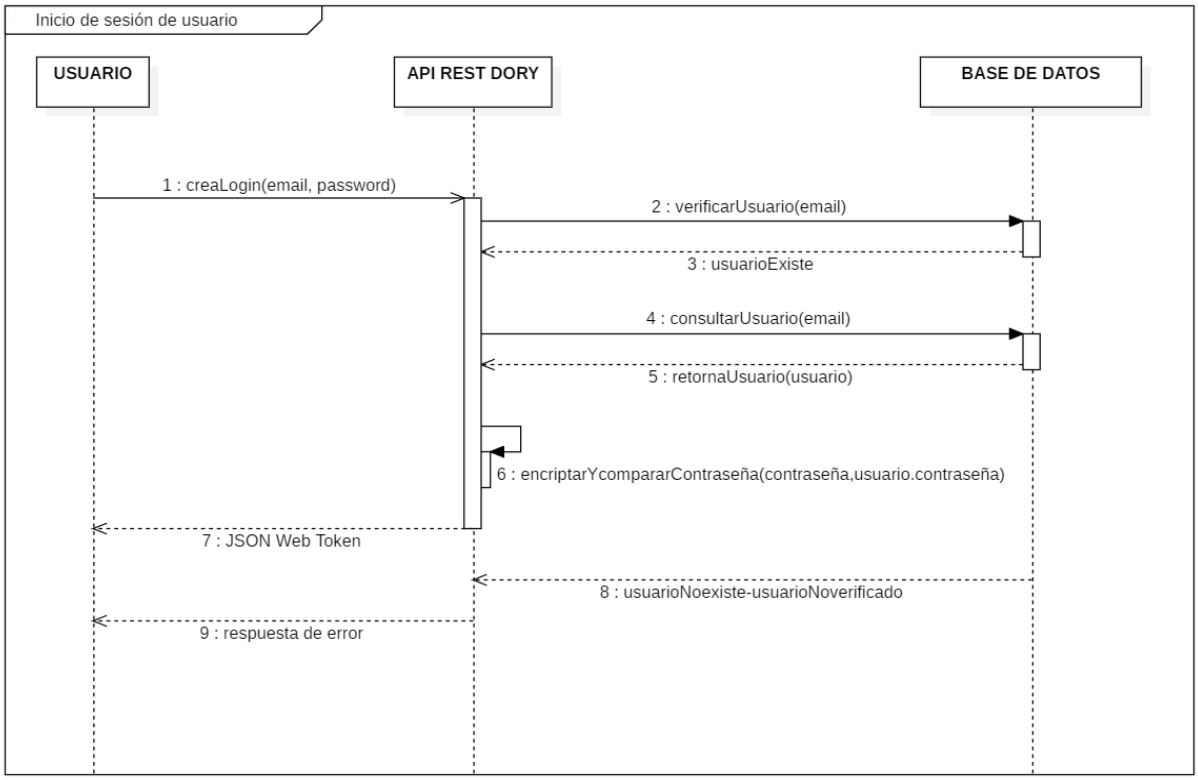
Como implementar el servicio de login ó inicio de sesión en el api rest de la plataforma web Dory

Objetivo:

Proporcionar una explicación de cómo se implementó el servicio de inicio de sesión en el api rest de la aplicación web Dory.



Recursos necesarios:

- Un servidor o plataforma en línea para implementar el inicio de sesión.
- Una base de datos para almacenar y gestionar la información de los usuarios.
- Usuario registrado de la base de datos

Piezas de software necesarias:

- Librería de encriptación de contraseñas (bcrypt) versión 5.0.1.
- Librería JWT-simple versión 0.5.6
- Librería para manejo de errores (http-errors) versión 2.0.0

Pasos:

1. Retomando la receta " implementación de servicio de inicio de sesión con Google en el api rest de la plataforma web Dory", en el servicio de inicio de sesión normal recibir los datos del usuario y verificar que los datos sean válidos (ver imagen 1).

```
async function createLogin(user){
  let email=user.email;
  let message = "El usuario no esta registrado o la contraseña es inválida";
  if(email!=undefined && email!=null)
  {
```

Recepción de los datos del usuario

Validación de email del usuario

Imagen 1. *Recepción de datos de usuario*

2. Verificar si el usuario este registrado (ver imagen 2).

```
let verificar = await db.query(
  `SELECT u.estaVerificado
  FROM usuarios as u
  WHERE u.estaVerificado=1 and
        u.email=?
  `,
  [email]
);
if(verificar.length<1){
  throw createError(404,"Usuario no verificado ó inexistete. Revise su correo electrónico y siga las instrucciones.");
}
```

Imagen 2. *Verificación de usuario*

3. Consultar el usuario en la base de datos de Dory (ver imagen 3).

```
const rows = await db.query(
  `SELECT u.password,u.email,u.id,tu.nombre_tipo_usuario,u.creadoCon
  FROM usuarios as u, tipos_usuarios as tu
  WHERE u.id_tipo_usuario=tu.id_tipo_usuario and
        u.email=?
  `,
  [email]
);
```

Imagen 3. *Consulta del usuario en la base de datos de Dory*

4. Si no se encuentra ningún resultado, mostrar un mensaje de error y terminar proceso (ver imagen 4).

```
throw createError(404,"Usuario no existe ó la contraseña es incorrecta.");
```

Imagen 4. *Mensaje de Error*

5. Si el usuario está verificado y existente. Obtener la contraseña almacenada y la contraseña proporcionada por el usuario, verificar si las contraseñas coinciden; si las contraseñas coinciden, generar un token de autenticación válido por un año y devolverlo junto con la información de cómo se creó el usuario (ver imagen 5).

Imagen 5. *Comparación de la contraseña del sistema con la ingresada por el usuario*

6. Si las contraseñas no coinciden, mostrar un mensaje y terminar proceso (ver imagen 4).

7. Si el usuario no está verificado, mostrar el mensaje de error por autorización y terminar proceso (ver imagen 6).

```
throw createError(401,message);
```

Imagen 6. *Mensaje de error*