eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Facultad de Informática

Grado de Ingeniería Informática

▪ **T**rabajo **F**in de **G**rado ▪

Ingeniería del Software

Herramienta para gestión de vacaciones

Aitor Ortega Rodríguez

2022

Dirección

Alfredo Goñi Sarriguren

Resumen

La empresa Uvesco [1], enfocada en el sector del gran consumo y de la distribución alimentaria, es responsable de las siguientes líneas de supermercados BM [2] y Super Amara [3].
Cuenta con un total de 4 sedes logísticas y 278 establecimientos, todos ellos distribuidos a nivel nacional. Actualmente, en grupo Uvesco trabajan en torno a 6000 personas, se encuentra en un proceso de expansión y se prevé que la cantidad de empleados/as continúe aumentando.

En la actualidad, los trabajadores/as de oficina no cuentan con un procedimiento común para solicitar sus vacaciones, lo que significa que cada departamento realiza este proceso de forma diferente.
En consecuencia, el tiempo destinado a la hora de realizar la planificación anual de las vacaciones, por parte de los jefe/as de cada departamento, es mayor al deseado. Esto supone una dificultad a la hora de consolidar los datos por parte del departamento de recursos humanos, ya que las diferentes solicitudes realizadas por cada empleado/a no se encuentran centralizadas.

Es por ello, que la motivación principal de este proyecto es desarrollar un nuevo proceso y funcionalidad en una aplicación ya existente, llamada SIU [4], que sea capaz de informar, gestionar y administrar el proceso de selección de las vacaciones, concretamente para aquellos empleados/as de las oficinas, además de facilitar la consolidación de los datos.

[1] https://www.uvesco.es/
[2] https://www.bmsupermercados.es/
[3] https://www.superamara.com/
[4] Sistema Integral Uvesco

Agradecimientos

En primer lugar agradezco a mi tutor, Alfredo Goñi, por haber dirigido este proyecto y haber prestado su colaboración en todo momento.

También me gustaría dar las gracias a Uvesco por la oportunidad que me ha brindado, de crecer tanto a nivel profesional como personal, al realizar las prácticas y el trabajo fin de grado junto a ellos. Especialmente a todas las personas que conforman el equipo de sistemas de la empresa y aquellas personas que colaboran de forma externa.

Por último, me gustaría mencionar a mi familia y amigos que han supuesto en todo momento un apoyo muy importante, ayudándome a superar los obstáculos que han surgido durante este periodo académico.

Índice general

Índice de Figuras y Tablas

Figuras

Tablas

1

Introducción

A día de hoy los empleados/as de la oficina no cuentan con un procedimiento común a través del cual poder solicitar las vacaciones en los diferentes departamentos de la empresa. Es por ello que cada sección coordina este proceso de diferente manera.

Con motivo de la situación previa, resulta necesario establecer un método común y desarrollar una nueva funcionalidad que facilite la labor de los empleados/as a la hora de solicitar sus vacaciones, llevar a cabo las revisiones de las diferentes solicitudes por los responsables de cada departamento y una final consolidación de los datos en la sección de recurso humanos.

Con ello lo que se pretende conseguir es informar, gestionar y administrar esta tarea de forma sencilla y ágil, logrando un mejor funcionamiento y control por cada uno de los departamentos de Uvesco.

2

Objetivos y planificación del proyecto

Este capítulo contiene los objetivos del proyecto y la planificación completa. Se detallarán los diferentes paquetes de trabajo identificados y las fases seguidas para ejecutarlos además de las estimaciones para cada una de ellas.

2.1 Objetivo

El objetivo del proyecto consiste en **definir un proceso de selección de vacaciones y desarrollar la funcionalidad** que permita llevar a cabo la gestión y administración del proceso previamente definido con el fin de establecer un único sistema centralizado mediante el que realizar esta tarea.

2.2 Alcance

En todo proyecto es imprescindible definir los límites que abarca, indicando las tareas a realizar y los diferentes aspectos que quedan excluidos, de manera que el ciclo de vida del proyecto concluya satisfactoriamente.

El proyecto tiene como objetivo la definición de un proceso de selección de vacaciones y desarrollo una nueva funcionalidad que permita llevar a cabo este proceso.
Esta nueva herramienta permitirá informar, administrar y gestionar las solicitudes realizadas por todos los empleados/as.
Lo que se pretende lograr con ello es que todos los datos queden centralizados, por lo que será necesario almacenarlos en la base de datos del servidor AS/400, y así establecer un único proceso para todos los departamentos de la oficina.

Por ello el trabajo se centrará en la definición del proceso de selección de las vacaciones y un posterior desarrollo de una nueva funcionalidad web, disponible para los empleado/as de oficina Uvesco y que dispongan del acceso a la herramienta SIU.
El trabajo concluirá con la implantación de la nueva funcionalidad en la herramienta.

2.2.1 Objetivos

Tal y como se ha mencionado previamente, el objetivo principal es la definición de un proceso de selección de vacaciones y desarrollo de la herramienta que permita llevarlo a cabo. Asimismo, se identifican lo siguientes objetivos secundarios:

Objetivo 1 Reducir el tiempo que se dedica a la planificación de las vacaciones anuales por los jefe/as de cada departamento.
Objetivo 2 Disminuir el tiempo que recursos humanos destina a la consolidación de los datos, respecto a las solicitudes realizadas por cada trabajador/a.
Objetivo 3 Proveer o dotar de la información, respecto a las vacaciones, a los empleados/as de la oficina para un mejor funcionamiento del día a día.
Objetivo 4 Proporcionar la información necesaria, a los empleados/as de la oficina, para el uso adecuado de la nueva herramienta desarrollada.
Objetivo 5 Integrar la nueva funcionalidad en el Sistema de Información de Uvesco.

2.2.2 Exclusiones

En lo que a la implementación del entorno web se refiere, se excluyen aspectos como la seguridad, dado que la herramienta en la que será integrada la nueva funcionalidad posee un sistema de seguridad y administración de tareas.

2.3 Estructura de Descomposición del Trabajo

Con el objetivo de comprender mejor el posterior Diagrama Gantt, a continuación, se muestra la Estructura de Descomposición del Trabajo (E.D.T) del proyecto.

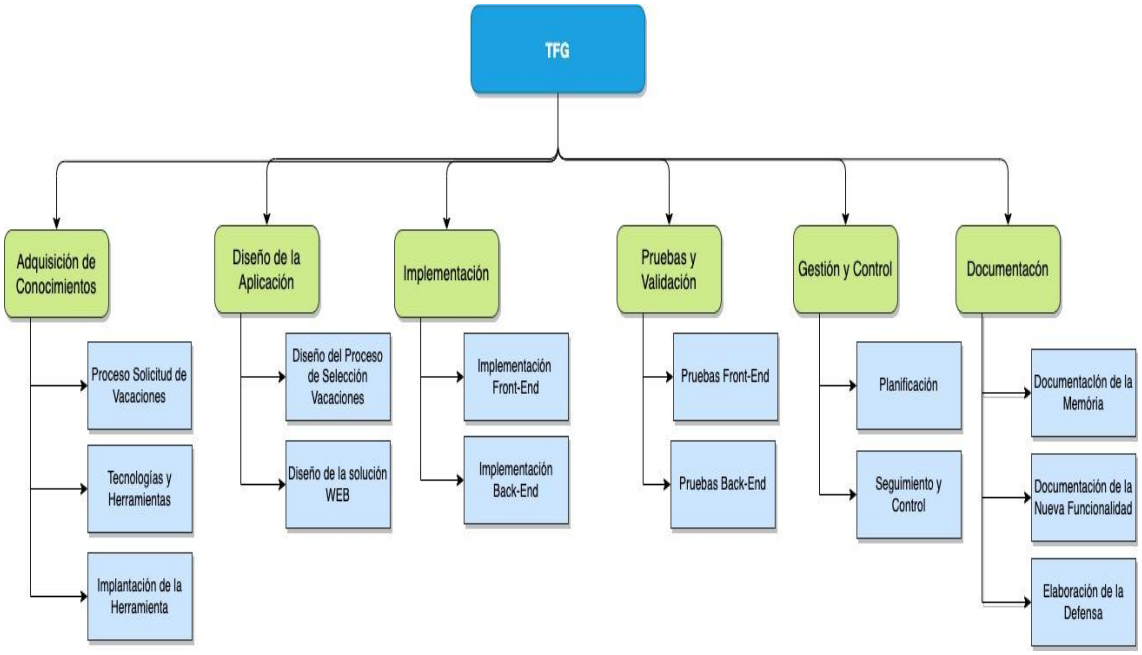


Fig.2.1 *Estructura de Descomposición del Trabajo (E.D.T)*

Adquisición de Conocimientos

La primera fase del proyecto, reúne aquellas tareas necesarias en cuanto a la adquisición de conocimientos y competencias necesarias para lograr los objetivos establecidos en el proyecto.

Diseño de la Aplicación

A continuación, en esta etapa se realizan las tareas necesarias para establecer un modelo sobre el que trabajar. Es importante realizar un diseño previo a cualquier implementación, de esta forma se establece una referencia hacia lo que se quiere conseguir respecto al producto final.

Implementación

Durante esta fase de trabajo se llevarán a cabo las tareas relacionadas con el desarrollo de la funcionalidad web en base al previo diseño realizado.

Pruebas y Validación

Una vez finalizada la implementación, en esta fase de trabajo se realizarán las pruebas necesarias sobre los diferentes paquetes de implementación, de forma que garantice el correcto funcionamiento de la herramienta ya que esta será utilizada en un entorno laboral real.

Gestión y Control

Esta etapa de trabajo une todas las tareas relativas a la gestión y planificación del proyecto. Las tareas desarrolladas, en esta fase, de forma correcta ayudarán a garantizar un proyecto que alcance los objetivos y requerimientos establecidos.

Documentación

En último lugar, esta fase de trabajo engloba las tareas asociadas a la elaboración de los documentos y los trámites relativos al trabajo fin de grado.

2.4 Diagrama Gantt

En esta sección se presenta el diagrama Gantt del proyecto, de forma mensual, el cual recoge las fechas de inicio y finalización previstas para cada una de las fases de trabajo previamente mencionadas, junto con las tareas a desarrollar e hitos del proyecto.

	Nombre de la tarea	Inicio	Finalizar
1	Trabajo fin de Grado	21/02/22	13/07/22
2	Hito 1: Inicio del proyecto	21/02/22	21/02/22
3	Hito 7: Matriculación del TFG	18/06/22	18/06/22
4	Hito 8: Subida del proyecto a la plataforma	26/06/22	26/06/22
5	Hito 9: Defensa del proyecto	12/07/22	12/07/22
6	Hito 10: Fin del proyecto y publicación de las notas	13/07/22	13/07/22
7	Adquisición de Conocimientos	22/02/22	09/03/22
8	Proceso Solicitud de Vacaciones (PSV)	22/02/22	09/03/22
9	Análisis del proceso de solicitud	22/02/22	09/03/22
10	Análisis del la jerarquía de la empresa, organigrama	22/02/22	09/03/22
11	Tecnologías y Herramientas (TH)	03/03/22	08/03/22
12	Análisis del entorno de desarrollo en el que se ejecutará la app web	03/03/22	08/03/22
13	Análisis de las tecnologías para desarrollos existentes de Interfaz	03/03/22	08/03/22
14	Análisis de las tecnologías utilizadas en desarrollos existentes.	03/03/22	08/03/22
15	Análisis de implementación de tablas en el AS400	03/03/22	08/03/22
16	Implantación de la Herramienta (IH)	09/03/22	09/03/22
17	Análisis del proceso de implantación de la herramienta	09/03/22	09/03/22
18	Análisis de la gestión del servidor	09/03/22	09/03/22
19	Hito 2: Fin adquisición de conocimientos	09/03/22	09/03/22
20	Diseño de la Aplicación	10/03/22	31/03/22
21	Diseño del proceso de selección vacaciones (DPSV)	10/03/22	14/03/22
22	Diseño de un proceso común	10/03/22	10/03/22
23	Elaboración del proceso	10/03/22	14/03/22
24	Diseño de la solución WEB (DWEB)	15/03/22	31/03/22
25	Hito 3: Análisis de requisitos web: Reunión	15/03/22	15/03/22
26	Diseño de las tablas en el servidor AS400	15/03/22	16/03/22
27	Diseño de la Arquitectura	17/03/22	17/03/22
28	Diseño del endPoint en el Web Service	18/03/22	22/03/22
29	Diseño de la Interfaz	22/03/22	31/03/22
30	Hito 4: Fin diseño de la aplicación	31/03/22	31/03/22
31	Implementación	01/04/22	07/06/22
32	Implementación del Front-End (IFE)	01/04/22	27/04/22
33	Desarrollo de las interfaces	01/04/22	07/04/22
34	Desarrollo de la funcionalidad de las interfaces	07/04/22	27/04/22
35	Implementación Back-End (IBE)	28/04/22	07/06/22
36	Desarrollo de las tablas para implementación en AS400	28/04/22	29/04/22
37	Desarrollo del WS	02/05/22	03/05/22
38	Desarrollo de la funcionalidades en base al análisis de requisitos	04/05/22	07/06/22
39	Hito 5: Fin implementación	07/06/22	07/06/22

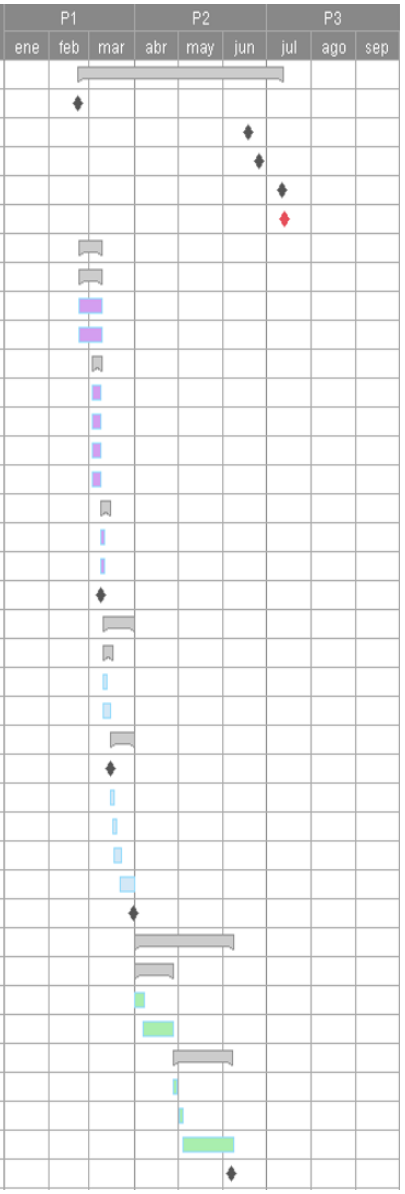
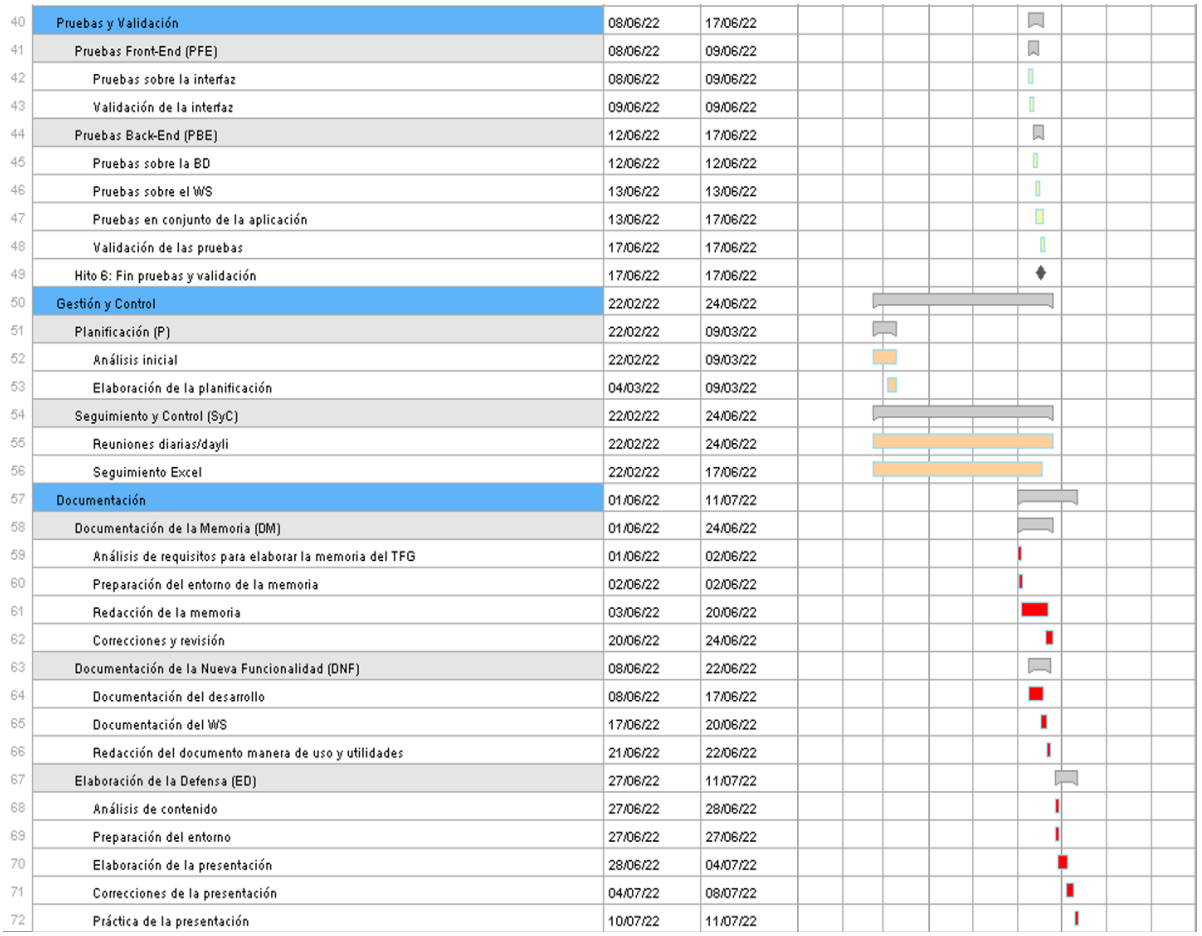


Fig.2.2 *Diagrama Gantt mensual - Primera parte*



40	Pruebas y Validación	08/06/22	17/06/22
41	Pruebas Front-End (PFE)	08/06/22	09/06/22
42	Pruebas sobre la interfaz	08/06/22	09/06/22
43	Validación de la interfaz	09/06/22	09/06/22
44	Pruebas Back-End (PBE)	12/06/22	17/06/22
45	Pruebas sobre la BD	12/06/22	12/06/22
46	Pruebas sobre el WS	13/06/22	13/06/22
47	Pruebas en conjunto de la aplicación	13/06/22	17/06/22
48	Validación de las pruebas	17/06/22	17/06/22
49	Hito 6: Fin pruebas y validación	17/06/22	17/06/22
50	Gestión y Control	22/02/22	24/06/22
51	Planificación (P)	22/02/22	09/03/22
52	Análisis inicial	22/02/22	09/03/22
53	Elaboración de la planificación	04/03/22	09/03/22
54	Seguimiento y Control (SyC)	22/02/22	24/06/22
55	Reuniones diarias/dayli	22/02/22	24/06/22
56	Seguimiento Excel	22/02/22	17/06/22
57	Documentación	01/06/22	11/07/22
58	Documentación de la Memoria (DM)	01/06/22	24/06/22
59	Análisis de requisitos para elaborar la memoria del TFG	01/06/22	02/06/22
60	Preparación del entorno de la memoria	02/06/22	02/06/22
61	Redacción de la memoria	03/06/22	20/06/22
62	Correcciones y revisión	20/06/22	24/06/22
63	Documentación de la Nueva Funcionalidad (DNF)	08/06/22	22/06/22
64	Documentación del desarrollo	08/06/22	17/06/22
65	Documentación del WS	17/06/22	20/06/22
66	Redacción del documento manera de uso y utilidades	21/06/22	22/06/22
67	Elaboración de la Defensa (ED)	27/06/22	11/07/22
68	Análisis de contenido	27/06/22	28/06/22
69	Preparación del entorno	27/06/22	27/06/22
70	Elaboración de la presentación	28/06/22	04/07/22
71	Correcciones de la presentación	04/07/22	08/07/22
72	Práctica de la presentación	10/07/22	11/07/22

Fig.2.3 *Diagrama Gantt mensual - Segunda parte*

2.4.1 Hitos del proyecto

- **Hito 1**: Inicio del proyecto (21/02/2022)
- **Hito 2**: Fin adquisición de conocimientos (09/03/2022)
- **Hito 3**: Análisis de requisitos web: Reunión (fecha estimada: 15/03/2022)
- **Hito 4**: Fin diseño de la aplicación (31/03/2022)
- **Hito 5**: Fin Implementación (07/06/2022)
- **Hito 6** :Fin pruebas y validación (07/06/2022)
- **Hito 7**: Matriculación del TFG (18/06/2022)
- **Hito 8**: Subida del proyecto a la plataforma (26/06/2022)
- **Hito 9**: Defensa del proyecto (fecha estimada: 12/07/2022)
- **Hito 10**: Fin del proyecto y publicación de las notas (13/07/2022)

2.4.2 Fases y fechas

	Nombre de la tarea	Inicio	Finalizar	P1			P2			P3		
				ene	feb	mar	abr	may	jun	jul	ago	sep
1	Trabajo fin de Grado	21/02/22	13/07/22									
7	Adquisición de Conocimientos	22/02/22	09/03/22									
20	Diseño de la Aplicación	10/03/22	31/03/22									
31	Implementación	01/04/22	07/06/22									
40	Pruebas y Validación	08/06/22	17/06/22									
50	Gestión y Control	22/02/22	24/06/22									
57	Documentación	01/06/22	11/07/22									

Fig.2.4 *Fases y fechas del proyecto*

En la anterior imagen podemos observar que tanto la fase de gestión y control como la de documentación del proyecto deberán realizarse de forma paralela a alguna o varias de las etapas.
Sin embargo, se puede observar que las cuatro primeras etapas se realizarán de forma sucesiva, es decir, hasta que la anterior no haya finalizado no se pasará a la siguiente.

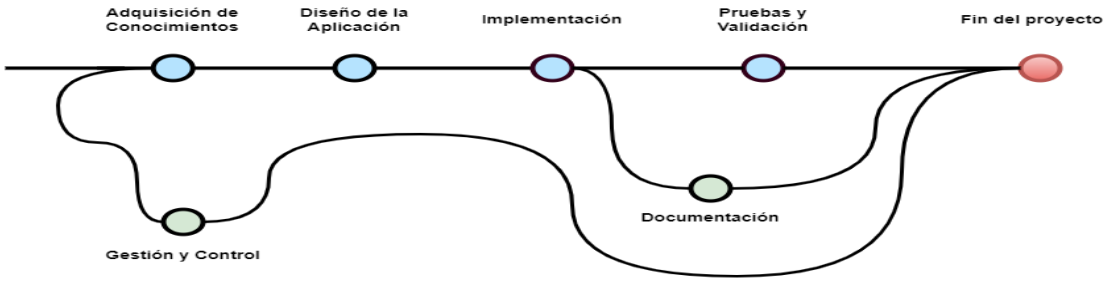


Fig.2.5 *Dependencia entre fases*

2.4.3 Estimación de tiempos

En este apartado se muestran los tiempos estimados para cada una de las fases y paquetes del proyecto, mostrando una estimación general finalmente.

Adquisición de Conocimientos	
PSV	17 h
TH	2 h
IH	1 h
Total	**20**

Diseño de la aplicación	
DPSV	10 h
DWEB	40 h
Total	**50**

Implementación	
IFE	80 h
IBE	105 h
Total	**185**

Pruebas y Validación	
PFE	10 h
PBE	15 h
Total	**25**

Gestión y Control	
P	15 h
SyC	25 h
Total	**40**

Documentación	
DM	25 h
DNF	5 h
ED	10 h
Total	**40**

Horas Totales del Proyecto 360

Tabla 2.1 *Estimación de tiempos por fases y paquetes*

2.5 Análisis de riesgos

Resulta necesario realizar un análisis y gestión de riesgos en todo proyecto con el objetivo de que este se desarrolle de forma natural a pesar de los diferentes factores que puedan perjudicar a su ciclo de desarrollo. En caso de que alguno de los riesgos impacte durante su proceso la finalidad es que genere la menor cantidad de costes, por lo que es necesario elaborar un plan para su mitigación.

En este apartado se identifican los posibles riesgos que pueden generar un impacto negativo en el proyecto.

R1. ***Coordinación y compaginación con las asignaturas***

El proyecto será desarrollado a lo largo del segundo cuatrimestre por lo que este se deberá coordinar y compaginar con asignaturas de este último cuarto curso.

El impacto que este puede generar es alto, dado que los plazos de realización y entregas podrían verse afectados de forma negativa, no cumpliendo con las fechas establecidas.

Es por ello que, como plan de mitigación, se plantea realizar modificaciones en la planificación inicial de forma que se pueda coordinar y compaginar las asignaturas de la mejor forma posible con el proyecto.

R2. ***Errores en la planificación***

Al tratarse de un proyecto de gran amplitud cabe la posibilidad de elaborar una planificación con errores o que no se tengan en cuenta ciertos aspectos que posteriormente surjan causando desviaciones en la realización de las tareas.

El impacto que puede generar es alto, provocando retrasos o afectando al cumplimiento de los plazos establecidos inicialmente.

Por ello, se plantea realizar modificaciones en los plazos de elaboración de las diferentes tareas de manera que el impacto que este riesgo pueda generar sea el menor posible.

R3. ***Análisis y desarrollo de la captura de requisitos***

Al tratarse de un desarrollo realizado en una empresa, cabe la posibilidad de que el cliente decida modificar o incluir nuevas funcionalidades durante el proceso de desarrollo.

Esto puede generar un impacto alto afectando a la implementación y diseño, dado que cabría con mayor probabilidad, la posibilidad de no cumplir con los objetivos establecidos en lo que a las tareas se refiere.

Como plan de mitigación se propone realizar una replanificación, dando prioridad a las tareas previamente existentes, es decir, una vez que finalicen las tareas iniciales darán comienzo las nuevas modificaciones o funcionalidades propuestas durante su desarrollo, siempre y cuando las nuevas propuestas carezcan de una prioridad mayor. Por el contrario, se realizarán las nuevas tareas propuestas replanificando las ya existentes.

R4. ***Errores o fallos técnicos***

Dado que se utilizará la base de datos del servidor AS/400 cualquier fallo en el servidor podría generar retrasos en el desarrollo de las tareas de implementación y pruebas.

El impacto que este puede generar es medio, debido a que existen dos servidores más a los que poder conectarse. Sin embargo, uno de ellos contiene datos de pruebas por lo que la aplicación se podría ejecutar solamente con aquellos datos idénticos a los del servidor principal.

Debido a ello, se plantea el siguiente plan de mitigación: conectarse al segundo servidor ya que este realiza un volcado de datos diario, manteniendo copias del servidor principal. No obstante, en el peor escenario, habría que realizar un volcado de aquellos datos necesarios a una base de datos local y poder continuar con el plan inicial.

R5. ***Problemas personales***

Existe la posibilidad de que durante el desarrollo del proyecto surjan problemas personales, dado que el periodo de tiempo de desarrollo es lo suficientemente amplio para que situaciones como tal puedan darse.

El impacto que pueda generar es medio, causando retrasos en la elaboración del trabajo y entrega de las tareas.

Se plantea, como plan de mitigación, una replanificación de las tareas y desarrollo en periodos vacacionales o no laborales, solventando así los posibles retrasos que pueda generar.

3

Análisis de Requisitos

En este capítulo se especifican los diferentes roles y perfiles de los usuarios que existen a la hora de hacer uso de la nueva funcionalidad, además de los requisitos con los que tendrá que cumplir la nueva herramienta para abordar todas las necesidades especificadas por la empresa de forma electrónica y presencial a través de reuniones.

Para una elaboración y desarrollo satisfactorio del proyecto se ha llevado a cabo un proceso de análisis mediante el cual se han obtenido los requisitos que más adelante se describen.
A través de esta fase lo que se pretende es asentar la base del resto del proyecto, de forma que se identifiquen aquellas necesidades que el producto final desarrollado deberá satisfacer, dado que se utilizará en un entorno laboral real.

El proceso de análisis de requisitos se ha realizado de la siguiente manera:

Inicialmente, se ha realizado un análisis de la situación actual de la empresa. Se han identificado los diferentes roles y perfiles además de estudiar el proceso llevado a cabo en cuanto a la selección de las vacaciones, para esclarecer la necesidad de creación de esta nueva herramienta.

Posteriormente, todas las dudas se han consultado a través del correo electrónico. Estas consultas han sido dirigidas al departamento de organización dado que es el encargado de la gestión y administración de nuevos proyectos.

Finalmente, se concretó una reunión presencial junto a los responsables de los departamentos de organización y de recursos humanos para resolver cualquier duda pendiente, concluyendo así con la captura y análisis de requisitos.

3.1 Roles de la aplicación

En este apartado se identificarán y describirán los diferentes roles y perfiles que harán uso de la herramienta desarrollada.
Principalmente se identifican dos tipos de roles: Usuario y Administrador.
Sin embargo, dentro del rol de usuario identificamos diferentes perfiles: solicitante, aprobador y visualizador.

3.1.1 Usuario

Este rol se desempeñará por cualquier empleado/a de la oficina Uvesco con acceso a la herramienta SIU, sin embargo, dependiendo del puesto y nivel del empleado/a se especializará en uno de los siguientes perfiles.

3.1.1.1 Solicitante

Exceptuando aquellos empleados/as con un nivel 2 o superior o perteneciente al puesto de dirección de recursos humanos, serán solicitantes de vacaciones.
Los trabajadores/as correspondientes a este nivel y puesto, quedan excluidos dado que no cuentan con aprobadores, por lo que sus solicitudes no serán procesadas.El resto de empleados/as, dispondrá de la posibilidad de realizar y gestionar las diferentes solicitudes.

3.1.1.2 Aprobador

Algunos empleados/as de la oficina serán usuarios, además de solicitantes, aprobadores. Se encargará de procesar las solicitudes de aquellos empleados/as de los que sea responsable.
La regla, que indica si un empleado/a es aprobador, se ha determinado según el nivel y departamento asignado en el organigrama [1].

	Nivel con Aprobador (o inmediatemente superior)	Nivel SIN Aprobador (o inmesiatamente superior)
Regla general	*5*	*2 y Director/a RRHH*
Excepción de la Regla según Departamento Administración y control de Gestión Recursos Humanos	*6*	
Excepción de la Regla según nivel y puesto Niveles 3 y 4 Nivel 5 Gerente Organización Nivel 10 Secretario/a Dirección General	*4 Director/a RRHH*	

Tabla 3.1 *Reglas y excepciones para la asignación de un perfil aprobador*

3.1.1.3 Visualizador

Todos los usuarios serán visualizadores, tienen la posibilidad de ver las solicitudes aprobadas de los empleados/as asociados a su mismo departamento.

Además, existe la posibilidad de ver todos los departamentos pero únicamente estará permitido para aquellos usuarios asociados a los siguientes puestos:

- Director/a general
- Subdirector/a general
- Director/a RRHH
- Jefe/a RRHH centrales
- Responsable de controlling RRHH

Por último, aquellos visualizadores que conformen el comité de dirección podrán ver, de la misma forma, las solicitudes realizadas por los integrantes de este.

3.1.2 Administrador

El rol administrador, será desempeñado por los administradores de la herramienta SIU y aquellos empleados/as con acceso a esta nueva tarea. El administrador se encargará de alimentar la nueva funcionalidad desarrollada mediante la creación y gestión de los calendarios, para un correcto funcionamiento y uso.

[1] "*Sinopsis o esquema de la organización de una entidad, de una empresa o de una tarea.*" [RAE y ASALE, 2021]

3.2 Descripción de los requisitos

Tras el análisis inicial, resolución de las dudas a través del correo y en una última instancia del proceso una reunión, se finalizó extrayendo los siguientes requisitos.

