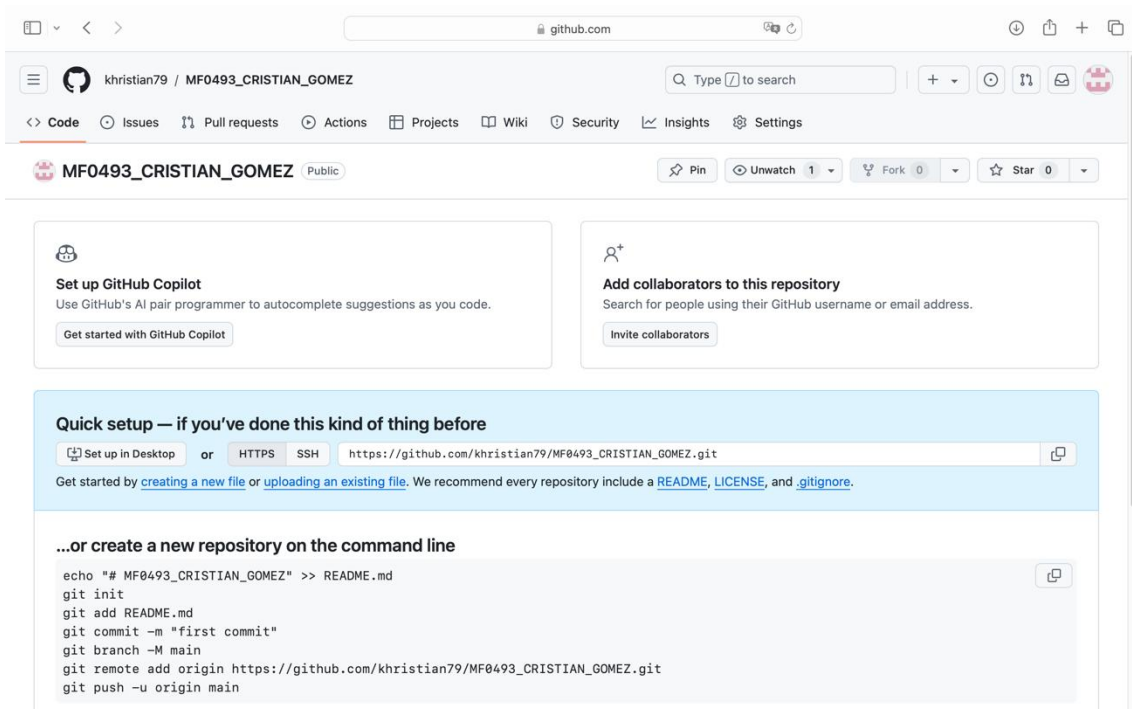
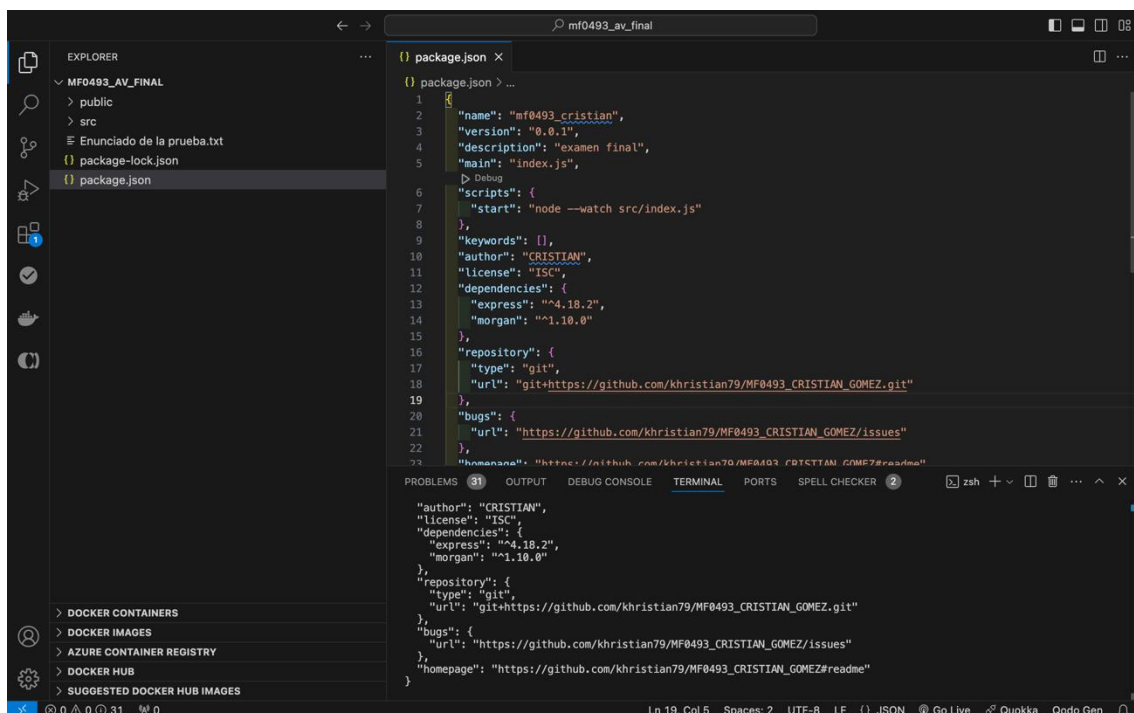
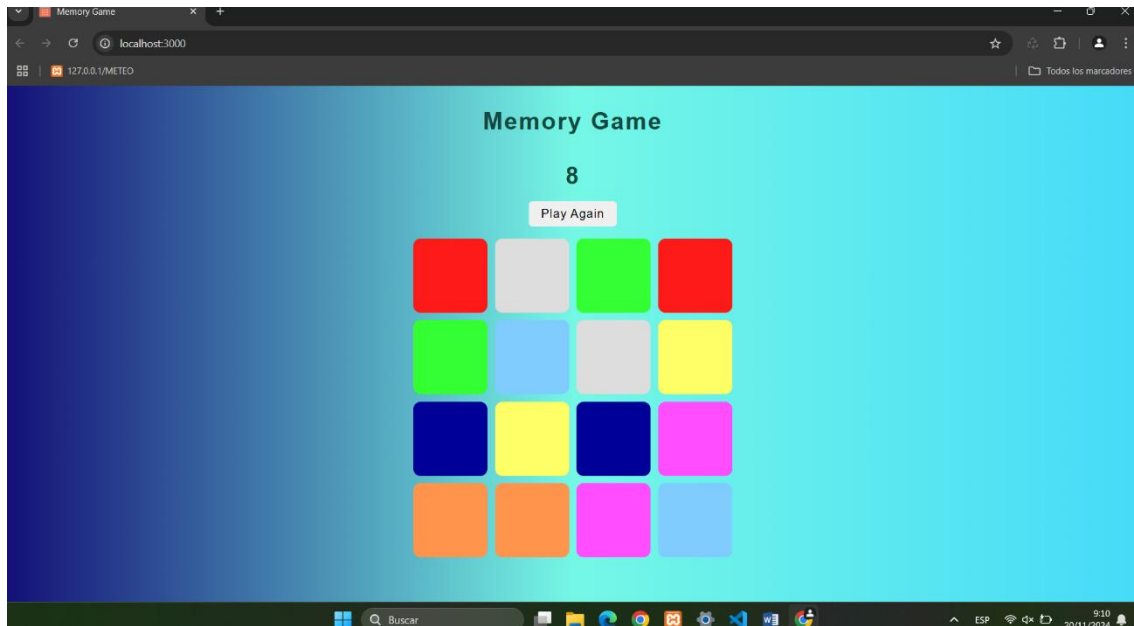




