Escuela Politécnica Superior de Linares

Grado en Ingeniería Telemática

UNIVERSIDAD DE JAÉN

Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

SERVICIO DE GESTIÓN DE RELACIONES DE NECESIDADES PARA DEPARTAMENTOS

Alumno: Pedro Javier Sáez Mira

Tutor: Prof. D. Juan Carlos Cuevas Martínez

Depto.: Ingeniería de Telecomunicación

Septiembre, 2020

Resumen

En este Trabajo de Fin de Grado, titulado ‘Servicio de gestión de relación de necesidades para departamentos’, se ha diseñado e implementado un servicio telemático sobre protocolo HTTP para la gestión del proceso administrativo de tramitación de relación de necesidades dentro de un departamento en la Universidad de Jaén. Este servicio se ha desplegado como un portal web el cual tiene varias vistas de usuario según el rol de usuario activo: personal docente e investigador, responsable de unidad de gasto o responsable de gestión.

El portal web, el cual está conectado con una base de datos, cuenta un servicio de autenticación de usuarios, creación de relación de necesidades y su posterior servicio de añadir pedidos, edición, modificación, validación y tramitación de esta relación. También cuenta con una bandeja de entrada en la cual se puede ir observando el estado de tramitación de dichas relaciones de necesidades.

Abstract

In this Final Degree Project, entitled 'Needs relationship management service for departments', a telematic service over HTTP protocol has been designed and implemented for the management of the administrative process of processing the needs relationship within a department in the Jaen University. This service has been deployed as a web portal which has various user views according to the active user role: teaching and research staff, head of the spending unit or head of management.

The web portal, which is connected to a database, has a user authentication service, creation of a relationship of needs and its subsequent service of adding orders, editing, modifying, validating and processing this relationship. It also has an inbox in which you can observe the status of the processing of these relationships of needs.

Índice

Índice de ilustraciones

Introducción

Hoy en día en este mundo dominado por la tecnología e Internet en la que todos los trámites administrativos, o prácticamente todos, que se pueden hacer de manera física tiene su réplica "on-line", un proceso tan común en la Universidad de Jaén como es el de gestionar las relaciones de necesidades del personal docente e investigador (PDI), debería tener su correspondiente servicio telemático donde poder llevar a cabo esa gestión. Dada la trascendencia de este servicio se propuso el presente Trabajo Fin de Grado (TFG) en el que se lleva a cabo el diseño e implementación de dicho servicio.

Una relación de necesidades es un documento que un PDI rellena con sus datos personales y dirección de entrega del pedido de material que desea realizar. También debe poner la unidad de gasto a la que se quiere acoger para realizar ese pedido con su respectivo código y, por supuesto, la parte principal que es la relación de artículos que conforma el pedido en cuestión que quiere realizar, añadiendo la cantidad de cada uno y su precio.

El proceso que sigue una relación desde que se crea hasta que se hace el pedido al proveedor es: el PDI rellena el documento de relación de necesidades con sus datos personales y con los pedidos que desea realizar, esta relación llega al responsable de unidad de gasto y éste debe revisar la relación y decidir en cuanto al presupuesto disponible de su unidad de gasto a la que se acoge esa relación si es posible llevar a cabo ese pedido o no. Si no da por válida la relación, ésta vuelve al profesor y éste debe modificarla. En cambio, si está todo correcto da su visto bueno y la relación pasa al responsable de gestión para que inicie los trámites pertinentes.

Actualmente, todo este proceso, desde que el PDI decide crear una relación, hasta que el responsable de gestión la tramita, se realiza manualmente, es decir, el PDI rellena el documento a mano y lo firma (en papel o en formato PDF), se lo entrega al responsable de unidad de gasto para su validación y firma (si éste no la da por válida se la entrega de nuevo al PDI y éste la modifica, se la devuelve). Cuando el responsable de la unidad de gasto la da por válida, se la entrega al responsable de gestión, por ejemplo por email, y éste ya realiza los trámites de contacto con el proveedor, facturación, recepción e inventariado del material, apoyándose en la plataforma de Universitas XXI *[11].*

Como se puede intuir, que todo este trámite se realice manualmente (aunque se incluyan documentos electrónicos) es tedioso, generando retrasos (mucho más si es en papel), posibles pérdidas de documentos en el correo, etc. Por lo tanto, viviendo en una época

tan tecnológica era necesario hacer un servicio para llevar todo este trámite de una forma más rápida, simple y donde se pueda crear una relación de necesidades, editarla, duplicarla, además de comprobar la evolución de la misma, todo ello desde un mismo sitio web. En consecuencia, este Trabajo Fin de Grado desarrollará este servicio a través del diseño e implementación de mismo pasando de realizarse de manera manual a ser un proceso completamente telemático.

El presente Trabajo Fin de Grado consistirá en un portal web el cual tiene varias vistas de usuario según el rol de usuario activo: PDI, responsable de unidad de gasto o responsable de gestión. Este portal web se apoyará en una base de datos y aportará los siguientes servicios: autenticación de usuarios, creación de relación de necesidades y su posterior servicio de añadir pedidos, edición, modificación, validación y tramitación de esta relación. También cuenta con una bandeja de entrada en la cual se puede ir observando el estado de tramitación de dichas relaciones de necesidades.

1.1 Estado del arte

El servicio web objeto del presente TFG incorpora, como casi todas las plataformas web de hoy en día, el acceso al mismo a través de usuario y contraseña. Una vez se accede con las credenciales de usuario se presentará una vista similar a la de un gestor de correo electrónico, como por ejemplo la aplicación web de Gmail, y en el cual se mostrará el menú de bandeja de entrada y demás estados como Pendientes de validación, Pendientes de Modificación, Validadas, Denegadas, Borradores y Eliminadas. Depende de la pestaña del menú que se seleccione, se podrá ver las relaciones en una vista previa y según la que se seleccione se podrá ver más detallada. Igual que si se abre un correo electrónico.

Por otra parte, también se basa en otros tipos de aplicaciones web como las de compraventa, como puede ser Amazon o eBay, ya que se puede añadir productos y realizar un pedido aunque más bien está basada en aplicaciones web de gestión administrativa

Analizando aplicaciones y plataformas web similares a este TFG se ha llegado a la conclusión de que las aplicaciones que hay en el mercado para clientes no corporativos son bastantes escasas y simples. Una de estas aplicaciones es "MisPedidos" que a continuación se explicará un poco más en detalle.

1.1.1 MisPedidos

Esta aplicación permite crear pedidos, añadir sus productos y enviarlo a sus proveedores. En la siguiente ilustración se podrá tener una idea principal de lo que nos permite hacer esta aplicación *[8]*.

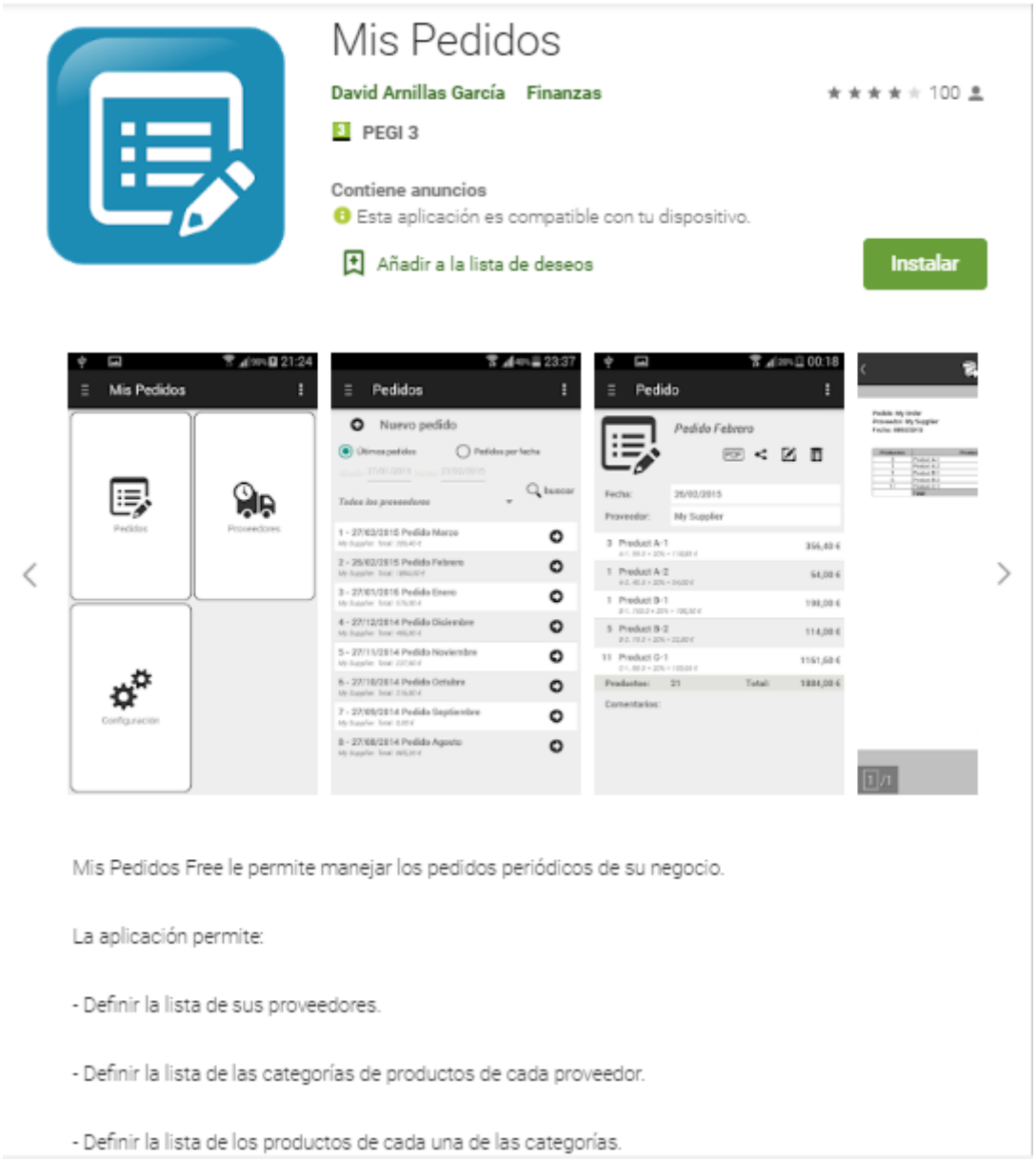


Ilustración 1. Aplicación de Mis Pedidos

1.1.2 Universitas XXI

También cabe destacar aquí la aplicación 'Universitas XXI' que es la que se usa actualmente para que el responsable de gestión realice el trámite de la relación. Esta aplicación es el software de gestión líder en universidades, empleándolo más de 100 universidades. Sin embargo, como sólo sirve para realizar el trámite de la relación una vez

esta es recibida por el responsable de gestión, todo el proceso que sigue dicha relación desde que un PDI decide crear una, hasta que se tramita, no está incluido en esta aplicación y de aquí la motivación de este Trabajo Fin de Grado.

Ilustración 2. Vista principal de la plataforma Universitas XXI

1.2 Objetivos

El presente Trabajo Fin de Grado tiene los siguientes objetivos principales:

1. Diseñar e implementar un servicio telemático sobre protocolo HTTP para la gestión del proceso administrativo de tramitación de relación de necesidades dentro de un departamento en la Universidad de Jaén, que proporcione las funciones de creación, modificación.
2. El servicio deberá contar, al menos, con una vista para el profesorado, otra para el responsable/s de gestión y otra para el responsable/s de las unidades de gasto.
3. El servicio deberá estar implementado en tecnologías actuales que permitan su visualización en el mayor número posible de clientes (navegadores web) y su adaptación a los diferentes tamaños de pantalla más habituales.
4. El servicio desarrollado deberá tener en cuenta las medidas de protección y seguridad necesarias para que pueda ser ofrecido con las debidas garantías a sus usuarios.

5. El servicio deberá de proporcionar los mecanismos adecuados de información y gestión de privacidad y gestión de datos de carácter personal.
6. El servicio deberá proporcionar, además del servicio general del punto 1., un servicio de registro y autenticación de usuarios, un servicio de registro de accesos y transacciones.
7. El servicio, según el trámite, deberá ofrecer la posibilidad de firmar digitalmente los documentos que se generen en el proceso administrativo.

Y como objetivos opcionales:

8.- Servicio de papelera de documentos, para prevenir la pérdida de los mismos.

9.- Autenticación con el servicio de la Universidad de Jaén (SIDUJA).

Desarrollo del proyecto

2.1 Proceso de una relación de necesidades

2

El presente proyecto, como ya se ha explicado anteriormente, surge de la necesidad de dar cobertura al proceso administrativo de una relación de necesidades dentro de un departamento universitario de la Universidad de Jaén. Por lo tanto, se va a proceder a explicar dicho proceso administrativo.

En primer lugar, antes de explicar con detalle el servicio que se ha realizado debemos explicar con más detalle qué es una relación de necesidades y el proceso que sigue desde que se inicia hasta que se tramita.

Una relación de necesidades es un documento, tipo solicitud, que un PDI formaliza para solicitar la adquisición de bienes vinculados con su labor docente y/o investigadora, tales como consumibles, equipos informáticos, material de oficina, etc. Estos documentos se rellenan con datos del solicitante tales como nombre, apellidos, teléfono, departamento/centro o servicio y edificio o dependencia donde entregar el pedido de material que desea realizar. También se debe poner la unidad de gasto a la que se quiere acoger para realizar ese pedido con su respectivo código, además del listado de artículos a adquirir, su cantidad y su precio. En la Ilustración 3 se puede ver un ejemplo de una relación de necesidades y en los siguientes apartados se explicará también los agentes que participan en este proceso con más detalle y el proceso que sigue una relación.

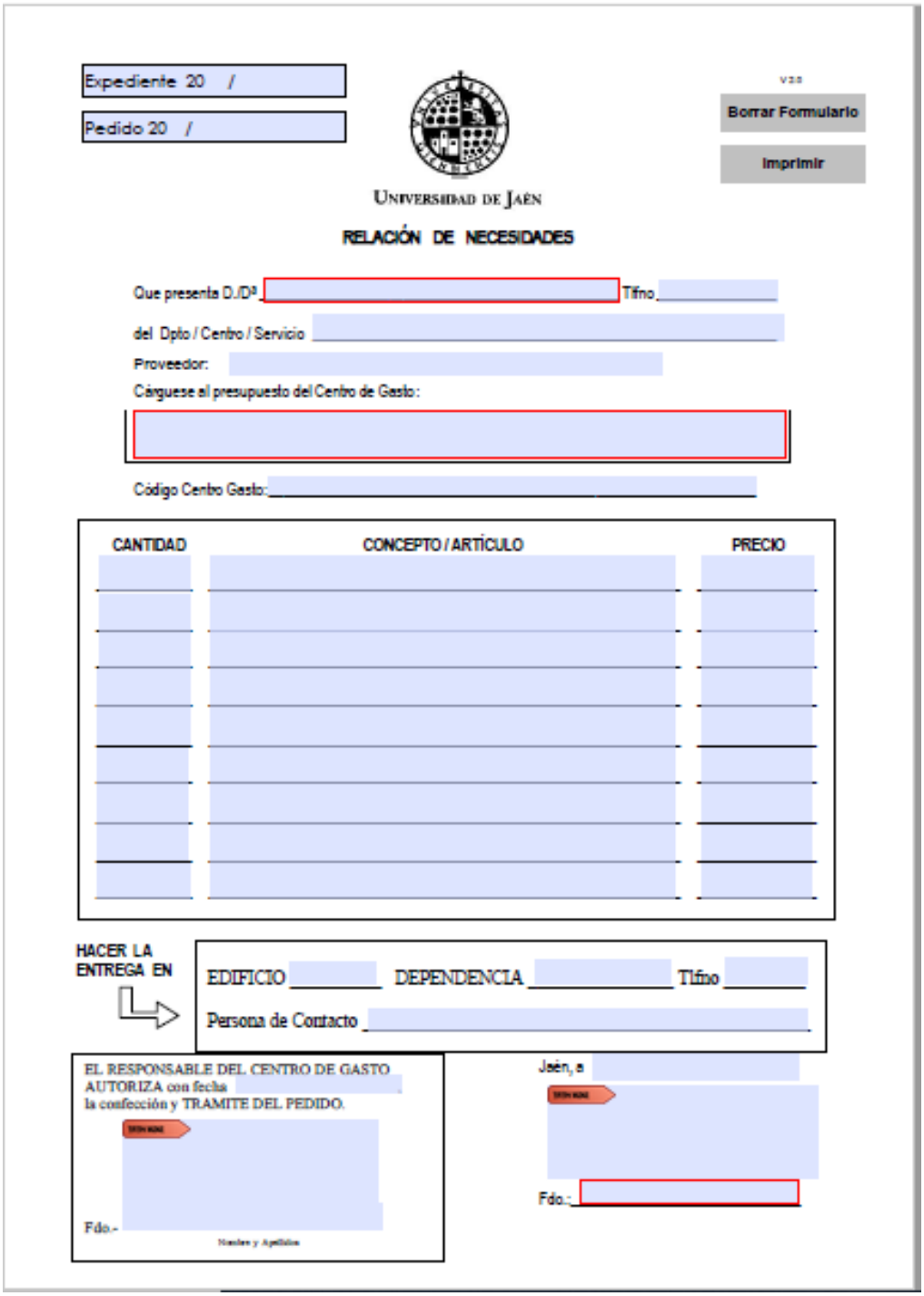


Ilustración 3. Relación de Necesidades

2.1.1 Agentes

Los agentes que intervienen en este proceso son:

- **PDI** → Persona que realiza la petición de servicio o suministro

Acción

- ✓ Solicitud de un relación de necesidades

- **Responsable de Unidad de Gasto→** Persona que autoriza la compra con cargo a su Unidad de Gasto (Cada Unidad de Gasto tiene un responsable).

  Acción

  - ✓ Visto bueno Relación de necesidades.
  - ✓ Firma de Informe de necesidad de pedido.

- **Responsable de Gestión→** Personal de Administración que gestiona los Pedidos.

  Acción

  - ✓ Realización Informe de Necesidad.
  - ✓ Realización de Pedido a Proveedores.
  - ✓ Realización de Fichas de Inventario y Fichas para el Servicio Central de Informática.

- **Proveedores→** Quien sirve el suministro o servicio.

2.1.2 Proceso de una Relación de necesidades

El proceso que sigue una relación desde que se crea hasta que se hace el pedido al proveedor es el siguiente:

1. El PDI rellena el documento de relación de necesidades con sus datos citados anteriormente y con los pedidos de productos que desea realizar
2. La relación llega al responsable de unidad de gasto y éste debe revisar la relación y decidir en cuanto al presupuesto disponible de su unidad de gasto a la que se acoge esa relación si es posible llevar a cabo ese pedido o no. Si no da por válida la relación, vuelve al profesor y éste debe modificarla. En cambio, si está todo correcto, da su visto bueno y la relación pasa al responsable de gestión para que inicie los trámites pertinentes.
3. El responsable de gestión escoge una de las relaciones disponibles que ya están validadas por el responsable de unidad de gasto e inicia los trámites pertinentes, es decir, ponerle el número de expediente, los proveedores correspondientes etc.

Todo este proceso desde que el PDI decide crear una relación, hasta que el responsable de gestión la tramita, se realiza manualmente, es decir, el PDI rellena el documento a mano (o en un PDF), se lo entrega al responsable de unidad de gasto, si no la

da por válida, se la entrega de nuevo al PDI y éste la modifica, se la vuelve a entregar al responsable de unidad de gasto. Cuando decide darla por válida, se la entrega al responsable de gestión y éste ya realiza los trámites a través de la plataforma de Universitas XXI.

2.1.3 Proceso de una Relación de necesidades en el portal web

Ahora en lo que respecta al proyecto y lo que se debe implementar es este proceso explicado anteriormente pero en una plataforma web donde cada agente perteneciente a este proceso puede acceder a esta plataforma a través de un proceso de autenticación con sus credenciales de usuario y dependiendo del agente que sea se mostrará una vista u otra donde podrás realizar unas determinadas acciones u otras tales como crear una relación, editarla, añadirle pedidos, validar, denegar, tramitar y visualizar todas estas relaciones según su estado, validar, denegar, tramitar etc. Todo este proceso de la plataforma se explicará con mayor detalle más adelante. En la Ilustración 4 se muestra un esquema de este proceso:

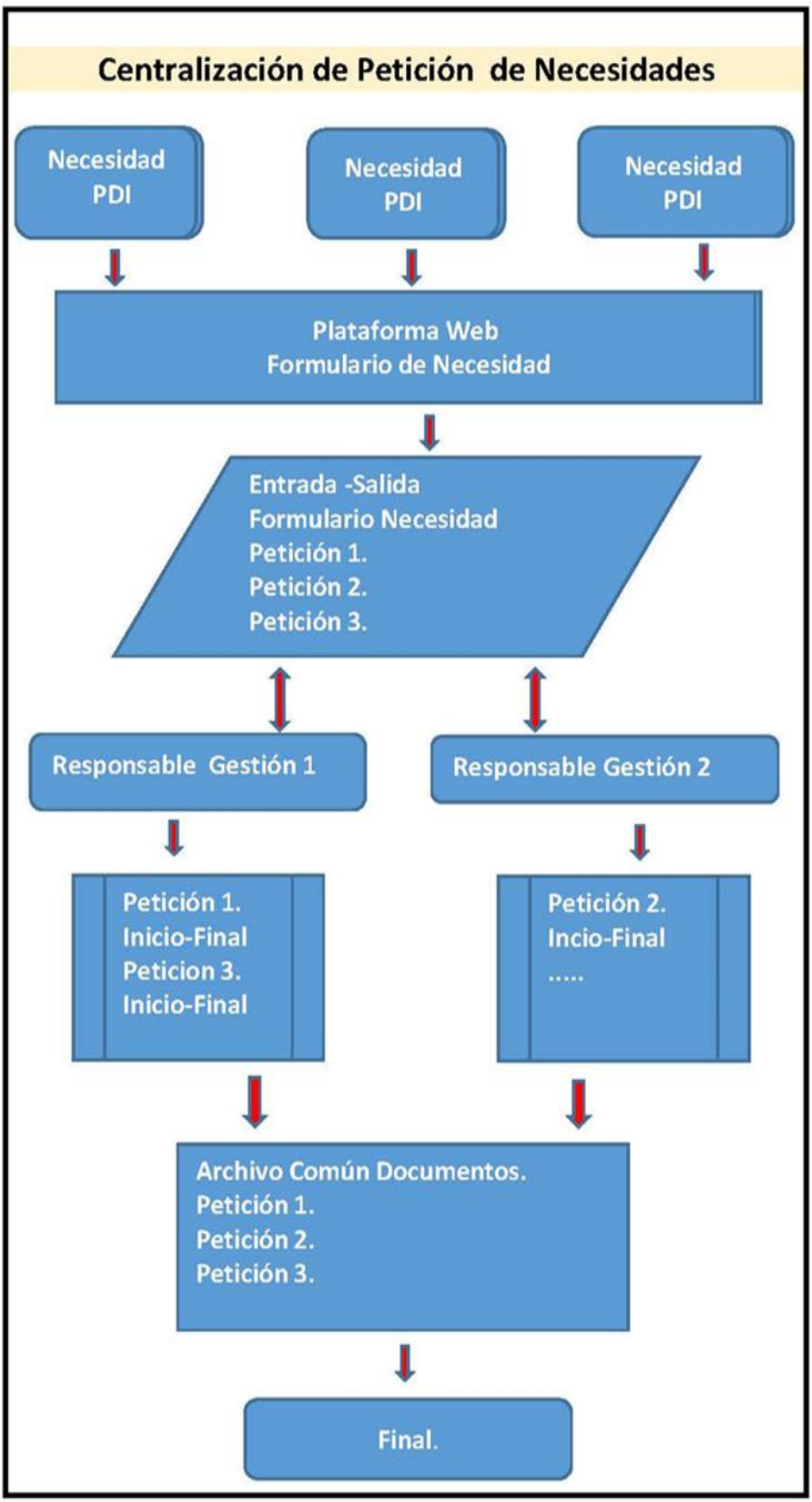


Ilustración 4. Proceso de una Relación de Necesidades

Y todo este proceso llevado a la plataforma y pasando por cada estado será:

- El PDI crea una relación en "*Nueva Relación"* donde ya vienen sus datos y el sólo tiene que añadir el título y fecha de dicha relación y la unidad de gasto a la que se quiere acoger.
- Una vez la cree, dicha relación se trasladará a *"Borradores"* donde podrá añadirle los productos que desee y la dirección de entrega y una vez la envíe se irá a *"Pendientes de Validación"* y a la vez también le llegará al Responsable de Unidad de Gasto a su *"Bandeja de Entrada".*
- Si el Responsable de Unidad de Gasto decide no dar por válida la relación, será devuelta al PDI y se alojará en el estado *"Denegadas (Pendientes de Modificación)"*. Una vez el PDI la corrija y la vuelva a enviar volverá a la *"Bandeja de Entrada"* del Responsable de Unidad de Gasto y cuando decida darla por válida irá a la "Bandeja de Entrada" del Responsable de Gestión y ya éste cuando decida tramitar una relación sólo deberá seleccionarla e iniciar el trámite.

