Código para que podamos registrar un usuario

```
// Registro de usuario
router.post('/register', async (req, res) => {
    const { username, password, email } = req.body;

    // Verificar si el usuario ya existe
    const userExists = users.find(user => user.username === username);
    if (userExists) {
        return res.status(400).json({ message: 'Usuario ya registrado' });
    }

    // Encriptar la contraseña
    const hashedPassword = await bcrypt.hash(password, 10);

    // Crear un nuevo usuario
    const newUser = { username, email, password: hashedPassword };
    users.push(newUser);

    res.status(201).json({ message: 'Usuario registrado con éxito', user: { username, email } });
});
```

Código para que podamos iniciar sesión

```
// Inicio de sesión
router.post('/login', async (req, res) => {
    const { username, password } = req.body;

    // Buscar el usuario
    const user = users.find(user => user.username === username);
    if (!user) {
        return res.status(400).json({ message: 'Usuario no encontrado' });
    }

    // Verificar la contraseña
    const isPasswordValid = await bcrypt.compare(password, user.password);
    if (!isPasswordValid) {
        return res.status(400).json({ message: 'Contraseña incorrecta' });
    }

    // Crear token JWT
    const token = jwt.sign({ username: user.username }, process.env.JWT_SECRET, { expiresIn: '1h' });

    res.status(200).json({ message: 'Inicio de sesión exitoso', token });
});
```

Código para acceder a recurso protegido

```
// Acceder a recurso protegido
router.get('/protected-resource', (req, res) => {
    const token = req.headers['authorization']?.split(' ')[1];
    if (!token) {
        return res.status(401).json({ message: 'Token no proporcionado' });
    }

    // Verificar token
    try {
        const decoded = jwt.verify(token, process.env.JWT_SECRET);
        res.status(200).json({ message: 'Acceso autorizado', user: decoded });
    } catch (err) {
        res.status(401).json({ message: 'Token no válido' });
    }
});
```

Código para cerrar sesión

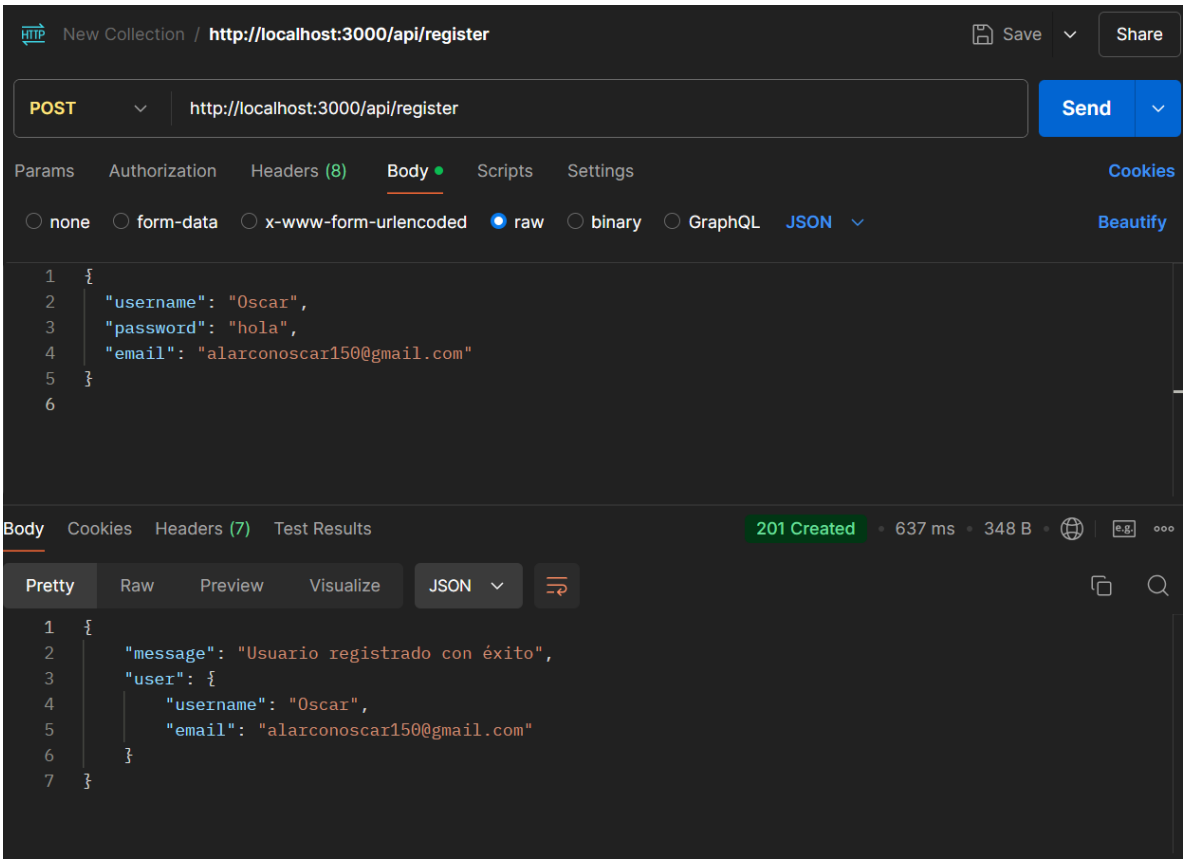
```
// Cerrar sesión
router.post('/logout', (req, res) => {
    res.status(200).json({ message: 'Cierre de sesión exitoso' });
});

module.exports = router;
```

