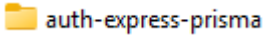
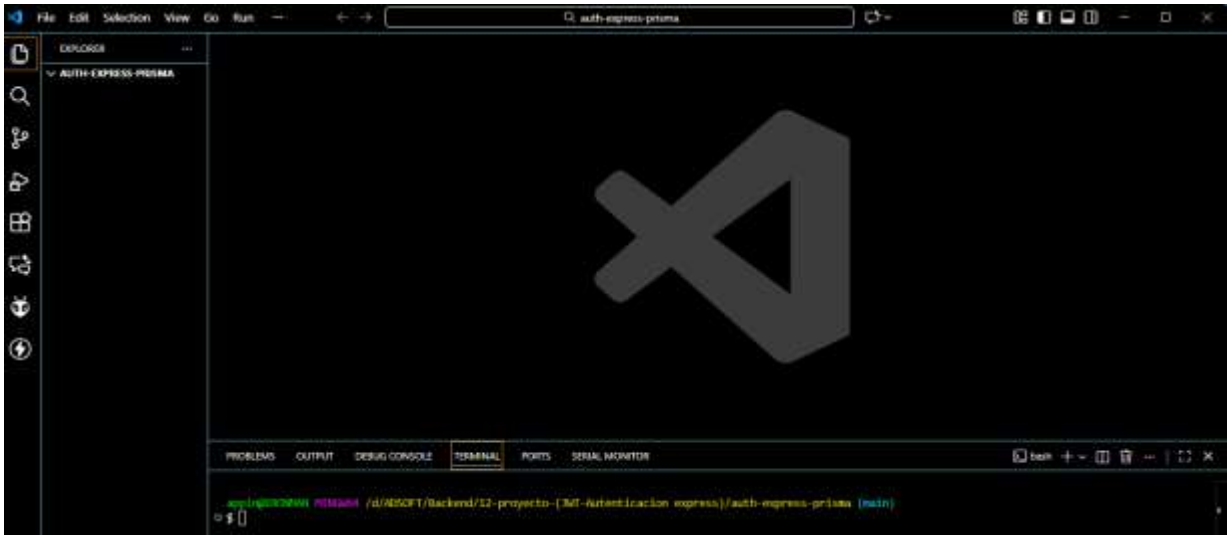
Ejemplo conceptual de flujo JWT en Express-2025

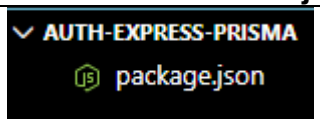
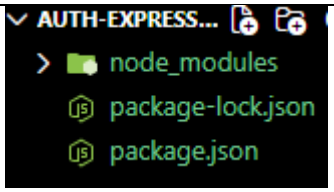
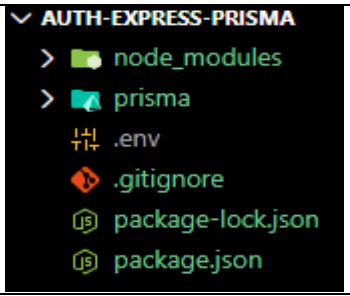
1. **Cree una carpeta**



2. **Acceda a la carpeta creada desde Visual Studio Code (VSC) y abra la terminal**



3. **Ejecute los siguientes comandos en la terminal desde VSC para crear el proyecto e instalar dependencias:**

Comandos	Lo que crea cuando se ejecuta
npm init -y	
npm install express jsonwebtoken bcryptjs cors	
npm install prisma@5.20 --save-dev	Instala una versión de prisma estable
npm install @prisma/client@5.20	Instala un cliente que se adecue a la versión instalada
npx prisma init	

4. **Configurar MySQL en `.env`**

```
DATABASE_URL="mysql://root:123456@localhost:3306/authdb"
JWT_SECRET="miClave1234*2025"
```

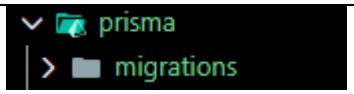
5. **Modifica y crea el modelo de usuario en `schema.prisma` que se encuentra dentro de la carpeta `prisma`**