Otros estados también son *"Validadas"* y *"Eliminadas"* donde tanto el PDI como el Responsable de Unidad de Gasto podrán consultar las Relaciones que se encuentren en dicho estado.

Una máquina de estados de este proceso sería la que se muestra en la Ilustración 5:

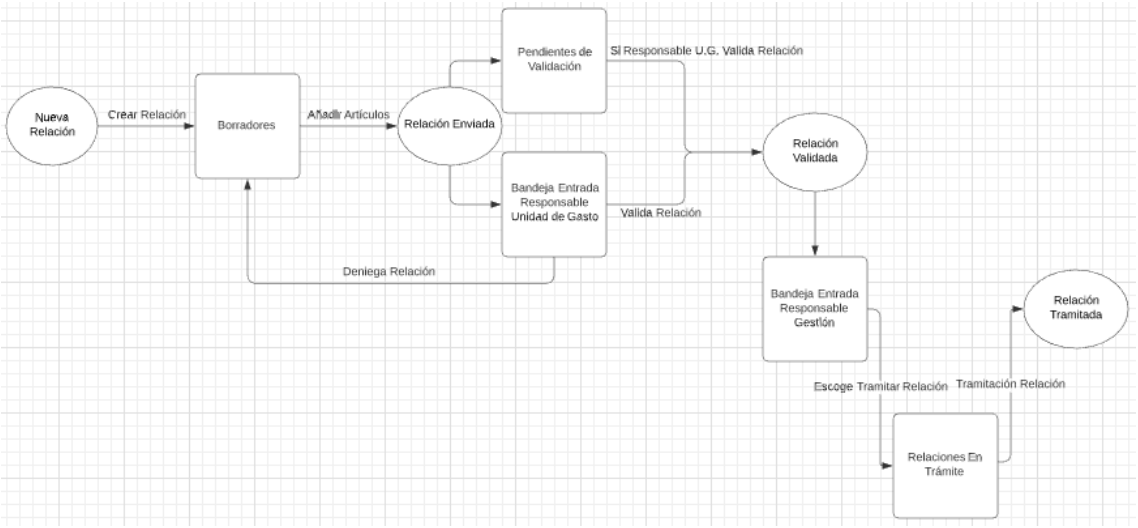


Ilustración 5. Máquina de Estados de una Relación en la Plataforma Web

2.2 Descripción del sistema desarrollado

En este apartado se va a hacer una descripción del sistema desarrollado como pueden ser las tecnologías usadas, la arquitectura, el servidor, cliente, base de datos etc.

2.2.1 Tecnologías usadas en este proyecto

Este proyecto se basa en el patrón de arquitectura Modelo-Vista-Controlador (MVC) que sirve para separar la representación de la información con la que interactúa el usuario y los componentes para poder llevar esta representación a cabo. En el presente TFG se ha empleado el Framework de Spring para poder desarrollar la aplicación con este patrón de arquitectura software. Como herramienta de desarrollo se ha empleado Eclipse, el cual incorpora amplias funcionalidades para el desarrollo con Spring empleando el lenguaje de programación Java.

El servicio desarrollado en este TFG también está basado en la arquitectura cliente-servidor, donde el cliente realiza una petición al servidor y esté le entrega una respuesta. Todo ello controlado por el protocolo de comunicación HTTP. También se hace una consulta o una operación a la base de datos (BBDD) para poder obtener los datos que necesita el cliente. También se usarán cookies para almacenar información de estado del cliente.

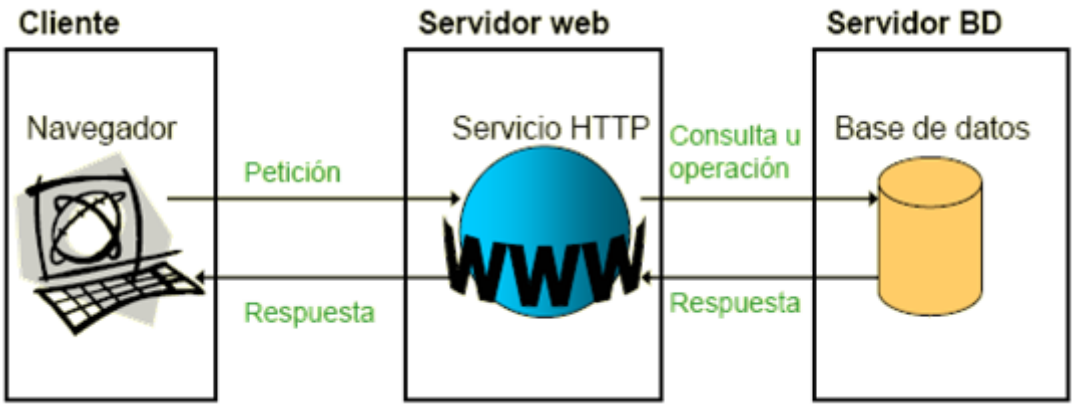


Ilustración 6. Arquitectura cliente-servidor

En el lado del cliente se usará la tecnología HTML, CSS y JavaScript y en el lado del servidor se usará Apache Tomcat. Y en el lado de la BBDD se usará el sistema gestor de base de datos MySQL. Y para englobar todas estas tecnologías se usará el servidor WAMP en el que proporciona en un PC con Windows el servidor Apache y el sistema gestor de base de datos MySQL. Además, WAMP incorpora una herramienta bastante útil como es PhpMyAdmin con la que poder manejar y administrar nuestra base de datos.

Estas son las tecnologías usadas en este proyecto y que se explicarán después con más detalle.

2.2.2 Arquitectura MVC

Como se ha comentado anteriormente para esta aplicación se ha usado el patrón de arquitectura Modelo-Vista-Controlador (MVC) para separar la representación de la información con la que interactúa el usuario y los componentes para poder llevar esta representación a cabo. Esta arquitectura nos ofrece una solución robusta para el desarrollo de aplicaciones robustas y que sean portables, escalables, extensibles y más fáciles de mantener.

Básicamente este patrón de arquitectura software tiene tres partes diferenciadas y que a la vez interactúan entre ellas que son:

- El modelo: es la capa de datos de la aplicación. Éste define los datos y suministra los métodos disponibles para manejarlos.
- La vista: define la capa de presentación de la aplicación, y es el interfaz que ve el usuario y con el que trabaja. Simultáneamente a la presentación de datos, suministra al usuario la capacidad de interacción.
- El controlador: representa el enlace entre las órdenes del usuario (eventos generados por la interacción del usuario con la vista) y la respuesta del modelo. Es el encargado de manejar el flujo de la aplicación. Si es necesario, hará cambios en el modelo conforme a las órdenes del usuario.

Mantener el modelo, la vista y el controlador lo más independientes posible entre ellos, hace más fácil una modificación posterior del código. Si es necesario hacer cambios en alguna de las capas (ej. cambiar la forma de presentar los datos al usuario), se minimiza cualquier cambio en el resto.

Particularizado a nuestra aplicación web, y usando tecnología servlet[1] y JSP (Páginas de servidor de Java, en inglés 'Java Server Pages'), un esquema de este modelo MVC podría ser el que se muestra en la Ilustración 7:

[1] Un servlet es una clase en el lenguaje de programación Java, utilizada para ampliar las capacidades de un servidor.

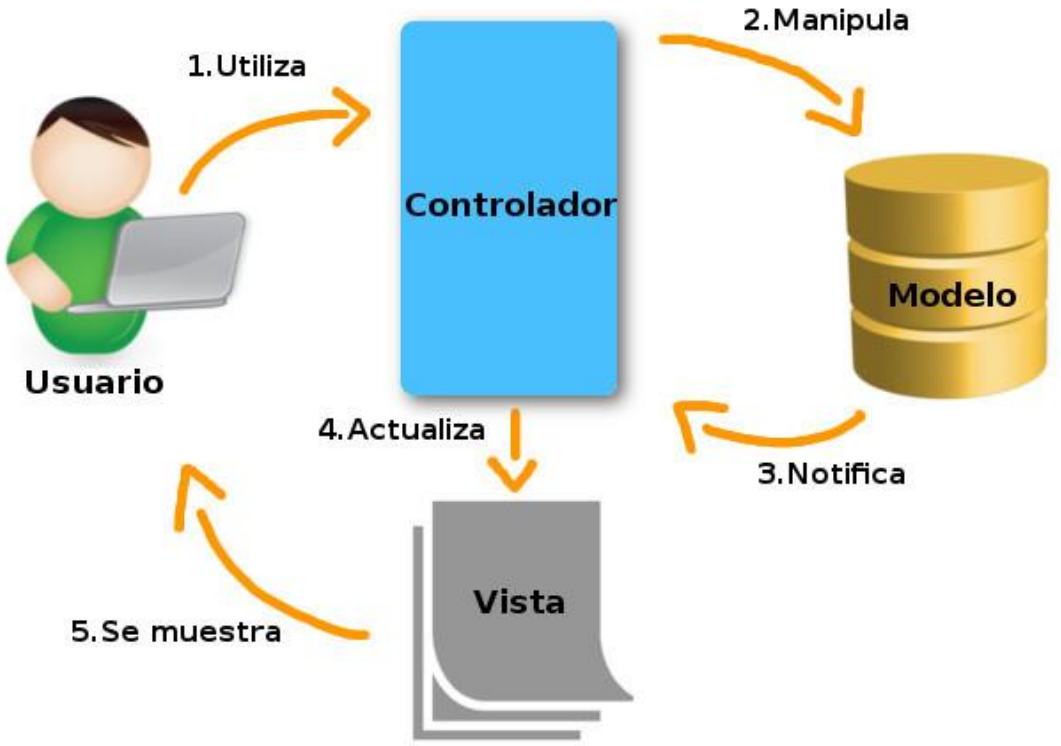


Ilustración 7. Arquitectura MVC

Haciendo una breve mención a los servlets hay que decir que éstos son programas que se ejecutan en el extremo servidor, introducidos para extender de forma dinámica la funcionalidad de un servidor web. La mayoría de las aplicaciones web desarrolladas en Java se basan en servlets.

- En la práctica un servlet no es más que una clase java en el servidor, a la cual las aplicaciones accederán a través de un esquema de petición-respuesta.
- El servidor necesita un contenedor especial para servlet (p.e. Apache Tomcat que es el que se ha usado para esta aplicación).
- Los servlets usados en esta aplicación son HttpServlets que como su propio nombre indica están basados en el protocolo Http.

2.2.2.1 Framework Spring MVC

Volviendo a la arquitectura MVC hay que decir que para poder aplicar este patrón de arquitectura hemos usado el Framework Spring MVC que es una plataforma Java que da soporte al desarrollo de aplicaciones. Hoy en día es de los Framework s de Java más usados.

Al crear este proyecto MVC ya se crean también los ficheros de configuración necesarios para el correcto funcionamiento de la aplicación como son ‘web.xml’, ‘servlet-context.xml’ y ‘pom.xml’ donde se deben añadir las dependencias y librerías necesarias para el proyecto para ‘JDBC y ‘MySQL’ para poder hacer la conexión con la base de datos y poder interactuar con ella.

Este patrón MVC lo que consiste es en que se tiene un controlador (HomeController) que es la clase principal del proyecto y dentro de la cual se tienen todos los métodos donde cada uno captura la URL de cada petición que el usuario realiza llamados ‘RequestMappings’ y el método HTTP de petición llamado ‘RequestMethod’ ya sea GET o POST normalmente.

Cada ‘RequestMapping’ tiene su respectiva clase con su respectivo servlet donde se implementa el código para que realice la acción que deseemos y para terminar se devuelve una vista. En cada vista el usuario interactúa con ella y a través de formularios envía los parámetros a la URL que se desean y de nuevo se vuelve al controlador donde el ‘RequestMapping’ en cuestión recogerá esos determinados parámetros con los que se podrá interactuar con una Base de Datos, realizar una determinada acción y volver a devolver una vista y así sucesivamente.

Mientras tanto el modelo sirve para que dentro de cada servlet se pueda pasar datos a la vista para poder mostrarlos.

Aquí en la Ilustración 9 se puede ver un simple ejemplo de todo esto:

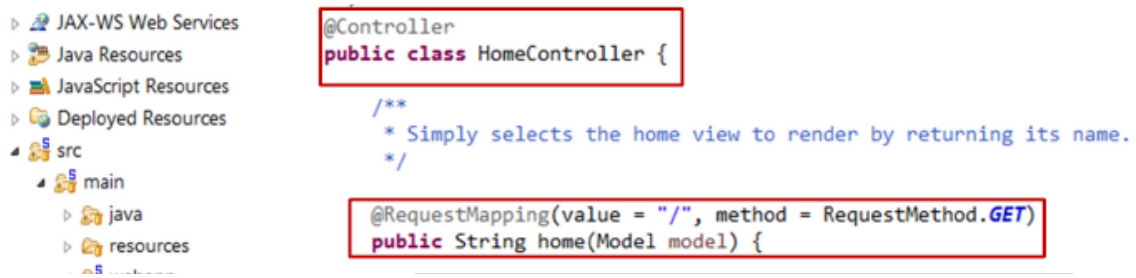


Ilustración 8. Home Controller

Ilustración 9. HomeController

Pero realmente la característica más destacable del Framework Spring es que permite usar técnicas conocidas como inyección de dependencias e inversión de control. Es decir, con Spring, se delega en el Framework la creación de los objetos importantes, y la ejecución se basa en una técnica conocida como Inversión de Control. Implementa un contenedor que se encarga de gestionar las instancias de los objetos del usuario. (En el caso de Spring MVC, se usa un servlet de Spring para este contenedor). También, como se ha mencionado, Spring da soporte a la inyección de dependencia basada tanto en métodos 'setters'[2] como en constructor. Además, también da soporte a una anotación directa en los atributos (@Autowired…).

Al delegar la creación de objetos en Spring, a cambio hay que configurar las dependencias en el Framework . Esto se puede hacer editando archivos .xml o usando las anotaciones.

Los objetos manejados por Spring se denominan 'beans'. Con poner en el código la siguiente anotación (@Autowired) se inyectará, como una instancia de DAO (Data Access Object) un 'bean' de una clase que implemente esa interfaz.

Y para crear ese 'bean' lo que se debe hacer es declararlo en el archivo de configuración 'servlet-context.xml' como se puede ver en la siguiente ilustración:

```
<beans:bean id="beanDao" class="org.tfg.relaciondenecesidades.DAOJdbc" >
<beans:property name="dataSource" ref="dataSource"/>
</beans:bean>
```

Ilustración 10. Creación del bean DAO

Ahora con ese DAO se podrá acceder a los objetos, pero para poder acceder a los objetos de la base de datos se necesitará usar JDBC (Java Database Connectivity) que es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el lenguaje SQL del modelo de base de datos que se utilice. Además, Spring facilita unos elementos de plantilla ('Templates' en inglés) que liberan al programador de tareas como el establecimiento de la conexión, la liberación de recursos y la captura de las posibles excepciones por cada consulta SQL. Cada 'Template' normalmente se usará dentro del DAO y hay que configurarlo con un 'DataSource', que

[2] Los métodos setters son métodos de acceso a los objetos DTO y sirve para establecer un valor inicial a un atributo

tendrá a su vez la configuración de la base de datos. Una forma común de hacerlo es especificando un método ‘setter’ para el ‘DataSource’, que será usado por el Framework con la técnica de inyección de dependencias. Para configurar este ‘DataSource’ se crea el siguiente ‘bean’ en el ‘servlet-context.xml’ con la URL y el puerto donde se aloja nuestra base de datos y sus credenciales:

```
<beans:bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
<beans:property name="driverClassName" value="com.mysql.jdbc.Driver"/>
<beans:property name="url" value="jdbc:mysql://127.0.0.1:3306/tfg"/>
<beans:property name="username" value="root"/>
<beans:property name="password" value=""/>
</beans:bean>
```

Ilustración 11. Credenciales Base de datos

Y ya está todo correctamente configurado para poder implementar la aplicación.

Y aquí en la Ilustración 12 se podría ver un ejemplo de la aplicación ya creada:

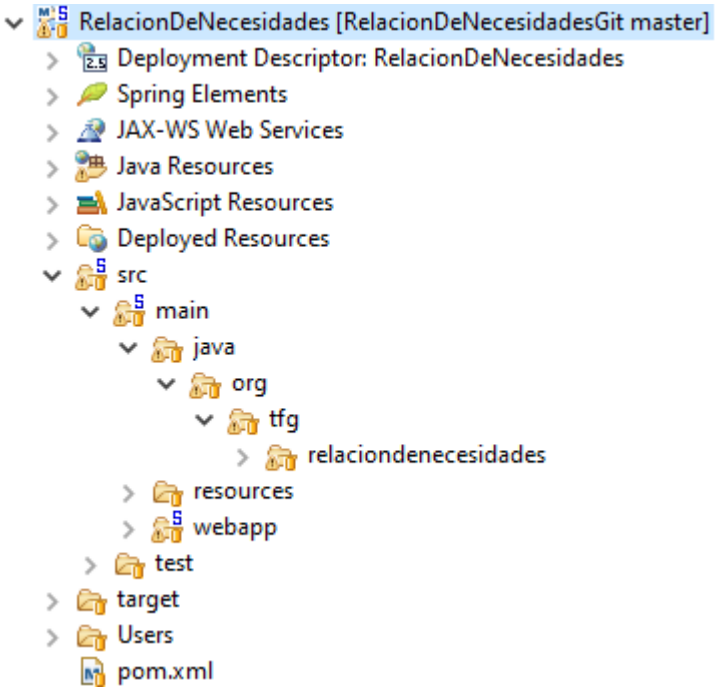


Ilustración 12. Proyecto framework Spring MVC

Y a la que se podrá acceder fácilmente de esta manera ejecutando el proyecto en Eclipse e introduciendo la siguiente URL en el navegador:

Ilustración 13. URL Proyecto

2.2.3 Servidores

En esta sección se comentan los servidores que se han empleado en el proyecto para los contenidos estáticos y los componentes dinámicos

2.2.3.1 Apache Tomcat

Para esta aplicación la parte del servidor se ha usado el Apache Tomcat versión 8.5 *[1]* que es fácilmente integrable con el entorno de desarrollo Eclipse. Una vez se haya configurado cada vez que se ejecute el proyecto se arrancará el servidor y permitirá ver el proyecto en localhost.

2.2.3.2 WampServer

WampServer es un paquete de aplicaciones para Windows que incluye un servidor web Apache y un motor de bases de datos MySQL. También contiene PHPMyAdmin que es el servicio que se ha usado y que sirve para crear y manejar la base de datos *[13]*.

2.2.4 Cliente

En la parte del cliente se han usado las siguientes tecnologías: HTML como lenguaje para crear cada elemento del portal web, CSS para la presentación del documento HTML y JavaScript para la interacción dinámica entre los elementos del documento HTML. En conjunto son las tecnologías básicas para poder diseñar una página web o portal web en nuestro caso. También se usará Bootstrap *[2]* que reúne plantillas de diseño de las tecnologías anteriormente mencionadas y W3Schools *[12]* que es un sitio web que contiene numerosos ejemplos de todas las tecnologías mencionadas anteriormente.

2.2.5 Base de Datos

El sistema gestor de base de datos que se ha usado en este proyecto es MySQL que es relacional él cual es un tipo de base de datos que almacena y proporciona acceso a puntos de datos relacionados entre sí. Las bases de datos relacionales se basan en el modelo relacional, una forma intuitiva y directa de representar datos en tablas. En una base de datos relacional, cada fila de la tabla es un registro con un ID único llamado *clave primaria*. Las columnas de la tabla contienen atributos de los datos, y cada registro generalmente tiene un valor para cada atributo, lo que facilita el establecimiento de las relaciones entre los puntos de datos. Además las tablas están relacionadas entre sí gracias a las *claves foráneas* que son una columna o grupo de columnas de una tabla que contiene valores que coinciden con la clave primaria de otra tabla.

Las bases de datos relacionales se basan en la organización de la información en partes pequeñas que se integran mediante identificadores; a diferencia de las bases de datos no relacionales que, como su nombre lo indica, no tienen un identificador que sirva para relacionar dos o más conjuntos de datos. Por ello se ha elegido usar una base de datos relacional a una no relacional.

La base de datos está formada por las siguientes tablas: PDI, Unidad de gasto, Responsable de gestión, Relación de necesidades, Pedido y Registro. A continuación se procederá a mostrar las distintas tablas con sus respectivos atributos.