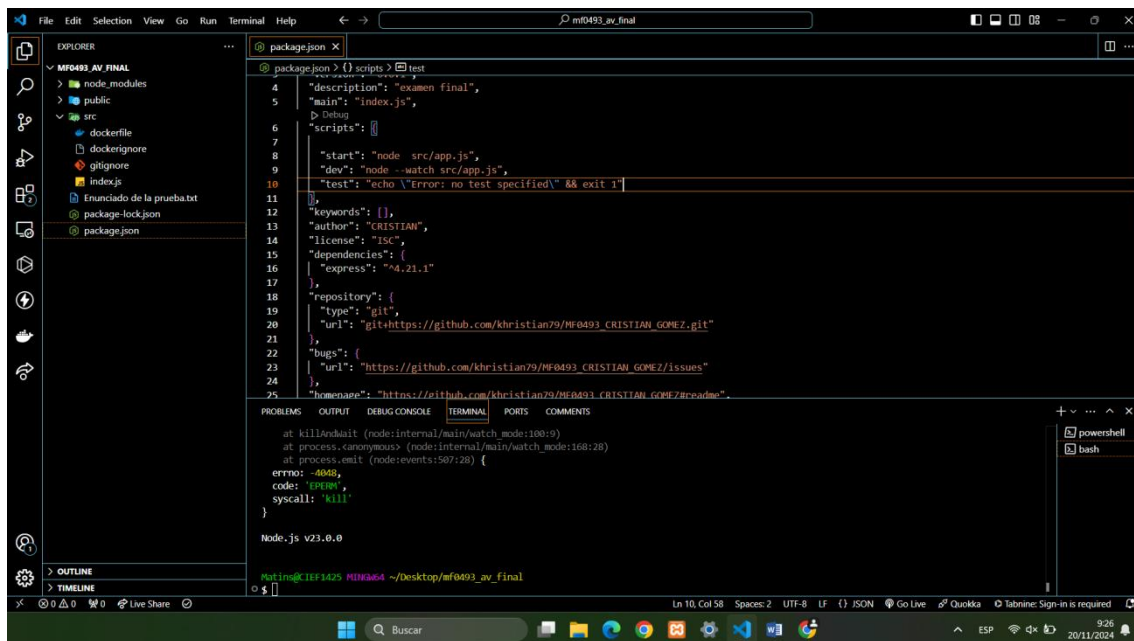
Creamos el repositorio de git hub para subir el programa.



Creamos los archivos packages.json/ package-lock.json. y su contenido.



Miramos si funciona la aplicación.



Insertamos en el package.json, dentro del scripts: el "start": , "dev":, "test":

The screenshot shows the Visual Studio Code interface with a project named 'mf0493_av_final'. The Explorer panel on the left shows the file structure, including 'node_modules', 'public', 'src', 'dockerfile', 'dockerignore', 'index.js', 'Enunciado de la prueba.txt', 'package-lock.json', and 'package.json'. The Dockerfile editor is open, showing the following content:

```
src > dockerfile > ...
1 # Identificar qué imagen queremos utilizar
2 FROM node:23-slim
3
4 # Indicar el directorio de trabajo
5 WORKDIR /app
6
7 # Copiar los archivos de package.json y package-lock.json
8 COPY package*.json ./
9
10 # Instalar las dependencias
11 RUN npm install \
12 && npm cache clean --force \
13 && rm -rf /tmp/* /root/.npm/_cacache
14
15 # Copiar el resto de los archivos
16 COPY . .
17
18 # Exponer el puerto en el que la aplicación se ejecutará
19 EXPOSE 3000
20
21 # Comando para ejecutar la aplicación
22 CMD ["npm", "start"]
```

The Terminal panel at the bottom shows the output of the Docker build process, including the command 'at killAndWait' and the error message 'errno: -4048, code: 'EPIPE', syscall: 'kill''. The status bar at the bottom indicates the file is 'Ln 19, Col 12' and the encoding is 'UTF-8'.