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "mysql"
  url      = env("DATABASE_URL")
}

model Usuario{
  id        Int       @id @default(autoincrement())
  nombre    String
  email     String    @unique
  password  String
  creadoEn  DateTime  @default(now())
}
```

6. **Ejecute el siguiente comando para comprobar que no hay errores de transcripción en los modelos agregados**

npx prisma generate

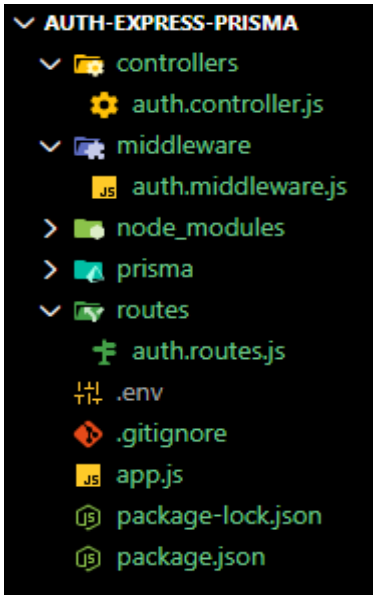
7. **Ejecute la migración**

Comando	Lo que crea cuando se ejecuta
npx prisma migrate dev --name init	

8. **Cree la siguiente estructura para el servidor**

Carpetas: `controllers , middleware , routes`

Archivos: `auth.controller.js , auth.middleware.js , auth.routes.js , app.js`



9. **Configurar Express** → `app.js`

```
const express = require("express");
const cors = require("cors");
const authRoutes = require("./routes/auth.routes");

const app = express();

app.use(cors());
app.use(express.json());
app.use("/api/auth", authRoutes);

app.listen(3000, () => {
  console.log("Servidor corriendo en http://localhost:3000");
});
```

10. **Controlador de autenticación** → `controllers/auth.controller.js`

```
const { PrismaClient } = require("@prisma/client");
const prisma = new PrismaClient();
const bcrypt = require("bcryptjs");
const jwt = require("jsonwebtoken");

const JWT_SECRET = process.env.JWT_SECRET;

// Registro
exports.register = async (req, res) => {
  const { nombre, email, password } = req.body;

  try {
    const existe = await prisma.usuario.findUnique({
      where: { email },
    });

    if (existe)
      return res.status(400).json({ msg: "El email ya existe" });

    const hashed = await bcrypt.hash(password, 10);

    const usuario = await prisma.usuario.create({
      data: { nombre, email, password: hashed },
    });

    res.json({ msg: "Usuario registrado", usuario });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};
```

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

  try {
    const usuario = await prisma.usuario.findUnique({
      where: { email },
    });

    if (!usuario)
      return res.status(400).json({ msg: "Credenciales inválidas" });

    const valid = await bcrypt.compare(password, usuario.password);

    if (!valid)
      return res.status(400).json({ msg: "Contraseña incorrecta" });

    const token = jwt.sign(
      {
        id: usuario.id,
        email: usuario.email,
      },
      JWT_SECRET,
      { expiresIn: "2h" }
    );

    res.json({ msg: "Login exitoso", token });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};
```

11. **Middleware que protege rutas** → `middleware/auth.middleware.js`

```
const jwt = require("jsonwebtoken");
const JWT_SECRET = process.env.JWT_SECRET;

module.exports = (req, res, next) => {
  const token = req.headers["authorization"];
  if (!token)
    return res.status(401).json({ msg: "Token requerido" });
  try {
    // El token llega como: "Bearer asd123123123"
    const decoded = jwt.verify(token.split(" ")[1], JWT_SECRET);
    req.user = decoded;
    next();
  } catch (error) {
    res.status(401).json({ msg: "Token inválido o expirado" });
  }
};
```

12. **Rutas** → `routes/auth.routes.js`

```
const express = require("express");
const router = express.Router();

const {register,login} = require("../controllers/auth.controller");
const auth=require("../middleware/auth.middleware");

router.post("/register", register);
router.post("/login", login);

router.get("/perfil",auth,(req,res)=>{
   res.json({msg:"Acceso permitido",user:req.user});
});

module.exports = router;
```

13. **Corra el servidor ejecutando el siguiente comando**

node app.js

Debe mostrar:

```
$ node app.js
Servidor corriendo en http://localhost:3000
```

14. **Realice la prueba en postman o Thunder client, para este caso se hace uso de thunder client**

15. **Primero registro de usuario (recuerde que los datos que intente registrar no se deben haber registrado antes sino informa con un mensaje que el correo ya existe)**

