Documentación de Análisis y Diseño de la gestión de usuarios

Documentación de Análisis y Diseño de Gestión de Usuarios
PMPV

Introducción

Propósito

El objetivo de esta documentación es describir el análisis y diseño de la gestión de usuarios, roles, permisos, autenticación y autorización en el sistema Planificación Mantenimiento Preventivo Vehicular (PMPV).

Alcance

La documentación aborda el análisis y diseño del servicio relativo a usuarios:

- Gestión de usuarios.
- Gestión de roles.
- Asignación y eliminación de permisos a un rol.
- Autorización y autenticación en el sistema.

Contexto del Proyecto

Descripción del Proyecto

El sistema de planificación de mantenimiento preventivo vehicular tiene como objetivo principal gestionar el mantenimiento programado de una flota de vehículos. La aplicación facilitará la gestión de las tareas de mantenimiento proporcionadas por el fabricante como de aquellas generadas por el propio operador. Además, se gestionarán las revisiones de las tareas, notificando cualquier cambio a los ingenieros responsables de la gestión y planificación del mantenimiento de vehículos.

Los ingenieros crearán planes de mantenimiento para vehículos similares que al aplicarlos generarán, automáticamente, órdenes de trabajo. Estas, serán gestionadas por los mecánicos, los cuales también podrán crear nuevos trabajos debido a la detección de errores durante la realización de sus trabajos. Además, la aplicación gestionará los diferentes tipos de usuario (ingenieros, mecánicos, administradores, etc..) que podrán acceder de forma segura a las diferentes funcionalidades de esta.

Cabe destacar que el plan de mantenimiento preventivo contribuye a prevenir fallos y garantizar el correcto funcionamiento de los vehículos. De hecho, la aplicación se deberá diseñar para que gestione un crecimiento y sea escalable y esté en la nube, con procesos automatizados para minimizar errores y garantizar su calidad.

Stakeholders

Administradores

Requisitos

Requisitos funcionales

PMPV-31: US-2023-Como administrador quiero dar de alta diferentes tipos de usuarios FINALIZADA

PMPV-32: US-2023-Como administrador quiero actualizar un usuario FINALIZADA

PMPV-34: US-2023-Como administrador quiero dar de baja un usuario FINALIZADA

PMPV-35: US-2023-Como administrador quiero crear un rol FINALIZADA

PMPV-36: US-2023-Como administrador quiero buscar un usuario por código de usuario FINALIZADA

PMPV-38: US-2023-Como administrador quiero ver la lista de usuarios FINALIZADA

PMPV-40: US-2023-Como administrador quiero eliminar un rol FINALIZADA

PMPV-41: US-2023-Como administrador quiero dar permisos a un rol FINALIZADA

PMPV-43: US-2023-Como usuario quiero iniciar sesión con mi nombre de usuario y contraseña. FINALIZADA

PMPV-46: US-2023-Como usuario quiero cerrar la sesión de una manera segura. FINALIZADA

PMPV-110: US-2023-Como administrador quiero quitar permisos a un rol FINALIZADA

Requisitos no funcionales

PMPV-100: RS-Usabilidad TAREAS POR HACER

PMPV-99: RS-Diseño Responsivo TAREAS POR HACER

Análisis

Permisos y Seguridad

En primer lugar, se definen los niveles de permisos para los distintos roles y se establecen las medidas de seguridad en el sistema. Esto garantiza la seguridad en la autenticación y autorización de la aplicación, de modo que un usuario autenticado solo acceda a aquellas áreas permitidas. A continuación, se detallan los diferentes niveles de acceso que habrá por defecto, ya que el administrador podrá configurarlo.