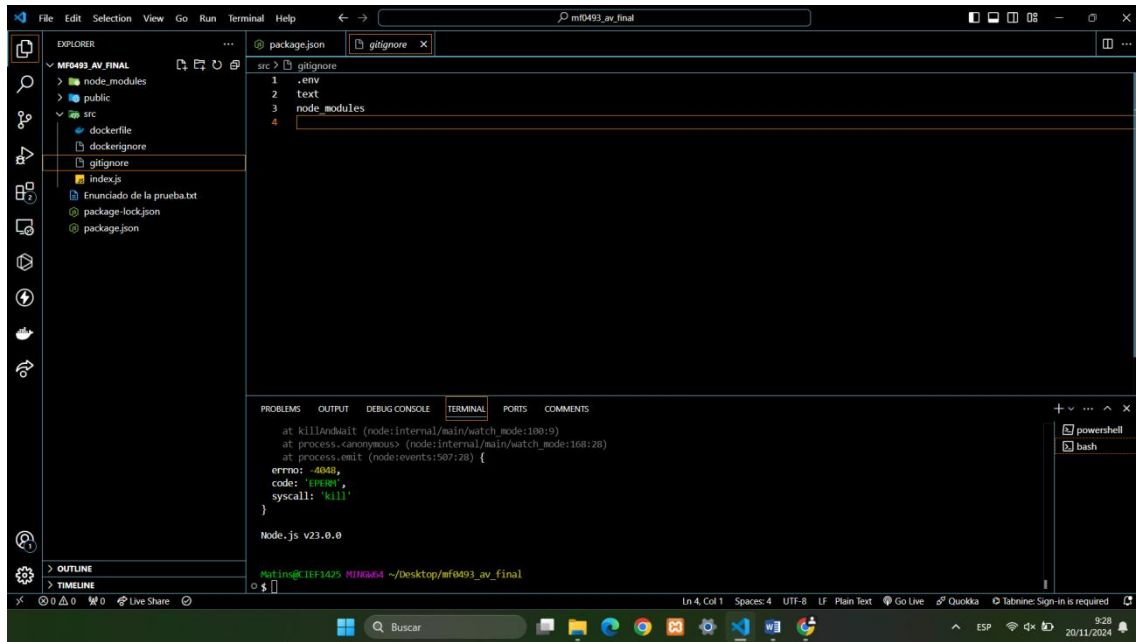
Introducimos los datos de configuración en el Dockerfile.

The screenshot shows the Visual Studio Code interface with the same project. The Dockerignore editor is open, showing the following content:

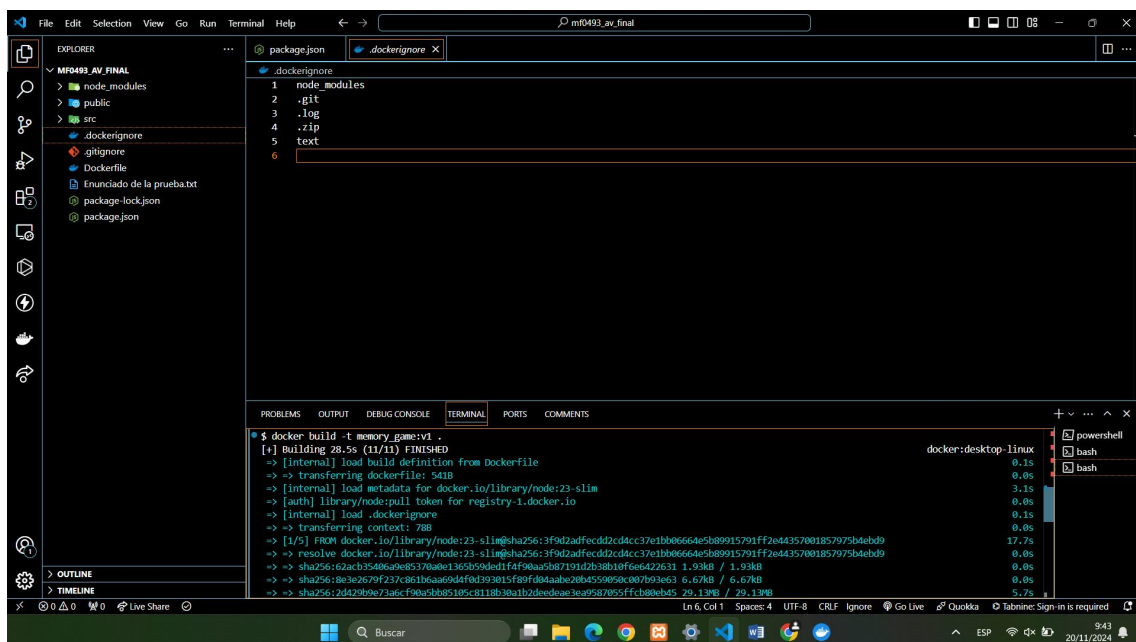
```
src > dockerignore
1 node_modules
2 .git
3 .log
4 .zip
5 text
6
```

The Terminal panel at the bottom shows the same output as the previous screenshot, including the command 'at killAndWait' and the error message 'errno: -4048, code: 'EPIPE', syscall: 'kill''. The status bar at the bottom indicates the file is 'Ln 6, Col 1' and the encoding is 'UTF-8'.

