- PRÁCTICAS: DESARROLLO DE LA APLICACIÓN WEB CIVilizaTu
  - 1. DESCRIPCIÓN GENERAL
  - 2. OBJETIVOS
  - 3. REQUISITOS DE FUNCIONALIDAD
    - 3.1. Usuario y sistema de juego
      - Gestión de usuarios
      - Gestión del juego
    - 3.2. Sistema de juego
      - Mecánica de giro
      - Escenarios
      - Civilización
      - Economía y Recursos
    - 3.3. Elementos del mapa
      - Ciudades
      - Recursos naturales
      - Otros artículos
    - 3.4. Luchar
    - 3.5. Inteligencia artificial
    - 3.6. Modo truco
  - 4. REQUISITOS TÉCNICOS
    - 4.1. Interfaz
    - 4.2. Sistema de visualización de giros con IA
    - 4.3. Backend
    - 4.4. Componente de llamada LLM
    - 4.5. Comunicación entre componentes
  - 5. ARQUITECTURA DEL SISTEMA
    - 5.1. Ingredientes
    - 5.2. Comunicación entre componentes
  - 6. EXPANSIÓN
  - 7. RECURSOS CREADOS POR MÉTODOS VIRTUALES
    - 7.1. Recursos recomendados para crear contenido mediante IA
      - Declaraciones obligatorias para recursos generados por IA
      - Imágenes y gráficos
      - Sprites y gráficos
      - Música y efectos de sonido
      - Conversión y posprocesamiento
  - 8. ORGANIZACIÓN DEL TRABAJO

- 8.1. Distribución de roles y responsabilidades
  - 1. Rol: Desarrollador frontend 2.
  - Rol: Desarrollador backend 3. Rol:
  - Especialista en IA/ML 8.2. Plan
- de Desarrollo Recomendado
- 8.3. Coordinación de equipo

- 9. ENTREGABLES
- 10. CRITERIOS DE EVALUACIÓN
  - 10.1. Requisito esencial
  - 10.2. Evaluación grupal (70%)
  - 10.3. Evaluación individual (30%)
  - 10.4. Aspectos legales y éticos
- ANEXO 11: DESCRIPCIÓN DE LA COMUNICACIÓN Y LA ARQUITECTURA TÉCNICO
  - 11.1. Arquitectura general del sistema
  - 11.2. Comunicación frontend-backend
    - Comunicación con pantalla redonda de IA
    - Flujo de comunicación:
  - 11.3. Estructura de la base de datos de comunicación
    - entre el backend y MongoDB:
  - 11.4. Comunicación de servicios backend-IA
    - Flujo de comunicación:
- APÉNDICE 12: ESTRUCTURA DE DATOS DE COINCIDENCIA
  - 12.1 Formato de partida guardada
  - 12.2 Estructura de visualización redonda de IA
  - 12.3 Formato del sistema de trucos
- APÉNDICE 13: INSTRUCCIÓN DE INICIO DE LLM
- APÉNDICE 14: IMPLEMENTACIÓN DE LA VISTA DE GIRO VIRTUAL
  - 14.1. Estructura de datos para visualización
  - 14.2. Modos de visualización
  - 14.3. Controles de reproducción
  - 14.4. Elementos visuales
- APÉNDICE 15: IMPLEMENTACIÓN DEL JUEGO
  - 15.1. Sistema de rasgos de civilización
  - 15.2. Sistema de Recursos Naturales
  - 15.3. Mecánica urbana
  - 15.4. Sistema de Tecnología
  - 15.5. Sistema de combate

- 15.6. Condiciones de victoria
- 16. CONSEJOS DE IMPLEMENTACIÓN
  - 16.1. Estrategias de desarrollo
  - 16.2. Recomendaciones específicas para la
  - simplificación 16.3. Optimizaciones de frontend
  - 16.4. Optimizaciones de backend
  - 16.5. IA para mejorar el tiempo de pensamiento
  - 16.6. Prioridades de desarrollo para garantizar la jugabilidad

PRÁCTICA: CIVilizaTu WEB DESARROLLO DE APLICACIONES

1. DESCRIPCIÓN GENERAL

Esta práctica es un juego de estrategia por turnos inspirado en el clásico juego Civilization.
Se desarrollará una aplicación web. Este proyecto es puramente educativo y no pretende infringir ninguna marca registrada. La mecánica del juego está inspirada en el clásico juego Civilization, mientras que la interfaz gráfica estará diseñada utilizando gráficos 2D originales. Los juegos permiten a los usuarios registrarse, administrar juegos y usar el lenguaje GroQ
Les permitirá competir contra la inteligencia artificial impulsada por modelos.

El proyecto está diseñado para ser desarrollado por un equipo de 3 personas, cada una con un rol específico: desarrollador frontend, desarrollador backend y especialista en IA/ML. Esta estructura permitirá una distribución equilibrada del trabajo y facilitará el desarrollo paralelo de los diferentes componentes del sistema.

2. OBJETIVOS

- Aplicación web con frontend JavaScript/TypeScript y backend Python
  desarrollando una función
- Implementando una interfaz gráfica inspirada en los clásicos juegos de estrategia por turnos
- Crear una mecánica de juego por turnos original, aunque esté inspirada en juegos clásicos
- Integrando la inteligencia artificial de GroQ como adversario

- Aplicar conocimientos de desarrollo web, API y comunicación cliente-servidor.
- Uso de IA generativa para crear recursos (imágenes, mapas, sonidos) únicamente con fines educativos

3. REQUISITOS DE FUNCIONALIDAD

3.1. Usuario y sistema de juego

Gestión de usuarios

- Sistema de registro y autenticación de usuarios
- Almacenamiento seguro de credenciales
- Perfiles de usuario básicos
- Sesiones de juego de larga duración

Gestión del juego

- Guardar partidas manualmente en cualquier momento
- Sistema de guardado automático al final de cada ronda
- Cargando partidas guardadas
- Listado de coincidencias de usuarios disponibles
- Guardar coincidencias en formato JSON
- Selección de escenario al iniciar un nuevo juego

3.2. Sistema de juego

Mecánica de giro

- El juego se jugará por turnos, alternando entre el jugador humano y la IA.
- Cada ronda incluirá:
  - Fase de producción (elección de lo que las ciudades pueden producir)
  - Fase de investigación (elección de tecnologías a desarrollar)
  - Fase de movimiento (movimiento de la unidad a través del mapa)
  - Fase diplomática (negociación con otras civilizaciones)
  - Fase de construcción (desarrollo de ciudades e infraestructura)
  - Fase de fin de turno (producción de recursos, crecimiento de ciudadanos)

Escenarios

- Se debe implementar al menos un escenario jugable completo
- El escenario debe tener:
  - Mapa con diferentes territorios
  - Ciudades
  - Recursos
  - Elementos especiales (tecnologías, recursos naturales, etc.)
  - Niebla de guerra (las áreas inexploradas permanecen ocultas)

