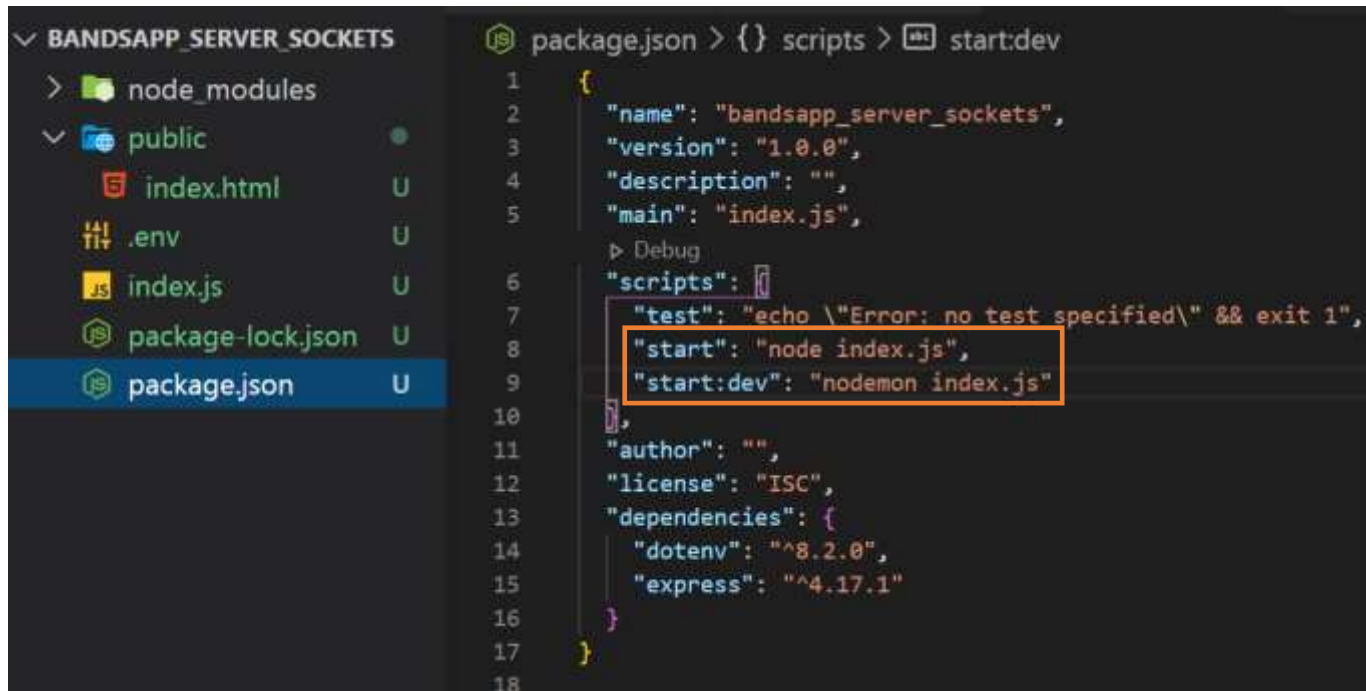


Socket Server - Express - Backend

npm init

npm i express

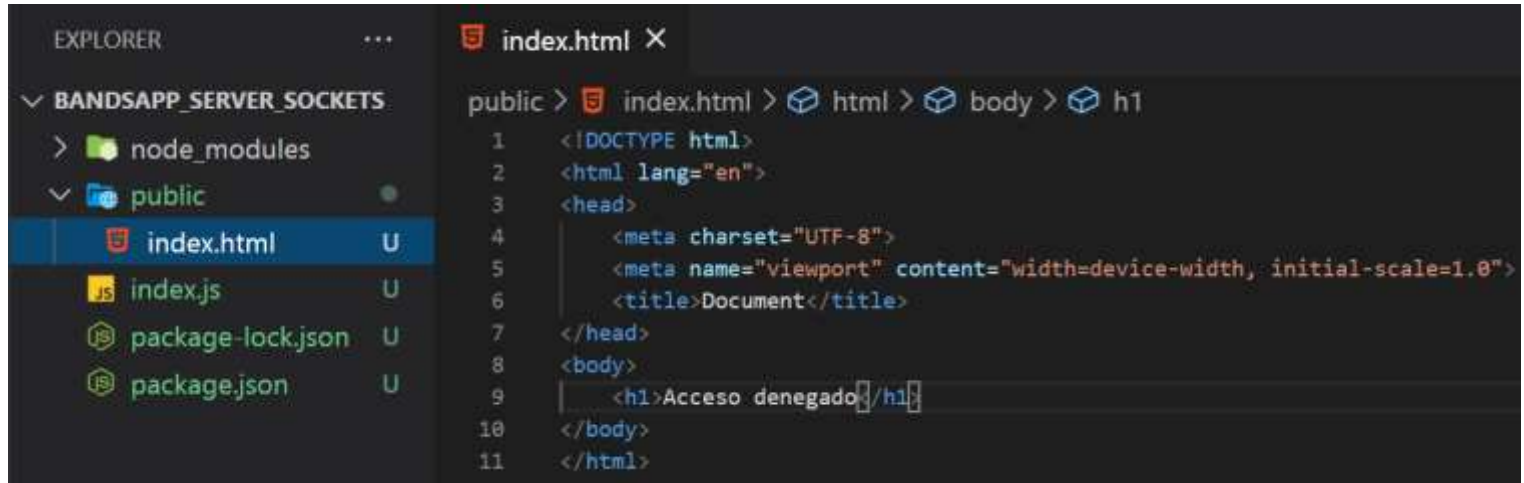


The screenshot shows the VS Code interface. On the left, the file explorer displays the project structure for 'BANDSAPP_SERVER_SOCKETS', including 'node_modules', 'public' (with 'index.html'), '.env', 'index.js', 'package-lock.json', and 'package.json'. The 'package.json' file is selected and open in the editor. The code in 'package.json' is as follows:

```
1 {
2   "name": "bandsapp_server_sockets",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1",
8     "start": "node index.js",
9     "start:dev": "nodemon index.js"
10  },
11   "author": "",
12   "license": "ISC",
13   "dependencies": {
14     "dotenv": "^8.2.0",
15     "express": "^4.17.1"
16  }
17 }
```

npm run start:dev

➔ Crear un directorio público

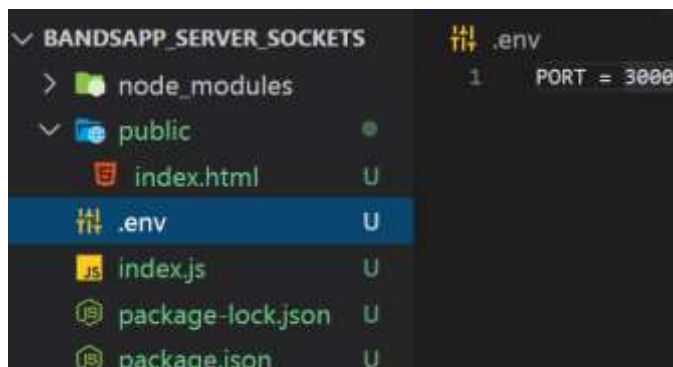


The screenshot shows the VS Code interface. On the left, the file explorer shows the 'public' directory selected. The 'index.html' file is open in the editor. The code in 'index.html' is as follows:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7 </head>
8 <body>
9   <h1>Acceso denegado</h1>
10 </body>
11 </html>
```

➔ Variables de entorno y Scripts

npm i dotenv



The screenshot shows the VS Code interface. On the left, the file explorer shows the '.env' file selected. The '.env' file is open in the editor. The code in '.env' is as follows:

```
1 PORT = 3000
```

Socket.io - Configuración inicial

npm install socket.io

Creando carpeta “sockets” y archivo “socket.js”:

```
const { io } = require("../index");

io.on("connection", (client) => {
  console.log("Cliente conectado");
  client.on("disconnect", () => {
    console.log("Cliente desconectado");
  });
  // Escuchando mensaje emitido desde index.html (script)
  client.on("mensaje", (payload) => {
    console.log("Mensaje recibido", payload);
    // Emitir a todos el mensaje
    io.emit("nuevo-mensaje", { admin: "Nuevo mensaje" });
  });
});
```

Configurar “index.js”:

```
const path = require("path");
// Lee las variables de entorno del archivo '.env'
require("dotenv").config();
// Express
const express = require("express");
const app = express();
// Servidor para el socket: https://socket.io/get-started/chat/
// Comunicación entre app y socket
var server = require("http").createServer(app);
// export io para usarlo en socket.js
module.exports.io = require("socket.io")(server);
/* Importando socket */
require("./sockets/socket");

