# TRABAJO PRÁCTICO TÉCNICAS DE DISEÑO

GRUPO 8

Repositorio GitLab: https://gitlab.com/macohen/tp-tdd-docker

**INTEGRANTES DEL GRUPO**


| Nombre y Apellido | Padrón | E-mail |
|---|---|---|
| Alvaro Iribarren | 100812 | airibarren@fi.uba.ar |
| Martin Cohen | 101049 | macohen@fi.uba.ar |
| Maximiliano Petrucci | 95872 | mpetrucci@fi.uba.ar |
| Martín Pereira | 95793 | mgpereira@fi.uba.ar |

# ESCENARIOS

Especificaciones:

- **Casos de Uso**: Registrar usuario

  **Descripción**: Es la funcionalidad que les permite poder crearse un perfil en el sistema para poder utilizar el servicio de archivos.

  **Actores Principales**: Usuario

  **Pre-condiciones**: El usuario tiene que tener un email válido

  **Flujo Principal**:
  1. El usuario ingresa Nombre, Email y Contraseña
  2. El sistema chequea que no exista un usuario con el mismo email (**E1**)
  3. El sistema almacena el base de datos la información del usuario
  4. El sistema le genera al usuario un Token y un Refresh Token
  5. El sistema lo lleva a la pantalla que muestra sus archivos/carpetas
  6. Finaliza el caso de uso

  **Flujo de Excepción**:

  **E1**: Email ya existente

  1. El usuario ingresa un email que pertenece a otro usuario ya registrado en el sistema
  2. El sistema alerta al usuario que el email ingresado ya pertenece a un usuario registrado

  **Post-condiciones**:

  a. Queda asentado en el sistema el nuevo usuario registrado

- **Caso de Uso**: Log in usuario mediante cuenta existente

  **Descripción**: E es la funcionalidad que les permite ingresar a su perfil ingresando los datos de su cuenta personal para poder utilizar el servicio de archivos

  **Actores Principales**: Usuario

  **Pre-condiciones**: El usuario tiene que tener una cuenta creada en el sistema

  **Flujo Principal**:

  1. El usuario ingresa Email y Contraseña
  2. El sistema chequea que la contraseña concuerde con la perteneciente a la cuenta existente (E1)

3. El sistema le genera al usuario un Token y un Refresh Token
4. El sistema lo lleva a la pantalla que muestra sus archivos/carpetas
5. Finaliza el caso de uso

**Flujo de Excepción**:

**E1**: La contraseña no concuerda

1. El usuario ingresa una contraseña que no concuerda con la almacenada en base de datos
2. El sistema alerta al usuario que la contraseña ingresada es incorrecta

**Post-condiciones**:

- **Caso de Uso**: Log in usuario mediante autenticación Google

  **Descripción**: Es la funcionalidad que les permite ingresar a su perfil ingresando los datos de su cuenta de google para poder utilizar el servicio de archivos

  **Actores Principales**: Usuario

  **Pre-condiciones**: El usuario tiene que tener una cuenta creada en Google

  **Flujo Principal**:
  1. El usuario ingresa Email y Contraseña de su cuenta de Google
  2. El sistema de Google chequea que la contraseña concuerde con la perteneciente a la cuenta existente
  3. El sistema le genera al usuario un Token y Refresh Token
  4. El sistema lo lleva a la pantalla que muestra sus archivos/carpetas
  5. Finaliza el caso de uso

  **Post-condiciones**:

- **Caso de Uso**: Subir archivos

  **Descripción:** Es la funcionalidad que les permite subir archivos al almacenamiento personal de cada usuario

  **Actores Principales**: Usuario

  **Pre-condiciones**: El usuario tiene que estar logueado en el sistema

  **Flujo Principal**:

  1. El usuario selecciona la opción para subir un archivo
  2. El usuario selecciona el archivo que quiere subir

3. El sistema recibe la información del archivo y lo sube al almacenamiento personal del usuario
4. El sistema muestra una miniatura del archivo en pantalla
5. Finaliza el caso de uso

**Post-condiciones**:

a. Queda agregado en la base de datos del servicio de archivos el nuevo archivo
b. Queda asentado a quien le pertenece el nuevo archivo

- **Caso de Uso**: Crear carpeta

  **Descripció**n: Es la funcionalidad que les permite crear una carpeta donde pueden subir archivos para dejarlos dentro

  **Actores Principales**: Usuario

  **Pre-condiciones**: El usuario tiene que tener una cuenta creada en el sistema

  **Flujo Principal**:

  1. El usuario selecciona la opción para crear una carpeta
  2. El usuario ingresa el nombre de la carpeta
  3. El sistema crear la carpeta vacía y se la asigna al usuario
  4. El sistema muestra una miniatura de la carpeta en pantalla
  5. Finaliza el caso de uso

  **Post-condiciones**:

  a. Queda agregado en la base de datos del servicio de archivos la nueva carpeta
  b. Queda asentado a quien le pertenece la nueva carpeta

- **Caso de Uso**: Compartir archivo/carpeta

  **Descripción**: Es la funcionalidad que les permite seleccionar un archivo/carpeta y compartirlo con otro usuario y dependiendo del modo elegido podrán solamente verlo o hasta editarlo

  **Actores Principales**: Usuario

  **Pre-condiciones**: El usuario tiene que tener una cuenta creada en el sistema y tiene que haber subido un archivo o creado una carpeta

  **Flujo Principal**:

  1. El usuario selecciona el archivo o carpeta que quiere compartir
  2. El usuario elige a que usuarios quiere compartirle y el modo (lectura o lectura y escritura)

3. El sistema envía una invitación de compartida a los usuarios seleccionados
4. El sistema devuelve al usuario a su pantalla de almacenamiento personal
5. Finaliza el caso de uso

**Post-condiciones**:

a. Queda asentado en el sistema a quienes se le compartió el archivo/carpeta

- **Caso de Uso**: Borrar archivo/carpeta

**Descripción**: Es la funcionalidad que les permite borrar un archivo o, en caso de ser una carpeta, todos los archivos y carpetas que esta contenga.