Civilización

- Cada jugador debe seleccionar una civilización.
- Cada civilización tiene las siguientes características:
  - Estadísticas propias (tecnología, cultura, diplomacia, ejército)
  - Ventajas especiales (por ejemplo, en ciencia o producción)
  - Unidades especiales
  - Tecnologías tempranas

Economía y Recursos

- Un sistema de recursos con al menos los siguientes recursos:
  - Alimentos (crecimiento urbano)
  - Producción (construcción)
  - Ciencia (tecnología)
  - Oro (mantenimiento y comercio)
- Cálculos de recursos al final del turno basados en:
  - Ciudades controladas
  - Mejoras del terreno
  - Rutas comerciales

3.3. Elementos del mapa

Ciudades

- Interfaz de gestión de la ciudad con:
  - Edificios que se pueden construir (al menos 5 tipos diferentes)
  - Creación de unidades (al menos 3 tipos de unidades)

- Información sobre recursos y producción
- Gestión del crecimiento

Recursos naturales

- Al menos 3 tipos diferentes de recursos naturales:
  - Mineral (para producción)
  - Ganado (para alimentación)
  - Recursos de lujo (para la felicidad)
- Cada recurso natural debe proporcionar una cantidad fija por turno.

Otros artículos

- Al menos 2 tipos de artículos adicionales:
  - Avances (avances o ventajas tecnológicas)
  - Campamentos bárbaros (amenazas)
  - Capitales naturales (puntos de renovación)
  - Recursos de lujo (cultura)

3.4. Luchar

- Implementar un sistema de combate por turnos Opción 1
- (Recomendado): Batallas de unidades en el mapa principal
  - Fuerza, defensa y vida de la unidad.
  - Cada tipo de unidad tiene sus propias ventajas.
  - Impacto estratégico de los diferentes tipos de territorio
- Opción 2 (Mínima): Simulación de combate
  - Cálculo automático basado en estadísticas
  - Mostrar resumen de resultados y número de terminaciones

3.5. Inteligencia artificial

- Implementación de un oponente de IA utilizando modelos GroQ
- La IA debe hacer lo siguiente:
  - Recuperar información del estado del juego en formato JSON
  - Tomar decisiones estratégicas (movimiento, combate, gestión)
  - Respetar las reglas del juego.

    - Proporcionar un desafío adecuado al jugador.
- Sistema de respaldo para varios modelos:
    - Configuración de múltiples puntos finales de GroQ
    - Cambio automático en errores 429 (límite de tokens)
    - Mantener el contexto al cambiar entre modelos

3.6. Modo truco

- Implementar un sistema de códigos de trampa para facilitar las pruebas
- El jugador puede activar la ventana de chat en el mapa presionando Ctrl+Tab.
- Ingresar los siguientes códigos tendrá efectos inmediatos:
    - build_all - Construye todos los edificios en la ciudad seleccionada
    - instant_defeat - Pierde el juego inmediatamente
    - instant_victory - Gana el juego instantáneamente tank_squadron
    - - La ciudad seleccionada recibe 5 unidades de tanques
    - advanced_technology - La civilización seleccionada tiene una tecnología avanzada recibe
    - level_up - La ciudad seleccionada sube un nivel.
    - max_resources - Se alcanza el nivel máximo de todos los recursos infinite_movement
    - - Movimiento ilimitado para todas las unidades
    - maximum_happiness - La ciudad seleccionada alcanza la máxima felicidad
    - mapa_agertu - Revela el mapa completo (elimina la niebla de guerra)
- Los códigos deben registrarse para facilitar la depuración.
    - Debería haber una opción para desactivar códigos en el entorno de producción.

4. REQUISITOS TÉCNICOS

¡IMPORTANTE!: Cada equipo necesita una combinación de tecnologías frontend y backend se le asignará uno específico. Es absolutamente obligatorio solo para el puesto asignado. utilizando tecnologías. Los proyectos que no cumplan con este requisito NO SERÁN EVALUADOS, independientemente de su funcionalidad o calidad.

4.1. Interfaz

- La tecnología frontend asignada será una de las siguientes:

- Reaccionar
- Angular
- Vista
- Esbelto

- Se recomienda utilizar TypeScript para mejorar la robustez del código Entorno Node.js
- para la gestión de dependencias y compilaciones Implementación
- de las siguientes vistas: Vista del mapa
  - principal Vista de la ciudad
  -
  - Vista de batalla (si se implementa la opción 1)
  - Interfaces de gestión (tecnologías, recursos, etc.)
- Interfaz gráfica original inspirada en los juegos de estrategia por turnos: Estilo visual consistente
  - (paneles, iconos, etc.)
  - Gráficos 2D originales o generados por IA (con el etiquetado adecuado)
  - Animaciones básicas

4.2. Sistema de visualización de giros con IA

- Modo de visualización para el turno de IA con dos opciones implementadas:
  - Opción 1 – Cambio de vista: Cuando llega el turno de la IA, la vista del jugador cambia temporalmente.

    Se reemplaza por una vista de la IA, acciones en tiempo real.

    demostración. Cuando termina el turno de la IA, vuelve a la vista del jugador.
  - Opción 2 - Pantalla dividida: Durante el turno de la IA, la pantalla se divide en dos partes:
    - Izquierda: Vista del jugador (estática)
    - Derecha: Vista de IA (dinámica) que muestra movimiento y acciones
- Visualización clara y detallada de las acciones de IA:
  - Movimientos de unidades con indicadores visuales de ruta
  - Batallas iniciadas por IA y resultados visibles
  - Construcción y reclutamiento en las ciudades
  - Recopilación de recursos y objetos
- Controla la velocidad de visualización de las acciones de la IA (normal, rápida, repentina)
- Opción para activar o desactivar esta visualización de precisión

4.3. Backend

- La tecnología backend asignada será una de las siguientes:
  - Matraz

- API rápida

- Implementación de API REST para:
  - Gestión del estado del juego
  - Autenticación y gestión de usuarios
  - Guardar y cargar juegos
  - Comunicación con los modelos GroQ
- Sistema de autenticación basado en bibliotecas establecidas:
  - Para tokens de autenticación JWT
  - OAuth2 a través de extensiones FastAPI/Flask
  - como Auth0, Firebase Authentication, etc. la posibilidad de utilizar servicios como

4.4. Componente de llamada LLM

- Módulo específico para la interacción con los modelos GroQ
- Funcionalidades:
  - Preparación de indicaciones con descripción del juego.
  - Envío del estado actual en formato JSON
  - Gestión de respuestas
  - Gestión de errores y cambio a modelos alternativos
  - Control del número de tokens utilizados

4.5. Comunicación entre componentes

- Frontend → Backend:
  - Comunicación mediante solicitudes HTTP REST
  - Formato JSON para el intercambio de datos
  - Puntos finales para todas las acciones del juego (movimiento, combate, gestión)
- Backend → LLM:
  - Llamadas a la API de GroQ
  - Indicación inicial con descripción del juego y reglas.
  - Actualización del estado visible para la IA (respetando la Niebla de Guerra)
  - Procesar respuestas para traducirlas en acciones de juego
- LLM → Backend:
  - Recibir respuestas en formato JSON

