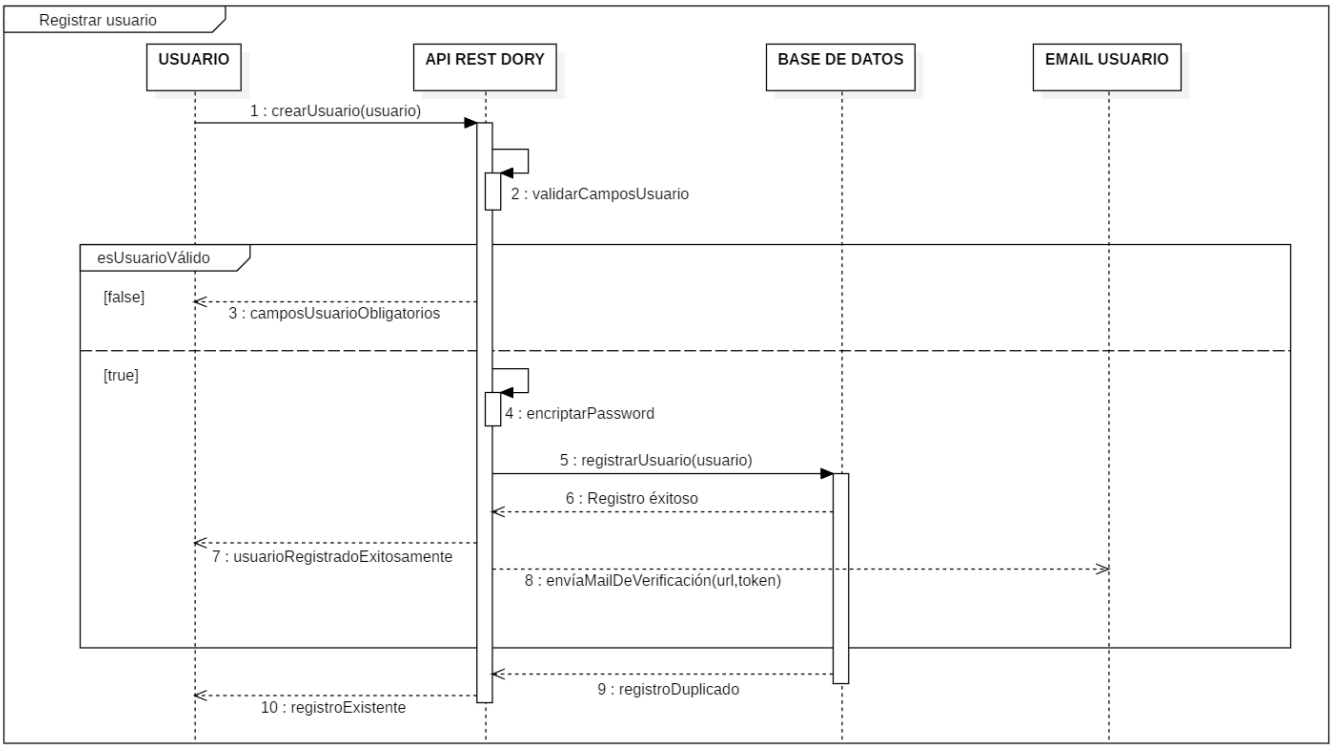
Como implementar el servicio de registro en el api rest de la aplicación web Dory

Objetivos:

Implementar el servicio de registro en el api rest de la plataforma web Dory.



Recursos necesarios:

- Node.js versión 16.14.1, Visual Studio Code versión 1.18.1, Express versión 4.17.1
- Cuenta de servicio SMTP de Google.
- API REST para la comunicación con la plataforma web y la base de datos.
- Base de datos para almacenar los datos de registro de los usuarios.
- Cliente de correo electrónico para recibir el mensaje de bienvenida.

Piezas de software necesarias:

- Librería de encriptación de contraseñas (bcrypt) versión 5.0.1.
- Librería para utilizar token (JWT- simple) versión 0.5.6
- Librería para manejo de errores (http-errors) versión 2.0.0
- Librería para envió de mensajes a correo electrónicos (nodemailer) versión 6.7.2

Pasos:

1. Retomando la receta " implementación de servicio de inicio de sesión o login del api rest de la plataforma web Dory", en el servicio de registro de usuario se debe recibir los datos del usuario y el token (ver imagen 1).

```
async function create(usuario,token){
        if(usuario.id_tipo_usuario == -1){
                if(token && validarToken(token)){
                        let payload=helper.parseJwt(token);
                        let rol= payload.rol;
                        if(rol!="Administrador"){
                                throw createError(401,"Usted no tiene autorización");
                        }
                }
        }
```