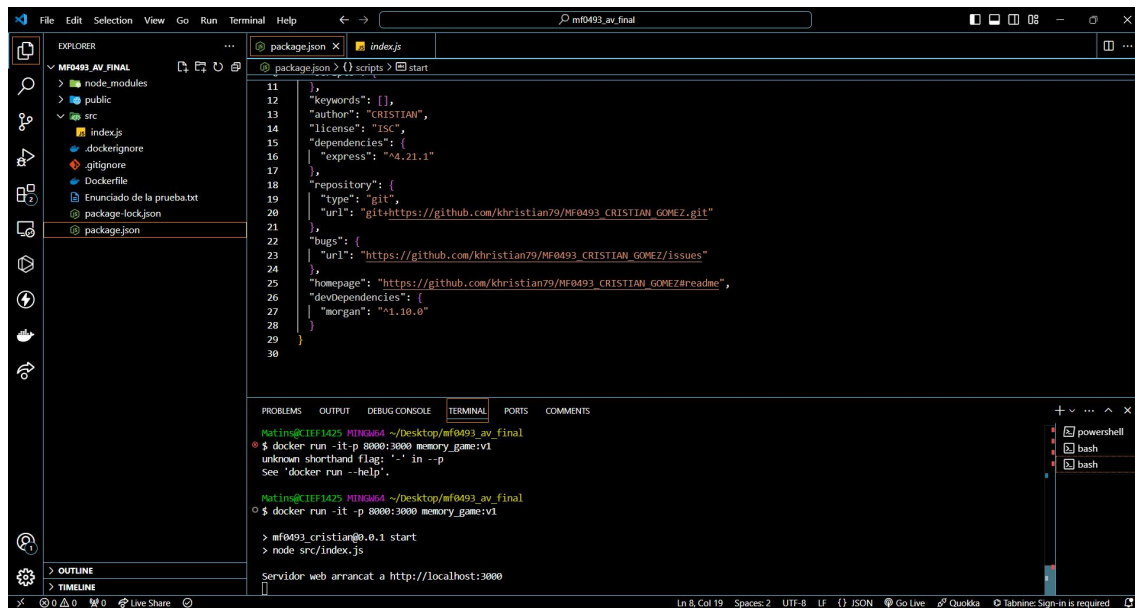
Ponemos las extensiones y los archivos que queremos ignorar en el dockerignore para que no se suba.



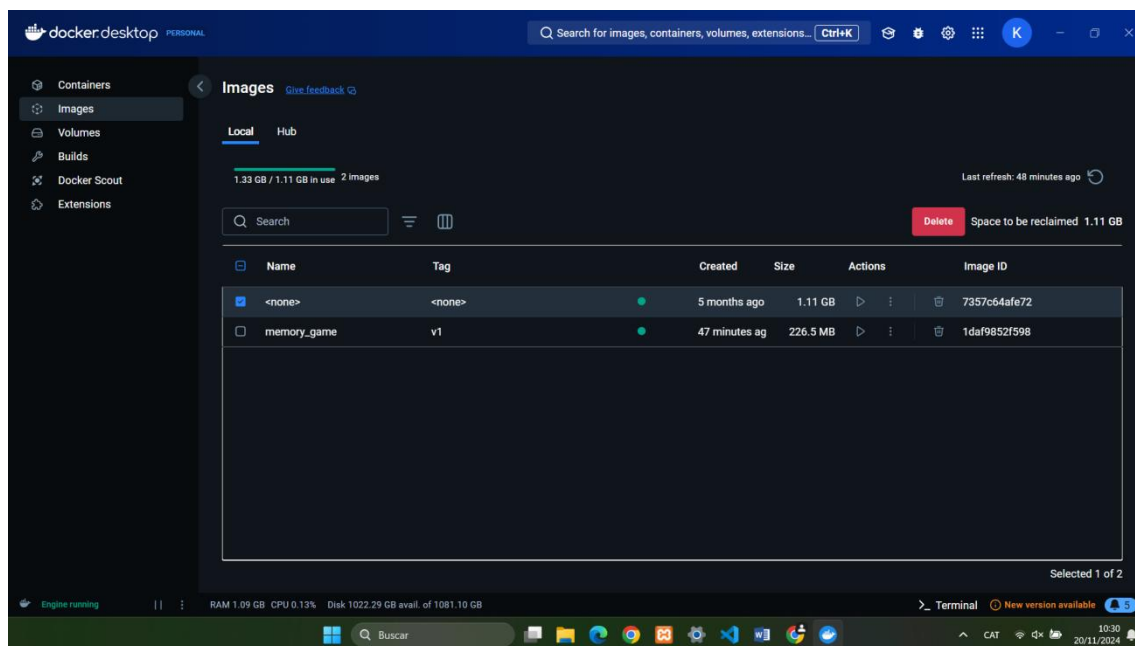
Ponemos los archivos y documentos en el gitignore.



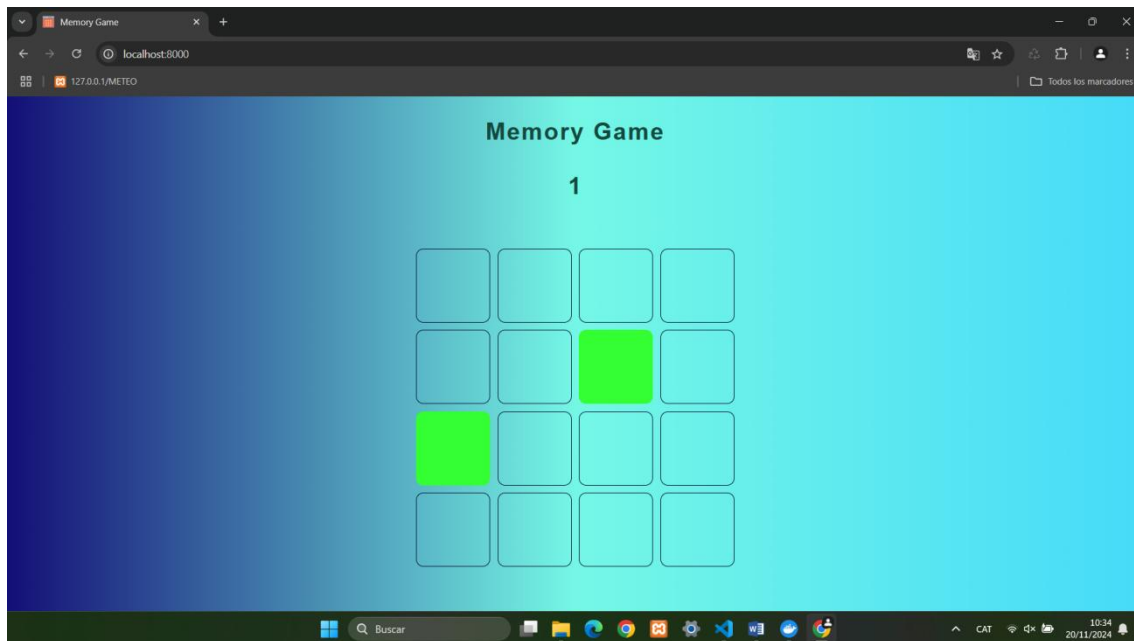
Construimos la imagen de memory_game.