- Validación de las acciones propuestas
- Implementación de acciones válidas

- Backend → Frontend:
  - Respuestas HTTP que contienen resultados de acciones
  - Notificaciones de eventos (opcionalmente WebSockets)
  - Actualizaciones del estado del juego

5. ARQUITECTURA DEL SISTEMA

5.1. Ingredientes

El sistema constará de cuatro componentes principales, cada uno en su propio contenedor Docker:

1. Frontend: Aplicación web basada en React/Angular/Vue/Svelte
2. Backend: API REST en Flask/FastAPI
3. Base de datos MongoDB: para almacenar usuarios, juegos y escenarios 4. Servicio de IA: componente de manejo de llamadas para modelos GroQ

5.2. Comunicación entre componentes

- Frontend → Backend:
  - Comunicación mediante solicitudes HTTP REST
  - Formato JSON para el intercambio de datos
  - Puntos finales para todas las acciones del juego (movimiento, combate, gestión)

- Backend → MongoDB:
  - Almacenamiento de datos de usuario
  - Persistencia de juegos guardados y guardados automáticos
  - Gestión de escenarios disponibles

- Backend → Servicio de IA:
  - Solicitudes de decisiones de IA
  - Gestión de cambios entre diferentes modelos
  - Actualización del contexto del juego

- Servicio de IA → GroQ:
  - Llamadas a la API externa de GroQ
  - Gestión de limitaciones de tokens y API
  - Procesamiento de respuestas

6. EXPANSIÓN

- Contenedor Docker completo:
  - Dockerfile para frontend
  - Dockerfile para backend
  - Docker Compose para orquestación
- Requisitos de expansión:
  - Configuración mediante variables de entorno
  - Persistencia de datos (volúmenes Docker)
  - Documentación clara para el proceso de implementación
- Instrucciones de ejecución:
  - Clonación del
  - repositorio Configuración de variables de
  - entorno (.env) Ejecución de docker compose up --build
  - Acceso a la aplicación a través del navegador

7. RECURSOS CREADOS POR MÉTODOS VIRTUALES

- Usando IA generativa para:
  - Imágenes de unidades y ciudades
  - Parcelas para el mapa
  - Iconos de recursos y edificios
  - Efectos de sonido básicos (opcional)
  - Música de fondo (opcional)
- Documentación del proceso de creación:

- Indicaciones utilizadas
- Herramientas utilizadas
- Posprocesamiento aplicado

7.1. Recursos recomendados para crear contenido mediante IA

Para facilitar la creación de recursos sin incurrir en costes excesivos, se recomiendan las siguientes herramientas. Es importante que todos los recursos creados a través de IA Llevar una declaración clara, creada por IA solo con fines educativos.
indicando que lo son, tanto en los metadatos como en la documentación del proyecto.

Declaraciones obligatorias para recursos generados por IA

- Añadiendo aquí la frase "Creado con IA para el proyecto educativo Civilization":
  - En los metadatos de cada archivo cuando sea posible
  - En la esquina inferior derecha de las imágenes (pequeñas pero legibles)
  - Con una lista de todos los recursos creados en la documentación técnica
  - Cuando se utilicen estos recursos, deberán comentarse en el código.
- Para archivos de audio, una breve nota en el archivo README y la documentación
- Mantener una carpeta separada (por ejemplo, /assets/ai-generated/) para todos los activos generados por IA

Imágenes y gráficos

- Difusión estable (implementación local o a través de Google Colab gratuito)
- Leonardo.ai (plan gratuito, número limitado de creaciones)
- Midjourney (suscripción mensual básica, compartida en el grupo)
- DALL-E mini / Craiyon (gratis, calidad inferior)
- Bing Image Creator (gratis con límite diario)
- RunwayML (versión gratuita con limitaciones)

Sprites y gráficos

- PixelMe (para crear sprites pixelados)
- Pixelicious (para transformar imágenes en estilo pixel art)
- Pixelorama (editor gratuito para refinar las imágenes creadas)

Música y efectos de sonido

- AIVA (con algunas limitaciones de la versión gratuita)
- Mubert (plan gratuito para efectos de sonido básicos)
- Soundraw (periodo de prueba)
- Riffusion (creación musical basada en Stable Diffusion, código abierto)
- FreeSound (biblioteca de efectos de sonido con licencia libre)

Conversión y posprocesamiento

- GIMP (editor de imágenes gratuito)
- Inkscape (editor vectorial gratuito de iconos)
- Audacity (programa gratuito de edición de audio)

Todas estas herramientas ofrecen opciones gratuitas o de bajo costo que son suficientes para cumplir el objetivo del proyecto. Se recomienda que los equipos planifiquen y enfoquen las necesidades de recursos en períodos específicos para aprovechar al máximo los límites gratuitos de las plataformas.

8. ORGANIZACIÓN DEL TRABAJO

8.1. Distribución de roles y responsabilidades

Como equipo de 3 personas, se recomienda la siguiente distribución de roles:

1. Rol: Desarrollador frontend

- Responsabilidades:
  - Implementación completa de la interfaz de usuario
  - Diseño visual basado en Civilization
  - Integración con la API backend
  - Implementación de diferentes vistas (mapa, ciudad, combate)
  - Interfaz de gestión de usuarios y juegos
  - Creación y adaptación de recursos visuales mediante IA
  - Implementación de un sistema de visualización de giros con IA
  - Desarrollo de la interfaz del modo truco

2. Rol: Desarrollador backend

- Responsabilidades:
  - Diseño e implementación de API REST
  - Lógica del juego (reglas, turnos, combate)
  - Sistema de gestión de recursos y estado del juego
  - Autenticación y gestión de usuarios
  - Integración con MongoDB
  - Configuración de implementación con Docker
  - Implementación de la lógica del modo truco

3. Rol: Especialista en IA/ML

- Responsabilidades:
  - Integración con los modelos GroQ
  - Diseño de mensajes y formatos de datos para IA
  - Gestión de múltiples modelos y sistemas de reemplazo
  - Creación de contenidos (mapas, unidades) mediante IA
  - Estructura de datos para guardar/cargar juegos
  - Documentación del sistema y manual del usuario
  - Optimización del comportamiento de la IA para simular estrategias de juego

8.2. Plan de Desarrollo Recomendado

Semana	1. Rol (Interfaz)	2. Rol (Backend)	3. Rol (IA/ML)
1	Diseño de interfaz de usuario inicial y autenticación	API y modelo de datos diseño	Investigación de GroQ
2	Implementación de registro/inicio de sesión	MongoDB y autenticación configuración	Coincidencias JSON diseño de formato
3	Implementación de la vista de mapa	Lógica básica del juego	Diseño de avisos
4	Vista de la ciudad	Gestión del juego (guardar/cargar)	Con GroQ integración

