



**EDUCACIÓN**  
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLOGICO  
NACIONAL DE MEXICO

## **INSTITUTO TECNOLOGICO DE OAXACA**

### *INTEGRACION DE PROCESOS DE DESARROLLO DE SOFTWARE*

GRUPO: 9SA            HORARIO: 07:00-08:00

DOCENTE: Espinosa Pérez Jacob

ALUMNOS:

Méndez Mendoza Luisa Michel

Pérez de Jesús Edith

Martínez García Yahir Omar

López García Lourdes Gloria

PRESENTA:

**SEGUNDO REPORTE**

# INDICE

Bitácora .....	3
Endpoint registro/login con JWT .....	4
Crear pantalla de registro .....	7
Crear pantalla de login .....	8
Conectar frontend de registro con backend.....	9
Conectar frontend de login con backend.....	12
Modelo de datos “usuario” .....	12
Pruebas iniciales de flujo login/registro.....	12
Correcciones y pruebas.....	12

# BITACORA

ITERACION 1 AUTENTIFICACION							
Actividad	Descripción	Responsable	Estatus	Resultado	Fecha inicio	Fecha fin	Observaciones
Endpoint registro/login con JWT	Se implementaron endpoints en backend para registrar e iniciar sesión de usuarios. Generan un token JWT y cuentan con middleware para proteger rutas privadas.	Michel	Finalizada 	Endpoints /api/auth/register y /api/auth/login funcionando, con JWT incluido.	01/10/25	02/10/25	No hubo problemas. Se validó la generación y verificación de tokens.
Crear pantalla de registro	Se desarrolló el formulario en React para registrar usuarios con nombre, email, contraseña y país.	Lourdes	Finalizada 	Componente AuthForm en modo registro implementado.	01/10/25	02/10/25	Se tenía pensado preguntar al inicio la edad del usuario, pero se optó en cambiarlo por país y la edad se incluirá al momento de querer ver un video
Crear pantalla de login	Se diseñó la interfaz en React para login de usuario con email y contraseña. Se agregó opción de inicio de sesión con Google.	Yahir	Finalizada 	Pantalla de login funcional con AuthForm y botón de Google Login.	01/10/25	02/10/25	Inicio de sesión con Google integrado mediante OAuth.
Conectar frontend de registro con backend	Se conectó el formulario de registro al endpoint del backend usando Axios.	Yahir	Finalizada 	Registro exitoso y guardado en MongoDB.	01/10/25	02/10/25	El sistema devuelve mensaje "Registro exitoso" al completar el flujo.
Conectar frontend de login con backend	Se conectó el formulario de login al backend. Se guarda el token JWT en localStorage y se autentica la sesión.	Edith	Finalizada 	Login funcional, con token almacenado y navegación a Home.	01/10/25	02/10/25	Incluye login con Google (/api/auth/google).
Modelo de datos "Usuario"	Se definió el modelo en Mongoose con validaciones para email, encriptación de contraseña y almacenamiento de historial de videos.	Michel	Finalizada 	Archivo User.js implementado con pre('save') y matchPassword.	01/10/25	02/10/25	Ninguna observación

Pruebas iniciales de flujo login/registro	Se realizaron pruebas del flujo completo de registro y login, validando mensajes de éxito y errores de credenciales.	Todos	Finalizada 	Pruebas manuales ejecutadas correctamente.	01/10/25	02/10/25	Se verificó el almacenamiento del token y protección de rutas privadas.
Correcciones y pruebas	Se ajustaron mensajes de error, manejo de token y validaciones en frontend.	Todos	Finalizada 	Código corregido en AuthPage.js y AuthForm.js.	01/10/25	02/10/25	Agregar validaciones en contraseña y colocar botón de visualización de la misma

## EDPOINT REGISTRO/LOGIN CON JWT

```
// Función auxiliar para generar JWT
const generateToken = (id) => {
  return jwt.sign({ id }, process.env.JWT_SECRET, {
    expiresIn: '30d',
  });
};

exports.registerUser = async (req, res) => {
  const { username, email, password, country } = req.body;

  try {
    const userExists = await User.findOne({ email });
    if (userExists) {
      return res.status(400).json({ message: 'El usuario ya existe con este email.' });
    }

    const user = await User.create({
      username,
      email,
      password,
      country
    });

    if (user) {
      res.status(201).json({
        _id: user._id,
        username: user.username,
        email: user.email,
        country: user.country,
        token: generateToken(user._id),
      });
    } else {
      res.status(400).json({ message: 'Datos de usuario inválidos.' });
    }
  } catch (error) {
    console.error('Error al registrar usuario:', error);
    res.status(500).json({ message: 'Error del servidor al registrar.' });
  }
};
```

```
exports.loginUser = async (req, res) => {
  const { email, password } = req.body;

  try {
    const user = await User.findOne({ email });

    if (user && (await user.matchPassword(password))) {
      res.json({
        _id: user._id,
        username: user.username,
        email: user.email,
        country: user.country,
        token: generateToken(user._id),
      });
    } else {
      res.status(401).json({ message: 'Email o contraseña inválidos.' });
    }
  } catch (error) {
    console.error('Error al iniciar sesión:', error);
    res.status(500).json({ message: 'Error del servidor al iniciar sesión.' });
  }
};
```

## CREAR PANTALLA DE LOGIN

Se implementa registro, login con email/contraseña y login con Google.

Al autenticar, guarda el token JWT para mantener la sesión.