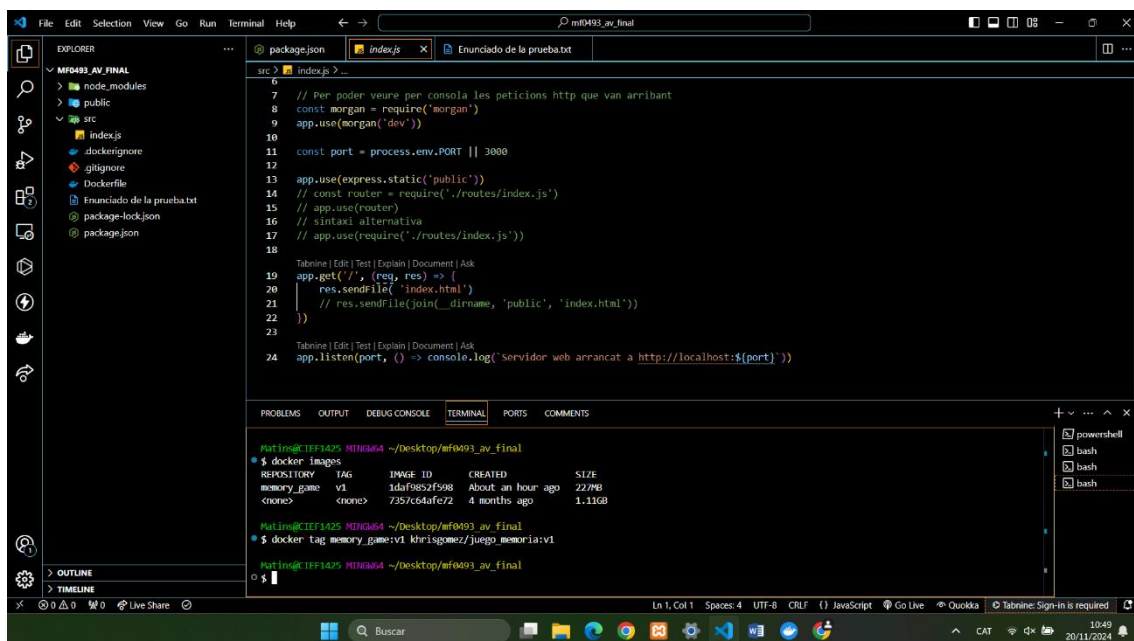
Creamos el contenedor.



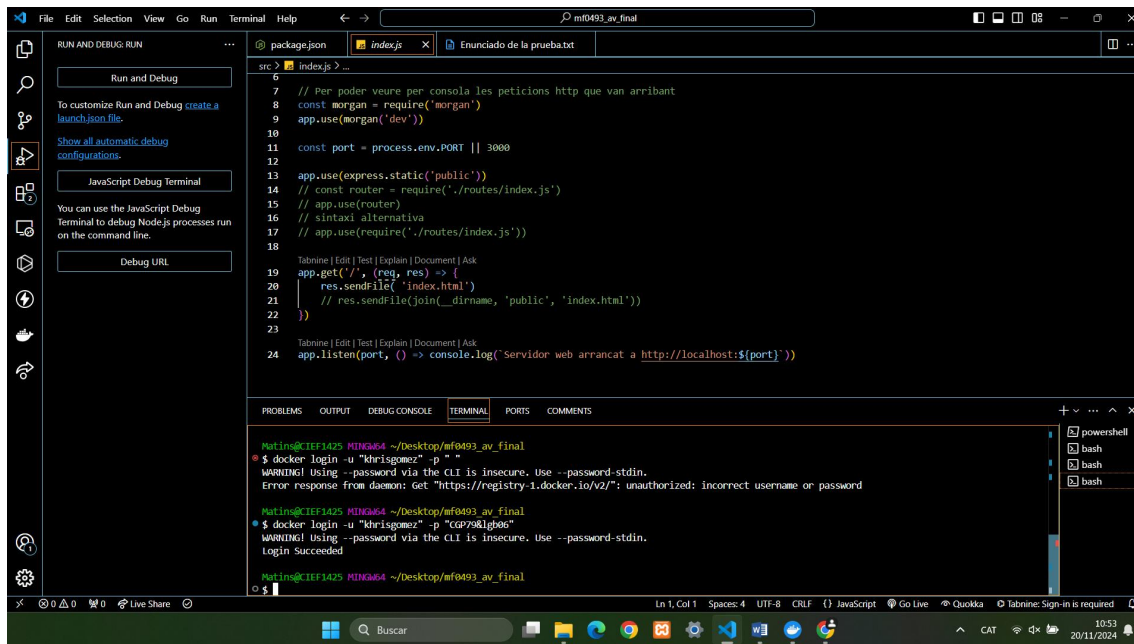
Hacemos una captura para ensenar que tenemos la imagen creada en docker desktop.



