|  |  |
| --- | --- |
| **NOMBRE GRUPO:** | Infinity Quality |
| **FECHA:** | 31-01-2025 |
|  |  |
|  |  |
| **TEMA:** | Arquitectura de la aplicación MEMORIA ARTÍSTICA |
|  |  |

Arquitectura Front-end

La aplicación web Memoria Artística, desarrollada con React JS, sigue una arquitectura modular y escalable que facilita el mantenimiento, la reutilización de código y la separación de responsabilidades. A continuación, se describe la estructura de la aplicación:

1. **Componentes Reutilizables**

En esta carpeta se encuentran todos los elementos visuales y funcionales que son reutilizables en diferentes partes de la aplicación.

Ejemplos:

* Tangram: Elementos visuales interactivos.
* LoadingCircle: Indicadores visuales que muestran el estado de carga mientras se realizan peticiones al servidor.
* ProtectionRoute: Componente que protege rutas privadas, asegurando que solo usuarios autenticados puedan acceder.

Estos componentes son independientes y reciben datos a través de contextos, lo que los hace altamente reutilizables.

1. **Contextos**

Los contextos de React se utilizan para gestionar el estado global de la aplicación y compartir datos entre componentes sin necesidad de pasar props manualmente.

Ejemplos:

* AuthContext: Gestiona la autenticación del usuario (login, logout, estado de sesión).
* GameRulesContext: Almacena las reglas y el estado del juego.
* StateFetchContext: Maneja el estado de las peticiones al servidor (carga, éxito, error)
* UserContext: Gestiona la información del usuario y sus permisos.

Los contextos se combinan con custom hooks para encapsular la lógica y evitar el uso directo de servicios en las páginas.

1. **Custom Hooks**

Los custom hooks son funciones personalizadas que encapsulan lógica reutilizable, como la gestión de autenticación, el estado del juego o las peticiones al servidor.

Ejemplos:

* useAuth: Encapsula la lógica de autenticación (login, logout, verificación de sesión).
* useState: Gestiona el estado local de un componente o página.
* useTherapist: Encapsula la lógica relacionada con la gestión de terapeutas (creación, edición, eliminación).

Estos hooks se inyectan en los contextos para proporcionar la lógica necesaria a los componentes y páginas.

1. **Servicios**

Los servicios son módulos encargados de realizar las solicitudes al servidor (API) y manejar la lógica relacionada con la comunicación externa.

Ejemplos:

* AuthService: Gestiona las peticiones de autenticación (login, registro, verificación de token).
* CrudService: Realiza operaciones CRUD (Crear, Leer, Actualizar, Eliminar) relacionadas con los datos de la aplicación.

Los servicios utilizan principalmente utilidades para manejar errores, formatear datos o realizar operaciones repetitivas.

1. **Utilidades**

En esta carpeta se encuentran funciones y valores constantes que se utilizan en diferentes partes de la aplicación.

Ejemplos:

* constants: Almacena valores constantes como URLs de la API, mensajes de error, etc.
* handleApiError: Función que maneja errores comunes en las peticiones al servidor.
* Utils: Funciones auxiliares que realizan tareas repetitivas, como formatear fechas o validar datos.

Estas utilidades se utilizan en los servicios, hooks y componentes para evitar la duplicación de código.

1. **Páginas**

Las páginas representan las vistas principales de la aplicación y se encargan de renderizar los componentes necesarios.

Ejemplos:

* Home: Página de inicio.
* Login: Página de inicio de sesión.
* MainGame: Página principal del juego.
* Patient: Página con información del paciente como partidas, historial de transferencias.
* Dashboard: Página de paneles de control.
* RegisterTherapist: Página de registro de terapeutas.

Las páginas consumen los contextos y utilizan los componentes para construir la interfaz de usuario.

1. **Punto de Entrada (*App.js* e *index.js*)**

App.js`: Es el núcleo de la aplicación. Aquí se configuran los providers de los contextos y se definen las rutas principales.

index.js: Es el archivo que monta la aplicación en el DOM. Importa App.js y lo renderiza en el elemento raíz *(`<div id="root"></div>`*).

index.html: Es el archivo HTML base que sirve como punto de salida para la aplicación. Contiene el elemento raíz donde se monta la aplicación.

