

Memoria Sistemas Multiagentes

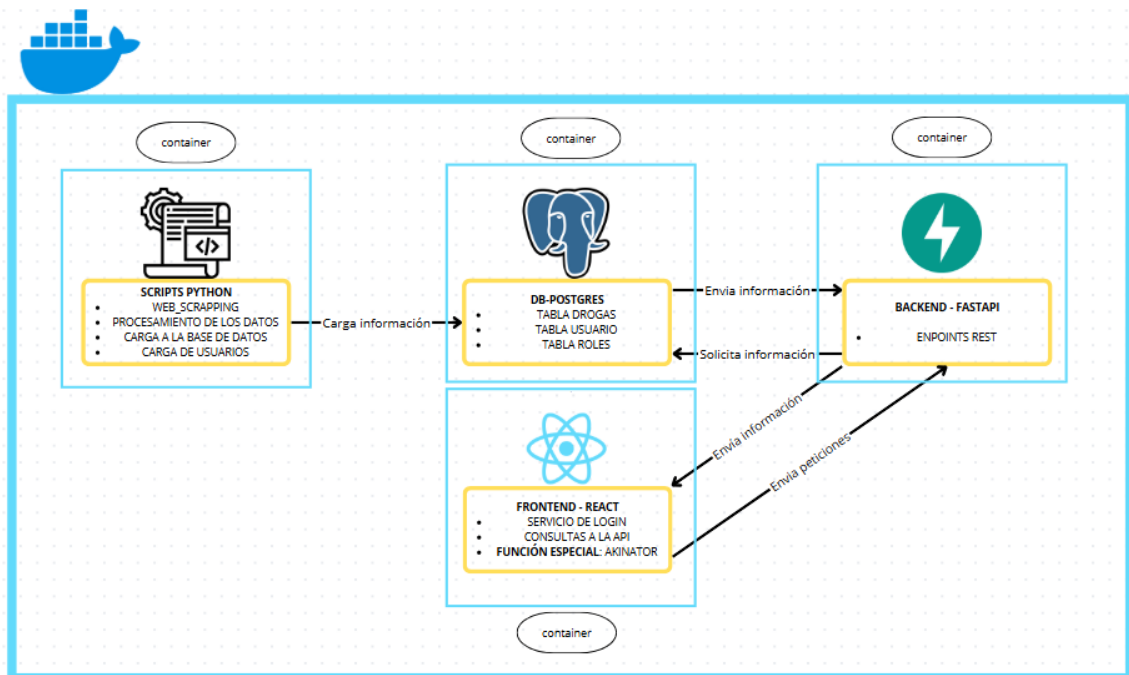
Manuel Cano García

Alberto Barraís Bellerín

Alejandro Cañas Borreguero

Sergio Pozuelo Martín-Consuegra

1. Arquitectura



Para la estructura general, hemos querido usar Docker ya que es una herramienta super usada en el mercado laboral vista en clase y hemos decidido de usarla para poder pelearnos con la herramienta y aprender una cosa más.

La arquitectura general de nuestra aplicación consta de varios módulos dentro de un orquestador en Docker, donde cada módulo está representado por un contenedor con una funcionalidad en concreto:

- **Scripts Python:** Modulo que se encargará de realizar todo el proceso del ETL, desde el proceso de extracción de datos a través de web scrapping, aprovechamiento de datasets utilizados en la práctica de minería de datos, siguiendo con el proceso de transformar los datos en información útil que queramos meter dentro de nuestra base de datos y por último una vez procesado esos datos, cargarlos a una base de datos. También se encargará de cargar por defecto un conjunto de usuarios con distintos

roles para que se pueda probar el diferente comportamiento según el rol que desempeña dichos usuarios.

Hemos decidido combinar tanto el aprovechamiento de los datasets como el uso del web scrapping para combinar metodologías




- **DB-Postgres:** Modulo dónde estará alojada la base de datos. Hemos elegido postgres porque la hemos visto en clase y querríamos utilizarla para afianzar conocimientos sobre las bases de datos relacionales.

Dicha base de datos estará compuesta principalmente por 3 tablas:

- 1) **Tabla drogas:** Guardaremos la información relacionada de las drogas más convencionales de nuestra sociedad como el alcohol, tabaco, marihuana entre otras.
- 2) **Tabla usuario:** Guardaremos la información de nuestro usuario, como puede ser su nombre, su contraseña codificada como un hash por motivos de seguridad y el rol que tiene esa persona dentro de la aplicación.