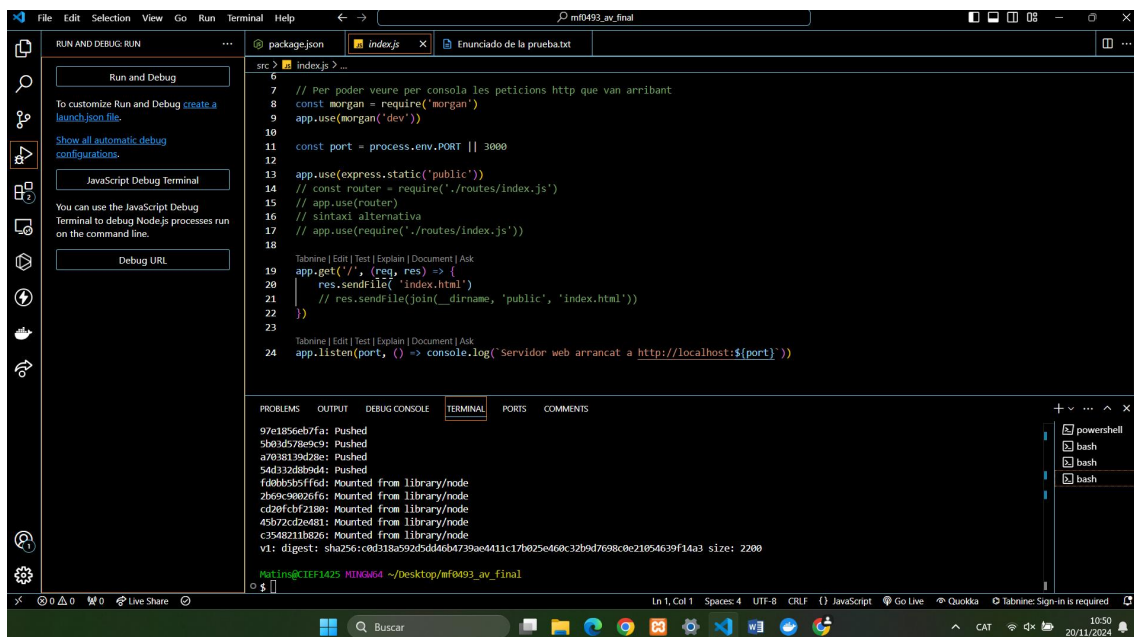
Miramos si funciona el programa de la imagen creada en docker desde el localhost: 8000



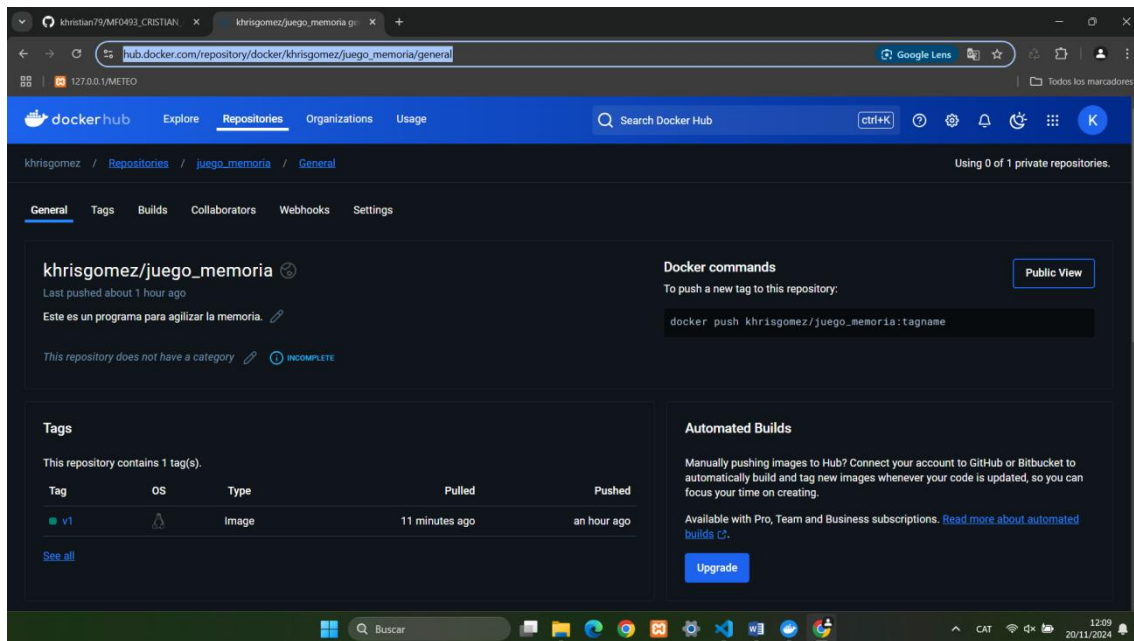
Le asignamos el nuevo nombre a la imagen Docker creada.