- PDI:
    - Id (INT): Clave primaria
    - Nombre (VARCHAR)
    - Apellidos (VARCHAR)
    - Departamento (VARCHAR)
    - Teléfono (INT)
    - Email (VARCHAR)
    - Edificio (VARCHAR)
    - Dependencia (VARCHAR)
    - Centro (VARCHAR)
    - Área (VARCHAR)
    - Usuario (VARCHAR)
    - Contraseña (VARCHAR)

- Responsable de Unidad de gasto:
    - Id (INT): Clave primaria
    - Código (INT)
    - Descripción (VARCHAR)
    - Presupuesto (INT):
    - Id PDI (INT): Clave foránea que se relaciona con el Id de la tabla ‘PDI’.

- Responsable de Gestión
    - Id (INT): Clave primaria
    - Nombre (VARCHAR)
    - Apellidos (VARCHAR)
    - Servicio (VARCHAR)
    - Teléfono (INT)
    - Email (VARCHAR)
    - Edificio (VARCHAR)
    - Centro (VARCHAR)
    - Administrador (TINYINT)
    - Usuario (VARCHAR)
    - Contraseña (VARCHAR)

- Relación de Necesidades
    - Id (INT): Clave primaria
    - Título (VARCHAR)
    - Fecha (DATE)
    - Expediente (VARCHAR)
    - Estado (VARCHAR)
    - Id PDI (INT)
    - Id Responsable de Gestión (INT)
    - Id Unidad de Gasto (INT)
    - Proveedor (VARCHAR)
    - Persona Contacto (VARCHAR)
    - Teléfono Contacto (INT)
    - Fecha Pedido (DATE)

- Pedido
    - Id (INT): Clave primaria
    - Artículo (VARCHAR)
    - Cantidad (INT)
    - Precio (INT)
    - Descripción (VARCHAR)
    - Id Relación de necesidades (INT)

- Registro
    - Id (INT): Clave primaria
    - Id PDI (INT)
    - Nombre PDI
    - Fecha Inicio Sesión (VARCHAR)
    - Id Relación (INT)
    - Título Relación (VARCHAR)
    - Fecha Envío Relación (VARCHAR)
    - Fecha Validación Relación (VARCHAR)
    - Id Unidad de Gasto (INT)
    - Nombre Unidad de Gasto (VARCHAR)
    - Id Responsable de Gestión (INT)
    - Nombre Responsable de Gestión (VARCHAR)
    - Fecha Tramitación Relación (VARCHAR)

Y el modelo Entidad-Relación de esta base de datos sería el que se muestra en la Ilustración 14:

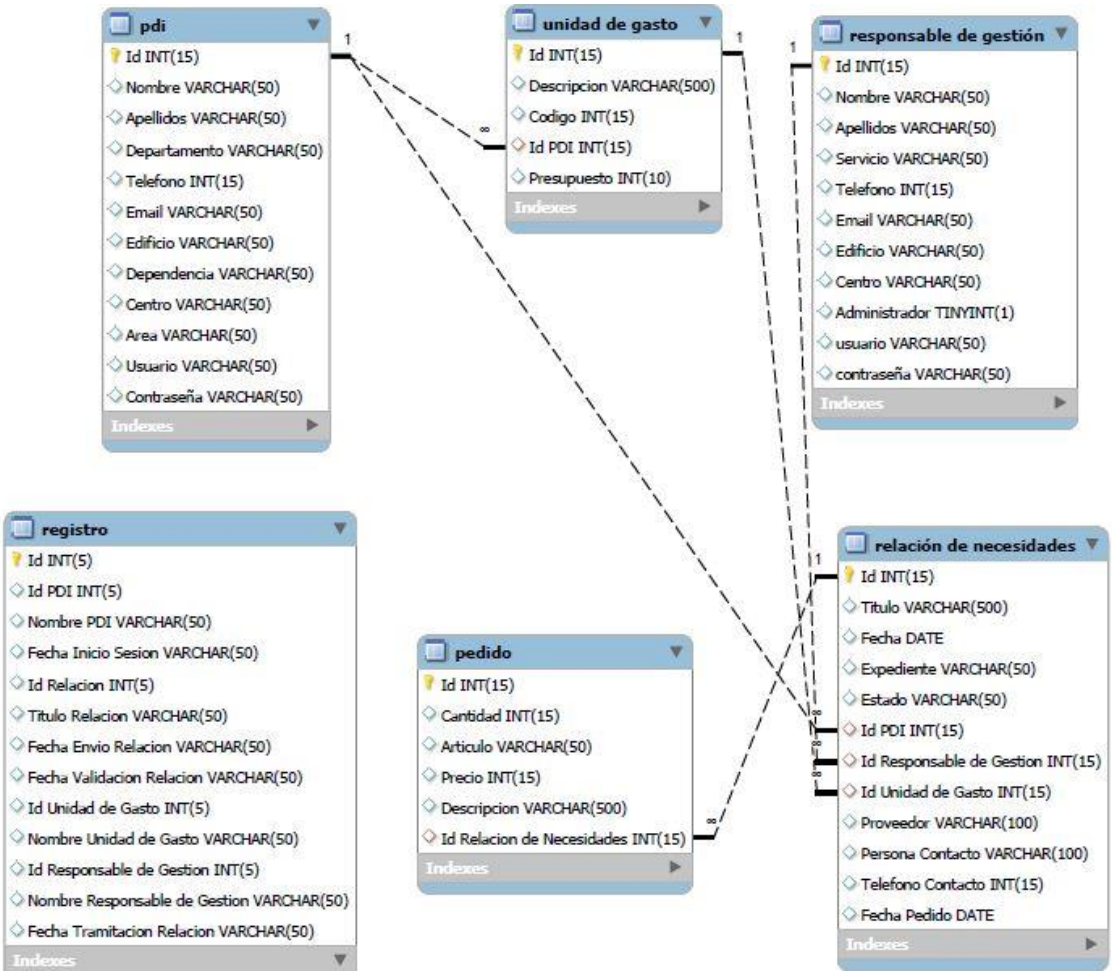


Ilustración 14. Modelo Entidad-Relación

2.2.6 Desarrollo de la aplicación

En esta sección se va a explicar cómo se ha desarrollado la aplicación tanto sus vistas como el controlador como las distintas clases java.

2.2.6.1 Vistas

En este apartado se van a explicar las diferentes vistas de la plataforma

2.2.6.1.1 Iniciosesion.jsp

Esta es la vista inicial de la Plataforma Web donde se tiene un botón que cuando se pulsa aparece un cuadro de diálogo modal el cual lleva un formulario para introducir el usuario y la contraseña. Los parámetros de este formulario serán recogidos en el método “Acceso” del “HomeController.java”.

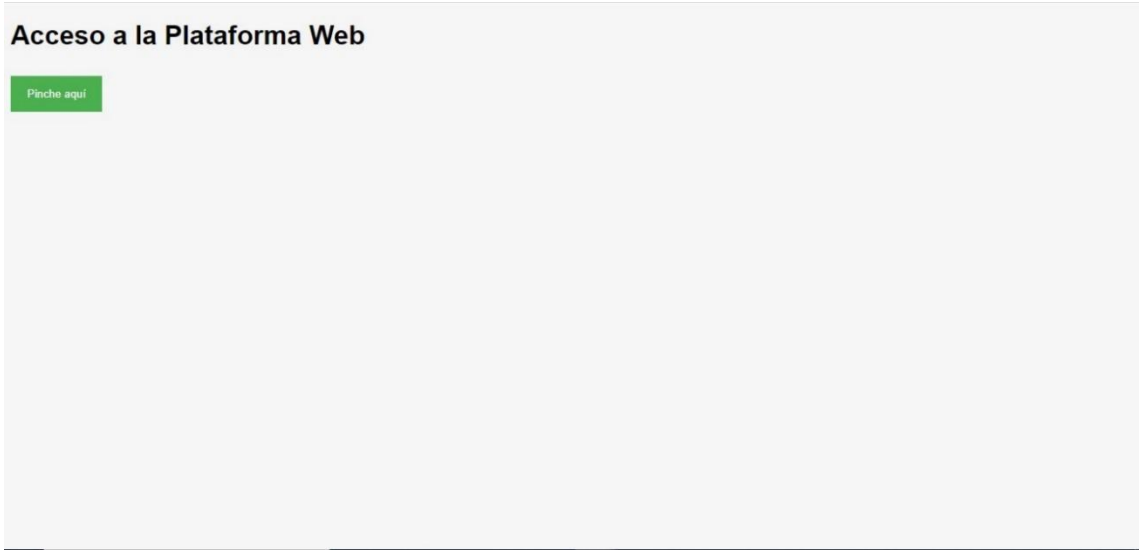


Ilustración 15. Iniciosesion.jsp

Ilustración 16. Modal Usuario y Contraseña

2.2.6.1.2 Contraseñaincorrecta.jsp

Si se introduce un usuario o contraseña incorrecta esta vista muestra una alerta diciendo “usuario o contraseña incorrectos”.

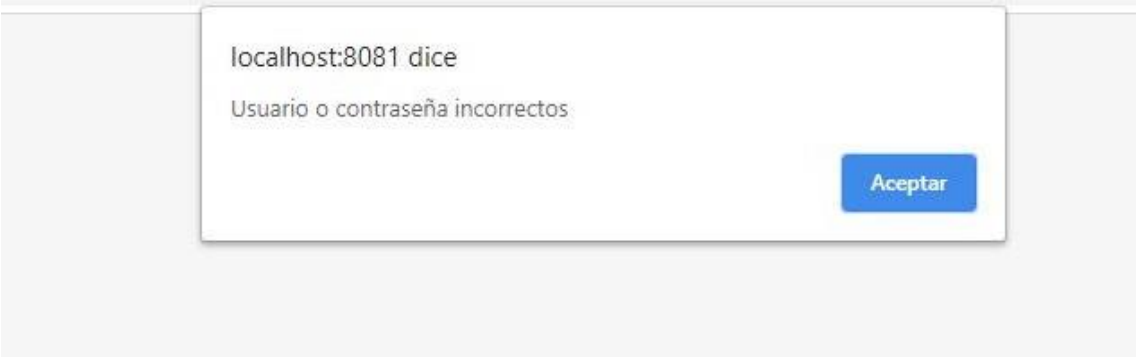


Ilustración 17. Contraseñaincorrecta.jsp

2.2.6.1.3 PlataformaPDI.jsp

Esta es la vista principal de la plataforma si se accede como PDI. A la izquierda se encuentra un menú de navegación con varios contenedores los cuáles cada uno contiene los distintos estados que puede tener una relación de necesidades (Pendientes de validación, Validadas, Denegadas (Pendientes de Modificación), Borradores y Eliminadas) y para crear una Nueva Relación de Necesidades.

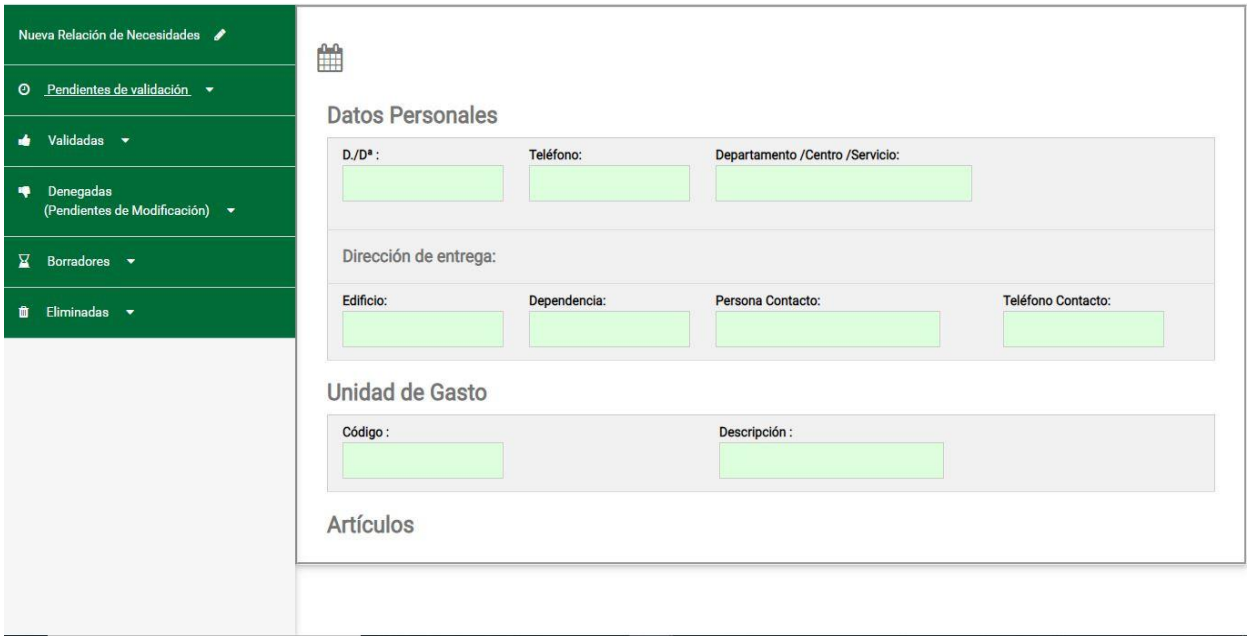


Ilustración 18. PlataformaPDI.jsp

Si se pincha en “Nueva Relación de Necesidades” se nos aparecerá un modal con varias pestañas la cual cada una de esas pestañas son los datos del PDI que ha iniciado sesión, otra pestaña para seleccionar la unidad de gasto y otra pestaña para poner el título y seleccionar la fecha de la relación. Todos estos parámetros están recogidos en un formulario y que luego posteriormente en el método “NuevaRelación” del “HomeController.java” se recogerán e interactuará con ellos como ya se explicará después cuando se explican los métodos del “HomeController.java”.

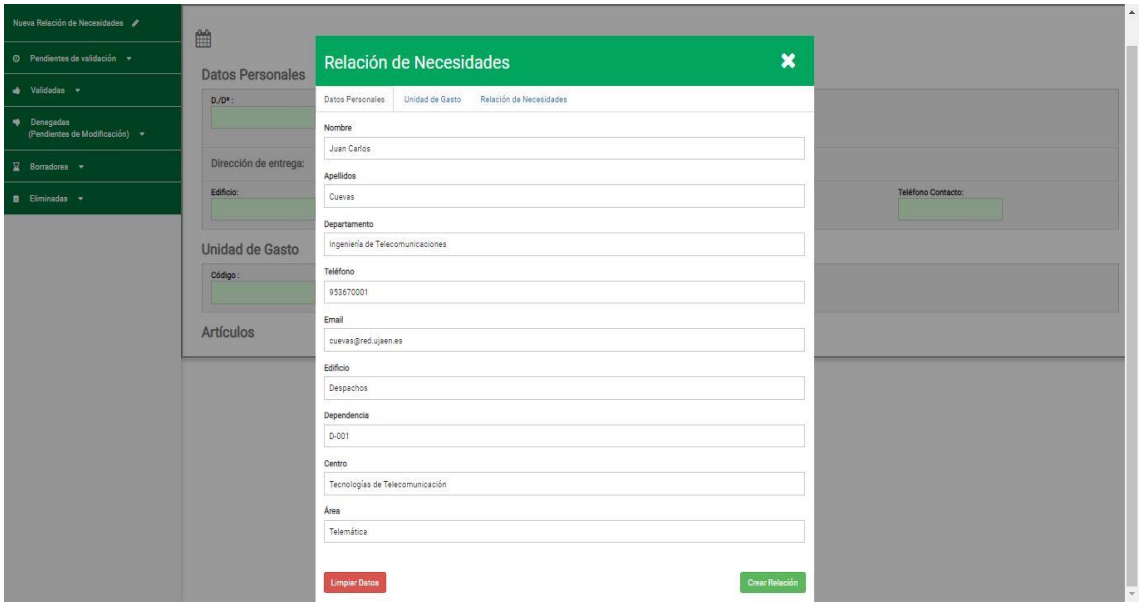


Ilustración 19. Modal Nueva Relación

Ilustración 20. Modal Nueva Relación Pestaña 1

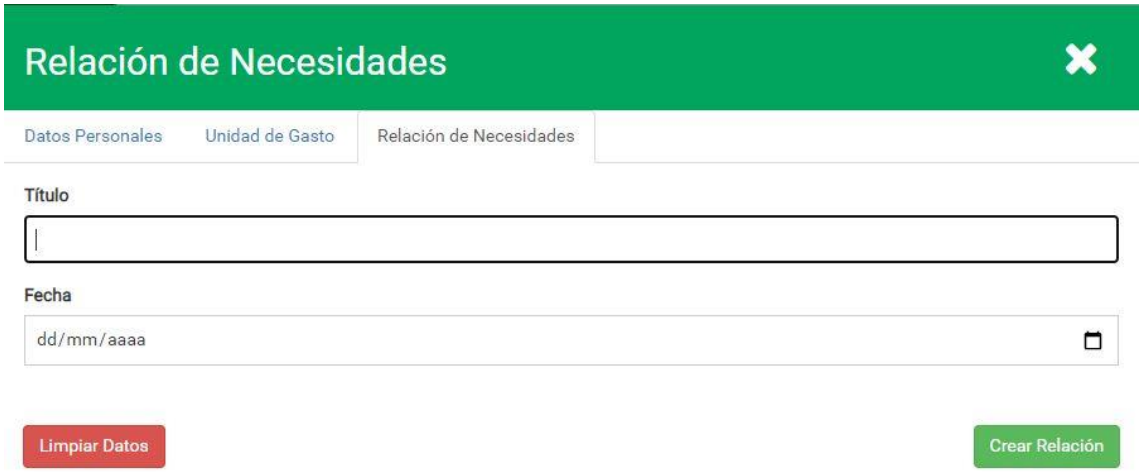


Ilustración 21. Modal Nueva Relación Pestaña 2

Cuando se crea esa relación se llama a la función "alertarelacioncreada" para que muestre una alerta de que la relación ha sido creada correctamente y que ahora se encuentra en "Borradores".

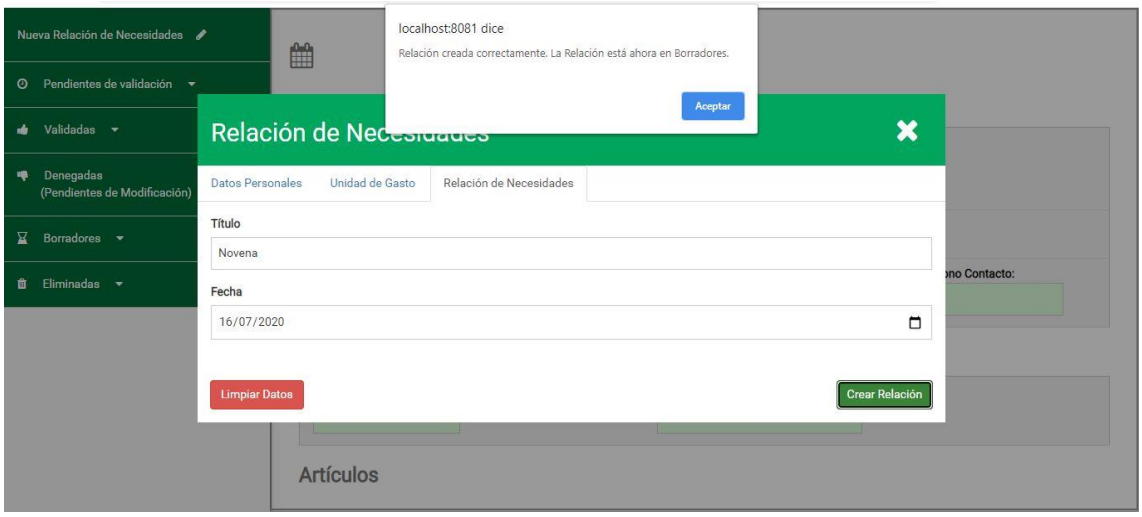


Ilustración 22. Alerta Relación creada

Los distintos submenús del menú de navegación que contienen las distintas relaciones para recorrerlas se usa el bucle "c:forEach" y para mostrar por pantalla el título y la fecha se usa el "c:out" . Con un "onclick" sobre la relación se llama a la función "enviaformulario" creado en esta misma vista y se envía el parámetro id de la relación para posteriormente recogerlo en el método correspondiente del "HomeController.java" y buscar

en la base de datos la relación que se ha seleccionado y enviarla de nuevo a la vista a través del modelo para poder mostrarla.

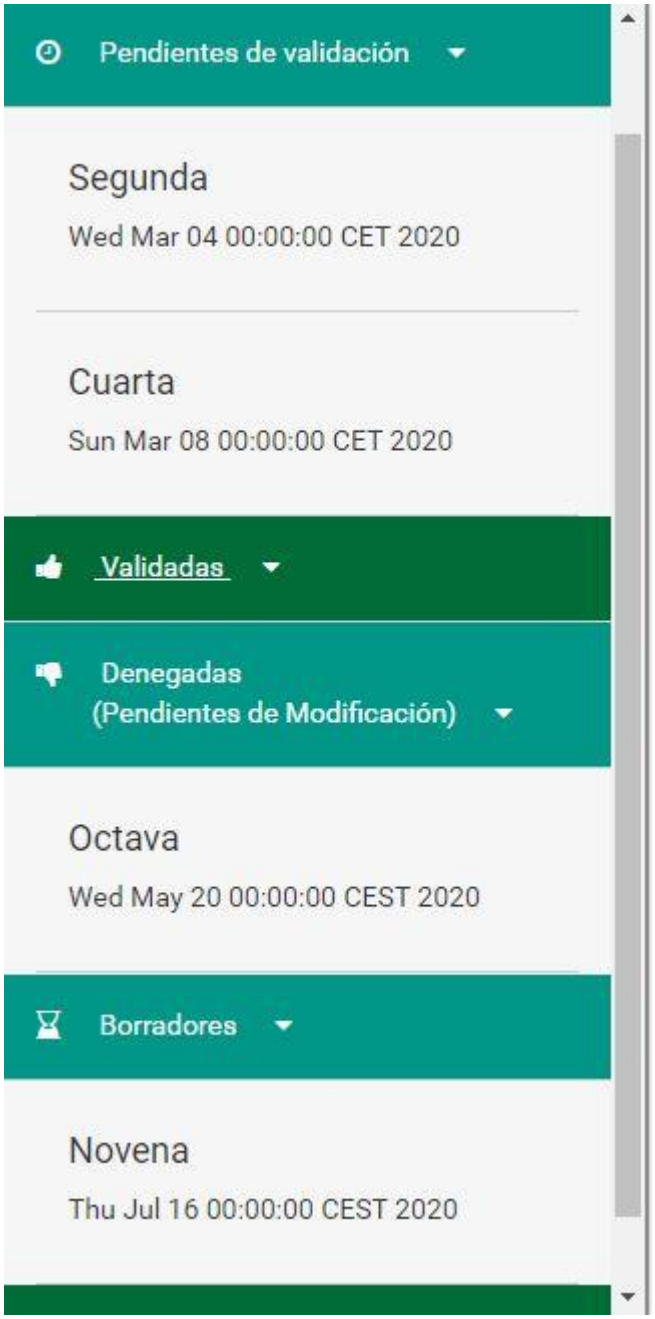


Ilustración 23. Menú de navegación con distintas relaciones

2.2.6.1.4 PlataformaPDIEditable.jsp

Cuando se selecciona una relación aparecerá esta vista representada en la Ilustración 24 donde se muestra la relación y con contenedores y tablas se muestran los datos personales del PDI, la unidad de gasto seleccionada y los artículos en el caso de que hubiera.