**Beneficios de esta Arquitectura**

* Modularidad: Cada parte de la aplicación está claramente separada, lo que facilita el mantenimiento y la escalabilidad.
* Reutilización: Los componentes, hooks y utilidades son reutilizables en diferentes partes de la aplicación.
* Separación de responsabilidades: La lógica de negocio (servicios), la gestión del estado (contextos) y la interfaz de usuario (componentes) están claramente separadas.
* Escalabilidad: Es fácil agregar nuevas funcionalidades sin afectar el resto del código.

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

Imagen 1. Diseño visual de arquitectura Front-end

Arquitectura Back-end

El back-end usado para guardar el progreso de los pacientes está hecho en Python y FastAPI, sigue una arquitectura modular y escalable que facilita el mantenimiento, la reutilización de código y la separación de responsabilidades. A continuación, se describe la estructura de la aplicación:

1. **Middlewares:**

Contiene el middleware verify\_session, que se encarga de verificar la autenticación del usuario en cada solicitud y utiliza las utilidades (jwt\_handle) para validar tokens JWT.

1. **Models:**

Define los modelos de la base de datos utilizando un ORM como Pydantic.

Ejemplos:

* Auth: Gestiona la autenticación de usuarios.
* Patient: Representa a los pacientes.
* Session: Gestiona las sesiones de los usuarios.
* Therapist: Representa a los terapeutas.
* Transfer: Gestiona las transferencias de datos.

1. **Routes:**

Define las rutas de la API además que cada ruta está asociada a un servicio específico.

Ejemplos:

* auth\_routes: Rutas para autenticación (login, registro, etc.).
* patient\_routes: Rutas para gestionar pacientes.
* session\_routes: Rutas para gestionar sesiones.
* therapist\_routes: Rutas para gestionar terapeutas.
* transfer\_routes: Rutas para gestionar transferencias.

1. **Services:**

Contiene la lógica de negocio de la aplicación y cada servicio está asociado a un modelo y a una ruta.

Ejemplos:

* auth\_service: Lógica para autenticación.
* patient\_service: Lógica para gestionar pacientes.
* session\_service: Lógica para gestionar sesiones.
* therapist\_service: Lógica para gestionar terapeutas.
* transfer\_service: Lógica para gestionar transferencias.

1. **Utils:**

Contiene funciones auxiliares reutilizables.

Ejemplos:

* bcrypt\_handle: Encargado de encriptar y verificar contraseñas.
* error\_handle: Maneja errores y excepciones de la aplicación.
* handle\_response: Formatea las respuestas de la API.
* jwt\_handle: Gestiona la creación y verificación de tokens JWT.

1. Punto de entrada:

main.py: Es el punto de entrada de la aplicación encargado de configura y arranca el servidor FastAPI, importar y utilizar las rutas, middlewares y la configuración de la base de datos.

**Beneficios de esta Arquitectura**

* Separación de responsabilidades: Cada módulo tiene una función clara y específica.
* Reutilización: Las utilidades y servicios son reutilizables en diferentes partes de la aplicación.
* Escalabilidad: Es fácil agregar nuevas rutas, servicios o modelos sin afectar el resto del código.
* Mantenibilidad: La lógica está organizada y modularizada, lo que facilita las pruebas y las actualizaciones.

Interfaz de usuario gráfica

Descripción generada automáticamente

Imagen 2. Diseño visual de arquitectura Back-end

Prototipos

El prototipo inicial presentado mostro a nuestro cliente fue corregido por nuestro cliente y aprobado, la funcionalidad principal que tendría la aplicación que es colorear espacios que coincidan las letras con los colores.

Forma

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente con confianza media

Imagen 3. Prototipo inicial de la aplicación Memoria Artística

Interfaces

Home es nuestra primera pantalla al ingresar a la aplicación, que tiene una única función de verificar si el usuario inicio sesión se redireccionara al dashboard de terapista y si no, entonces le redireccionará al de iniciar sesión.