Realitzo el login. Y utilizamos la contraseña que tenemos



Subimos la imagen al docker hub.



https://hub.docker.com/repository/docker/khrisgomez/juego_memoria/general

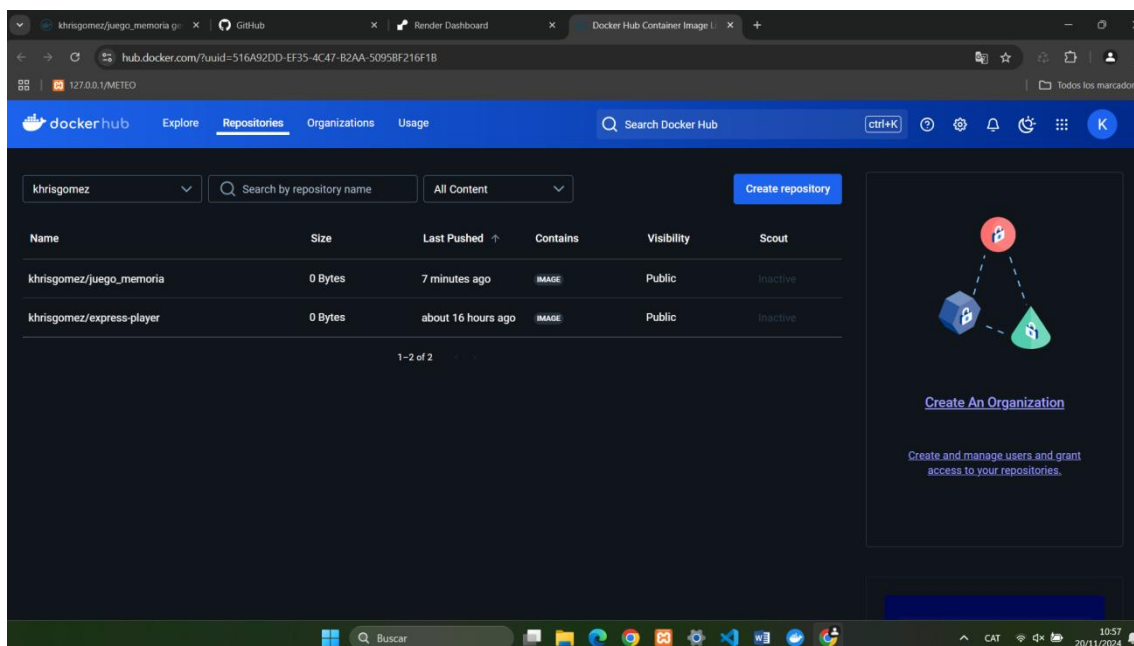
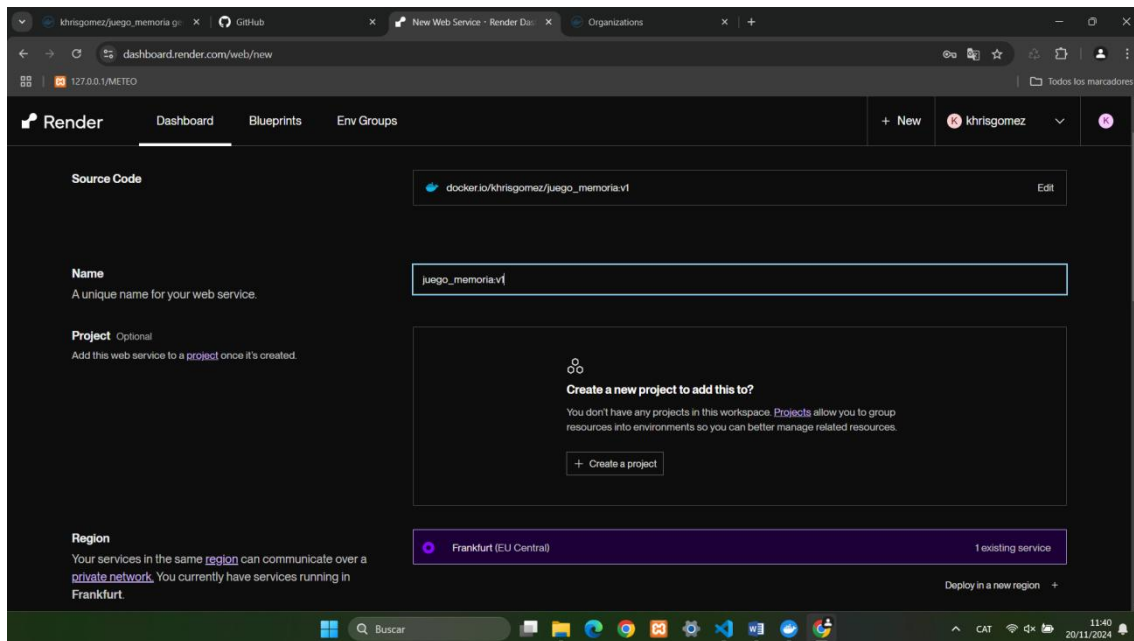
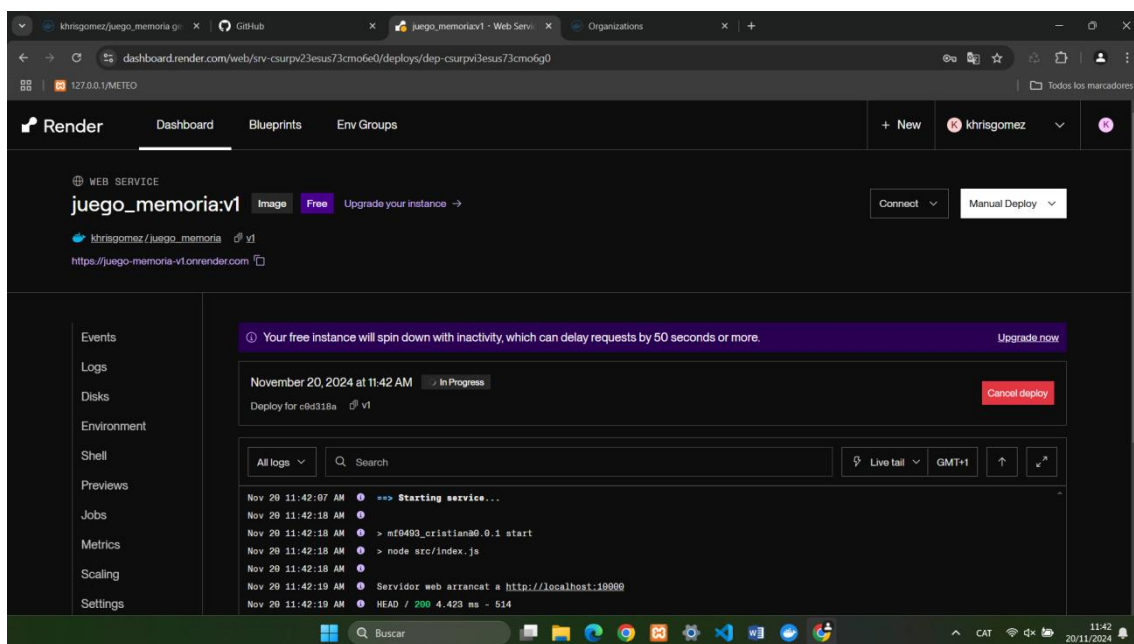