Módulo	Admin	Operador	Ingeniero	Mecánico
Gestión de usuarios				
Gestión de roles				
Gestión de permisos				
Gestión de tareas				
Gestión de plan de mantenimiento				
Gestión de vehículos				
Gestión de órdenes de mantenimiento				

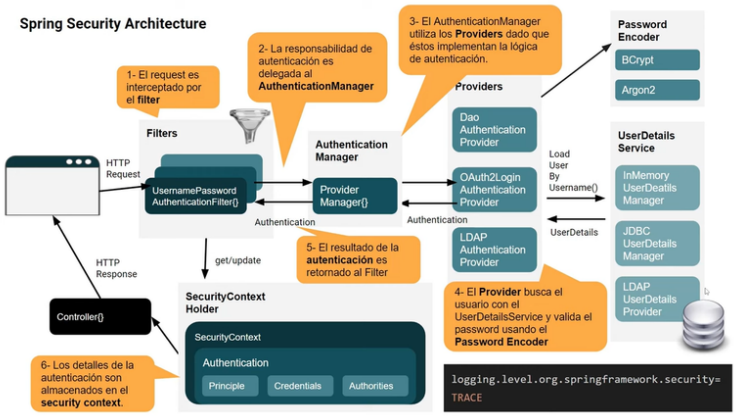
Niveles de permisos

Después de exponer los niveles de acceso, es esencial comprender cómo se diseña la seguridad. Para facilitar este proceso, se empleará la seguridad de *Spring Boot* y *JSON Web Tokens* (JWT). Inicialmente, un JWT actúa como un token de autenticación y no de autorización, verificando la identidad del usuario, pero sin proporcionar acceso directo a los recursos solicitados. Sin embargo, es importante resaltar que cuando un JWT contiene claims[1], proporciona información de autorización además de autenticación.

Con el objetivo de facilitar la transferencia segura y eficiente de la información de autenticación y autorización entre diferentes sistemas a través de la web, se utiliza JWS[1]. Un token en formato JSON generado por JWS contiene la información estructurada de la siguiente manera:

- Encabezado: información del tipo de token y del algoritmo de firma.
- Cuerpo: contiene los datos de la sesión, usuario, roles y permisos.
- Firma: para verificar la integridad del token.

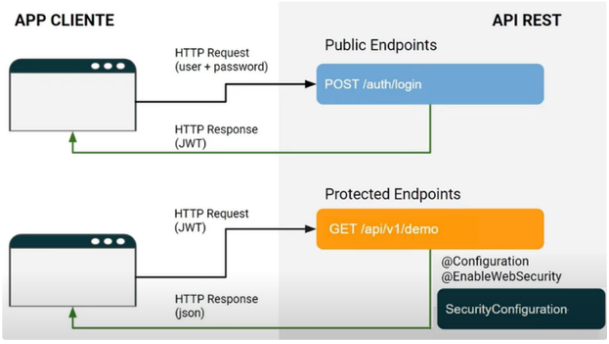
Por lo tanto, utilizando el patrón API Gateway y el patrón Token de Acceso proporcionado por *Spring Security*, las personas autorizadas pueden acceder de manera segura a las funcionalidades correspondientes. Esto se logra gracias a la firma digital, así como a la inclusión de los roles y permisos del usuario en el token. A continuación, se ilustra su arquitectura:



Fuente: https://www.youtube.com/watch?v=qiPh0yrDNas

Tal y como se puede observar, antes de llamar al controlador, el filtro de autenticación verifica las credenciales de la sesión a través del proveedor OAuth2Login y los datos obtenidos del usuario. Una vez autenticado el usuario, se almacena en el contexto de seguridad de *Spring Boot* y se envía al cliente para que este pueda utilizarlo en aquellas solicitudes que estén protegidas y requieran autenticación.

A continuación, se muestra el proceso de autenticación utilizando *Spring Security* y *JSON Web Tokens* (JWT), donde el usuario se loga en el sistema y accede al área del sistema en el que tiene acceso.