3.2.1 Requisitos funcionales de aplicación

- **RFP.1** Mostrará calendarios personalizados para cada empleado/a, indicando para cada uno de ellos los festivos y vacaciones que le correspondan.
- **RFP.2** Crear solicitudes de vacaciones por intervalos de fechas.
- **RFP.3** No permitirá realizar solicitudes con solapes entre fechas.
- **RFP.4** Proporcionará información detallada acerca del estado en el que se encuentra cada solicitud.
- **RFP.5** Permitirá la gestión de las solicitudes, además de solicitarlas se podrán modificar, eliminar y enviar dependiendo del estado en el que se encuentre la solicitud.
- Siempre y cuando el usuario sea aprobador:
    - **RFP.6** La aplicación posibilitará el procesado de las solicitudes a través de la modificación del estado, se podrán aprobar o rechazar.
    - **RFP.7** Posibilitará un aprobado en conjunto de todas las solicitudes realizadas por un empleado/a.
    - **RFP.8** Previamente a la confirmación del rechazo de una solicitud se añadirá el motivo por el cual será rechazada.
    - **RFP.9** Proporcionará filtros de búsqueda de solicitudes aprobadas y/o pendientes de revisar.
- **RFP.10** Permitirá realizar la descarga de datos, respecto a las solicitudes realizadas con estado aprobado, del departamento al que el usuario pertenece.
- **RFP.11** La aplicación, exclusivamente al responsable del departamento de recursos humanos, facilitará el proceso de carga de las diferentes solicitudes, con estado aprobado en el programa de nóminas de Castilla [2] a través de la descarga de los datos de todos los departamentos.
- **RFP.12** Únicamente el responsable del departamento de recursos humanos podrá indicar el envío en Castilla o lo que es lo mismo indicar la alimentación de los datos en el programa de nóminas.
- **RFP.13** La aplicación realizará envíos de notificaciones, una vez que los datos hayan sido consolidados en Castilla, es decir, cuando se haya indicado el envío por el responsable de recursos humanos. De esta forma los revisores, trabajadores/as y departamento de recursos humanos quedarán informados acerca de las modificaciones realizadas fuera del plazo.
- **RFP.14** Dependiendo del tipo de usuario y perfil mostrará mayor o menor funcionalidad.

[2] Castilla se denomina al programa que utiliza el departamento de recursos humanos para la gestión y administración de: nóminas, altas, bajas etc.

- Siempre y cuando el usuario cuente con la funcionalidad de administrador
    - **RFP.15** La aplicación permitirá crear calendarios .
    - **RFP.16** Dispondrán la posibilidad de modificar calendarios.
    - **RFP.17** Se podrán eliminar calendarios.
    - **RFP.18** Proporcionará la posibilidad de visualizar calendarios pasados.
    - **RFP.19** Se podrán realizar consultas generales en la que se muestren todos los calendarios y asociaciones correspondientes a cada empleado/a.
    - **RFP.20** Se podrán añadir festivos al calendario.
    - **RFP.21** Eliminar festivos de un calendario.
    - **RFP.22** Modificar los días festivos de un empleado/a incrementando o decrementando el total de días a disfrutar.
    - **RFP.23** Asociar cuentas de cotización.
    - **RFP.24** Desasociar cuentas de cotización.

3.2.2 Requisitos funcionales de interfaz gráfica

- **RFI.1** Tanto los días festivos y no laborales como los rangos de solicitudes que se encuentren en los siguientes estados: aprobados, pendientes de enviar y pendientes de aprobar, se mostrarán señalados en el calendario de cada empleado/a con colores diferentes.
- **RFI.2** Se mostrará una leyenda indicando la situación general respecto a las posibles solicitudes a realizar y ya realizadas a través de contadores indicando el total.
- **RFI.3** Al crear o modificar una solicitud mostrará una leyenda en la que indicará: el motivo por el que no se permite llevar a cabo la operación o el número total de días que se solicitarán en caso de cumplir con las condiciones para realizar la operación.
- **RFI.4** Siempre y cuando el empleado/a sea aprobador mostrará información acerca del número de días, respecto a las solicitudes pendientes de aprobar y/o aprobadas , de forma mensual por cada empleado/a del que es responsable.
- **RFI.5** Siendo usuario aprobador, permitirá visualizar a detalle las vacaciones seleccionadas, pendientes de aprobar y/o aprobadas, por los empleados/as de los que es responsable. Es decir, podrá visualizar de forma horizontal/mensual los días de vacaciones de cada empleado/a asociado al mismo departamento que el aprobador.
- **RFI.6** Se dará la posibilidad, a aquellos usuarios aprobadores, de redireccionamiento al panel de procesado de solicitudes desde el panel general de visualizado horizontal/mensual del departamento.
- **RFI.7** Mostrará la cantidad de solicitudes a procesar, en caso de que existan, para un aprobador.
- **RFI.8** Permitirá realizar búsquedas de texto en ciertas funcionalidades de la herramienta.
    - Visualizar Departamento
    - Revisar Vacaciones
    - Visualizar Vacaciones
- **RFI.9** Existirá la posibilidad de agrupación y desagrupación de los datos en las siguientes funcionalidades
    - Revisar Vacaciones
    - Visualizar Vacaciones

3.2.3 Requisitos no funcionales

- **RNF.1** La nueva herramienta deberá ser integrada en la aplicación SIU.
- **RNF.2** Los datos de la aplicación deberán almacenarse en la base de datos del servidor AS/400.
- **RNF.3** Para acceder a la funcionalidad de solicitud se utilizará un WEB Service mediante el que se obtendrán los datos correspondientes del empleado/a como nivel, departamento etc. En base a la información obtenida se le asignará el calendario correspondiente y rol/perfil además de posibles excepciones que puedan existir asociadas al empleado/a como el número de días disponibles para solicitar.
- **RNF.4** La herramienta principal dispone un sistema para asociar permisos sobre funcionalidades y acciones a realizar dentro de la aplicación.
- **RNF.5** El formato de descarga de datos, respecto a las solicitudes realizadas, está basado en XML y habilitado para macros. El formato es el siguiente .xls
- **RNF.6** La nueva funcionalidad se encontrará siempre accesible a pesar del proceso de solicitud establecido.
- **RNF.7** El diseño e implementación de las interfaces deberá seguir la misma línea que las demás funcionalidades ya existentes en la herramienta SIU.

En la siguiente imagen se muestran los diferentes casos de uso, en base a la captura previa de requisitos, asociados a cada uno de los roles y perfiles existentes.

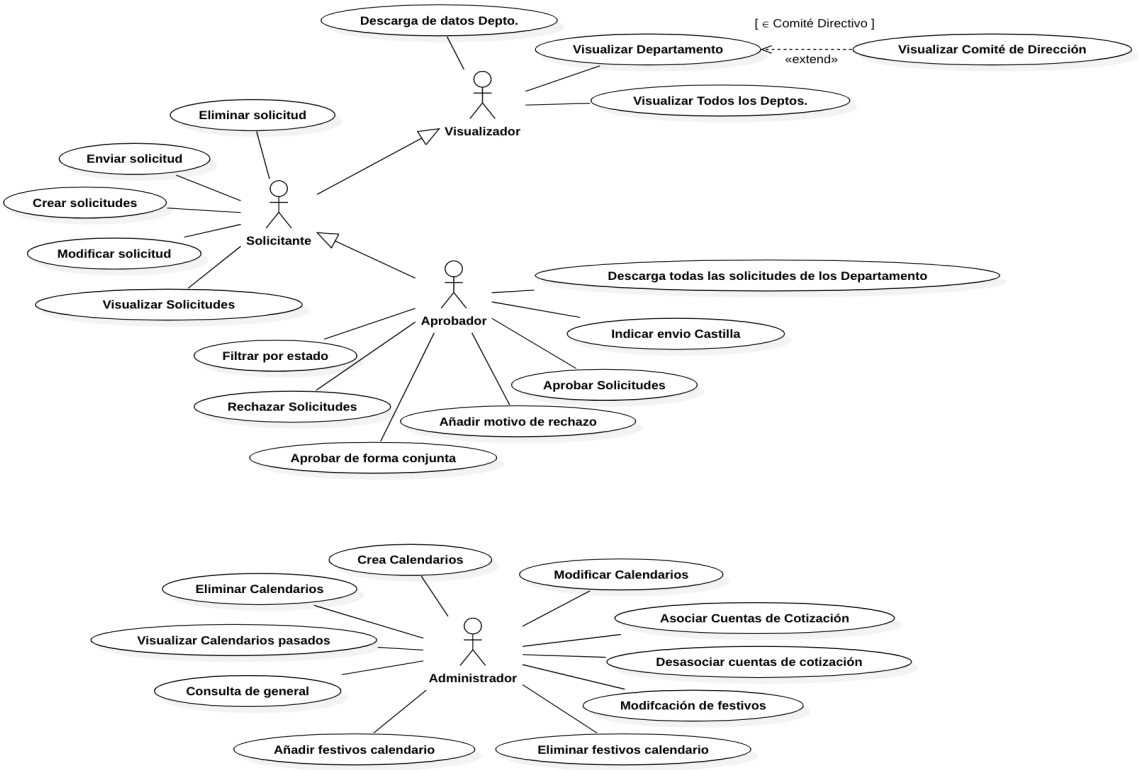


Fig.3.1 *Diagrama de casos de uso*

En la imagen podemos observar que algunos de los perfiles son generalización de otros, obteniendo cada vez mayor funcionalidad dependiendo del rol y perfil que se asigne a la hora utilizar la aplicación.

4

Herramientas y tecnologías

En este capítulo se describen las principales herramientas y tecnologías utilizadas para la gestión y desarrollo del proyecto.

Debido a la arquitectura y desarrollo de los proyectos ya existentes en la empresa se han utilizado las siguientes herramientas y tecnologías, de forma que la nueva funcionalidad desarrollada sea fácil de integrar y mantener en la aplicación.

4.1 Lenguajes

4.1.1 Java [1]

Java es un lenguaje de programación orientado a objetos, por lo que permite crear programas modulares y reutilizables, con el fin de lograr una mejor legibilidad y manejabilidad. Una de las ventajas más importantes es su capacidad multiplataforma, es decir, la facilidad al pasar de un sistema informático a otro o ejecutar un mismo programa en sistemas diferentes.

4.1.2 HTML

HTML es un lenguaje de marcación que sirve para definir y estructurar el contenido de las páginas web. Basado en etiquetas, también conocidas como marcas o tags, con las que podemos representar partes de un documento como; cabecera, cuerpo, encabezados, párrafos, etc.
Es admitido por todos los navegadores, proporcionando una alta escalabilidad.

4.1.3 CSS

CSS es un lenguaje de estilos o diseño gráfico y es utilizado para el desarrollo de la presentación de las páginas web. Se utiliza para definir la forma y apariencia externa de la UI [2].
Una de las ventajas principales es que una vez definido el estilo se aplica de manera consistente, es decir, una instrucción es capaz de controlar varias áreas por lo que mejora la velocidad de la página.

4.1.4 SQL

SQL es un lenguaje de consulta estructurado, enfocado en bases de datos relacionales. Basado en álgebra y cálculo relacional lo que permite obtener una gran cantidad de datos y procesar diferentes operaciones en muy poco tiempo.

4.1.5 JQuery

JQuery es un lenguaje de programación para el desarrollo web, concretamente es una librería de JavaScript. Su propósito es hacer más sencillo el uso de JS [3], simplificar las llamadas AJAX [4], la manipulación del DOM [5] etc. Es conciso, ya que existen secuencias de comandos al contrario que JS.

---

[1] https://www.java.com/es/
[2] User Interface
[3] JavaScript
[4] Asynchronous JavaScript and XML
[5] Document Object Model

4.2 Framework

4.2.1 Struts [6]

Struts es un framework para construir aplicaciones web Java basadas en la arquitectura Modelo Vista Controlador (MVC). Con esta herramienta el desarrollador web solo tiene que implementar la lógica de negocio. Además, proporciona una biblioteca de etiquetas la cual ayuda a crear JSP[7] fácilmente y las clases action pueden ser usadas para realizar operaciones eficientes.La herramienta facilita la resolución de problemas cuando hay un error [EDUCBA, 2022].

4.2.2 Bootstrap [8]

Bootstrap es un framework que permite crear una interfaz web utilizando CSS y JS, adaptando específicamente la apariencia del sitio web al tamaño del dispositivo en el que se muestra, es decir, el sitio web se ajusta automáticamente al tamaño del dispositivo.

Es por ello que su principal objetivo es llevar a cabo el uso de esta técnica de diseño y desarrollo conocida como diseño adaptativo o “responsive design” [Solis, 2014].

4.3 Entornos de desarrollo

4.3.1 Eclipse [9]

Eclipse es un entorno de desarrollo integrado (IDE [10]), de código abierto y multiplataforma que soporta múltiples lenguajes de programación. Existe una gran variedad de complementos o plugins que se pueden añadir incrementando su funcionalidad y facilitando los desarrollos.

Es un entorno que permite trabajar con proyectos de gran magnitud.

4.4 Otras tecnologías

4.4.1 Bitbucket [11]

Bitbucket es un servicio de alojamiento web para proyectos que utilizan los sistemas de control de versiones Mercurial y Git. Permite revertir cambios y mantener un control exhaustivo del desarrollo realizado.

Se puede integrar fácilmente con Jira, software enfocado en la gestión de proyectos, seguimiento de errores e incidencias.

---

[6] https://struts.apache.org/

[7] Jakarta Server Pages

[8] https://getbootstrap.com/

[9] https://www.eclipse.org/

[10] Integrated Development Environment

[11] https://bitbucket.org/

4.4.2 SmartGit [12]

SmartGit es un cliente gráfico de Git, capaz de integrarse con GitHub, BitBucket, GitLab, JIRA y otros servicios. Una de sus ventajas es que oculta la complejidad de Git, por lo que ayuda a evitar errores comunes de comandos proporcionando aún así la información y funcionalidad necesaria para aquellos usuarios más avanzados en Git.

4.4.3 Google Drive [13]

Google Drive es un servicio de alojamiento de archivos, uno de los más conocidos respecto al almacenamiento en la nube. Permite acceder a los datos desde cualquier dispositivo y en cualquier momento.

4.4.4 PostMan [14]

PostMan es una aplicación enfocada al testeo de APIs. Es un cliente HTTP y a través de este podemos testear solicitudes HTTP mediante una interfaz gráfica de usuario, en la que obtendremos los diferentes tipos de respuesta dependiendo de la solicitud realizada.

4.4.5 DBeaver [15]

DBeaver es una aplicación multiplataforma utilizada para trabajar con bases de datos. Soporta cualquier tipo de base de datos como MySQL, PostgreSQL, Oracle etc. Posee una gran cantidad de funcionalidades, mayormente enfocadas a la administración de base de datos, lo que implica una mayor complejidad de uso.

4.4.6 JSON

JSON es un formato ligero utilizado para estructurar datos en forma de texto con el fin de almacenarlos y transportarlos de forma sencilla y rápida. Su uso más frecuente es en aplicaciones cliente-servidor. Una forma de uso es a través de AJAX, donde la aplicación recupera los datos almacenados en el servidor sin necesidad de recargar la página.

4.4.7 StarUML [16]

StarUML es una aplicación de código abierto. Se ha utilizado para el diseño y modelado de diagramas del proyecto.

4.4.8 Smartsheet [17]

Smartsheet es un software que presta un servicio de colaboración y gestión del trabajo. Principalmente esta herramienta se ha utilizado para llevar a cabo la gestión del proyecto a lo largo de su desarrollo.

---

[12] https://www.syntevo.com/smartgit/
[13] https://drive.google.com/drive/
[14] https://www.postman.com/
[15] https://dbeaver.com/
[16] https://staruml.io/
[17] https://es.smartsheet.com/

5

Diseño de la aplicación

En el siguiente capítulo se muestra el diseño llevado a cabo en cuanto al proceso de solicitud propuesto, las tablas en las que se almacenarán los datos, las diferentes vistas que poseerá la aplicación y el diseño de la arquitectura del sistema.

5.1 Diseño del proceso

El procedimiento de selección de vacaciones propuesto para todos los empleados/as de Uvesco es el siguiente:

Inicialmente el administrador deberá realizar una gestión y administración de los calendarios antes de finalizar el año, de forma que todos los empleados/as queden asociados a un calendario concreto de cara al siguiente año.

Una vez realizada la tarea anterior, los trabajadores/as alimentarán la herramienta con la propuesta de días de vacaciones a disfrutar a lo largo del año, desde el 1 de enero hasta el día 15 de febrero.

Aquellos empleados/as que cumplan las condiciones de aprobador, dispondrán desde principios de año hasta el 28 de febrero para revisar la información de cada empleado/a de los que es responsable.
En caso de que alguno de los intervalos sea rechazado por el responsable, el empleado/a asociado a ese intervalo deberá reajustar la propuesta hasta completar el total de días de vacaciones que le corresponden.
Antes del día 15 de marzo, las vacaciones de todos los empleados/as deben quedar aprobadas por su responsable.

Finalmente, el responsable del departamento de recursos humanos alimentará el programa de nóminas con todas las propuestas realizadas por cada empleado/a de Uvesco antes del 31 de marzo.

En caso de que se realicen modificaciones, en las propuestas de las vacaciones utilizadas para la alimentación del programa de nóminas, posteriores a la fecha anterior se notificarán tanto al solicitante, como a su responsable y departamento de recursos humanos de forma que los datos sean corregidos lo antes posible en el programa de nóminas.

En la siguiente imagen se muestra de forma gráfica el proceso propuesto a llevar a cabo.

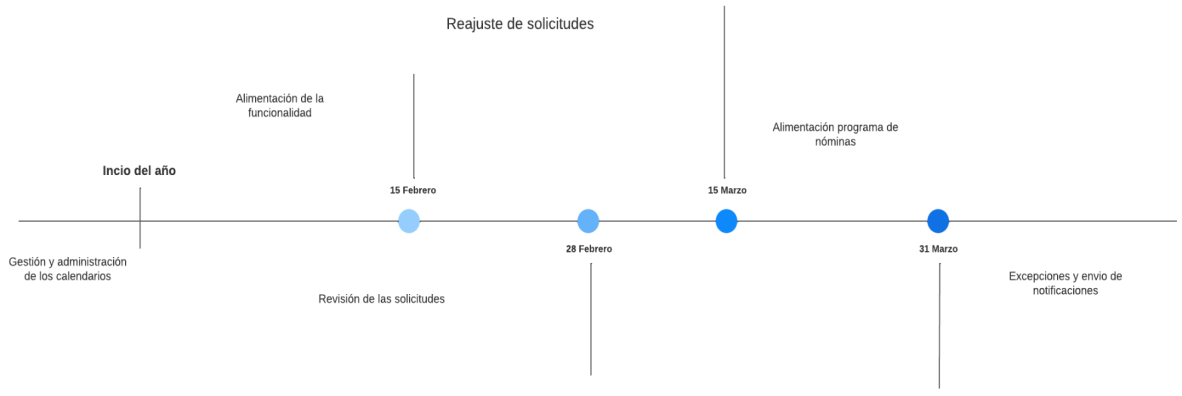


Fig.5.1 *Proceso temporal de la selección*

Sin embargo, cabe destacar que la aplicación se encontrará disponible durante todo el año, por lo que ninguna de las funcionalidades dejará de poder utilizarse en base a las fechas previamente indicadas.

5.2 Diseño de la base de datos

Las tablas se integrarán en la base de datos del servidor AS/400, se han diseñado haciendo uso de la nomenclatura del servidor, posteriormente en el capítulo de implementación se mostrará el fichero de instrucciones para su creación.
En las tablas se almacenarán los datos correspondientes a los calendarios y las solicitudes realizadas por los empleados/as.

5.2.1 Diagrama de tablas

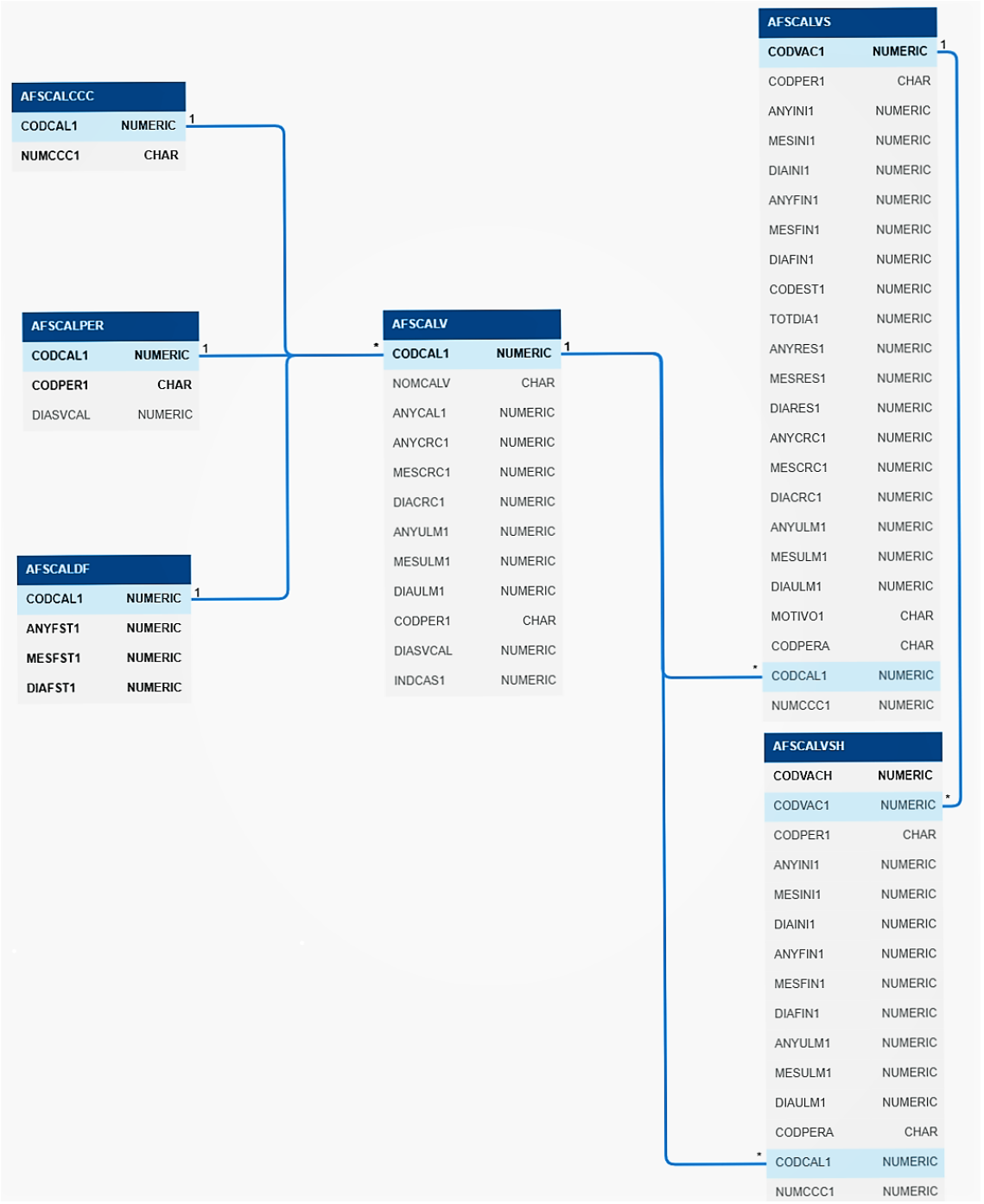


Fig.5.2 *Diagrama de tablas*

5.2.2 Descripción de tablas y atributos

Calendario Vacaciones (AFSCALV) : Se almacenarán los datos de los calendarios.

Nombre	Tipo	Descripción
CODCAL1	NUMERIC	Código del calendario
NOMCALV	CHAR	Nombre del calendario
ANYCAL1	NUMERIC	Año del calendario
ANYCRC1	NUMERIC	Año de creación
MESCRC1	NUMERIC	Mes de creación
DIACRC1	NUMERIC	Día de creación
ANYULM1	NUMERIC	Año última modificación
MESULM1	NUMERIC	Mes última modificación
DIAULM1	NUMERIC	Día última modificación
CODPER1	CHAR	Codigo del creador
DIASVCAL	NUMERIC	Número días festivos del calendario
INDCAS1	NUMERIC	Indicador de envío de datos

Tabla 5.1 *Tabla AFSCALV*

Calendario Dias Festivos (AFSCALDF) : La siguiente tabla almacenará los días festivos asociados a un calendario.

Nombre	Tipo	Descripción
CODCAL1	NUMERIC	Código del calendario
ANYFST1	NUMERIC	Año festivo del calendario
MESFST1	NUMERIC	Mes festivo del calendario
DIAFST1	NUMERIC	Día festivo del calendario

Tabla 5.2 *Tabla AFSCALDF*

Calendario Cuentas de Cotización (AFSCALCCC) : Almacenará las cuentas de cotización asociadas a los calendarios.

Nombre	Tipo	Descripción
CODCAL1	NUMERIC	Código del calendario
NUMCCC1	CHAR	Cuenta de Cotización

Tabla 5.3 *Tabla AFSCALCCC*

Calendario excepciones Personas (AFSCALPER) : Contendrá los datos de aquellas personas que por algún motivo no contienen los mismos días festivos que el calendario al que están asociadas.

Nombre	Tipo	Descripción
CODCAL1	NUMERIC	Código del calendario
CODPER1	CHAR	Código de la persona
DIASVCAL	NUMERIC	Número días totales festivos de la persona

Tabla 5.4 *Tabla AFSCALPER*

Calendario Vacaciones Solicitudes (AFSCALVS) : Esta tabla contendrá las solicitudes realizadas por los empleados/as de las oficinas de Uvesco.

Nombre	Tipo	Descripción
CODVAC1	NUMERIC	Código de la solicitud
CODPER1	CHAR	Código de la persona
ANYINI1	NUMERIC	Año inicio solicitud
MESINI1	NUMERIC	Mes inicio solicitud
DIAINI1	NUMERIC	Día inicio solicitud
ANYFIN1	NUMERIC	Año fin solicitud
MESFIN1	NUMERIC	Mes fin solicitud
DIAFIN1	NUMERIC	Día fin solicitud
CODEST1	NUMERIC	Estado de la solicitud
TOTDIA1	NUMERIC	Total de días de solicitados
ANYRES1	NUMERIC	Año resolución de la solicitud
MESRES1	NUMERIC	Mes resolución de la solicitud
DIARES1	NUMERIC	Día resolución de la solicitud
ANYCRC1	NUMERIC	Año creación de la solicitud
MESCRC1	NUMERIC	Mes creación de la solicitud
DIACRC1	NUMERIC	Día creación de la solicitud
ANYULM1	NUMERIC	Año última modificación solicitud
MESULM1	NUMERIC	Mes última modificación solicitud
DIAULM1	NUMERIC	Día última modificación solicitud
MOTIVO1	CHAR	Motivo rechazo de la solicitud
CODPERA	CHAR	Código del aprobador
CODCAL1	NUMERIC	Código del calendario
NUMCCC1	CHAR	Cuenta de Cotización

Tabla 5.5 *Tabla AFSCALVS*

El estado de las solicitudes obtendrá los siguientes valores:

0 : Aprobada
1 : Pendiente de aprobar
2 : Pendiente de enviar
3 : Rechazada
-1 : Eliminada

Calendario Vacaciones Históricos (AFSCALVSH) : La siguiente tabla recoge los datos de aquellas solicitudes que han sufrido algún cambió una vez se haya indicado el envío a Castilla. El objetivo es mantener un registro de estas solicitudes ya que será imprescindible conocerlo para realizar las modificaciones correspondientes en el programa de nóminas, no dando pie a malentendidos.

Nombre	Tipo	Descripción
CODVACH	NUMERIC	Código de la solicitud histórico
CODVAC1	NUMERIC	Código de la solicitud
CODPER1	CHAR	Código del aprobador
ANYINI1	NUMERIC	Año inicio de la solicitud
MESINI1	NUMERIC	Mes inicio de la solicitud
DIAINI1	NUMERIC	Día inicio de la solicitud
ANYFIN1	NUMERIC	Año fin de la solicitud
MESFIN1	NUMERIC	Mes fin de la solicitud
DIAFIN1	NUMERIC	Día fin de la solicitud
ANYULM1	NUMERIC	Año última modificación solicitud
MESULM1	NUMERIC	Mes última modificación solicitud
DIAULM1	NUMERIC	Día última modificación solicitud
CODPERA	CHAR	Código del aprobador
CODCAL1	NUMERIC	Código del calendario
NUMCCC1	CHAR	Cuenta de Cotización

Tabla 5.6 *Tabla AFSCALVSH*

5.3 Diseño de la interfaz

Se creará una sección nueva en el menú de la aplicación SIU, llamada Vacaciones. Esta nueva opción estará disponible para todos los empleados/as que dispongan del acceso a la aplicación. Sin embargo, dependiendo del rol que tome el trabajador/a dentro de la herramienta se le mostrará una de las siguientes funcionalidades o ambas.

Este proceso se realizará mediante el desarrollo estático de las páginas, es decir, únicamente se implementarán los componentes gráficos de la aplicación, careciendo de funcionalidad. De esta forma el diseño, en cuanto a la interfaz se refiere, se realizará por módulos.

5.3.1 Vistas y páginas

Se ha dividido en dos apartados, los cuales indican el acceso a las diferentes funcionalidades de la herramienta desarrollada "***Administrar Calendarios***" y "***Mis Vacaciones***".

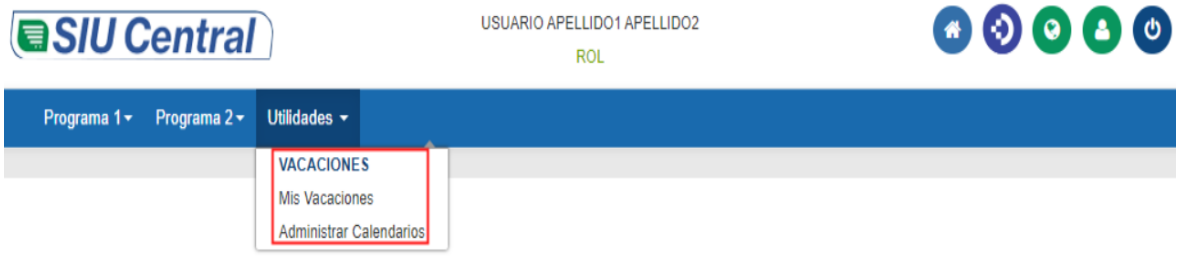


Fig.5.3 *Acceso a la funcionalidad*

En los siguientes apartados se describirán las vistas que se mostrarán al usuario y administrador para cumplir con los requisitos anteriormente expuestos en cada una de las opciones.

5.3.1.1 Administrar Calendarios

Esta funcionalidad solo se encuentra disponible para los administradores del sistema y responsables de la administración y gestión de los calendarios. Cuenta con un total de tres páginas diferentes. A continuación se describe cada una de ellas.

Página: Administración Calendarios

En esta pantalla inicial se muestra una vista general de los calendarios existentes.

- Se podrán visualizar calendarios presentes, pasados y futuros.
- Permitirá la creación y eliminación de calendarios de forma general, es decir, se podrá especificar el nombre y número de días festivos correspondientes al calendario, siempre y cuando el año seleccionado sea el presente o futuro.

Sin embargo, no se podrán manipular para realizar asociaciones, modificaciones etc. Para ello se deberá gestionar y administrar desde la siguiente pantalla.

Página: Modificación Calendarios

Siempre que se haya seleccionado en la pantalla anterior el año presente o futuro se podrán modificar los calendario previamente creados, además de realizar los siguientes tipo de operaciones:

- **Asociar y desasociar cuentas** de cotización, las cuales cada una de ellas está asociada a N empleados/as.

- **Modificar** los días de los empleados/as, incrementando o decrementando el número total de días festivos que podrá solicitar, siendo condición inicial el número de días festivos del calendario.
- **Asociar y desasociar días** no laborales/festivos al calendario.

Página: Consulta General Calendarios

En esta última página, en lo que a la administración se refiere, se podrá visualizar la información de todos los empleados/as asociados a alguno de los calendarios existentes para el año actual.

5.3.1.2 Mis Vacaciones

Esta funcionalidad está enfocada a todos los trabajadores/as de tipo usuario. Las operaciones de esta funcionalidad se llevarán a cabo desde una única página en la que se mostrará dependiendo del perfil de usuario más o menos vistas, lo que implica mayor o menor funcionalidad.

Vista: Calendario

En esta primera vista se mostrará una leyenda indicando el resumen de las vacaciones del usuario además de una opción “Ver Detalle” que describiremos más adelante. Bajo la leyenda se mostrará un calendario anual mediante el que se podrán realizar las diferentes solicitudes de vacaciones durante el año.

Vista: Ver Detalles

Mostrará un listado por estados de las diferentes solicitudes realizadas. Cada uno de los intervalos, agrupados por estado, se podrán modificar, eliminar o enviar dependiendo del estado en el que se encuentre la solicitud.

- **Modificar:** actualiza las fechas de la solicitud, con las nuevas fechas de inicio y fin. Una vez realizada la operación, la solicitud, pasará al estado pendiente de enviar.
- **Eliminar:** modificará su estado a -1, desaparecerá de la interfaz pero no de la base de datos.
- **Enviar:** la solicitud pasará del estado pendiente de enviar al siguiente estado, pendiente de aprobar.