El rol del usuario es muy importante a la hora de poder utilizar las funcionalidades de nuestra aplicación, porque no todo el mundo va a tener acceso a todas las funcionalidades, esto lo hacemos por temas de seguridad donde un usuario “básico” no pueda modificar el contenido de la base de datos o funciones más enfocadas a la gestión de dicha información que no debería ser accesible para un tipo de usuario básico.

- 3) **Tabla rol:** Guardaremos tanto el tipo de rol como sus privilegios a la hora de poder gestionar la información almacenada en la base de datos. Con esto en mente tenemos 3 tipos de roles con sus correspondientes privilegios:

ROL	can_get	can_post	can_put	can_delete
 SIGMA	✓	✓	✓	✓
 MEWER	✓	✓	✗	✗
 RIZZLER	✓	✗	✗	✗

- **Fastapi:** Modulo donde tendremos nuestro backend, hemos decidido usarla ya que es una herramienta vista en clase y hemos querido explorarla más. En este módulo, tendremos la API con los correspondientes endpoints para que el usuario pueda, dependiendo de su rol, poder hacer unas cosas u otras con los datos.

Los endpoints correspondientes a la API REST son:

- **/get-drug/{id}** (GET): Obtiene información de una droga especificada por su ID.
- **/get-drug-by-name/{name}** (GET): Obtiene información de una droga especificada por su nombre.
- **/post-drug** (POST): Crea un nuevo registro de droga en la base de datos.
- **/delete-drug/{id}** (DELETE): Elimina una droga especificada por su ID.
- **/delete-drug-by-name/{name}** (DELETE): Elimina una droga especificada por su nombre.
- **/update-drug/{id}** (PUT): Actualiza la información de una droga especificada por su ID.
- **/update-drug-by-name** (PUT): Actualiza la información de una droga especificada por su nombre.

Otros endpoints:

- **/token** (POST): Maneja el inicio de sesión de usuarios y genera tokens JWT para la autenticación.
- **/healthcheck** (GET): Verifica que el servicio está funcionando correctamente.
- **/get-questions** (GET): Obtiene las preguntas para el cuestionario de Akinator.
- **/get-response** (POST): Procesa las respuestas del cuestionario y devuelve una predicción.

Flujo de autenticación:

Login (/token):

1. Recibe username/password via *OAuth2PasswordRequestForm*
2. Verifica credenciales contra base de datos
3. Genera token JWT si autenticación exitosa

Verificación de Token:

1. Middleware *get_current_user* verifica token en cada request
2. Decodifica JWT usando *SECRET_KEY*
3. Valida expiración y payload

- **Página web:** Para la página web hemos decidido usar el lenguaje de programación react ya que es uno de lo más usado en el desarrollo web y uno de los que tiene más documentación y aprovechar esta práctica para aprender un poco de desarrollo web.

La página web consistirá en:

- Un **login** donde la persona escribirá su usuario y contraseña para poder acceder a la página principal de la aplicación.
- La **página principal** consistirá en una serie de botones donde el usuario poder hacer las diferentes consultas a la base de datos (siempre y cuando tenga los privilegios adecuados para poder hacerlo). También tendremos un botón extra para poder entrar a la funcionalidad especial que más adelante vamos a hablar.
- La **funcionalidad especial** de nuestra practica hemos decidido ambientarlo al tópico de nuestra practica de minería, que consistía en diversas adicciones. El enfoque de esta funcionalidad se basa en el funcionamiento intuitivo de la famosa página de ordenador “Akinator” pero en vez de predecir lo que estamos pensando (personajes históricos, youtubers, actores...), lo que va a hacer nuestro “Addictinator” será predecir qué tipo de adicción la persona que lo está usando podría tener.

En esencia, el usuario tendrá que responder una serie de preguntas relacionadas al consumo de drogas con “Sí”, “No”, “No sé”, “Probablemente” y “Probablemente no”. Después, se recogerán las respuestas del usuario y para la generación de la respuesta, se lo enviaremos a un LLM para que dictamine un resultado.