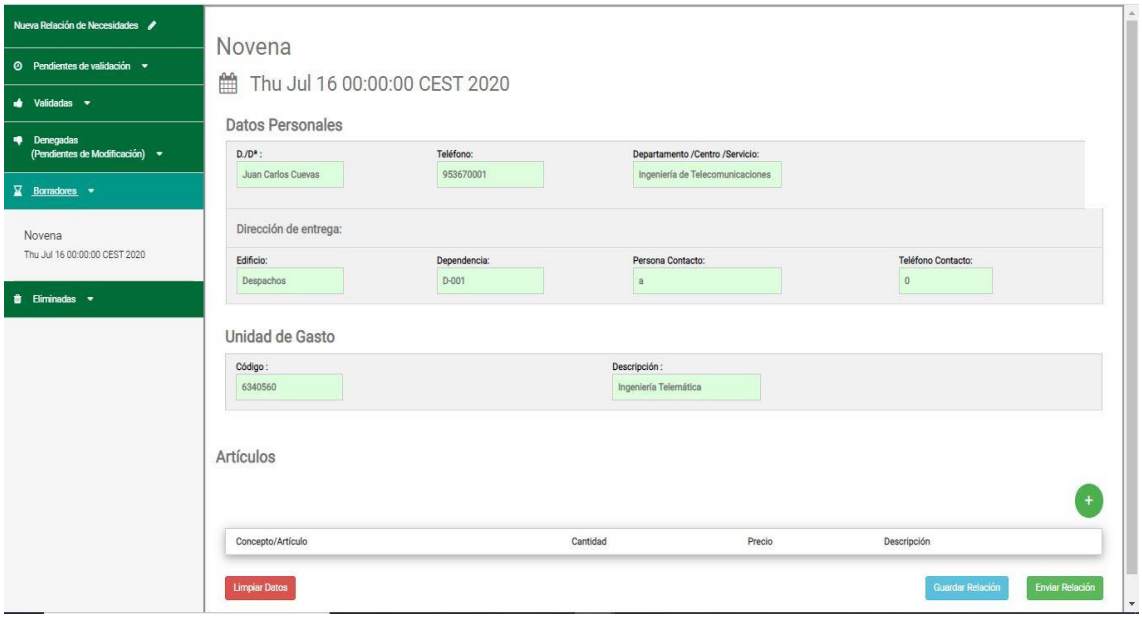


Ilustración 24. PlataformaEditable.jsp

Para añadir estos artículos se pulsa el botón verde con un "+" y se creará una fila en la tabla con las columnas Concepto/Artículo, Cantidad, Precio y Descripción que será las entradas ("inputs") que se deben rellenar del formulario. Esto es posible gracias a la función "añadirproductos" en la cual primero obtiene el elemento al que se está refiriendo con el método "document.getElementbyId" y con los métodos "insertRow" e "insertCell" crea una fila y celdas respectivamente. Después se crea una variable llamada "campo" para cada celda y con "document.createElement" se crea un elemento de tipo "input". Se accede a los atributos de esa variable llamada `'campo'` y se le pone el tipo ya sea texto o número y con".setAttribute" se le pone el nombre de la variable `"name"` que será el nombre del parámetro a recoger luego y por ejemplo para la primera celda sería (`"name", "articulo"+id"`). Si también se quiere cambiar otros atributos por ejemplo el tamaño del input también sería con ".setAttribute" y por ejemplo (`"size", "30"`). La última celda será un input de tipo botón para borrar la fila y para ello con el "onclick" se llama a una función la cual identifica la fila que es con "this.parentNode.parentNode" y elimina esa fila con ".removeChild". Para terminar se añaden todos estos inputs a las celdas con ".appendChild" y se incrementa la variable id para la siguiente iteración.

Ilustración 25. Añadir Artículos

Si se guarda la relación sin enviarla se podrá volver a acceder a ella desde Borradores y poder ver los artículos añadidos anteriormente y añadir nuevos.

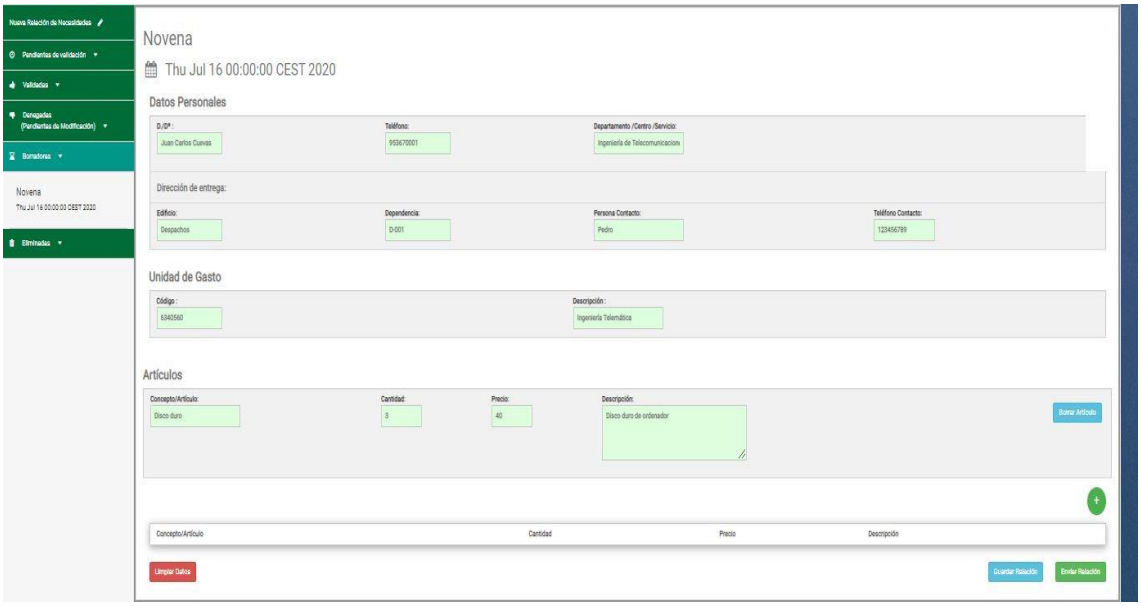


Ilustración 26. Relación Guardada

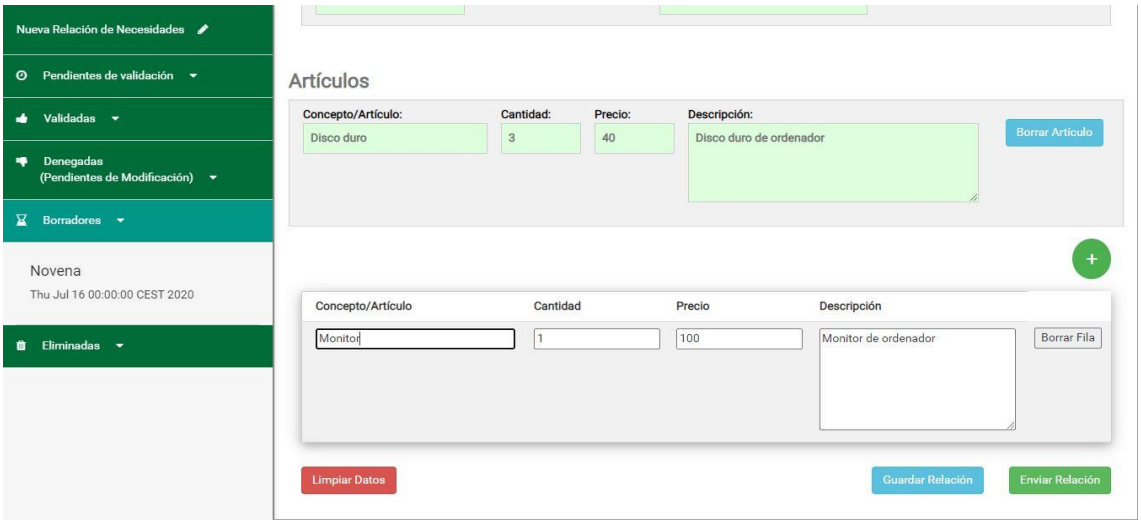


Ilustración 27. Añadir Nuevo Artículo

Y una vez se envía la relación se puede observar cómo se va a "Pendientes de Validación".

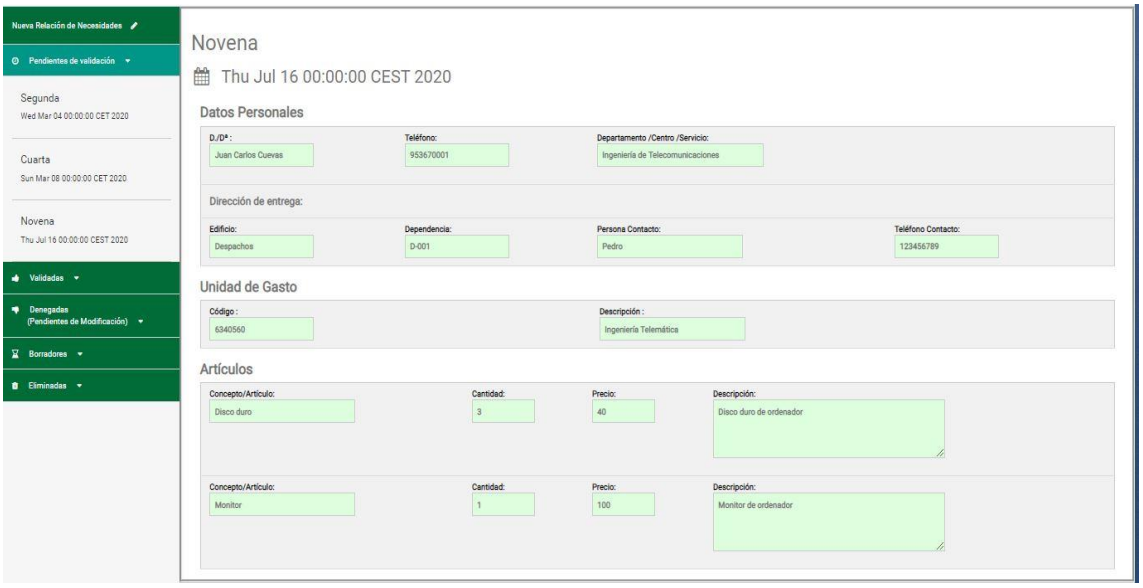


Ilustración 28. Relación Enviada

2.2.6.1.5 PlataformaPDIDenegada.jsp

En esta vista se visualizan las relaciones denegadas por el responsable de unidad de gasto y se pueden modificar para volver a enviarlas, guardarlas y eliminarlas.

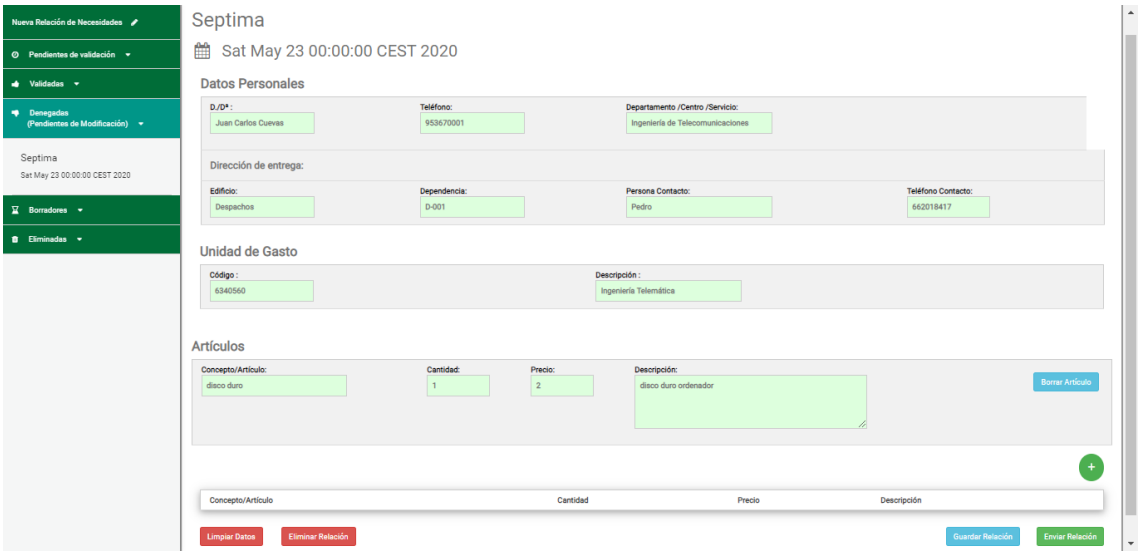


Ilustración 29. PlataformaPDIDenegada.jsp

2.2.6.1.6 PlataformaUnidadGasto.jsp

Esta vista es igual que la vista "PlataformaPDI" con la diferencia de que se añade otro menú de navegación para las relaciones que tenga que validar o denegar este responsable de unidad de gasto. Pero también puede crear una relación como PDI que es y observar el estado de sus relaciones.

Ilustración 30. PlataformaUnidadGasto.jsp

2.2.6.1.7 PlataformaUnidadGastoEditable.jsp

Esta vista es igual que la vista "PlataformaPDIEditable" con la misma diferencia que en la vista anterior, es decir, tiene dos menús de navegación (como PDI y como responsable de unidad de gasto).

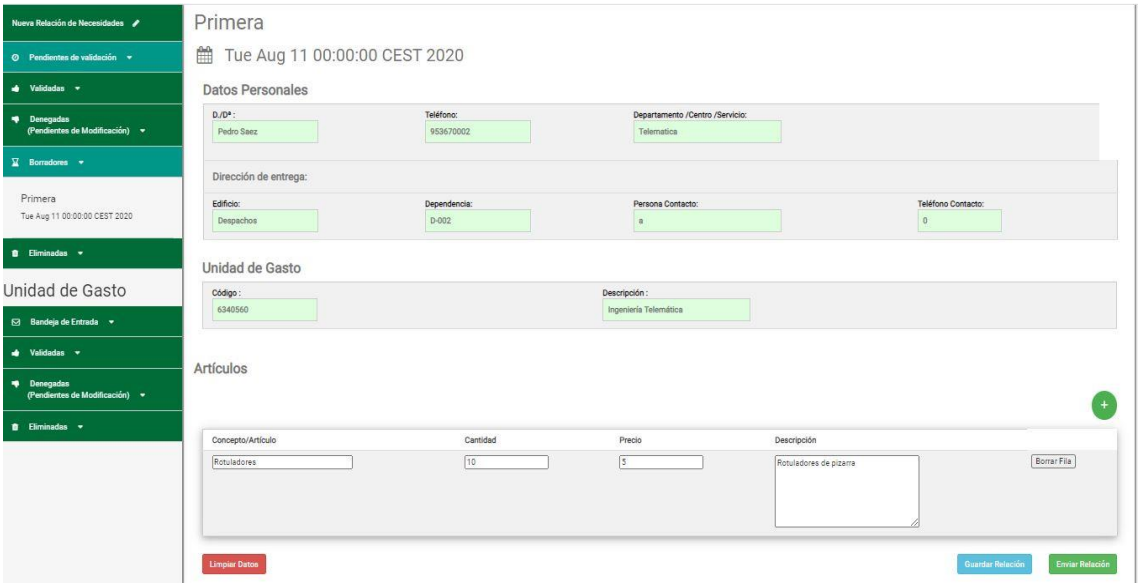


Ilustración 31. PlataformaUnidadGastoEditable.jsp

2.2.6.1.8 PlataformaUnidadGastoValidable.jsp

Aquí el responsable de unidad de gasto tiene dos opciones, validar o denegar la relación que está pendiente de validación y por lo tanto en su bandeja de entrada. Si la valida ira a Validadas y por tanto pasará al responsable de gestión para que comience la gestión y si la deniega a Denegadas para que el PDI la modifique.

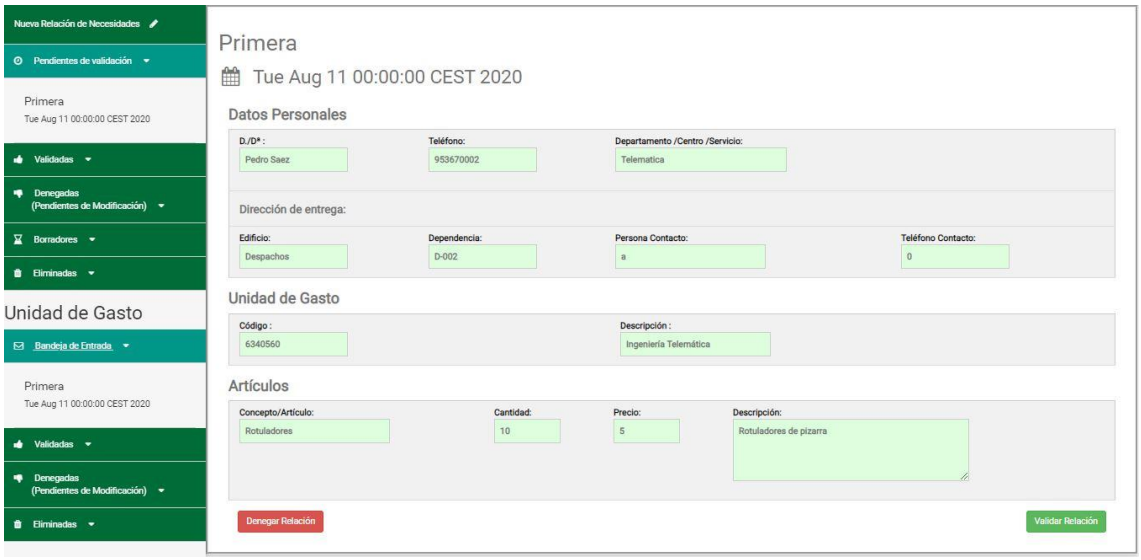


Ilustración 32. PlataformaUnidadGastoValidable.jsp

2.2.6.1.9 PlataformaResponsableGestion.jsp

Esta vista representada en la Ilustración 33 es similar a las anteriores donde se puede observar que también tiene un menú de navegación con los distintos estados que puede tener un responsable de gestión de la relación y un div principal para mostrar los datos de la relación seleccionada.

Ilustración 33. PlataformaResponsableGestión.jsp

2.2.6.1.10 PlataformaResponsableGestionTramitable.jsp

Aquí después de seleccionar una relación de la bandeja de entrada para tramitarla se muestra y pulsando el botón de "Tramitar Relación" pasará a estar "En trámite".

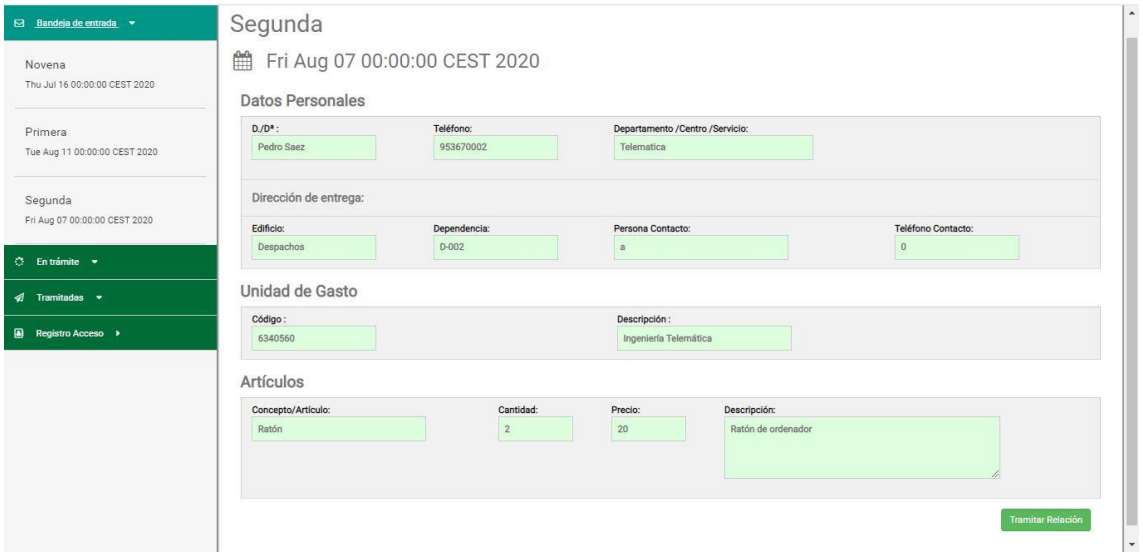


Ilustración 34. PlataformaResponsableGestionTramitable.jsp

2.2.6.1.11 PlataformaResponsableGestiónEnTramite.jsp

En esta vista se muestra la relación seleccionada a tramitar con todos sus datos y donde se deben rellenar los parámetros de tramitación que son: Expediente, Proveedor y Fecha Pedido ya que los demás parámetros vienen rellenos. Después en su método correspondiente recogerá estos parámetros y se los añadirá a la relación adecuada mediante los métodos DAO.

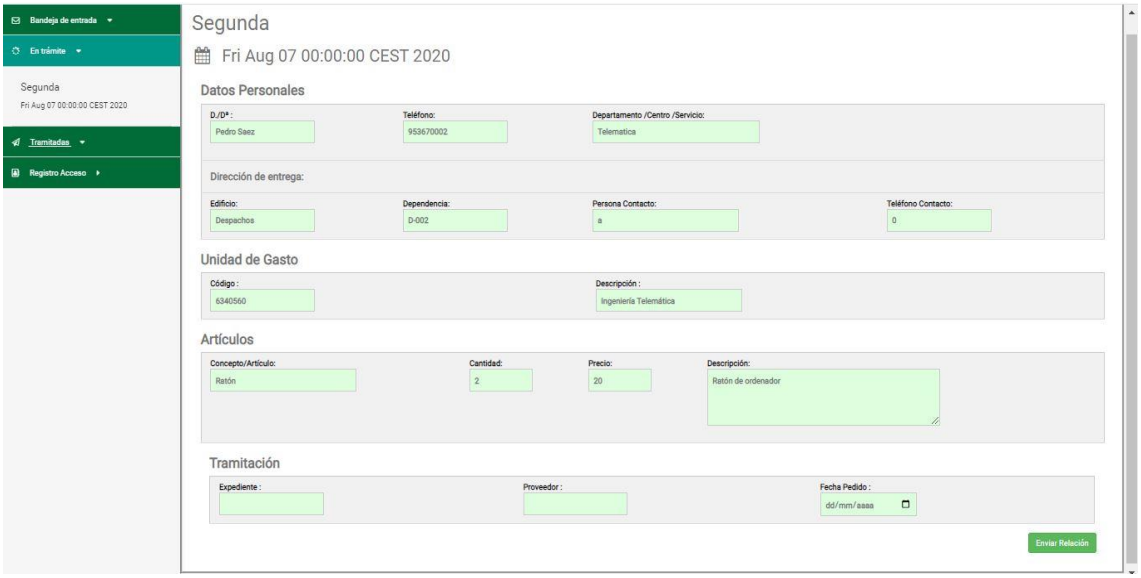


Ilustración 35. PlataformaResponsableGestiónEnTramite.jsp

2.2.6.1.12 PlataformaResponsableGestiónTramitada.jsp

En esta vista se muestra la relación que se ha seleccionado del apartado “Tramitadas” con todos sus datos.