**Actores Principales**: Usuario

**Pre-condiciones**: El usuario tiene que tener una cuenta creada en el sistema y tiene que haber subido un archivo o creado una carpeta

**Flujo Principal**:

1. El usuario selecciona el archivo o carpeta que quiere borrar
2. El usuario elige la opción de borrar
3. El sistema borrar el archivo del sistema y las relaciones que tuviera con otros usuarios
4. El sistema devuelve al usuario a su pantalla de almacenamiento personal
5. Finaliza el caso de uso

**Post-condiciones**:

a. Queda eliminado el archivo/carpeta de la base de datos del servicio de archivos
b. Quedan eliminada las relaciones entre el archivo/carpeta con los usuarios

- **Caso de Uso**: Buscar archivo/carpeta

**Descripción**: Es la funcionalidad que les permite buscar un archivo o carpeta por su nombre

**Actores Principales**: Usuario

**Pre-condiciones**: El usuario tiene que tener una cuenta creada en el sistema y tiene que haber subido un archivo o creado una carpeta

Flujo Principal:

1. El usuario ingresa el nombre del archivo/carpeta en el buscador que proporciona el servicio de archivos
2. El sistema busca entre los archivos/carpetas del usuario si concuerda el nombre con alguno

3. El sistema muestra por pantalla el/los archivos/carpetas que concuerdan o, en caso de no haber ninguno, muestra vacío
4. Finaliza el caso de uso

**Post-condiciones**:

- **Caso de Uso**: Ver invitaciones a archivos/carpetas

  **Descripción**: Es la funcionalidad que les permite ver sus invitaciones a archivos/carpetas compartidos por otros usuarios y poder aceptar o rechazar las mismas

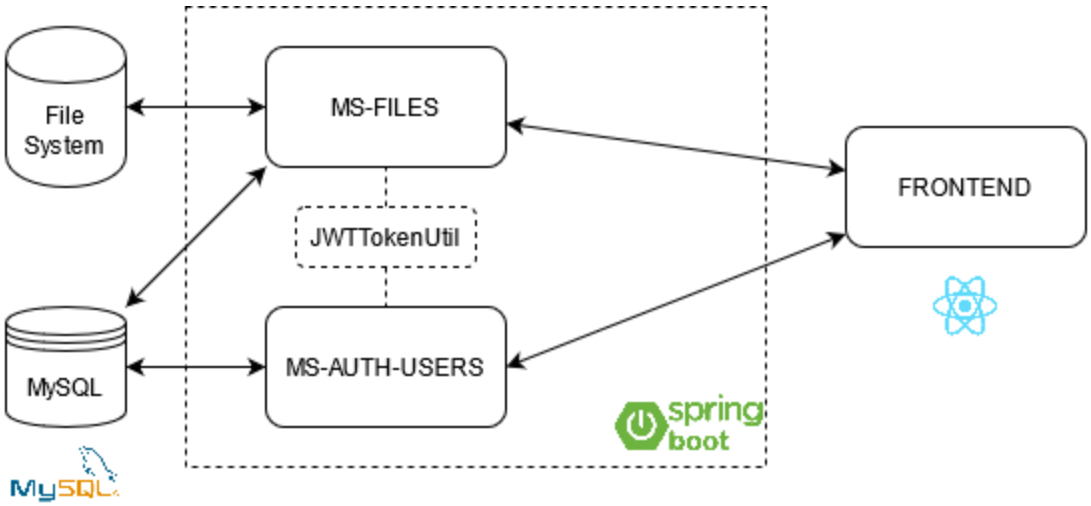
  **Actores Principales**: Usuario

  **Pre-condiciones**: El usuario tiene que tener una cuenta creada en el sistema

  **Flujo Principal**:
  1. El usuario ingresa a la sección donde tiene sus invitaciones
  2. El sistema muestra en pantalla todas las invitaciones recibidas
  3. El usuario puede aceptar o rechazar las invitaciones
  4. Si el usuario acepta la invitación:
     a. El usuario puede ver o editar el archivo/carpeta dependiendo del modo que hayan elegido
  5. Si el usuario rechaza la invitación:
     a. El sistema borra la invitación de la sección de invitaciones
  6. Finaliza el caso de uso

  **Post-condiciones**: Si el usuario aceptó la invitación:

  a. Queda asentado en el sistema que el usuario aceptó la invitación y se actualiza la información del archivo/carpeta para reflejar esto.




Teniendo en cuenta los escenarios mencionados anteriormente, pasamos a hacer una breve explicación de cómo decidimos plantear la arquitectura del proyecto.

Separamos el proyecto en 2 micro servicios: 1 para la parte de usuarios y autenticación y el otro para la parte del sistema de archivos. El micro servicio de archivos tiene su propia DB donde almacena los archivos e información relevante de los mismos. El de usuarios tiene su propia DB donde almacena información de los usuarios que se utiliza por ejemplo al momento de loguear/registrarse.

Nos pareció una buena idea separar las funcionalidades en distintos micro servicios, que se comuniquen entre ellos en caso de necesitar el uno al otro, porque tienen responsabilidades totalmente distintas y por temas de Interoperabilidad, Modificabilidad y Seguridad, los cuales creemos que son los atributos de calidad principales que queremos en nuestra arquitectura.

En un principio habíamos pensado en tener 3 micro servicios, uno para usuarios, otro para autenticación y otro para los archivos pero a medida que fuimos desarrollando los mismos, llegamos a la conclusión que el de usuarios y autenticación estaban muy relacionados y prácticamente se necesitaban comunicar muy seguido, por lo que nos pareció buena idea juntarlos.

A continuación pasaremos a explicar tecnologías que implementamos o utilizamos en los distintos micro servicios para resolver las normativas impuestas por el trabajo práctico a realizar.

## Micro servicio Autorización

Para la parte de autorización de usuarios para realizar requests a los micro servicios, nosotros decidimos utilizar 2 tokens JWT (JSON Web Token). Uno llamado “Token” y otro llamado “Refresh Token”.

Al momento de loguearse ya sea con la cuenta creada en el micro servicio o utilizando el log in por google, al usuario se le asignan los 2 tokens. El tiempo de expiración del Token es de 15 minutos, mientras que el del Refresh Token es de 1 año.

Cuando el usuario quiere realizar una acción y envía una request a uno de los micro servicios, el sistema chequea que el Token enviado pertenezca al usuario que quiere realizar la acción.

En caso de que el token no pertenezca a un usuario autorizado, no se permite el acceso.

En caso de que pertenezca y el Token no esté expirado, le deja realizar la acción.

En caso de que pertenezca y el Token este expirado, el sistema le avisa que esta expirado y el usuario le entrega su Refresh Token para recibir un nuevo Token con 15 minutos de duración nuevamente. Cuando sucede esto también se envía un nuevo Refresh Token válido por un año.

Cuando el usuario hace un log out de su cuenta se invalida el Refresh Token para que el usuario no pueda pedir nuevos tokens luego de que expire el Token que tiene actualmente.

## Front End

Para el front end utilizamos diversas librerías para facilitar el desarrollo del mismo que vamos a pasar a enumerar a continuación con su respectiva explicación de por qué las utilizamos.

Para la parte de los componentes y CSS utilizamos las librerías de Bootstrap y material UI que nos ahorraron el trabajo de implementar todo desde 0

Para la página de home, registro, log in y log out utilizamos react nativo.

Para la parte del File system lo que utilizamos fue Redux ya que nos ayudó a la fácil transferencia de los estados entre componentes, además de la facilidad que tiene para debugear.

Para el envío de requests al Backend utilizamos Axios ya que nos evitó tener que implementar toda la parte del abort, timeout que el fetch básico de React no tiene, además del parseo de la response que nos ayudó bastante. Además en Axios se utilizaron interceptores para poder realizar el flujo de la autenticación de forma transparente.

## File System

Se creó una interfaz definiendo los métodos principales para el manejo de un File System: save, get, delete y fileExists.