Imagen 1. Recepción de los datos *de usuario y token en servicio de registro*

2. En el archivo principal "index.js" se debe tener la configuración del servidor de Express (ver imagen 2).

```
const express = require('express');
const app = express();
const bodyParser = require('body-parser');
const port = process.env.PORT || 3000;
const {verifyToken, validarToken} = require ('./middelware/auth');
require('dotenv').config();

/*Socket.io con express*/
const http = require('http');
const server = http.createServer(app);
const { Server } = require("socket.io");
const io = new Server(server,{
  cors: {
    origin:"*.*"
  }
});

server.listen(port, () => {
  console.log('listening on *:'+port);
});
```

Imagen 2. *Configuración del servidor Express*

3. *Se debe tener implementado la ruta para registrar un usuario en el archivo principal "index.js" (ver imagen 3).*

```
const usuarioRouter = require('./routes/usuario');
app.use('/api/usuario', usuarioRouter)
```

Imagen 3. *Implementación de la ruta para el registro de usuario en el "index.js"*

4. Se debe tener definido la función "createToken" para generar un nuevo token de aplicación en el archivo "helper.js" (ver imagen 4).

```
function createToken (user,minutes) {
  var payload = {
  email:user.email,
  sub: user.id,
  rol: user.nombre_tipo_usuario,
  iat: moment().unix(),
  exp: moment().add(minutes, "minutes").unix(),
};
return  jwt.encode(payload, config.TOKEN_SECRET);
};
```

Imagen 4. Función "createToken" en archivo "helper.js"

5. Se debe tener definido la función "sendEmail" para el envío de mensajes al correo electrónico del usuario en el archivo "helper.js" (ver imagen 5).

```
function sendEmail(email,tema,contentHtml) {
  let transporter = nodemailer.createTransport({
  host: "smtp.gmail.com",
  port: 587,
  secure: false, // true for 465, false for other ports --color gris #343A40
  auth: {
  user: "plataforma.piscicola@gmail.com", // generated ethereal user
  pass: process.env.GOOGLE_PASSWORD_APP, // generated ethereal password
  },
  });
  // send mail with defined transport object
  let info = transporter.sendMail({
  from: tema+' <plataforma.piscicola@gmail.com>', // user
  to: email, //email---ginelect@unisucre.edu.co
  subject: tema, // Subject line
  html: contentHtml, // html body
  });

  let message = 'Enlace de'+tema+'enviado con éxito al correo eléctronico';
  return {message};
};
```

Imagen 5. Función "sendEmail" en archivo "helper.js"

6. Se debe tener el archivo "auth.js" creado en el módulo "middelware" ubicado en la raíz principal de proyecto.

7. En el archivo "auth.js" debe estar implementado la función para validar el token de usuario y ser exportada para ser usado en otros archivos (ver imagen 6).

```
const validarToken= function(token2){
  try {
      var token =token2.split(" ")[1];/*Obtenemos el token del string*/
      var payload = jwt.decode(token, config.TOKEN_SECRET);
      if (payload.exp <= moment().unix()) {
        return false;
      }
       return true;
  } catch (error) {
      console.log('Error:',error);
      return false;
  }
};

module.exports =
{
  verifyToken,
  validarToken
}
```

Imagen 6. *Función validarToken en el archivo "auth.js"*

8. En el módulo “routes” ubicado en la raíz principal del proyecto debe estar creado el archivo de rutas “usuario.js”.

9. En el archivo “usuario.js” debe estar creado el endpoint para registrar un usuario y debe tener el llamado al servicio para realizar dicho proceso (ver imagen 7).

```
router.post('/create', async function(req, res, next) {
  try {
    var token=req.headers.authorization;
    res.json(await usuario.create(req.body,token));
  } catch (err) {
    console.error(`Error al registrar el usuario`, err.message);
    next(err);
  }
});/* Create del usuario*/
```

Imagen 7. *Endpoint para registrar un usuario en el archivo de rutas “usuario.js”*

10. En el módulo “Services” ubicado en la raíz principal del proyecto, debe estar el archivo de servicios de usuarios llamado “usuario.js”.

11. En el archivo de servicios “usuario.js” se deben importar las librerías y archivos necesarios para la implementación del servicio de registro de usuario (ver imagen 8).

```
const db = require('./db');
const helper = require('../helper');
const config = require('../config');
var createError = require('http-errors');
const bcrypt= require('bcrypt');
const nodemailer = require('nodemailer');
const {validarToken} = require ('../middelware/auth');
```

Imagen 8. *Importación de librerías y archivos en el servicio de usuarios “usuario.js”*

