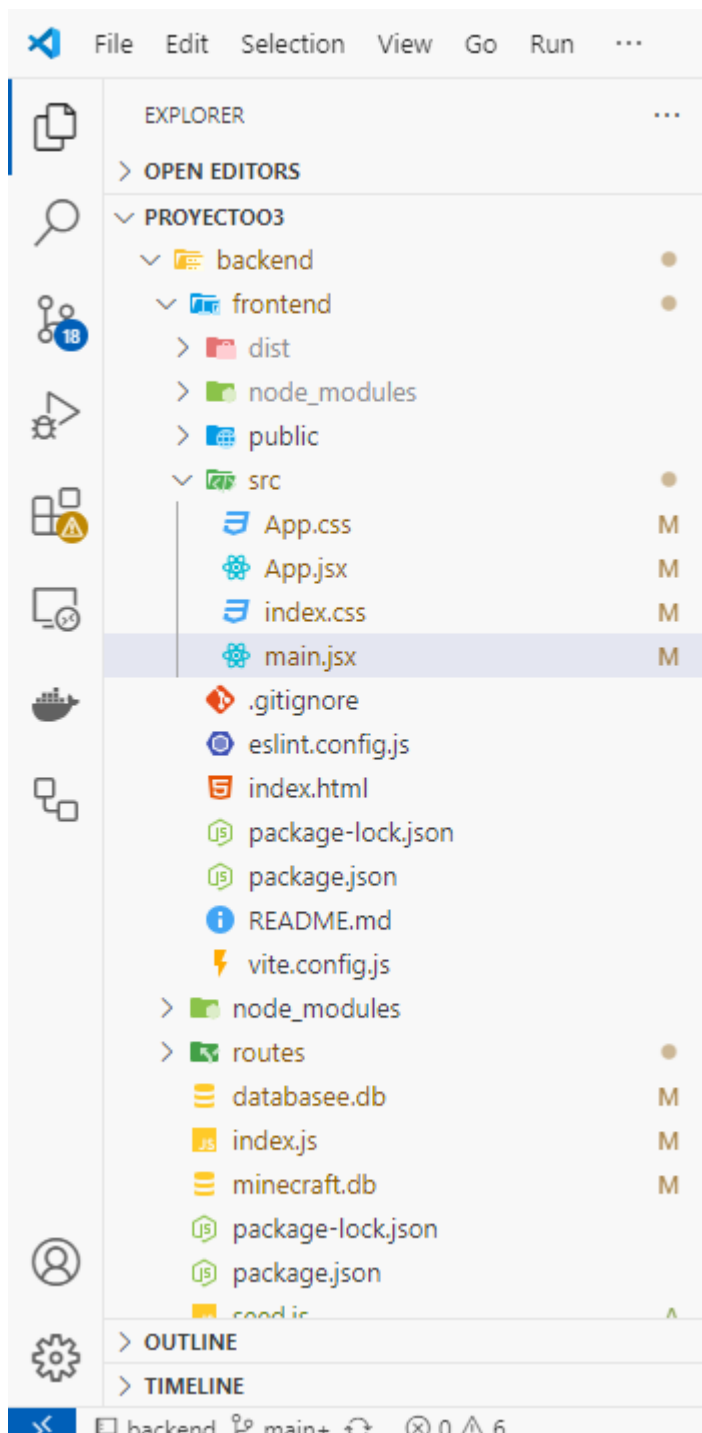


Estructura de Archivos



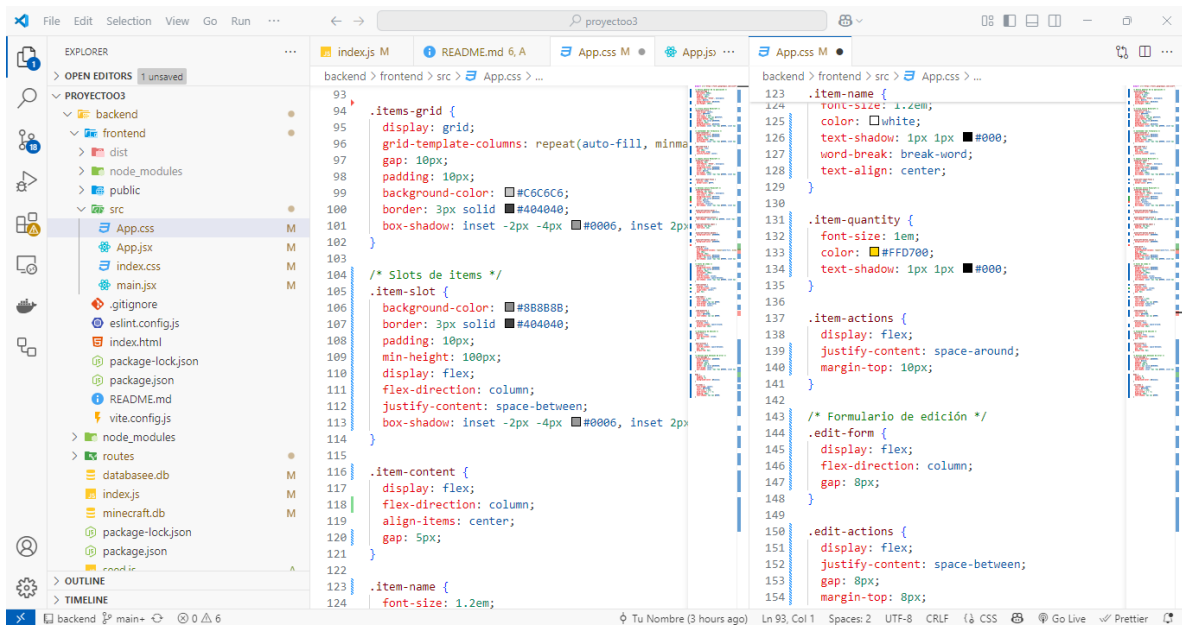
Objetivo De Proyecto

1 README.md

```
1
2
3 El Proyecto consiste en un sistema de inventario inspirado en Minecraft donde puedes agregar, eliminar
  y gestionar items como si fuera el juego real.
4
5 Que podemos hacer con el?
6 -crear nuevos items de Minecraft
7 - Puedes borrar items que ya no quieras
8 - Muestra una lista de todos tus items
9 - Los items se guardan en una base de datos
10
11 Que Componentes tiene en uso?
12
13 - Frontend:React
14 - Backend: Node.js con Express
15 - Base de datos: SQLite3
16 - Estilos: CSS
```

```
28 .input-container {
29 }
30
31 .add-item-form {
32   display: flex;
33   gap: 10px;
34   flex-wrap: wrap;
35   justify-content: center;
36 }
37
38 /* Inputs estilo Minecraft */
39 .minecraft-input {
40   padding: 8px;
41   font-family: 'VT323', monospace;
42   font-size: 1.2em;
43   background-color: #888888;
44   border: 3px solid #404040;
45   color: white;
46   min-width: 150px;
47   box-shadow: inset -2px -4px #0006, inset 2px 2px #0006, inset 2px 2px #0006;
48 }
49
50 .minecraft-input:focus {
51   outline: none;
52   border-color: #FFFF;
53 }
54
55 /* Botones estilo Minecraft */
56 .minecraft-button {
57   padding: 6px 12px;
58   font-family: 'VT323', monospace;
59   font-size: 1.2em;
60 }
61
62 .minecraft-button:hover {
63   background-color: #7F7FF7;
64   border: 3px solid #404040;
65   color: white;
66   cursor: pointer;
67   position: relative;
68   box-shadow: inset -2px -4px #0006, inset 2px 2px #0006, inset 2px 2px #0006;
69 }
70
71 .minecraft-button:active {
72   background-color: #9F9F9F;
73 }
74
75 .minecraft-button:small {
76   padding: 4px 8px;
77   font-size: 1em;
78 }
79
80 .minecraft-button.delete {
81   background-color: #880000;
82 }
83
84 .minecraft-button.cancel {
85   background-color: #880000;
86 }
87
88 .minecraft-button.delete:hover,
89 .minecraft-button.cancel:hover,
90 .items-grid {
91 }
```