Con esto, se implementó una clase (LocalFS) la cual implementa a esta interface y se encarga de guardar los archivos dentro del disco local.

Si a futuro se quisiera guardar los archivos en otro almacenamiento, como por ejemplo S3, sólo se debe implementar la interfaz mencionada y desarrollar la clase que se encargará de guardar, obtener, borrar y comprobar si un archivo existe en S3

## Plugins

Se puso a disposición un Manager (PluginManager) el cual, durante su etapa de inicialización, se encarga de leer los plugins agregados que existan en la carpeta "C:/MSFILES7510/plugins" (se puede cambiar).

Estos plugins son proyectos Spring Boot con una compilación distinta (./gradlew jar) los cuales deben estar bajo el paquete “fiuba.tecnicas.plugins” para poder ser leídos por el manager.

La idea es que este PluginManager tenga un EndPoint para subir los archivos .jar hechos por los desarrolladores terceros y que estos puedan ser cargados desde fuera.

Dado que este Manager lee los archivos en su inicialización es necesario reiniciar este servicio cada vez que la gente sube proyectos nuevos.

A futuro, estaría bueno extraer de los otros micro servicios las entidades principales (como User o Node) para llevarlas a una dependencia la cual usen tanto el sistema principal como el plugin hecho por terceros.

# VISTA LÓGICA

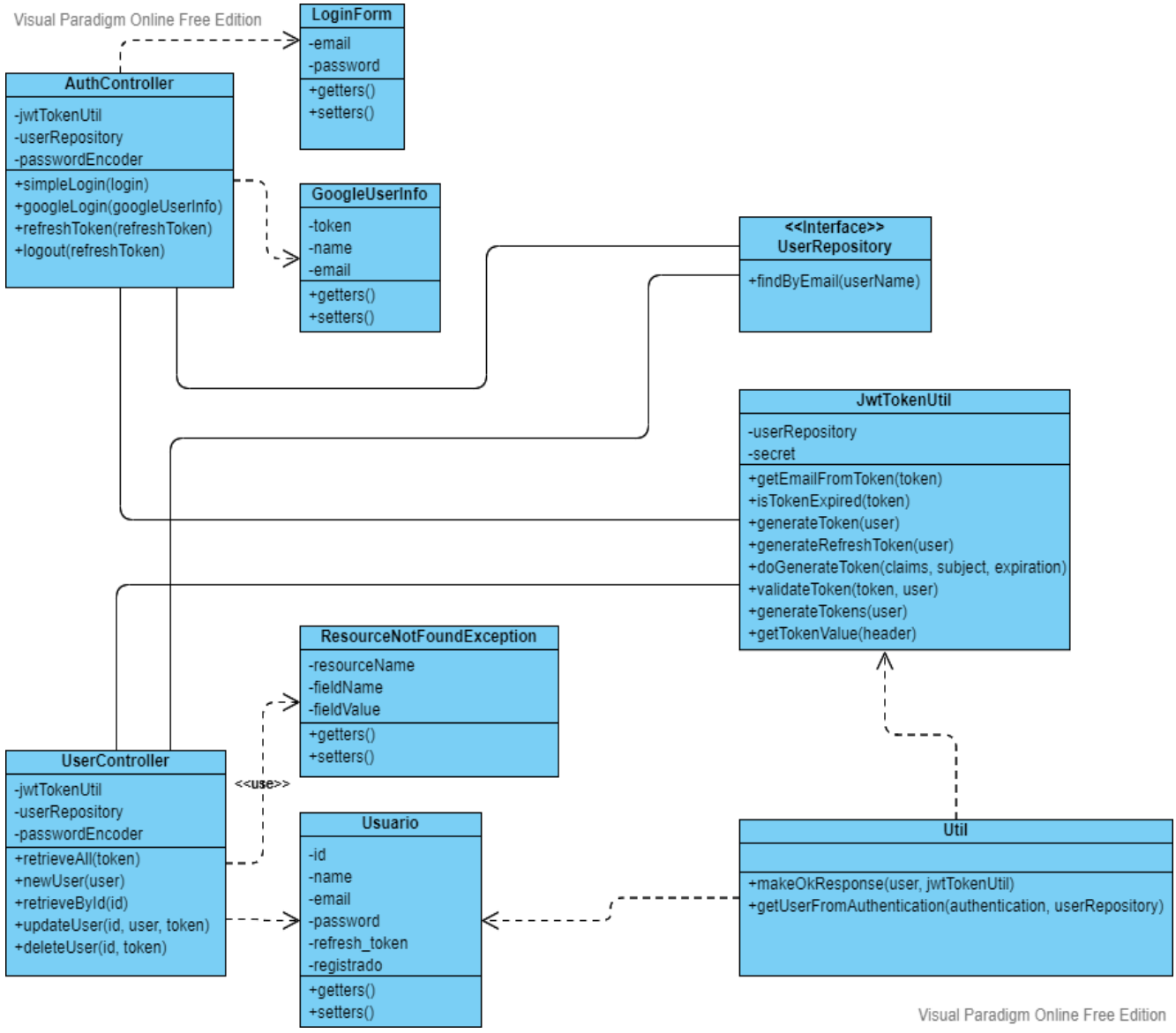


Diagrama de clases para micro servicio Usuarios y Autorización

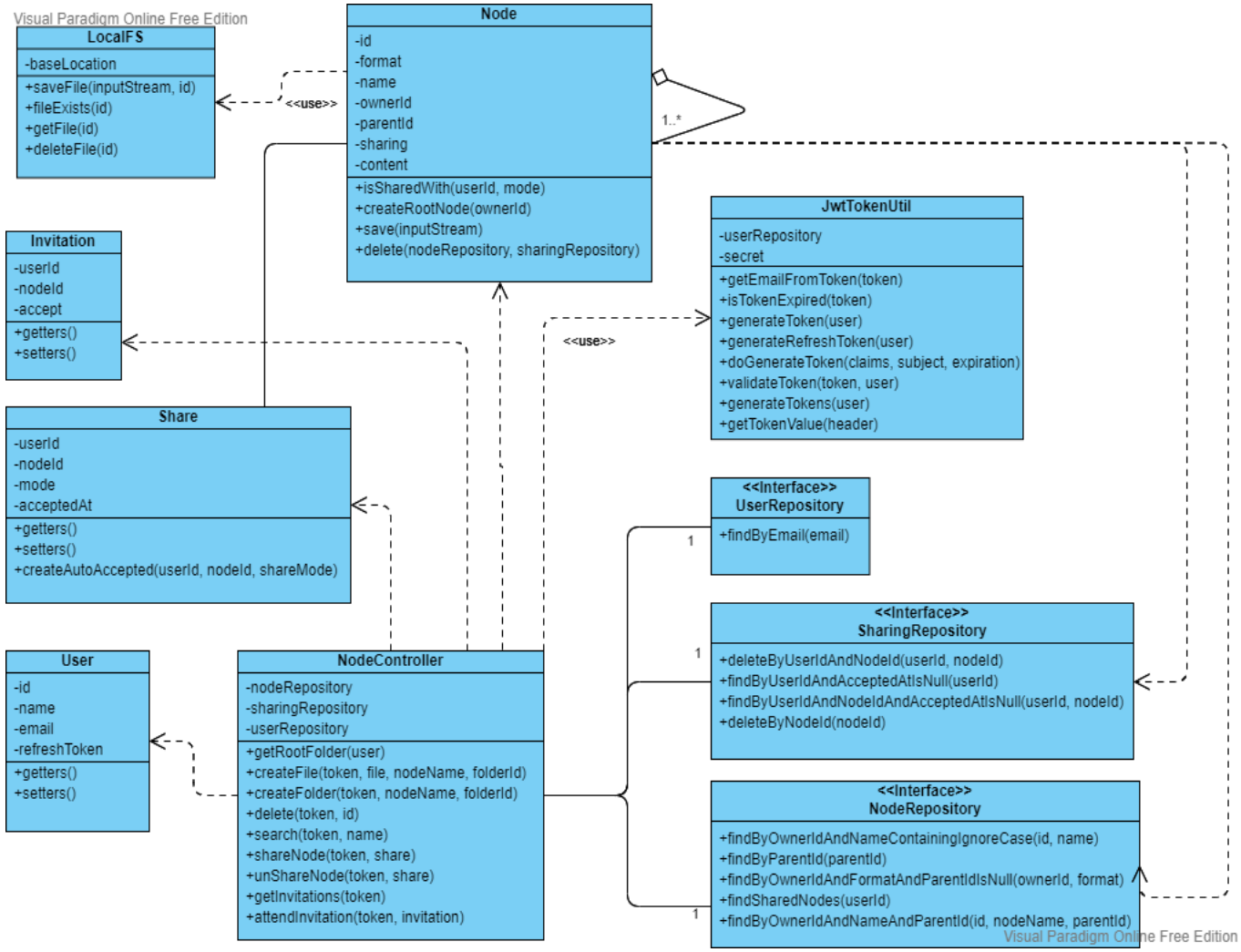



Diagrama de clases para micro servicio File System

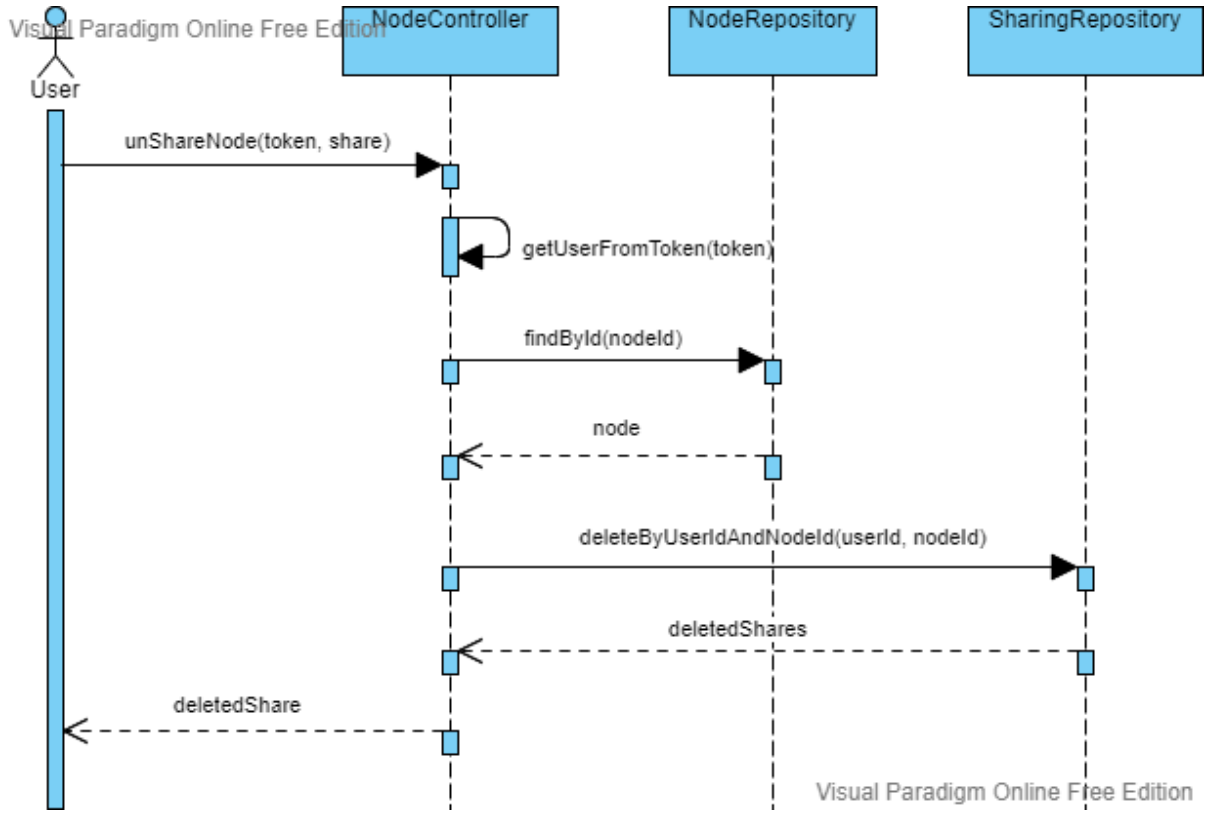



Diagrama de secuencia para caso de uso: Descompartir archivo/carpeta

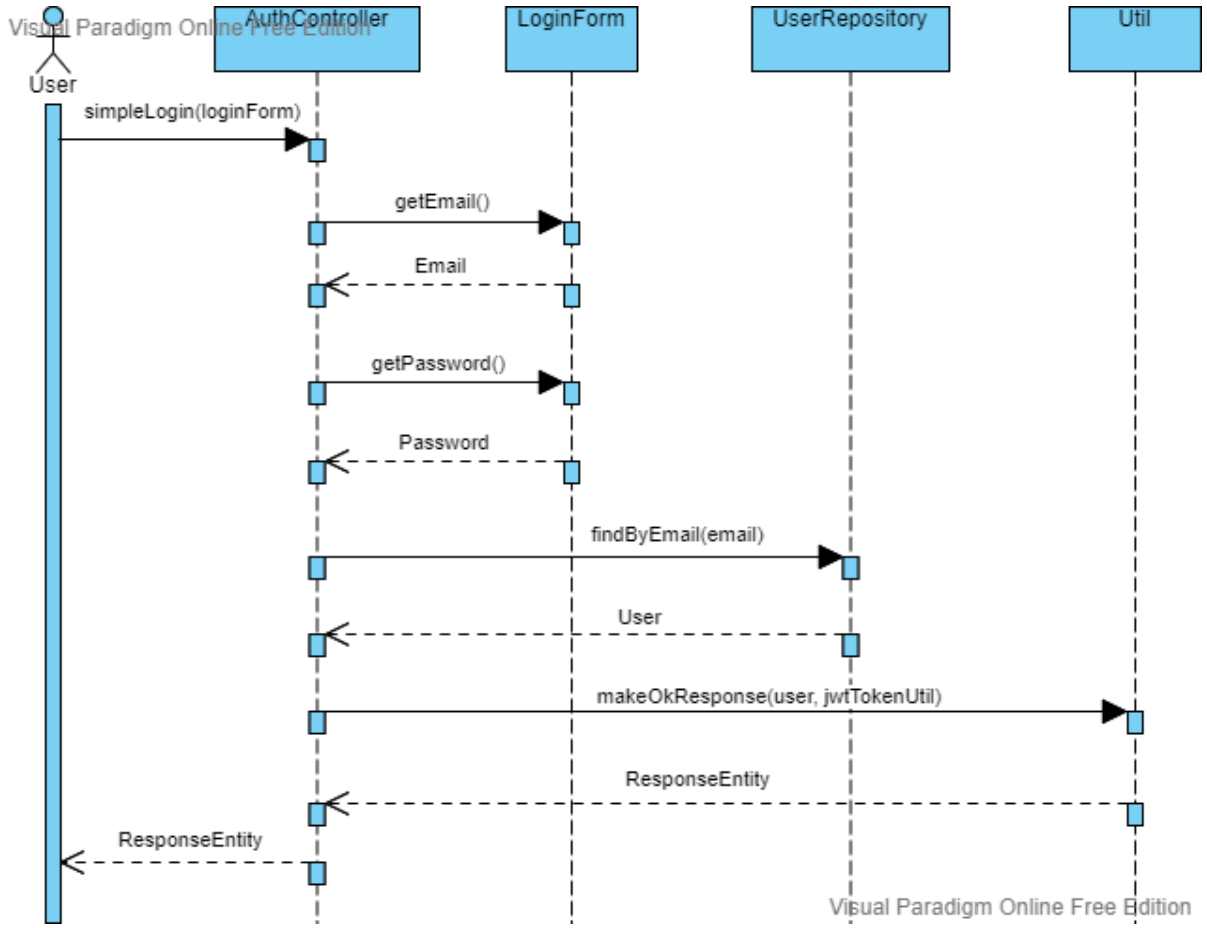



Diagrama de secuencia para caso de uso: Loguear usuario mediante cuenta existente