Vista: Visualizar Departamento

Esta vista estará disponible para aquellos usuarios de tipo aprobador. En ella se mostrará un listado de trabajadores/as de un departamento, junto con columnas mensuales que contienen el número de días de vacaciones, aprobadas y/o pendientes de aprobar, por mes junto al total de días.
Además mostrará a detalle los días de cada solicitud por cada uno de los empleados/as de forma mensual.
El visualizado dependerá de los siguientes filtros:

- **Todos:** se mostrarán los mapas temporales previamente descritos con aquellas solicitudes con estado aprobado y pendientes de aprobar.
- **Aprobados:** se mostrarán exclusivamente aquellas solicitudes con estado aprobado.

Vista: Revisar Vacaciones

Mostrará un listado de aquellos trabajadores/as de los que el usuario aprobador es responsable, siempre y cuando los trabajadores/as tengan asociadas solicitudes en un estado pendiente de aprobar, rechazado,aprobado o eliminado.
Dispondrá de las siguientes funcionalidades:

- **Aprobar la solicitud**, modificando el estado de esta a estado aprobada.
- **Rechazar una solicitud**, modificará su estado a rechazada y añadirá el motivo de su rechazo.
- **Aprobar en conjunto**, podrá aprobar todas las solicitudes asociadas a un empleado/a de manera conjunta.

Además esta vista proporcionará un indicador en la parte superior mostrando el número de solicitudes pendientes por procesar en caso de existir.

Vista: Visualizar Vacaciones

Esta última vista mostrará la información de aquellas solicitudes aprobadas de un departamento.
Ofrecerá de forma común las siguientes funcionalidades:

- Agrupar y desagrupar los datos de la pantalla
- Descargar en formato excel la información de las solicitudes
- Realizar búsquedas de texto.

Además, si el usuario pertenece al comité ejecutivo mostrará de la misma forma que se muestran las solicitudes aprobadas del departamento, las solicitudes asociadas a los integrantes del comité ejecutivo en caso de existir.

Finalmente, esta vista proporcionará dos funcionalidades más, exclusivamente al responsable del departamento de recursos humanos.

- Descargar datos en formato Excel a Castilla
- Indicar envío en Castilla

A través de estas funcionalidades se llevará a cabo el envío de notificaciones una vez los datos hayan sido consolidados y facilitará el proceso de carga de datos en el programa de nóminas.

5.3.2 Permisos

En este apartado se muestran los permisos, en relación a las páginas y vistas descritas anteriormente, que tiene cada empleado/a en función del rol y perfil que desempeñe.

	Calendario y Ver Detalles	Visualizar Departamento	Revisar Vacaciones	Visualizar Vacaciones
Visualizador				x
Solicitante	x			x
Aprobador	x	x	x	x

	Administración Calendarios	Modificación Calendarios	Consulta General Calendarios
Administrador	x	x	x

Tabla 5.7 *Permisos según rol y perfil*

5.4 Arquitectura del sistema

A continuación se describe el diseño de los siguientes apartados mostrando la arquitectura lógica de la aplicación y el diseño del nuevo endpoint en el Web Service.

5.4.1 Modelo

El modelo de la arquitectura lógica consta de los siguientes componentes que posteriormente serán descritos: SIU Vacaciones, Web Service Organigrama, Base de Datos AS/400. A través de ellos la aplicación será ejecutada, haciendo uso de servicios REST [1] o bien accediendo directamente a la base de datos del servidor.

5.4.1.1 Componentes

SIU Vacaciones: Es el proyecto en el que se llevará a cabo el desarrollo principal. Dentro de la aplicación SIU se integrarán las vistas y funcionalidades previamente definidas. Respecto a la funcionalidad “Mis vacaciones”, se realizará una llamada al web service para obtener los datos necesarios y ejecutar la aplicación de forma correcta.
Desde este componente también se accederá a la base de datos, donde se realizarán las operaciones correspondientes dependiendo del rol del empleado/a y vista en la que se encuentre.

[1] Representational State Transfer

Web Service Organigrama: Se creará un nuevo endpoint en el proyecto del WEB Service de forma que, previamente al acceso de la funcionalidad "Mis Vacaciones", se realizará una llamada a este para la obtención de información del empleado/a y así posteriormente asignar al trabajador/a uno de los roles/perfiles previamente mencionados.
De esta forma es como se obtendrá la información para proporcionar el acceso a las vistas y funcionalidades correspondientes.

Base de Datos AS/400: Es la capa de datos, donde se almacenan las solicitudes, calendarios y se obtiene la información de los empleados/as y departamentos. En ella se integrarán las tablas anteriormente expuestas para la gestión y administración de los datos de esta nueva funcionalidad.

Para obtener una visión más general acerca del modelo de la arquitectura lógica, véase la siguiente imagen.

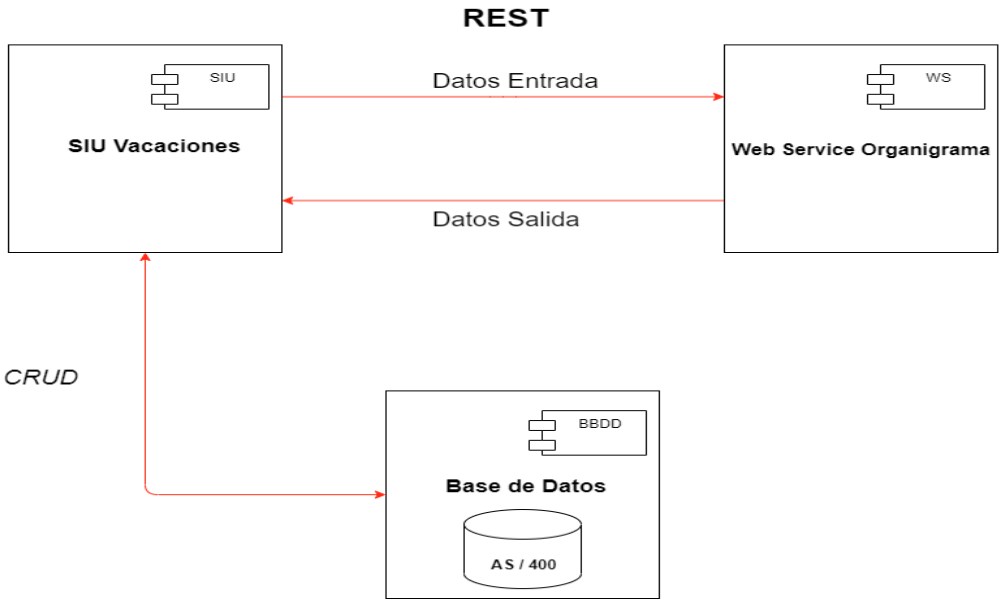


Fig.5.4 *Arquitectura lógica del sistema*

5.4.2 EndPoint

El endpoint a desarrollar, se llamará desde la herramienta SIU para obtener la información necesaria del usuario y poder asignarle el rol y perfil, calendario y revisor correspondiente además de proporcionarle mayor o menor funcionalidad en base a la información obtenida.

El intercambio de datos y comunicaciones se realizarán a través de servicios **REST**, es decir, cualquier interfaz que utilice el protocolo HTTP [2] para obtener datos o generar operaciones sobre esos datos en los siguientes tipos de formato JSON [3] o XML [4], siendo este primero el más utilizado ya que es más ligero y legible. [OpenWebinars, 2018]

---

[2] Hypertext Transfer Protocol
[3] JavaScript Object Notation
[4] Extensible Markup Language

Para ello se ha definido el siguiente endPoint con la siguiente cabecera

/WSOrganigrama/m2/revisor/{Código del Usuario}/

El método utilizado para el envío de los datos es **GET,** de forma que los parámetros se pueden guardar junto a la dirección URL[5]. En el siguiente anexo B.1 se puede observar el código del endPoint desarrollado.

A través de este nuevo endPoint se obtiene la siguiente información:

- **Usuario :** Nivel, departamento, código del departamento, puesto, dependencia del usuario en el organigrama de la empresa y el área al que corresponde.
- **Departamento:** Identificador del departamento dependiente, el nivel de su responsable, el nivel del siguiente responsable
- **Revisor:** Código del revisor

En la siguiente imagen se muestra, en formato JSON, la estructura de los datos obtenidos al realizar una solicitud al servidor a través de PostMan.


```
{
    "estado": 1,
    "vac": {
        "user": {
            "nivel": 5,
            "depto": "Comercial
            "cod_depto": 2,
            "puesto": "GERENTE SUPER AMARA
            "dependencia": 102,
            "id_area": "-1 "
        },
        "depto": {
            "id_puesto": "01102    ",
            "nivel": 3,
            "id_depende": "45"
        },
        "revisor": {
            "codper": "02300011"
        }
    }
}
```


Fig.5.5 *Estructura y datos JSON*

En el siguiente capítulo se muestra la implementación del nuevo endPoint, entiendo mejor su funcionamiento y obtención de información.

[5] Universal Resource Locator

6

Implementación de la aplicación

En este capítulo se mostrará la implementación y desarrollo de aquellos elementos y componentes necesarios, más relevantes, a la hora de ejecutar esta nueva funcionalidad.

6.1 Implementación de tablas en AS/400

El sistema AS/400 es un equipo desarrollado por IBM [1]. Es un sistema multiusuario, con una interfaz controlada mediante menús y comandos CL [2] que utiliza terminales y un sistema operativo basado en objetos y bibliotecas, denominado OS/400.

Uno de los aspectos más importantes del sistema operativo es su integración con la base de datos relacional DB2/400, en la cual se han creado las tablas expuestas en el anterior capítulo.

DB2 es un motor de bases de datos enfocado a sistemas multiusuarios, es decir, permite que muchos usuarios concurrentes accedan a los mismos datos de forma simultánea.
Este se encuentra disponible en las siguientes plataformas principales: Linux, Unix y Windows. El uso y gestión que realice el administrador será diferente dependiendo de la plataforma en la que se encuentre, sin embargo, cuenta con una alta compatibilidad con SQL por lo que a pesar de realizar un uso y gestión de forma diferente, los desarrolladores pueden escribir código SQL que probablemente funcione en cualquiera de las plataformas realizando pocos o ningún cambio [ComputerWeekely, 2021].

Una vez conocido el entorno de desarrollo del servidor en el siguiente anexo B.4 se muestran los DDL [3] para la creación de las tablas en SQL definidas en el capítulo de diseño de la aplicación.

Sin embargo, cabe destacar que las tablas no han sido creadas directamente desde sentencias SQL, estas previamente han sido creadas en scripts que posteriormente han sido introducidos en la interfaz del AS/400 para su creación.

En las siguientes imágenes se muestra la descripción de campos y estructura de los scripts[Programación Conceptos de DDS, 2014]

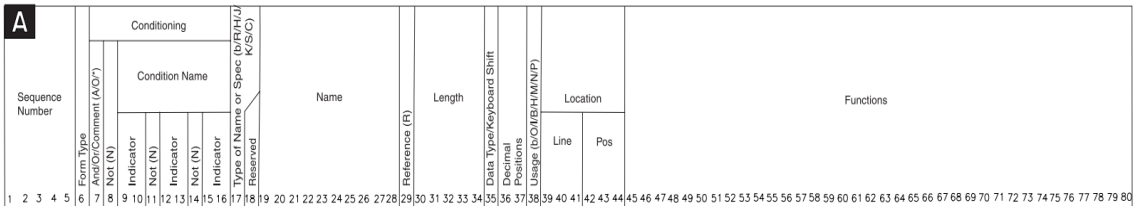


Fig.6.1 *Descripción y especificación de datos AS/400*

[1] *https://www.ibm.com/es*
[2] *Control Language*
[3] *Data Definition Language*

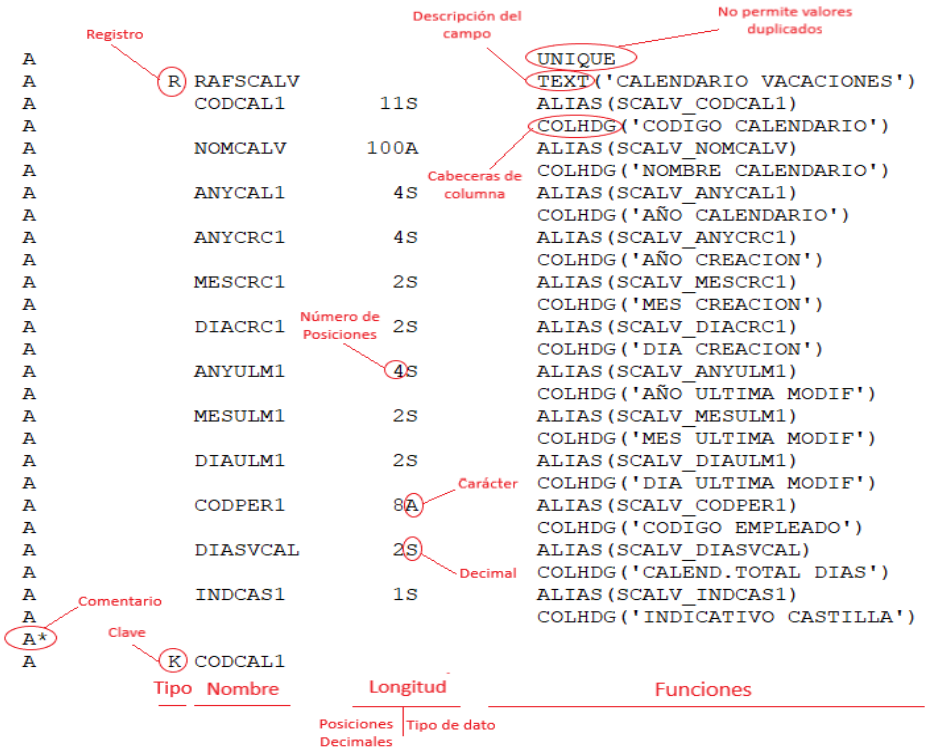


Fig.6.2 *Estructura y datos archivo físico AS/400*

Los documentos con las secuencias de comandos, de cada una de las tablas para la introducción en la interfaz del servidor, se encuentran en el siguiente anexo B.5.

6.2 Implementación del EndPoint en el WEB Service

El nuevo endPoint ha sido desarrollado sobre un WEB Service ya existente llamado "WSOrganigrama". El proyecto ha sido desarrollado con Jersey, framework utilizado para el desarrollo de aplicaciones a través de un marco REST que amplía las herramientas proporcionadas por JAX-RS a través de una API[4], facilitando y simplificando el servicio RESTful. En el siguiente anexo A.1 se muestra la estructura lógica de un servicio REST [Eclipse Jersey].

6.2.1 Diagrama de secuencia

A través de este nuevo endPoint obtendremos los datos del empleado/a, previamente mencionados en el capítulo de diseño, de forma que antes de acceder a la funcionalidad "Mis Vacaciones" se conozca el revisor que le corresponde al trabajador/a entre otros datos además de mostrar las vistas correspondientes.

En la siguiente imagen se muestra el diagrama de secuencias que se lleva a cabo para acceder a la funcionalidad.

[4] Application Programming Interface

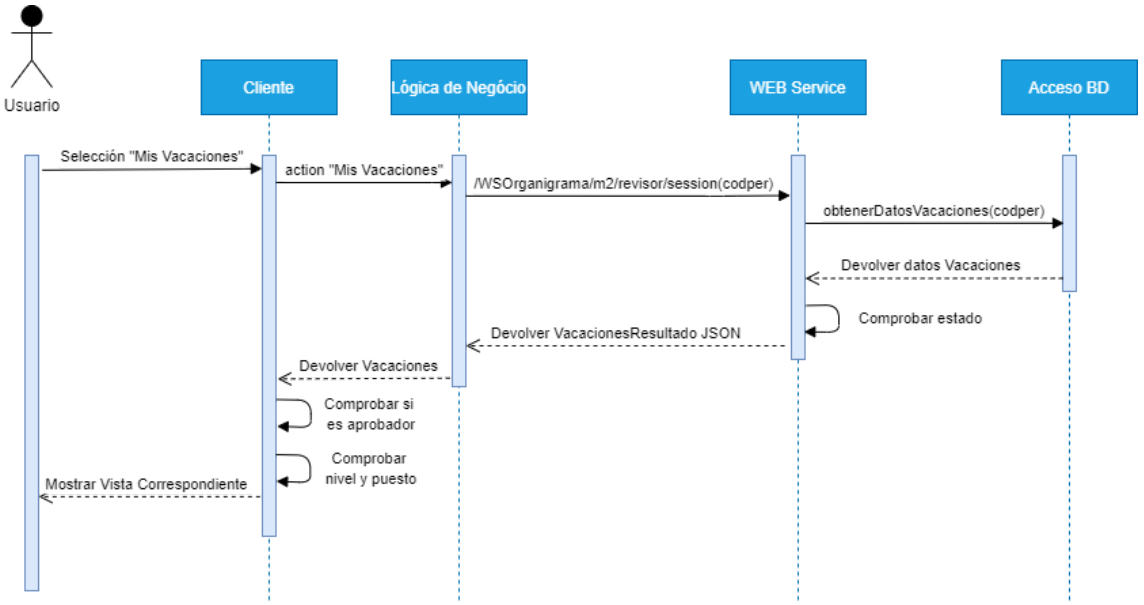


Fig.6.3 *Diagrama de secuencia acceso Mis Vacaciones*

6.2.2 Pseudocódigo

Una vez visto el diagrama de secuencia para el acceso a la funcionalidad, el siguiente pseudocódigo muestra el funcionamiento de este nuevo endPoint y cómo se obtienen los datos para transformarlos posteriormente a formato JSON, no obstante en el siguiente anexo B.2 se puede observar el código java original desarrollado para la obtención de información en base a las reglas establecidas.

```
public Vacaciones obtenerDatosVacaciones(String codper) {
Vacaciones vac = new Vacaciones();
usrV = obtenerUsuarioVacaciones(codper);
      if (usrV.Nivel  > 2 && !usrV.Puesto=="DIRECTOR/A RRHH") {
            if (usrV.Nivel <= 4 || usrV.Puesto=="SECRETARIO/A DIRECCION GENERAL" || usrV.Puesto=="GERENTE ORGANIZACION") {
                  dependencia = obtenerDeptoDependiente( Codigo Depto DIRECTOR/A RRHH );

                usrR = obtenerRevisor(dependencia.Id Puesto , usrV.Id Area);
            } else {
                  dependencia = obtenerDeptoDependiente(usrV.Depto Dependiente);
                  if (usrV.Codigo Depto == 1 || usrV.Codigo Depto == 12) {
                        while (dependencia.Nivel Depto Dependiente > 6) {

                        dependencia = obtenerDeptoDependiente(dependencia.Depto Dependiente);
                        }
                  usrR = obtenerRevisor(dependencia.Id Puesto , usrV.Id Area);
                  } else {
                        while (dependencia.Nivel Depto Dependiente  > 5) {
                                    dependencia = obtenerDeptoDependiente(dependencia.Depto Dependiente);
                        }
                  usrR = obtenerRevisor(dependencia.Id Puesto, "-1 ");
                  }
            }
      }
return vac ;
}
```

Trás ver el pseudocódigo, en los siguientes anexos B.1, B.2 y B.3 se muestra el código completo que se ejecutará al realizar la llamada al endPoint del WEB Service además de la transformación de aquellos datos devueltos a formato JSON, haciendo uso de las siguientes librerías de google : GsonBuilder y Gson.

Mediante estas librerías lo que nos permiten realizar es la transformación de objetos Java a JSON y viceversa. Cabe destacar que se hace uso de la librería GsonBuilder además de Gson ya que se realizan configuraciones de implementación diferentes a las predeterminadas de la clase Gson. [HowTo In Java, 2022]

6.3 Implementación del Cliente

El front-end es la parte con la que interactúa el usuario final, es decir, el empleado/a de las oficinas. A través de la interfaces o vistas se establece la comunicación con el servidor de forma que se puedan llevar a cabo las diferentes operaciones proporcionadas por la aplicación. Para ello se han desarrollado las siguientes páginas .jsp

6.3.1 Páginas

La nueva funcionalidad cuenta con un total de cuatro páginas diferentes.Estas han sido desarrolladas con las siguientes tecnologías: HTML, CSS, JQuery y Bootstrap.

Las siguientes páginas únicamente las podrán visualizar aquellos usuarios que desempeñen el rol de administrador:

- administradorCalendarios.jsp
- administradorCalendariosEditar.jsp
- consultaEmpleadosGeneral.jsp

Aquellos trabajadores/as que desempeñen el rol de usuario tendrán acceso a esta última página desarrollada:

- vacaciones.jsp

Para esta última página dependiendo del nivel y puesto del usuario se le mostrarán más o menos vistas, lo que implica mayor o menor funcionalidad. En el siguiente código podemos observar cómo se ha implementado el control para mostrar las diferentes vistas.

```
$(document).ready(function() {
      visualizarTabs($('#aprobadorh').val(), $('#nivelh').val());
      if($('#nivelh').val()<=2 || ($('#codDepartamentoh').val() == 12 && $('#nivelh').val()==4) ){
            $("#tab4").trigger('click');
            $( "#localtabs" ).tabs({ active: 3 });
      }
});
 function visualizarTabs(aprobador, nivel){
      if(aprobador=='true' ){
            $("#tab3").show();
            $("#tab2").show();
      }
      if(nivel <= 2 || ($('#codDepartamentoh').val() == 12 && nivel==4)){ $("#tab1").hide(); }
}
```

En el siguiente anexo C se muestra cada una de las páginas desarrolladas junto a las diferentes vistas que proporcionan las funcionalidades, véase la figura 5.3 .

6.3.2 Componentes

En este apartado se describirán los componentes más relevantes utilizados en el desarrollo de las interfaces. Para ello se han incluido las siguientes librerías en las páginas .jsp

```
<%@ taglib prefix="s" uri="/struts-tags"%>
<%@ taglib prefix="sj" uri="/struts-jquery-tags"%>
<%@ taglib prefix="sjg" uri="/struts-jquery-grid-tags"%>
```

Fig.6.4 *Librerías de Struts*

Grids o rejillas

Este componente se ha utilizado principalmente para mostrar las información asociada a los calendarios, es decir, ha sido utilizado en las páginas de administración. En la siguiente imagen se muestra un ejemplo de código de este componente.

```
<s:url var="urlGridCCCcalendario" action="leerCCC_PersonasGeneral"></s:url>
<div class="table-responsive">
    <sjg:grid
        id="gridEmpleadosGeneral"
        gridModel="gridCCC_PersonasGeneral"
        dataType="json"
        loadonce="true"
        href="%{urlGridCCCcalendario}"
        caption=""
        gridview="true"
        pager="true"
        rowList="15,20,35"
        rowNum="30"
        shrinkToFit="true"
        autowidth="true"
        viewrecords="false"
        sortable="true"
        hidegrid="false"
        viewsortcols="[true,'vertical',true]"
        navigatorSearchOptions="{multipleSearch:true}"

        navigator="true"
        navigatorSearch="true"
        navigatorAdd="false"
        navigatorRefresh="false"
        navigatorEdit="false"
        navigatorDelete="false"
        onCompleteTopics=""
        onSelectRowTopics=""

        scroll="true"
        height="700"
        scrollrows="true"
    >
    <sjg:gridColumn name="gnombre" index="gnombre" title="Nombre" sortable="true" align="center" searchoptions="{sopt:['eq','ne','cn','nc']}" />
     <sjg:gridColumn name="gcodemp" index="gcodemp" title="Cod. Empresa" sortable="true" align="center" searchoptions="{sopt:['eq','ne','cn','nc']}"/>
     <sjg:gridColumn name="gcodPer" index="gcodPer" title="Cod. Operario" sortable="true" align="center" searchoptions="{sopt:['eq','ne','cn','nc']}"/>
    <sjg:gridColumn name="gccc" index="gccc" title="CCC" sortable="true" align="center" searchoptions="{sopt:['eq','ne','cn','nc']}"/>
     <sjg:gridColumn name="gcalendario" index="gcalendario" title="Calendario" sortable="true" align="center" searchoptions="{sopt:['eq','ne','cn','nc']}" />
    <sjg:gridColumn name="gdias" index="gdias" title="Días asociados" sortable="true" align="center" searchoptions="{sopt:['eq','ne','cn','nc']}" />
    </sjg:grid>
```

Llamada action

Atributos

Columnas

Fig.6.5 *Estructura e implementación componente Grid*

Tabbed Panel

Es un componente contenedor que permite al usuario cambiar de vista haciendo clic en las diferentes pestañas. Este se ha utilizado para mostrar mayor o menor funcionalidad dependiendo del rol y perfil del trabajador/a, mostrándola de forma separada y ordenada.

```
<sj:tabbedpanel id="localtabs" cssClass="uves-bloques-tab">

<sj:tab id="tab1" target="divMisVacaciones" label="Mis Vacaciones"/>
        <!-- CÓDIGO -->
<sj:tab id="tab2" target="divVisualizarDepartamento" label="Visualizar Departamento" cssStyle="display:none;"/>
        <!-- CÓDIGO -->
<sj:tab id="tab3" target="divRevisarVacaciones" label="Revisar Vacaciones"   cssStyle="display:none;" />
            <!-- CÓDIGO -->
<sj:tab id="tab4" target="divVisualizarVacaciones" label="Visualizar Vacaciones" />
            <!-- CÓDIGO -->
</sj:tabbedpanel>
```

Fig.6.6 *Estructura e implementación componente Tabbed Panel*

Modal

Este componente se ha utilizado para notificar avisos o adición de funcionalidades. Es una ventana emergente que se incorpora sobre cualquier página, en ella se pueden añadir cuadros de diálogo entre otros objetos. Su estructura a la hora de implementarlo es la siguiente.

```
<div id="X" class="modal" tabindex="-1" role="dialog">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
            <!-- OBJETOS A AÑADIR EN LA CABECERA DEL MODAL -->
      </div>
      <div class="modal-body">
         <!-- OBJETOS A AÑADIR EN LA PARTE CENTRAL DEL MODAL -->
      </div>
      <div class="modal-footer">
         <!-- OBJETOS A AÑADIR LA PARTE INFERIOR DEL MODAL -->
      </div>
    </div>
  </div>
</div>
```

Fig.6.7 *Estructura e implementación componente Modal*

Calendario

A través del siguiente código se muestra la creación del calendario principal correspondiente al año actual. En el siguiente anexo B.6 se puede ver el código completo de la función que genera el calendario además de añadir los días con el color correspondiente en base a su estado e indicando si es o no laboral.

```
$.post('/SIUC2-Vacaciones/obtenerDiasFestivosCalendario.action', {
      codCal :$.codigoCal,
}).done(function (data){
      $.listaDiasFF = data.calB.fechasFestivos;
      if($.listaDiasFF.length == 0){
            console.log("Atención: No hay días festivos para ese calendario");
      }else{
            $.each($.listaDiasFF, function(i,d){
                  diasAsignados.push({
                  startDate: new Date(d.fechaIni),
                  endDate: new Date(d.fechaFin),
                  color: '#e4edec'
                  });
            });
      }

      var anyoActual= new Date().getFullYear();
      var primerDia =new Date(anyoActual, 0, 1);
      var ultimoDia =new Date(anyoActual, 11, 31);

      const calendarAnual = new Calendar('#calendarioDiasDescansoAnual', {
            style:'background',
            language: 'es',
            enableRangeSelection: false,
            minDate: primerDia,
            maxDate: ultimoDia,
            displayHeader: true,
            customDayRenderer: function(element, date) {
                  if (date.getDay() === 6 || date.getDay() === 0) {
                        $(element).css('background-color', '#e4edec');
                        $(element).css('border-radius', '0px');
                        }
                  },
            });
      calendarAnual.setDataSource(diasAsignados);
});
```

6.4 Implementación del Servidor

Es la parte que se encuentra activa en todo momento para responder ante cualquier solicitud o petición realizada por el cliente. La empresa cuenta con un total de tres servidores, sin embargo, para llevar a cabo el desarrollo en local se ha utilizado un servidor Tomcat v7.0 mediante el que se han realizado las diferentes conexiones a los servidores de Uvesco para la obtención de los datos respecto a las tablas implementadas.

Fig.6.8 *Proyecto y estructura del servidor*

Cabe destacar que el proyecto ha sido desarrollado con Struts 2. En el siguiente anexo A.2 podemos observar su arquitectura de forma detallada, no obstante en el siguiente apartado se describe el ciclo que sigue al procesar una petición HTTP de forma que se entiendan con mayor claridad los siguientes apartados.

6.4.1 Ciclo de una petición HTTP con Struts

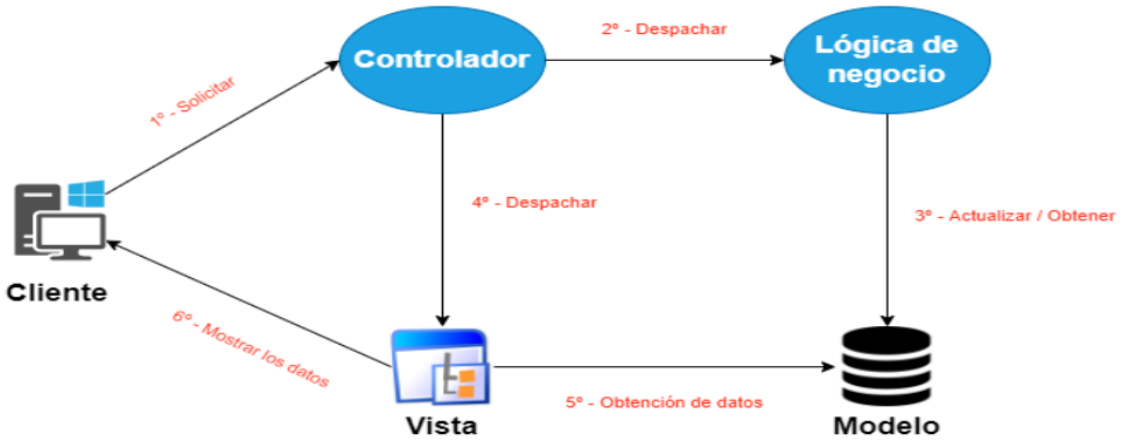


Fig.6.9 *Ciclo de una petición en Struts*

Inicialmente el cliente lanza una solicitud HTTP que recibirá el controlador de Struts, la petición no es una URL sino un nombre simbólico el cual indica una acción en el fichero struts.xml .
El controlador resuelve la acción concluyendo con ella, identificando y disparando la lógica de negocio correspondiente.
La lógica de negocio actualiza y/o obtiene los datos del modelo en base al método solicitado en el anterior fichero.
Dependiendo del valor que la lógica de negocio devuelva, el controlador mostrará la vista correspondiente, en nuestro caso las páginas .jsp.
Finalmente la vista toma los datos y los muestra al cliente, concluyendo así con la petición realizada.[Universidad de Alicante, 2014]

6.4.2 Estructura del proyecto

El proyecto ha sido estructurado de la siguiente forma, véase en la siguiente imagen

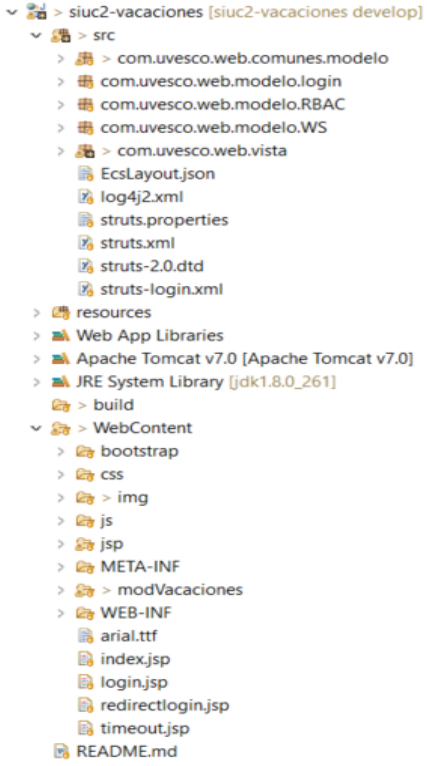


Fig.6.10 *Estructura del proyecto*

A continuación, se detallan aquellos paquetes, archivos y carpetas con mayor importancia en el desarrollo del proyecto.

6.4.2.1 Paquetes

Los nombres utilizados para los paquetes ha sido en base a los demás proyectos ya existentes en la empresa, de manera que todos continúen con una misma línea y nomenclatura.

- El paquete **com.uvesco.web.comunes.modelo** contiene las clases que accederán a la capa de datos.
- El siguiente paquete **com.uvesco.web.modelo.WS** contiene las clases que realizarán la llamada al endPoint del WEB Service.
- El contenido del paquete **com.uvesco.web.vista** hace referencia los objetos que se crearán cuando struts ejecute desde el fichero struts.xml la acción correspondiente en la lógica de negocio.

6.4.2.2 Archivos

El archivo **struts.xml** contiene la información de configuración que se ejecutará a medida que se desarrollen las acciones.
En la arquitectura que Struts ofrece,MVC, este archivo realiza la función de controlador, junto al archivo **web.xml** del servidor.