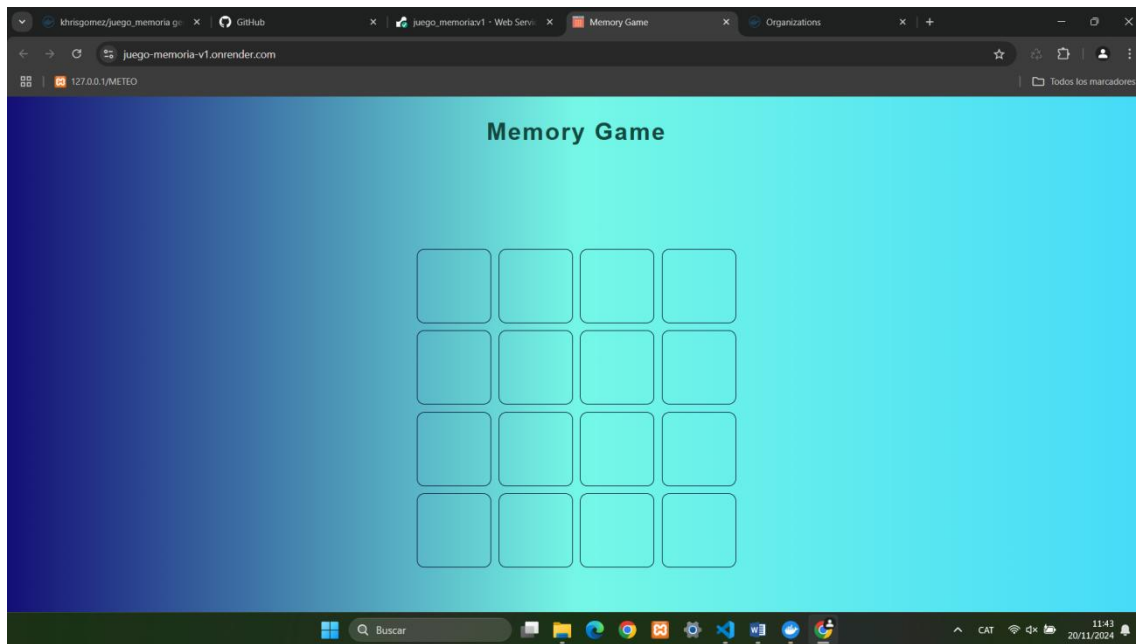
Imagen en docker hub.



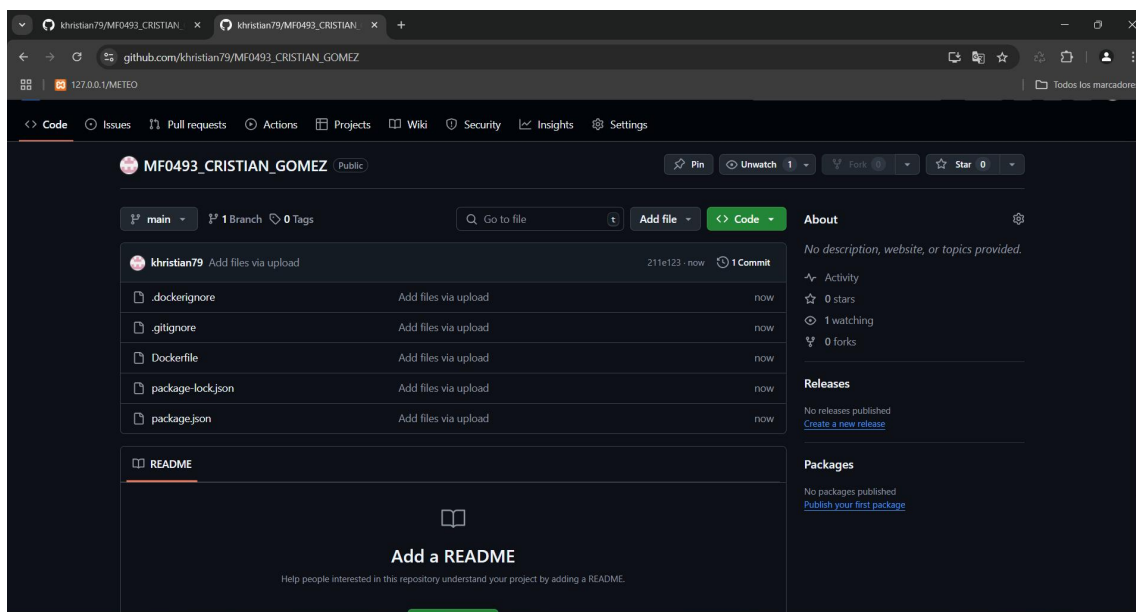
Desplegamos el programa dentro de render y le damos la región Frankfurt.



Creamos el <https://juego-memoria-v1.onrender.com>. Para que el programa se ejecute en web.



Ejecutamos para ver si funciona, el programa desde el https.



Subido a github.

Se crea un repositorio en Github

https://github.com/khristian79/MF0493_CRISTIAN_GOMEZ