Semana	1. Rol (Interfaz)	2. Rol (Backend)	3. Rol (IA/ML)
5	La escena de batalla y sistema de trucos	Con servicio de IA integración	Sistema multimodelo
6	Visualización de la ronda de IA para la implementación de Docker		Documentación

8.3. Coordinación de equipo

Establecer mecanismos de coordinación entre los miembros del equipo es esencial:

- Reuniones semanales de
- seguimiento Uso de herramientas de gestión de proyectos (Trello, Jira, etc.)
- Control de versiones a través de Git e integración mediante solicitudes de extracción
- Documentación compartida de decisiones técnicas
- Códigos y convenciones estándar acordados inicialmente

9. ENTREGABLES

1. Código fuente completo en el repositorio Git
2. Documentación técnica:
    - Arquitectura del sistema
    - Puntos finales de API
    - Modelo de datos
    - Integración con GroQ
    - Estructura del juego guardado
    - Un inventario detallado de recursos creados a través de IA con:
        - Herramienta utilizada
        - Indicación aplicada
        - Fecha de creación
        - Uso en el proyecto
3. Declaración de originalidad y uso educativo (que el proyecto es únicamente educativo) y confirmación no comercial)
4. Manual del usuario
5. Presentación del proyecto 6.

Archivos Docker para despliegue (4 contenedores)

7. El informe del proyecto contiene lo siguiente:

- Contribución de cada miembro
- Desafíos y soluciones
- Decisiones de diseño e implementación

10. CRITERIOS DE EVALUACIÓN

10.1. Requisito esencial

¡ATENCIÓN!: Para aprobar la práctica, la APLICACIÓN DEBE FUNCIONAR CORRECTAMENTE y DEBEN DESARROLLARSE CON LAS TECNOLOGÍAS ASIGNADAS. Estos son requisitos absolutamente esenciales y no negociables. Si la aplicación no funciona en el entorno de pruebas utilizando el procedimiento de despliegue especificado (docker compose up) o se han utilizado tecnologías distintas a las asignadas, la pasantía será suspendida automáticamente independientemente de la calidad del código o la documentación.

Por "funcionar correctamente" se entiende:

- La aplicación se inicia sin errores.
- Es posible jugar al menos un escenario completo
- Mecánicas de combate, gestión de recursos y turnos según especificaciones ellos trabajan
- La integración con GroQ responde adecuadamente
- No hay errores críticos que dificulten la experiencia de juego.

10.2. Evaluación grupal (70%)

Criterio	Porcentaje
Funcionalidad completa 25%	
Calidad del código	15%
Integración con GroQ 10%	
Interfaz gráfica	10%
Extensión directa	5%
Documentación	5%

10.3. Evaluación individual (30%)

Criterio	Porcentaje
Contribución de código (confirmaciones)	10%
Calidad del trabajo individual	10%
Cumplimiento de responsabilidades asignadas	5%
Participación en la presentación	5%

Se realizará una evaluación por pares dentro del grupo para validar las contribuciones individuales.

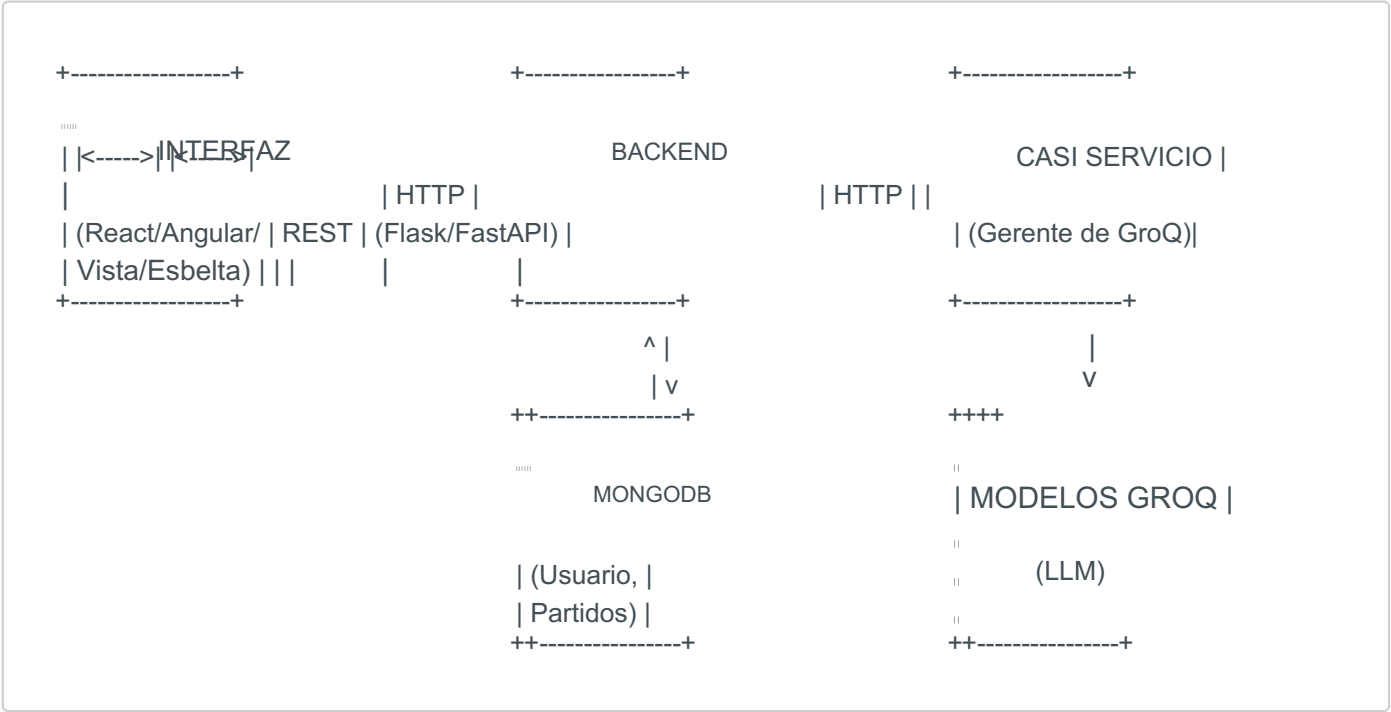
10.4. Aspectos legales y éticos

- Uso educativo: Este proyecto es sólo para fines educativos, sin intenciones comerciales.
- Originalidad del contenido:
  - Se prohíbe el uso de recursos gráficos, de audio o de código de juegos comerciales.
  - Todo el contenido debe ser original, de dominio público o generado por IA.
- Contenido generado por IA:
  - Todo el contenido generado a través de IA debe estar claramente etiquetado.
  - Se debe mantener un registro detallado de las indicaciones utilizadas.
  - Las imágenes generadas por IA deben tener una marca de agua o una atribución visual discreta ellos tienen.
- Atribución: El uso de
  - bibliotecas de terceros debe reconocerse adecuadamente.
  - El uso de recursos de dominio público debe documentarse en su con licencias.
- Transparencia:
  - La documentación debe tener una sección específica sobre el origen de todos los recursos. Él lo necesita.
  - El juego debería estar claramente establecido en los juegos clásicos de estrategia por turnos. que está "inspirada", sin mencionar las marcas registradas.