6.4.2.3 Carpetas

Dentro de la siguiente carpeta **WebContent,** se encuentran las carpetas del contenido web:

- **modVacaciones,** en ella se han desarrollado las páginas .jsp previamente vistas.
- La carpeta **WEB-INF** contiene las librerías necesarias para la ejecución del proyecto como las librerías de Struts, transformación de los datos a JSON, registro de Logs, conexiones con las bases de datos etc.

En las demás carpetas que se pueden observar en la figura 6.10 se encuentran los archivos relacionados con el desarrollo de la parte del cliente como funciones, estilos etc.

6.4.3 Conexiones

Las conexiones con los diferentes servidores de la empresa se han realizado a través del fichero, del servidor local Tomcat v7.0, **context.xml**. Mediante este archivo se ha indicado al servidor local a cual de los tres servidores queremos conectarnos al ejecutar el entorno de desarrollo. En la siguiente imagen se muestra el código XML necesario para realizar la conexión a uno de los tres servidores.

```
<!-- DESARROLLO AS400  -->

    <Resource auth="Container" driverClassName="com.ibm.as400.access.AS400JDBCDriver" initialSize="0" maxActive="8" maxIdle="8" maxWait="3000" minIdle="1"
      name="NOMBRE DEL SERVIDOR" password="CONTRASEÑA" removeAbandoned="true" removeAbandonedTimeout="600" type="javax.sql.DataSource"
      url="URL CONEXION AL SERVIDOR" username="NOMBRE USUARIO" />
```

Fig.6.11 *Conexión e implementación al servidor*

Además se ha realizado la siguiente conexión a MySQL, a una base de datos local, para detectar y analizar los posibles errores surgidos durante el desarrollo a través del registro de los Logs[5] en los programas [Humio, 2021].

```
<!-- MySQL local para logs -->

    <Resource name="jdbc_MYSQLLocalhost" auth="Container"  maxActive="8" maxIdle="8" minIdle="1" maxWait="10000" initialSize="0"
        type="javax.sql.DataSource"
        driverClassName="com.mysql.jdbc.Driver"
        username="NOMBRE USUARIO" password="CONTRASEÑA"
        url="CONEXION A LA MÁQUINA LOCAL" validationQuery="SELECT 1" testOnBorrow="true" />
```

Fig.6.12 *Conexión e implementación a MySQL*

6.4.4 Operaciones

En este apartado se mostrarán las operaciones más relevantes que conforman la aplicación.

6.4.4.1 Acceso a datos

A través de las siguientes operaciones se han realizado las consultas y actualizaciones en la base de datos del servidor, estas se caracterizan por ser de los siguientes tipos : INSERT, UPDATE, SELECT y DELETE. Para su implementación principalmente se ha utilizado el lenguaje SQL.

En las siguientes imágenes se muestran algunos ejemplos del desarrollo de estas operaciones además del código java de una de las funciones:

[5] "Grabación secuencial en un archivo o en una base de datos de todos los acontecimientos que afectan a un proceso concreto"[Wikipedia, 2021]

```
String sql = "SELECT COUNT(*)  FROM AFSCALVS WHERE CODPER1 = ?  AND CODEST1 IN  (0,1,2)  ";
       sql+="AND (((CAST (ANYINI1||'-'||MESINI1||'-'||DIAINI1 AS DATE) BETWEEN '"+fIni+"' AND '"+fFin+"' ";
       sql+="OR CAST (ANYFIN1||'-'||MESFIN1||'-'||DIAFIN1 AS DATE) BETWEEN '"+fIni+"' AND '"+fFin+"') )  ";
       sql+="OR ( '"+fIni+"' BETWEEN CAST (ANYINI1||'-'||MESINI1||'-'||DIAINI1 AS DATE) AND CAST (ANYFIN1||'-'||MESFIN1||'-'||DIAFIN1 AS DATE) ) )  ";
```

Fig.6.13 *Implementación SQL detección solapes entre fechas*

```
conn = daoC.getConection("DATEXP");
String  sql = "WITH TMPC(ID, NIVEL,DEPENDE,DESDEP1,D_AREAS,DPUEST,IDPUES,ID_PUESTO,ID_EMPRESA,COD_DEP,ID_AREA,CODDEP1) ";
        sql += "AS (SELECT A.ID,A.NIVEL,A.DEPENDE,B.DESDEP1,C.D_AREAS,D.DPUEST,D.IDPUES,E.ID_PUESTO,E.ID_EMPRESA, B.SDEPAR_CODDEP1,A.SORGAN_DEP_AREA,B.CODDEP1 ";
        sql += "FROM AFSORGAN A ";
        sql += "JOIN AFSDEPAR B ON B.CODDEP1 = A.ID_DEPART ";
        sql += "JOIN [illegible]/AREAS C ON C.ID_AREAS = A.ID_AREA ";
        sql += "JOIN [illegible]/NLTPUEST D ON D.IDPUES = A.ID_PUESTO ";
        sql += "JOIN [illegible]/PUESTOS E ON E.ID_PUESTO_TIPO = A.ID_PUESTO AND A.ID_AREA = E.ID_AREA AND E.ID_AREA IS NOT NULL UNION ";
        sql += "SELECT A.ID,A.NIVEL,A.DEPENDE,B.DESDEP1,'-1' ,D.DPUEST,D.IDPUES,E.ID_PUESTO,E.ID_EMPRESA,B.SDEPAR_CODDEP1,A.SORGAN_DEP_AREA ,B.CODDEP1 ";
        sql += "FROM AFSORGAN A JOIN AFSDEPAR B ON B.CODDEP1 = A.ID_DEPART ";
        sql += "JOIN [illegible]/NLTPUEST D ON D.IDPUES = A.ID_PUESTO ";
        sql += "JOIN [illegible]/PUESTOS E ON E.ID_PUESTO_TIPO = A.ID_PUESTO AND A.ID_AREA = '-1' AND E.ID_AREA IS NULL ) ";
        sql += "SELECT  CONCAT(RIGHT(G.NUMEM,3),RIGHT(G.NUMPR,5 )) AS CODPER, ";
        sql += "IFNULL(TRIM(I.NOMBRE), '-')||' '||IFNULL(TRIM(I.APELLIDO1), '')||' '||IFNULL(TRIM(I.APELLIDO2), '') AS NOMBRE, ";
        sql += "CASE IFNULL(L.DIASVCAL, 0)  WHEN 0 THEN M.DIASVCAL  ELSE L.DIASVCAL END AS DIASCAL, M.CODCAL1 ";
        sql += "FROM TMPC E JOIN [illegible]VIV_ORG F ON CONCAT(E.ID_EMPRESA, E.ID_PUESTO) = CONCAT(F.ID_EMPRESA, F.ID_PUESTO) ";
        sql += "JOIN [illegible]NLPUEST G ON F.ID_NIVEL = G.NIVEL AND G.PDEFECT = 1 AND (FINIP <= CURRENT_DATE AND (G.FFINP IS NULL ";
        sql += "OR (G.FFINP IS NOT NULL AND G.FFINP >= CURRENT_DATE ))) ";
        sql += "JOIN [illegible]/COD_TRA H ON CONCAT(RIGHT(G.NUMEM,3),RIGHT(G.NUMPR,5 )) = CONCAT(RIGHT(H.ID_EMPRESA,3),RIGHT(H.ID_TRABAJADOR,5)) ";
        sql += "JOIN [illegible]/PERSONAS I ON H.NIF = I.NIF ";
        sql += "LEFT JOIN AFTRTE J ON CONCAT(RIGHT(G.NUMEM,3),RIGHT(G.NUMPR,5 ))=CONCAT(RIGHT(J.CODEMP1,3),RIGHT(J.CODOPE1,5 )) ";
        sql += "LEFT JOIN AFSCALCCC K ON K.NUMCCC1=J.TRTE_CUECOT1 ";
        sql += "LEFT JOIN AFSCALPER L ON L.CODPER1= CONCAT(RIGHT(J.CODEMP1,3),RIGHT(J.CODOPE1,5 )) ";
        sql += "LEFT JOIN AFSCALV M ON K.CODCAL1 = M.CODCAL1 ";
        sql += "LEFT JOIN AFSUSR O ON O.SUSR_CODPER1= CONCAT(RIGHT(G.NUMEM,3),RIGHT(G.NUMPR,5 ))  ";

        if(codDepto == 7 ) { // DEPTO = LOGÍSTICA , PUEDEN EXISTIR NIVELES 8,9 Y 10 CON SIU POR LO QUE SE FILTAR
            sql += "WHERE E.CODDEP1 IN (?) AND M.ANYCAL1=YEAR(CURDATE()) AND O.ACTPER1=1 AND  E.NIVEL NOT IN (8,9,10) ORDER BY NOMBRE ";
        }else if(codDepto == 2 ) { //DEPTO COMERCIAL SE EXCLUYE TODOS LOS MPLEADOS DE BARBED EXCEPTO "JEFE/A TRADICIONAL BARBED" QUE NO CUENTA CON AREA = 'BAR'
            sql += "WHERE E.CODDEP1 IN (?) AND M.ANYCAL1=YEAR(CURDATE()) AND O.ACTPER1=1 AND E.ID_AREA <> 'BAR' ORDER BY NOMBRE ";
        }else { // FILTRO PARA AQUELLOS EMPLEADOS CON SIU
            sql += "WHERE E.CODDEP1 IN (?) AND M.ANYCAL1=YEAR(CURDATE()) AND O.ACTPER1=1 ORDER BY NOMBRE ";
        }
```

Fig.6.14 *Implementación SQL visualización departamento anual*

```
/**
* Modifica el estado de una solictud, a estado "PENDIENTE DE REVISAR (1)", de forma que pueda ser procesada por
 * el revisor correspondiente
 * @param codSol = CODIGO DE LA SOLICITUD
 */
      public void enviarSolicitudDAO(long codSol) {
          Connection conn = null;
          PreparedStatement pstmt = null;

          DaoConexion daoC = new DaoConexion();

            try {
                  conn = daoC.getConection('[illegible]');

                  String sql = "UPDATE AFSCALVS SET CODEST1=1 WHERE CODVAC1=?";

                  pstmt = conn.prepareStatement(sql);
                  pstmt.setString(1, String.valueOf(codSol) );

                  pstmt.executeUpdate();

                if (pstmt != null) try { pstmt.close(); } catch (SQLException logOrIgnore) {}
                if (conn != null) try { conn.close(); } catch (SQLException logOrIgnore) {}

                  } catch (SQLException e) {
                        logger.error("enviarSolicitudDAO ERROR:"+e.getMessage());
                  }
                  finally
                  {
                      if (pstmt != null) try { pstmt.close(); } catch (SQLException logOrIgnore) {}
                      if (conn != null) try { conn.close(); } catch (SQLException logOrIgnore) {}
                  }
      }
```

6.4.4.2 Envío de Notificaciones

Esta operación se realizará una vez se haya alimentado el programa de nóminas e indicado la consolidación de los datos respecto a las diferentes solicitudes de las vacaciones realizadas por todos los trabajadores/as.
De esta forma lo que se pretende es llevar a cabo un seguimiento sencillo de aquellas solicitudes creadas o modificadas fuera del plazo, además de mantener informado tanto al solicitante, revisor y departamento de recursos humanos para su posterior modificación en el programa de nóminas.

En el siguiente código se muestra la formación del correo cuando un revisor aprueba o rechaza una solicitud posteriormente al indicar la consolidación de los datos en Castilla.

```
case Email.SUPERVISOR_EMPLEADO: // CORREO DEL REVISOR AL EMPLEADO = 2 (AL PROCESAR CUALQUIER SOLICITUD)

        para = "";
        e.setEmailTo(para);
        e.setEmailCC("");
        e.setEmailBCC("");
        e.setTitulo("REVISIÓN SOLICITUD VACACIONES ");
        EmailDatosModificacionEmail datosMailS = mailDAO.obtenerDatosEmailRevision(codVac);

mnsg = "<br/><br/>" + datosMailS.getNombre() + " ha "; if (aprobadoEmail == 0) {mnsg += " RECHAZADO ";} else {mnsg += " APROBADO ";}
        mnsg += "el siguiente rango de vacaciones:  <br/><br/>";
        mnsg += "Inicio: " + datosMailS.getFchaini().replace('-', '/') +
        "  Fin: "+ datosMailS.getFchafin().replace('-', '/') + ", con un total de " + datosMailS.getTotDias()+ " días. <br/><br/> ";
        e.setEmailTo(mailDAO.obtenerEmailReceptor(datosMailS.getCodRev())); // CORREO DEL DESTINATARIO (EMPLEADO)
        e.setMensaje(mnsg);

        break;
}
        if (!mailDAO.enviarCorreo(e)) {
        logger.error("formarEmail ERROR: No se ha podido enviar el tipo de mail " + tipoEmail+ " para informar de la soclicitud");
        }
return SUCCESS;
}
```

En la siguiente imágen podemos ver el resultado de una prueba realizada al aprobar una solicitud tras indicar el envío a Castilla.

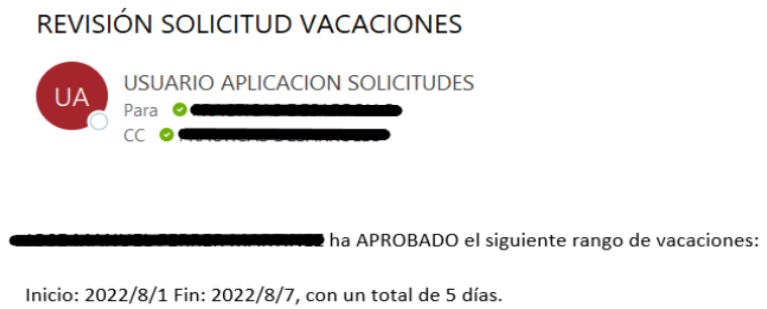


Fig.6.15 *Resultado envio de notificaciones supervisor - empleado*

6.4.4.3 LLamadas externas

Mediante la siguiente llamada, con los parámetros correspondientes, se llama al EndPoint del WEB Service de forma que nos devuelva los datos asociados al trabajador/a que haya accedido a la funcionalidad y así posteriormente transformarlos a formato JSON .

```
@SuppressWarnings("unchecked")
public String misVacaciones() throws ParseException {

	logger.debug("misVacaciones.begin");
	String codigoPersona = session.get("codper").toString();
	String metodo = "GET";
	String ruta = "/WSOrganigrama/m2/revisor/" + codigoPersona;
	String parametros = "";

	LlamadaWS resulWS = llamarConsultaWS(metodo, ruta, parametros);

	if (resulWS.getResultadoCodigo() == 200) {

		Gson gson = new GsonBuilder().create();

		Map<String, Object> mapeoJSONLlamada = gson.fromJson(resulWS.getRespuestaJSON(), HashMap.class);

		Object unArrayList = null;
		if ((Double) mapeoJSONLlamada.get("estado") == 1) { // 1-ok
			for (Object key : mapeoJSONLlamada.keySet()) {
				if (!key.toString().equals("estado")) {
					unArrayList = (Object) mapeoJSONLlamada.get(key);
					break;
				}
			}
		String usRTXT = gson.toJson(unArrayList);
		setVac(gson.fromJson(usRTXT, Vacaciones.class));
		setNivel(vac.getUser().getNivel());
		setDepartamento(vac.getUser().getDepto().trim());
		setCodDepto(vac.getUser().getCod_depto());
		setPuesto(vac.getUser().getPuesto().trim());
		try {
			setCodRevisor(vac.getRevisor().getCodper());
		} catch (Exception e) {
			setCodRevisor(" ");
		}
		setAprobador(isAprobador(getNivel(), getCodDepto()));
		}
	}
	return SUCCESS;
}
```

6.4.4.4 Transformación de datos

Este tipo de operaciones hacen referencia a la descarga de los datos en formato .xls .

Lo que se pretende conseguir con estas operaciones es pasar los datos que se visualizan en el cliente a un documento que se pueda manipular. De forma que el proceso de consolidación de los datos sea más ágil y sencillo, además de abrir un amplio abanico de operaciones a realizar sobre el documento como la inserción de macros [6] para la obtención de estadísticas entre otras posibilidades.

En el siguiente código podemos observar las librerías e instrucciones para la generación de los datos en formato .xls

[6] Acción o conjunto de acciones que se pueden ejecutar N veces en documentos Excel.

```
<!--para generar el excel  -->
<script type="text/javascript" src="<s:url value="/js/excelGen/FileSaver.min.js"/>"></script>
<script type="text/javascript" src="<s:url value="/js/excelGen/generarExcel.js"/>"></script>

/* GENERA EL EXCEL CON FORMATO CASTILLA */
function generarExcelNormalCastilla(data){
      if(data.length != 0){
            var html='';
            $("#resumenVacCastilla").empty();
                  html += '<table id="Resumen Vacaciones Castilla" class="table">';
                        html += '<thead>';
                              html += '<tr>';
                                    html += '<th>NOMBRE</th>';
                                    html += '<th>ID EMPRESA</th>';
                                    html += '<th>NIF</th>';
                                    html += '<th>Fecha  Inicio</th>';
                                    html += '<th>Fecha  Fin</th>';
                              html += '</tr>';
                        html += '</thead>';
                        html += '<tbody>';
            $.each(data.listaVisualizarVacaciones, function(i,x){
                  $.each(x.infoVisualizarVacaciones, function(i,y){
                        var nombreExcell=y.apellidolEx+" "+y.apellido2Ex+", "+y.nombreEx;
                        $.each(y.infoVacaciones, function(i,z){
                              html += '<tr>';
                                    html += '<td>' + nombreExcell + '</td>';
                                    html += '<td>'+y.idEmpresa+'</td>';
                                    html += '<td> '+y.nif+' </td>';
                                    html += '<td> '+moment(z.fechaIni,
'YYYY-MM-DD[T]HH:mm:ss').format('DD/MM/YYYY')+' </td>';
                                    html += '<td> '+moment(z.fechaFin,
'YYYY-MM-DD[T]HH:mm:ss').format('DD/MM/YYYY')+' </td>';
                              html += '</tr>';
                        });
                  });
            });
                  html += '</tbody>';
            html += '</table>';
            $("#resumenVac").append(html);
            $("#resumenVac").addClass("table-responsive");
```

6.4.4.5 Navegación entre páginas

La navegación entre las páginas se realiza de la siguiente forma, véase el código JQuery de la página administradorCalendarios.jsp para acceder a la página administradorCalendariosEditar.jsp además del paso e instancia de parámetros entre ellas y poder modificar el calendario seleccionado.

```
/* Redireccionamiento a Modificar el Calendario  */
      $('#btnModificarCalendario').click(function(e) {
            if (!$("#btnModificarCalendario").hasClass("disabled")) {
                  $.post('/SIUC2-Vacaciones/asociarCCC.action', {
                        codCal : $("#hhidCalendario").val(),
                        responCal : $('#hhresponsable').val(),
                        fechaCrCal : $('#hhfechaCreacion').val(),
                        fechaModCal : $('#hhfechaUlmMod').val(),
                        nameCal : $('#hhnombre').val(),
                        diasCal : $('#hhdias').val(),
                        yearSelected : $('#selAnno').val(),
                  }).done(function (res){
                        $('#contenido').hide();
                        $('#contenido2').show();
                        $('#contenido2').html(res);
                  });
            }
      });
```

En el código podemos observar la llamada(***/SIUC2-Vacaciones/asociarCCC.action***) mediante la cual se llevará a cabo la acción de redireccionamiento una vez pulsado el botón de modificar de esta primera vista además del paso de parámetros.

Tras ejecutar la llamada, en el archivo de configuración **struts.xml** se indicará la acción que se ha de llevar a cabo además de invocar al método de la lógica de negocio correspondiente para poder instanciar los parámetros.

```
<!-- Administradores de calendario modificación -->
<action class="com.uvesco.web.vista.MantenimeintoDispatcher"
    name="asociarCCC" method="asociarCCC">
    <result>/modVacaciones/administradorCalendariosEditar.jsp</result>
</action>
```

Fig.6.16 *Acción struts.xml*

Finalmente una vez identificada la acción en el fichero, se ha implementado en la clase indicada el método asociado a la acción además de instanciar aquellos datos enviados como parámetros en el código inicial .

```
/**
* Redirecciona a la página de modificación del calendario seleccionado
* @return Un string indicando si se ha podido llevare a cabo la acción
*/
      public String asociarCCC() {
            logger.debug("asociarCCC.begin");
            calendario.setFechaCreacion(fechaCrCal);
            calendario.setNombre(nameCal.trim());
            calendario.setResponsable(responCal);
            calendario.setFechaUlmMod(fechaModCal);
            setYearSelected(yearSelected);
            return SUCCESS;

      }
```

Se puede observar, en ambos fragmentos de código, cómo se envían e instancian los datos para posteriormente poder utilizarlos en la página de redireccionamiento.

```
<sj:div cssClass="col-lg-12 col-md-12 col-sm-12 col-xs-12 form-group">
    <label class="col-lg-4 col-md-4 col-sm-12 col-xs-12 control-label">Fecha Creación: </label>
    <sj:div cssClass="col-lg-8 col-md-8 col-sm-12 col-xs-12"><span class="texto-azul" id="creadoPor">${calendario.fechaCreacion}</span>
    </sj:div>
</sj:div>
```

Fig.6.17 *Implementación obtención de datos entre páginas*

En la imagen podemos observar cómo se obtiene el dato de la fecha de creación del calendario en la página redireccionada.

Una de las ventajas de Struts 2 es que busca simplificar las clases de los objetos. Por lo que los actions del fichero struts.xml se convierten en POJO [7]s. Los POJOs son clases que cuentan con getters y setters para poder recibir los valores desde las páginas y posteriormente poder instanciarlos. De esta forma podemos obtener esos mismos valores en las demás páginas de la aplicación desarrollada.[Arquitectura Java, 2021]

[7] Plain Old Java Object

6.5 Integración de la herramienta

En este último apartado respecto al capítulo de implementación se mostrará el proceso llevado a cabo para la integración de la aplicación en los diferentes servidores de la empresa.

Antes de comenzar con el proceso de integración, los proyectos que han sido modificados o creados para el desarrollo de esta nueva funcionalidad se han subido a Bitbucket a través de SmartGit llevando a cabo el control y gestión de versiones.

A continuación, se muestra el proceso llevado a cabo para la integración de la aplicación en los servidores.

Inicialmente se realiza la selección de aquellos proyectos que serán exportados en formato WAR [8] y se indicará la carpeta en la que estos serán almacenados.
Los archivos WAR son como los archivos JAR [9] usados por Java, pero en este caso se utilizan para distribuir aplicaciones web que pueden contener páginas JSP, Servlets, clases Java, páginas web estáticas, y cualquier otro recurso utilizado por la aplicación web.
[Krasis Consulting, 2021]

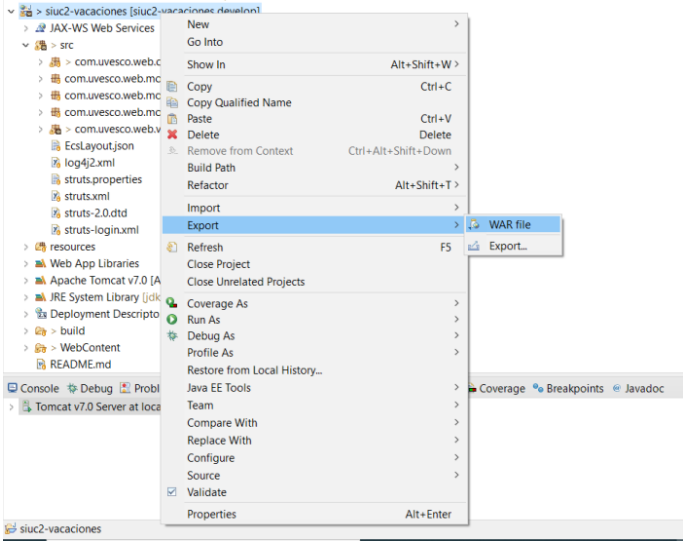


Fig.6.18 *Exportar proyecto a archivo WAR*

Una vez exportado el proyecto, se abrirá el gestor de aplicaciones web de Tomcat en el navegador indicando la dirección y puerto de entrada al servidor en la URL. En la siguiente imagen se puede observar un ejemplo de la dirección de acceso al gestor de aplicaciones de un servidor.

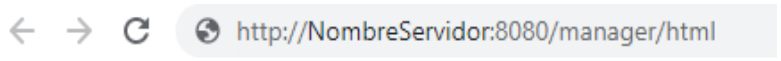


Fig.6.19 *URL de acceso al gestor de aplicaciones web del Tomcat*

[8] Web application ARchive
[9] Java ARchive

Previamente al acceso al gestor de aplicaciones se solicitará el usuario y contraseña. Una vez insertados los datos correctamente se visualizará la siguiente página del gestor.

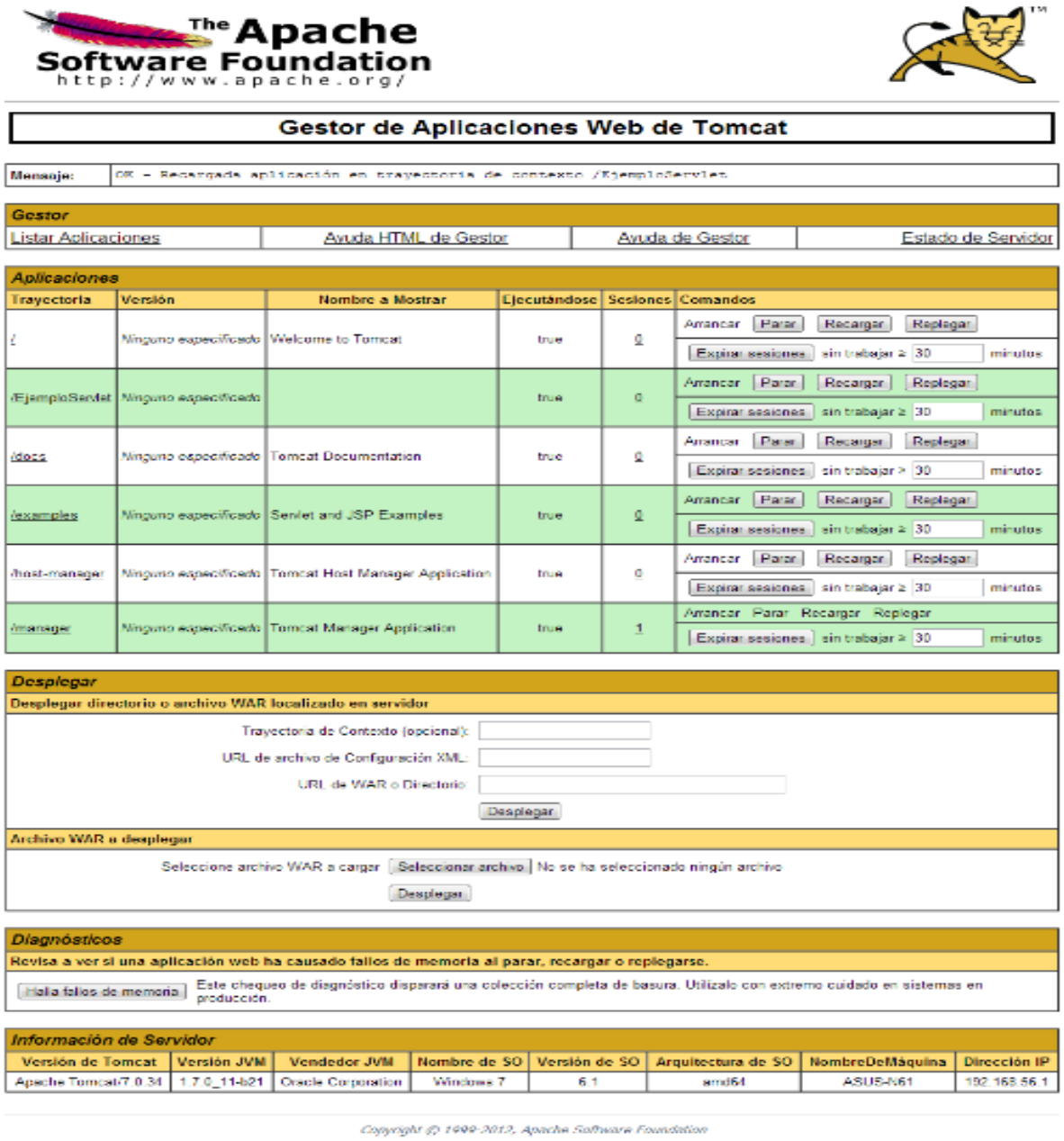


Fig.6.20 *Página del gestor de aplicaciones web del Tomcat*

Desde ella se podrán gestionar los archivos WARs replegando y desplegándolos.
Para ello se deberá seleccionar el archivo almacenado en la carpeta indicada previamente y finalmente desplegarlo.
Una vez el gestor indique que la subida del archivo se ha realizado con éxito se podrá acceder a la funcionalidad.

Este proceso se llevará a cabo tantas veces como servidores existan, de forma que todos ellos cuenten con los mismos archivos y versiones exactas de cada WAR.

7

Pruebas

En este capítulo se observan las pruebas más relevantes realizadas sobre el desarrollo del proyecto. Estas se han llevado a cabo en la parte del cliente y servidor con el objetivo de asegurar el correcto funcionamiento y cumplimiento de los requisitos funcionales previamente analizados.

7.1 Front End

Este apartado recoge las pruebas realizadas sobre el cliente, es decir, aquellas pruebas necesarias para el aseguramiento de los requisitos funcionales de interfaz gráfica.

7.1.1 Casos de prueba cliente

Para ello se ha realizado y comprobado, tanto en las vistas de usuario como administrador, la siguiente batería de pruebas sobre el cliente :

- **CPC-1** Acceso a la funcionalidad desde diferentes roles y perfiles, de forma que la aplicación muestre mayor o menor funcionalidad.
- **CPC-2** Visualizado de los diferentes colores de los días en el calendario referenciando el estado en el que se encuentran las solicitudes e indicando aquellos días no laborables y festivos.
- **CPC-3** Desde la vista del calendario y ver detalles de las vacaciones se han llevado a cabo las siguientes operaciones: creación, modificación, eliminado y envío de solicitudes observando el actualizado de la leyenda en base a las operaciones previamente mencionadas de forma que indique correctamente el número de días totales de vacaciones correspondientes al trabajador/a.
- **CPC-4** Se han realizado creaciones y modificaciones de solicitudes en las vistas del calendario y ver detalles de forma que en la ventana emergente de la selección de rangos aparezca el número total de días seleccionado o bien alguno de los siguientes motivos por los que no se permite llevar a cabo alguna de las dos operaciones:
    - Solapes entre fechas
    - Selección de días no laborales
    - Superación del límite de días disponibles
    - Selección correcta
    - Selección de fechas pasadas
- **CPC-5** Acceso a la funcionalidad Visualizar Departamento desde diferentes usuarios y puestos asegurando el correcto funcionamiento de la generación de tablas dinámicas en la vista .
- **CPC-6** Redireccionamiento desde, las tablas dinámicas, la vista Visualizar Departamento a Revisar Vacaciones .
- **CPC-7** En base a la ejecución de las operaciones de aprobación y rechazo de las solicitudes se ha comprobado el correcto actualizado del indicador de las solicitudes a procesar.
- **CPC-8** Búsquedas de texto en las diferentes vistas de la aplicación y componentes.
- **CPC-9** Agrupación y desagrupación de los datos, mostrando correctamente la información.

7.2 BackEnd

A continuación se mostrarán los casos de prueba realizados en las diferentes funcionalidades desarrolladas. En este apartado encontraremos los casos de prueba relacionados en la parte del servidor, es decir, las pruebas vinculadas con los requisitos funcionales de aplicación.

7.2.1 Casos de prueba servidor

Los siguientes casos de prueba se han dividido en base a las diferentes funcionalidades desarrolladas.

7.2.1.1 Funcionalidad Mis Vacaciones

Vista Calendario

CPS-1 A pesar de que en el apartado anterior se haya realizado el siguiente caso de prueba **CPC-2** en el apartado del servidor nos enfocaremos en la correcta inserción en la base de datos además del comportamiento que adopta ante las diferentes restricciones de selección.

