Servidor usando Node y Express

```
import express from "express"; // Importa el framework Express para crear servidores web en Node.js
import bodyParser from "body-parser"; // Importa el middleware 'body-parser' para analizar datos de formularios codificados en la URL
import session from "express-session"; // Importa el middleware 'express-session' para manejar sesiones de usuario
import expressValidator from "express-validator"; // Importa la biblioteca express-validator para validar datos de entrada
import csrf from "csurf"; // Importa el paquete csrf para protección contra ataques CSRF
import helmet from "helmet"; // Importa el paquete helmet para mejorar la seguridad de la aplicación
import morgan from "morgan"; // Importa el paquete morgan para registro de eventos
import path from "path"; // Importa el módulo 'path' para manejar rutas de archivos
import { fileURLToPath } from "url"; // Importa la función 'fileURLToPath' del módulo 'url' para convertir la URL del archivo en una ruta de sistema de archivos
import { Pool } from "pg"; // Importa el paquete pg para interactuar con PostgreSQL

const __dirname = path.dirname(fileURLToPath(import.meta.url)); // Obtiene el directorio actual utilizando 'dirname' y 'fileURLToPath'
const port = 3000; // Define el puerto en el que se ejecutará el servidor

// Configuración de la conexión a PostgreSQL
const pool = new Pool({
  user: "tu_usuario",
  host: "localhost",
  database: "tu_base_de_datos",
  password: "tu_contraseña",
  port: 5432,
});

// Configuración de la aplicación Express
const app = express();
```

```
// Middleware
app.use(bodyParser.urlencoded({ extended: true })); // Configura el middleware 'body-parser' para analizar datos de formularios codificados en la URL
app.use(expressValidator()); // Configura el middleware 'express-validator' para validar y sanitizar datos de entrada
app.use(session({ secret: "your-secret-key", resave: false, saveUninitialized: true })); // Configura el middleware 'express-session' para manejar sesiones de usuario
app.use(csrf()); // Configura el middleware 'csurf' para protección contra ataques CSRF
app.use(helmet()); // Configura el middleware 'helmet' para mejorar la seguridad de la aplicación
app.use(morgan("combined")); // Configura el middleware 'morgan' para registro de eventos

// Rutas protegidas con autenticación y autorización
app.get("/profile", isAuthenticated, (req, res) => {
  // Ruta protegida: muestra el perfil del usuario autenticado
});

// Middleware para verificar si el usuario está autenticado
function isAuthenticated(req, res, next) {
  if (req.session && req.session.user) {
    return next();
  } else {
    return res.redirect("/login");
  }
}

// Rutas de autenticación
app.get("/login", (req, res) => {
  // Ruta para mostrar el formulario de inicio de sesión
});
app.post("/login", (req, res) => {
  // Ruta para manejar el inicio de sesión del usuario
});
app.get("/logout", (req, res) => {
  // Ruta para cerrar sesión y redirigir al usuario a la página de inicio
});

// Manejo de errores
app.use((err, req, res, next) => {
  // Middleware de manejo de errores para manejar errores de manera uniforme en toda la aplicación
});

// El servidor comienza a escuchar las solicitudes en el puerto especificado
app.listen(port, () => {
  console.log(`Servidor escuchando en el puerto ${port}`);
});
```