Una pequeña motivación respecto a esta funcionalidad es mencionar el uso responsable de este tipo de drogas, aunque algunas su consumo puede ser ilegal, mucha gente cae en la tentación de consumirlas, otras como el alcohol o el tabaco, son mejores vista, pero su uso excesivo puede tener efectos devastadores en la vida de la persona afectada y sus familias, por lo que un entorno preventivo para poder interceptar posibles adicciones es bien recibido.

2. Manual de Usuario

a. Prerequisitos

Prerequisitos del Sistema

- Docker y Docker Compose instalados
- Sistema operativo compatible con Docker (Linux, Windows con WSL2, o macOS)
- Conexión a internet para descargar las imágenes de Docker
- Al menos 4GB de RAM disponible
- Espacio en disco para los contenedores y volúmenes (~1GB)

Requisitos de Software (se instalan automáticamente)

1. Backend (Python/FastAPI):

- Python 3.9 o superior
- FastAPI >= 0.113.0
- SQLAlchemy >= 2.0.0
- Pydantic >= 2.7.0
- Uvicorn >= 0.22.0
- psycopg2 >= 2.9
- python-jose para JWT
- bcrypt == 3.2.2
- passlib[bcrypt] == 1.7.4
- groq (para el modelo de IA)
- python-dotenv

2. Frontend (React):

- Node.js
- React >= 18.3.1
- React Router DOM >= 7.0.2
- React Scripts 5.0.1

3. Base de Datos:

- PostgreSQL 13

Archivos de Configuración Necesarios

Los `.env` no aparecen en el github pero van incluidos en el `.zip` de la entrega

1. `.env` en el directorio backend con:

- SECRET_KEY
- API_KEY (para el modelo groq)

2. `.env` en el directorio scripts con:

- DEFAULT_PASSWORD

Puertos Requeridos

- 3000: Frontend React
- 8000: Backend FastAPI
- 5432: PostgreSQL

Comandos para iniciar

Situado en el interior de la carpeta proyecto ejecutar el comando:

`docker compose -p drogas_no up --build`

b. Como usar la interfaz:

Si todo ha salido bien debería verse algo así:

```
[+] Running 9/9: Image
✔ Service python-scripts      Built
✔ Service react-app           Built
✔ Service fastapi-app         Built
✔ Network drogao_no_app-network Created
✔ Volume drogao_no_postgres_data Created
✔ Container react-app         Created
✔ Container postgres-db       Created
✔ Container python-scripts    Created
✔ Container fastapi-app       Created
Attaching to fastapi-app, postgres-db, python-scripts, react-app
react-app    /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
react-app    /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
react-app    18-listen-on-ipvs-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
react-app    18-listen-on-ipvs-by-default.sh: info: /etc/nginx/conf.d/default.conf differs from the packaged version
react-app    /docker-entrypoint.sh: Sourcing /docker-entrypoint.d/55-local-requires.env.sh
react-app    /docker-entrypoint.sh: Launching /docker-entrypoint.d/28-envsubst-on-templates.sh
react-app    /docker-entrypoint.sh: Launching /docker-entrypoint.d/28-tune-worker-processes.sh
react-app    /docker-entrypoint.sh: Configuration complete; ready for start up
react-app    2024/12/16 18:52:51 [notice] 1#1: nginx/1.27.3
react-app    2024/12/16 18:52:51 [notice] 1#1: built by gcc 13.2.1 202408309 (Alpine 13.2.1_git202408309)
react-app    2024/12/16 18:52:51 [notice] 1#1: OS: Linux 6.12.4-arch11
react-app    2024/12/16 18:52:51 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1024:524288
react-app    2024/12/16 18:52:51 [notice] 1#1: start worker processes
react-app    2024/12/16 18:52:51 [notice] 1#1: start worker process 29
react-app    2024/12/16 18:52:51 [notice] 1#1: start worker process 30
react-app    2024/12/16 18:52:51 [notice] 1#1: start worker process 31
react-app    2024/12/16 18:52:51 [notice] 1#1: start worker process 32
react-app    2024/12/16 18:52:51 [notice] 1#1: start worker process 33
react-app    2024/12/16 18:52:51 [notice] 1#1: start worker process 34
react-app    2024/12/16 18:52:51 [notice] 1#1: start worker process 35
react-app    2024/12/16 18:52:51 [notice] 1#1: start worker process 36
react-app    The files belonging to this database system will be owned by user "postgres".
react-app    This user must also own the server process.
postgres-db  The database cluster will be initialized with locale "en_US.utf8".
postgres-db  The default database encoding has accordingly been set to "utf8".
postgres-db  The default text search configuration will be set to "english".
postgres-db  Data page checksums are disabled.
postgres-db  fixing permissions on existing directory /var/lib/postgresql/data ... ok
postgres-db  creating subdirectories ... ok
postgres-db  selecting dynamic shared memory implementation ... posix
postgres-db  selecting default max_connections ... 100
postgres-db  selecting default shared_buffers ... 128MB
postgres-db  selecting default time zone ... Etc/UTC
postgres-db  creating configuration files ... ok
postgres-db  running bootstrap script ... ok
postgres-db  performing post-bootstrap initialization ... ok
postgres-db  syncing data to disk ... ok

postgres-db  Success. You can now start the database server using:
postgres-db  pg_ctl -D /var/lib/postgresql/data -l logfile start

postgres-db  initdb: warning: enabling "trust" authentication for local connections
postgres-db  You can change this by editing pg_hba.conf or using the option -A, or
postgres-db  --auth-local and --auth-host, the next time you run initdb.
postgres-db  waiting for server to start... 2024-12-16 18:52:52.068 UTC [48] LOG: starting PostgreSQL 13.18 (Debian 13.18-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
postgres-db  2024-12-16 18:52:52.061 UTC [48] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
postgres-db  2024-12-16 18:52:52.064 UTC [48] LOG: database system was shut down at 2024-12-16 18:52:53 UTC
postgres-db  2024-12-16 18:52:52.068 UTC [48] LOG: database system is ready to accept connections
postgres-db  done
postgres-db  server started
postgres-db  CREATE DATABASE
postgres-db  /usr/local/bin/docker-entrypoint.sh: ignoring /docker-entrypoint-initdb.d/*
postgres-db  waiting for server to shut down... 2024-12-16 18:52:52.323 UTC [48] LOG: received fast shutdown request
postgres-db  2024-12-16 18:52:52.324 UTC [48] LOG: aborting any active transactions
postgres-db  2024-12-16 18:52:52.326 UTC [48] LOG: background worker "logical replication launcher" (PID 55) exited with exit code 1
postgres-db  2024-12-16 18:52:52.326 UTC [50] LOG: shutting down
postgres-db  2024-12-16 18:52:52.334 UTC [48] LOG: database system is shut down
postgres-db  done
postgres-db  server stopped
postgres-db  PostgreSQL init process complete; ready for start up.
postgres-db  2024-12-16 18:52:52.455 UTC [1] LOG: starting PostgreSQL 13.18 (Debian 13.18-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
postgres-db  2024-12-16 18:52:52.456 UTC [1] LOG: listening on IPv6 address ":::0" port 5432
postgres-db  2024-12-16 18:52:52.456 UTC [1] LOG: listening on IPv6 address ":::0", port 5432
postgres-db  2024-12-16 18:52:52.457 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
postgres-db  2024-12-16 18:52:52.461 UTC [48] LOG: database system was shut down at 2024-12-16 18:52:53 UTC
postgres-db  2024-12-16 18:52:52.465 UTC [1] LOG: database system is ready to accept connections
python-scripts Database created
python-scripts Roles created
python-scripts Roles cargados
python-scripts Database created
python-scripts Roles cargados
python-scripts All users deleted successfully!
python-scripts Added user Juan with role sigma.
python-scripts Added user Maria with role sigma.
python-scripts Added user Jose with role mewer.
python-scripts Added user Ruben with role mewer.
python-scripts Added user Sofia with role rizzler.
python-scripts Added user Fran with role rizzler.
python-scripts Users loaded successfully!
python-scripts Usuarios cargados
python-scripts Datos extraídos
python-scripts Datos tratados
python-scripts Drugs loaded successfully!
python-scripts Datos cargados
fastapi-app  INFO: Started server process [1]
fastapi-app  INFO: Waiting for application startup.
fastapi-app  INFO: Application startup complete.
fastapi-app  INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
fastapi-app  INFO: 127.0.0.1:47398 - "GET /healthcheck HTTP/1.1" 200 OK
fastapi-app  INFO: 127.0.0.1:47398 - "GET /healthcheck HTTP/1.1" 200 OK
```

Una vez este todo iniciado correctamente diríjase a la dirección:

<http://localhost:3000/>

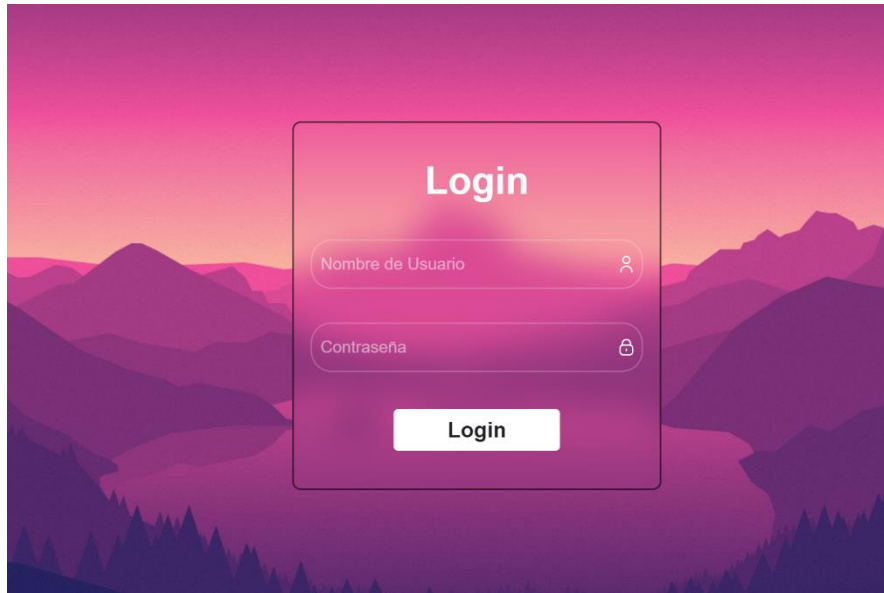
En caso de no poder acceder o no poder usar el menú asegúrese de que se ha cargado todo correctamente, si todavía no ha salido el mensaje:

```
python-scripts exited with code 0
fastapi-app | INFO: Started server process [1]
fastapi-app | INFO: Waiting for application startup.
fastapi-app | INFO: Application startup complete.
fastapi-app | INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
fastapi-app | INFO: 127.0.0.1:47398 - "GET /healthcheck HTTP/1.1" 200 OK
```

espere porque puede tardar un poco.

Login

Esta página nos permitirá redirigirnos al menú principal, tras introducir un usuario válido que se encuentre en la base de datos:



Dependiendo del usuario con el que te registres tendrás un tipo de rol, el cual afectará a los permisos que tengas en el Menú Principal.

Los usuarios validos en nuestra página son:

Rol= sigma: Juan y Maria


Rol= mewer: Jose y Ruben

Rol= rizzler: Sofia y Fran

Menú Principal

El menú principal consta de 5 botones que son:

- 1- Añadir: Este botón permitirá añadir diferentes drogas en la base de datos, tras pulsarlo se te abrirá un formulario en el que deberás introducir los diferentes datos necesarios para añadir dicha droga.

Añadir Nueva Droga 

Nombre de la Droga

Efectos Inmediatos

Efectos a Largo Plazo

Historia


Rango de Edad de Consumo

Frecuencia de Consumo

Probabilidad de Abandono

Añadir Droga

- 2- Obtener: Este botón permitirá obtener la información de una droga que se encuentre en la base de datos, tras pulsarlo únicamente deberá introducir el nombre de la droga que desea saber su información

Consultar Detalles de Droga 

Nombre de la Droga

Buscar

Tras pulsar en buscar verás una pestaña como la que se muestra a continuación en la que tendrás toda la información:

Consultar Detalles de Droga



Nombre de la Droga

Alcohol

Buscar

Detalles de la Droga

Nombre: Alcohol

Efectos Inmediatos: Intoxicación etílica, conductas de riesgo
Efectos a Largo Plazo: Hipertensión arterial, Alteraciones del sueño, Gastritis, Agresividad, Úlcera gastroduodenal, Depresión, Cirrosis hepática, Disfunciones sexuales, Cardiopatías,

Deterioro cognitivo, Encefalopatías, Demencia, Cáncer, Psicosis
Historia: Los persas conocieron el alcohol extraído del vino por destilación a partir del sigloIX. El alquimista persaMuhammad ibn Zakariyā al-Rāzīperfeccionó sus métodos de destilación.[5]

