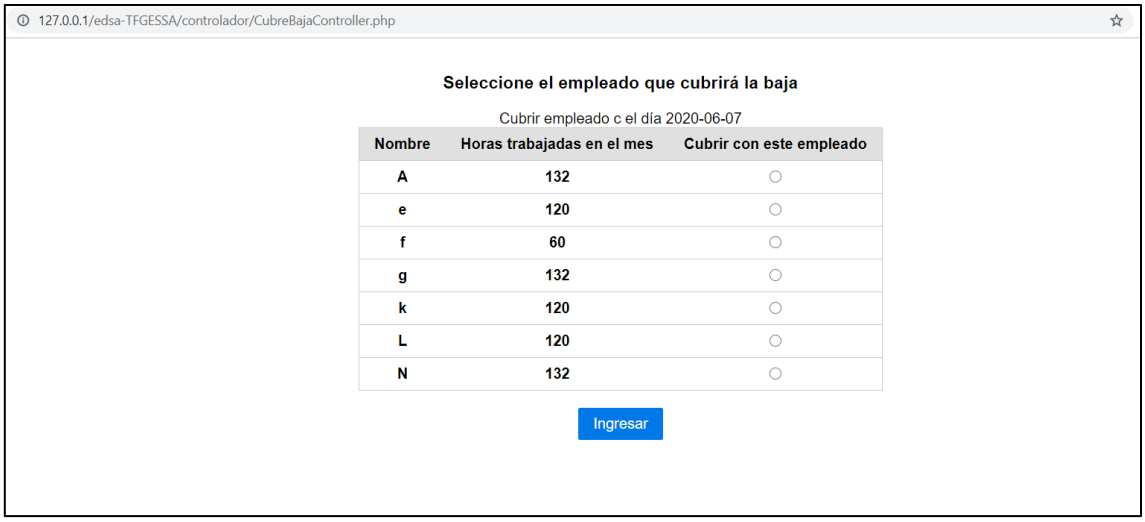


Figura 5-29. Selección de empleado

Esta pantalla muestra los empleados que pueden cubrir la baja seleccionada, siguiendo los requisitos establecidos que han sido comentados en la definición del caso de uso correspondiente. Justo encima de la tabla se puede observar un pequeño texto que recuerda el empleado y el día que se está supliendo. En la tabla de los candidatos se puede ver el nombre, las horas trabajadas ese mes y el mismo selector que en la página anterior. Cuando se escoge al trabajador que va a cubrir la baja y se presiona el botón correspondiente, se muestra el siguiente mensaje:

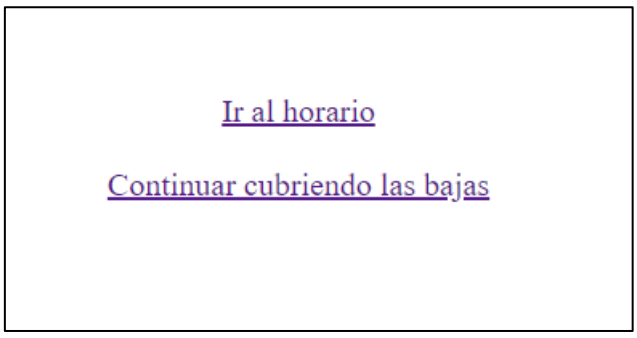


Figura 5-30. Baja cubierta

De esta forma, el administrador puede seguir cubriendo bajas o volver al horario.

Si se pincha sobre el enlace “Exportar a Excel”, el sistema descargará automáticamente un archivo de tipo hoja de cálculo que contiene el cuadrante correspondiente.