Vista Ver Detalles

- **CPS-2** Envío de solicitud para revisión
- **CPS-3** Eliminado de solicitudes en los estados en los que se ofrezca la posibilidad
- **CPS-4** Modificación de las solicitudes (Esta funcionalidad se encuentra relacionada con las pruebas realizadas en la vista del calendario)

Vista Visualizar Departamento

- **CPS-5** Filtro Todos
- **CPS-6** Filtro Aprobados

Vista Revisar Vacaciones

- **CPS-7** Aprobación de solicitudes de manera individual
- **CPS-8** Aprobación de solicitudes en conjunto
- **CPS-9** Rechazo de solicitudes

Vista Visualizar Vacaciones

- **CPS-10** Descarga Excel
- **CPS-11** Descarga Excel para Castilla
- **CPS-12** Indicador envio en Castilla
- **CPS-13** Visualizado de los departamentos
- **CPS-14** Envío de notificaciones a través del correo

7.2.1.2 Funcionalidad Administrar Calendarios

Administración calendarios

- **CPS-15** Eliminar Calendario
- **CPS-16** Añadir Calendario

Modificación Calendarios

- **CPS-17** Modificar Calendario
- **CPS-18** Asociar Cuenta Cotización
- **CPS-19** Desasociar Cuenta Cotización
- **CPS-20** Modificar días totales de vacaciones a empleados/as
- **CPS-21** Añadir festivos
- **CPS-22** Eliminar festivos

7.2.1.3 EndPoint

En este apartado se han realizado diversas llamadas al endpoint con diferentes códigos de usuario de forma que devuelva los datos JSON correspondientes a este. Para ello se han realizado las siguientes pruebas.

- **CPS-23** Usuarios de diferentes departamentos
- **CPS-24** Usuarios del mismo departamento pero diferente área.
- **CPS-25** Usuarios sin aprobador.

Todos los casos de prueba realizados en la parte de front-end y back-end se encuentran detallados en el siguiente anexo D.

7.3 Pruebas con usuarios reales

Previamente a la finalización del desarrollo, la herramienta ha sido testeada por algunos usuarios asociados al departamento de sistemas y organización. De esta manera lo que se pretendía era asegurar que la herramienta cumpliera con todos los requisitos inicialmente establecidos. El resultado de estas pruebas ha sido satisfactorio, concluyendo con esta primera fase.
Sin embargo, la herramienta no estará disponible para todos los empleados/as de la oficina hasta finales de año ya que se llevará a cabo el siguiente proceso:

El proceso consta de tres fases diferentes, estas se ejecutarán de forma sucesiva, es decir, hasta que no se hayan validado las pruebas en cada una de las fases no se continuará con las siguientes etapas. De esta forma lo que se pretende es asegurar el correcto funcionamiento a la hora de hacer accesible la funcionalidad para todos los trabajadores/as.

En esta primera fase, la herramienta ha sido probada por ciertos empleados/as del departamento de sistemas y organización. El objetivo principal, además de la detección de errores, es asegurar el cumplimineto de los requisitos establecidos

A continuación, en la segunda fase, la aplicación estará disponible para ciertos usuarios de todos los departamentos que conforman la empresa. Su objetivo es asegurar un correcto funcionamiento de manera controlada previamente a su uso por parte de todos los usuarios.

Finalmente la nueva funcionalidad desarrollada, tras haber pasado por las fases anteriormente descritas, se pondrá a disposición de todos los trabajadores/as de la oficina que dispongan del acceso a la herramienta SIU.

En la siguiente imagen se puede observar el proceso descrito de forma gráfica

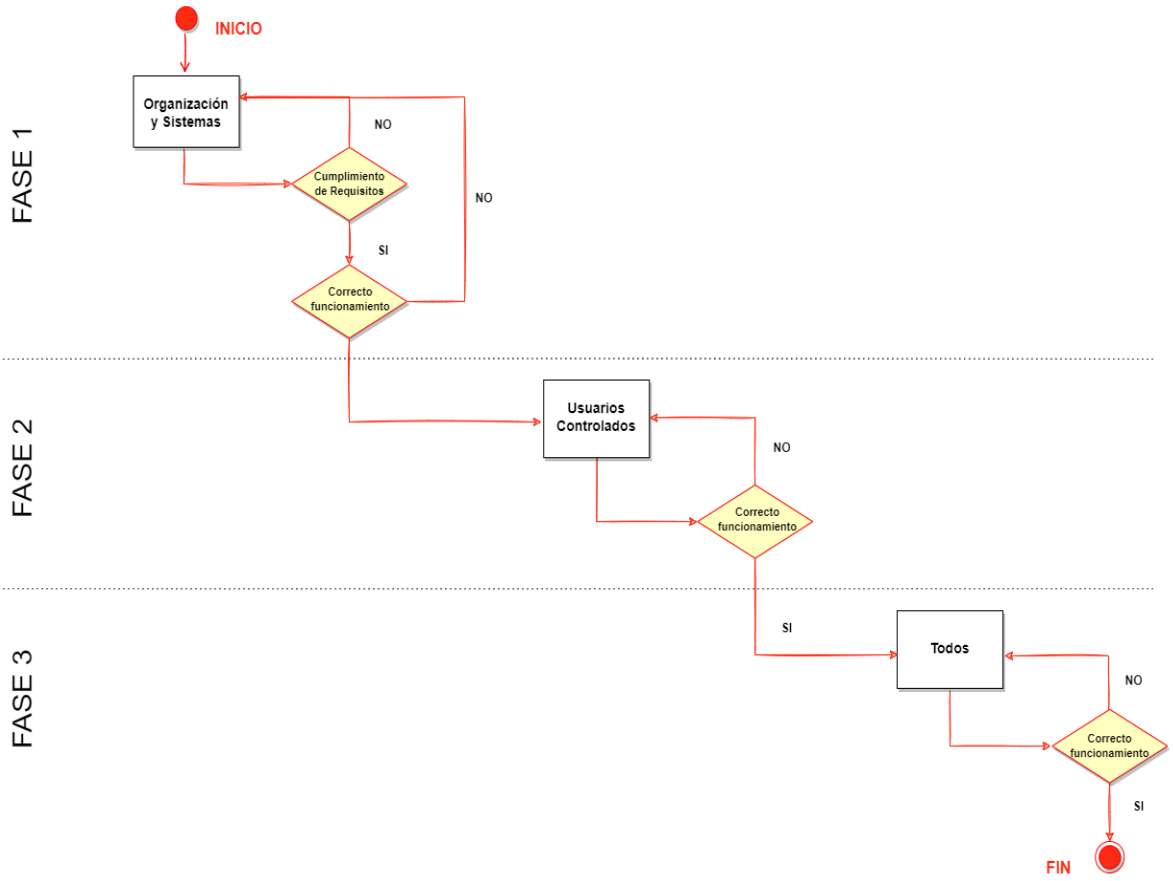


Fig.7.1 *Proceso de pruebas*

Durante el periodo de pruebas se han detectado diferentes errores, la mayoría asociados a las pruebas realizadas en el back-end, concretamente en la capa de acceso a los datos; no obstante, estos han sido solucionados.

Cabe destacar que el proceso ha concluido satisfactoriamente hasta la primera fase. La herramienta desarrollada cumple con cada uno de los requisitos inicialmente establecidos y el feedback recibido por parte de los usuarios, que han podido testear la funcionalidad en esta primera fase, ha sido favorable mostrando su agrado ante el trabajo realizado durante este periodo.

8

Seguimiento y Control

En este capítulo se observará cómo se ha desarrollado el proyecto realmente, mostrando las desviaciones surgidas respecto a la planificación elaborada en una primera instancia. En consecuencia, se observarán también aquellos riesgos e incidencias que han repercutido durante este periodo de tiempo además de observar la comparación respecto al tiempo estimado e invertido realmente.

8.1 Incidencias

En este apartado se detallarán las incidencias surgidas durante el desarrollo del proyecto. Algunas se encuentran relacionadas con los riesgos ya previstos en la planificación, sin embargo, no han podido ser evitados y es por ello que se ha llevado a cabo el plan de mitigación de riesgos descrito en estos.

8.1.1 Coordinación y compaginación con las asignaturas

Cabe destacar esta incidencia, referenciada al primer riesgo R1 detallado en el análisis, dado que ha causado un gran impacto en el desarrollo del proyecto.
El objetivo inicial era concluir con el proyecto en julio con la defensa final, sin embargo, debido a la coordinación de los exámenes de junio y la elaboración de la memoria junto a la defensa los plazos eran demasiado ajustados como para llevar a cabo la revisión y finalización de la etapa de documentación.

8.1.1.1 Contexto

En la figura 8.1 podemos observar las fechas, que hacen referencia a las dos últimas asignaturas a falta de concluir con el grado, en el mes de junio. Además de los plazos límite establecidos para realizar la subida del proyecto a la plataforma y concluir con el proyecto.
Es destacable el día 16 de junio dado que era el examen de extraordinaria de Minería de Datos, es decir, si este no se superaba a diferencia del examen del día 3 el siguiente curso académico tendría que volver a cursar únicamente esa asignatura durante un cuatrimestre entero finalizando en enero con el grado.

Es por ello que durante la semana del 6 al 16 de junio el proyecto se detuvo, dado que el tiempo se invirtió completamente al estudio de esta asignatura y en cierto modo asegurar la superación de esta, logrando finalmente un resultado satisfactorio.

No obstante, la solicitud de la defensa en GAUR de la tercera convocatoria, era el día 20 junio como plazo máximo, concluyendo con la subida del trabajo a la plataforma digital ADDI con fecha límite el día 26 del mismo mes.
Durante ese periodo de tiempo habría que concluir con la finalización del desarrollo de la memoria junto al póster y las posibles correcciones por parte del tutor.

Debido a ello, se tomó la decisión de finalizar el proyecto en la cuarta convocatoria incrementando así los plazos de manera que fuera más viable realizar el proceso de revisión y finalización de la etapa de documentación.
En la imagen se muestran los periodos de tiempo por los que se tomó la decisión final.

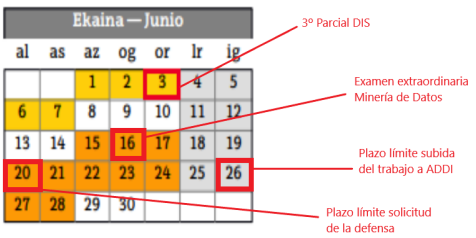


Fig.8.1 *Calendario Junio 2021/2022*

8.1.2 Administración de los calendarios

Una vez finalizada la etapa de adquisición de conocimientos y trás la resolución de las dudas mediante correos, se observó que la aplicación a desarrollar carecía de los datos respecto a los calendarios a asociar a cada empleado/a, es decir, estos datos no se encontraban disponibles en el servidor en el que se iba a implantar la funcionalidad. Por ello se propuso y desarrolló esta segunda funcionalidad en la herramienta de la empresa mediante la cual poder llevar a cabo las diferentes operaciones de administración y gestión de los calendarios.

En la siguiente sección se detallarán las desviaciones respecto a la planificación final elaborada.

8.2 Desviaciones de la planificación

Como se ha indicado en la sección anterior, la idea principal era concluir con el proyecto en la tercera convocatoria. Sin embargo, debido al impacto generado por el riesgo previamente mencionado se tomó la decisión de concluir con el proyecto en la cuarta convocatoria. En el siguiente anexo E.1 se muestra la planificación final elaborada.

Para esta planificación se estimaba concluir el proyecto en 400 horas, es decir, un incremento de 40 horas respecto a la planificación inicial. Esta ha sido la desviación más destacable en el proyecto.
En la siguiente imagen podemos observar la distribución por horas de forma detallada para cada una de las etapas y paquetes de trabajo de esta segunda planificación realizada.

Adquisición de Conocimientos	
PSV	17 h
TH	2 h
IH	1 h
Total	**20**

Diseño de la aplicación	
DPSV	10 h
DWEB	40 h
Total	**50**

Implementación	
IFE	90 h
IBE	110 h
Total	**200**

Pruebas y Validación	
PFE	20 h
PBE	25 h
Total	**45**

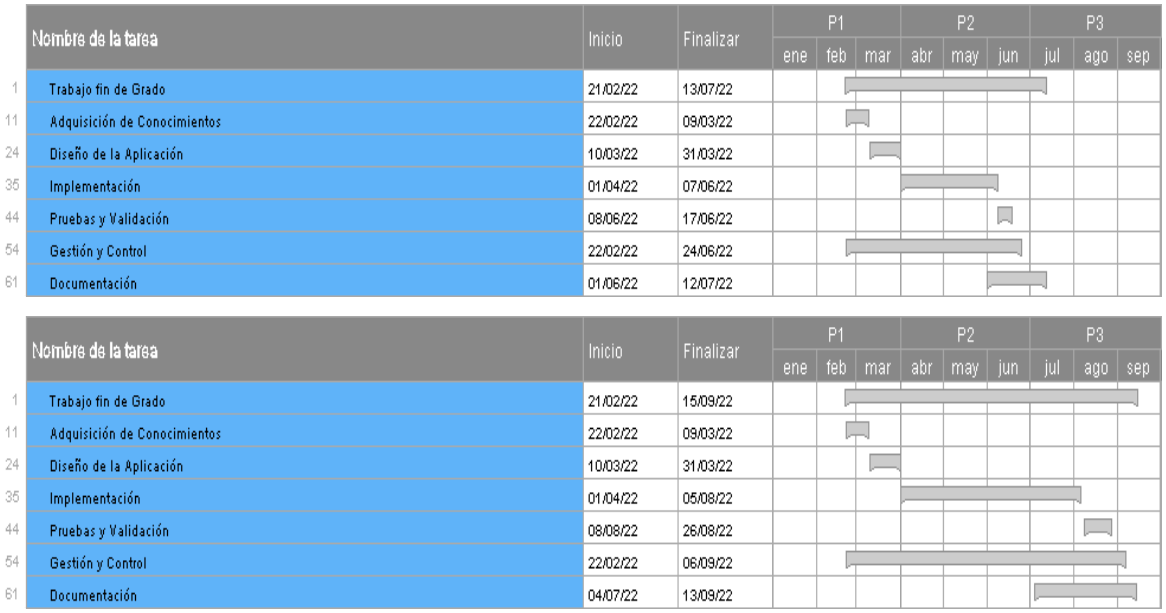
Gestión y Control	
P	15 h
SyC	30 h
Total	**45**

Documentación	
DM	25 h
DNF	5 h
ED	10 h
Total	**40**

Horas Totales del Proyecto	**400**

Tabla 8.1 *Estimación de tiempos por fases y paquetes - 2ª planificación*

En las siguientes imágenes se puede observar el impacto que el riesgo ha generado a través de la comparación de fechas entre las diferentes etapas o fases para cada una de las planificaciones.

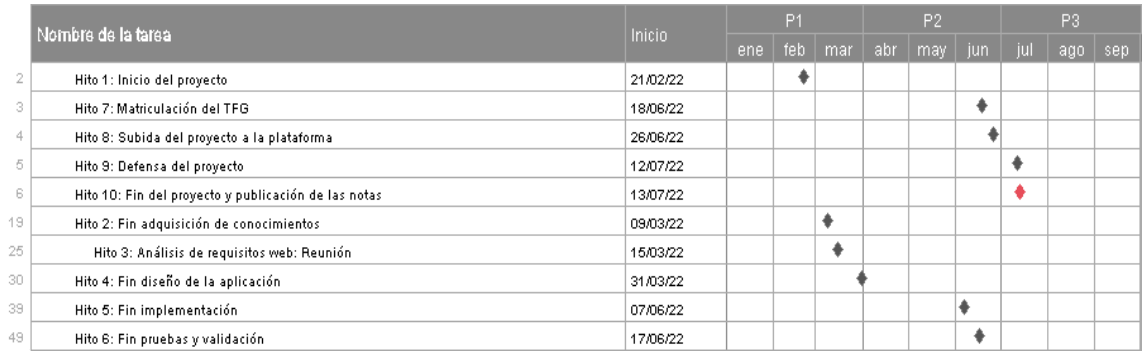


	Nombre de la tarea	Inicio	Finalizar
1	Trabajo fin de Grado	21/02/22	13/07/22
11	Adquisición de Conocimientos	22/02/22	09/03/22
24	Diseño de la Aplicación	10/03/22	31/03/22
35	Implementación	01/04/22	07/06/22
44	Pruebas y Validación	08/06/22	17/06/22
54	Gestión y Control	22/02/22	24/06/22
61	Documentación	01/06/22	12/07/22

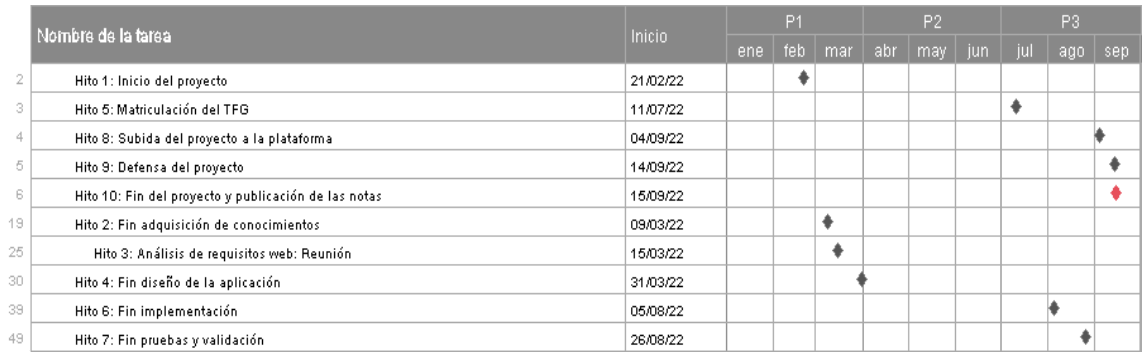
	Nombre de la tarea	Inicio	Finalizar
1	Trabajo fin de Grado	21/02/22	15/09/22
11	Adquisición de Conocimientos	22/02/22	09/03/22
24	Diseño de la Aplicación	10/03/22	31/03/22
35	Implementación	01/04/22	05/08/22
44	Pruebas y Validación	08/08/22	26/08/22
54	Gestión y Control	22/02/22	06/09/22
61	Documentación	04/07/22	13/09/22

Fig.8.2 *Comparación de desviaciones entre fases y fechas de la planificación*

Como se puede observar hay etapas que han sido modificadas debido a la prolongación del convenio. Además, en las siguientes imágenes se muestra cómo algunos hitos del proyecto también han sido modificados a causa de ello.



	Nombre de la tarea	Inicio
2	Hito 1: Inicio del proyecto	21/02/22
3	Hito 7: Matriculación del TFG	18/06/22
4	Hito 8: Subida del proyecto a la plataforma	26/06/22
5	Hito 9: Defensa del proyecto	12/07/22
6	Hito 10: Fin del proyecto y publicación de las notas	13/07/22
19	Hito 2: Fin adquisición de conocimientos	09/03/22
25	Hito 3: Análisis de requisitos web: Reunión	15/03/22
30	Hito 4: Fin diseño de la aplicación	31/03/22
39	Hito 5: Fin implementación	07/06/22
49	Hito 6: Fin pruebas y validación	17/06/22



	Nombre de la tarea	Inicio
2	Hito 1: Inicio del proyecto	21/02/22
3	Hito 5: Matriculación del TFG	11/07/22
4	Hito 8: Subida del proyecto a la plataforma	04/09/22
5	Hito 9: Defensa del proyecto	14/09/22
6	Hito 10: Fin del proyecto y publicación de las notas	15/09/22
19	Hito 2: Fin adquisición de conocimientos	09/03/22
25	Hito 3: Análisis de requisitos web: Reunión	15/03/22
30	Hito 4: Fin diseño de la aplicación	31/03/22
39	Hito 6: Fin implementación	05/08/22
49	Hito 7: Fin pruebas y validación	26/08/22

Fig.8.3 *Comparación de desviaciones en las fechas de los hitos*

8.3 Desviaciones de tiempo

En esta sección se detallarán las desviaciones que han surgido sobre esta última planificación, es decir, al ser modificada la planificación inicial esta segunda se ha tomado como principal del proyecto. Por ello, las desviaciones de tiempos surgidas se mostrarán en base a la segunda planificación pero siempre teniendo en cuenta las desviaciones ya existentes en la inicial dado que hay etapas que no han sido modificadas como previamente hemos podido observar.

8.3.1 Dedicaciones reales y estimadas

En la siguiente tabla podemos observar las horas estimadas tras realizar la planificación y las horas reales invertidas en cada una de las fases y paquetes en los que se ha dividido el proyecto.

	Horas Estimadas	Horas Reales	Desviación (h)
Adquisición de Conocimientos			
PSV	17	20	3
TH	2	4	2
IH	1	1	-
Total	**20**	**25**	**5**
Diseño de la aplicación			
DPSV	10	4	6
DWEB	40	20	20
Total	**50**	**24**	**26**
Implementación			
IFE	90	86	4
IBE	110	157	47
Total	**200**	**243**	**43**
Pruebas y Validación			
PFE	20	12	8
PBE	25	28	3
Total	**45**	**40**	**5**
Gestión y Control			
P	15	15	-
SyC	30	27	3
Total	**45**	**42**	3
Documentación			
DM	25 h	53	28
DNF	5 h	4	1
ED	10 h	14	4
Total	**40**	**71**	31
Horas Totales del Proyecto	**400**	**445**	**45**

Tabla 8.2 *Desviación de las estimaciones*

En la tabla podemos observar cómo en las siguientes etapas se ha invertido un mayor tiempo al esperado: Adquisición de Conocimientos, Implementación y Documentación.

Es destacable las etapas de Documentación e Implementación ya que en ellas ha surgido el mayor desvío de todos, en cierto modo ha sido debido al desconocimiento de las tecnologías como Struts y los diferentes componentes utilizados a la hora de su desarrollo además de una mala previsión y estimación en cuanto a la elaboración de la memoria del proyecto.

No obstante, las siguientes etapas han sufrido una desviación positiva: Diseño de la aplicación, Pruebas & Validación y Gestión & Control. Entre estas etapas se destaca la fase de Diseño de la aplicación y Gestión y control.

En esta primera las desviaciones se deben a una sobreestimación en las tareas de diseño respecto al endPoint del WEB Service y el diseño de las tablas. En la segunda, en cambio, las desviaciones son debidas a que no todo los días se ha realizado un seguimiento y control. Como previamente se ha indicado, el proyecto en las semanas del 6 al 16 de junio se detuvo a causa del examen de extraordinaria, es por ello que surge una desviación "positiva" visto así.

En la siguiente gráfica podemos observar las desviaciones surgidas en las diferentes fases que conforman el proyecto, de forma que se obtenga una visión general en cuanto a los tiempos estimados e invertidos.

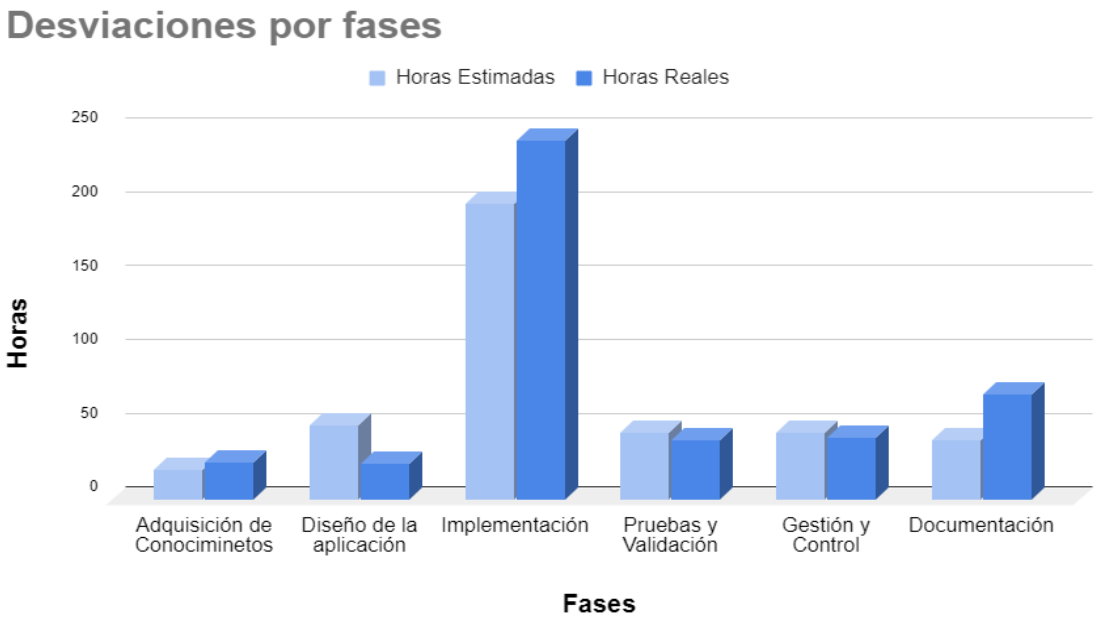


Fig.8.4 *Desviación de tiempos por fases*

Finalmente, en esta última gráfica podemos observar como las horas estimadas se superponen a las reales y viceversa, obteniendo así una visión más detallada de las desviaciones surgidas en cada uno de los paquetes de trabajo.

Fig.8.5 *Desviación de tiempos por paquetes de trabajo*

9

Conclusiones

En este último capítulo se observan las reflexiones realizadas acerca del proyecto además de indicar algunas mejoras y líneas futuras en base al estado actual en el que se encuentra el proyecto tras finalizarlo.

9.1 Conclusiones

Tras finalizar el proyecto a pesar de no haber cumplido con las fechas inicialmente establecidas en cuanto a la fase de documentación, los demás objetivos han sido cumplidos con éxito.
El realizar un seguimiento continuo del proyecto y a la vez mantener una buena comunicación, durante su desarrollo, con el cliente me ha ayudado a hacerme una idea más precisa sobre el funcionamiento de una empresa en el ámbito profesional del desarrollo del software. Además, me ha supuesto una gran ayuda a la hora de tomar las diferentes decisiones descritas en el anterior capítulo, valorando mejor la situación y las posibles alternativas.
Cabe destacar que debido a la prolongación del convenio ha sido posible incrementar el tiempo dedicado a las pruebas, de forma que se han detectado y solucionado errores que en una primera instancia no habían sido identificados.

Finalmente, el haber concluido con un proyecto de ingeniería en el ámbito laboral ha sido un experiencia de un gran aprendizaje y satisfacción personal dado que durante el periodo académico la elaboración de los proyectos ha sido diferente. En este proyecto se ha hecho uso de muchas de las tecnologías utilizadas a lo largo de la carrera además de exponer aquellos conocimientos y prácticas aprendidas.

9.2 Líneas futuras

Una vez concluido el proyecto, cabe destacar que este ha sido enfocado exclusivamente a los empleados/as de la oficina que dispongan del acceso a la herramienta SIU. Sin embargo, la empresa cuenta con muchos más trabajadores/as ya sea en logística, comercios, almacenes etc.
Lo que se propone es escalar esta nueva funcionalidad a todos los trabajadores/as de la empresa de manera que la tarea de selección de las vacaciones llegue a convertirse en un proceso sencillo y rápido en todos los campos de Uvesco.
De esta forma el banco de datos del servidor incrementaría de forma significativa pudiendo así obtener diferentes tipos de estadísticas proporcionando un mayor conocimiento de la empresa y favoreciendo la anticipación a diferentes y posibles situaciones que puedan darse a lo largo del año.

9.3 Propuesta de mejoras

Una de las propuestas realizada ha sido el desarrollo de baterías de pruebas unitarias, haciendo uso de la tecnología JUnit [1] y el framework Mockito [2]. De esta forma lo que se pretende conseguir es una mayor fiabilidad y un mejor mantenimiento sobre los nuevos desarrollos en los diferentes proyectos.
Ambas tecnologías nos permiten sistematizar el proceso de pruebas incluso elaborarlas a pesar de que el desarrollo del proyecto no se haya finalizado. Además la propia librería JUnit nos indicará el porcentaje de cobertura obtenido en base a las pruebas realizadas sobre el proyecto, es decir, mide el grado en el que el código del programa ha sido comprobado. Con esta medida podemos determinar la fiabilidad de los programas, estableciendo diferentes criterios sobre ellos. Sin embargo, este desarrollo no se ha podido llevar a cabo dado que no ha habido tiempo suficiente como para realizar ambas implementaciones de forma paralela.

[1] https://junit.org/junit5/docs/current/user-guide/
[2] https://site.mockito.org/

Otra posible mejora es la refactorización del código y unido a ello, la propuesta de solicitud de Pull Request en SmartGit. Es la acción de validar el código por los demás desarrolladores del equipo antes de unirlo a otra rama. El objetivo es que el código que se vaya a subir al repositorio cumpla con los estándares establecidos por el equipo, de manera que las futuras revisiones y consultas no sean una tarea complicada de realizar.

Esta última propuesta se ha realizado dado que en las empresas es común la incorporación de nuevos trabajadores/as, por lo que la inversión de tiempo a la hora de revisar o consultar código de los demás compañeros puede llegar a ser mayor al deseado.