Sin embargo, en Europa su descubrimiento se remonta a principios del sigloXIV, atribuyéndose al médicoArnau de Villanova,alquimistay profesor de medicina enMontpellier. La quinta esencia deRamon Llullno era otra cosa que el alcohol rectificado a una más suave temperatura.Lavoisierfue quien dio a conocer el origen y la manera de producir el alcohol por medio de la fermentación vínica, demostrando que bajo la influencia de la levadurade cerveza, el azúcar de uva se transforma enácido carbónicoy alcohol. Fue además estudiado porScheele,Gehle,Thénard, Duma yBoullayy en 1854,Berthelotlo obtuvo por síntesis.[6]

Rango de Edad: 14 años

Frecuencia de Consumo: 80

Probabilidad de Abandono: 0.2325720659895497

- 3- Modificar: Este botón permitirá modificar la información de una droga que ya se encuentre en la base de datos, tras pulsarlo veras algo como:

Introducir Nombre de Droga



Nombre de la Droga

Alcohol|

Buscar Droga

Tras escribir el nombre de la droga que deseas modificar, únicamente tendrás que pulsar el botón Buscar Droga el cual te mostrará la información de dicha droga con la posibilidad de modificarla

Editar Droga

Nombre de la Droga

Alcohol

Historia

Los persas conocieron el alcohol extraído del vino por destilación a partir del sigloIX. El alquimista persaMuhammad ibn Zakariyā al-Rāziperfeccionó sus métodos de destilación.151 Sin embaroo, en

Efectos Inmediatos

intoxicación etílica, conductas de riesgo

Efectos a Largo Plazo

Hipertensión arterial, Alteraciones del sueño, Gastritis, Agresividad, Úlcera gastroduodenal, Depresión, Cirrosis hepática, Disfunciones sexuales, Cardiopatías, Deterioro cooñitivo, Encefalopatías.

Rango de Edad de Consumo

14 años

Frecuencia de Consumo

80

Probabilidad de Abandono

0.2325720659895497

Editar Droga

- 4- Eliminar: Este botón permitirá eliminar drogas que se encuentren en la base de datos, tras pulsarlo veras una pestaña en la que tras introducir el nombre de una droga que se encuentre en la base de datos y darle a Eliminar, la eliminará permanentemente de la base de datos.

Eliminar Droga

Nombre de la Droga a Eliminar

Ingrese el nombre de la droga

Eliminar

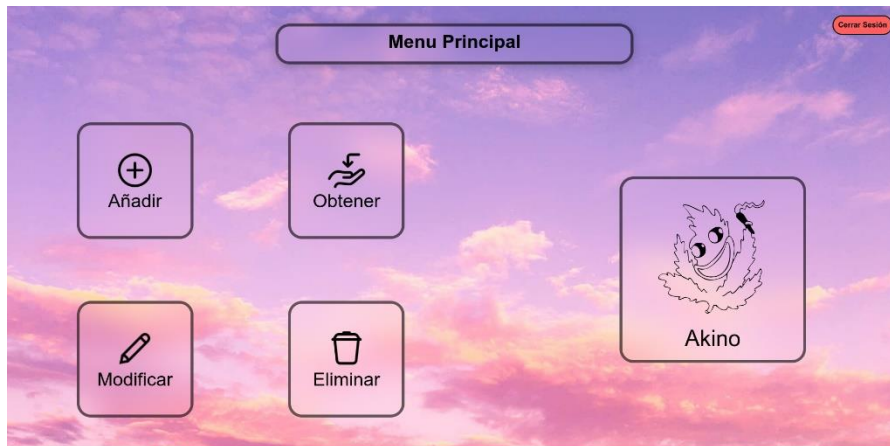
- 5- Akino: Este botón te permitirá redirigirte a la funcionalidad especial, la cual consiste en un Akinator que intentará deducir que adicción podrías tener.