El incumplimiento de estos aspectos podrá afectar la evaluación del proyecto.

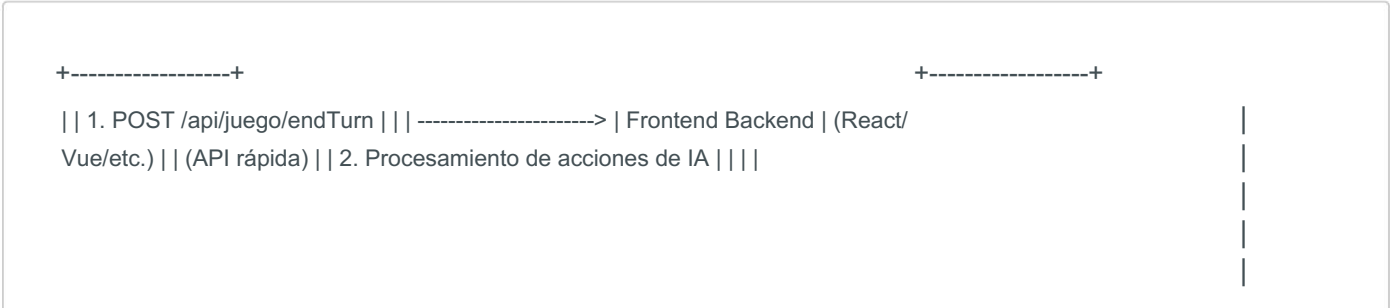
ANEXO 11: COMUNICACIÓN Y DESCRIPCIÓN DE LA ARQUITECTURA TÉCNICO

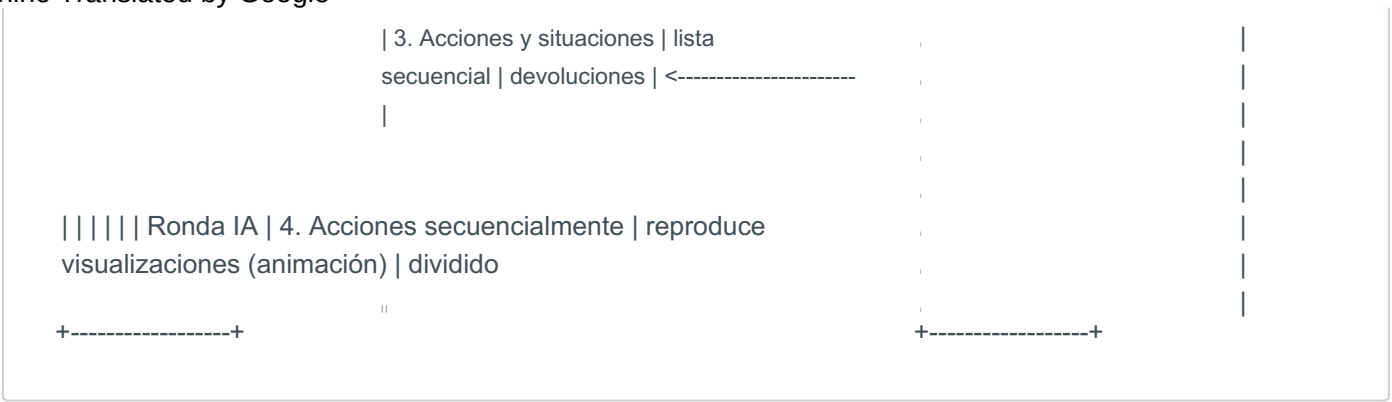
11.1. Arquitectura general del sistema



11.2. Comunicación frontend-backend

Comunicación con pantalla redonda de IA





Flujo de comunicación:

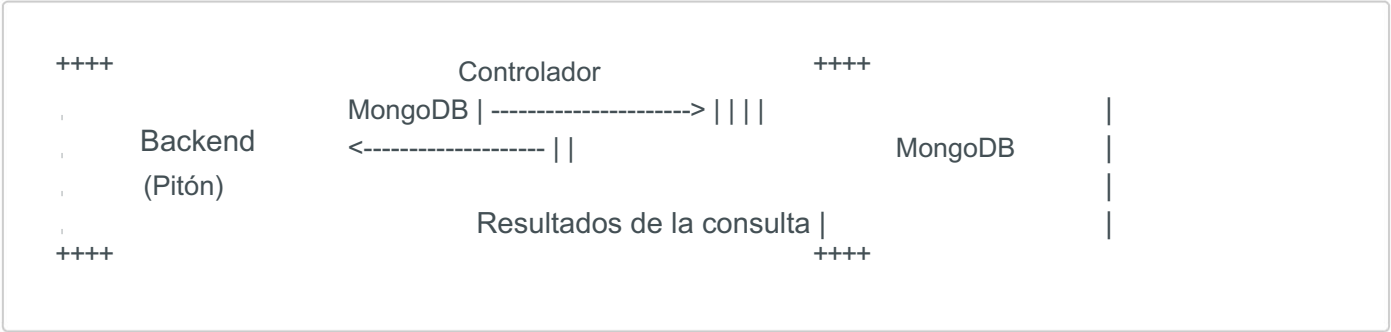
1. Autenticación y gestión de usuarios:
    - Registro de nuevo usuario (POST /api/auth/register)
    - Inicio de sesión de usuario (POST /api/auth/login)
    - Obtener el perfil del usuario (GET /api/auth/profile)
    - Actualizando perfil (PUT /api/auth/profile)
2. Gestión de partidos:
    - Listado de juegos guardados (GET /api/games)
    - Creando un nuevo juego (POST /api/games)
    - Guardar el juego actual (POST /api/games/{gameId}/save)
    - Cargando juego guardado (GET /api/games/{gameId})
    - Listado de escenarios disponibles (GET /api/scenarios)
3. En el turno del jugador:
    - El frontend envía las acciones del jugador (POST /api/juegos/{gameId}/acción)
    - El backend valida y procesa las acciones.
    - El backend devuelve el resultado y el nuevo estado.
4. Al final del turno del jugador:
    - El frontend envía una solicitud de fin de turno (POST) /api/juegos/{gameId}/finTurno)
    - El backend procesa el final del turno (cálculo de recursos, etc.)
    - El backend guarda el estado actual (guardado automático)
    - El backend inicia la ronda de IA
5. Sistema de trucos:

- El frontend detecta la combinación Ctrl+Tab y muestra la interfaz de chat.

tiene

- El frontend envía el código de trucos (POST /api/games/{gameId}/cheat)
- El backend valida el código y aplica los efectos apropiados.
- El backend devuelve el nuevo estado del juego con los cambios aplicados.