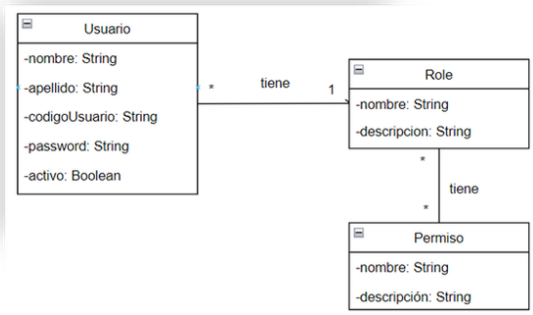


Fuente: https://www.youtube.com/watch?v=qiPh0yrDNas

Un punto a destacar es la importancia de configurar adecuadamente el intercambio de recursos de origen cruzado (CORS) para permitir solicitudes desde diferentes dominios, ya que es esencial para prevenir posibles problemas de acceso y garantizar un intercambio seguro y eficiente de la información entre sistemas de dominios diferentes.

Modelo de Dominio

Una vez se han especificado los diferentes permisos, se establece el modelo de dominio del subdominio relativo a la gestión de usuarios que se corresponderá con el servicio Usuario. Para ello, se tendrán en cuenta principalmente los nombres de las historias de usuario. Esto permitirá definir las operaciones del sistema.



Modelo de dominio del servicio de usuario

Operaciones del Sistema

Una vez definido el modelo de dominio, se identifican las solicitudes que podrá realizar un usuario, a estas solicitudes se les llamará operaciones y, en este caso, se tendrán en cuenta los verbos. Dado que se aplica el patrón CQRS para simplificar el modelo se tendrán dos tipos de operaciones: comandos y consultas. Además, estas operaciones se corresponderán con los endpoints de la aplicación.

Comandos:

Actor	Caso de uso	Nombre comando	Descripción
Administrador	Crear usuario	*crearUsuario()*	Crear un usuario activo que acceda a la aplicación.
Administrador	Modificar usuario	*modificarUsuario()*	Modificar los datos del usuario.
Administrador	Dar Baja Usuario	*bajaUsuario()*	Cambiar el estado de activo a inactivo.
Sistema	Generar Contraseña Temporal	*generarPasswordTemporal()*	Generar una contraseña.
Administrador	Crear Rol	*crearRol()*	Crear un rol.
Administrador	Eliminar Rol	*eliminarRol()*	Eliminar un rol.
Administrador	Asignar Permiso Rol	*asignarPermisoRol()*	Asignar un permiso a un rol.
Administrador	Quitar Permiso Rol	*quitarPermisoRol()*	Quitar un permiso a un rol.

Comandos Gestión de Usuarios

Consultas:

Actor	Caso de uso	Nombre comando	Descripción
Administrador	Ver Lista Usuarios	*consultarUsuarios()*	Recupera la información de los usuarios.
Administrador	Buscar Usuario por código	*BuscarUsuarioPorCodigo()*	Recupera la información de un usuario por el código de usuario.
Administrador	Ver Lista Roles	*consultarRoles()*	Recuperar los roles.

Consultas Gestión de Usuarios

A continuación, se muestra el diagrama de casos de uso que permite visualizar más claramente la iteración entre el administrador y el sistema.

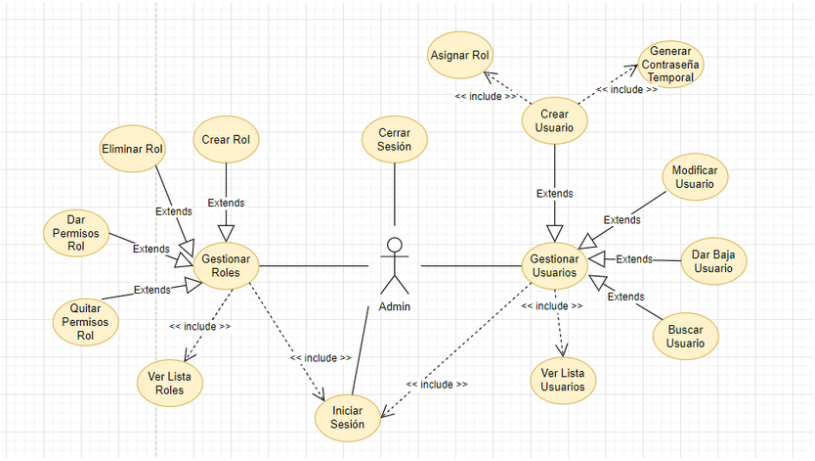


Diagrama de casos de uso de la gestión de usuarios y roles

Especificación de los comandos

En la especificación se definen los parámetros del comando, los valores que retornará y el comportamiento que tendrá, es decir, las precondiciones y postcondiciones de la clase.