/* Path de la carpeta pública */
// __dirname => http://dominio... => heroku
const publicPath = path.resolve(__dirname, "public");

/* middleware */
app.use(express.static(publicPath));

server.listen(process.env.PORT, (err) => {
  if (err) throw new Error(err);
  console.log("Servidor corriendo en puerto", process.env.PORT);
});
```

Escuchar emisiones y enviar eventos desde “index.html”:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <h1>Acceso denegado</h1>
    <script src="socket.io/socket.io.js"></script>
    <script>
      var socket = io();

      socket.on("connect", function () {
        console.log("Conectado al servidor");
      });

      socket.on("disconnect", function () {
        console.log("Perdimos comunicación con el servidor");
      });
    </script>
  </body>
</html>
```

```

socket.emit("mensaje", { nombre: "Danicode" });

socket.on("nuevo-mensaje", function (payload) {
  console.log("Escuhando:", payload);
});
</script>
</body>
</html>

```

BandNames Flutter + Socket Backend

➔ Backend Lógica para el manejo de las votaciones

npm i uuid

Crear los modelos “Band” y “Bands” (que hará el trabajo de BD):



```

models > .js band.js > ...
1  const { v4: uuidV4 } = require("uuid");
2
3  class Band {
4    constructor(name = "no-name") {
5      this.id = uuidV4();
6      this.name = name;
7      this.vote = 1;
8    }
9  }
10
11  module.exports = Band;
12

```

Clase Bands:

```

/* Clase que manejará todas las bandas */

const Band = require("./band");

class Bands {
  constructor() {
    this.bands = [];
  }

  addBand(band = new Band()) {
    this.bands.push(band);
  }

  getBands() {
    return this.bands;
  }

  deleteBand(id = "") {
    this.bands = this.bands.filter((band) => band.id !== id);
    return this.bands;
  }

  voteBand(id = "") {
    this.bands = this.bands.map((band) => {
      if (band.id === id) {
        band.vote += 1;
        return band;
      } else {
        return band;
      }
    });
  }
}

module.exports = Bands;

```

➔Socket Emitir bandas registradas

En el archivo “sockets.js”:

```
const { io } = require("../index");
const Band = require("../models/band");
const Bands = require("../models/bands");

const bands = new Bands();
// Añadiendo algunas bandas
bands.addBand(new Band("Queen"));
bands.addBand(new Band("Radiohead"));
bands.addBand(new Band("Ariana Grande"));
bands.addBand(new Band("Oasis"));
/* console.log(bands["bands"][0].name); */

io.on("connection", (client) => {
  console.log("Cliente conectado");
  // Emitir al cliente que se conecta las bandas registradas
  client.emit("active-bands", bands.getBands());
  client.on("disconnect", () => {
    console.log("Cliente desconectado");
  });
  // Escuchando mensaje emitido desde index.html (script)
  client.on("mensaje", (payload) => {
    console.log("Mensaje recibido", payload);
    // Emitir a todos el mensaje
    io.emit("nuevo-mensaje", { admin: "Nuevo mensaje" });
  });

  client.on("emitir-mensaje", (payload) => {
    console.log(payload);
    /* io.emit("nuevo-mensaje", payload); => Emite a todos */
    client.broadcast.emit("nuevo-mensaje", payload); // Todos, menos al que lo emite
  });
});
```

Ver la lista de bandas en la consola del navegador:

En el index.html de la carpeta “public”:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <h1>Acceso denegado</h1>
    <script src="socket.io/socket.io.js"></script>
    <script>
      var socket = io();

      socket.on("connect", function () {
        console.log("Conectado al servidor");
      });

      socket.on("disconnect", function () {
        console.log("Perdimos comunicación con el servidor");
      });

      socket.emit("mensaje", { nombre: "Danicode" });

      socket.on("nuevo-mensaje", function (payload) {
        console.log("Escuchando:", payload);
      });

      /* Escuchando la lista de bandas => Evento "active-bands" */
      socket.on("active-bands", function (payload) {
        console.clear();
```

```
    console.table(payload);  
  });  
</script>  
</body>  
</html>
```

➔ Socket Service - Conectar nuestra App con el Socket Server

Instalar *socket.io-client-dart*, *provider* (manejador de estados – tipo patrón bloc)

```
cupertino_icons: ^0.1.3  
socket_io_client: ^0.9.11  
provider: ^4.3.2+2
```