Bibliografía

[RAE y ASALE, 2021] RAE y ASALE (2021). Organigrama | Diccionario de la lengua española. https://dle.rae.es/organigrama

[EDUCBA, 2022] EDUCBA (2022). Struts in Java. https://www.educba.com/struts-in-java/

[Solis, 2014] Solis, J. (2014). ¿Qué es Bootstrap y cómo funciona en el diseño web?. https://www.arweb.com/blog/%C2%BFque-es-bootstrap-y-como-funciona-en-el-diseno-web/

[OpenWebinars, 2018] Rosa Moncayo, J.M (2018). ¿Qué es REST? Conoce su potencia. https://openwebinars.net/blog/que-es-rest-conoce-su-potencia/

[Wikipedia, 2022] Wikipedia (2022). AS/400. https://es.wikipedia.org/wiki/AS/400

[ComputerWeekely, 2021] Craig, S. Mullins (2021). Db2. https://www.computerweekly.com/es/definicion/Db2

[Programación Conceptos de DDS, 2014] IBM (2014). Rellenar el formulario de DDS. https://www.ibm.com/docs/es/ssw_ibm_i_72/dds/rbafppdf.pdf

[Eclipse Jersey] Eclipse Foundation . Eclipse Jersey is a REST framework that provides a JAX-RS (JSR-370) implementation and more. https://eclipse-ee4j.github.io/jersey/

[HowToDoInJava, 2022] Lokesh, G. (2022). Gson – GsonBuilder Configuration. https://howtodoinjava.com/gson/gson-gsonbuilder-configuration/

[Universidad de Alicante, 2014] Dept. Ciencia de la Computación (2014). Introducción a Struts: El controlador y las acciones. http://www.jtech.ua.es/j2ee/publico/struts-2010-11/sesion01-struts-apuntes.pdf

[Humio, 2021] Humio Ltd. (2021). What is a Log File?. https://www.sumologic.com/glossary/log-file/

[Wikipedia, 2021] Wikipedia (2021). Log (informática). https://es.wikipedia.org/wiki/Log_(inform%C3%A1tica)

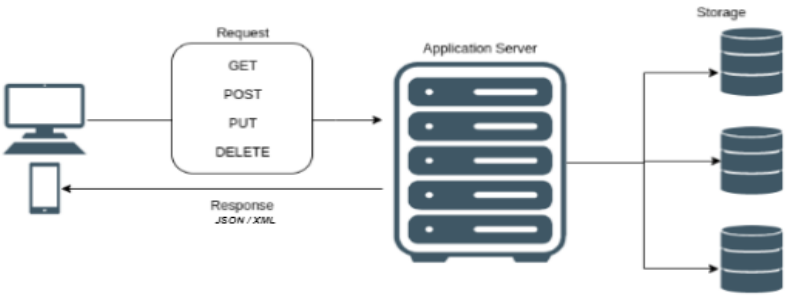
[Arquitectura Java, 2021] Cecilio Álvarez Caules (2021). ¿Qué es un POJO en Java? . https://www.arquitecturajava.com/que-es-un-pojo-en-java/

[Krasis Consulting, 2021] José Manuel Alarcón (2021). En qué se diferencia un archivo .jar de uno .war en Java. https://www.campusmvp.es/recursos/post/en-que-se-diferencia-un-archivo-jar-de-uno-war-en-java.aspx

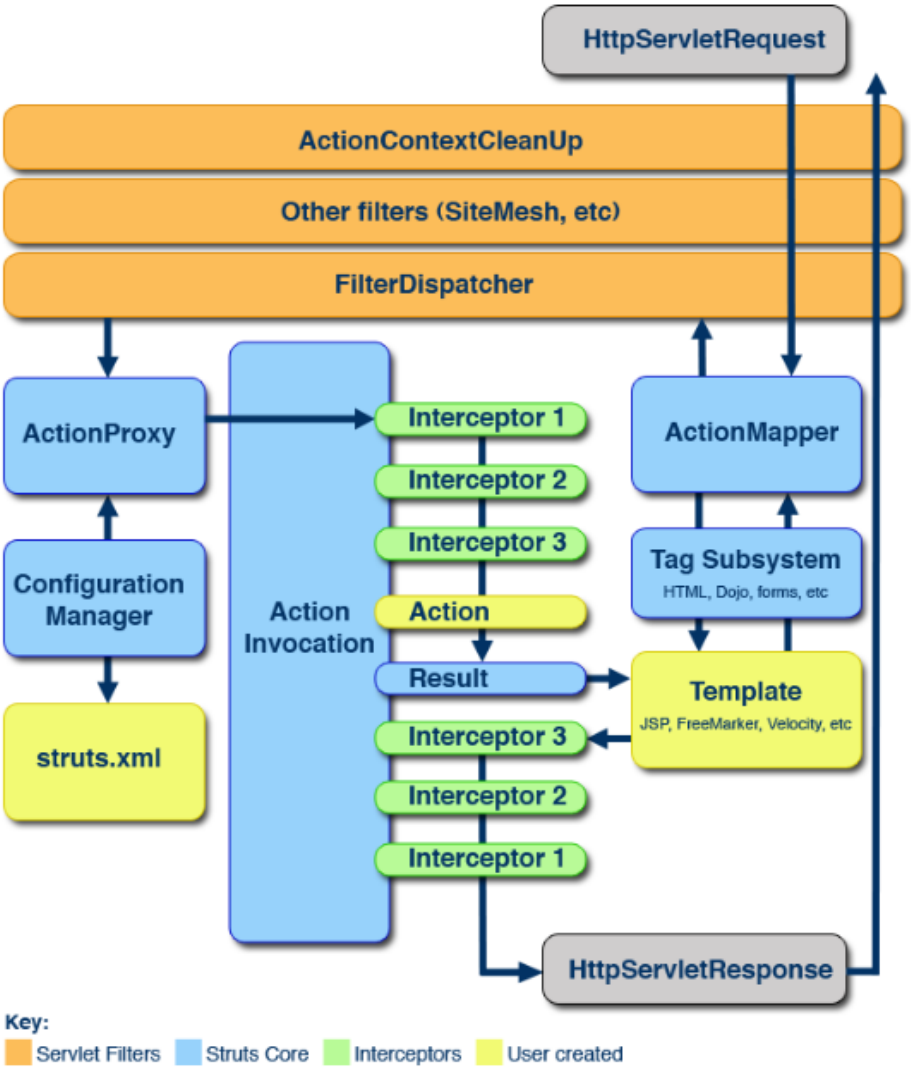
Anexos

A. Modelos de arquitectura

A.1 Arquitectura servicio REST



A.2 Arquitectura lógica STRUTS detallada



B. Código

B.1 Implementación de la llamada al endPoint

```
@GET
@Produces(MediaType.APPLICATION_JSON)
@Path("/m2/revisor/{cdoper}/")
//http://localhost/WSOrganigrama///
//Obtiene el codigo del revisor ,de las vacaciones, de un empleado cocncreto a traves de su codper
/* Salida
    {
    "estado": 1,
    "vac": {
        "user": {
            "nivel": 7,
            "depto": "Sistemas
            "cod_depto": 13,
            "puesto": "ANALISTA PROGRAMADOR/A NUEVAS TECNOLOGIAS
            "dependencia": 48,
            "id_area": "-1 "
        },
        "depto": {
            "id_puesto": "00608   ",
            "nivel": 4,
            "id_depende": "45"
        },
        "revisor": {
            "codper": "00023000034"
        }
    }
}
*/
public Response leerRevisorVacaciones(
        @PathParam("cdoper") String codper
        ) {
    logger.info("leerRevisorVacaciones.begin");

     VacacionesResultado resultado =  new VacacionesResultado();

    resultado.setEstado(ValoresEstado.SERVICIO_NO_DIPONIBLE);

    GsonBuilder gsonBuilder = new GsonBuilder().registerTypeAdapter(LocalDate.class, new LocalDateUtiles.LocalDateAdapter().nullSafe());
    Gson gson = gsonBuilder.create();
    String resultadoJSON = gson.toJson(resultado);

    try {
        OrganigramaControlador controlador = new OrganigramaControlador();
        resultado.setVac(controlador.obtenerDatosVacaciones(codper));
        resultado.setEstado(ValoresEstado.OK);
        resultadoJSON = gson.toJson(resultado);

    } catch (Exception e) {

        logger.error("leerRevisorVacaciones. Resultado estado: 2"); //2 -SERVICIO_NO_DIPONIBLE
        logger.error("leerRevisorVacaciones ERROR: "+e.getMessage());
    }

    logger.info("leerRevisorVacaciones.end");
    return Response.status(200).entity(resultadoJSON).build();
}
```

B.2 Implementación del EndPoint

```
public Vacaciones obtenerDatosVacaciones(String codper) {

    Vacaciones vac = new Vacaciones();

    UsuarioVacaciones usrV = new UsuarioVacaciones();
    usrV = obtenerUsuarioVacaciones(codper);

    if (usrV.getNivel() > 2 && !usrV.getPuesto().trim().equals("DIRECTOR/A RRHH")) {          // NIVELES DIFERENTES DE 1, 2 Y PUESTO DIRECTOR/A RRHH ESTOS
                                                                                             // NO TIENEN APROBADOR
        DepartamentoDependiente dependencia = new DepartamentoDependiente();
        UsuarioRevisor usrR = new UsuarioRevisor();

        if (usrV.getNivel() <= 4 || usrV.getPuesto().trim().equals("SECRETARIO/A DIRECCION GENERAL")
                || usrV.getPuesto().trim().equals("GERENTE ORGANIZACION")) {                 // NIVELES 3,4 Y Puestos:GERENTE ORGANIZACIÓN y SECRETARIO/A DIRECCIÓN GENERAL
            dependencia = obtenerDeptoDependiente(String.valueOf(2));                        // REVISOR = DIRECTOR/A RRHH
            usrR = obtenerRevisor(dependencia.getId_puesto(), usrV.getId_area());

        } else {
            dependencia = obtenerDeptoDependiente(String.valueOf(usrV.getDependencia()));
            if (usrV.getCod_depto() == 1 || usrV.getCod_depto() == 12) {                     // SI DEPARTAMENTO 1(ADMINISTRACION) o 12(RRHH), APROBADORES 6 HACIA ABAJO
                while (dependencia.getNivel() > 6) {                                         // MIENTRAS NO SE OBTENGA UN NIVEL 6 O INFERIOR CONTINUA
                    dependencia = obtenerDeptoDependiente(String.valueOf(dependencia.getId_depende()));
                }
                usrR = obtenerRevisor(dependencia.getId_puesto(), usrV.getId_area());

            } else {                                                                         // SI DEPARTAMENTO DIFERENTE 1 o 12, APROBADORES = 5 HACIA ABAJO
                while (dependencia.getNivel() > 5) {                                         // MIENTRAS NO SE OBTENGA UN NIVEL 5 O INFERIOR CONTINUA
                    dependencia = obtenerDeptoDependiente(String.valueOf(dependencia.getId_depende()));
                }
                usrR = obtenerRevisor(dependencia.getId_puesto(), "-1 ");                    // EXISTEN DEPARTAMENTOS CON AREA <> -1, PERO
                                                                                             // CON LA CONDICION DE QUE SU REVISOR SEA MENOR 5
            }                                                                                // POR LO QUE FUERZO EL AREA A -1 OBLIGANDO A QUE NO ENTRE EN LA SQL POR LA
                                                                                             // CONDICION DE AREA YA QUE NO ENCONTRARIA A SU SUPERVISOR
        }                                                                                    // POR QUE ESTE NO TIENE EL MISMO AREA QUE EL EMPLEADO
        vac.setDepto(dependencia);
        vac.setRevisor(usrR);
    }
    vac.setUser(usrV);
    return vac;
}
```

Código Principal del nuevo EndPoint desarrollado

B.3 Funciones auxiliares del endPoint

```
public UsuarioVacaciones obtenerUsuarioVacaciones(String codper) {

    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs  = null;

    UsuarioVacaciones userV = new UsuarioVacaciones();

    DaoConexion daoC = new DaoConexion();

    try {

        String codigoEmp = "00"+codper.substring(0, 3);
        String codper1 = "0"+codper.substring(3, 8);

        conn = daoC.conectarAS("[illegible]");

        String sql =
                "With TMPC(ID, NIVEL, DEPENDE,  DESDEP1, D_AREAS, DPUEST, IDPUES, " +
                  "ID_PUESTO, ID_EMPRESA,COD_DEP,ID_AREA) AS ( " +
                  "SELECT A.ID, A.NIVEL, A.DEPENDE,  B.DESDEP1, C.D_AREAS, D.DPUEST, " +
                  "D.IDPUES, E.ID_PUESTO, E.ID_EMPRESA, B.SDEPAR_CODDEP1, A.SORGAN_DEP_AREA  " +
                      "FROM AFSORGAN A " +
                      "JOIN AFSDEPAR B ON  B.CODDEP1 = A.ID_DEPART " +
                      "JOIN [illegible]AREAS C ON C.ID_AREAS = A.ID_AREA " +
                      "JOIN [illegible]NLTPUEST D ON D.IDPUES = A.ID_PUESTO " +
                      "JOIN [illegible]PUESTOS E ON  E.ID_PUESTO_TIPO = A.ID_PUESTO " +
                      "AND A.ID_AREA = E.ID_AREA AND E.ID_AREA IS NOT NULL " +
                  "UNION " +
                  "SELECT A.ID, A.NIVEL, A.DEPENDE,  B.DESDEP1, '-1' , D.DPUEST, " +
                  "D.IDPUES, E.ID_PUESTO, E.ID_EMPRESA, B.SDEPAR_CODDEP1, A.SORGAN_DEP_AREA " +
                      "FROM AFSORGAN A  " +
                      "JOIN AFSDEPAR B ON  B.CODDEP1 = A.ID_DEPART " +
                      "JOIN [illegible]NLTPUEST D ON D.IDPUES = A.ID_PUESTO " +
                      "JOIN [illegible]PUESTOS E ON E.ID_PUESTO_TIPO = A.ID_PUESTO " +
                      "AND A.ID_AREA = '-1' AND E.ID_AREA IS NULL " +
                   ") SELECT E.ID, E.NIVEL, E.DEPENDE,  E.DESDEP1, E.D_AREAS, E.DPUEST, " +
                   "E.IDPUES, I.NOMBRE, I.APELLIDO1, I.APELLIDO2, I.NIF, E.COD_DEP, E.ID_AREA " +
                   "FROM TMPC E " +
                "JOIN [illegible]/NIV_ORG F ON " +
                "CONCAT(E.ID_EMPRESA, E.ID_PUESTO) = " +
                "CONCAT(F.ID_EMPRESA, F.ID_PUESTO) " +
                "JOIN [illegible]NLPUEST G ON F.ID_NIVEL=G.NIVEL AND G.PDEFECT = 1 AND (FINIP <=  CURRENT_DATE AND (G.FFINP IS NULL  OR  (G.FFINP IS NOT NULL AND G.FFINP >= CURRENT_DATE ))) " +
                "JOIN [illegible]COD_TRA H ON "  +
                "CONCAT(RIGHT(G.NUMEM, 3), RIGHT(G.NUMPR, 5 )) =  " +
                "CONCAT(RIGHT(H.ID_EMPRESA, 3), RIGHT(H.ID_TRABAJADOR, 5)) " +
                "JOIN [illegible]PERSONAS I ON H.NIF=I.NIF " +
               " WHERE  ? = F.ID_EMPRESA AND H.ID_TRABAJADOR= ?";

        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1,codigoEmp );
        pstmt.setString(2,codper1 );

        rs = pstmt.executeQuery();

        while (rs.next()){

            userV.setNivel(rs.getInt(2));
            userV.setDependencia(rs.getInt(3));
            userV.setDepto(rs.getString(4));
            userV.setPuesto(rs.getString(6));
            userV.setCod_depto(rs.getInt(12));
            userV.setId_area(rs.getString(13));

        }

        //cerramos la conexion
        if (rs != null) try { rs.close(); } catch (SQLException logOrIgnore) {}
        if (pstmt != null) try { pstmt.close(); } catch (SQLException logOrIgnore) {}
        if (conn != null) try { conn.close(); } catch (SQLException logOrIgnore) {}

        }
    catch (SQLException e)
    {
            Logger.error(e.getMessage());
    }
    finally
    {
        if (rs != null) try { rs.close(); } catch (SQLException logOrIgnore) {}
        if (pstmt != null) try { pstmt.close(); } catch (SQLException logOrIgnore) {}
        if (conn != null) try { conn.close(); } catch (SQLException logOrIgnore) {}
    }

    return userV;
}
```

Función obtenerUsuarioVacaciones

```
public DepartamentoDependiente obtenerDeptoDependiente(String idDepende) {

    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs  = null;

    DepartamentoDependiente depto = new DepartamentoDependiente();
    DaoConexion daoC = new DaoConexion();

    try {
        conn = daoC.conectarAS("[illegible]");

        String sql = "SELECT SORGAN_ID_PUESTO, SORGAN_NIVEL , SORGAN_DEPENDE " +
                "FROM AFSORGAN " +
                "WHERE SORGAN_ID = ? ";

        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1,idDepende);
        rs = pstmt.executeQuery();

        while (rs.next()){

            depto.setId_puesto(rs.getString(1));
            depto.setNivel(rs.getInt(2));
            depto.setId_depende(rs.getString(3));
        }
        //cerramos la conexion
        if (rs != null) try { rs.close(); } catch (SQLException logOrIgnore) {}
        if (pstmt != null) try { pstmt.close(); } catch (SQLException logOrIgnore) {}
        if (conn != null) try { conn.close(); } catch (SQLException logOrIgnore) {}
        }
    catch (SQLException e)
    {
            Logger.error(e.getMessage());
    }
    finally
    {
        if (rs != null) try { rs.close(); } catch (SQLException logOrIgnore) {}
        if (pstmt != null) try { pstmt.close(); } catch (SQLException logOrIgnore) {}
        if (conn != null) try { conn.close(); } catch (SQLException logOrIgnore) {}
    }
    return depto;
}
```

Función obtenerDeptoDependiente

```
public UsuarioRevisor obtenerRevisor(String idPuesto, String id_area) {

    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs  = null;

    UsuarioRevisor userR = new UsuarioRevisor();
    DaoConexion daoC = new DaoConexion();

    try {
        conn = daoC.conectarAS("[illegible]");

        String sql = "SELECT DISTINCT RIGHT(B.NUMEM, 3), RIGHT(B.NUMPR, 5 )  "+
                "FROM [illegible]/NIV_ORG A "+
                "JOIN [illegible]/NLPUEST B ON A.ID_NIVEL=B.NIVEL AND B.PDEFECT = 1 AND ( B.FFINP IS NULL OR  B.FFINP > CURRENT_DATE) "+
                "JOIN [illegible]/COD_TRA C ON "+
                "CONCAT(RIGHT(B.NUMEM, 3), RIGHT(B.NUMPR, 5 )) =  "+
                "CONCAT(RIGHT(C.ID_EMPRESA, 3), RIGHT(C.ID_TRABAJADOR, 5)) "+
                "JOIN[illegible] PERSONAS D ON D.NIF=C.NIF "+
                "JOIN[illegible] PUESTOS E ON A.ID_PUESTO=E.ID_PUESTO AND A.ID_EMPRESA=E.ID_EMPRESA "+
                "WHERE E.ID_PUESTO_TIPO =  ? ";

        // EN CASO DE QUE CUENTE CON AREA (!-1) SE AÑADE LA SIGUIENTE CONDICION A LAS SQL Y ASI OBTENER EL REVISOR DE DEPARTAMENTOS 1(ADMINISTRACION Y CONTROL DE GESTION) Y 12(RRHH)
        if(!id_area.equals("-1 ")) { // EL VALOR ID_ AREA EN LA BASE DE DATOS ES UNA CHAR DE 3 -> "-1 " (UTILIZAR TRIM PARA SOLUCIONAR LA CONDICION)
            sql += " AND (E.ID_AREA IS NULL OR E.ID_AREA='-1 ' OR E.ID_AREA= ? )";
            pstmt = conn.prepareStatement(sql);
            pstmt.setString(1,idPuesto);
            pstmt.setString(2,id_area);
        }else {
            pstmt = conn.prepareStatement(sql);
            pstmt.setString(1,idPuesto);
        }
        rs = pstmt.executeQuery();
        while (rs.next()){
            userR.setCodper(rs.getString(1)+rs.getString(2));
        }
        //cerramos la conexion
        if (rs != null) try { rs.close(); } catch (SQLException logOrIgnore) {}
        if (pstmt != null) try { pstmt.close(); } catch (SQLException logOrIgnore) {}
        if (conn != null) try { conn.close(); } catch (SQLException logOrIgnore) {}
        }
    catch (SQLException e)
    {
            Logger.error(e.getMessage());
    }
    finally
    {
        if (rs != null) try { rs.close(); } catch (SQLException logOrIgnore) {}
        if (pstmt != null) try { pstmt.close(); } catch (SQLException logOrIgnore) {}
        if (conn != null) try { conn.close(); } catch (SQLException logOrIgnore) {}
    }
    return userR;
}
```

Función obtenerRevisor

B.4 DDL Tablas AS/400

```
CREATE TABLE [illegible].AFSCALVSH (
    SCALVSH_CODVACH NUMERIC(11,0) NOT NULL,
    SCALVSH_CODVAC1 NUMERIC(11,0) NOT NULL,
    SCALVSH_CODPER1 CHAR(8) NOT NULL,
    SCALVSH_ANYINI1 NUMERIC(4,0) NOT NULL,
    SCALVSH_MESINI1 NUMERIC(2,0) NOT NULL,
    SCALVSH_DIAINI1 NUMERIC(2,0) NOT NULL,
    SCALVSH_ANYFIN1 NUMERIC(4,0) NOT NULL,
    SCALVSH_MESFIN1 NUMERIC(2,0) NOT NULL,
    SCALVSH_DIAFIN1 NUMERIC(2,0) NOT NULL,
    SCALVSH_ANYULM1 NUMERIC(4,0) NOT NULL,
    SCALVSH_MESULM1 NUMERIC(2,0) NOT NULL,
    SCALVSH_DIAULM1 NUMERIC(2,0) NOT NULL,
    SCALVSH_CODPERA CHAR(8) NOT NULL,
    SCALVSH_CODCAL1 NUMERIC(11,0) NOT NULL,
    SCALVSH_NUMCCC1 CHAR(11) NOT NULL
);
```

DDL Tabla AFSCALV

```
CREATE TABLE [illegible].AFSCALVS (
    SCALVS_CODVAC1 NUMERIC(11,0) NOT NULL,
    SCALVS_CODPER1 CHAR(8) NOT NULL,
    SCALVS_ANYINI1 NUMERIC(4,0) NOT NULL,
    SCALVS_MESINI1 NUMERIC(2,0) NOT NULL,
    SCALVS_DIAINI1 NUMERIC(2,0) NOT NULL,
    SCALVS_ANYFIN1 NUMERIC(4,0) NOT NULL,
    SCALVS_MESFIN1 NUMERIC(2,0) NOT NULL,
    SCALVS_DIAFIN1 NUMERIC(2,0) NOT NULL,
    SCALVS_CODEST1 NUMERIC(2,0) NOT NULL,
    SCALVS_TOTDIA1 NUMERIC(2,0) NOT NULL,
    SCALVS_ANYRES1 NUMERIC(4,0) NOT NULL,
    SCALVS_MESRES1 NUMERIC(2,0) NOT NULL,
    SCALVS_DIARES1 NUMERIC(2,0) NOT NULL,
    SCALVS_ANYCRC1 NUMERIC(4,0) NOT NULL,
    SCALVS_MESCRC1 NUMERIC(2,0) NOT NULL,
    SCALVS_DIACRC1 NUMERIC(2,0) NOT NULL,
    SCALVS_ANYULM1 NUMERIC(4,0) NOT NULL,
    SCALVS_MESULM1 NUMERIC(2,0) NOT NULL,
    SCALVS_DIAULM1 NUMERIC(2,0) NOT NULL,
    SCALVS_MOTIVO1 CHAR(200) NOT NULL,
    SCALVS_CODPERA CHAR(8) NOT NULL,
    SCALVS_CODCAL1 NUMERIC(11,0) NOT NULL,
    SCALVS_NUMCCC1 CHAR(11) NOT NULL
);
CREATE UNIQUE INDEX AFSCALVS ON [illegible].AFSCALVS (SCALVS_CODVAC1);
```

DDL Tabla AFSCALVS

```
CREATE TABLE [illegible]AFSCALDF (
    SCALDF_CODCAL1 NUMERIC(11,0) NOT NULL,
    SCALDF_ANYFST1 NUMERIC(4,0) NOT NULL,
    SCALDF_MESFST1 NUMERIC(2,0) NOT NULL,
    SCALDF_DIAFST1 NUMERIC(2,0) NOT NULL
);
CREATE UNIQUE INDEX AFSCALDF ON [illegible].AFSCALDF (SCALDF_CODCAL1,SCALDF_ANYFST1,SCALDF_MESFST1,SCALDF_DIAFST1);
```

DDL Tabla AFSCALDF

```
CREATE TABLE [illegible].AFSCALCCC (
    SCALCCC_CODCAL1 NUMERIC(11,0) NOT NULL,
    SCALCCC_NUMCCC1 CHAR(11) NOT NULL
);
CREATE UNIQUE INDEX AFSCALCCC ON [illegible].AFSCALCCC (SCALCCC_CODCAL1,SCALCCC_NUMCCC1);
```

DDL Tabla AFSCALCCC

```
CREATE TABLE [illegible]AFSCALPER (
    SCALCCC_CODCAL1 NUMERIC(11,0) NOT NULL,
    SCALPER_CODPER1 CHAR(8) NOT NULL,
    SCALPER_DIASVCAL NUMERIC(2,0) NOT NULL
);
CREATE UNIQUE INDEX AFSCALPER ON [illegible].AFSCALPER (SCALCCC_CODCAL1,SCALPER_CODPER1);
```

DDL Tabla AFSCALPER

```
CREATE TABLE [illegible].AFSCALVSH (
    SCALVSH_CODVACH NUMERIC(11,0) NOT NULL,
    SCALVSH_CODVAC1 NUMERIC(11,0) NOT NULL,
    SCALVSH_CODPER1 CHAR(8) NOT NULL,
    SCALVSH_ANYINI1 NUMERIC(4,0) NOT NULL,
    SCALVSH_MESINI1 NUMERIC(2,0) NOT NULL,
    SCALVSH_DIAINI1 NUMERIC(2,0) NOT NULL,
    SCALVSH_ANYFIN1 NUMERIC(4,0) NOT NULL,
    SCALVSH_MESFIN1 NUMERIC(2,0) NOT NULL,
    SCALVSH_DIAFIN1 NUMERIC(2,0) NOT NULL,
    SCALVSH_ANYULM1 NUMERIC(4,0) NOT NULL,
    SCALVSH_MESULM1 NUMERIC(2,0) NOT NULL,
    SCALVSH_DIAULM1 NUMERIC(2,0) NOT NULL,
    SCALVSH_CODPERA CHAR(8) NOT NULL,
    SCALVSH_CODCAL1 NUMERIC(11,0) NOT NULL,
    SCALVSH_NUMCCC1 CHAR(11) NOT NULL
);
```

DDL Tabla AFSCALVSH

B.5 Scripts tablas AS/400

```
A                                       UNIQUE
A          R RAFSCALV                   TEXT('CALENDARIO VACACIONES')
A            CODCAL1       11S         ALIAS(SCALV_CODCAL1)
A                                       COLHDG('CODIGO CALENDARIO')
A            NOMCALV      100A         ALIAS(SCALV_NOMCALV)
A                                       COLHDG('NOMBRE CALENDARIO')
A            ANYCAL1        4S         ALIAS(SCALV_ANYCAL1)
A                                       COLHDG('AÑO CALENDARIO')
A            ANYCRC1        4S         ALIAS(SCALV_ANYCRC1)
A                                       COLHDG('AÑO CREACION')
A            MESCRC1        2S         ALIAS(SCALV_MESCRC1)
A                                       COLHDG('MES CREACION')
A            DIACRC1        2S         ALIAS(SCALV_DIACRC1)
A                                       COLHDG('DIA CREACION')
A            ANYULM1        4S         ALIAS(SCALV_ANYULM1)
A                                       COLHDG('AÑO ULTIMA MODIF')
A            MESULM1        2S         ALIAS(SCALV_MESULM1)
A                                       COLHDG('MES ULTIMA MODIF')
A            DIAULM1        2S         ALIAS(SCALV_DIAULM1)
A                                       COLHDG('DIA ULTIMA MODIF')
A            CODPER1        8A         ALIAS(SCALV_CODPER1)
A                                       COLHDG('CODIGO EMPLEADO')
A            DIASVCAL       2S         ALIAS(SCALV_DIASVCAL)
A                                       COLHDG('CALEND.TOTAL DIAS')
A            INDCAS1        1S         ALIAS(SCALV_INDCAS1)
A                                       COLHDG('INDICATIVO CASTILLA')
A*
A          K CODCAL1
```

Script creación tabla AFSCALV

```
A                                       UNIQUE
A          R RAFSCALDF                  TEXT('CALENDARIO DIAS FESTIVOS')
A            CODCAL1       11S         ALIAS(SCALDF_CODCAL1)
A                                       COLHDG('CODIGO CALENDARIO')
A            ANYFST1        4S         ALIAS(SCALDF_ANYFST1)
A                                       COLHDG('AÑO FESTIVO')
A            MESFST1        2S         ALIAS(SCALDF_MESFST1)
A                                       COLHDG('MES FESTIVO')
A            DIAFST1        2S         ALIAS(SCALDF_DIAFST1)
A                                       COLHDG('DIA FESTIVO')
A*
A          K CODCAL1
A          K ANYFST1
A          K MESFST1
A          K DIAFST1
```

Script creación tabla AFSCALDF

```
A                                       UNIQUE
A          R RAFSCALCCC                 TEXT('CALENDARIO CCC')
A            CODCAL1       11S         ALIAS(SCALCCC_CODCAL1)
A                                       COLHDG('CODIGO CALENDARIO')
A            NUMCCC1       11A         ALIAS(SCALCCC_NUMCCC1)
A                                       COLHDG('CUENTA COTIZACION')
A*
A          K CODCAL1
A          K NUMCCC1
```

Script creación tabla AFSCALCCC

```
A                                       UNIQUE
A          R RAFSCALPER                 TEXT('CALENDARIO EXCEP. PERSONAS')
A            CODCAL1       11S         ALIAS(SCALCCC_CODCAL1)
A                                       COLHDG('CODIGO CALENDARIO')
A            CODPER1        8A         ALIAS(SCALPER_CODPER1)
A                                       COLHDG('CODIGO PERSONA')
A            DIASVCAL       2S         ALIAS(SCALPER_DIASVCAL)
A                                       COLHDG('CALEND.TOT.DIAS.PER')
A*
A          K CODCAL1
A          K CODPER1
```

Script creación tabla AFSCALPER

```
A                                       UNIQUE
A           R RAFSCALVS                 TEXT('CALEN. VAC. SOLICITADAS')
A             CODVAC1       11S         ALIAS(SCALVS_CODVAC1)
A                                       COLHDG('COD.VAC.SOLICITADAS')
A             CODPER1        8A         ALIAS(SCALVS_CODPER1)
A                                       COLHDG('CODIGO PERSONA')
A             ANYINI1        4S         ALIAS(SCALVS_ANYINI1)
A                                       COLHDG('AÑO INICIO')
A             MESINI1        2S         ALIAS(SCALVS_MESINI1)
A                                       COLHDG('MES INICIO')
A             DIAINI1        2S         ALIAS(SCALVS_DIAINI1)
A                                       COLHDG('DIA INICIO ')
A             ANYFIN1        4S         ALIAS(SCALVS_ANYFIN1)
A                                       COLHDG('AÑO FIN')
A             MESFIN1        2S         ALIAS(SCALVS_MESFIN1)
A                                       COLHDG('MES FIN')
A             DIAFIN1        2S         ALIAS(SCALVS_DIAFIN1)
A                                       COLHDG('DIA FIN')
A             CODEST1        2S         ALIAS(SCALVS_CODEST1)
A                                       COLHDG('ESTADO')
A             TOTDIA1        2S         ALIAS(SCALVS_TOTDIA1)
A                                       COLHDG('TOTAL DIAS')
A             ANYRES1        4S         ALIAS(SCALVS_ANYRES1)
A                                       COLHDG('AÑO RESOLUCION')
A             MESRES1        2S         ALIAS(SCALVS_MESRES1)
A                                       COLHDG('MES RESOLUCION')
A             DIARES1        2S         ALIAS(SCALVS_DIARES1)
A                                       COLHDG('DIA RESOLUCION')
A             ANYCRC1        4S         ALIAS(SCALVS_ANYCRC1)
A                                       COLHDG('AÑO CREACION')
A             MESCRC1        2S         ALIAS(SCALVS_MESCRC1)
A                                       COLHDG('MES CREACION')
A             DIACRC1        2S         ALIAS(SCALVS_DIACRC1)
A                                       COLHDG('DIA CREACION')
A             ANYULM1        4S         ALIAS(SCALVS_ANYULM1)
A                                       COLHDG('AÑO ULTIMA MODIF')
A             MESULM1        2S         ALIAS(SCALVS_MESULM1)
A                                       COLHDG('MES ULTIMA MODIF')
A             DIAULM1        2S         ALIAS(SCALVS_DIAULM1)
A                                       COLHDG('DIA ULTIMA MODIF')
A             MOTIVO1      200A         ALIAS(SCALVS_MOTIVO1)
A                                       COLHDG('MOTIVO RECHAZO')
A             CODPERA        8A         ALIAS(SCALVS_CODPERA)
A                                       COLHDG('PERSONA APROB.')
A             CODCAL1       11S         ALIAS(SCALVS_CODCAL1)
A                                       COLHDG('CODIGO CALENDARIO')
A*
A           K CODVAC1
```

Script creación tabla AFSCALVS

```
A                                       UNIQUE
A           R RAFSCALVSH                TEXT('HISTORICO VAC.SOLICITADAS')
A             CODVACH       11S         ALIAS(SCALVSH_CODVACH)
A                                       COLHDG('HISTOR.VAC.SOLICI')
A             CODVAC1       11S         ALIAS(SCALVSH_CODVAC1)
A                                       COLHDG('COD.VAC.SOLICITADAS')
A             CODPER1        8A         ALIAS(SCALVSH_CODPER1)
A                                       COLHDG('CODIGO PERSONA')
A             ANYINI1        4S         ALIAS(SCALVSH_ANYINI1)
A                                       COLHDG('AÑO INICIO')
A             MESINI1        2S         ALIAS(SCALVSH_MESINI1)
A                                       COLHDG('MES INICIO')
A             DIAINI1        2S         ALIAS(SCALVSH_DIAINI1)
A                                       COLHDG('DIA INICIO ')
A             ANYFIN1        4S         ALIAS(SCALVSH_ANYFIN1)
A                                       COLHDG('AÑO FIN')
A             MESFIN1        2S         ALIAS(SCALVSH_MESFIN1)
A                                       COLHDG('MES FIN')
A             DIAFIN1        2S         ALIAS(SCALVSH_DIAFIN1)
A                                       COLHDG('DIA FIN')
A             ANYULM1        4S         ALIAS(SCALVSH_ANYULM1)
A                                       COLHDG('AÑO ULTIMA MODIF')
A             MESULM1        2S         ALIAS(SCALVSH_MESULM1)
A                                       COLHDG('MES ULTIMA MODIF')
A             DIAULM1        2S         ALIAS(SCALVSH_DIAULM1)
A                                       COLHDG('DIA ULTIMA MODIF')
A             CODPERA        8A         ALIAS(SCALVSH_CODPERA)
A                                       COLHDG('PERSONA APROB.')
A             CODCAL1       11S         ALIAS(SCALVSH_CODCAL1)
A                                       COLHDG('CODIGO CALENDARIO')
A*
A           K CODVACH
```