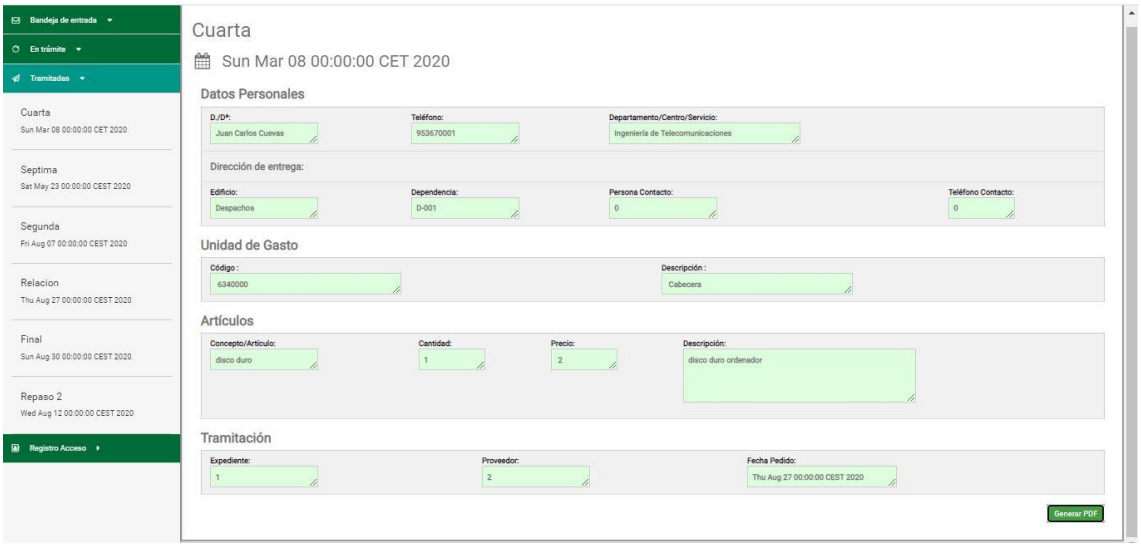


Ilustración 36. PlataformaResponsableGestiónTramitada.jsp

Si se pulsa sobre el botón de "Generar PDF" se llamará a la función "generarPDF()" que usando la librería "jspdf" generará un PDF. Para ello se crea un objeto jsPDF y a este objeto usando métodos como '.text' se le podrá añadir texto a la página con las medidas deseadas o con '.addPage' se añade una nueva página. También ha de recogerse el elemento HTML que se quiere introducir en el PDF con "`$(nombre del elemento HTML).html()`" haciendo uso de 'Jquery'.

2.2.6.1.13 PlataformaResponsableGestiónRegistro.jsp

En esta vista se muestra el registro de acceso a la aplicación en una tabla con sus filas y sus celdas y que imprime los valores que vengan del modelo.

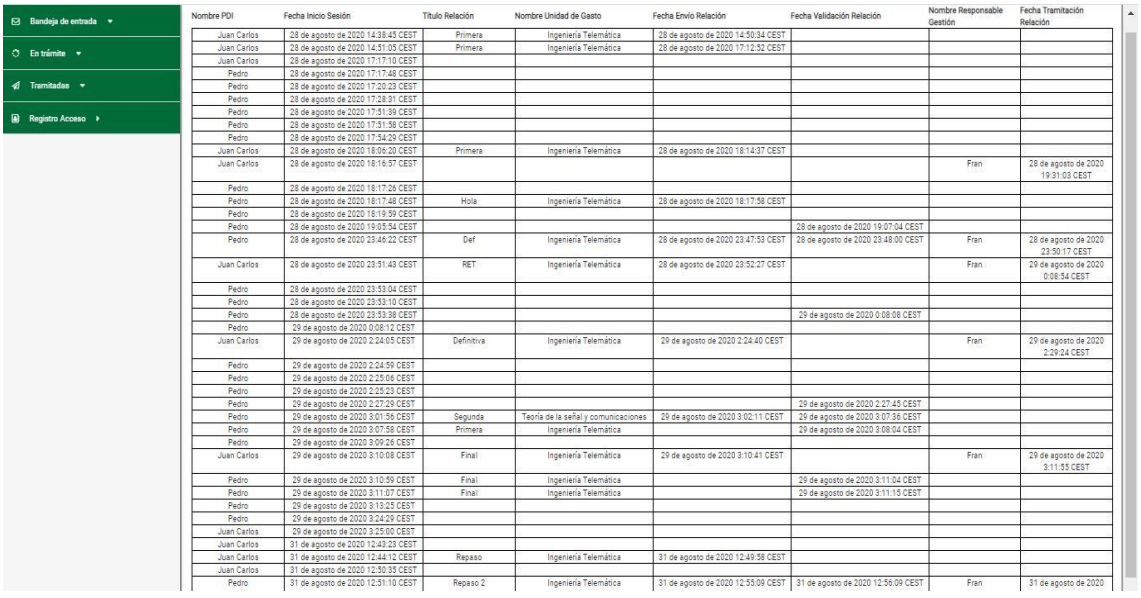


Ilustración 37. PlataformaResponsableGestiónRegistro.jsp

Pedro	31 de agosto de 2020 12:51:10 CEST	Repaso 2	Ingeniería Telemática	31 de agosto de 2020 12:55:09 CEST	31 de agosto de 2020 12:56:09 CEST	Fran	31 de agosto de 2020 21:04:52 CEST

Ilustración 38. Fila Registro Acceso Usuarios

2.2.6.2 Clases Java

En este apartado se explicarán las distintas clases java del proyecto.

2.2.6.2.1 DAOJdbc.java

En esta clase se programan los distintos métodos para interactuar con la base de datos como puede ser agregar un usuario, modificarlo, eliminarlo, leer sus datos etc. Estos métodos reciben los parámetros que se le pasa desde el controlador cuando se les invoca e interactúan con la base de datos con una sentencia SQL.

2.2.6.2.2 HomeController.java

Esta clase es la principal de la aplicación, el controlador. Aquí es donde se programan todos los 'RequestMapping' dónde recoge cada URL posible de la aplicación y con cada 'RequestMapping' está asociado un método en el que se implementa la función que se desea realizar. La estructura de cada método es más o menos parecida y consiste en recoger los parámetros enviados en un formulario en la vista JSP, usar las clases DTO para crear los objetos, invocar los métodos DAO para realizar la función que desees hacer con esos parámetros y la base de datos y al final devolver una vista JSP. Los diferentes métodos que hay son:

- Home

  Este método primero busca si hay alguna cookie activa mediante el parámetro "emailAddress" que es el email del usuario y el identificador de la cookie para mantener la sesión activada o iniciar una nueva.

  Luego a través del DAO y el método "buscaPDI" extrae el id del PDI para saber de qué PDI se trata. Una vez se ha conseguido esto va buscando las listas de relaciones que haya en la base de datos y con el id del PDI que haya iniciado sesión, para sólo mostrarle las suyas. Las recorre con un bucle "for" y se las pasa al modelo para luego en la vista mostrarlas. Para hacer esto primero se crea un 'ArrayList' del tipo 'RelacionDTO' para crear la lista de relaciones y guardar las listas que se vayan encontrando en ese 'Array'. Se

invoca el método “buscaRelacionporEstado” para buscar las relaciones según el estado que se le pase ya sea Borradores, Pendientes, Validadas, Denegadas y Eliminadas y se recorre con un iterador y se le pasa al modelo para poder mostrarlas en la vista.

Después hay que buscar en la base de datos las distintas unidades de gasto que haya para pasárselas al modelo también. Para ello se crea un ‘ArrayList’ de ‘UnidaddegastoDTO’ y a través del método “leeUnidaddegasto” se busca estas unidades de gasto y ya se le pasa al modelo.

Y para terminar existe la opción de mostrar diferentes vistas, “iniciosesion.jsp” en el caso de que no haya sesión activada ya que el “emailAddress” esté vacío y entonces te muestra esta vista donde necesitas introducir las credenciales y la vista correspondiente de cada usuario si está la sesión activa y entonces se cogen los datos del usuario de la sesión y se le pasa al modelo para poder utilizarlos en la vista. Si es PDI devolverá “PlataformaPDI”, si es Responsable de unidad de gasto devolverá “PlataformaUnidadGasto” y si es Responsable de gestión devolverá “PlataformaResponsableGestión”.

- Acceso

Primero recoge los parámetros del usuario que son el email y la contraseña introducidos en el formulario de “iniciosesion.jsp” y con el email busca en la base de datos que PDI es y recoge su id. Después está el código para recorrer las listas de relaciones y el código para buscar las distintas unidades de gasto explicados ambos en el método anterior.

Con varios “if” se comprueba si es PDI, Responsable de unidad de gasto o Responsable de gestión o si la contraseña es incorrecta y dependiendo de la condición que se cumpla devolverá una vista u otra. Si es PDI devolverá “PlataformaPDI”, si es Responsable de unidad de gasto devolverá “PlataformaUnidadGasto”, si es Responsable de gestión devolverá “PlataformaResponsableGestión” y si la contraseña es incorrecta devolverá “contraseñaincorrecta”.

Además, si es PDI se implementará un código para hacer el registro de acceso que consiste en usar el método “DateFormat.getDateTimeInstance”

para obtener la fecha actual en la que se accede a la página. También con el método “dao.buscaPDIporId” se busca el PDI que ha iniciado sesión y ya obtengo sus datos como el nombre. Se crea un objeto registro del tipo RegistroDTO y con los métodos “set” se le ponen los datos recogidos anteriormente y ya con el método “dao.insertaRegistro” se inserta este registro en la base de datos.

Después se crea la cookie del usuario para poder mantener la sesión activa y se pasa al modelo los datos de dicho usuario.

- NuevaRelacion

Como en los anteriores métodos primero se busca si hay cookie de sesión, se busca el id del PDI y se busca las relaciones y unidades de gasto existentes en la base de datos.

Si no hay cookie devolverá la vista “iniciosesion” para que introduzca sus credenciales y si hay cookie devolverá “PlataformaPDI”.

Y ahora recoge los parámetros introducidos en el formulario de “NuevaRelación” de la vista como son el título, el id de la unidad de gasto y la fecha. Los datos del PDI ya los teníamos de antes y el estado se le pone “Borradores”. Se crea una ‘RelaciónDTO’ con estos datos e introduce esta nueva relación en la base de datos a través del método “insertaRelacion”.

Luego se vuelve a usar el código para recorrer las relaciones existentes en las bases de datos para mostrar esta nueva relación en la vista.

- SeleccionoRelacion

Se repite todo el proceso de buscar la cookie de sesión o sino devolver la vista de “iniciosesion.jsp”, buscar el id del PDI, las relaciones y las unidades de gasto.

Después se recoge el parámetro “idrelacion” para saber qué relación ha seleccionado el usuario y se busca en la base de datos mediante el método “buscaRelacionporId”. Una vez la tenga se le pasa al modelo todos los datos de la relación seleccionada para poder mostrarla en la vista “PlataformaPDI”.

- SeleccionoRelacionEditable

Se repite todo el proceso del método anterior. Después se recoge también los parámetros de dirección de entrega que son "personacontacto" y "telefonocontacto" y se insertan en la base de datos mediante el método "modificaRelacionString" para el parámetro "personacontacto" y "modificaRelacionint" para el parámetro "telefonocontacto".

Para terminar, se vuelve a poner el código que busca las relaciones existentes para actualizar el portal web y devuelve la vista "PlataformaPDIEditable.jsp".

- SeleccionoRelacionDenegada

  Es igual que el método "SeleccionoRelaciónPDI" pero en vez de devolver la vista "PlataformaPDI" devuelve "PlataformaPDIDenegada".

- EnviarRelacion

  Todo el proceso igual que el método anterior y además se recogen los parámetros del pedido y para ello se crea un 'ArrayList' de 'PedidoDTO' y con un bucle "for" se va recorriendo cada artículo introducido y recogiendo los parámetros de cada artículo es decir el artículo en sí, su descripción, cantidad, precio e id de la relación a la que pertenece y se introduce en un 'nuevopedido' que es de tipo 'PedidoDTO' y se añade a la lista de pedidos creada anteriormente. Para introducirlo en la base de datos se invoca al método "insertaPedido" y con el método "cambiaEstadoRelación" se le cambia el estado a "Pendientes". Si por algún caso el artículo viene vacío con un iterador se recorre ese artículo y se borra. Todo ello controlado por un 'try-catch' por si hubiera algún error.

  Después se pone el código para llevar el registro de acceso y transacciones donde se recoge la fecha en la que acaba de enviar la relación con el método "DateFormat.getDateTimeInstance". Se recoge también el id del relación para saber de qué relación se trata y obtener mediante el método "dao.BuscaRelacionporId" su título, unidad de gasto etc. También con el método "dao.BuscaUnidaddegastoporId" se busca esa unidad de gasto y su nombre. Por último se llama al método "dao.InsertaRegistroporId" y se le

pasa los parámetros obtenidos anteriormente, es decir, “fechaenviorelacion”, “idrelacion”, “titulorelacion”, “idunidaddegasto”, “nombreunidaddegasto” y el “idpdi” para saber a qué PDI se está refiriendo e insertarlo en esa fila.

- GuardarRelacion

  Este método es exactamente igual que el anterior sólo que aquí no se le cambia el estado a la relación ya que sólo se quiere introducir nuevos datos y guardarla, no enviarla por lo que sigue en el estado de “Borradores”.

- SeleccionoRelacionUnidadGasto

  Es igual que el método “SeleccionoRelación” pero en vez de devolver la vista “PlataformaPDI” devuelve “PlataformaUnidadGasto”

- SeleccionoRelacionUnidadGastoValidable

  Es igual que el método “SeleccionoRelaciónUnidadGasto” pero en vez de devolver la vista “PlataformaUnidadGasto” devuelve “PlataformaUnidadGastoValidable”.

- SeleccionoRelacionUnidadGastoDenegada

  Es igual que el método “SeleccionoRelaciónPDIDenegada” pero en vez de devolver la vista “PlataformaPDIDenegada” devuelve “PlataformaUnidadGastoDenegada”.

- ValidarRelacion

  Se repite todo el proceso de todos los métodos de buscar la cookie de sesión y sino devolver la vista de “iniciosesion”, buscar el id del PDI, las relaciones y las unidades de gasto existentes y se le pasa al modelo para mostrarlo en la vista.

  Luego se recoge el parámetro “idrelacion” y se invoca el método “cambiaEstadoRelacion” para cambiarle el estado a “Validadas” y se le pasa como parámetro ese “idrelacion”.

Después se pone el código para llevar el registro de acceso y transacciones y se hace lo mismo que se explicó en el método "EnviarRelación" sólo que ahora la fecha recogida es en la que se valida la relación pero los demás parámetros son los mismos.

Y se devuelve como vista "PlataformaUnidadGasto".

- DenegarRelacion

  Igual que el método anterior solo que se le cambia el estado a "Denegadas" en vez de a "Validadas".

- EliminarRelacion

  Igual que el método anterior solo que se le cambia el estado a "Eliminadas" en vez de a "Denegadas".

- SeleccionoRelacionEditableUnidadGasto

  Igual que el método "SeleccionoRelacionEditable" pero en vez de devolver la vista "PlataformaPDIEditable" devuelve "PlataformaUnidadGastoEditable".

- EnviarRelacionUnidadGasto

  Igual que el método "EnviarRelacion" pero en vez de devolver la vista "PlataformaPDIEditable" devuelve "PlataformaUnidadGastoEditable".

- GuardarRelacionUnidadGasto

  Igual que el método "GuardarRelacion" pero en vez de devolver la vista "PlataformaPDIEditable" devuelve "PlataformaUnidadGastoEditable".

- NuevaRelacionUnidadGasto

  Igual que el método "NuevaRelacion" pero en vez de devolver la vista "PlataformaPDI" devuelve "PlataformaUnidadGasto".

- SeleccionoRelacionResponsableGestion

  Es igual que el método “SeleccionoRelación” pero en vez de devolver la vista “PlataformaPDI” devuelve “PlataformarResponsableGestion”

- SeleccionoRelacionResponsableGestionTramitable

  Es igual que el método “SeleccionoRelaciónResponsableGestion” pero en vez de devolver la vista “PlataformaResponsableGestion” devuelve “PlataformarResponsableGestionTramitable”.

- TramitarRelacion

  Igual que el método “ValidarRelacion” solo que se le cambia el estado a “En trámite” en vez de a “Validadas” y se devuelve la vista “PlataformaResponsableGestion”.

- EliminarRelacionUnidadGasto

  Es igual que el método “EliminarRelacion” pero en vez de devolver la vista “PlataformaPDI” devuelve “PlataformaUnidadGasto”.

- EnviarRelacionTramitada

  Igual que los métodos anteriores y además se recogen los parámetros de tramitación de la relación que son el `Expediente, Proveedor y Fecha Pedido`. Se insertan en la base de datos a través del método “dao.TramiteRelacion” y se le cambia el estado a la relación a “Tramitadas” a través del método “dao.CambiaEstadoRelacion”.

  Después se pone el código para llevar el registro de acceso y transacciones y se obtiene la fecha en la que se tramita la relación. También se recoge el `id de la relación, el id del responsable y su nombre y el id del PDI` y estos son los parámetros que se le pasan al método “dao.insertaRegistroporIdTramitación”.

  Se vuelve a poner el código para recorrer las relaciones existentes en las bases de datos ya que su estado ha cambiado y se devuelve la vista “PlataformaResponsableGestion”

- SeleccionoRelacionResponsableGestionTramitada

  Es igual que el método "SeleccionoRelaciónResponsableGestion" pero en vez de devolver la vista "PlataformaResponsableGestion" devuelve "PlataformarResponsableGestionTramitada".

- LlamarRegistro

  Se repite el proceso de todos los métodos anteriores de las cookies, recorrer las relaciones y las unidades de gasto existentes en la base de datos y después se crea una lista de registro para guardar todos los registros que se recogen de la base de datos con el método "dao.leeRegistro" y se le pasa al modelo para que se muestren en la vista "PlataformaResponsableGestionRegistro".

- BorrarPedidoUnidadGasto

  Se repite el proceso de todos los métodos anteriores y para borrar el pedido se recoge el parámetro **"idpedido"** para saber qué pedido es el que se debe borrar y se llama al método "dao.borrapedido" para que lo busque y lo borre de la base de datos. Se vuelven a recorrer las relaciones existentes y finalmente se devuelve la vista "PlataformaUnidadGasto"

- BorrarPedido

  Es igual que el método "BorrarPedidoUnidadGasto" pero en vez de devolver la vista "PlataformaResponsableGestion" devuelve "PlataformarPDI".

2.2.6.2.3 Interfaz.java

Aquí se declaran todos los métodos usados en DAOJdbc.java.

2.2.6.2.4 DTO.java

Esta clase sirve para poder crear un objeto y poder usarlo posteriormente en las demás clases. Es un constructor donde los parámetros que se le pasan son los atributos del objeto.

También tiene sus métodos “getters” para poder obtener los valores de los atributos y “setters” para poder ponerle valor a esos atributos. Clases de este tipo están ‘PDIDTO.java’, ‘PedidoDTO.java’, ‘RelacionDTO.java’, ‘ResponsableDTO.java’ y ‘UnidaddegastoDTO.java’. Es decir, un DTO por cada tabla de la base de datos y con los mismos atributos que tenían en la base de datos y que se mencionaron anteriormente.

2.2.6.2.5 Mapper.java

En esta clase se implementa el mapeo entre el DTO y la base de datos, es decir, permite para los valores introducidos en el objeto DTO del proyecto introducirlos en la base de datos sin tener que hacerlo manualmente cada vez.

Presupuesto

El proyecto recoge el trabajo del alumno en la confección del presente Trabajo Fin de Grado, lo cual consiste en 300 horas de trabajo en total y se divide entre las distintas actividades que se han realizado, estableciéndose un coste de 40,00 € por hora. A continuación, se presenta el desglose del presupuesto:

Horas	Actividad	Precio por hora	Precio total
40	Estudio del proceso administrativo a implementar y de las tecnologías a usar	40,00 €	1.600,00 €
5	Diseño preliminar	40,00 €	200,00 €
5	Implementación del servidor	40,00 €	200,00 €
120	Implementación del cliente	40,00 €	4.800,00 €
10	Implementación de la base de datos	40,00 €	400,00 €
60	Pruebas y corrección de errores	40,00 €	2.400,00 €
60	Preparación de la documentación	40,00 €	2.400,00 €
		SUBTOTAL	12.000,00 €
		IVA (21%)	2.520,00€
		TOTAL PRESUPUESTO	14.520,00 €

Tabla 1. Presupuesto

El presupuesto para el proyecto asciende a una cantidad total de CATORCE MIL QUINIENTOS VEINTE EUROS (14.520€).

4 Conclusiones

Analizando los objetivos iniciales del proyecto se puede decir que se han cumplido. También durante el proyecto se han añadido algunas funcionalidades nuevas que no son objetivos como tal pero que hacen mejor la aplicación como puede ser el mantenimiento de la sesión mediante cookies o poder borrar algún artículo añadido anteriormente. También se ha de decir que con el objetivo de la firma digital se han tenido problemas para llevarlo a cabo y por ello se ha optado por generar el PDF en la plataforma y que los usuarios lo firmen con la aplicación de Autofirma.

Líneas de futuro

Como líneas de futuro se podría estudiar la viabilidad de mejorar el sistema de firma digital para que la propia aplicación sea capaz de generar y firmar digitalmente los documentos. También se podría añadir una base de datos de los proveedores para gestionar mejor todo el proceso después de tramitación de los pedidos. También se podría estudiar la compatibilidad de esta plataforma con la plataforma de Universitas XXI explicada anteriormente en esta memoria o vincular el acceso con la plataforma SIDUJA que es la plataforma de usuarios de la Universidad de Jaén.