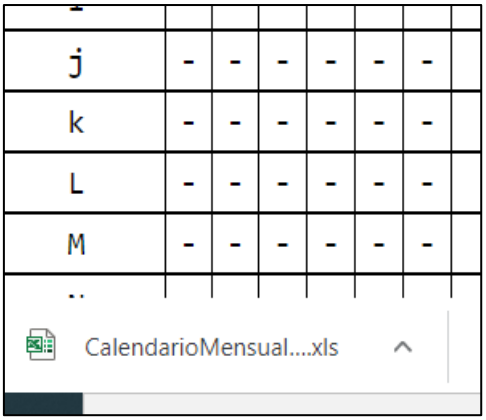
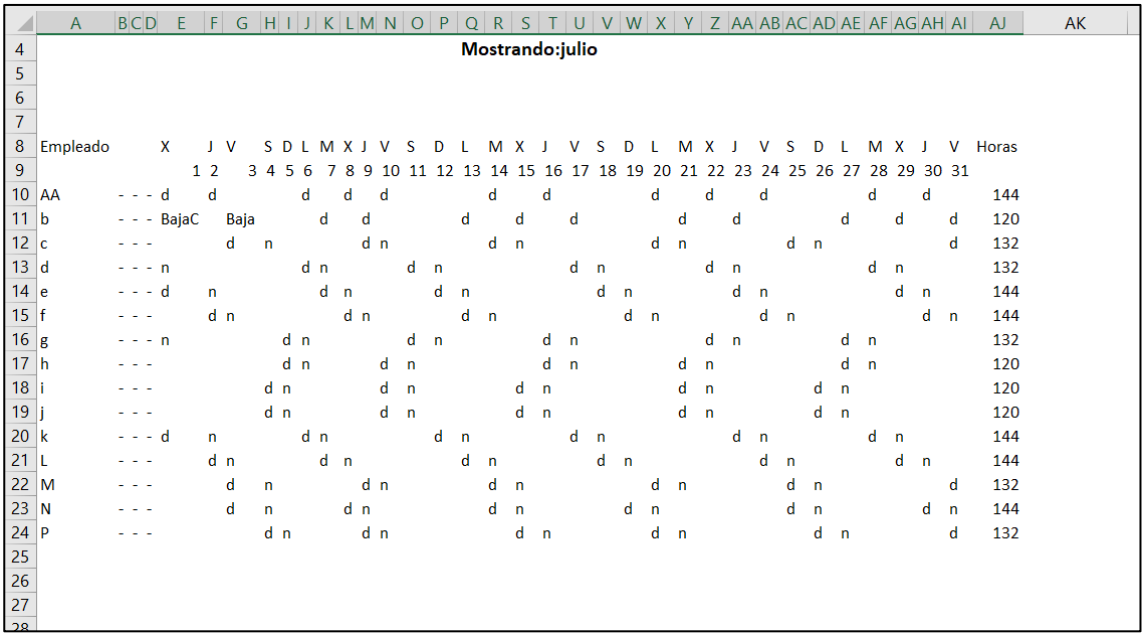


Figura 5-31. Descarga de hoja de cálculo

Esta funcionalidad permite tanto a administradores como a empleados descargar y almacenar en donde quieran el calendario correspondiente. A continuación, se muestra una imagen del archivo obtenido.



Mostrando:julio

Empleado	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	S	D	L	M	X	J	V	Horas
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
AA	d	d				d		d		d					d		d				d		d		d				d		d		144
b	BajaC		Baja				d		d				d		d		d					d		d				d		d		d	120
c		d		n					d	n					d	n					d	n				d	n				d	132
d	n					d	n				d	n						d	n				d	n					d	n			132
e	d		n				d	n				d	n						d	n				d	n					d	n		144
f		d	n					d	n				d	n						d	n				d	n					d	n	144
g	n					d	n				d	n				d	n						d	n				d	n				132
h						d	n			d	n					d	n					d	n					d	n				120
i					d	n				d	n					d	n					d	n				d	n					120
j					d	n				d	n					d	n					d	n				d	n					120
k	d		n				d	n				d	n					d	n					d	n					d	n			144
L			d	n				d	n				d	n					d	n					d	n					d	n		144
M				d		n				d	n				d	n					d	n				d	n					d		132
N				d		n			d	n					d	n				d		n				d	n					d	n	144
P					d	n				d	n					d	n					d	n					d	n				d		132

Figura 5-32. Archivo descargado

5.3.2.7 Cerrar sesión

Cuando se pincha sobre este enlace, el sistema cierra la sesión actual y redirige hacia la página de inicio. Esta funcionalidad es exactamente igual para el caso en el que el actor es un empleado.

5.3.3 Página de Empleado.

Si el usuario que inicia sesión es de tipo Empleado, se muestra la siguiente pantalla, desde donde se da acceso al resto:

Figura 5-33. Página de empleado

5.3.3.1 Solicitar Baja

Aunque la funcionalidad final es la misma que en el caso del Administrador, los empleados siguen un proceso distinto, dado que podrían introducir un DNI que no fuera suyo a la hora de rellenar el formulario.

Figura 5-34. Solicitud de baja de empleado

Similar a su homóloga para los administradores, esta página se diferencia en que no es necesario introducir el DNI, ya que el empleado que va a solicitar la baja ha sido identificado mediante su usuario. Cuando se introducen los datos, se imprime el mismo mensaje que en Figura 5-23. Baja registrada.