Script creación tabla AFSCALVSH

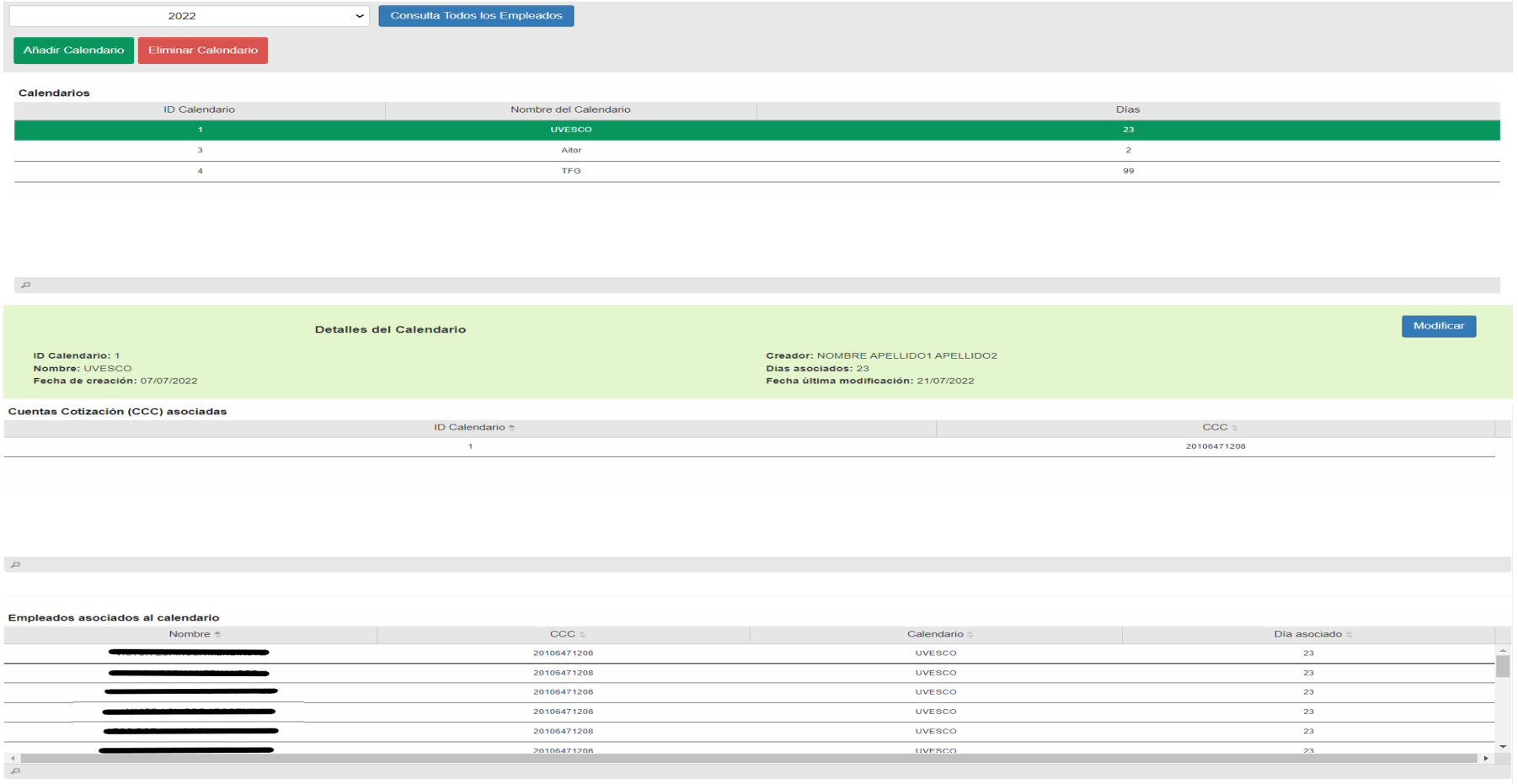
B.6 Implementación del Calendario

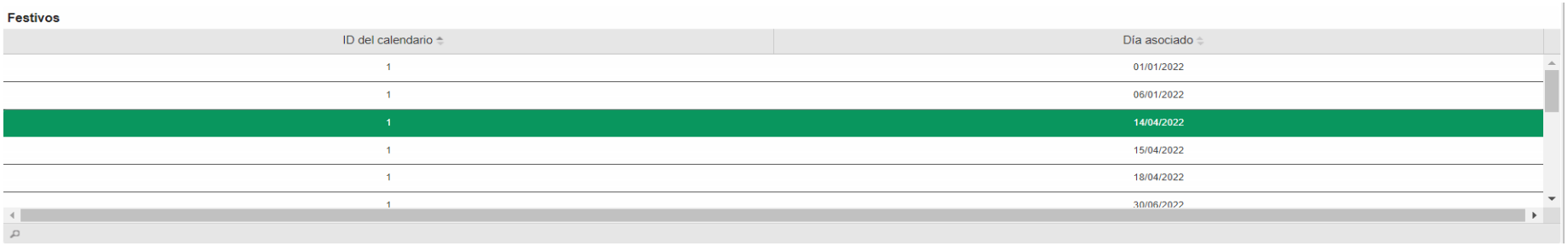
```
function mostrarCalendario(){
      var diasAsignados = [];
          $.post('/SIUC2-Vacaciones/obtenerVacacionesPorEstado.action').done(function (res){
                    $.listaDiasAA = res.calB.vacAprobadas;
              $.totAprobadas = 0;
                      if($.listaDiasAA.length == 0){
                          console.log("Atención: Empleado sin Días Descanso asignados.");
                      }else{

                          $.each($.listaDiasAA, function(i,d){
                              $.totAprobadas+=d.totDias;
                              diasAsignados.push({
                                   startDate: new Date(d.fechaIni),
                                   endDate: new Date(d.fechaFin),
                                   color: '#81b774'
                               });
                          });
                       }
      $('#aprobados').text(parseInt($.totAprobadas));

                      $.listaDiasPP = res.calB.vacPendientes;
                      $.totPendientes = 0;
                      if($.listaDiasPP.length == 0){
                          console.log("Atención: Empleado sin Días Pendientes asignados.");
                      }else{

                          $.each($.listaDiasPP, function(i,d){
                              $.totPendientes+= d.totDias;
                              diasAsignados.push({
                                   startDate: new Date(d.fechaIni),
                                   endDate: new Date(d.fechaFin),
                                   color: '#e3b419'
                               });
                          });
                       }
                    $.listaDiasPE = res.calB.vacPendientesEnviar;
                    $.totPendientesEnv = 0;
                    if($.listaDiasPE.length == 0){
                        console.log("Atención: Empleado sin Días Pendientes de Eniviar asignados.");
                    }else{

                        $.each($.listaDiasPE, function(i,d){
                            $.totPendientesEnv+=d.totDias;
                            diasAsignados.push({
                                 startDate: new Date(d.fechaIni),
                                 endDate: new Date(d.fechaFin),
                                 color: '#e3b419'
                             });
                        });

                     }
                    $.listaDiasRE = res.calB.vacRechazadas;
                    $.totRechazadas = 0;
                    $.each($.listaDiasRE, function(i,d){
                        $.totRechazadas+=d.totDias;
                    });

    $('#rechazados').text(parseInt($.totRechazadas));
    $('#pendientes').text(parseInt($.totPendientes)+parseInt($.totPendientesEnv));
    $('#disponibles').text( parseInt($.numDiasTotalCal)-(parseInt($.totAprobadas)+parseInt($.totPendientes)+parseInt($.totPendientesEnv)));

        $.post('/SIUC2-Vacaciones/obtenerDiasFestivosCalendario.action', {
            codCal :$.codigoCal,

        }).done(function (data){
            $.listaDiasFF = data.calB.fechasFestivos;
            if($.listaDiasFF.length == 0){
                console.log("Atención: No hay días festivos para ese calendario");
            }else{
                    $.each($.listaDiasFF, function(i,d){
                        diasAsignados.push({
                         startDate: new Date(d.fechaIni),
                         endDate: new Date(d.fechaFin),
                         color: '#e4edec'
                     });
                });
         }

                var anyoActual= new Date().getFullYear();
                var primerDia =new Date(anyoActual, 0, 1);
                var ultimoDia =new Date(anyoActual, 11, 31);

                const calendarAnual = new Calendar('#calendarioDiasDescansoAnual', {
                        style:'background',
                        language: 'es',
                        enableRangeSelection: false,
                        minDate: primerDia,
                        maxDate: ultimoDia,
                        displayHeader: true,
                        customDayRenderer: function(element, date) {
                            if (date.getDay() === 6 || date.getDay() === 0) {
                               $(element).css('background-color', '#e4edec');
                                $(element).css('border-radius', '0px');
                            }
                        },
                    });
                    calendarAnual.setDataSource(diasAsignados);
                    cargarDetallesVacaciones();
                });
      });
}
```

Código completo para la creación del calendario correspondiente a un empleado concreto.

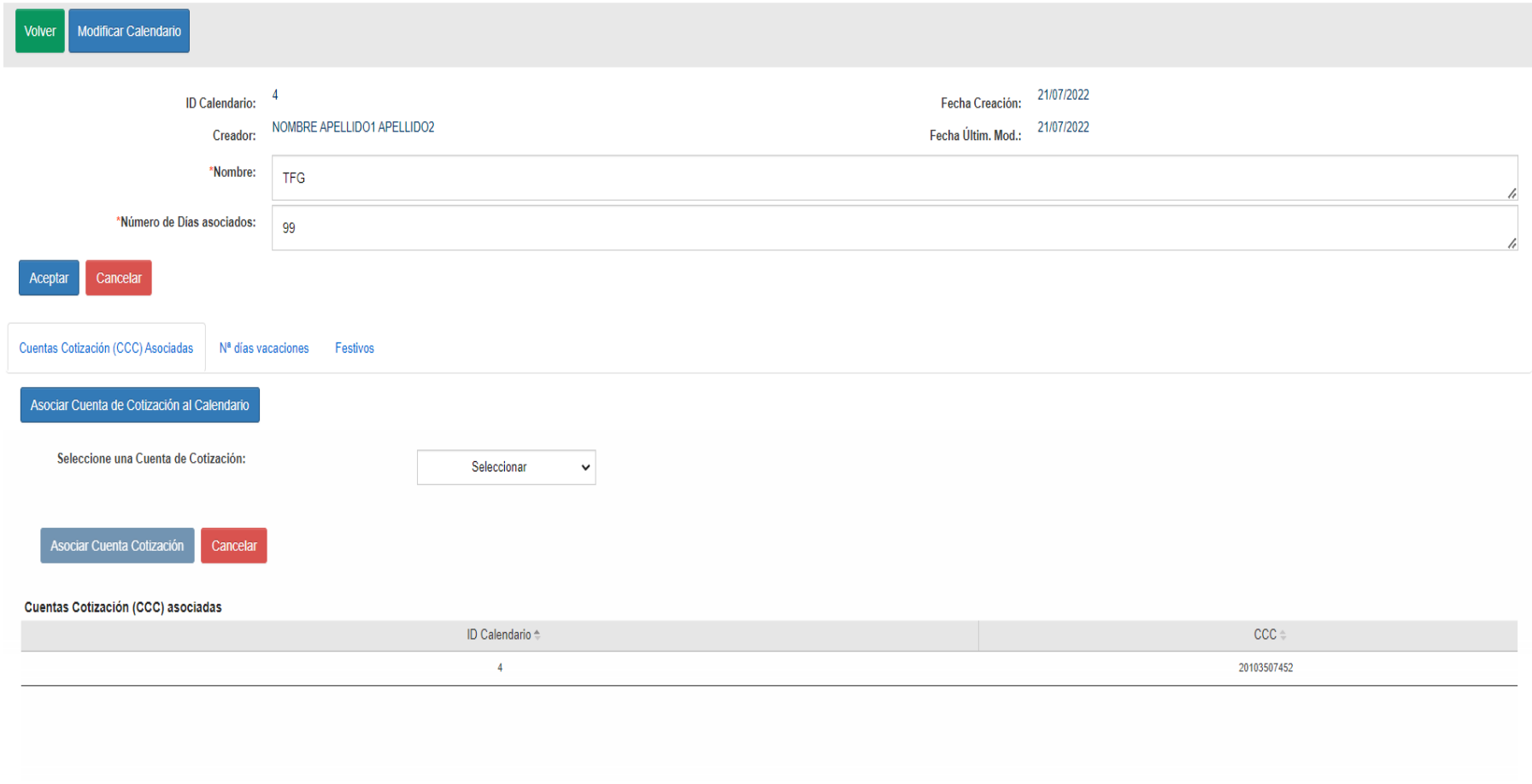
C. Páginas y vistas

C.1 Administrar Calendarios

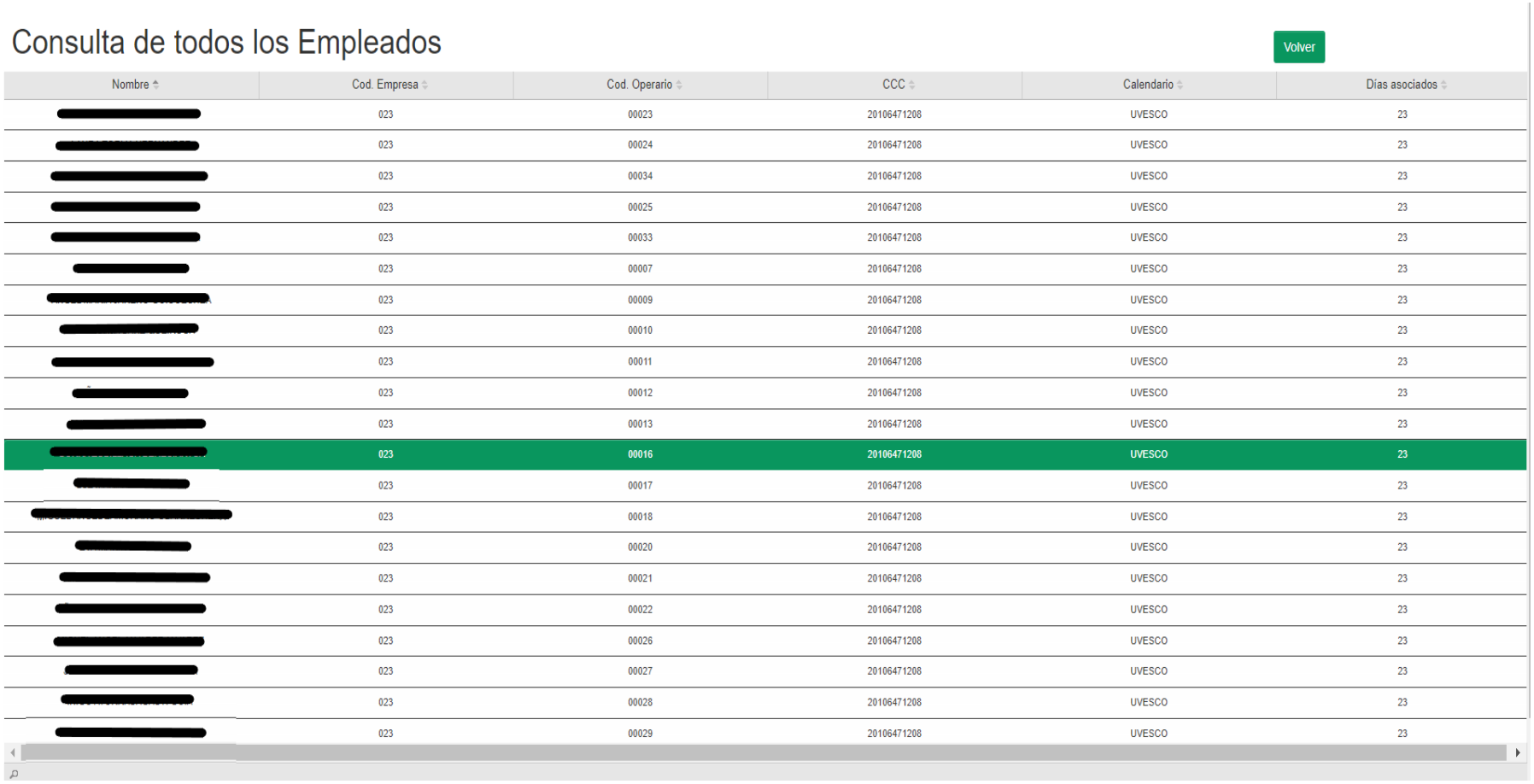




Página administradorCalendarios.jsp

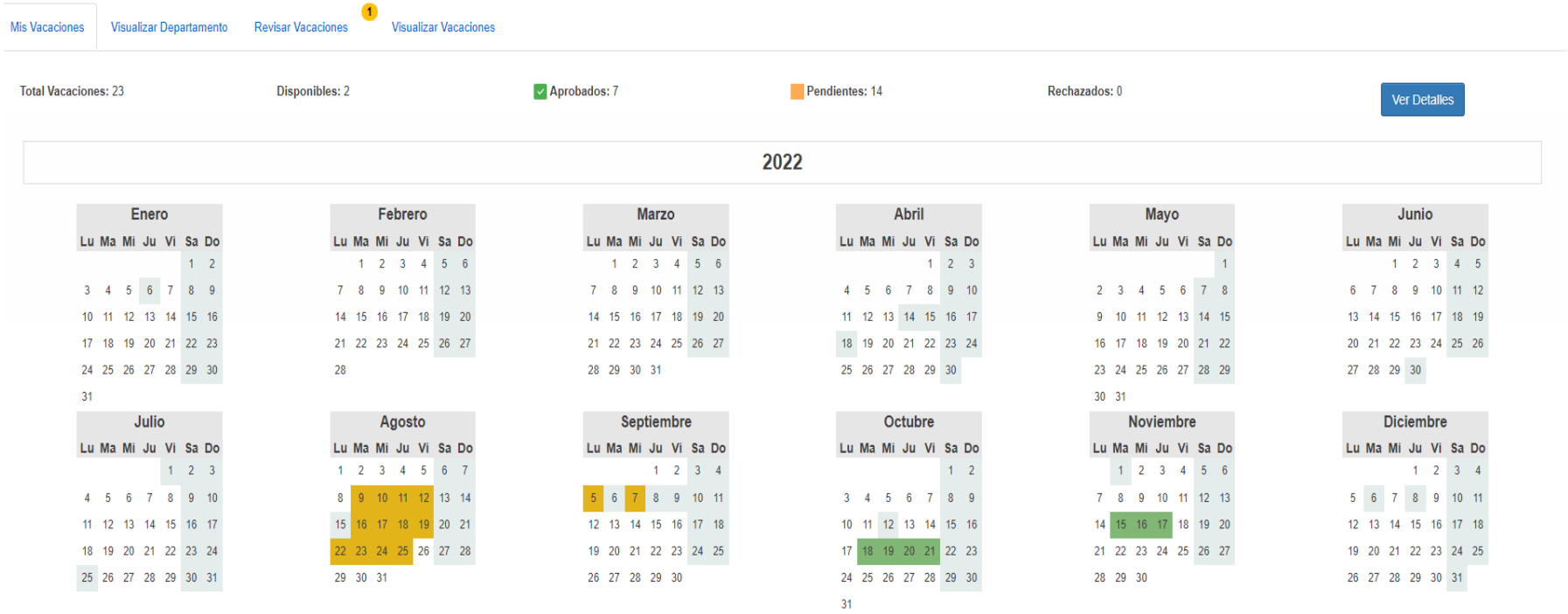


Página administradorCalendariosEditar.jsp

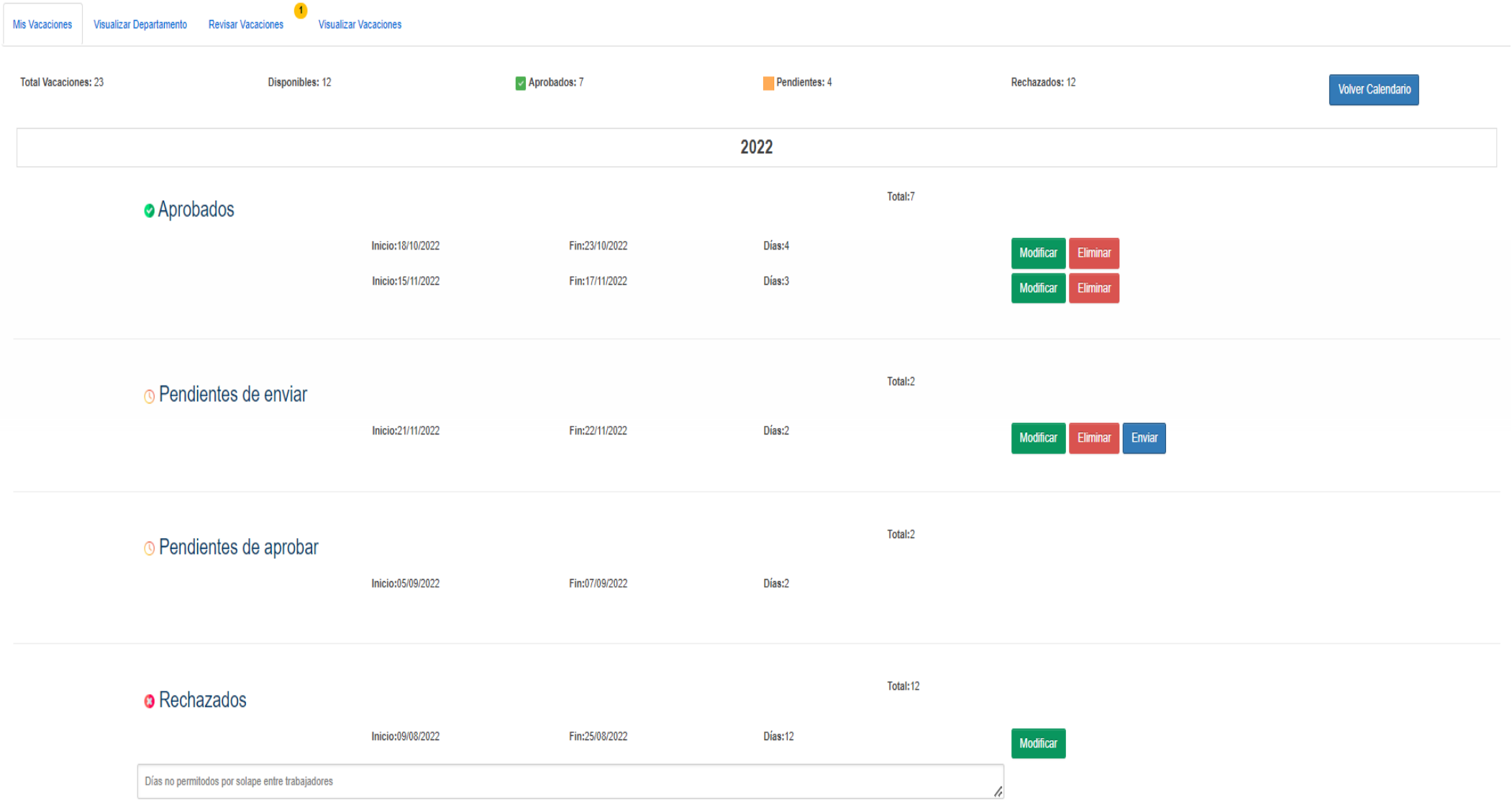


Página consultaEmpleadosGeneral.jsp

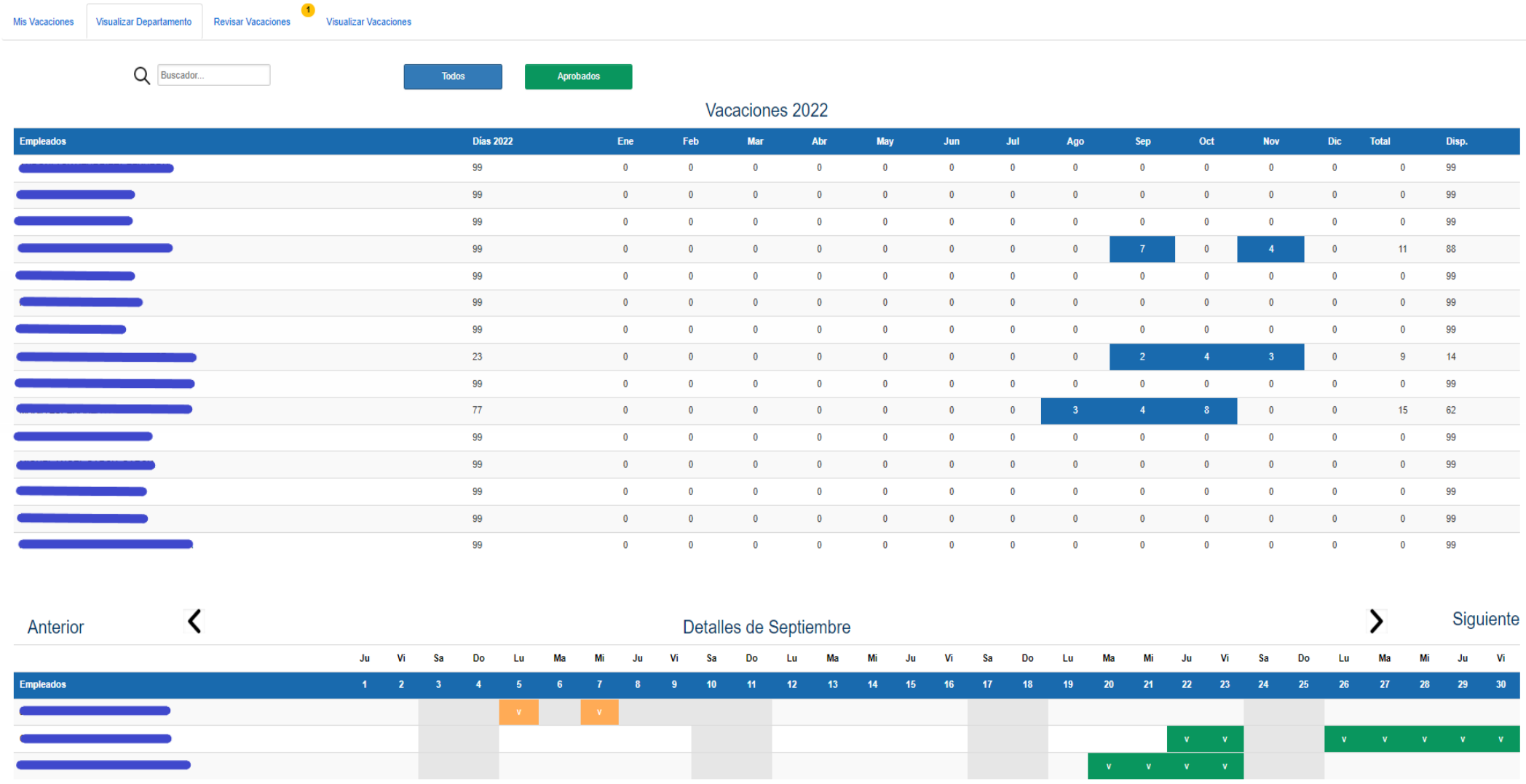
C.2 Mis Vacaciones



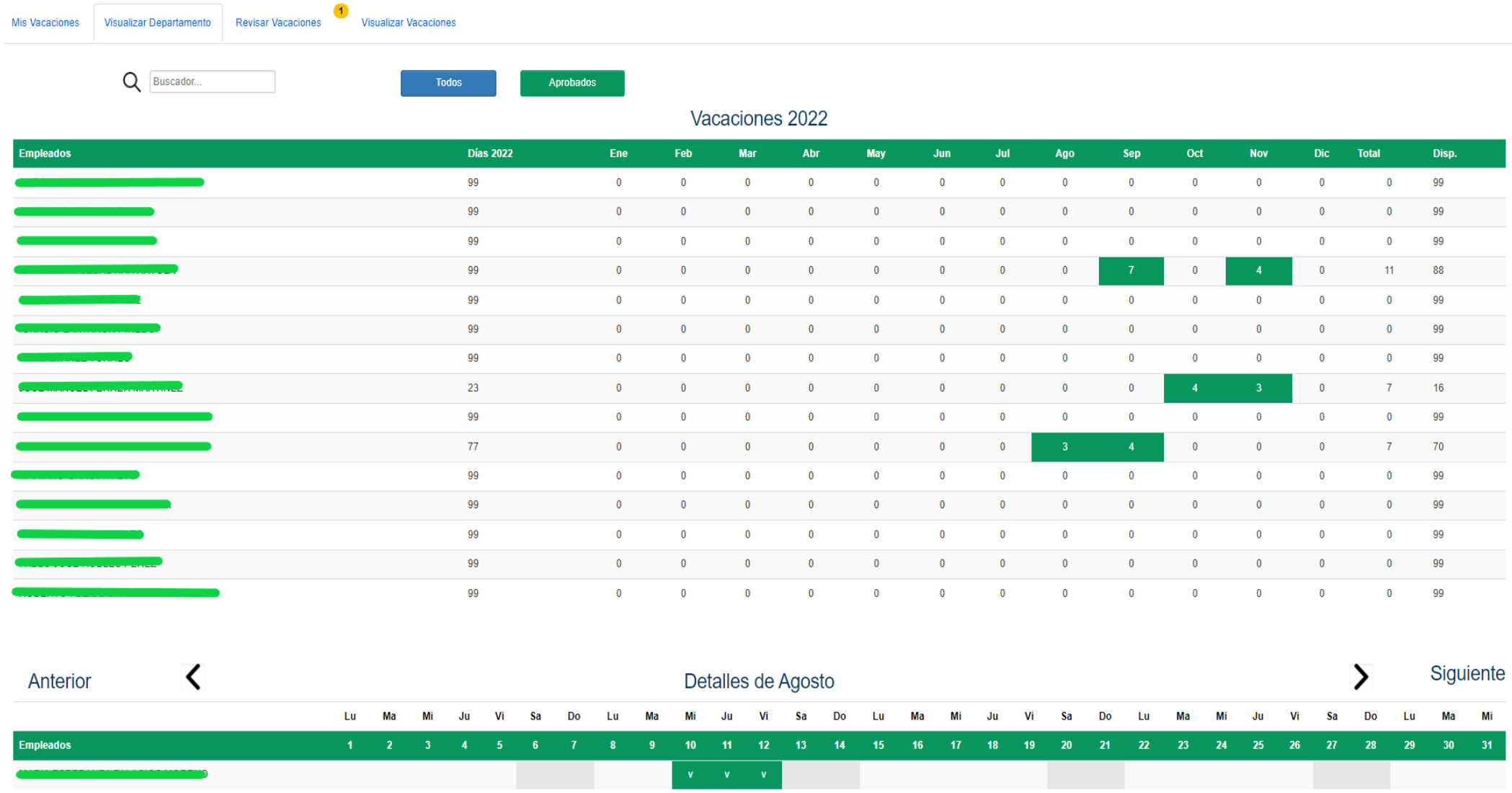
Página vacaciones.jsp y vista Calendario



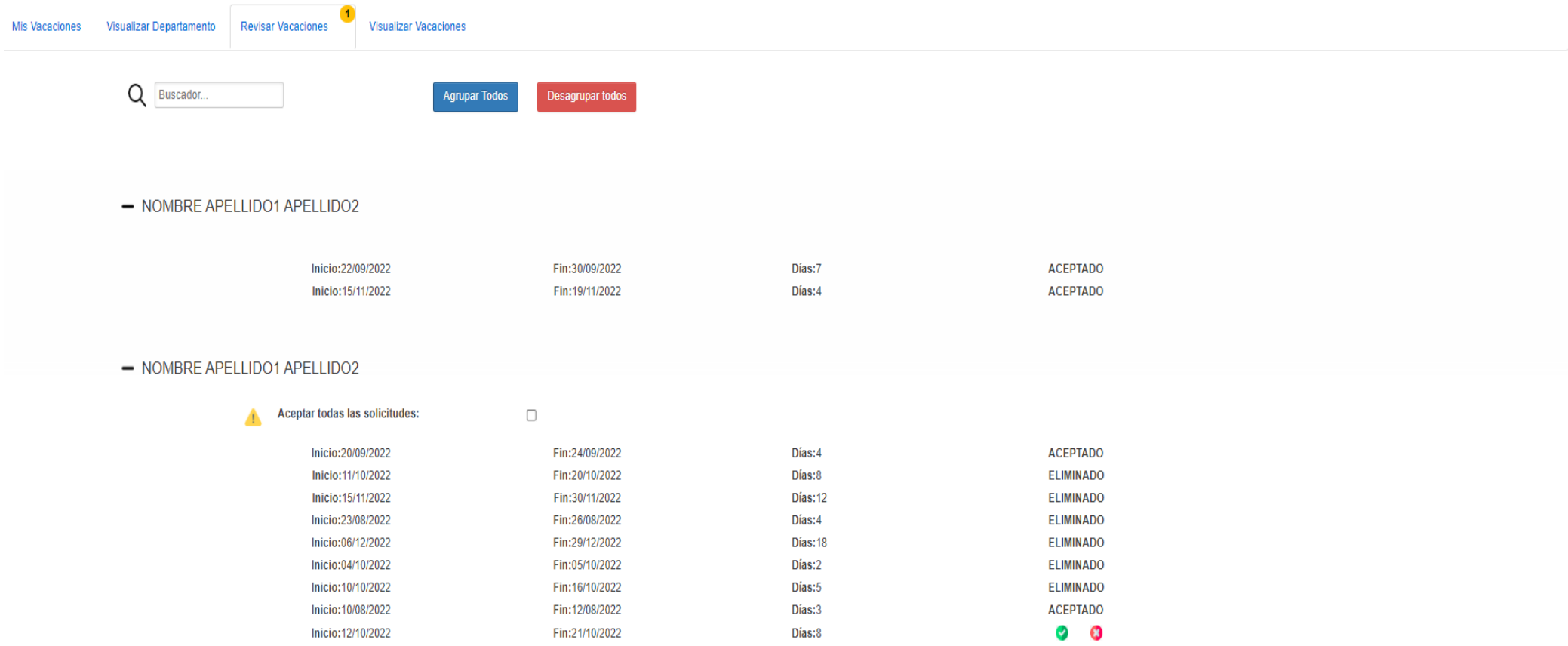
Página vacaciones.jsp y vista Ver Detalles