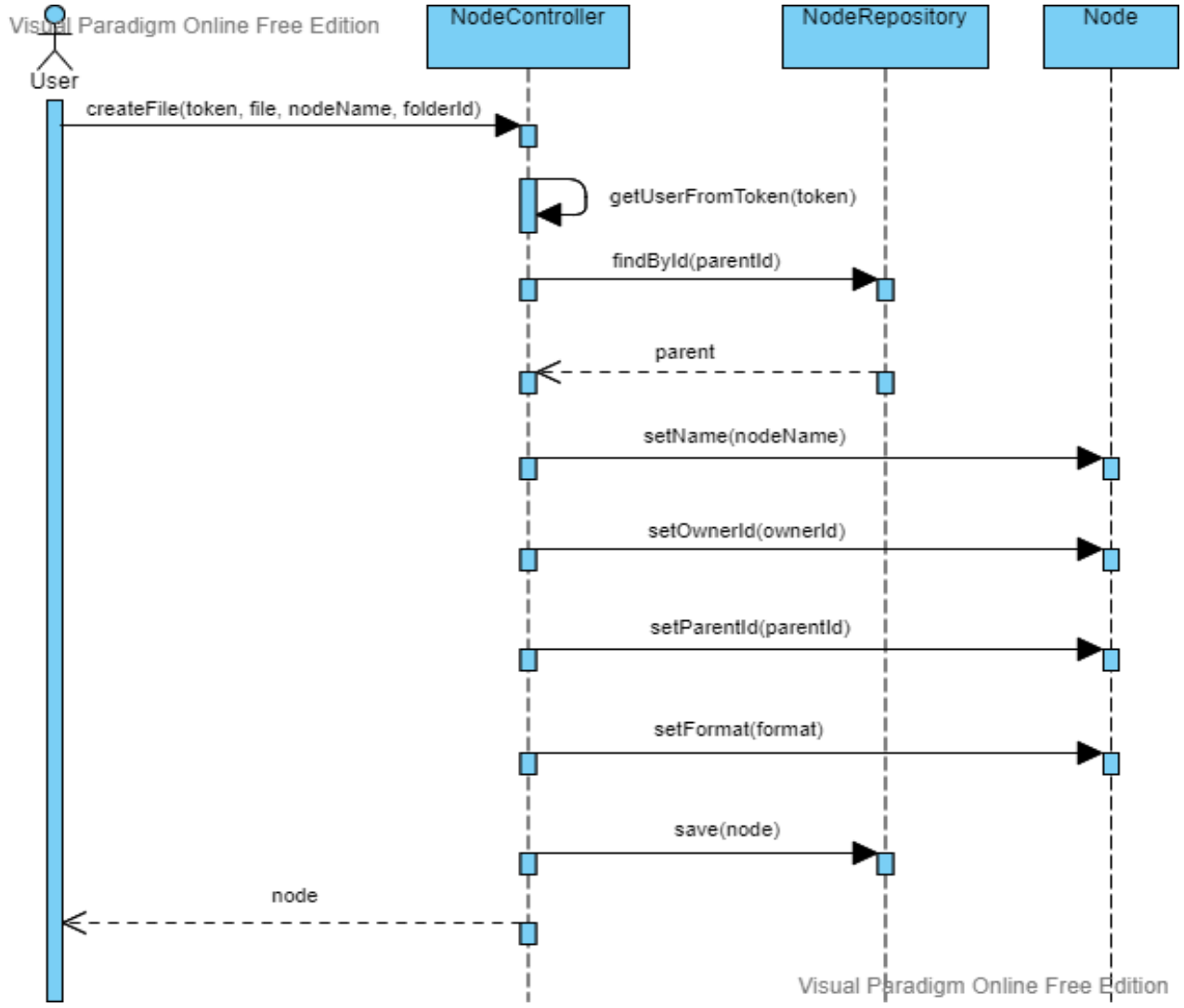



Diagrama de secuencia para caso de uso: Crear archivo

# VISTA DE PROCESOS

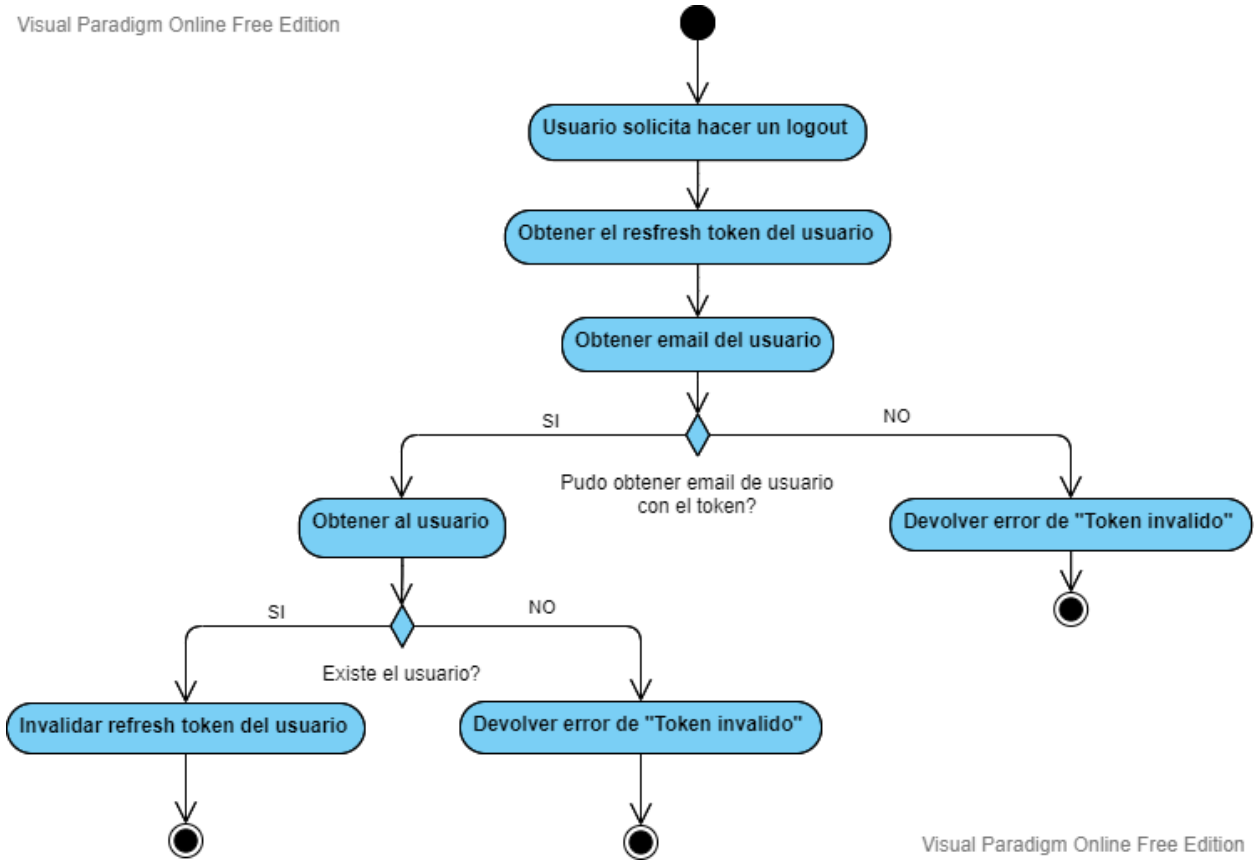



Diagrama de actividad para caso de uso: Log Out

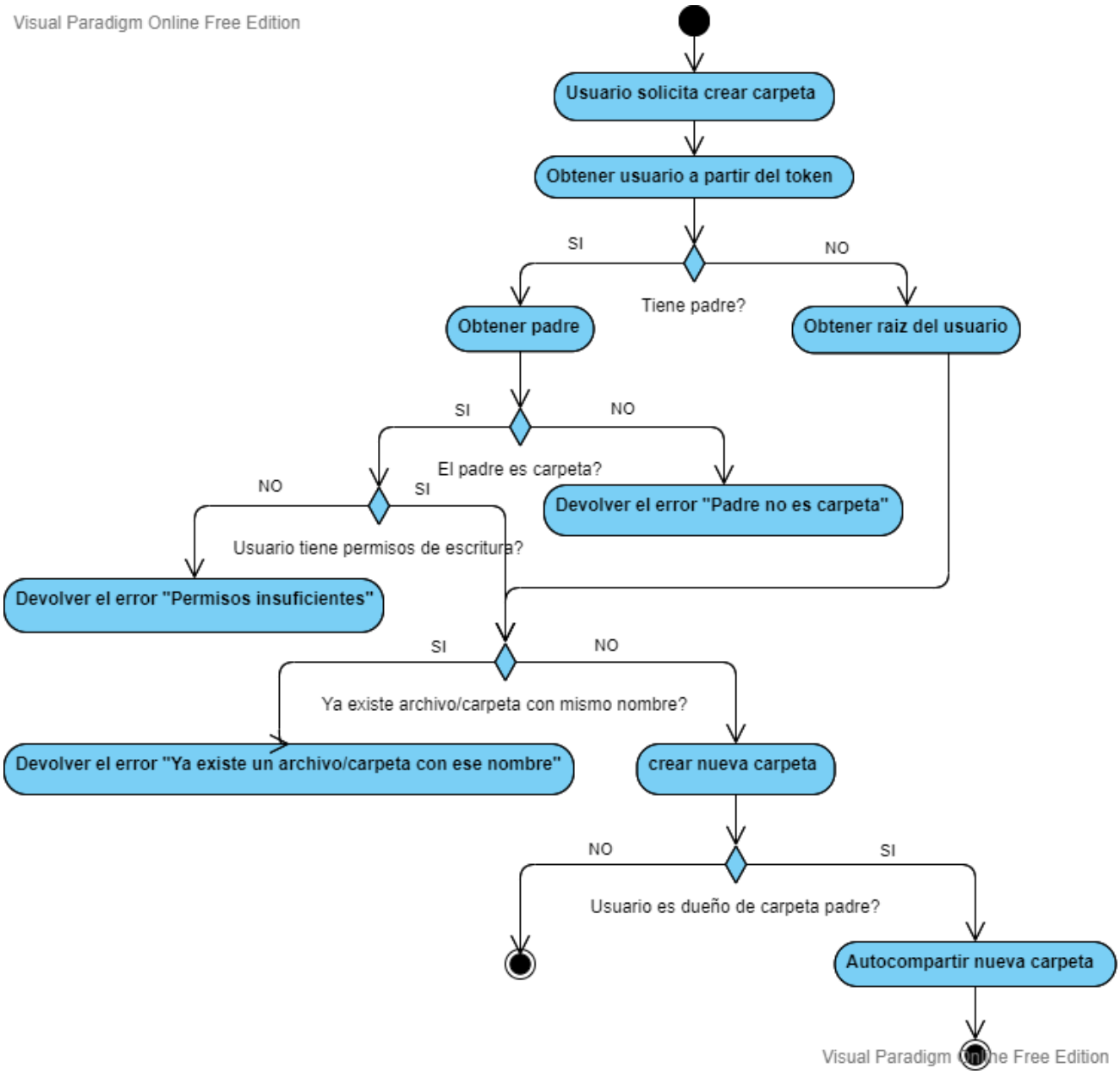