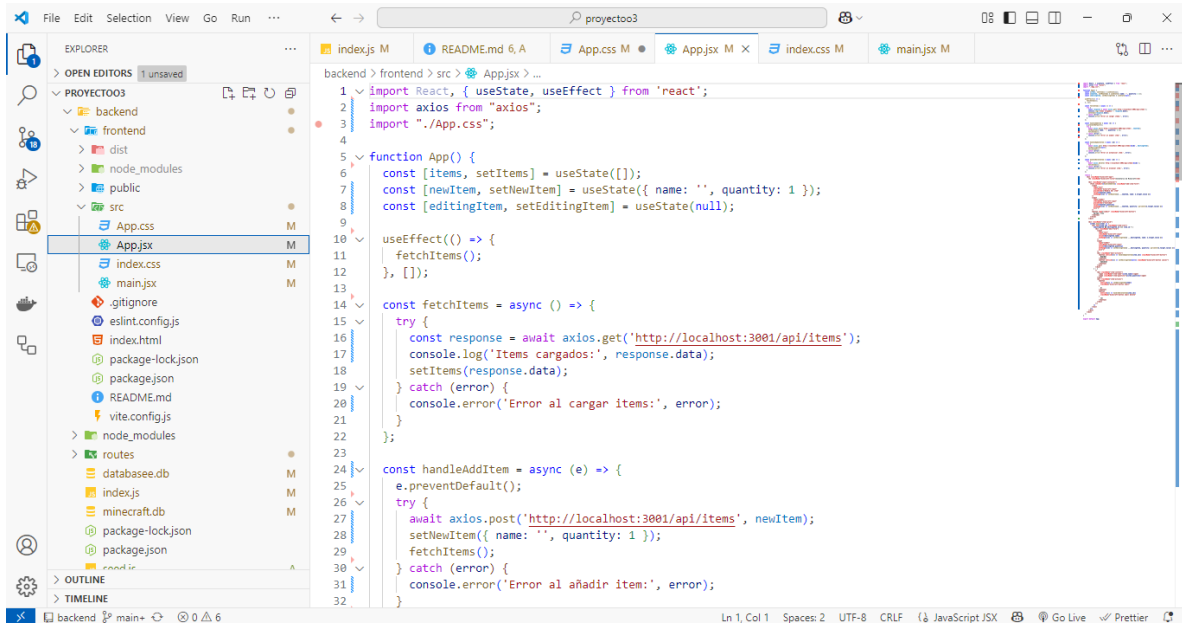
```
28 .input-container {
29 }
30
31 .add-item-form {
32   display: flex;
33   gap: 10px;
34   flex-wrap: wrap;
35   justify-content: center;
36 }
37
38 /* Inputs estilo Minecraft */
39 .minecraft-input {
40   padding: 8px;
41   font-family: 'VT323', monospace;
42   font-size: 1.2em;
43   background-color: #888888;
44   border: 3px solid #404040;
45   color: white;
46   min-width: 150px;
47   box-shadow: inset -2px -4px #0006, inset 2px 2px #0006, inset 2px 2px #0006;
48 }
49
50 .minecraft-input:focus {
51   outline: none;
52   border-color: #FFFF;
53 }
54
55 /* Botones estilo Minecraft */
56 .minecraft-button {
57   padding: 6px 12px;
58   font-family: 'VT323', monospace;
59   font-size: 1.2em;
60 }
61
62 .minecraft-button:hover {
63   background-color: #7F7FF7;
64   border: 3px solid #404040;
65   color: white;
66   cursor: pointer;
67   position: relative;
68   box-shadow: inset -2px -4px #0006, inset 2px 2px #0006, inset 2px 2px #0006;
69 }
70
71 .minecraft-button:active {
72   background-color: #9F9F9F;
73 }
74
75 .minecraft-button:small {
76   padding: 4px 8px;
77   font-size: 1em;
78 }
79
80 .minecraft-button.delete {
81   background-color: #880000;
82 }
83
84 .minecraft-button.cancel {
85   background-color: #880000;
86 }
87
88 .minecraft-button.delete:hover,
89 .minecraft-button.cancel:hover,
90 .items-grid {
91 }
```