5.3.3.2 Visualizar cuadrante actual

El empleado podrá ver el cuadrante existente, de forma similar a la página mostrada para los administradores, salvo por la falta del enlace hacia cubrir las bajas. También podrán descargar la hoja de cálculo con el calendario actual.

5.3.3.3 Modificar usuario

Esta funcionalidad es también idéntica al caso tratado anteriormente para los administradores.

6 CONCLUSIÓN Y FUTURAS LÍNEAS DE TRABAJO

Durante este trabajo se ha desarrollado un sistema de información útil para la gestión del personal de un hospital, atendiendo al cumplimiento de unos requisitos obtenidos de una experiencia real. Mediante la base de datos y la página web que la acompaña, la persona a cargo de la planta podrá ejecutar los métodos manuales que utilizaba para asignar a los trabajadores a sus turnos correspondientes de forma más cómoda y moderna.

Además, no solo ha cubierto las necesidades más acuciantes de ayuda a la generación del cuadrante, si no que permite a los empleados disponer de una plataforma a través de la cual pueden interactuar de una forma distinta con el equipo de gestión.

Dada la modularidad y escalabilidad del código desarrollado, el sistema podría ser adaptado en el futuro para atender a nuevas necesidades que puedan surgir desde la organización, que podrían ir desde añadir otro tipo de empleados que no se hayan modelado hasta adaptar el apartado visual y estilo, u otras mejoras que puedan deducirse tras el uso continuado del sistema.

Otra de las posibles líneas de trabajo sería incluir, a pesar de los impedimentos del caso real, el modelado y resolución de la asignación como un NRP. Esto permitiría no solo optimizar el trabajo del administrador, si no los recursos y atención de la demanda del propio hospital.

Referencias

[1] ALMODÓVAR, A., BLANCO, A. y RIVERO, M. VII Encuesta Nacional de Condiciones de trabajo 2011. *Vasa*, 2011, pp. 1-57.

[2] MARY, A. F. Implementing a program of cyclical scheduling of nursing personnel. *Hospitals*, 1966, vol. 40, no 14, p. 108.

[3] HOWELL, John P. Cyclical scheduling of nursing personnel. *Hospitals*, 1966, vol. 40, no 2, pp. 77-85.

[4] ABERNATHY, J. et al. A three-stage manpower planning and scheduling model—a service-sector example. *Operations Research*, 1973, vol. 21, no 3, pp. 693-711.

[5] ISKEN, M. W. An implicit tour scheduling model with applications in healthcare. *Annals of Operations Research*, 2004, vol. 128, no 1-4, pp. 91-109.

[6] DELLA CROCE, F. y SALASSA, F. A variable neighborhood search based matheuristic for nurse rostering problems. *Annals of Operations Research*, 2014, vol. 218, no 1, pp. 185-199.

[8] Jingpeng L. et al. The falling tide algorithm: a new multi-objective approach for complex workforce scheduling. *Omega*, 2012, vol. 40, no 3, pp. 283-293.

[9] AZAIEZ, M. N. y AL SHARIF, S. A 0-1 goal programming model for nurse scheduling. *Computers and Operations Research*, 2005, vol. 32, no 3, pp. 491-507.

[10] TOPALOGLU, S. y OZKARAHAN, I. An implicit goal programming model for the tour scheduling problem considering the employee work preferences. *Annals of Operations Research*, 2004, vol. 128, no 1-4, pp. 135-158.

[11] CHEANG, B. et al. Nurse rostering problems—a bibliographic survey. *European journal of operational research*, 2003, vol. 151, no 3, pp. 447-460.

[12] OKADA, M. y OKADA, M. Prolog-based system for nursing staff scheduling implemented on a personal computer. *Computers and Biomedical Research*, 1988, vol. 21, no 1, pp. 53-63.

[13] RANDHAWA, S. U. y SITOMPUL, D. A heuristic-based computerized nurse scheduling system. *Computers and operations research*, 1993, vol. 20, no 8, pp. 837-844.

[14] GENDREAU, M. y POTVIN, J. Metaheuristics in combinatorial optimization. *Annals of Operations Research*, 2005, vol. 140, no 1, pp. 189-213.

[15] MLADENOVIC, N. A variable neighborhood algorithm-a new metaheuristic for combinatorial optimization. En *Papers presented at Optimization Days*. 1995.

[16] BURKE, E.K., et al. The state of the art of nurse rostering. *Journal of scheduling*, 2004, vol. 7, no 6, pp. 441-499.