POST → `http://localhost:3000/api/auth/register`
Body:

```
{
 "nombre":"Juan Manuel",
 "email":"juanmanuel@example.com",
 "password":"1234569"
}
```

Respuesta:

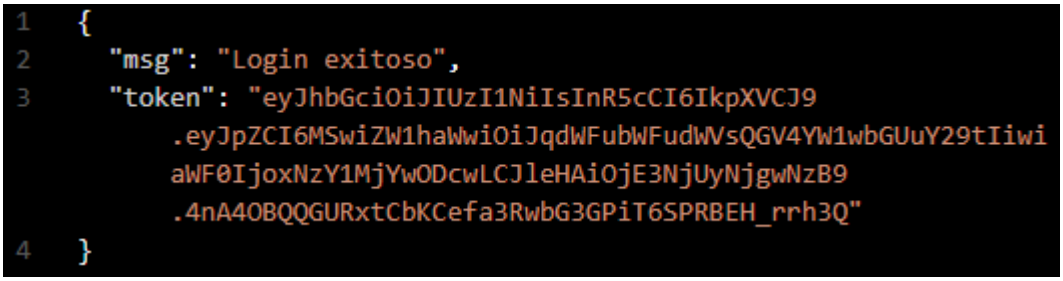

```
{
  "msg": "Usuario registrado",
  "usuario": {
    "id": 1,
    "nombre": "Juan Manuel",
    "email": "juanmanuel@example.com",
    "password": "$2b$10$acivDZWmBDsdVtXkbM8X2ueC37bP7
        .I6RT8TLv0tTnI9KgQXRstSi",
    "creadoEn": "2025-12-09T06:05:31.646Z"
  }
}
```

16. **Ingreso del Login (en caso de que no se correcto llega un mensaje “credenciales invalidas”)**

POST → `http://localhost:3000/api/auth/login`
Body:

```
{
  "email":"juanmanuel@example.com",
  "password":"1234569"
}
```

Respuesta

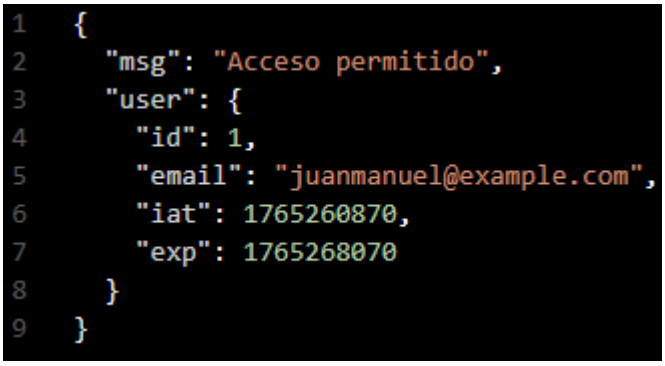


```
{
  "msg": "Login exitoso",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
      .eyJpZCI6MSwiZW1haWwiOiJqdWFubWFudWVsQGV4YW1wbGUuY29tIiwi
      aWF0IjoxNzY1MjYwODcwLCJleHAiOjE3NjUyNjgwNzB9
      .4nA4OBQQGURxtCbKCefa3RwbG3GPiT6SPRBEH_rrh3Q"
}
```

17. **Acceder a ruta protegida (En caso de que no se pueda acceder aparece un mensaje como “Token inválido o expirado”)**

GET → `http://localhost:3000/api/auth/perfil`
Header:
Authorization Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiZW1haWwiOiJqdWFubWFudWVsQGV4YW1wbGUuY29tIiwiaWF0IjoxNzY1MjYwODcwLCJleHAiOjE3NjUyNjgwNzB9.4nA4OBQQGURxtCbKCefa3RwbG3GPiT6SPRBEH_rrh3Q

Respuesta



```
{
  "msg": "Acceso permitido",
  "user": {
    "id": 1,
    "email": "juanmanuel@example.com",
    "iat": 1765260870,
    "exp": 1765268070
  }
}
```

18. **En la base de datos se puede observar los datos registrados en la tabla “usuario”**

id	nombre	email	password	creadoEn
1	Juan Manuel	juanmanuel@example.com	$2b$10$acivDZWmBDsdVtXkbM8X2ueC37bP7.I...	2025-12-09 06:05:31.646
NULL	NULL	NULL	NULL	NULL