FT_TRANSCENDENCE

PREPARACIÓN

LA WEB

(Opcional) Backend en Ruby puro	Puede ser otro si se usa el módulo **Framework**. Si se usa una base de datos, se deben usar las limitaciones del módulo **Database**.
Frontend en JavaScript puro vainilla	Puede ser otro si se usa el módulo **FrontEnd**.
La página web debe ser SPA	Es decir, se carga la página una vez y el resto se actualiza dinámicamente sin recargar.
Los botones Back y Forward	Deben manejar el **historial de estados** de la aplicación en vez del funcionamiento típico.
Compatible el último Chrome	Cosa rara que en los **Macs** esté instalado.
No errores ni advertencias sin gestionar	
Todo debe lanzarse con un único comando	Se usará **Docker**, por ejemplo: ***docker-compose up --build***. Los archivos que **Docker** usa en tiempo de ejecución, como **volúmenes**, etc. se guardan en **/goinfre** o **/sgoinfre**.

EL JUEGO

Local 1 vs 1	Debe usarse el mismo **teclado**.
Torneo	La implementación es a **discreción del equipo**, así que tenemos un poco de libertad aquí. Pero debe indicar **quién juega contra quién** y el **orden** de los matches.
Registro	Cada participante debe indicar un **alias** antes iniciar el torneo y se **resetea** con un nuevo torneo. El registro puede ser modificado si se usa el módulo **Standard User Management**.
Matchmaking	Sistema de matchmaking que **empareja** a los participantes y **anuncia** la siguiente partida.
Fairness	Todos deben tener las mismas reglas, incluida la AI. Misma velocidad de la paleta y demás.
Frontend	Se debe usar el mismo **frontend** de la página web, o no, si se usa el módulo **Graphics**.

LA SEGURIDAD

Cifrado	Las contraseñas almacenadas deben estar **cifradas**.
Protección ante ataques	La web debe estar protegida frente a **SQL injections** y **XSS**.
TLS	Si se usa un **backend**, se debe usar **HTTPS** para todo.
Formularios	Deben **validarse** los formularios.
Credenciales	Por supuesto no se debe subir el **.env** con las credenciales y demás al **repositorio**.

MÓDULOS

Se necesitan 7 **Major Modules** para el proyecto base.

2 **Minor Modules** equivalen a 1 **Major Module**.

Para el bonus:

5 puntos por cada **Minor Module**.
10 puntos por cada **Major Module**.

¿Cuántos módulos son necesarios para el bonus?

MÓDULOS

WEB

MM - **BackEnd**	Debe usarse el framework **Django**.
mm - **FrontEnd**	Debe usar **Bootstrap toolkit**.
mm - **Database**	Debe usar **PostgreSQL**.
MM - **BlockChain**	Guardar las puntuaciones en **Ethereum**.

USER MANAGEMENT

MM - **Registro de usuarios**	Login, historial, perfiles, etc.
MM - **OAuth 2.0**	Login con **42**.

GAMEPLAY AND USER EXPERIENCE

MM - **Juego remoto**	Pues eso, partidas **online**.
MM - **Multiples jugadores**	Varios **jugadores** en una sola partida.
MM - **Chat**	Un **chat** funcional durante la partida.

AI-ALGO

MM - **Oponente AI**	Un rival controlado por **AI**.
mm - **Dashboard**	Dashboard con las **estadísticas**.

CYBERSECURITY

MM - **WAF/ModSecurity**	**Firewall** y **seguridad** de datos.
mm - **Cumplimiento GDPR**	**Privacidad** y **anonimización**.
MM - **Autenticación 2FA**	Verificación en dos pasos y **JWT**.

DEVOPS

MM - **Infraestructura**	Gestión de **logs** con **ELK**.
mm - **Sistema de Monitoreo**	Usando **Prometheus** y **Grafana**.
MM - **Microservicios**	**Backend** en **microservicios**.

GAME & GRAPHICS

MM - **Otro juego**	Añadir otro **juego** diferente.
mm - **Personalizaciones**	Añadir **personalizaciones** al juego.
MM - **Gráficos 3D**	Implementar gráficos **3D**.