Bibliografía

[1] Apache Tomcat®, [En línea] [Último acceso: 22 agosto 2020] Disponible: http://tomcat.apache.org/

6 [2] Bootstrap · The most popular HTML, CSS, and JS library in the world, [En línea] [Último acceso: 22 agosto 2020] Disponible: https://getbootstrap.com/

[3] Creador de Diagramas de Flujo Online | Lucidchart, [En línea] [Último acceso: 22 agosto 2020] Disponible: https://www.lucidchart.com/pages/es/ejemplos/diagrama-de-flujo-online

[4] Eclipse, [En línea] [Último acceso: 22 agosto 2020] Disponible: https://eclipse-sdk.softonic.com/

[5] Ecodeup - Desarrollo de Aplicaciones Web y Móvil, [En línea] [Último acceso: 22 agosto 2020] Disponible: https://www.ecodeup.com/

[6] Inyección de dependencias en Spring - Programando o Intentándolo, [En línea] [Último acceso: 22 agosto 2020] Disponible: https://programandoointentandolo.com/2013/05/inyeccion-de-dependencias-en-spring.html

[7] MDN, [En línea] [Último acceso: 22 agosto 2020] Disponible: https://developer.mozilla.org/es/

[8] Mis Pedidos - Aplicaciones en Google Play, [En línea] [Último acceso: 22 agosto 2020] Disponible: https://play.google.com/store/apps/details?id=com.darngar.android.myordersfree&hl=es

[9] Stack Overflow, [En línea] [Último acceso: 22 agosto 2020] Disponible: http://stackoverflow.com/

[10] UNIVERSITAS XXI Soluciones y Tecnología para la Universidad, [En línea] [Último acceso: 22 agosto 2020] Disponible: https://www.universitasxxi.com/

[11] W3Schools Online Web Tutorials, [En línea] [Último acceso: 22 agosto 2020] Disponible: https://www.w3schools.com/

[12] WampServer, la plate-forme de développement Web sous Windows - Apache, MySQL, PHP, [En línea] [Último acceso: 22 agosto 2020] Disponible: https://www.wampserver.com/en/

[13] YouTube, [En línea] [Último acceso: 22 agosto 2020] Disponible: https://www.youtube.com

ANEXO I. Manual de instalación del servidor y base de datos para el correcto funcionamiento de la aplicación

Lo primero que se ha de hacer es instalar el servidor, en este caso Apache Tomcat la versión 8.5 es tener instalado el JDK de Java que es el kit de desarrollo de Java que provee herramientas para poder desarrollar la creación de programas en Java. Se puede hacer desde este enlace https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html que es el sitio web de Oracle.

Ahora ya se puede proceder a instalar Apache Tomcat que será descargado de la siguiente página web http://tomcat.apache.org/download-80.cgi la versión que se desee. Se instala como cualquier programa, es decir, el asistente de instalación es trivial. Una vez se llegue al paso de la Ilustración 38 se debe poner los puertos que debe usar la aplicación. Los que se ven son los que trae por defecto.

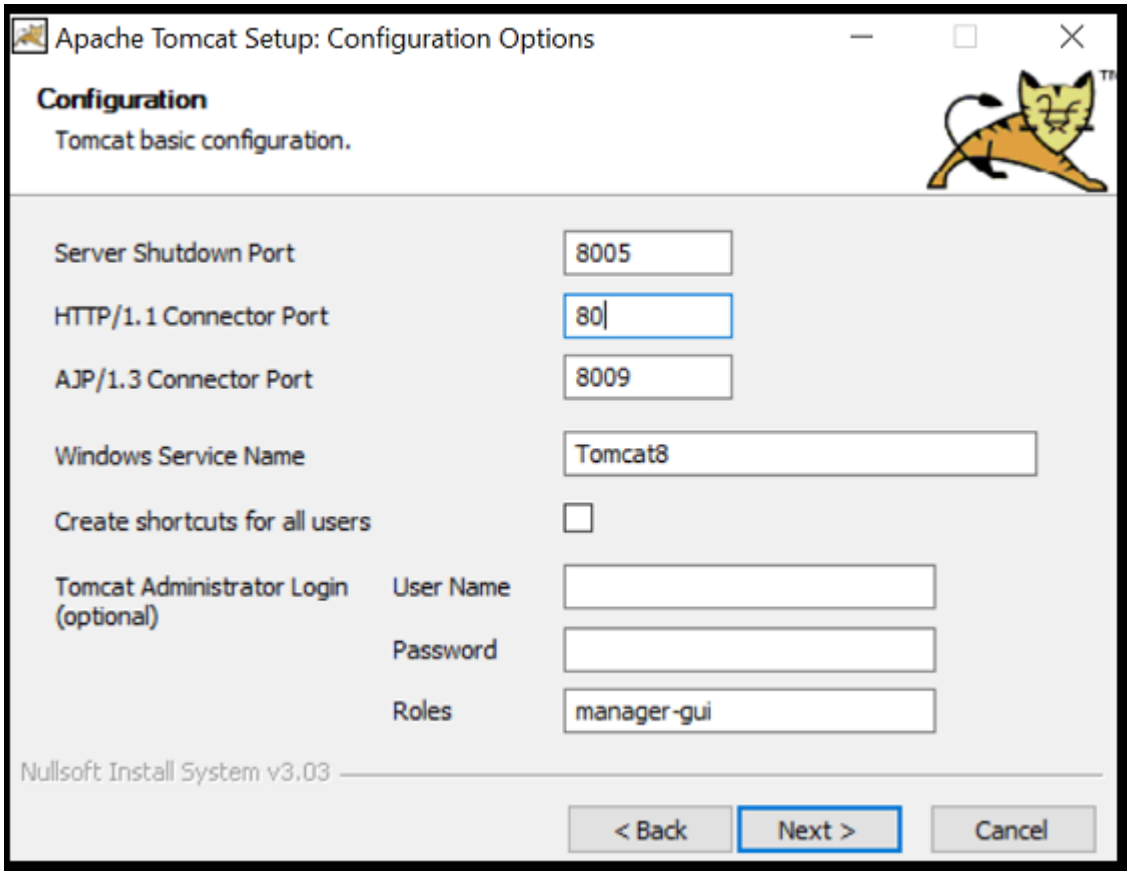


Ilustración 39. Instalación Apache Tomcat Paso 1

Una vez se termine el asistente de instalación se ha de iniciar Tomcat y en el apartado de Java se debe dejar vacíos los valores de los campos ‘Initial memory pool’ y ‘Maximum memory pool’.

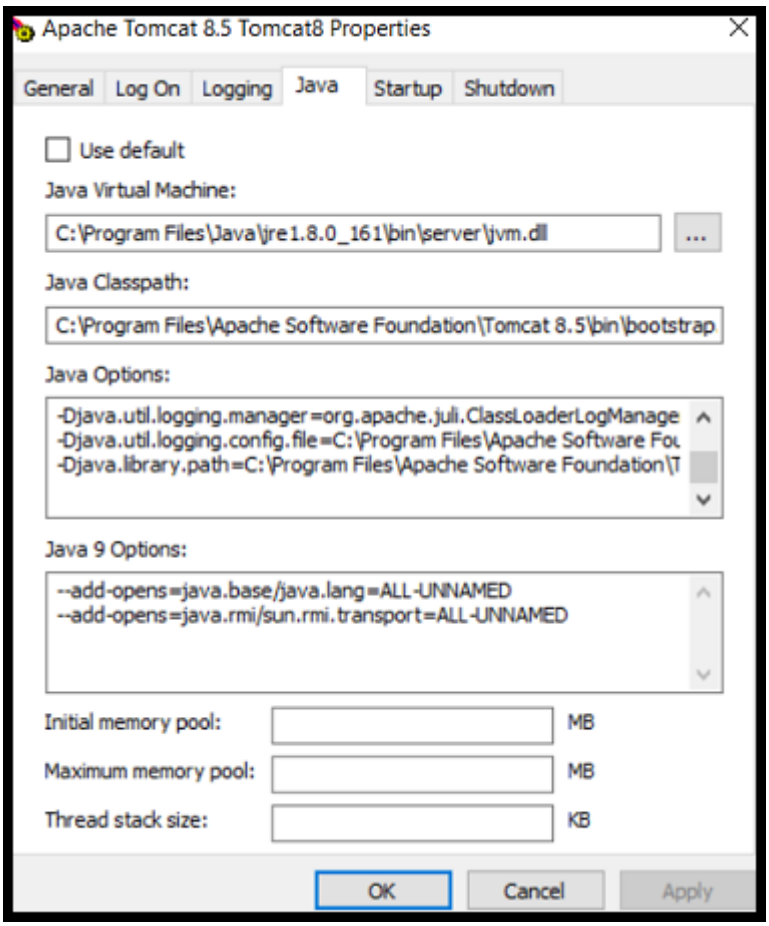


Ilustración 40. Instalación Apache Tomcat Paso 2

En la ubicación de la instalación de Tomcat, se debe abrir ‘*CATALINA_HOME/conf/web.xml’*.

Se debe reemplazar la página de error por defecto mediante la adición de lo siguiente al fichero web.xml:

<error-page><exception-type>java.lang.Throwable</exception-type><location>/error.jsp</location></error-page>

En la ubicación de la instalación de Tomcat, se debe abrir ‘*CATALINA_HOME/conf/server.xml’* y añadir lo siguiente dentro de las etiquetas <Host> </Host>:

<Valve className="org.apache.catalina.valves.ErrorReportValve" showReport="false" showServerInfo="false" />

Y ya estaría configurado correctamente el servidor Apache Tomcat

Ahora para integrarlo con el programa Eclipse se debe hacer los siguientes pasos:

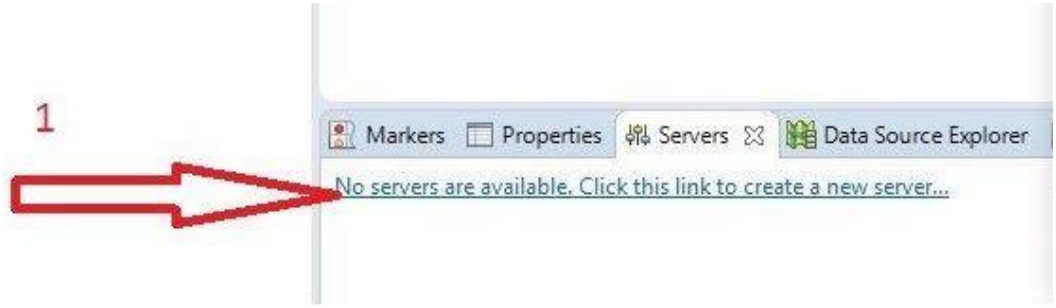


Ilustración 41. Integración Apache Tomcat en Eclipse Paso 1

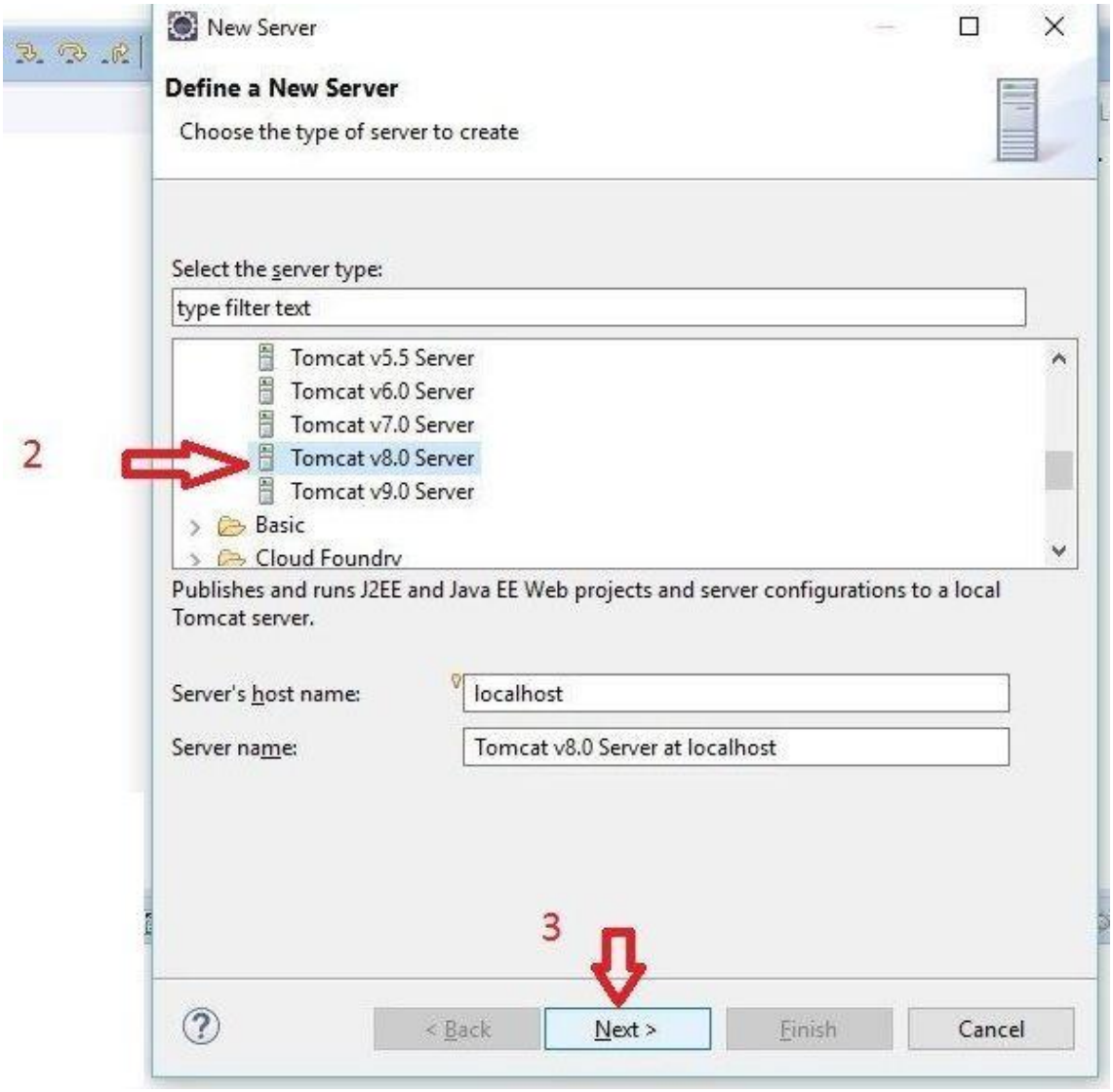


Ilustración 42. Integración Apache Tomcat en Eclipse Paso 2

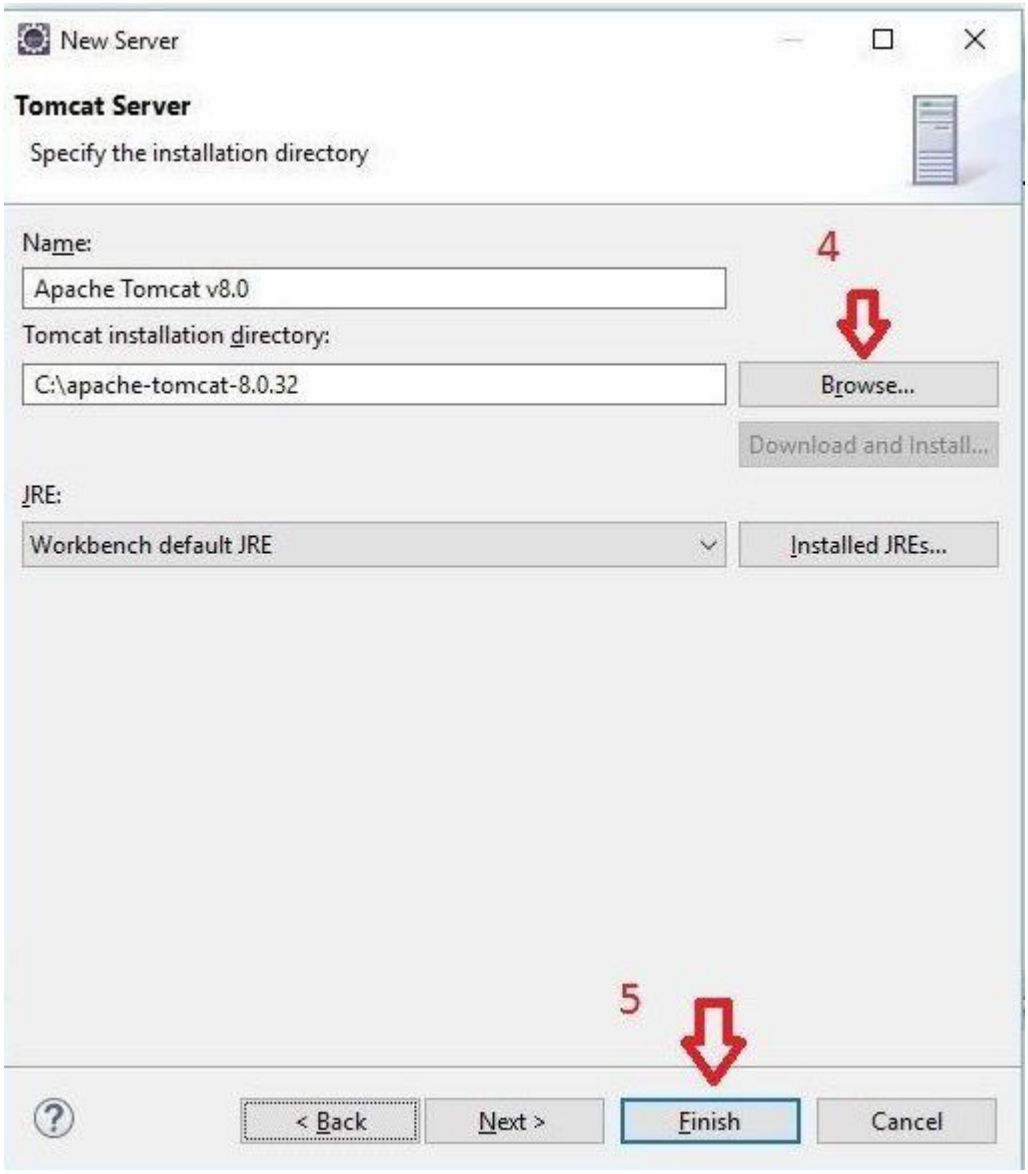


Ilustración 43. Integración Apache Tomcat en Eclipse Paso 3

Una vez se haya configurado esto, cada vez que se ejecute el proyecto se arrancará el servidor y permitirá ver el proyecto en localhost poniendo la URL del proyecto que es: "localhost: el puerto elegido, en nuestro caso 8081, y el nombre del proyecto

("relaciondenecesidades") en cualquier navegador web y estará la aplicación funcionando como se puede observar en la Ilustración 38

Ilustración 44. Acceso a la Plataforma Web

También se ha de arrancar el motor MySQL para poner en marcha la base de datos, en este caso se arranca este motor desde WampServer que lo trae integrado. Se abre el programa WampServer y cuando esté en color verde, como en la Ilustración 43, ya está el servidor activo y la base de datos también.

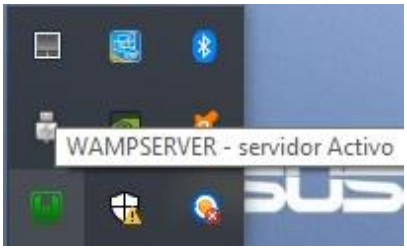


Ilustración 45. WampServer

Esta base de datos se puede manejar desde PHPmyAdmin como se puede observar en la Ilustración 44:

Ilustración 46. PhpMyAdmin

ANEXO II. Manual de usuario

Este manual de usuario enseñará a cualquier usuario que vaya a usar la aplicación cómo usarla correctamente ya sea PDI, responsable de unidad de gasto o responsable de gestión.

8

Ilustración 47. Vista principal portal web

El usuario deberá pulsar el botón “Pinche aquí” para que se le abra la ventana de inicio de sesión donde introducir su usuario y contraseña. Según su rol la aplicación le llevará a una vista u otra.

Ilustración 48. Usuario y contraseña

Si las credenciales son incorrectas saldrá la siguiente alerta de la Ilustración 41 indicando que el usuario o la contraseña son incorrectos

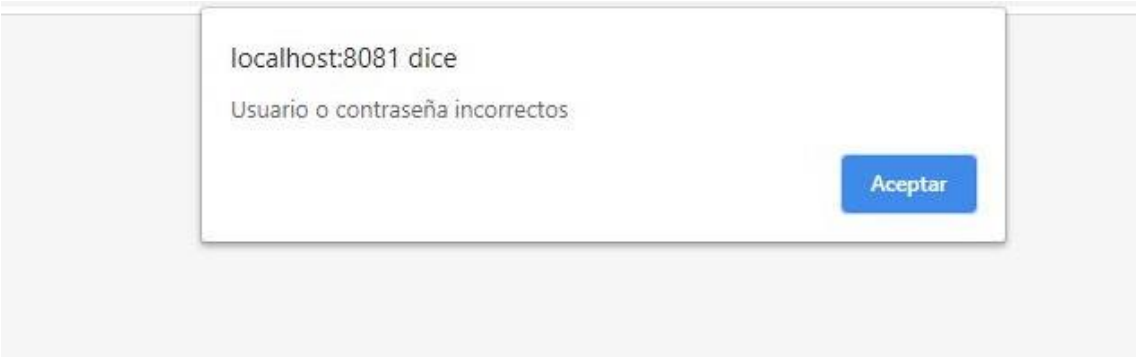


Ilustración 49. Contraseña incorrecta

8.1 Perfil de ingreso PDI

Si se accede a la aplicación como PDI la vista resultante será la mostrada en la Ilustración 50:

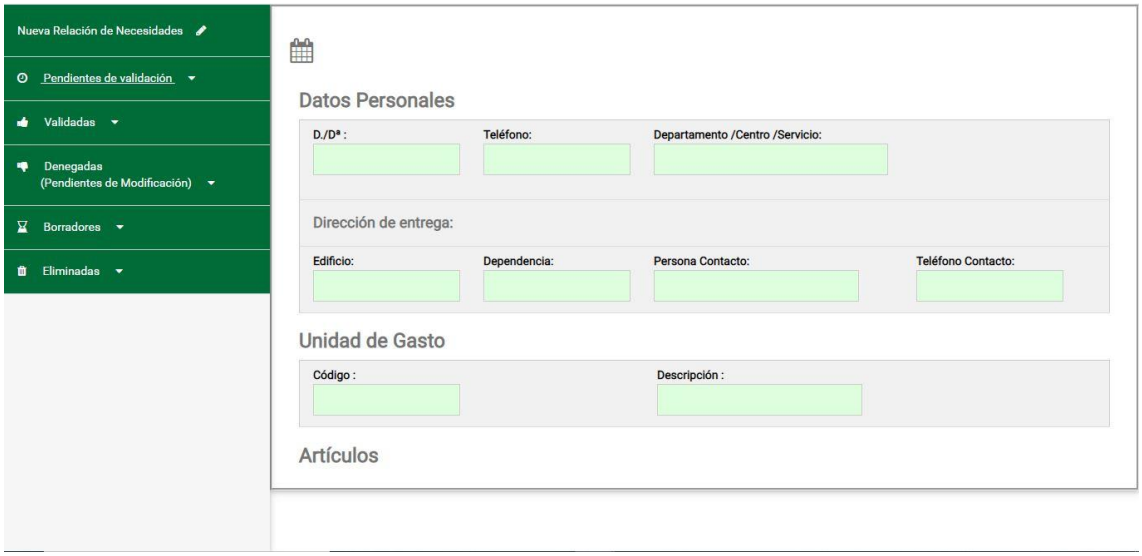


Ilustración 50. Vista principal PDI

A la izquierda se puede encontrar un menú de navegación donde poder consultar las relaciones según el estado en el que se encuentren (Pendientes de validación, Validadas, Denegadas (Pendientes de Modificación), Borradores y Eliminadas). También si se quiere crear una nueva relación deberás pinchar sobre "Nueva Relación de Necesidades". Se abrirá el siguiente recuadro donde deberás rellenar los datos de las distintas pestañas.