[17DOWSLAND, K. A. Nurse scheduling with tabu search and strategic oscillation. *European journal of operational research*, 1998, vol. 106, no 2-3, pp. 393-407.

[18] BESTER, M. J., NIEUWOUDT, I. y VAN VUUREN, J. Finding good nurse duty schedules: a case study. *Journal of Scheduling*, 2007, vol. 10, no 6, pp. 387-405.

[19] BELLANTI, F. et al. A greedy-based neighborhood search approach to a nurse rostering

problem. *European Journal of Operational Research*, 2004, vol. 153, no 1, pp. 28-40.

[20] AICKELIN, U. y WHITE, P. Building better nurse scheduling algorithms. *Annals of Operations Research*, 2004, vol. 128, no 1-4, pp 159-177.

[21] AICKELIN, U. y DOWSLAND, K. A. An indirect genetic algorithm for a nurse-scheduling problem. *Computers and operations research*, 2004, vol. 31, no 5, pp. 761-778.

[22] BAI, R. et al. A hybrid evolutionary approach to the nurse rostering problem. *IEEE Transactions on Evolutionary Computation*, 2010, vol. 14, no 4, pp. 580-590.

[23] KIRKPATRICK, S., GELATT, C.D. y VECCHI, M.P. Optimization by simulated annealing. *science*, 1983, vol. 220, no 4598, pp. 671-680.

[24]GEYER, C. J. y THOMPSON, E. A. Annealing Markov chain Monte Carlo with applications to ancestral inference. *Journal of the American Statistical Association*, 1995, vol. 90, no 431, pp. 909-*920.*

[25] *SERVICEMAX. Disponible* en: https://www.servicemax.com/es

[26] *SYNCHROTEAM*. Disponible en: https://www.synchrotDeam.com/es/

[27] HIBERUS, KVP. Disponible en: https://www.hiberus.com/kvp/plataforma

[28] ATURNOS. Disponible en: https://www.aturnos.com/

[30] *PHP: Historia De PHP Y Proyectos Relacionados - Manual*. [online] [Consulta: el 3 de Julio de 2020]. Disponible en: https://web.archive.org/web/20071030180009/http://es2.php.net/history.

[31] BERNERS-LEE, Tim. Re: SGML/HTML docs, X Browser Disponible en https://lists.w3.org/Archives/Public/www-talk/1991NovDec/0020.html. 9 de diciembre de 1991; 15:33:46 [Consulta: el 4 de Julio de 2020]

[32] Christensson, P. (marzo, 2007). SQL Definition.[Consulta: el 3 de Julio de 2020]. Disponible en: https://techterms.com.

[33]Christensson, P. (2016). Relational Database Definition. [Consulta: el 2 de Julio de 2020]. Disponible en: https://techterms.com.

[34] *DB-Engines Ranking*. DB-Engines. (2020). [Consulta: el 1 de Julio de 2020]. Disponible en: https://db-engines.com/en/ranking.

[35] *Enterprise architect*, ©2000-2020 Sparx Systems Pty Ltd. [Consulta: 5 de julio de 2020] Disponible en: https://sparxsystems.com/products/ea/index.html

[36] *Aptana Studio.* © 2005 – 2018 Axway, Inc. [Consulta: el 3 de Julio de 2020]. Disponible en: http://www.aptana.com/

[37] EGUILUZ, J. LibrosWeb. *Introducción a JAVASCRIPT.* Publicado: 25 de Marzo de 2009 [consulta: 15 febrero 2020]Disponible en: http://roa.ult.edu.cu/bitstream/123456789/440/1/introduccion_javascript.pdf

[38] *Javascript*. desarrolloweb.com [Consulta: el 3 de Julio de 2020]. Disponible en: https://desarrolloweb.com/home/javascript

[39] *About.* © 2003 – 2020 phpMyAdmin contributors [Consulta: el 3 de Julio de 2020]. Disponible en: https://www.phpmyadmin.net/

[40] *EasyPHP Devserver* 2000 – 2019 EasyPHP [Consulta: 4 de julio de 2020] Disponible en: https://www.easyphp.org/

[41]JUNTA DE ANDALUCÍA. *Guía para la redacción de casos de uso*, 2020. [online] [Consulta: 26 Mayo 2020]. Disponible en: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/416

[42] CHEN, P. P. The entity-relationship model—toward a unified view of data. *ACM transactions on database systems (TODS)*, 1976, vol. 1, no 1, pp. 9-36.

[43] CODD, E. F. A relational model of data for large shared data banks. En *Software pioneers*. Springer, Berlin, Heidelberg, 2002. pp. 263-294.