Operación	***crearUsuario***(nombre, apellido, codigoUsuario, *password*, rolId)

Retorno	usuarioId
Precondiciones	El usuario no existe en el sistema. El rol existe en el sistema.
Post-condiciones	El usuario se crea con estado ACTIVO. La contraseña se almacena encriptada.

Operación	***bajaUsuario***(usuarioId)
Retorno	True/False
Precondiciones	El usuario existe en el sistema. El estado del usuario es ACTIVO.
Post-condiciones	El estado del usuario se cambia a INACTIVO.

Operación	***generarPasswordTemporal***()
Retorno	passwordTemporal
Precondiciones	-
Post-condiciones	El sistema generara una *password* temporal.

Operación	***crearRol***(nombre, descripción)
Retorno	rolId
Precondiciones	El rol no existe en el sistema.
Post-condiciones	El rol se crea en el sistema.

Operación	***eliminarRol***(rolId)
Retorno	True/False
Precondiciones	El rol existe en el sistema. El rol no está asignado a ningún usuario.
Post-condiciones	Los permisos del rol se quitan de la lista del rol. El rol se elimina del sistema.

Operación	***asignarPermisoRol***(rolId, permisoId)
Retorno	True/False
Precondiciones	El rol existe en el sistema. El permiso existe en el sistema.
Post-condiciones	Se añade el permiso a la lista de permisos del rol.

Operación	***quitarPermisoRol***(rolId, permisoId)
Retorno	True/False
Precondiciones	El rol existe en el sistema. El permiso existe en el sistema.
Post-condiciones	Se elimina el permiso de la lista de permisos del rol.

Diseño

Base de Datos

Un aspecto clave del diseño es la relación entre los diferentes elementos que conforman la gestión de usuarios. Por lo tanto, se proporciona el diagrama relacional que muestra las relaciones entre los diferentes elementos permitiendo la gestión de usuarios roles y permisos de manera efectiva.

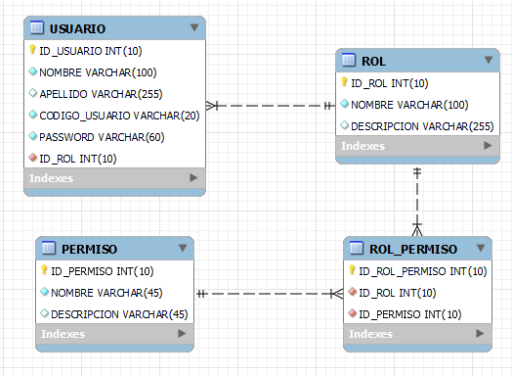


Diagrama relacional del modelo de datos de la gestión usuarios

Autenticación y Autorización

Del mismo modo, es importante comprender la iteración entre los componentes que formarán parte del proceso de autenticación y autorización, anteriormente explicado en detalle. De modo que, para facilitar su comprensión, se proporciona el siguiente diagrama de secuencia.