Se manejan mensajes de error para guiar al usuario.

```
const API_URL = 'http://localhost:5000/api/auth';

const handleSubmit = async (e) => {
    e.preventDefault();
    setMessage('');
    setError('');

    try {
        let res;
        if (isRegister) {
            res = await axios.post(` ${API_URL}/register`, { username, email, password, country });
            setMessage('Registro exitoso! Ya puedes iniciar sesión.');
            setIsRegister(false);
        } else {
            res = await axios.post(` ${API_URL}/login`, { email, password });
            setMessage('Inicio de sesión exitoso!');
            localStorage.setItem('token', res.data.token);
            setAuthToken(res.data.token);
            navigate('/');
        }
    } catch (err) {
        console.error('Error de autenticación:', err.response ? err.response.data : err.message);
        setError(err.response && err.response.data.message ? err.response.data.message : 'Error de red o de conexión');
    }
};
```

```
// Función para manejar el éxito del login con Google
const handleGoogleSuccess = async (credentialResponse) => {
  const idToken = credentialResponse.credential;
  try {
    // Envía el token de Google a tu backend para verificación y login/registro
    const res = await axios.post(`#${API_URL}/google`, {
      token: idToken,
    });

    setMessage('Inicio de sesión con Google exitoso!');
    localStorage.setItem('token', res.data.token); // Guarda el token de TU backend
    setAuthToken(res.data.token);
    navigate('/'); // Redirige a la página de inicio
  } catch (err) {
    console.error('Error en el login con Google:', err);
    setError('No se pudo iniciar sesión con Google. Inténtalo de nuevo.');
  }
};

// Función para manejar el error del login con Google
const handleGoogleFailure = () => {
  console.error('El inicio de sesión con Google ha fallado.');
  setError('El inicio de sesión con Google ha fallado. Por favor, intenta de nuevo.');
};
```

## *CONCETAR FRONTEND DE REGISTRO CON BACKEND*

```
if (isRegister) {  
    res = await axios.post(`#${API_URL}/register`, { username, email, password, country });  
    setMessage('Registro exitoso! Ya puedes iniciar sesión.');// @access Public  
    setIsRegister(false);  
    router.post('/login', loginUser);  
}
```

## *CONCETAR FRONTEND DE LOGIN CON BACKEND*

```
} else {  
    res = await axios.post(`#${API_URL}/login`, { email, password });  
    setMessage('Inicio de sesión exitoso!');  
    localStorage.setItem('token', res.data.token);  
    setAuthToken(res.data.token);  
    navigate('/');// @access Public  
    router.post('/login', loginUser);  
}
```

```
// Función para manejar el éxito del login con Google
const handleGoogleSuccess = async (credentialResponse) => {
  const idToken = credentialResponse.credential;
  try {
    // Envía el token de Google a tu backend para verificación y login/registro
    const res = await axios.post(`#${API_URL}/google`, {
      token: idToken,
    });

    setMessage('Inicio de sesión con Google exitoso!');
    localStorage.setItem('token', res.data.token); // Guarda el token de TU backend
    setAuthToken(res.data.token);
    navigate('/'); // Redirige a la página de inicio
  } catch (err) {
    console.error('Error en el login con Google:', err);
    setError('No se pudo iniciar sesión con Google. Inténtalo de nuevo.');
  }
};

router.post('/google', googleLogin);
```

## MODELO DE DATOS “USUARIO”

```
const userSchema = new mongoose.Schema({  
    username: {  
        type: String,  
        required: true,  
        unique: true,  
        trim: true  
    },  
    email: {  
        type: String,  
        required: true,  
        unique: true,  
        trim: true,  
        lowercase: true,  
        match: [/^[\w+.-]+\w+@\w+([\.-]\w+)+[\w+.-]?\w+$/, 'Por favor, introduce un email válido']  
    },  
    password: {  
        type: String,  
        required: true,  
        minlength: 6  
    },  
    country: { // Para filtrar videos por el país del usuario si fuera necesario  
        type: String,  
        trim: true,  
        default: 'Desconocido'  
    },  
    createdAt: {  
        type: Date,  
        default: Date.now  
    }  
})
```

## PRUEBAS INICIALES DE FLUJO LOGIN/REGISTRO

The image displays three screenshots of the GeoTube application interface:

- Login Screen:** Shows the GeoTube logo at the top. Below it, there are fields for "Email" (containing "edithperezdejesus00@gmail.com") and "Contraseña" (password). A note below the password field specifies requirements: "Mínimo 8 caracteres", "Una mayúscula", "Una minúscula", and "Un símbolo". At the bottom are buttons for "Iniciar Sesión" (Login) and "Iniciar sesión con Google" (Login with Google), along with a link to "Regístrate" (Register).
- Registration Screen:** Shows the GeoTube logo at the top. Below it, there are fields for "Nombre" (containing "Edith") and "País" (containing "Mexico"). There are also fields for "Email" (containing "edithperezdejesus00@gmail.com") and "Contraseña" (password). A note below the password field specifies requirements: "Mínimo 8 caracteres", "Una mayúscula", "Una minúscula", and "Un símbolo". At the bottom are buttons for "Registrar" (Register) and "Iniciar sesión con Google" (Login with Google).
- Location Request Screen:** Shows a search bar with placeholder text "Busca videos en tu área..." and a magnifying glass icon. Below it, a message reads "Recomendaciones según tu ubicación: 🌎". A white callout box titled "Información sobre tu ubicación" contains text about location permissions and two buttons: "Permitir" (Allow) and "Ahora no" (Not Now).