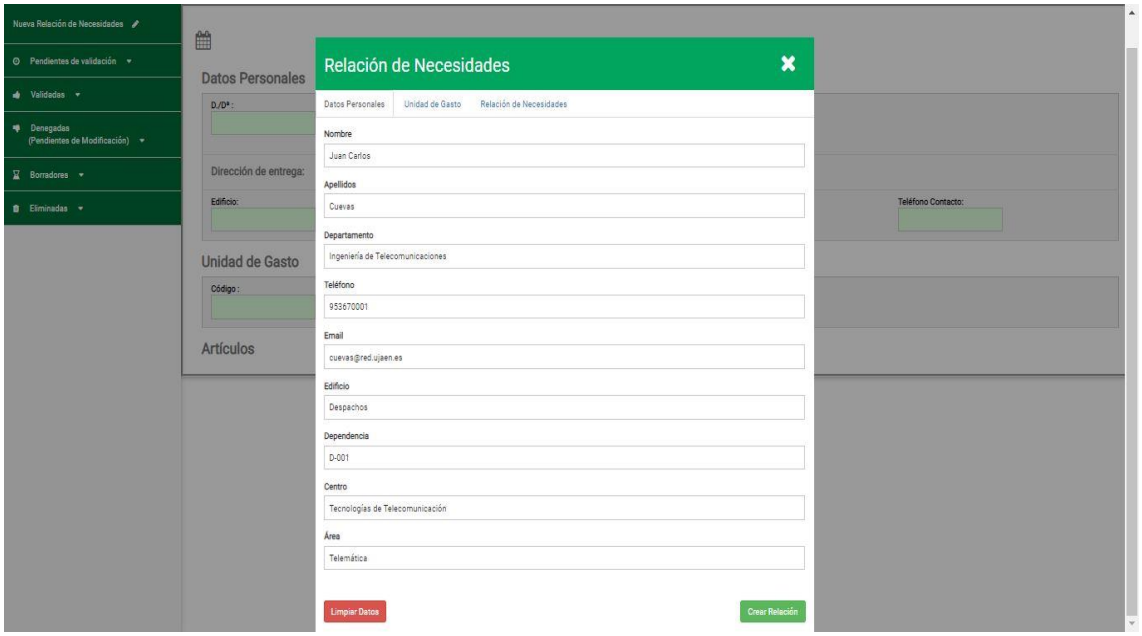


Ilustración 51. Crear Nueva Relación

Los datos personales como se puede observar ya vienen rellenos de la sesión. Se deberá rellenar la pestaña "Unidad de Gasto" donde se deberá escribir la Unidad de Gasto que se desee y saldrán las unidades de gasto coincidentes con esas letras que se están escribiendo y se deberá seleccionar una de ellas para que se escriba su respectivo Id.

También se deberá rellenar la pestaña de "Relación de Necesidades" donde se deberá introducir el título y seleccionar la fecha.

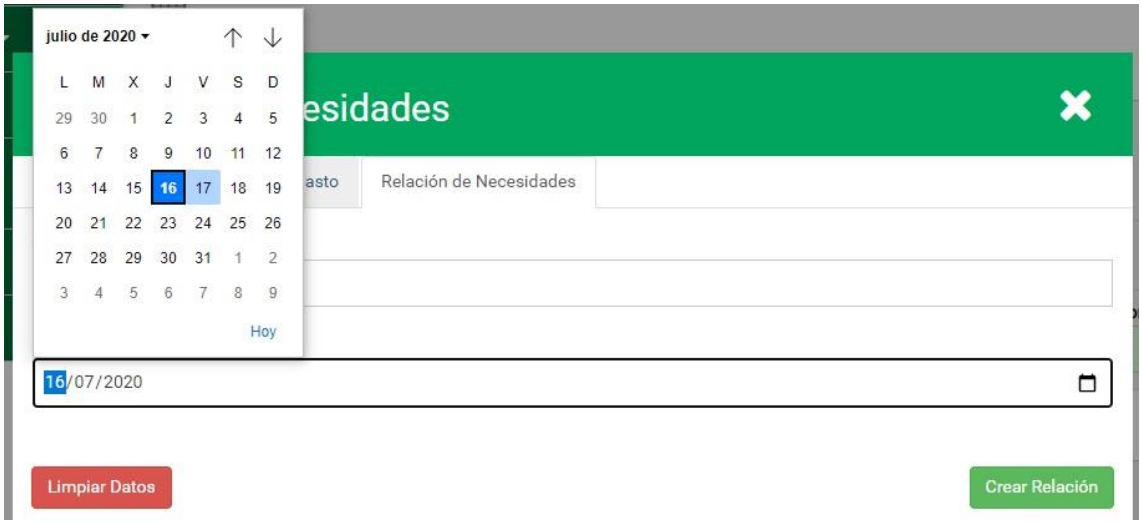


Ilustración 53. Pestaña Relación de Necesidades

Y una vez se pulse el botón de “Crear Relación” saldrá una alerta diciendo que la relación ha sido creada correctamente y que ahora se encuentra en Borradores. Si por el contrario no se está satisfecho con los datos introducidos se le puede dar al botón de “Limpiar Datos” para que se reseteen.

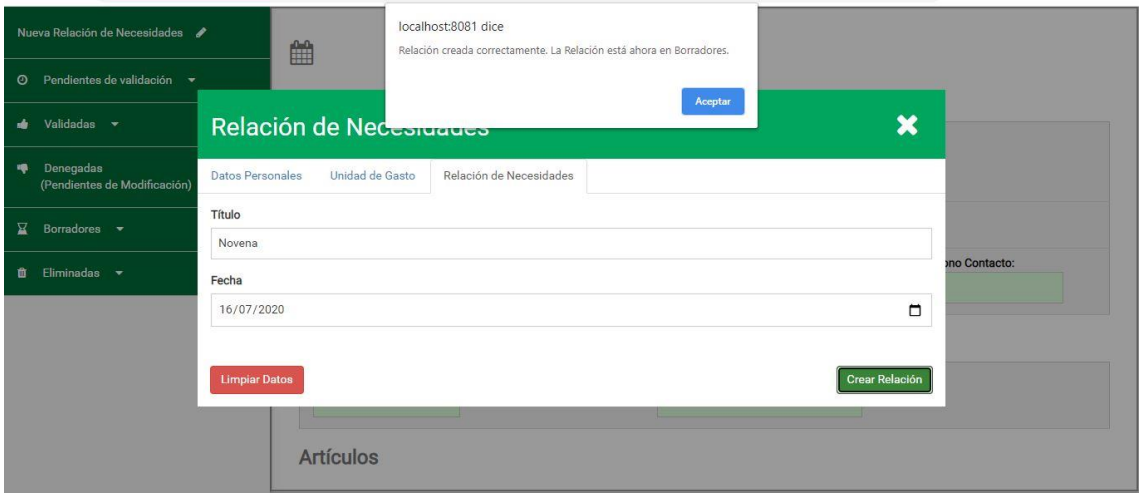


Ilustración 54. Alerta Relación creada correctamente

Una vez creada la relación se debe ir a Borradores para seleccionarla y añadirle los artículos o editar algunos datos como “Persona Contacto” o “Teléfono Contacto” pertenecientes a la dirección de entrega.

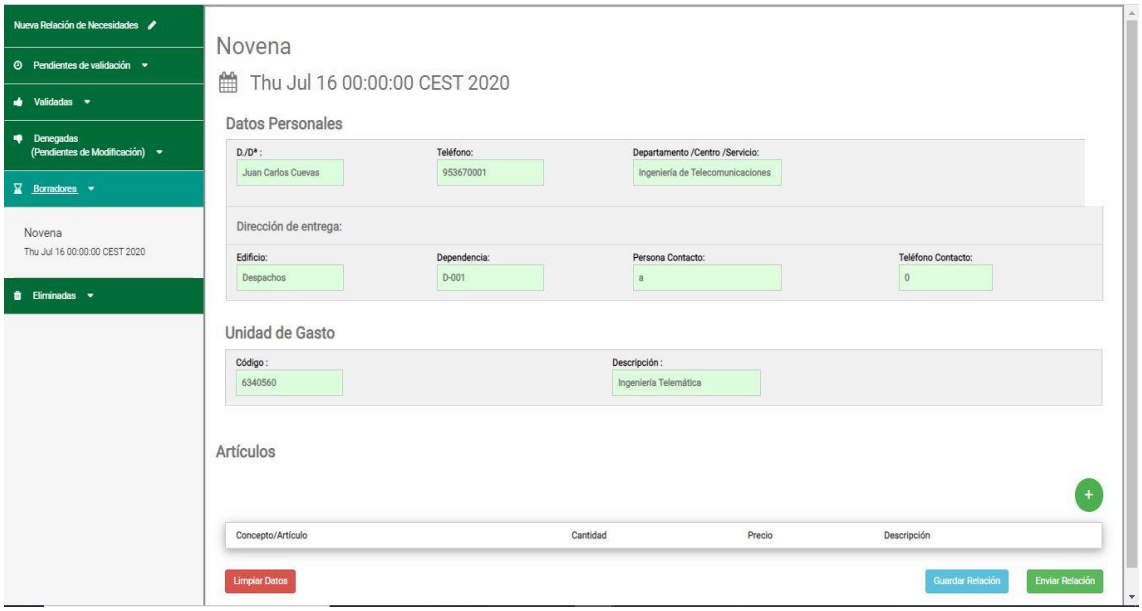


Ilustración 55. Relación en Borradores

Ilustración 56. Añadir Artículos

Aquí se puede observar los datos de la relación seleccionada y añadir artículos pulsando el botón verde con un '+'. Cuando se pulse ese botón saldrá una fila en la tabla igual que la de la siguiente ilustración donde se deberá rellenar los datos.

Si se quiere añadir más artículos, se deberá pulsar el botón tantas veces como artículos desee añadir. También puede pulsar el botón de "Limpiar Datos" para resetear los datos y "Guardar Relación" para guardar la relación con los artículos que ha añadido y añadir más después o enviarla en un futuro. Una vez se pulse el botón de "Enviar Relación" la relación será enviada y se quedará en el estado de "Pendientes de Validación" esperando a que el responsable de unidad de gasto la valide.

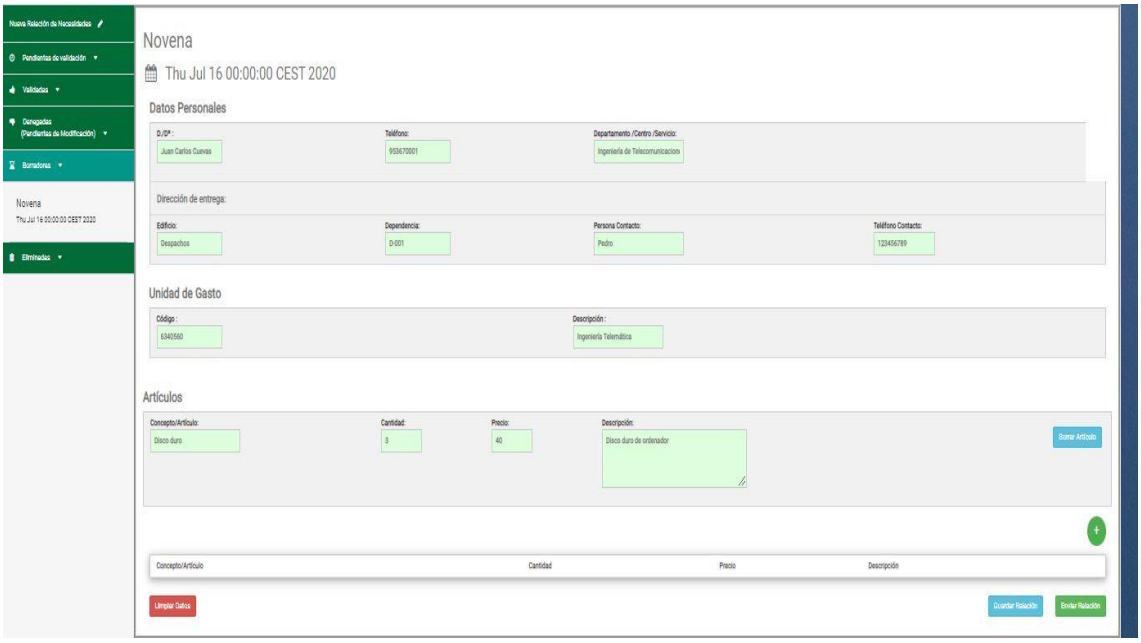
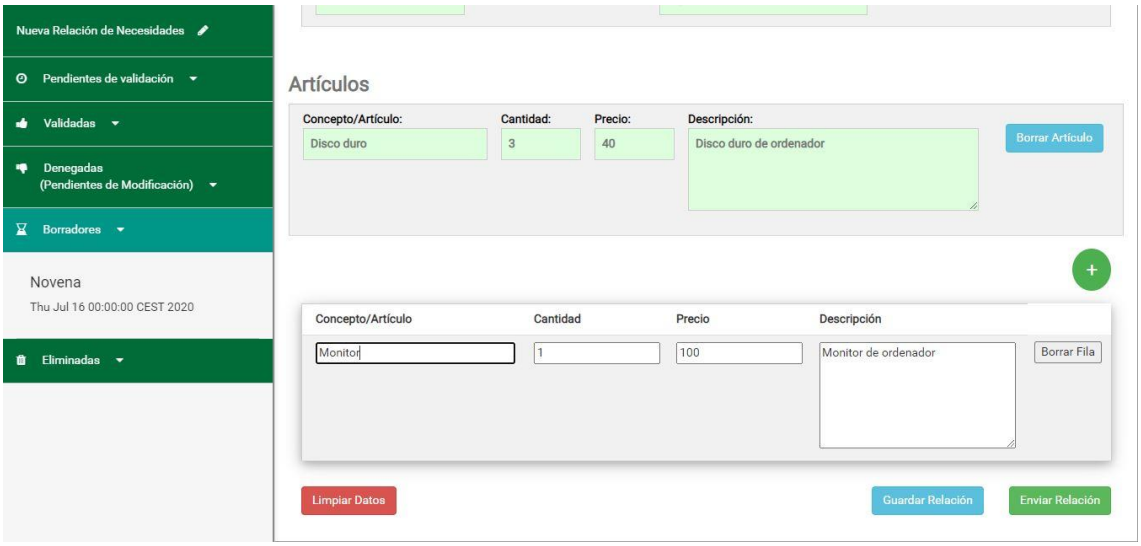


Ilustración 57. Relación guardada



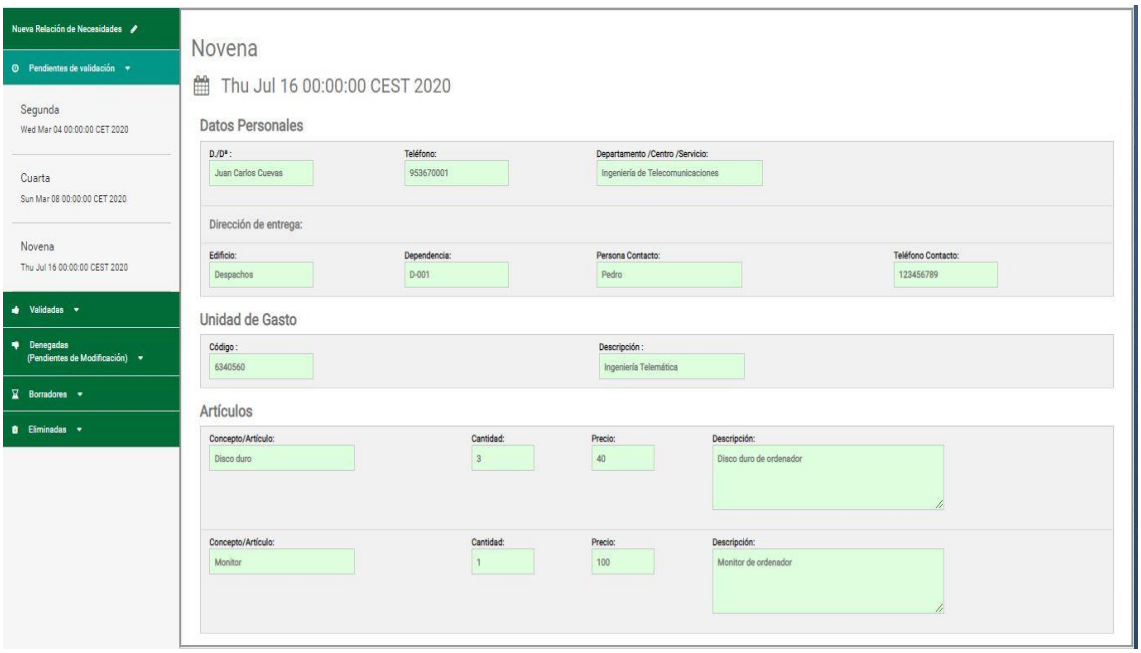


Ilustración 58. Relación en Pendientes de validación

Si el Responsable de Unidad de Gasto ha denegado la relación, esta relación se irá al apartado "Denegadas (Pendientes de Modificación)" donde se podrá modificar esta relación, añadirle o quitarle nuevos artículos, guardarla, enviarla y eliminarla. Si se guarda seguirá en este apartado, si se envía se volverá a ir al estado "Pendientes" y si se elimina se irá al estado "Eliminadas".

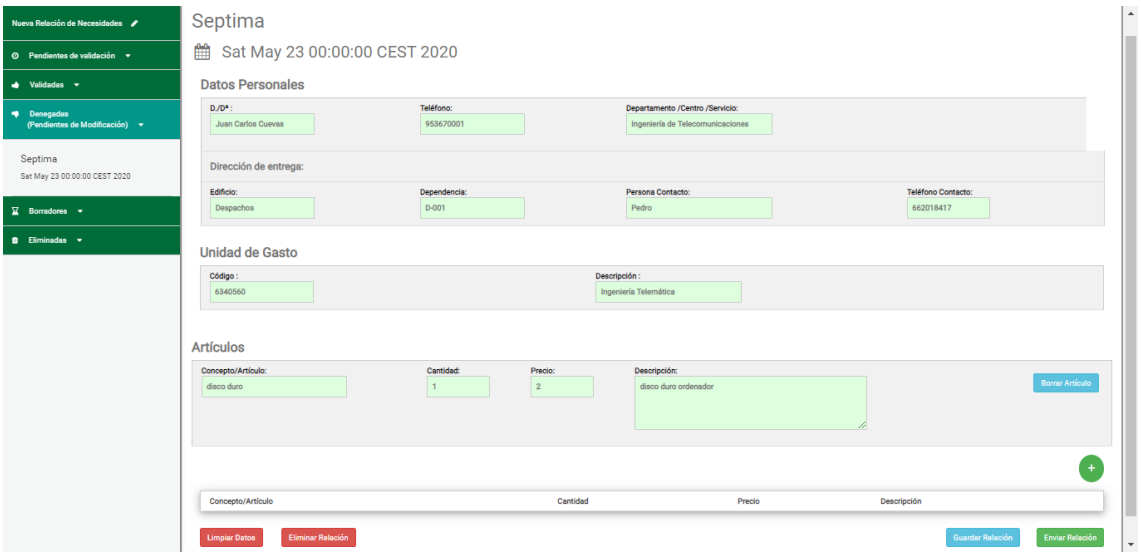


Ilustración 59. Relación Denegada (Pendiente de Modificación)

8.2 Perfil de ingreso de Responsable Unidad de Gasto

Esta es la vista principal de un responsable de unidad de gasto donde se tiene un menú de navegación igual que el de un PDI, más su menú como responsable. Como PDI puede crear una relación, añadir artículos a las relaciones, consultar las distintas relaciones y modificarlas.