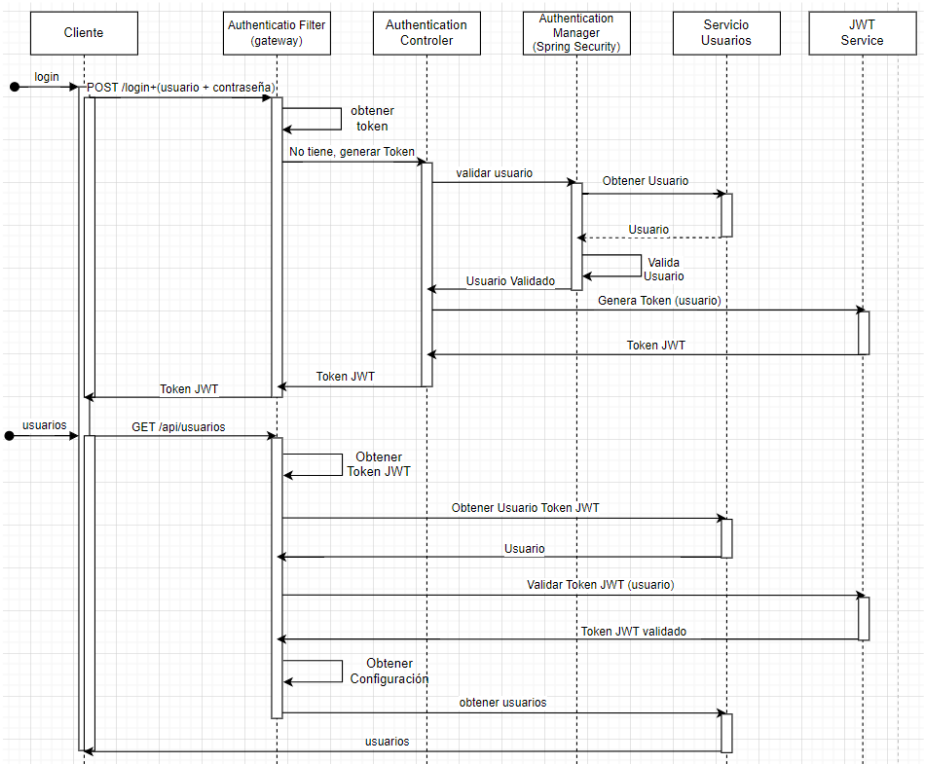


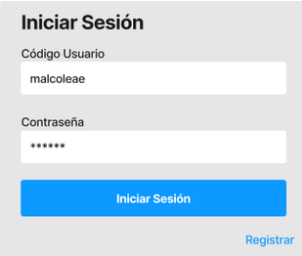
Diagrama de secuencia de autenticación y autorización

Interfaz de usuario

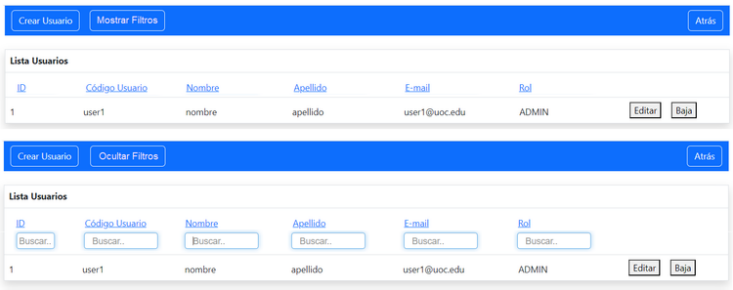
Finalmente, una vez clara la funcionalidad del sistema, se diseña la interfaz de usuario. Para ello, se consideran las restricciones impuestas por el *Product Owner*: la usabilidad del sistema debe ser intuitiva, fácil de usar y *responsive*. Estas directrices garantizarán una gran experiencia de usuario.

Con el objetivo de lograrlo, se ha realizado un prototipo de baja fidelidad en *Figma*[1] en el que se cumplen los siete principios del diseño universal[2].

La elección de que sea de baja fidelidad es porque es fácil de crearlo y permite a los interesados evaluar la interfaz antes de realizar el desarrollo. A continuación, se muestran las pantallas del listado de usuarios y del inicio de sesión.



Pantalla del inicio de sesión



Pantalla de la lista de usuarios

[1] *Figma* es una herramienta que genera prototipos.

[2] Ronald L. Mace acuñó el término diseño universal para referirse al tipo de diseño que cualquier persona puede entender y usar. Este diseño define siete principios que se tienen que cumplir: igualdad de uso, flexibilidad, simple e intuitivo, información fácil de percibir, tolerante a errores, escaso esfuerzo físico y dimensiones apropiadas

[1] Un claim proporciona información relativa a un usuario, permisos, roles y otros datos relevantes que se agrupan y certifican mediante un token de seguridad.

[2] JSON Web Signature (JWS) garantiza la integridad del contenido mediante firmas digitales o códigos de autenticación de mensajes en formato JSON.