Imagen 4. Diseño de pantalla de inicio

Inicio de sesión, la interfaz encargada de iniciar sesión, en caso exitoso se direcciona al dashboard del terapista caso contrario le aparece un mensaje indicándole detalles del error.

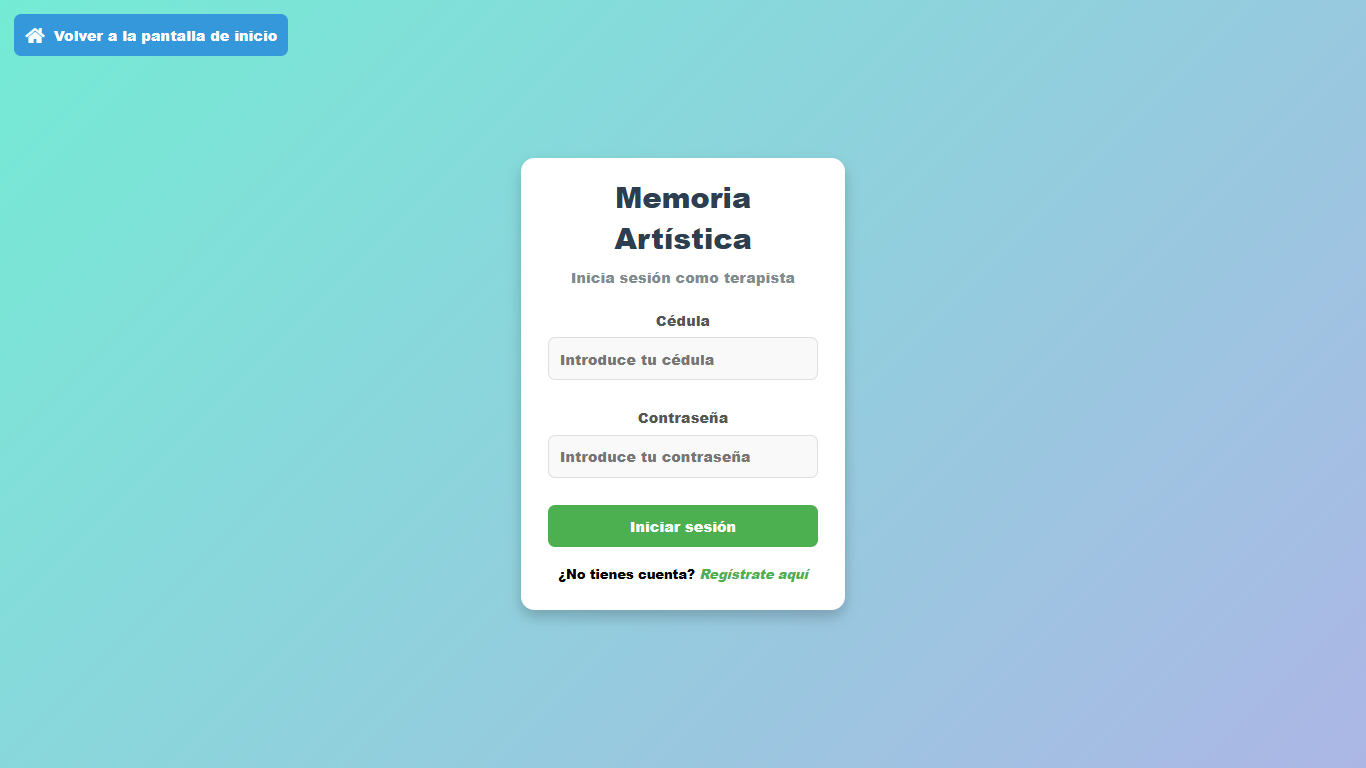


Imagen 5. Diseño de pantalla de inicio de sesión

Registro de usuarios, los terapistas, encargada de capturar todos los datos necesarios para el registro, con validadores de datos. Si el registro es exitoso entonces se redirige a la pantalla de inicio de sesión caso contrario le muestra un mensaje especificando el error.