The screenshot shows the VS Code editor with the Explorer sidebar on the left displaying the project structure. The main editor area shows the `App.css` file with the following CSS code:

```
93 .items-grid {
94   display: grid;
95   grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
96   gap: 10px;
97   padding: 10px;
98   background-color: #C6C6C6;
99   border: 3px solid #404040;
100   box-shadow: inset -2px -4px #0006, inset 2px 2px #0006;
101 }
102
103 /* Slots de items */
104 .item-slot {
105   background-color: #888888;
106   border: 3px solid #404040;
107   padding: 10px;
108   min-height: 100px;
109   display: flex;
110   flex-direction: column;
111   justify-content: space-between;
112   box-shadow: inset -2px -4px #0006, inset 2px 2px #0006;
113 }
114
115 .item-content {
116   display: flex;
117   flex-direction: column;
118   align-items: center;
119   gap: 5px;
120 }
121
122 .item-name {
123   font-size: 1.2em;
124 }
```

The status bar at the bottom indicates the file is `App.css` at line 93, column 1, with 2 spaces, UTF-8 encoding, and CRLF line endings.



The screenshot shows the VS Code editor with the Explorer sidebar on the left displaying the project structure. The main editor area shows the `App.jsx` file with the following JavaScript code:

```
1 import React, { useState, useEffect } from 'react';
2 import axios from 'axios';
3 import './App.css';
4
5 function App() {
6   const [items, setItems] = useState([]);
7   const [newItem, setNewItem] = useState({ name: '', quantity: 1 });
8   const [editingItem, setEditingItem] = useState(null);
9
10  useEffect(() => {
11    fetchItems();
12  }, []);
13
14  const fetchItems = async () => {
15    try {
16      const response = await axios.get('http://localhost:3001/api/items');
17      console.log('Items cargados:', response.data);
18      setItems(response.data);
19    } catch (error) {
20      console.error('Error al cargar items:', error);
21    }
22  };
23
24  const handleAddItem = async (e) => {
25    e.preventDefault();
26    try {
27      await axios.post('http://localhost:3001/api/items', newItem);
28      setNewItem({ name: '', quantity: 1 });
29      fetchItems();
30    } catch (error) {
31      console.error('Error al añadir item:', error);
32    }
33  };
34 }
```

The status bar at the bottom indicates the file is `App.jsx` at line 1, column 1, with 2 spaces, UTF-8 encoding, and CRLF line endings.

File Edit Selection View Go Run ...

project003

EXPLORER

OPEN EDITORS 1 unsaved

PROYECTO03

backend

frontend

dist

node_modules

public

src

App.css

App.jsx

index.css

main.jsx

.gitignore

eslint.config.js

index.html

package-lock.json

package.json

README.md

vite.config.js

node_modules

routes

database.db

index.js

minecraft.db

package-lock.json

package.json

node_modules

OUTLINE

TIMELINE

backend > frontend > src > App.jsx > ...

```
1 import React, { useState, useEffect } from 'react';
2 import axios from 'axios';
3 import './App.css';
4
5 function App() {
6   const [items, setItems] = useState([]);
7   const [newItem, setNewItem] = useState({ name: '', quantity: 1 });
8   const [editingItem, setEditingItem] = useState(null);
9
10  useEffect(() => {
11    fetchItems();
12  }, []);
13
14  const fetchItems = async () => {
15    try {
16      const response = await axios.get('http://localhost:3001/api/items');
17      console.log('Items cargados:', response.data);
18      setItems(response.data);
19    } catch (error) {
20      console.error('Error al cargar items:', error);
21    }
22  };
23
24  const handleAddItem = async (e) => {
25    e.preventDefault();
26    try {
27      await axios.post('http://localhost:3001/api/items', newItem);
28      setNewItem({ name: '', quantity: 1 });
29      fetchItems();
30    } catch (error) {
31      console.error('Error al añadir item:', error);
32    }
33  };
34}
```

File Edit Selection View Go Run ...

project003

EXPLORER

OPEN EDITORS 1 unsaved

PROYECTO03

backend

frontend

dist

node_modules

public

src

App.css

App.jsx

index.css

main.jsx

.gitignore

eslint.config.js

index.html

package-lock.json

package.json

README.md

vite.config.js

node_modules

OUTLINE

TIMELINE

backend > frontend > src > main.jsx

```
1 import React from 'react'
2 import ReactDOM from 'react-dom/client'
3 import App from './App'
4 import './index.css'
5
6 ReactDOM.createRoot(document.getElementById('root')).render(
7   <React.StrictMode>
8     <App />
9   </React.StrictMode>
10 )
```