SERVER-SIDE PONG

MM - **Server-Side Pong**	Pong en el **servidor** con **API**.
MM - **Pong por CLI**	Permitir jugar Pong en **CLI**.

ACCESSIBILITY

mm - **Dispositivos**	Asegurar funcionalidad en todos los dispositivos.
mm - **Compatibilidad**	Ampliar soporte a otro navegador web.
mm - **Multi-Idioma**	Implementar soporte para varios idiomas.
mm - **Accesibilidad visual**	Accesibilidad para usuarios con discapacidad visual.
mm - **Integración de SSR**	Implementar renderizado del lado del servidor.

1. Microservicio de Autenticación (2)

- **Función:** Gestiona la autenticación de usuarios, incluyendo OAuth 2.0 con 42, autenticación básica y autenticación de dos factores (2FA).
- **Responsabilidades:**
  - Autenticación básica (registro, inicio de sesión).
  - Integración OAuth 2.0 para login con 42.
  - 2FA para asegurar cuentas (mediante OTP o SMS, según preferencias).
  - Gestión de tokens de acceso y refresco, necesarios para acceder a los otros servicios.
- **Comunicación:**
  - API REST para las solicitudes de autenticación.
  - Base de datos de usuarios para almacenar datos de inicio de sesión y configuración de 2FA.

---

2. Microservicio de Perfil y Estadísticas de Usuario (1)

- **Función:** Almacena y gestiona el perfil del usuario, su historial, estadísticas de juego y preferencias.
- **Responsabilidades:**
  - Almacenar datos de perfil (nombre, imagen, etc.).
  - Registrar estadísticas (partidas ganadas, perdidas, puntuaciones).
  - Gestionar el historial de partidas y ajustes de configuración.
- **Comunicación:**
  - API REST para consultar y actualizar el perfil y estadísticas.
  - Base de datos para almacenar estos datos.
  - Microservicio de Juego para recibir actualizaciones de las estadísticas tras cada partida.

---

3. Microservicio de Gestión de Partidas (Matchmaking)

- **Función:** Conecta jugadores que desean jugar en tiempo real, creando sesiones de juego.
- **Responsabilidades:**
  - Crear y gestionar las sesiones de partida.
  - Asignar oponentes según disponibilidad y configuración del usuario.
  - Validar las reglas del juego y gestionar opciones (como partidas con o sin IA).
- **Comunicación:**
  - API REST para la creación de partidas.
  - WebSocket para comunicar el estado de la partida en tiempo real entre jugadores y el servidor.
  - Microservicio de Configuración del Juego para verificar los ajustes personalizados antes de cada partida.

---

4. Microservicio de Juego en Tiempo Real (1)

- **Función:** Ejecutar la lógica del juego y gestionar las partidas en tiempo real.
- **Responsabilidades:**
  - Ejecutar la lógica básica de colisión y movimiento de la pelota y las paletas.
  - Mantener sincronizados los estados del juego entre jugadores y transmitir actualizaciones en tiempo real.
  - Gestionar las partidas multijugador (sin IA) y en un solo dispositivo (jugadores en el mismo teclado).
- **Comunicación:**
  - WebSocket: para sincronizar en tiempo real el estado de juego y responder a eventos de los jugadores.
  - API REST: para compartir estadísticas y datos del juego finalizado.

5. Microservicio de IA (1)

- **Función:** Controlar el comportamiento de la IA en partidas jugador vs. máquina.
- **Responsabilidades:**
  - Ejecutar la lógica de movimiento de la IA, adaptando su nivel y estrategias según la dificultad seleccionada.
  - Proporcionar funciones de IA específicas como detección de power-ups, ajustes de velocidad, y otras tácticas.
  - Permitir niveles de dificultad configurables o específicos según la partida.
- **Comunicación:**
  - API REST: para que el microservicio de Juego en Tiempo Real consulte la acción de la IA en momentos clave del juego.
  - RPC o Mensajería Interna (opcional): optimiza la comunicación si la IA es compleja y necesita llamadas frecuentes o inmediatas.