Para correr el programa utilizaremos los siguientes comandos

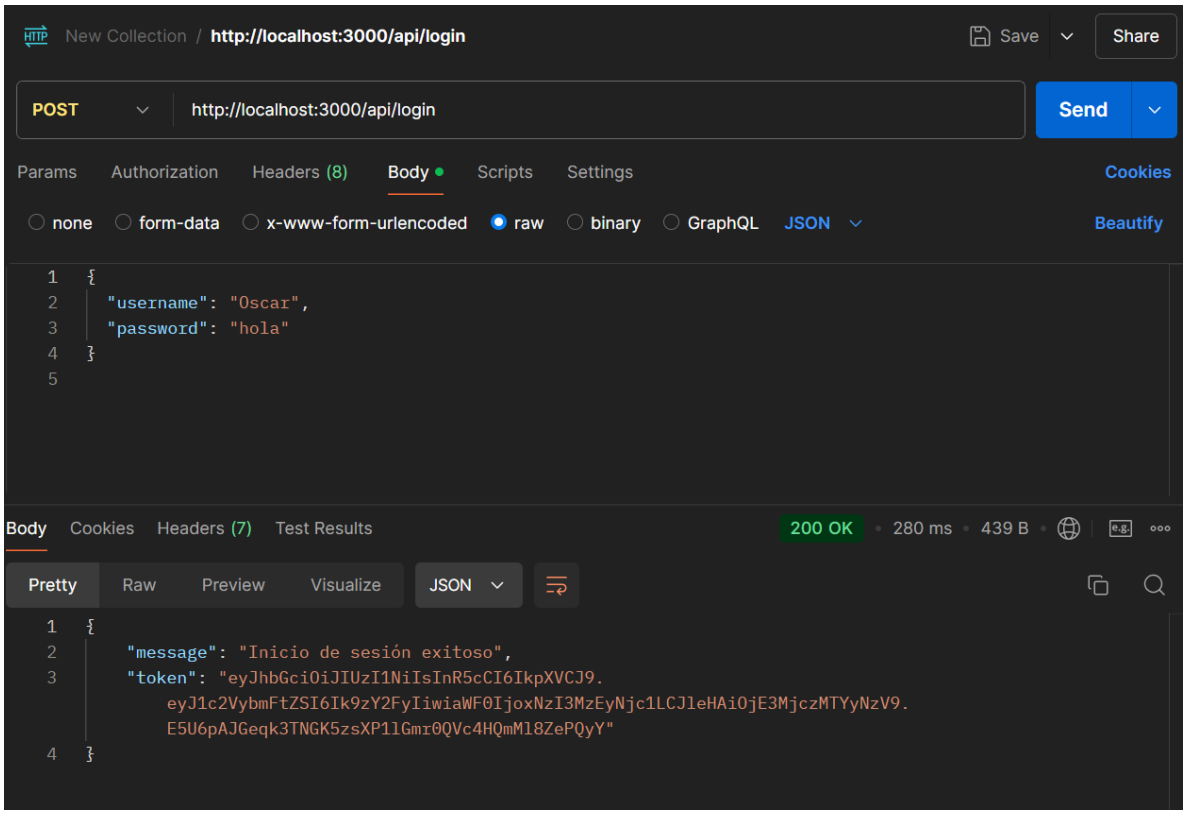
```
PS C:\Users\Oscar\OneDrive\Escritorio\API 2_Postman> cd "C:\Users\Oscar\OneDrive\Escritorio\API 2_Postman\API 2\auth-api"
PS C:\Users\Oscar\OneDrive\Escritorio\API 2_Postman\API 2\auth-api> node server.js
Servidor corriendo en el puerto 3000
```