12. El servicio para registrar al usuario se verifica el token y los datos del usuario (ver imagen 9).
    - Se verifica la autenticidad y validez de los datos enviados por el usuario.
    - Se procesa la solicitud y se extrae los datos del registro del usuario.
    - Se comprueba si el token pertenece a un administrador.

```
if(usuario.id_tipo_usuario == -1){
        if(token && validarToken(token)){
            let payload=helper.parseJwt(token);
            let rol= payload.rol;
            if(rol!="Administrador"){
                throw createError(401,"Usted no tiene autorización");
            }
        }
}
    if(usuario.nombres === undefined ||
      usuario.apellidos === undefined ||
      usuario.id_tipo_usuario === undefined ||
      usuario.email === undefined ||
      usuario.password === undefined ||
      usuario.foto === undefined ||
      usuario.id_departamento === undefined ||
      usuario.id_municipio === undefined ||
      usuario.latitud === undefined ||
      usuario.longitud === undefined ||
      usuario.creadoCon === undefined ||
      usuario.id_sexo === undefined ||
      usuario.id_etnia === undefined ){
        throw createError(400,"Debe enviar todos los datos requeridos para el registro del usuario");
    }
```

Imagen 9. *Verificación de token y datos del usuario*

13. Se realiza el contenido del mensaje que se enviará al correo electrónico del usuario para terminar su registro (ver imagen 10).

```
let contentHtml="";
let mensaje="Bienvenido(a), "+usuario.nombres+" "+"estamos emocionados de que te hayas registrado con nosotros, somos un equip
if(usuario.creadoCon && usuario.creadoCon=="google"){      You, hace 8 meses • Restricción del tipo de usuario administrador
        contentHtml = `<center>
        <img src="${process.env.URL_LOGO}" style='width: 100px; height: 100px;'/>
        <h2 style='color:grey'>Bienvenido a la plataforma piscícola Dory</h2>
        <p style='color:grey; text-align:justify; margin-bottom:20px'>${mensaje}</p>
        <form>
        </center>
        </br>
        `;
}else{
        let mensaje2="Sólo falta verificar su cuenta.   Has clic en el siguiente enlace para confirmar su correo electrónico";
        let token=helper.createToken(usuario,4320);/*token de 3 días*/
        usuario.creadoCon="email";
        contentHtml = `<center>
        <img src="${process.env.URL_LOGO}" style='width: 100px; height: 100px;' />
        <h2 style='color:grey'>Bienvenido a la plataforma piscícola Dory</h2>
        <p style='color:grey; text-align:justify; margin-bottom:20px'>${mensaje}</p>
        <p style='color:grey; text-align:justify; margin-bottom:20px'>${mensaje2}</p>
        <form>
        <a href="${process.env.DORY_WEB_APP_URL}/verify-account?token=${token}" style=" color:#ffffff; text-decoration:none;  bo
        </center>
        </br>
        `;
}
```

Imagen 10. *Contenido html del mensaje enviado al correo electrónico del usuario.*

14. Se encripta la contraseña del usuario (ver imagen 11).

```
try {
  const saltRounds= 10;
  const salt= bcrypt.genSaltSync(saltRounds);//generate a salt
  const passwordHash= bcrypt.hashSync( usuario.password , salt);//generate a password Hash (salt+hash)
  usuario.password=passwordHash;//Re-assign hashed generate a salt version over original, plain text password
  usuario.estaVerificado=0;
} catch  {
  throw createError(500,"Un problema al crear el usuario");
}
```

Imagen 11. *Encriptación de la contraseña del usuario*

15. Se realiza proceso del registro de los datos del usuario en la base de datos y se envía el mensaje de registro exitoso al correo electrónico (ver imagen 12); en caso contrario se envía un mensaje de error (ver imagen 13).

```
const result = await db.query(
  `INSERT INTO usuarios(nombres,apellidos,id_tipo_usuario,email,password,foto,id_departamento,id_municipio,latitud,longitu
  [
    usuario.nombres,
    usuario.apellidos,
    usuario.id_tipo_usuario,
    usuario.email,
    usuario.password,
    usuario.foto,
    usuario.id_departamento,
    usuario.id_municipio,
    usuario.latitud,
    usuario.longitud,
    usuario.estaVerificado,
    usuario.creadoCon,
    usuario.id_sexo,
    usuario.id_etnia
  ]
);
    if (result.affectedRows) {
            insertId=result.insertId;
            message = 'Usuario registrado exitosamente';
            let mensaje="Bienvenido(a), "+usuario.nombres+" "+"estamos emocionados de que te hayas registrado con nosotros
            let mensaje2="Sólo falta verificar su cuenta.   Has clic en el siguiente enlace para confirmar su correo elect
            let token=helper.createToken(usuario,4320);/*token de 3 días*/
            let tema="Bienvenido a Dory";
            helper.sendEmail(usuario.email,tema,contentHtml);
```

Imagen 12. Registro de usuario y envió de mensaje al correo electrónico

```
throw createError(500,"Ocurrió un problema al registrar un usuario");
```

Imagen 13. *Mensaje de error del registro*

16. El usuario recibe el mensaje de bienvenida en su correo electrónico (ver imagen 14).

Imagen 14. *Mensaje de bienvenida recibido en el email de usuario*