Diagrama de actividad para caso de uso: Crear carpeta

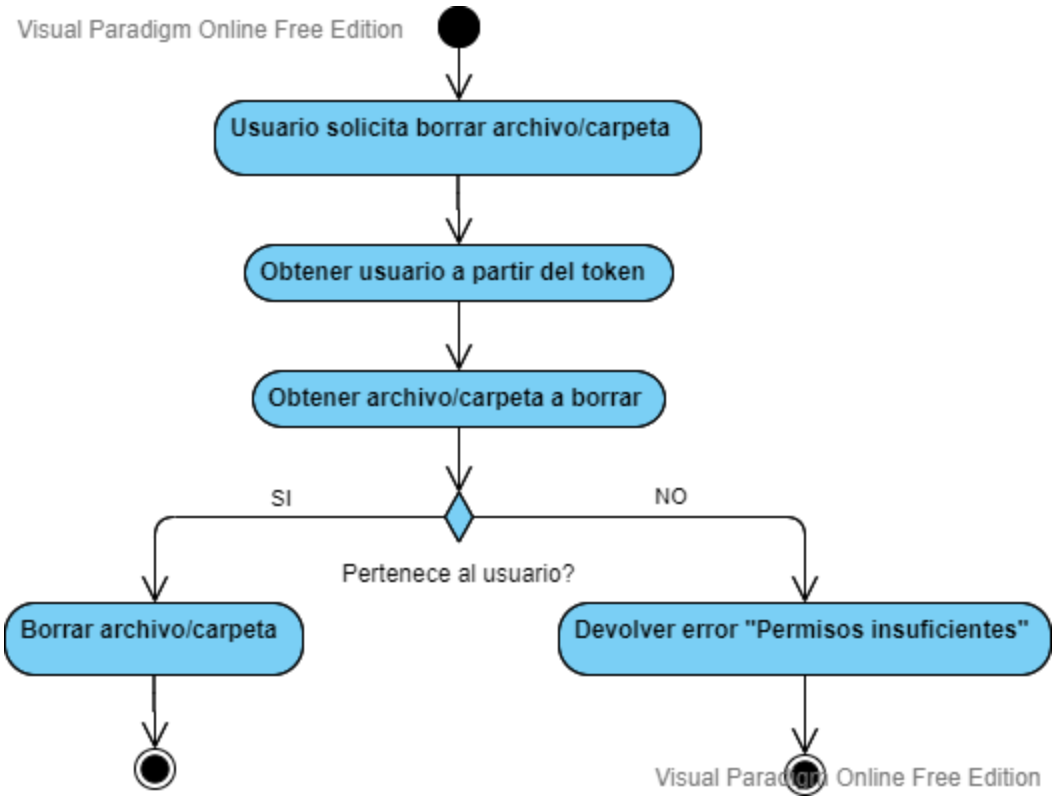



Diagrama de actividad para caso de uso: Borrar archivo/carpeta

# VISTA FÍSICA

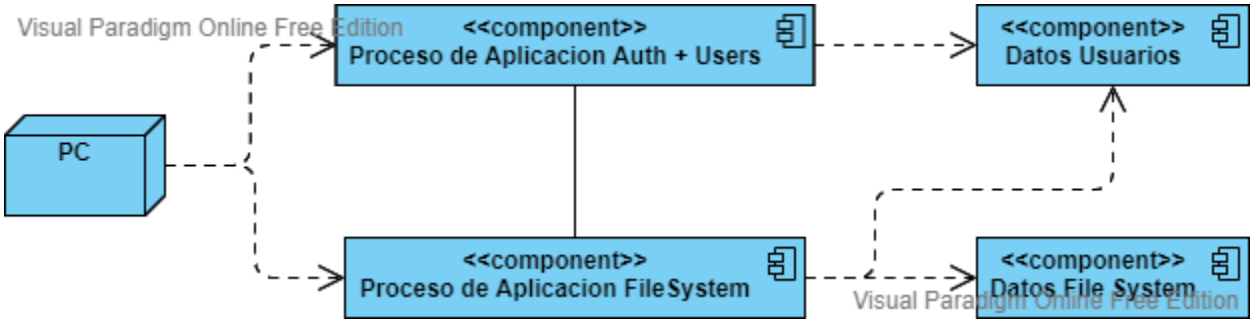