PRUEBAS

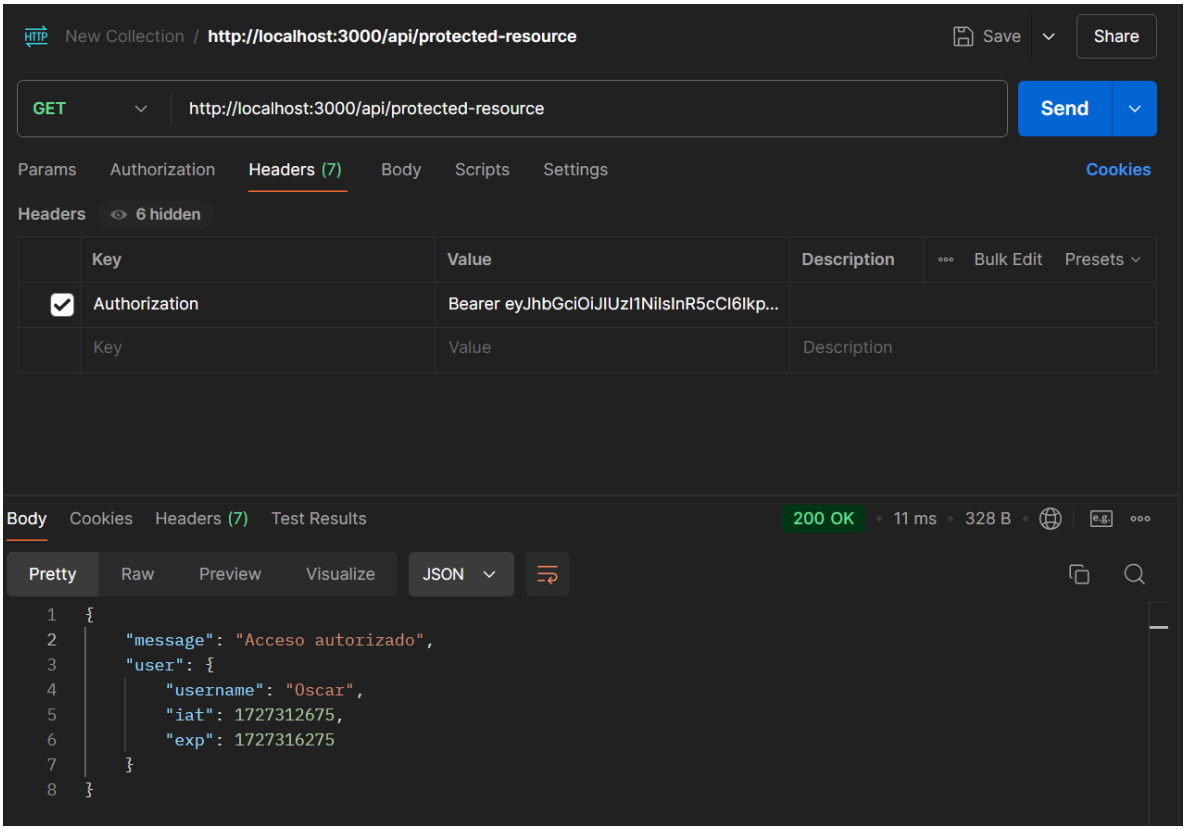
Apartado para registrar un usuario



Apartado de inicio de sesión



Apartado para comprar si puede recibir el archivo con acceso especial



Apartado de cierre de sesión