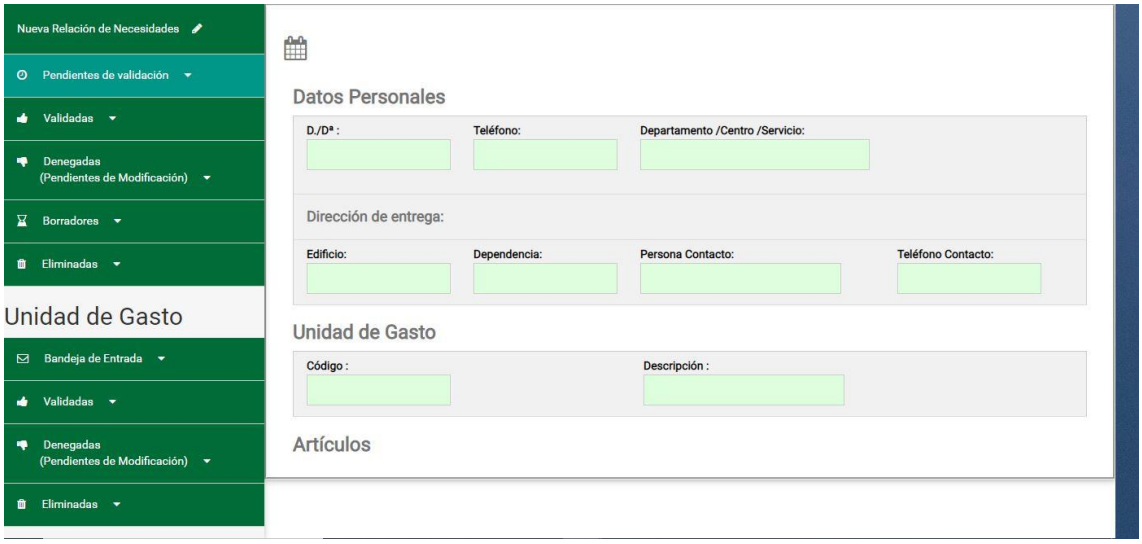


Ilustración 60. Vista principal Responsable Unidad de Gasto

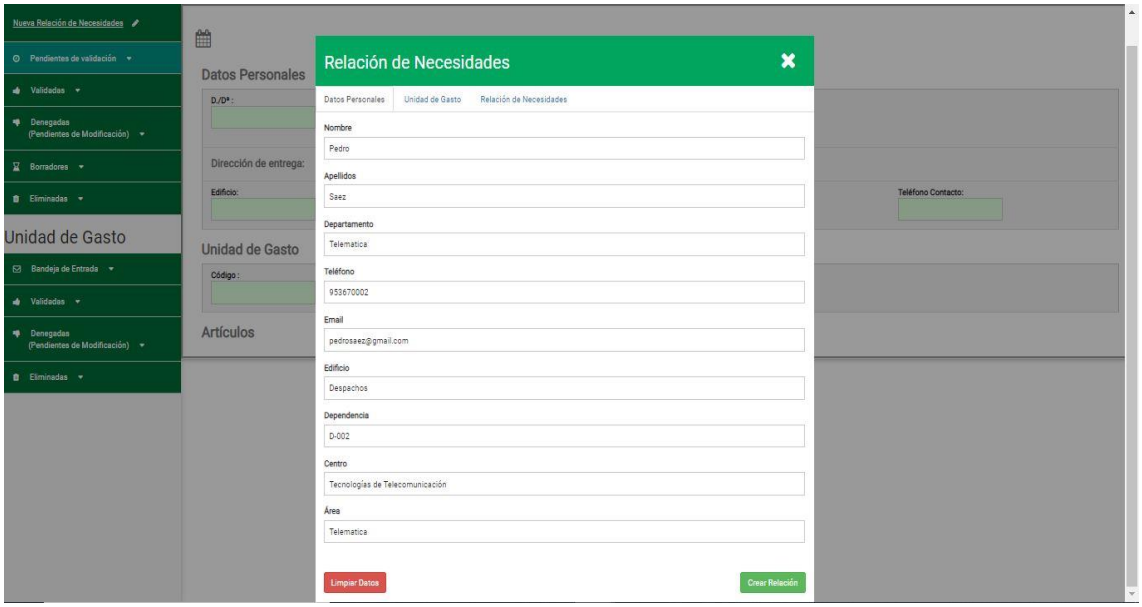


Ilustración 61. Crear Relación Responsable Unidad de Gasto

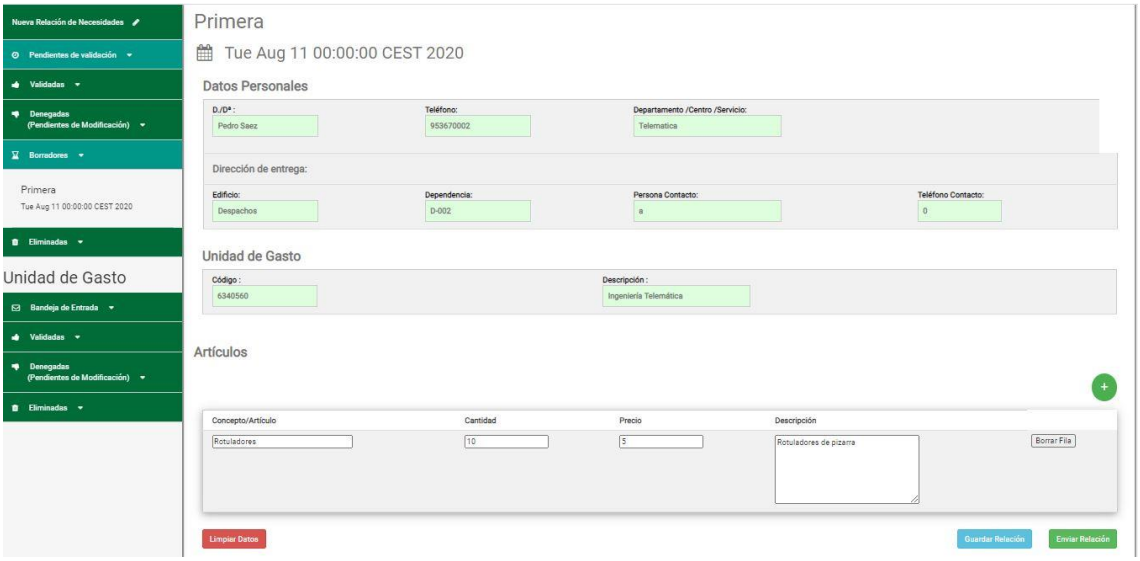


Ilustración 62. Relación creada Responsable Unidad de Gasto

Y una vez se cree y se le añada los artículos y se envíe estará tanto en el estado "Pendientes de Validación" como en la "Bandeja de Entrada" de la Unidad de Gasto si la Unidad de Gasto que se ha elegido es la suya.

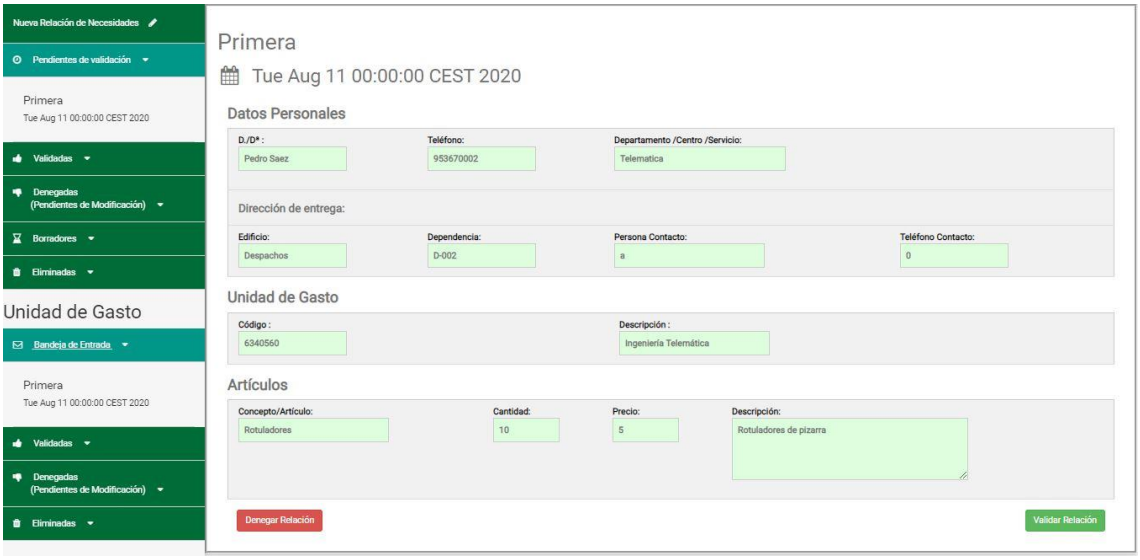


Ilustración 63. Validar o Denegar Relación

Aquí en la Bandeja de Entrada se podrá seleccionar la relación que se desee y validarla o denegarla seleccionando el botón correspondiente y ya se irá al estado "Validadas" o "Denegadas" tanto para él como responsable como al PDI correspondiente.

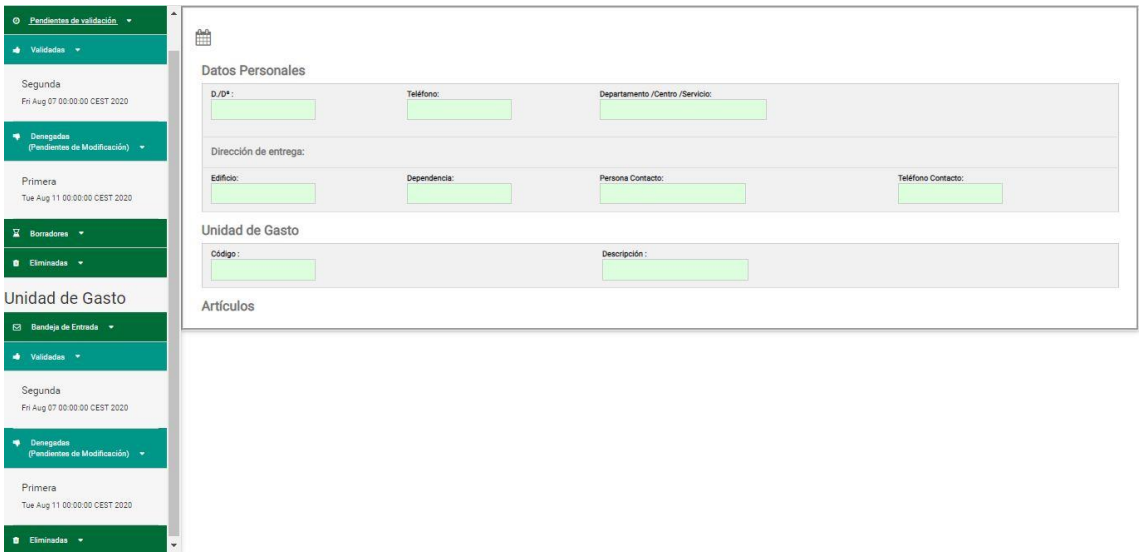


Ilustración 64. Relaciones Validadas y Denegadas

8.3 Perfil de ingreso de Responsable de Gestión

Esta es la vista principal de un responsable de gestión que es similar a las anteriores donde tiene su menú de navegación con su Bandeja de Entrada, las relaciones que están "En trámite", las relaciones tramitadas y el registro de acceso de los usuarios. Para comenzar a tramitar una relación deberá seleccionar la relación correspondiente y pulsar el botón de "Tramitar Relación".

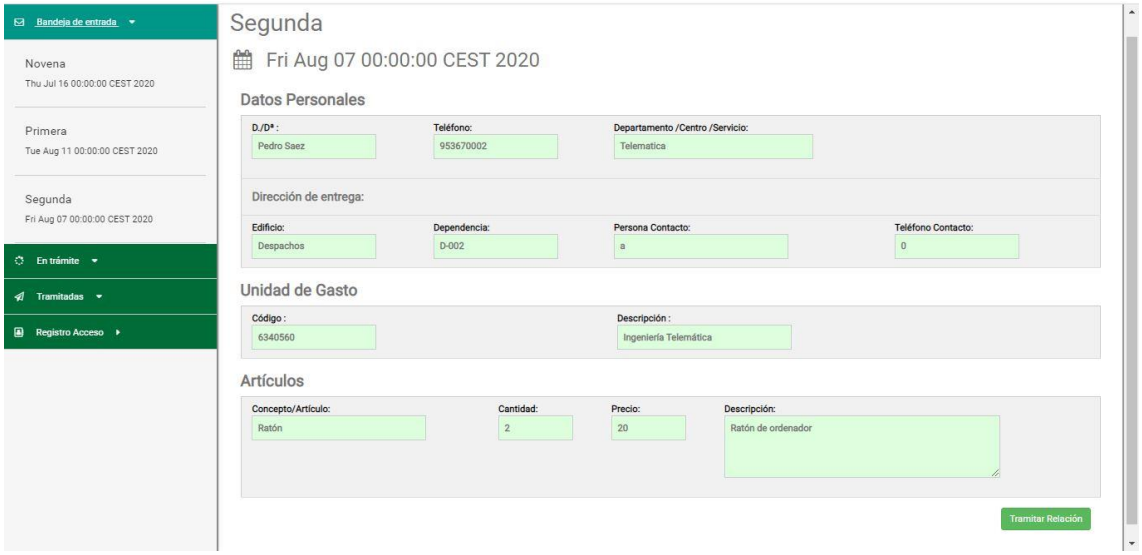


Ilustración 66. Tramitar Relación

Una vez se pulse ese botón la relación pasará a estar en trámite suyo y de ningún responsable más ya que usted es el que ha elegido tramitar esa relación y le saltará una alerta indicando que debe dirigirse a Relaciones “En Trámite” para completar su tramitación.

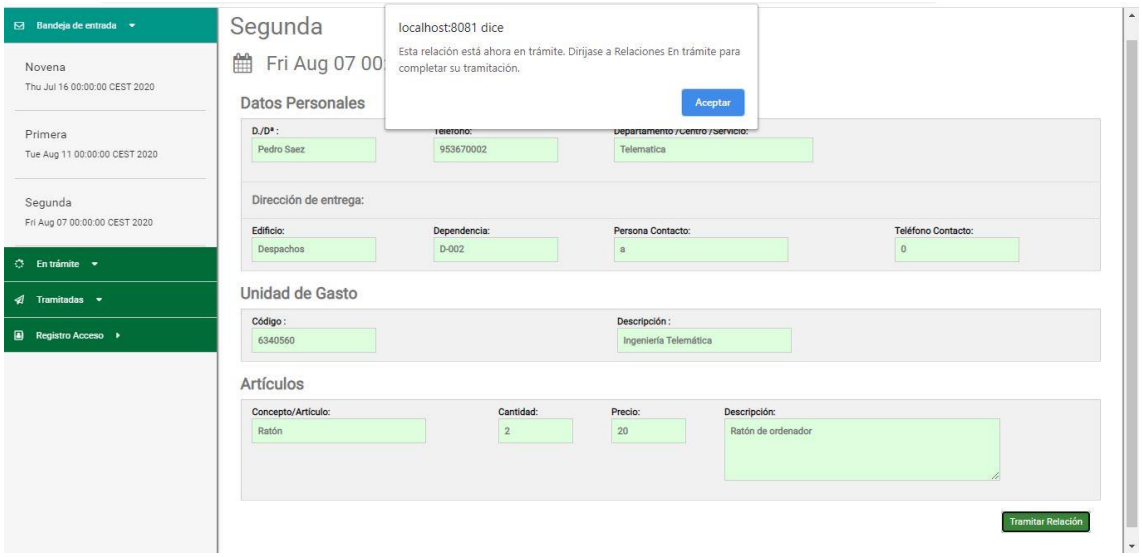


Ilustración 67. Alerta Relación en Trámite

Una vez se seleccione la relación que desea continuar la tramitación le saldrá la relación seleccionada con todos sus datos y donde se deberá rellenar los campos “Expediente”, “Proveedor” y “Fecha Pedido” y enviar la relación una vez se complete todo esto.

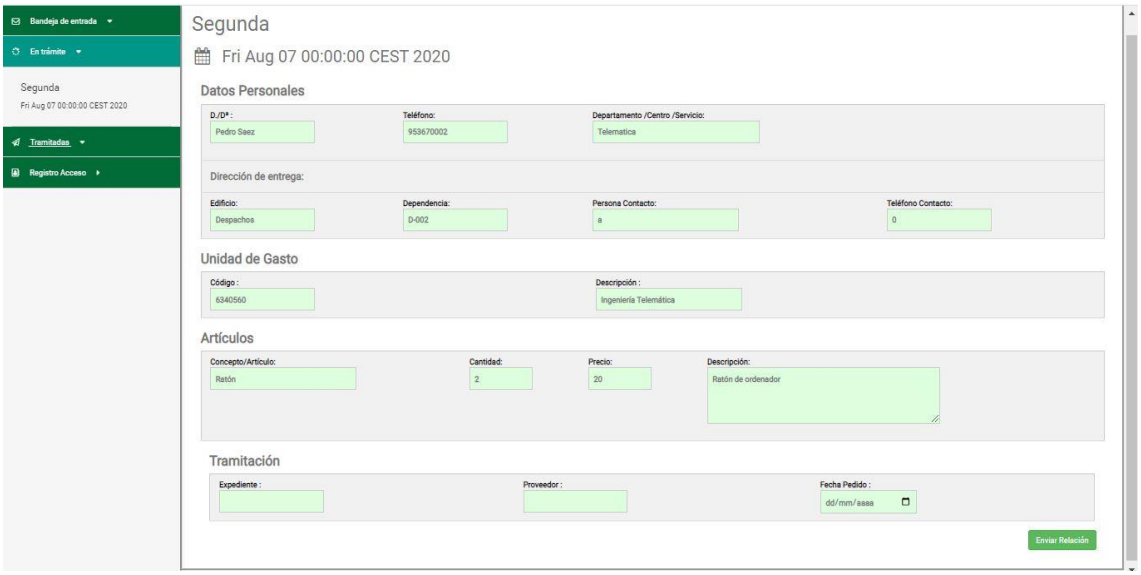


Ilustración 65. Tramitación Relación

También se pueden consultar las relaciones tramitadas con todos sus datos seleccionando la sección “Tramitadas”.

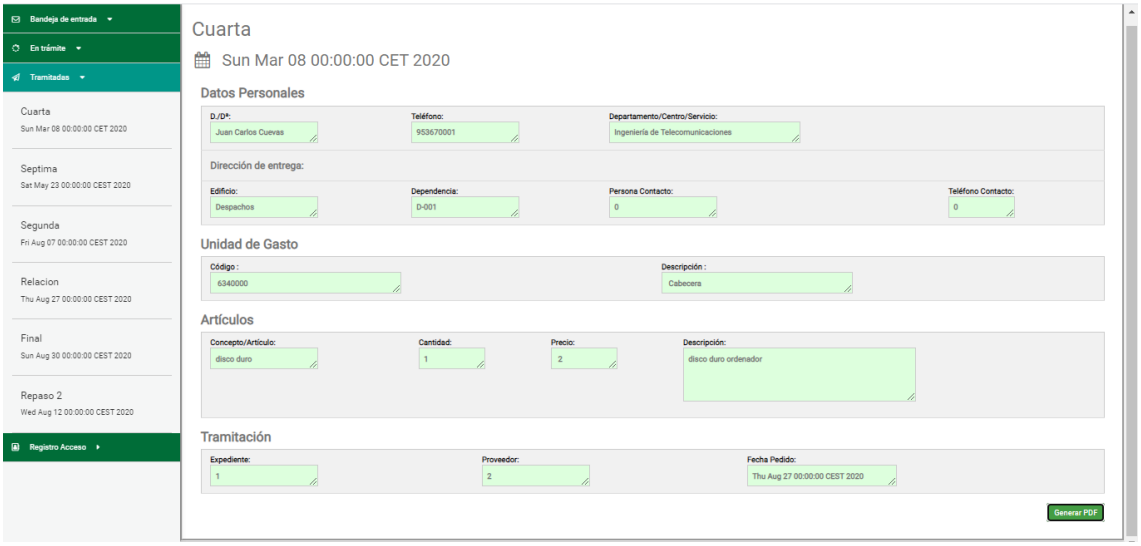


Ilustración 66. Relación Tramitada

Si se quiere generar el PDF de esta relación y posteriormente firmarlo digitalmente con la aplicación de Autofirma se deberá pinchar en el botón de “Generar PDF”. Y se generará un PDF como se puede ver en la Ilustración 66.

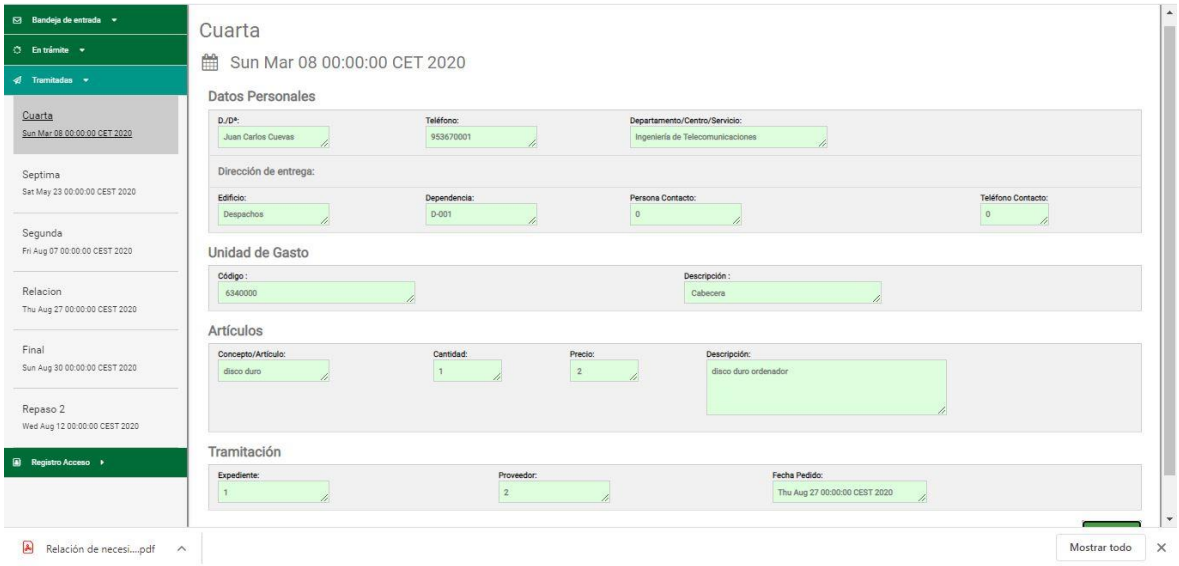


Ilustración 68. Generar PDF

Y si se quiere consultar el registro de acceso de los usuarios que entran a la plataforma web tanto sus datos como las fechas de cuando inician sesión, envían una relación, la validan, la tramitan etc. deberá pulsar en el apartado "Registro Acceso" y podrá consultar todo esto en una tabla.

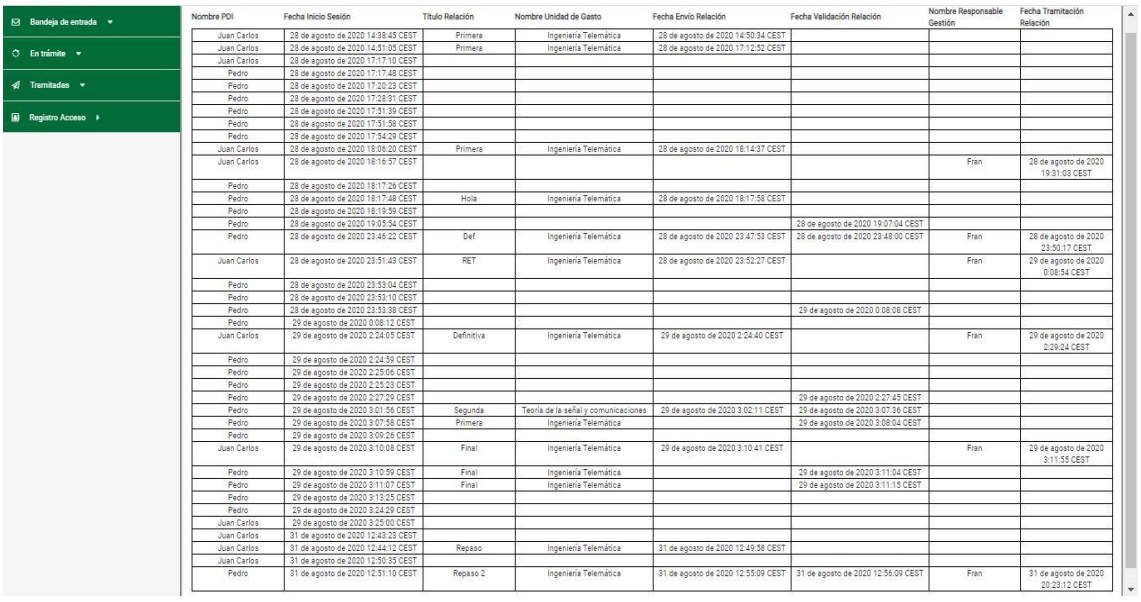


Ilustración 69. Registro Acceso Usuarios

Pedro	31 de agosto de 2020 12:51:10 CEST	Repaso 2	Ingeniería Telemática	31 de agosto de 2020 12:55:09 CEST	31 de agosto de 2020 12:56:09 CEST	Fran	31 de agosto de 2020 21:04:52 CEST

Ilustración 70. Fila Registro Acceso Usuarios