Diagrama de despliegue de los micro servicios

# VISTA DE DESARROLLO

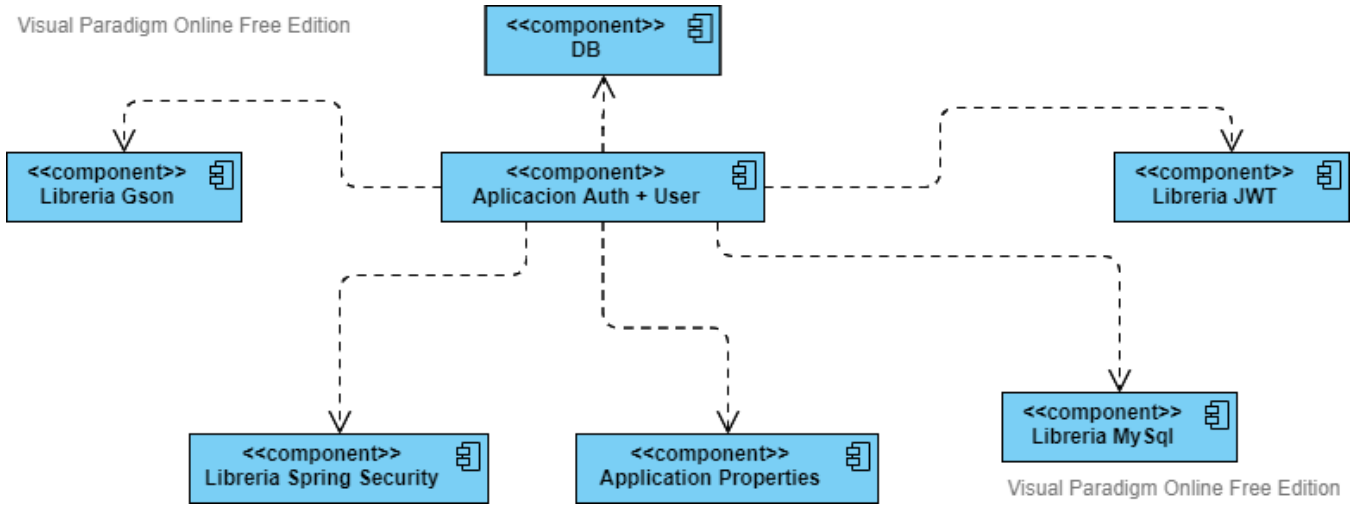



Diagrama de componentes micro servicio Usuarios y Autorización

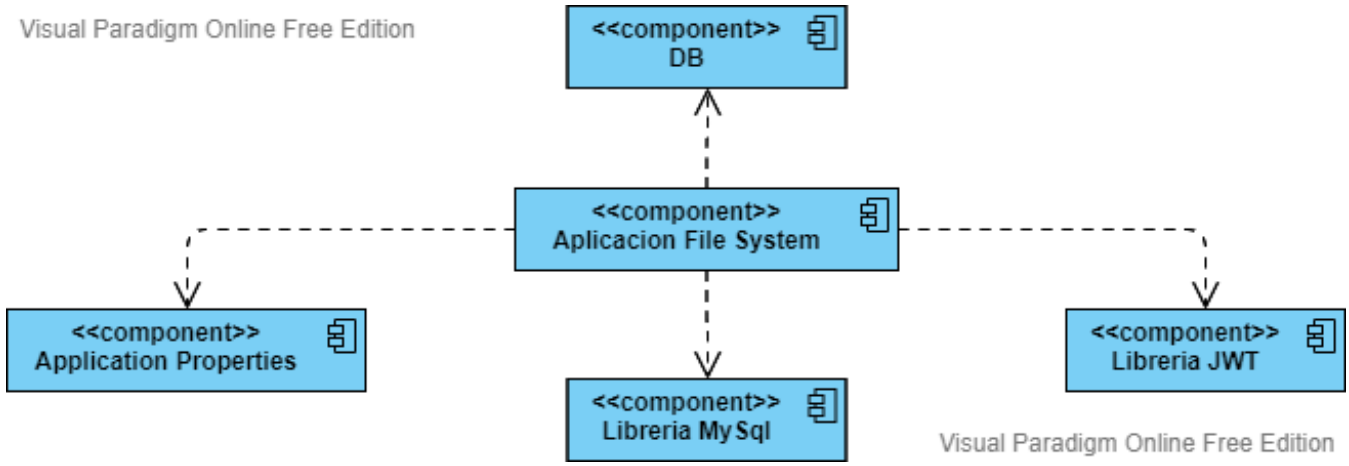