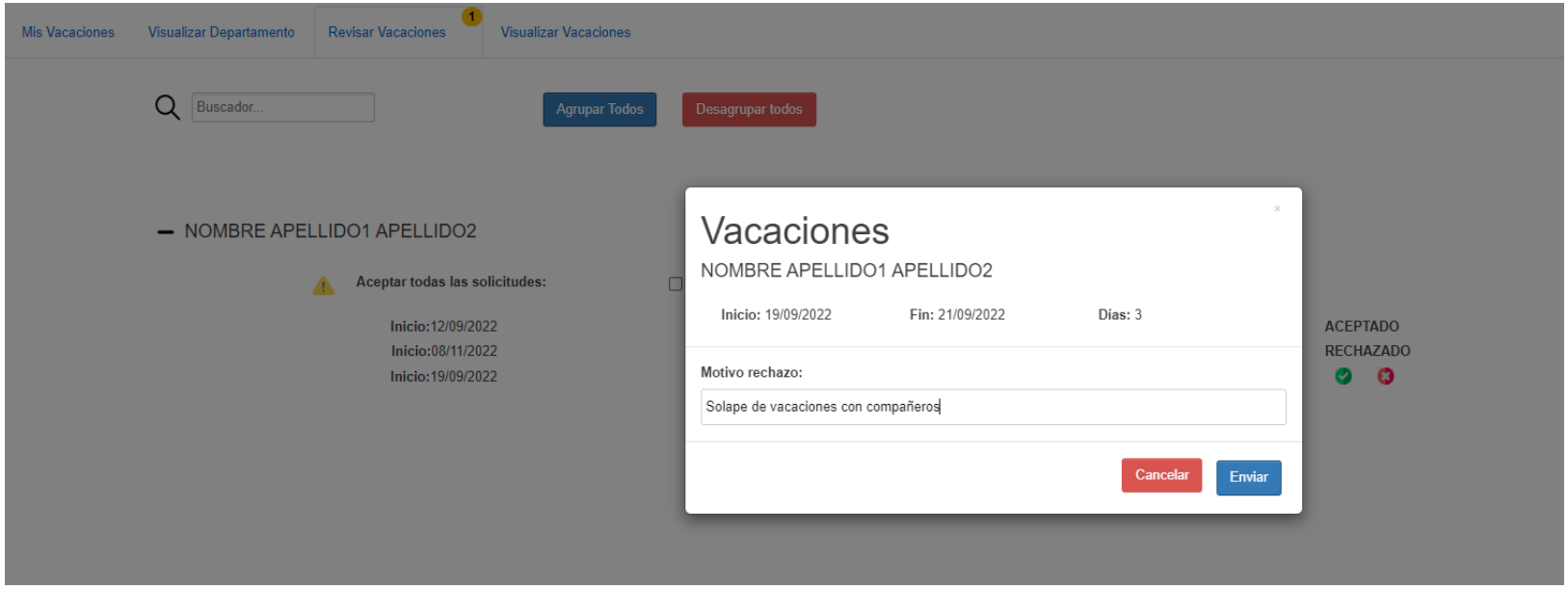
Página vacaciones.jsp y vista Visualizar Departamento + filtro Todos



Página vacaciones.jsp y vista Visualizar Departamento + filtro Aprobados

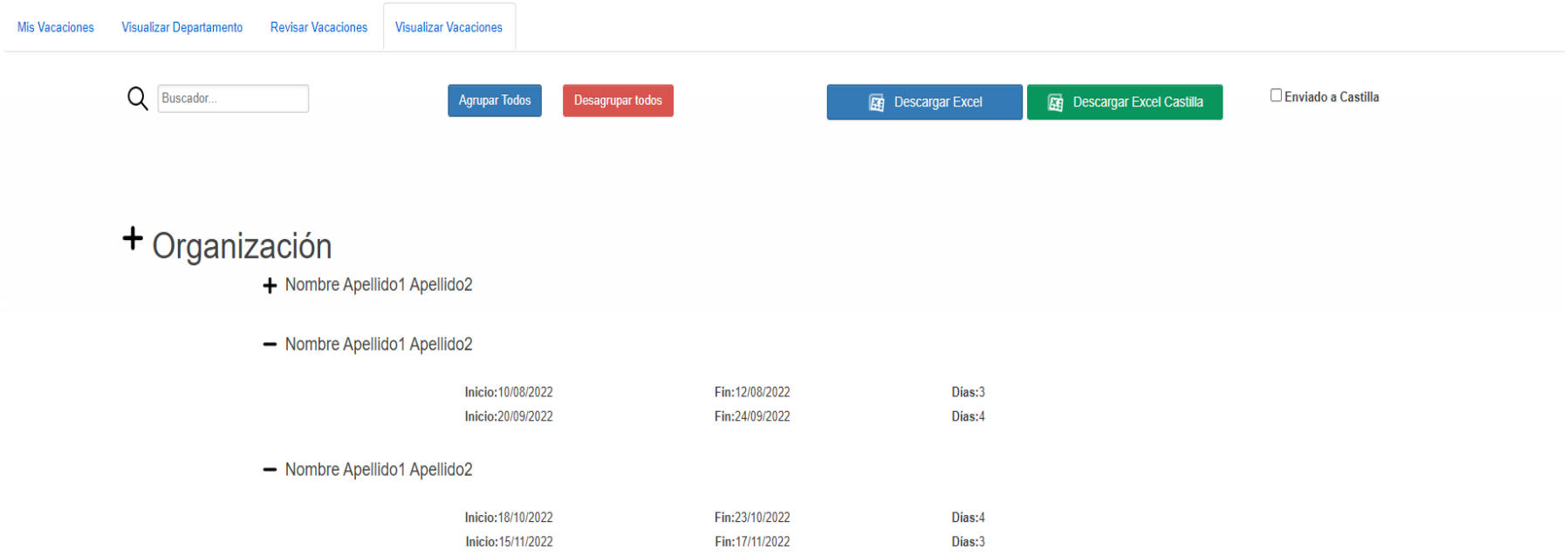


Página vacaciones.jsp y vista Revisar Vacaciones

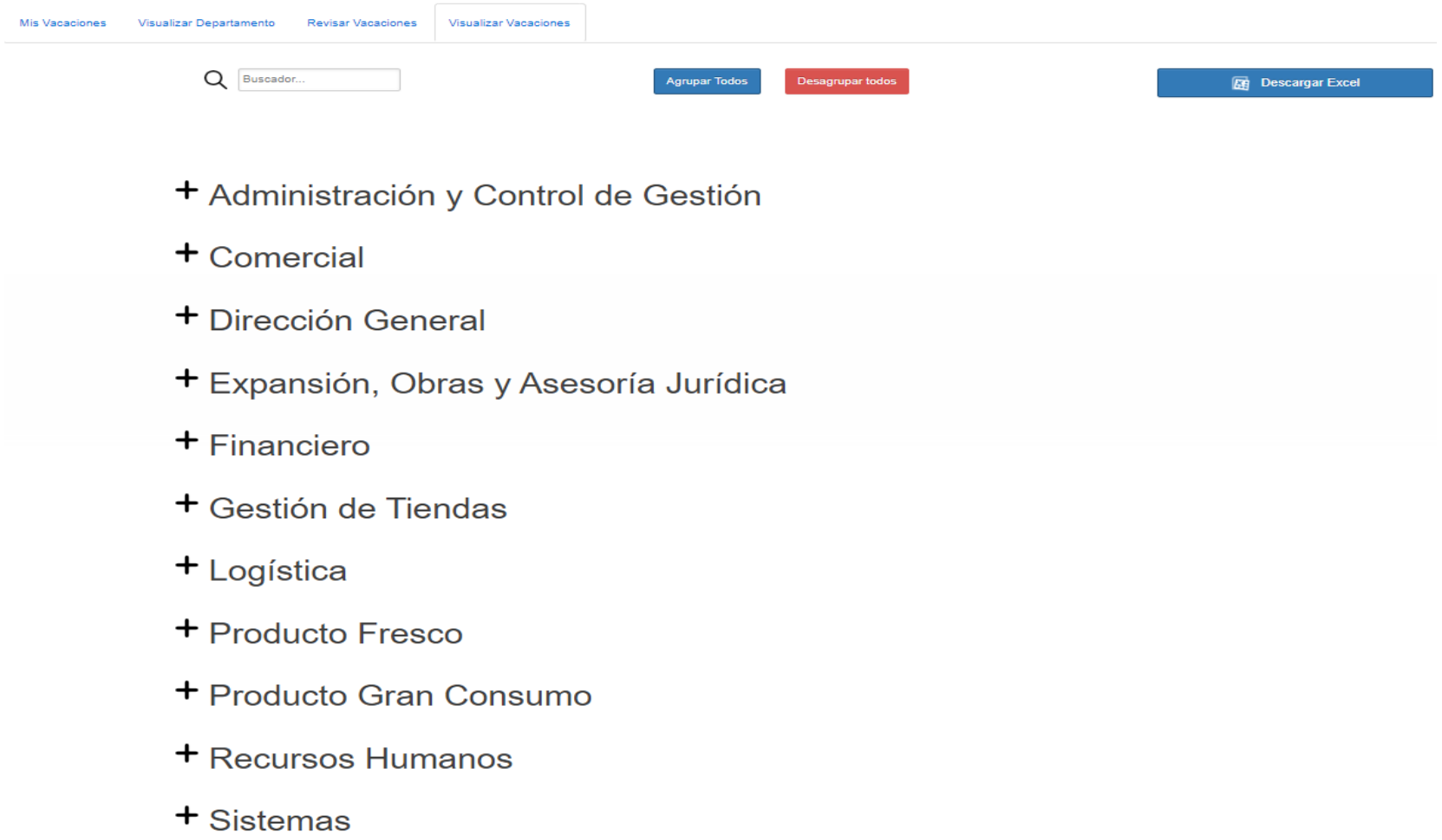


Página vacaciones.jsp y vista Revisar Vacaciones + Rechazo de vacaciones

Página vacaciones.jsp y vista Visualizar Vacaciones + Comité de Dirección



Página vacaciones.jsp y vista Visualizar Vacaciones + gestión y administración Castilla



Página vacaciones.jsp y vista Visualizar Vacaciones + Todos los departamentos

D. Pruebas*

D.1 Casos de prueba Cliente

CPC-1

Datos de prueba: Código de diferentes roles y perfiles de trabajador: visualizador, solicitante, aprobador y administrador.

Resultado esperado: Se visualizarán las páginas y visitas indicadas en la tabla 5.7 por cada uno de los roles y perfiles correspondientes.

Resultado obtenido: resultado esperado

CPC-2

Datos de prueba: Solicitudes con estado (0,1,2,3 y -1) y días festivos asociados al calendario en el que se realizarán las pruebas

Resultado esperado: visualizar los diferentes estados de las solicitudes con su color correspondiente (0= verde, 1 y 2 = naranja, 3 y -1 = no se visualizará con ningún color, días festivos o no laborales = gris)

Resultado obtenido: resultado esperado

CPC-3

Datos de prueba: Solicitudes con estado (0,1,2,3) y operaciones (crear, modificar, eliminar y enviar)

Resultado esperado: Actualizado de los contadores según su estado mostrando el número de días totales de vacaciones correspondientes al trabajador/a.

Resultado obtenido: resultado esperado

CPC-4

CPC-4.1

Datos de prueba: Selección de vacaciones con solapes

Resultado esperado: Aviso : “Existe alguna solicitud en el rango seleccionado”

Resultado obtenido: resultado esperado

CPC-4.2

Datos de prueba: Selección de días no laborales

Resultado esperado: Aviso: “Seleccione un día laboral”

Resultado obtenido: resultado esperado

CPC-4.3

Datos de prueba: Selección de fechas sin disponibilidad de días para realizar la acción

Resultado esperado: Aviso: “Supera el límite de días disponibles en la solicitud”

Resultado obtenido: resultado esperado

CPC-4.4

Datos de prueba: Selección de un rango no seleccionado previamente

Resultado esperado: Total de días seleccionado

Resultado obtenido: resultado esperado

CPC-4.5

Datos de prueba: Selección de fechas pasadas

Resultado esperado: Alerta indicando que la fecha seleccionada es pasada

Resultado obtenido: resultado esperado

*Los códigos de usuario en las pruebas no se muestran por privacidad dado que cada uno de ellos identifica a una persona

CPC-5

Datos de prueba: Acceso a la funcionalidad con diferentes perfiles de revisor

Resultado esperado: Generación de tablas en base a los empleados/as de los que es responsable el revisor

Resultado obtenido: resultado esperado

CPC-6

Datos de prueba: tablas dinámicas previamente generadas en base a los diferentes revisores.

Resultado esperado: Correcto redireccionamiento desde cualquier fila y tabla

Resultado obtenido: resultado esperado

CPC-7

Datos de prueba: Operación aceptar solicitud

Resultado esperado: Decremento del indicador de solicitudes por procesar

Resultado obtenido: resultado esperado

Datos de prueba: Operación rechazar solicitud

Resultado esperado: Decremento del indicador de solicitudes por procesar

Resultado obtenido: resultado esperado

CPC-8

Datos de prueba: Búsqueda de texto existente en la página

Resultado esperado: Palabra resaltada en base a la búsqueda

Resultado obtenido: resultado esperado

Datos de prueba: Búsqueda de texto no existente en la página

Resultado esperado: Ningun palabra resaltado

Resultado obtenido: resultado esperado

CPC-9

Datos de prueba: Solicitudes mostradas en la pantalla en las vistas : Revisar Vacaciones y Visualizar Vacaciones

Resultado esperado: Correcto despliegue y repliegue vertical de la información ocultándola o mostrándola

Resultado obtenido: resultado esperado

D.2 Casos de prueba Servidor

CPS-1

Datos de prueba: Creación de una solicitud

Resultado esperado: Inserción en la base de datos con estado 2

Resultado obtenido: resultado esperado

CPS-2

Datos de prueba: Envío de una solicitud para ser revisar

Resultado esperado: Actualizado de estado de la solicitud de 2 a 1

Resultado obtenido: resultado esperado

CPS-3

Datos de prueba: Eliminación de diferentes solicitudes

Resultado esperado: Actualizado del estado 2 o 0 a -1

Resultado obtenido: resultado esperado

CPS-4

Datos de prueba: Solicitud ya existente y nueva selección de fechas

Resultado esperado: Actualizado de campos y de las fecha inicio y fin en una solicitud ya existente

Resultado obtenido: resultado esperado.

CPS-5

Datos de prueba: Selección de aquellas solicitudes con estado 0 y 1 asociadas al código del revisor que ha accedido a la funcionalidad

Resultado esperado: Solicitudes con estado 0 y 1

Resultado obtenido: resultado esperado.

CPS-6

Datos de prueba: Selección de solicitudes con estado 1 asociadas al código del revisor que ha accedido a la funcionalidad

Resultado esperado: Solicitudes con estado 0

Resultado obtenido: resultado esperado.

CPS-7

Datos de prueba: Solicitud con estado 1

Resultado esperado: Actualizado de campos y estado de la solicitud a 0

Resultado obtenido: resultado esperado

CPS-8

Datos de prueba: Solicitudes con estado 1 asociadas a un empleado/a concreto

Resultado esperado: Actualizado de campos y estado de las diferentes solicitudes asociadas al empleada/o a estado 0

Resultado obtenido: resultado esperado

CPS-9

CPS-9.1

Datos de prueba: Solicitud con estado 1 e inserción de motivo

Resultado esperado: Actualizado de los campos de la solicitud y estado a 3

Resultado obtenido: resultado esperado

CPS-9.2

Datos de prueba: Solicitud con estado 1 sin inserción del motivo de rechazo

Resultado esperado: No actualizado de la solicitud, se mantiene en el estado actual

Resultado obtenido: resultado esperado

CPS-10

Datos de prueba: Información de las solicitudes mostrada o existente en la página "Visualizar Vacaciones"

Resultado esperado: Descarga de datos en formato .xls con las solicitudes desagrupadas por empleado/a y campos : departamento, nombre, nº intervalo, fecha inicio, fecha fin y nº días.

Resultado obtenido: resultado esperado

CPS-11

Datos de prueba: Información de las solicitudes mostrada o existente en la página "Visualizar Vacaciones"

Resultado esperado: Descarga de datos en formato .xls con las solicitudes desagrupadas por empleado/a y campos: nombre, id empresa, nif, fecha inicio y fecha fin

Resultado obtenido: resultado esperado

CPS-12

Datos de prueba: Calendarios existentes para el año actual

Resultado esperado: Actualizado en el campo INDCAS1 a 1 , véase en la tabla 5.1, para todos los calendarios existentes en el año actual

Resultado obtenido: resultado esperado

CPS-13

CPS-13.1

Datos de prueba: Código del empleado/a no asociado al comité de dirección y no perteneciente a los puestos descritos en el siguiente apartado, 3.1.1.3 Visualizador, del capítulo de analisis de requisitos

Resultado esperado: Visualizado de las solicitudes, con estado aprobado, del departamento al que está asociado el trabajador/a.

Resultado obtenido: resultado esperado

CPS-13.2

Datos de prueba: Código del empleado/a asociado al comité de dirección pero no perteneciente a los puestos descritos en el siguiente apartado 3.1.1.3 Visualizador

Resultado esperado: Visualizado de las solicitudes, con estado aprobado, del departamento al que está asociado además de las solicitudes asociadas a los diferentes integrantes del comité de dirección.

Resultado obtenido: resultado esperado

CPS-13.3

Datos de prueba: Código del empleado/a asociado al comité de dirección y perteneciente a alguno de los puestos descritos en el siguiente apartado 3.1.1.3 Visualizador

Resultado esperado: Visualizado de las solicitudes aprobadas en cada uno de los departamentos asociadas a cada empleado/a además del visualizado de las solicitudes asociadas al comité de dirección.

Resultado obtenido: resultado esperado

CPS-14

CPS-14.1

Datos de prueba: Solicitud realizada por el empleado/a

Resultado esperado: Envío de la notificación indicando la creación de la solicitud

Resultado obtenido: resultado esperado

CPS-14.2

Datos de prueba: Solicitud modificada por el empleado/a

Resultado esperado: Envío de correo indicando la modificación realizada

Resultado obtenido: resultado esperado

CPS-14.3

Datos de prueba: Solicitud procesada por el revisor

Resultado esperado: Envío de correo indicando el aprobado o rechazo por parte del supervisor

Resultado obtenido: resultado esperado

CPS-15

Datos de prueba: Calendario existente

Resultado esperado: eliminación del calendario y sus correspondientes asociaciones como: cuentas de cotización, días festivos y posibles excepciones de días para empleados/as

Resultado obtenido: resultado esperado

CPS-16

Datos de prueba: Nombre del calendario y número de días

Resultado esperado: Inserción en la base de datos de un calendario sin asociaciones

Resultado obtenido: resultado esperado

CPS-17

CPS-17.1

Datos de prueba: Nombre del calendario e ID del calendario

Resultado esperado: Actualizado del nombre, en la base de datos, del calendario seleccionado

Resultado obtenido: resultado esperado

CPS-17.2

Datos de prueba: Número de días entre 0 y 99 e ID del calendario

Resultado esperado: Actualizado del número de días para el calendario seleccionado

Resultado obtenido: resultado esperado

CPS-17.3

Datos de prueba: Número de días diferente de [0-99] e ID del calendario

Resultado esperado: No actualizado del calendario, manteniéndolo en el estado actual

Resultado obtenido: resultado esperado

CPS-18

Datos de prueba: ID del calendario y número de cuenta de cotización

Resultado esperado: Inserción en la base de datos asociando el calendario con la cuenta de cotización.

Resultado obtenido: resultado esperado

CPS-19

Datos de prueba: ID del calendario y número de cuenta de cotización

Resultado esperado: Eliminado, en la base datos, de la asociación respecto al calendario además del eliminado de las posibles excepciones para empleados/as correspondientes a esa cuenta

Resultado obtenido: resultado esperado

CPS-20

Datos de prueba: ID del calendario, número de cuenta de cotización, código del empleado/a y un número el cual indicará los días a modificar

Resultado esperado: Inserción o actualizado respecto a los días que el empleado/a podrá disfrutará de vacaciones

Resultado obtenido: resultado esperado

CPS-21

CPS-21.1

Datos de prueba: Fecha del año actual no asociada

Resultado esperado: Inserción en la base de datos

Resultado obtenido: resultado esperado

CPS-21.2

Datos de prueba: Fecha ya asociada del año actual

Resultado esperado: Aviso y no asociación / inserción en la base de datos

Resultado obtenido: resultado esperado

CPS-22

Datos de prueba: Fecha asociada al calendario
Resultado esperado: Eliminación de la fecha asociada al calendario como festivo
Resultado obtenido: resultado esperado

D.3 EndPoint

CPS-23

CPS-23.1

Datos de prueba: Código usuario asociado al departamento de sistemas
Resultado esperado: Datos JSON asociados al código del usuario
Resultado obtenido: resultado esperado

CPS-23.1

Datos de prueba: Código usuario asociado al departamento de recursos humanos
Resultado esperado:Datos JSON asociados al código del usuario
Resultado obtenido: resultado esperado

CPS-23.3

Datos de prueba: Código usuario no existente
Resultado esperado: Usuario no encontrado devolviendo un estado 5 en la estructura de datos del JSON
Resultado obtenido: resultado esperado

CPS-24

CPS-24.1

Datos de prueba: Código usuario asociado al departamento RRHH y área -1
Resultado esperado: Datos JSON asociados al código del usuario
Resultado obtenido: resultado esperado

CPS-24.2

Datos de prueba: Código usuario asociado al departamento RRHH y área RH1
Resultado esperado: Datos JSON asociados al código del usuario
Resultado obtenido: resultado esperado

CPS-25

CPS-25.1

Datos de prueba: Código usuario nivel 2 ya que no cuenta con aprobado, vease en la tabla 3.1
Resultado esperado: Datos JSON con los datos del usuario únicamente
Resultado obtenido: resultado esperado

CPS-25.2

Datos de prueba: Código usuario nivel 1
Resultado esperado: Datos JSON con los datos del usuario únicamente
Resultado obtenido: resultado esperado

CPS-25.3

Datos de prueba: Código usuario correspondiente al Director/a RRHH
Resultado esperado: Datos JSON con los datos del usuario únicamente
Resultado obtenido: resultado esperado

CPS-25.4

Datos de prueba: Código usuario, con aprobador, nivel 6
Resultado esperado: Datos JSON asociados al usuario
Resultado obtenido: resultado esperado

E. Planificación

E.1 Planificiacíon Cuarta Convocatoria

	Nombre de la tarea	Inicio	Finalizar
1	Trabajo fin de Grado	21/02/22	15/09/22
2	Hito 1: Inicio del proyecto	21/02/22	21/02/22
3	Hito 5: Matriculación del TFG	11/07/22	11/07/22
4	Hito 8: Subida del proyecto a la plataforma	04/09/22	04/09/22
5	Hito 9: Defensa del proyecto	14/09/22	14/09/22
6	Hito 10: Fin del proyecto y publicación de las notas	15/09/22	15/09/22
7	Adquisición de Conocimientos	22/02/22	09/03/22
8	Proceso Solicitud de Vacaciones (PSV)	22/02/22	09/03/22
9	Análisis del proceso de solicitud	22/02/22	09/03/22
10	Análisis del la jerarquía de la empresa, organigrama	22/02/22	09/03/22
11	Tecnologías y Herramientas (TH)	03/03/22	08/03/22
12	Análisis del entorno de desarrollo en el que se ejecutará la app web	03/03/22	08/03/22
13	Análisis de las tecnologías para desarrollos existentes de Interfaz	03/03/22	08/03/22
14	Análisis de las tecnologías utilizadas en desarrollos existentes.	03/03/22	08/03/22
15	Análisis de implementación de tablas en el AS400	03/03/22	08/03/22
16	Implantación de la Herramienta (IH)	09/03/22	09/03/22
17	Análisis del proceso de implantación de la herramienta	09/03/22	09/03/22
18	Análisis de la gestión del servidor	09/03/22	09/03/22
19	Hito 2: Fin adquisición de conocimientos	09/03/22	09/03/22
20	Diseño de la Aplicación	10/03/22	31/03/22
21	Diseño del proceso de selección vacaciones (DPSV)	10/03/22	14/03/22
22	Diseño de un proceso común	10/03/22	10/03/22
23	Elaboración del proceso	10/03/22	14/03/22
24	Diseño de la solución WEB (DWEB)	15/03/22	31/03/22
25	Hito 3: Análisis de requisitos web: Reunión	15/03/22	15/03/22
26	Diseño de las tablas en el servidor AS400	15/03/22	16/03/22
27	Diseño de la Arquitectura	17/03/22	17/03/22
28	Diseño del endPoint en el Web Service	18/03/22	22/03/22
29	Diseño de la Interfaz	22/03/22	31/03/22
30	Hito 4: Fin diseño de la aplicación	31/03/22	31/03/22
31	Implementación	01/04/22	05/08/22
32	Implementación del Front-End (IFE)	01/04/22	04/05/22
33	Desarrollo de las interfaces	01/04/22	11/04/22
34	Desarrollo de la funcionalidad de las interfaces	12/04/22	04/05/22
35	Implementación Back-End (IBE)	05/05/22	05/08/22
36	Desarrollo de las tablas para implementación en AS400	05/05/22	06/05/22
37	Desarrollo del WS	09/05/22	13/05/22
38	Desarrollo de la funcionalidades en base al análisis de requisitos	16/05/22	05/08/22
39	Hito 6: Fin implementación	05/08/22	05/08/22

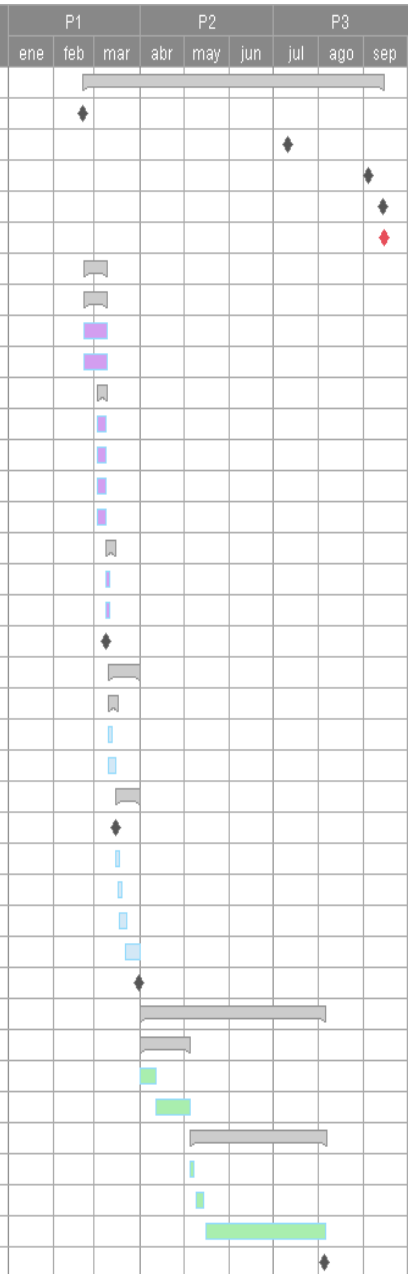
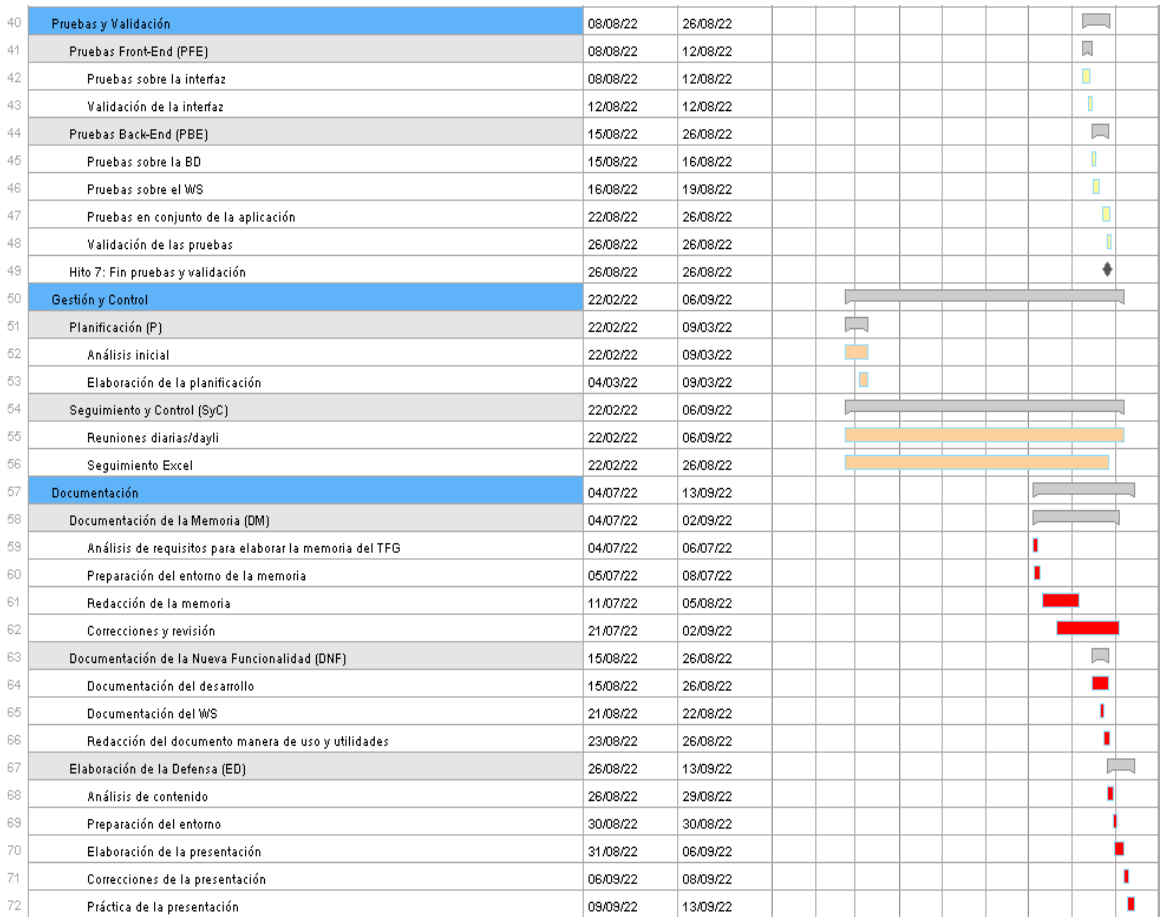


Diagrama Gantt mensual 4º Convocatoria - primera parte



40	Pruebas y Validación	08/08/22	26/08/22
41	Pruebas Front-End (PFE)	08/08/22	12/08/22
42	Pruebas sobre la interfaz	08/08/22	12/08/22
43	Validación de la interfaz	12/08/22	12/08/22
44	Pruebas Back-End (PBE)	15/08/22	26/08/22
45	Pruebas sobre la BD	15/08/22	16/08/22
46	Pruebas sobre el WS	16/08/22	19/08/22
47	Pruebas en conjunto de la aplicación	22/08/22	26/08/22
48	Validación de las pruebas	26/08/22	26/08/22
49	Hito 7: Fin pruebas y validación	26/08/22	26/08/22
50	Gestión y Control	22/02/22	06/09/22
51	Planificación (P)	22/02/22	09/03/22
52	Análisis inicial	22/02/22	09/03/22
53	Elaboración de la planificación	04/03/22	09/03/22
54	Seguimiento y Control (SyC)	22/02/22	06/09/22
55	Reuniones diarias/dayli	22/02/22	06/09/22
56	Seguimiento Excel	22/02/22	26/08/22
57	Documentación	04/07/22	13/09/22
58	Documentación de la Memoria (DM)	04/07/22	02/09/22
59	Análisis de requisitos para elaborar la memoria del TFG	04/07/22	06/07/22
60	Preparación del entorno de la memoria	05/07/22	08/07/22
61	Redacción de la memoria	11/07/22	05/08/22
62	Correcciones y revisión	21/07/22	02/09/22
63	Documentación de la Nueva Funcionalidad (DNF)	15/08/22	26/08/22
64	Documentación del desarrollo	15/08/22	26/08/22
65	Documentación del WS	21/08/22	22/08/22
66	Redacción del documento manera de uso y utilidades	23/08/22	26/08/22
67	Elaboración de la Defensa (ED)	26/08/22	13/09/22
68	Análisis de contenido	26/08/22	29/08/22
69	Preparación del entorno	30/08/22	30/08/22
70	Elaboración de la presentación	31/08/22	06/09/22
71	Correcciones de la presentación	06/09/22	08/09/22
72	Práctica de la presentación	09/09/22	13/09/22

Diagrama Gantt mensual 4º Convocatoria - segunda parte