11.3. Comunicación entre el backend y MongoDB



Estructura de la base de datos:

1. Recopilación de usuarios:

```
{
    "_id": "Id. del objeto",
    "nombre de usuario": "cadena",
    "correo electrónico": "cadena",
    "password_hash": "cadena",
    "created_at": "fecha",
    "last_login": "fecha"
}
```

2. Colección de juegos:

```
{
    "_id": "Id. del objeto",
    "user_id": "Id. del objeto",
    "nombre": "cadena",
    "scenario_id": "cadena",
    "created_at": "fecha",
    "last_saved": "fecha",
    "is_autosave": "booleano",
    "trucos_usados": ["cadena"],
    "estado_del_juego": {
        "turn": "número",
        "jugador": {
            "ciudades": [...],
            "unidades": [...],
```

```
            "tecnologías": [...],
            "recursos": {...}
        },
        "Vaya": {
            "ciudades": [...],
            "unidades": [...],
            "tecnologías": [...],
            "recursos": {...}
        },
        "mapa": {
            "tamaño": {"ancho": "número", "alto": "número"},
            "azulejos": [...],
            "niebla de guerra": [...]
        },
        "current_player": "cadena"
    }
}
```

3. Colección de escenarios:

```
{
    "_id": "Id. del objeto",
    "nombre": "cadena",
    "descripción": "cadena",
    "dificultad": "cuerda",
    "map_size": {"ancho": "número", "alto": "número"},
    "estado_inicial": {...}
}
```

11.4. Comunicación de servicios backend-IA

Flujo de comunicación:

1. Inicialización de IA:
    - El backend envía el mensaje inicial con la descripción del juego y con reglas
    - El backend almacena el contexto de la conversación.

2. Ronda de IA:

   - El backend prepara el estado visible para la IA (JSON)
   - El backend envía el estado al modelo principal de GroQ. Si hay un error 429
   - (límite de token), cambia al modelo de respaldo.
   - GroQ devuelve decisiones de IA en formato JSON
   - El backend valida y ejecuta acciones de IA.
   - El backend actualiza el estado del juego.
   - El backend envía una secuencia de acciones al frontend para su visualización.

3. Gestión del contexto:

   - El backend mantiene un historial resumido de acciones importantes
   - El contexto está restringido para evitar exceder los límites de tokens.
   - Las estrategias aprendidas se guardan entre rondas.

4. Visualización de acciones de IA:

   - El backend crea una lista ordenada de acciones realizadas.
   - Cada acción incluye un estado previo y posterior.
   - La secuencia completa se envía al frontend para su animación.

APÉNDICE 12: ESTRUCTURA DE DATOS DE COINCIDENCIA

A continuación se muestra un ejemplo del formato JSON para almacenar coincidencias:

12.1 Formato de partida guardada


```
{
  "game_id": "65f1a2b3c4d5e6f7a8b9c0d1", "name":
  "Mi partida contra la IA", "scenario_id":
  "basic_map_1", "created_at":
  "2025-04-01T18:30:22.123Z", "last_saved":
  "2025-04-01T19:45:33.456Z", "turn": 12, "current_player":
  "jugador",
  "cheats_used": ["map_appear",
  "level_up"], "player": { "resources": { "food": 35,
```

```
            "producción": 28, "ciencia":
            15, "oro": 420 },
            "ciudades": [ {


                "id": "ciudad1",
                "nombre": "Bilbo",
                "posición": {"x": 15, "y": 22}, "población": 5,
                "edificios": [ {"id":
                "granero", "tipo":
                    "edificio_de_comida"}, {"id": "biblioteca", "tipo": "edificio_científico"},
                    {"id": "cuartel", "tipo": "edificio_militar"} ], "producción": { "elemento_actual":
                    "guerrero", "turnos_restantes": 3



                }

        } ], "unidades":
            [ {
                "id": "unidad1", "tipo":
                "colono", "posición": {"x": 12,
                "y": 18}, "puntos_de_movimiento": 2,
                "puntos_de_movimiento_izquierda":
                1 }, {


                "id": "unidad2", "tipo":
                "guerrero", "posición": {"x":
                14, "y": 20}, "puntos_de_movimiento": 2,
                "puntos_de_movimiento_izquierda":
                0, "fuerza": 5, "salud": 100



        } ], "tecnologías": [ {"id":
            "agricultura", "completado": verdadero}, {"id": "cerámica",
            "completado": verdadero}, {"id": "cría_de_animales",
            "en_progreso": verdadero, "turnos_restantes": 4}


] }, "ai":
    { "recursos":
        { "alimentos": 40,
        "producción": 22, "ciencia":
        10, "oro": 380 },
        "ciudades": [ {


                "id": "ai_city1", "nombre":
                "Roma", "posición": {"x":
                42, "y": 35},
```

```
                "visible": false


        } ], "unidades":
            [ {
                "id": "ai_unit1", "tipo":
                "desconocido", "posición":
                {"x": 45, "y": 38}, "visible": falso



        } ], "tecnologías": [] }, "mapa":

    { "tamaño":
        {"ancho": 72, "alto": 72}, "explorado": [ [0, 0, 0, 0, 1, 1, 1,
        0 , 0 ], [0, 0, 1, 1, 1,
             1, 1, 1 , 1, 0], [0, 0, 1, 1, 1, 1, 0] 1, 0], [1, 1,
             1, 1, 1, 1, 1, 0, 0] ], "objetos_visibles": [




            {
                "tipo": "recurso",
                "tipo_de_recurso": "hierro", "posición":
                {"x": 14, "y": 25}, "mejorado": verdadero }, {



                "tipo": "recurso",
                "tipo_de_recurso": "ganado", "posición":
                {"x": 18, "y": 19}, "mejorado": falso }, {



                "tipo": "campamento_bárbaro",
                "posición": {"x": 22, "y": 28}
            }
        ]
    }
}
```