A estos botones no tiene acceso todo el mundo, tus posibilidades vendrán dispuestas dependiendo del rol que tengas dentro de la aplicación, esto es con la intención de que un usuario corriente no pueda por ejemplo eliminar todos los

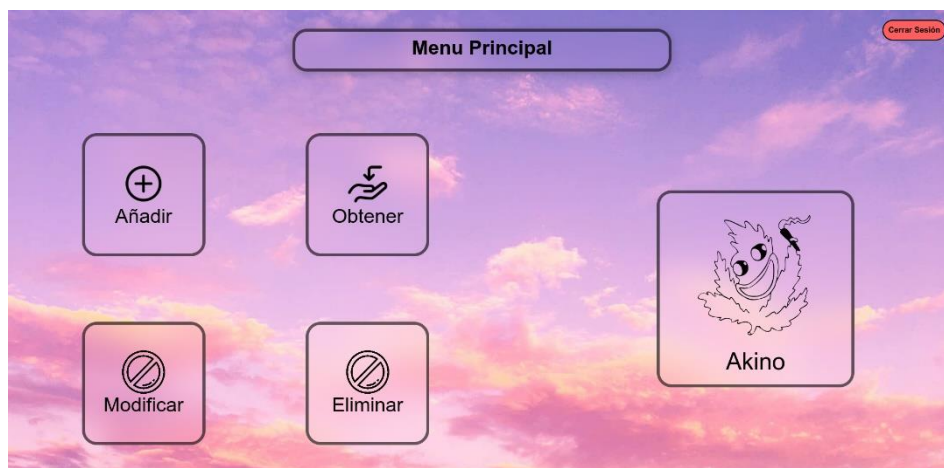
datos de la base de datos o añadir diferentes drogas las cuales tengan toda la información incorrecta.

Hay tres tipos de usuarios dependiendo del rol:

Si tienes el rol sigma, tendrás acceso a todo el menú y veras algo como:

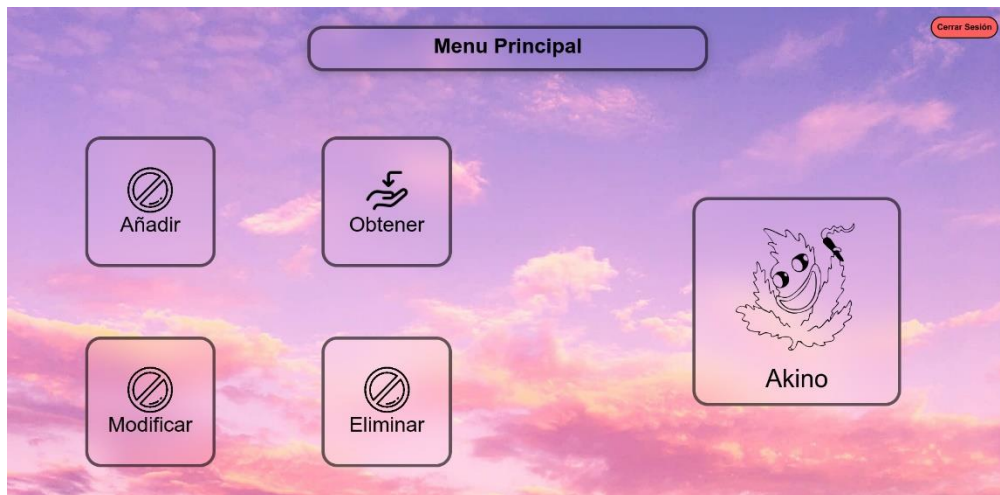


En caso de tener el rol mewer, verás algo como:



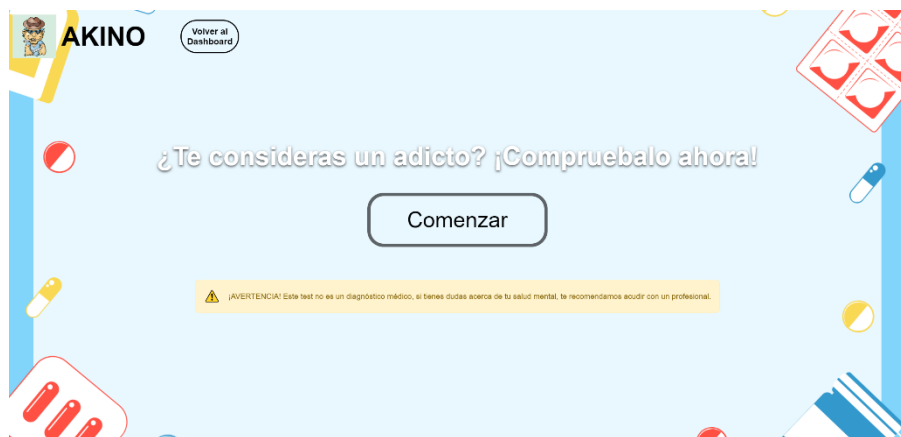
Como podemos ver en este rol, ya existen 2 botones a los cuales no tienes acceso