Diagrama de componentes micro servicio File System

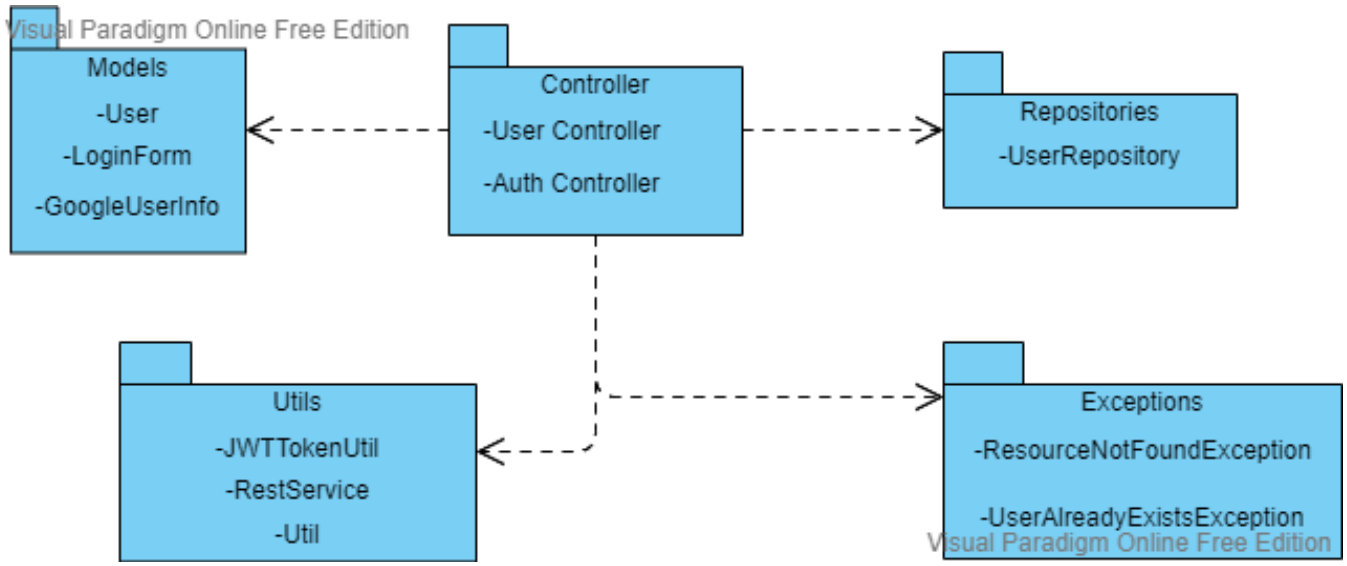



Diagrama de paquetes micro servicio Usuarios y Autorización

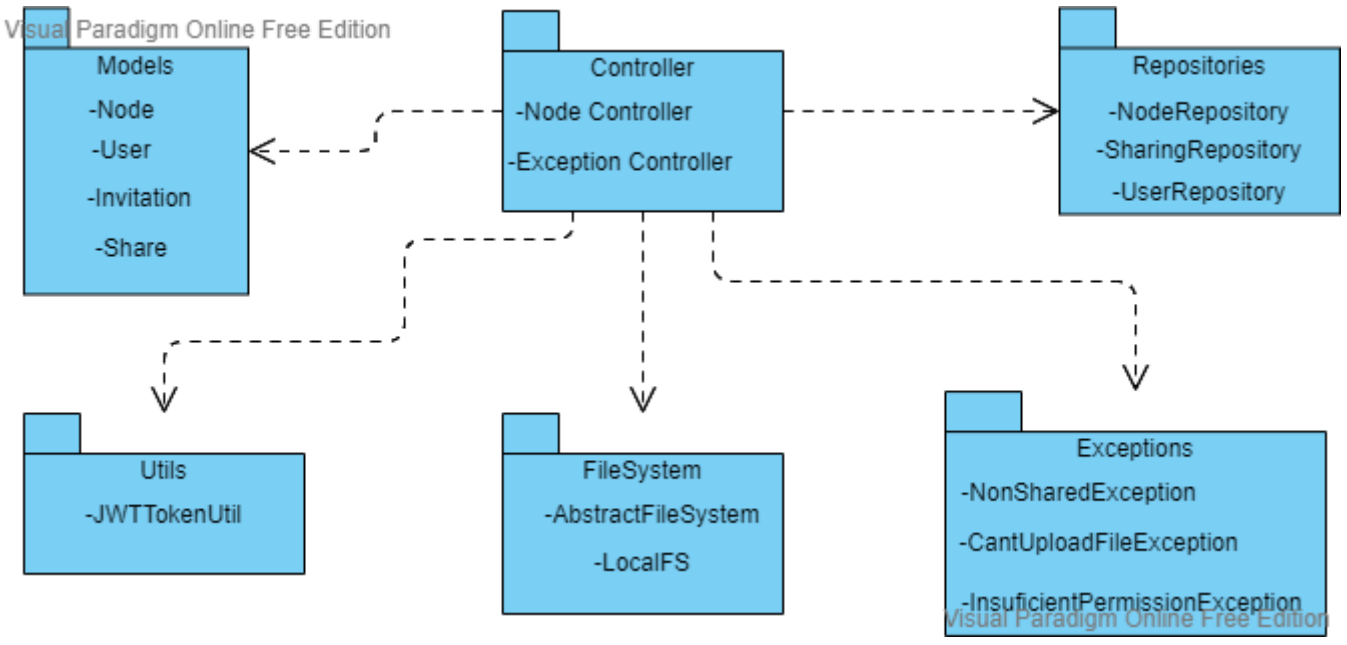



Diagrama de paquetes micro servicio File System