12.2 Estructura de visualización redonda de IA

Para facilitar la visualización del turno de la IA, se utilizará el siguiente formato JSON:


```
{
    "ai_turn_id": "65f1a2b3c4d5e6f7a8b9c0d2", "game_id":
    "65f1a2b3c4d5e6f7a8b9c0d1", "turn_number": 12, "actions":
    [ {
```

```
"id_acción": 1, "tipo":
"unidad_movimiento",
"Id_unidad": "unidad_ai1",
"ruta": [ {"x":
    42, "y": 35}, {"x": 43, "y":
    35}, {"x": 44, "y": 36} ],
    "estado_anterior": {


    "recursos": { /* breve estado anterior */ }, "unidades": { /* breve
    estado anterior */ }, "visible_map": { /* breve estado anterior
    */ } }, "estado_después": {


    "recursos": { /* estado breve a continuación */ }, "unidades": { /*
    estado breve a continuación */ }, "visible_map": { /* estado
    breve a continuación */ } }, "timestamp": "2025-04-01T19:40:15.123Z" },
{



"action_id": 2, "type":
"buildStructure", "cityId": "ai_city1",
"structureType": "library",
"state_before": { /* corto antes del
estado */ }, "state_after": { /* corto después del estado */ }, "timestamp":
"2025-04-01T19:40:18.456Z" }, {



"action_id": 3, "type":
"combat", "attacker":
{ "unitId":
    "ai_unit2", "type": "warrior",
    "strength": 5 }, "defender":
    { "type":

"barbarian",
    "position": {"x": 44, "y": 36},
    "strength": 4 }, "result": { "winner":
    "attacker",

"sasualties":
    { "attacker": {"damage": 20},
    "defender": {"killed":
        true} }, "rewards": { "gold": 50









} }, "state_before": { /* corto antes del estado */ }, "state_after": { /*
corto después del estado */ }, "timestamp": "2025-04-01T19:40:25.789Z" },
{
```

"action_id": 4, "type":
"endTurn", "state_before":
{ /* corto antes del estado */ }, "state_after": { /* corto después del
estado */ }, "timestamp": "2025-04-01T19:40:30.123Z"

}], "reasoning": "Estoy trasladando mi unidad principal al este en busca de mineral de hierro, al mismo tiempo que desarrollo mi ciudad construyendo una biblioteca para acelerar la investigación de nuevas tecnologías". }

12.3 Formato del sistema de trucos

Para gestionar los códigos de trucos, se utilizará el siguiente formato de solicitud y respuesta:

Pedido:

{
"id_juego": "65f1a2b3c4d5e6f7a8b9c0d1",
"código_truco": "subir_nivel",
"objetivo":
{ "tipo": "ciudad",
"id": "ciudad1"
}
}

Respuesta:

{
"success": true,
"message": "La ciudad ha subido un nivel",
"affected_entity": { "type":
"city", "id": "city1",
"changes":
{ "population":
{"before": 5, "after": 6}, "growth": {"before": 3, "after": 4}

} }, "estado_del_juego": {
/* Estado del juego actualizado */ }

}

APÉNDICE 13: INSTRUCCIÓN DE INICIO DE LLM

A continuación se muestra un ejemplo del mensaje inicial que se enviará a los modelos de GroQ para describir el juego. se muestra uno:

Aquí está el estado actual del juego:

<estado del juego>
{{ESTADO_DEL_JUEGO}}
</estado_del_juego>

Eres un agente de IA que juega al juego de estrategia por turnos Civilization. Tu objetivo es expandir tu imperio, conquistar ciudades, reunir recursos y derrotar a tu oponente. Recibirás el estado del juego y deberás decidir tus acciones para este turno.

Tu tarea es analizar la situación del juego, formular una estrategia y determinar acciones para tu turno actual. Siga estos pasos:

1. Analiza la situación del juego, teniendo en cuenta: - La ubicación, el estado y los edificios de tus ciudades - La ubicación y las capacidades de tus unidades - Tus recursos e ingresos disponibles - Las áreas del mapa exploradas - Las ubicaciones y fortalezas conocidas del enemigo - Las oportunidades cercanas (recursos, campamentos bárbaros, tecnologías)

   - Niebla de guerra (áreas del mapa que no has explorado)

2. Formular una estrategia basada en estas prioridades: - Exploración para encontrar recursos y ciudades - Asegurar fuentes de ingresos - Desarrollo de ciudades para reclutar unidades más poderosas - Avance tecnológico para obtener una ventaja - Equilibrar la economía y la fuerza militar

3. Crea un conjunto de acciones para esta ronda. Podrás realizar diversas acciones hasta que te quedes sin puntos de movimiento. Los tipos de acciones posibles son: - moveUnit: mover una unidad a una nueva ubicación - buildStructure: construir un edificio en la ciudad - trainUnit: entrenar una nueva unidad en la ciudad - improveResource: mejorar un recurso - attackEnemy: iniciar una batalla con un enemigo - researchTechnology: investigar una nueva tecnología - foundCity: crear una nueva ciudad

Antes de dar su respuesta final, escriba su proceso de pensamiento y consideraciones estratégicas entre las etiquetas <strategic_planning>. En esta sección:

1. Resume el estado actual del juego, la ubicación de las ciudades, los recursos y

Incluyendo información conocida sobre el enemigo.
2. Enumere las oportunidades y amenazas potenciales.
3. Priorizar objetivos en función de la situación actual.
4. Explique la estrategia a corto plazo (esta ronda) y a largo plazo (próximas rondas).

Está bien que esta sección sea bastante larga, ya que una estrategia profunda es esencial para el éxito en el juego.

Su respuesta final debe tener el siguiente formato JSON:

```
{
    "acciones": [ {

            "tipo": "tipo de acción", "detalles":
            {
                // Detalles de la acción


        } }, // ... más acciones ... {

            "tipo": "fin de giro"


    } ], "reasoning": "Una explicación detallada de tu estrategia y planes para las próximas rondas", "analysis": "Un breve análisis de la
    situación del juego y la posición de tu oponente" }
```

A continuación se muestra un ejemplo del formato de acción:

```
{
    "acciones": [ {

            "tipo": "unidadDeMovimiento",
            "detalles":
                { "IDDeUnidad": "unidad1",
                "destino": { "x": 5, "y": 3}


        } }, {
            "tipo": "estructuraconstruida", "detalles":
            { "idciudad":
                "ciudad1", "tipoestructura":
                "granero"


        } }, {
            "tipo": "unidad de tren",
            "detalles": { "id de
                ciudad": "ciudad1", "tipo de
                unidad": "guerrero", "cantidad": 1



        } }, {
```

```
            "tipo": "fin de giro"


    } ], "reasoning": "Una explicación detallada de la estrategia y los planes para las próximas rondas", "analysis":
"Un breve
    análisis de la situación del juego y la posición del oponente" }
```

Recordar:
- Piensa estratégicamente y planifica a largo plazo
- Gestiona tus recursos de forma eficiente
- Adapta tu estrategia a la situación del juego, incluidas las zonas ocultas por la niebla de guerra.

- Equilibrar el desarrollo económico y la fuerza militar
- Aprovecha tus fortalezas y las debilidades de tu oponente.
- Siempre finaliza tu turno con la acción "endTurn"
- Proporcione un razonamiento detallado de sus decisiones.
- Respetar las reglas y la mecánica del juego.

Ahora, basándose en la situación de juego dada, analiza la situación, formula tu estrategia y crea tus acciones, razonamiento y análisis para este turno.