---

6. Microservicio de Configuración del Juego (½)

- **Función:** Almacena y aplica configuraciones previas a las partidas (velocidad, IA, power-ups, etc.).
- **Responsabilidades:**
  - Registrar y almacenar configuraciones de la partida antes de su inicio.
  - Enviar los parámetros configurados al Microservicio de Juego en Tiempo Real.
- **Comunicación:**
  - API REST para recibir y almacenar las configuraciones enviadas por el cliente.
  - Comunicación con el Microservicio de Juego en Tiempo Real para aplicar configuraciones antes de iniciar la partida.

---

7. Microservicio de Smart Contract y Blockchain (1)

- **Función:** Interactúa con la blockchain para almacenar y recuperar las puntuaciones de los torneos.
- **Responsabilidades:**
  - **I**nteractuar con un contrato inteligente en la red de Sepolia para almacenar puntuaciones.
  - Consultar puntuaciones cuando sea necesario (por ejemplo, en el dashboard).
- **Comunicación:**
  - Web3.js o Ether.js (biblioteca de JavaScript) para realizar llamadas a la blockchain.
  - API REST para consultar y registrar puntuaciones de torneos.
  - Acceso restringido al Microservicio de Juego para registrar automáticamente la puntuación al finalizar una partida.

---

8. Microservicio de Chat en Vivo (1)

- **Función:** Facilita la comunicación en tiempo real entre jugadores.
- **Responsabilidades:**
  - Gestión de mensajes entre jugadores en la misma partida.
  - Registro de mensajes si es necesario para historial de chat.
- **Comunicación:**
  - WebSocket para mantener la comunicación en tiempo real.
  - Comunicación con Microservicio de Gestión de Partidas para validar las conexiones de chat por sesión.

9. Microservicio de Monitoreo (Prometheus & Grafana) (½)

- **Función:** Monitorear el rendimiento de los microservicios, estado de las partidas, errores, etc.
- **Responsabilidades:**
  - Recopilar métricas de uso y rendimiento de cada microservicio.
  - Mostrar información sobre la utilización y latencia de la infraestructura.
  - Generar alertas en caso de problemas (caída de un microservicio, aumento de errores).
- **Comunicación:**
  - Prometheus para recolectar métricas.
  - Grafana para visualizar y analizar datos.
  - Integración directa con cada microservicio mediante endpoints específicos.

---

10. Microservicio de Logs (ELK Stack) (1)

- **Función:** Maneja los logs de cada microservicio y centraliza la información.
- **Responsabilidades:**
  - Recopilar logs de errores, accesos y auditoría.
  - Indexar y hacer consultas de registros en ElasticSearch.
- **Comunicación:**
  - Filebeat o Logstash para recolectar logs de cada microservicio.
  - Kibana para visualizar y analizar la información de logs.

Comunicación entre Microservicios

- **WebSockets:** Principalmente usados para la comunicación en tiempo real (juego y chat).
- **API REST:** Para funcionalidades de autenticación, perfil, configuraciones, y consultas de puntuaciones.
- **Message Queue o gRPC (opcional):** Facilitaría el manejo de eventos entre servicios como juego y estadísticas.

Frontend (Bootstrap), Backend (Django), PostgreSQL, Server-Side Pong & Microservices (4)

TOTAL = 13 MM

Ejemplo de Estructura de Carpeta

- `ft_transcendence/`
  - `docker-compose.yml`
  - `.env`
  - `Backend/`
    - `authentication/` **# Microservicio de autenticación**
    - `profile/` **# Microservicio de perfil y estadísticas**
    - `matchmaking/` **# Microservicio de gestión de partidas**
    - `game/` **# Microservicio de juego en tiempo real**
    - `AI/` **# Microservicio de la IA**
    - `settings/` **# Microservicio de configuración del juego**
    - `blockchain/` **# Microservicio de contrato inteligente**
    - `chat/` **# Microservicio de chat en vivo**
    - `monitoring/` **# Microservicio de monitoreo**
    - `logs/` **# Microservicio de infraestructura y logs**
  - `Frontend/` **# Todo el frontend basado en SPA**