Imagen 6. Diseño de pantalla de registro de nuevos terapistas

Dashboard del terapista, indica el nombre del terapista que ha iniciado sesión, todos los pacientes que están bajo su cuidado y un formulario en caso de querer agregar un nuevo paciente. En los pacientes, en caso de dar a “Jugar” iniciará una partida, en caso de dar “ver estadísticas” me mostrará en detalle el progreso del paciente, y en caso de dar “transferir” se abrirá un formulario para transferir el paciente a otro terapista.



Imagen 7. Diseño de pantalla de perfil de terapista con sus acciones

Pantalla principal, encargada de mostrar el juego en sí, dos tangram y los colores disponibles para colorear, estos tangram son aleatorios, así como las letras y los colores (dentro de una gama de colores brillantes que fue lo que uno de nuestros clientes, Sandra Sánchez, sugirió). Al terminar la partida muestra luego de un minuto los detalles de la partida como tiempo, aciertos y errores, estos datos servirán al terapista.

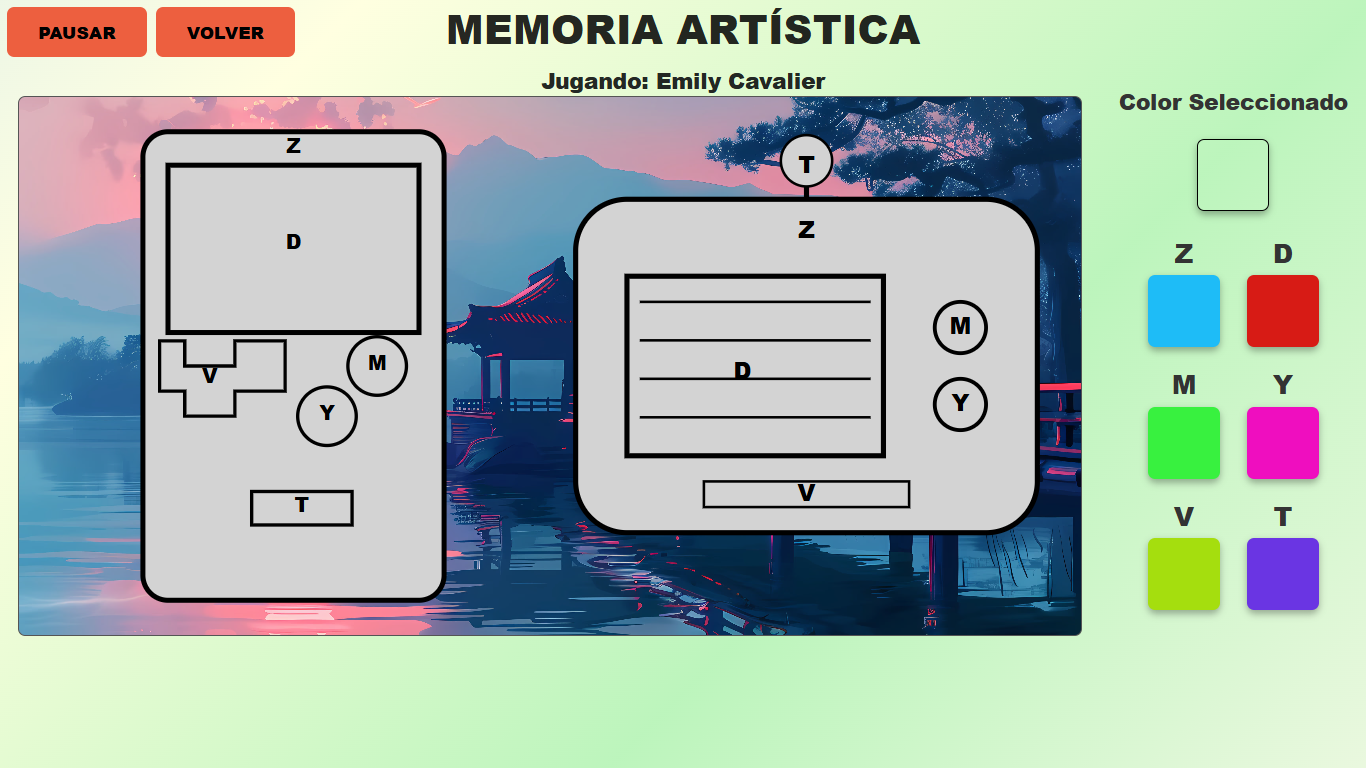


Imagen 8. Diseño de pantalla de principal

Detalles del paciente, nos muestra todas las partidas realizadas por el paciente, una interfaz muy intuitiva y simple, se añadió un diagrama de líneas para mostrar visualmente el progreso del paciente ayudando al terapista a evitar ver a detalle cada partida, este diagrama es interactivo se puede hacer uso de los filtros ubicado en la parte superior y permitiendo la descarga de la imagen generada.

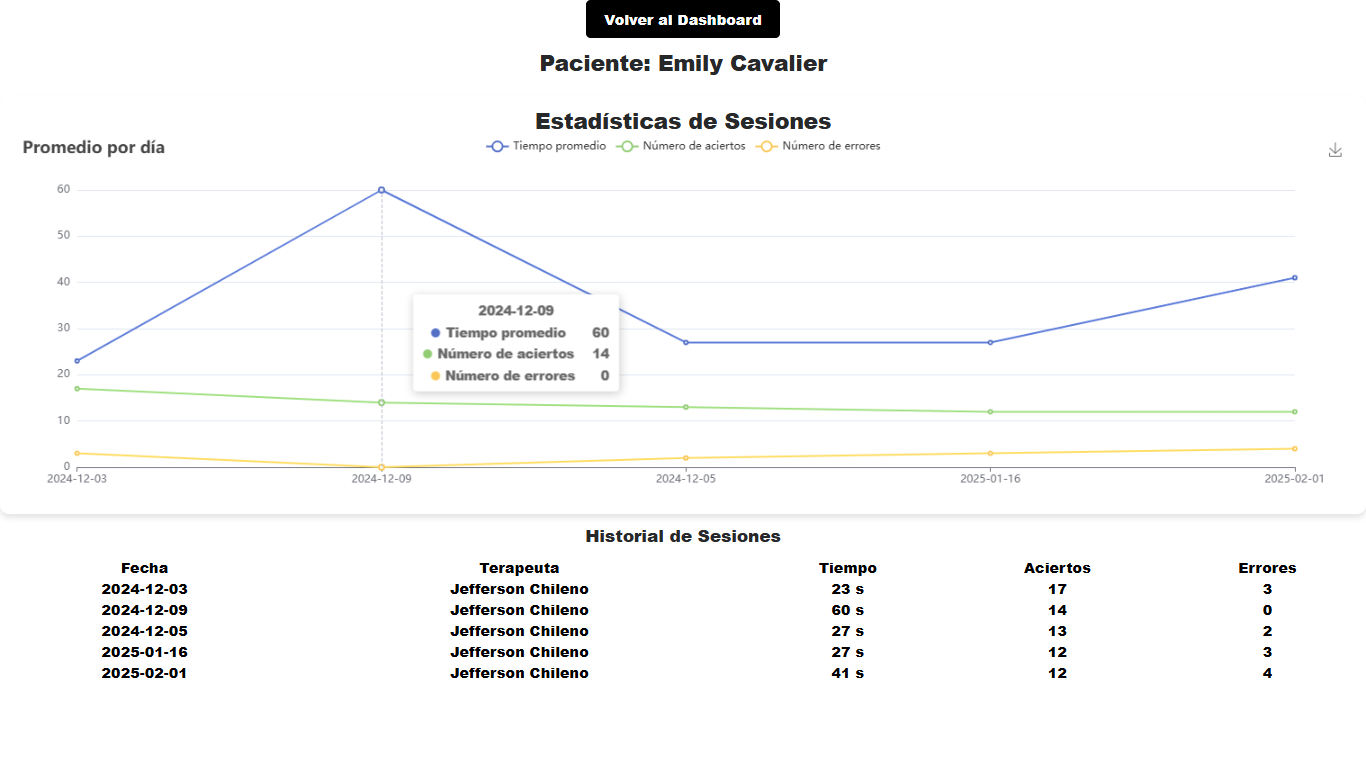


Imagen 9. Diseño de pantalla de inicio

Transferencia de pacientes encargado de recibir los datos del nuevo terapista encargado del paciente, adicionalmente se puede dejar un comentario sobre la transferencia.