Por último, tendríamos el rol rizzler, en el que veras algo como:



Akino

Esta funcionalidad está disponible para cualquier tipo de usuario, consiste en un Akinator que permite intentar predecir qué tipo de adicción podrías tener. Esto no es ningún tipo de diagnóstico médico por lo que en caso de que realmente seas consciente de que podrías tener algún tipo de problema por favor consulte con un profesional. Tras pulsar el botón de Akino del Menu Principal veras algo como:

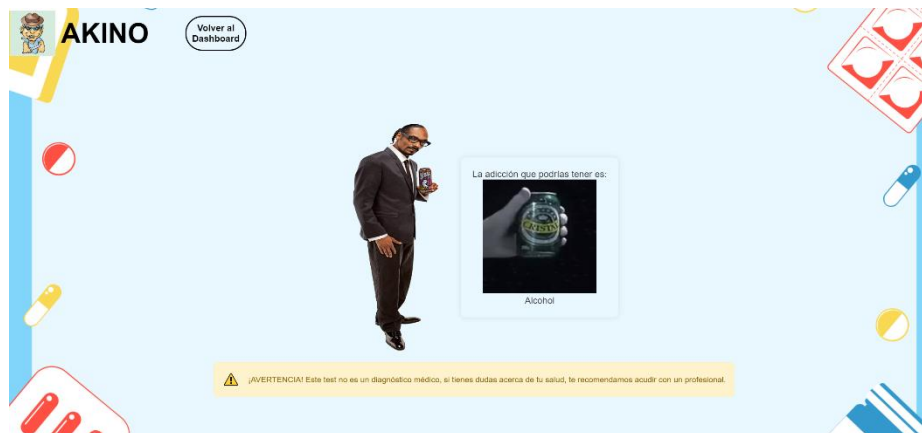


En este menú te permite volver al Menú Principal, para ello deberás pulsar el botón Volver al Dashboard que se encuentra en la parte superior de la página.

En caso de querer hacer el cuestionario pulsaríamos el botón Comenzar donde veríamos algo como:

The screenshot shows the AKINO app interface. At the top left is the AKINO logo and a 'Volver al Dashboard' button. A man in a suit is standing on the left, holding a smartphone. The main content area displays the question: '1/30 - ¿Te resulta difícil controlar tus emociones o impulsos en situaciones estresantes?'. Below the question are five green buttons with the following options: 'Si', 'No', 'No se', 'Probablemente', and 'Probablemente no'. At the bottom, there is a yellow warning box with a triangle icon and the text: '¡AVERTENCIA! Este test no es un diagnóstico médico, si tienes dudas acerca de tu salud, te recomendamos acudir con un profesional.'

Donde nos encontraremos con la primera pregunta de una lista de 30, las cuales deberás ir contestando según se adecuen mejor a tu estado, tras contestar todas las preguntas la página mostrará algo como:



En este caso tras realizar la prueba se podría considerar que tendríamos una adicción al alcohol.

3. Lecciones aprendidas

- 1- Tener desde el principio una mejor organización de ramas, ya que al principio era sencillo manejar las ramas, pero conforme fue creciendo el proyecto, no había una muy buena organización de que ramas eran las más actualizadas. Por ejemplo, inicialmente teníamos una rama llamada *dev/frontend*, pero terminamos utilizándola también para tareas de backend. Si tuviéramos que hacerlo de nuevo, optaríamos por crear una sola rama de desarrollo principal y subramas específicas para las adiciones de frontend o backend. Además, emplearíamos etiquetas para señalar claramente si los cambios corresponden a una u otra parte.
- 2- La importancia de ir documentando poco a poco el proyecto, para no tener que hacerlo de golpe, pudiéndose perder información relevante que habíamos considerado en el momento, pero por no plasmarlo en la documentación se nos ha olvidado. Llevar la documentación del proyecto al día nos permite plasmar eficazmente lo que pensamos durante todo el proceso del proyecto.
- 3- Fomentar una comunicación más clara para evitar problemas como la ubicación incierta de archivos *.env* o comentarios en las issues sobre añadidos o cambios que pasan desapercibidos.
- 4- Tener un mejor control de errores tanto en la API como en la página web, para identificar de forma más precisa la ubicación exacta del problema.