File Edit Selection View Go Run ...

project003

EXPLORER

OPEN EDITORS 2 unsaved

PROYECTO03

backend

frontend

dist

node_modules

public

src

App.css

App.jsx

index.css

main.jsx

.gitignore

eslint.config.js

index.html

package-lock.json

package.json

README.md

vite.config.js

node_modules

routes

database.db

index.js

minecraft.db

package-lock.json

package.json

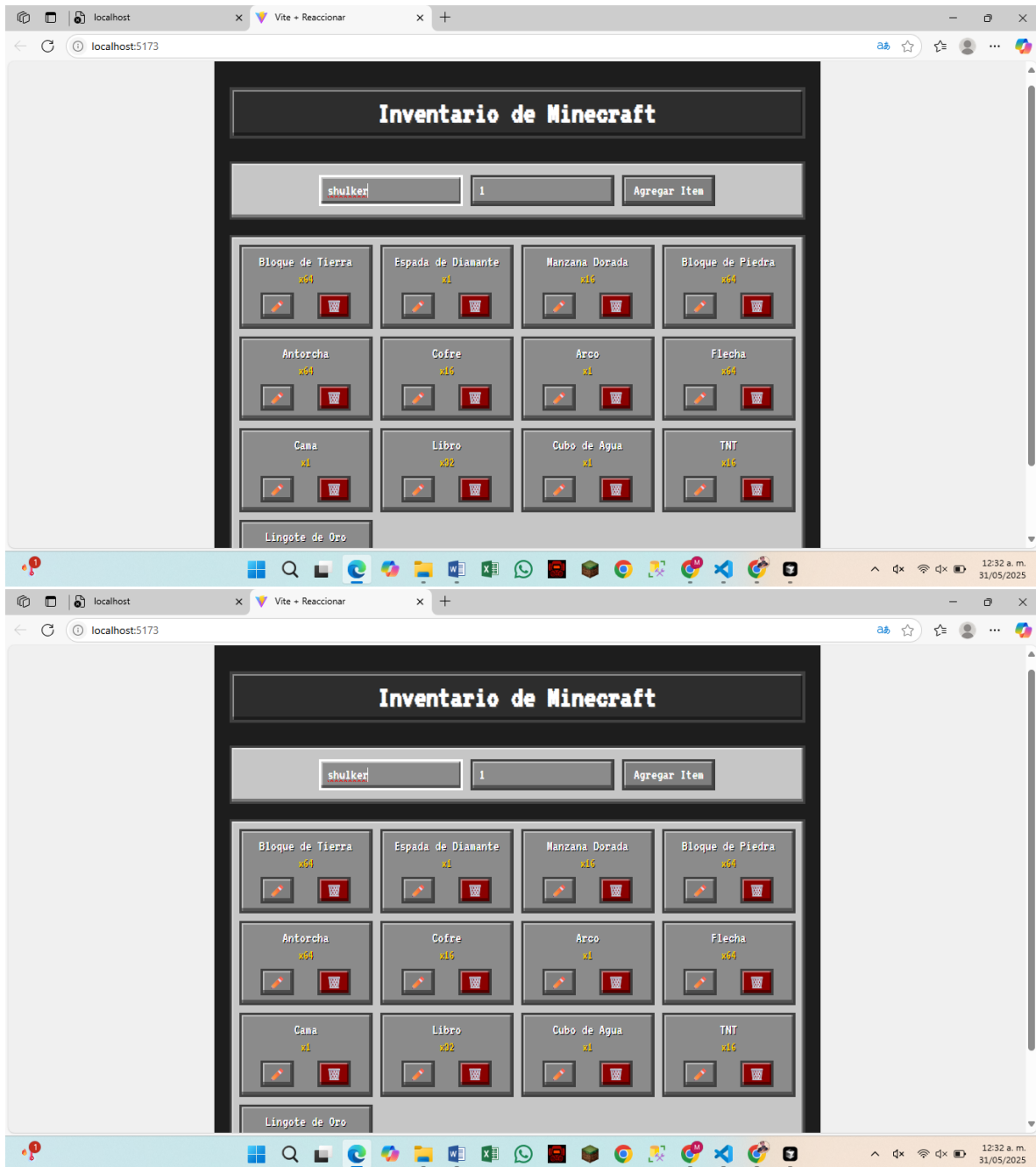
node_modules

OUTLINE

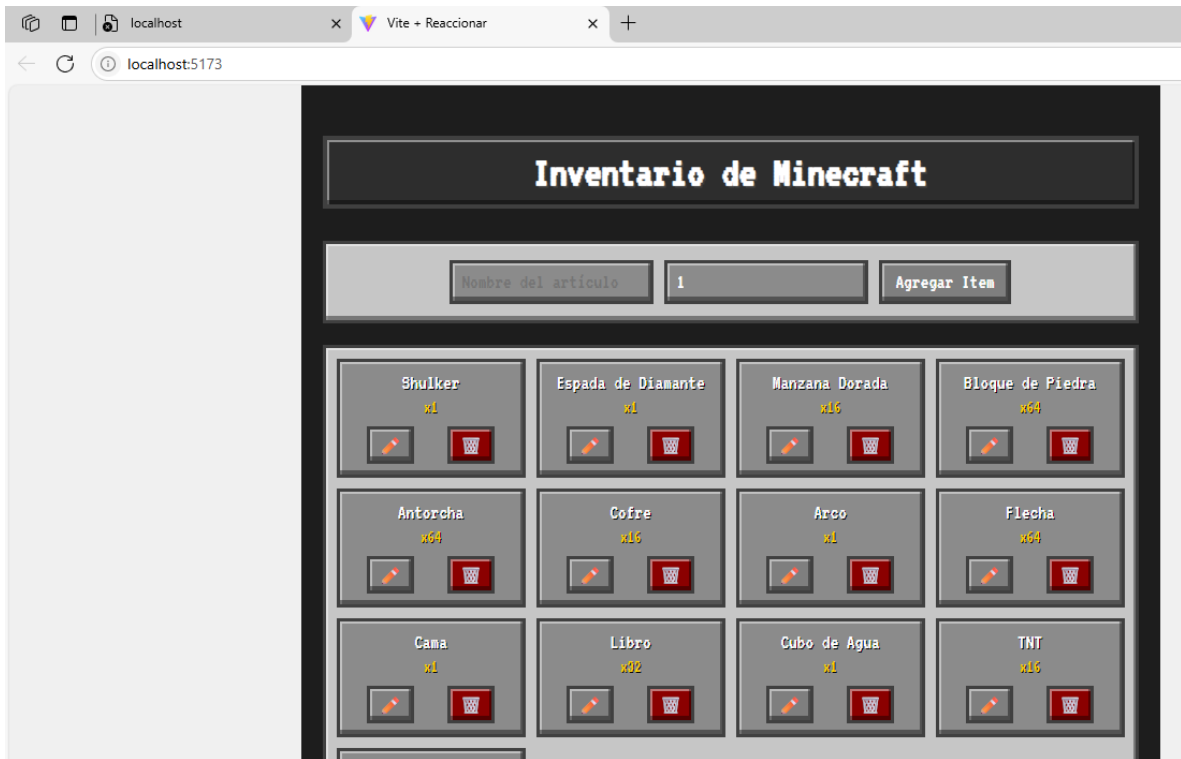
TIMELINE

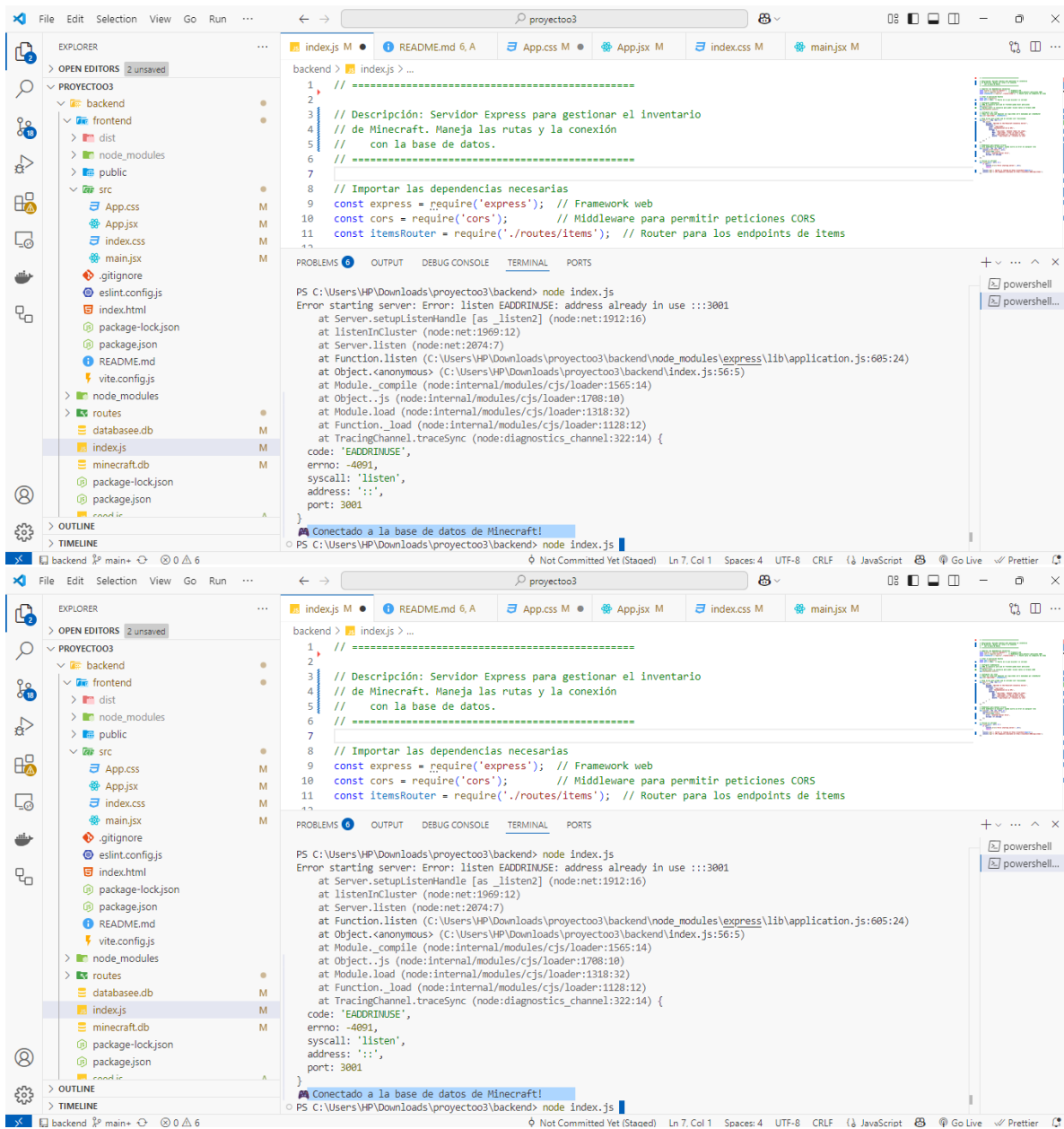
backend > index.js > ...

```
1 // =====
2
3 // Descripción: Servidor Express para gestionar el inventario
4 // de Minecraft. Maneja las rutas y la conexión
5 // con la base de datos.
6 // =====
7
8 // Importar las dependencias necesarias
9 const express = require('express'); // Framework web
10 const cors = require('cors'); // Middleware para permitir peticiones CORS
11 const itemsRouter = require('./routes/items'); // Router para los endpoints de items
12
13 // Crear la aplicación Express
14 const app = express();
15 const port = 3001; // Puerto en el que escuchará el servidor
16
17 // Configurar middlewares
18 // CORS es necesario para que el frontend pueda hacer peticiones
19 app.use(cors());
20 // express.json() es necesario para poder recibir datos en formato JSON
21 app.use(express.json());
22
23 // Configurar las rutas
24 // Todas las rutas que empiecen con /api/items serán manejadas por itemsRouter
25 app.use('/api/items', itemsRouter);
26
27 // Ruta básica para probar que el servidor está funcionando
28 app.get('/', (req, res) => {
29   res.json({
30     message: 'Welcome to the Minecraft Inventory Server!',
31     endpoints: {
32       items: '/api/items',
33     }
34   });
35 });
```









<http://localhost:5174/>

<https://github.com/Anton1oSQL1s/maincra.git>