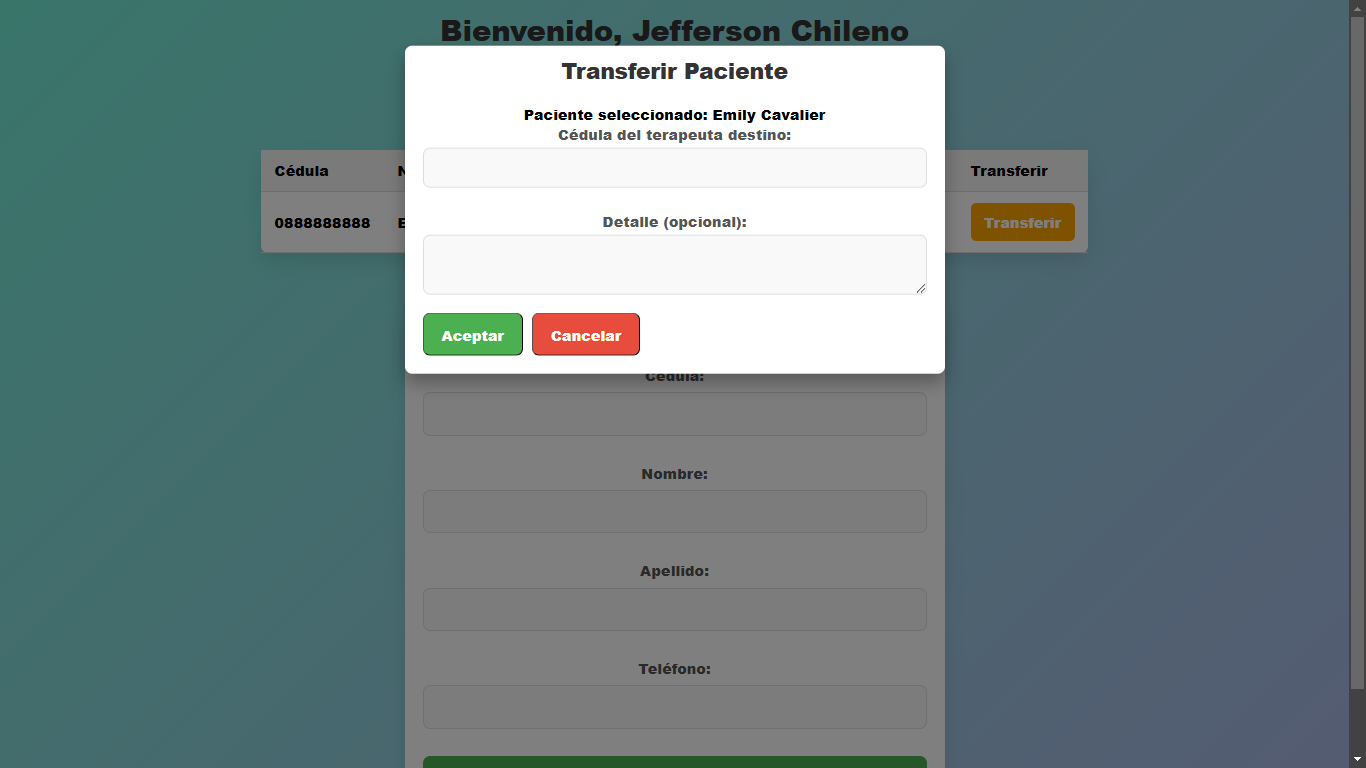


Imagen 10. Diseño de formulario de transferencia de paciente