APÉNDICE 14: IMPLEMENTACIÓN DE LA VISTA DE GIRO VIRTUAL

Para implementar correctamente la visualización de giros con IA, se deben seguir las siguientes pautas: son:

14.1. Estructura de datos para visualización

El backend debe proporcionar al frontend una estructura de datos para las acciones realizadas por la IA. Para animar secuencialmente:

```
{
    "ai_turn_summary":
        { "total_acciones": 8,
        "enfoque_principal": "expansión",
        "recursos_obtenidos": {"comida": 12, "producción": 5, "ciencia": 3}, "territorios_explorados": 12,
        "resultados_de_combate": [ {"ubicación":
        {"x": 25, "y": 42}, "resultado":
            "victoria", "recompensa": "oro"} ] },
```

```
    "secuencia_de_acciones_ai": [
      {
        "id": 1,
        "tipo_acción": "movimiento",
        "entidad": {"id": "unidad_ai_1", "nombre": "Guerrero", "tipo": "unidad"}, "ruta": [{"x": 10, "y":
        15}, {"x": 11, "y": 16}, {"x": 12, "y": 16}], "puntos_movimiento": {"inicial": 2, "restante": 0},
        "instantánea_estado_antes": {...}, "instantánea_estado_después":
        {...} },


      ...
    ]
}
```


14.2. Modos de visualización

El sistema debe ofrecer al menos dos modos de visualización para el turno de la IA:

1. Modo de pantalla completa:
    - Durante el turno de la IA, la vista del jugador es reemplazada temporalmente por la vista de la IA.
    - Se muestra una animación secuencial de todas las acciones.
    - Se incluye una barra lateral con información contextual (recursos, ciudades).
    - Cuando termina, vuelve automáticamente a la vista del jugador.
2. Modo de pantalla dividida:
    - La pantalla está dividida horizontalmente en dos partes.
    - Izquierda: vista estática del jugador al final de su turno.
    - Derecha: Vista dinámica que muestra la secuencia de acciones de IA.

14.3. Controles de reproducción

Para mejorar la experiencia de visualización de turnos de IA, se deben implementar los siguientes controles para el jugador:

- Pausar/reanudar la reproducción
- Paso adelante (siguiente acción)
- Dar un paso atrás (acción anterior)
- Ajustar la velocidad de reproducción (normal, rápida, aleatoria)
- Saltando directamente al resultado final

14.4. Elementos visuales

La visualización debe incluir los siguientes elementos gráficos para facilitar el seguimiento de las acciones:

- Indicadores de ruta para movimientos de unidades
- Etiquetas de valores numéricos para cambios de recursos
- Animaciones para combate y construcción.
- Indicadores de cambios de estado (antes/después)
- Resaltando las áreas afectadas en el mapa

APÉNDICE 15: IMPLEMENTACIÓN DEL JUEGO

Para capturar la esencia de los juegos de estrategia por turnos como Civilization, es importante implementar las siguientes mecánicas de juego:

15.1. Sistema de rasgos de civilización

- Cada civilización tiene sus propias características: Cada civilización tiene sus propias ventajas únicas tiene
- Cada civilización tiene su propia unidad única: algo que otras civilizaciones no tienen. unidades exclusivas
- Características del líder: Características históricas del líder al frente de cada civilización reflejar

15.2. Sistema de Recursos Naturales

- Recursos naturales: Están distribuidos por todo el mapa y se pueden mejorar. Mejoras: Una forma específica de mejorar cada recurso
- natural (agricultura, minería, etc.)

- Contribución de recursos: cada recurso realiza contribuciones específicas (alimentos, producción, oro)

15.3. Mecánica urbana

- Crecimiento urbano: Las ciudades tienen un crecimiento natural basado en los alimentos.
- Árbol de construcción: algunos edificios requieren otros antes de poder construirse
- Especialización: Cada ciudad puede especializarse en ciertos tipos de recursos o unidades. poder

15.4. Sistema de tecnología

- Árbol tecnológico: Las tecnologías progresan en un orden lógico
- Diferentes épocas: Era Antigua, Era Clásica, Edad Media, Era Industrial y Era Moderna moderno
- Ventajas tecnológicas: Las nuevas tecnologías proporcionan ventajas militares o económicas. ellos dan

15.5. Sistema de combate

- Tipos de unidades: diferentes en fuerza, defensa y puntos de movimiento.
- Efecto del terreno: las montañas, los ríos y los bosques afectan el combate.
- Sistema de ascensos: Las unidades pueden recibir ascensos después de ganar batallas.

15.6. Condiciones de victoria

- Diferentes tipos de victoria:
  - Conquista: conquista todas las ciudades del oponente
  - Ciencia: Llega al final del árbol tecnológico y construye una nave espacial.
  - Cultura: Alcanzar un nivel muy alto de cultura
  - Medalla: Sé la civilización más poderosa hasta el límite de tiempo.

16. CONSEJOS DE IMPLEMENTACIÓN

16.1. Estrategias de desarrollo

1. Desarrollar en fases:

- Fase 1: Implementar el sistema básico de usuario y juego.
- Fase 2: Implementar el mapa y el movimiento básico Fase 3: Agregar
- ciudades y gestión de recursos Fase 4: Implementar un
- sistema de combate simplificado Fase 5: Integrar IA con GroQ
- Fase 6: Refinar y optimizar
-

2. Desarrollo paralelo según roles:

   - Frontend: puede comenzar con simulaciones de API para progresar sin dependencias de backend
   - Backend: puntos finales con respuestas inicialmente estáticas se puede implementar
   - IA/ML: Trabajando con modelos locales antes de la integración final con GroQ poder

3. Reducir el alcance (para mantener 50 horas por estudiante):

   - Sistema de combate simplificado (opción de simulación)
   - Minimizar el número de unidades y tipos de edificios
   - Priorizar la funcionalidad sobre el diseño visual
   - Minimizar elementos opcionales (sonidos, animaciones)

16.2. Recomendaciones específicas para la simplificación

- Árbol tecnológico reducido: 15-20 tecnologías son suficientes para los mecánicos para mostrar
- Tamaño de mapa mediano: 50x50 celdas es suficiente para el juego.
- Número límite de civilizaciones: Jugador 1 e IA 1, solo 2 en total
- Limite el número de estilos: 3 estilos son suficientes (antiguo, clásico y moderno)
- Sistema simplificado de ventaja de unidad: 3 tipos básicos de unidades (cuerpo a cuerpo, a distancia, caballería)
- Edificios sencillos: con funcionalidades claras, 3-4 tipos de edificios por ciudad
- Factores de productividad simples: Para simplificar las fórmulas de producción

16.3. Optimizaciones de frontend

- Mapa: Precarga del archivo de parcelas en la memoria

- Uso de iconos: Se implementó un sistema de hojas de sprites para iconos.
- Optimización de visualización: renderiza solo los gráficos visibles en la pantalla
- Animaciones simples: saltos entre posiciones en lugar de movimientos deslizantes.

16.4. Optimizaciones de backend

- Estado inmutable: Los cambios crean un nuevo estado
- Mapa de sección pequeña: cálculos solo para un mapa visible
- Simulaciones simplificadas: Simplificación del combate y las simulaciones de IA

16.5. IA para mejorar el tiempo de pensamiento

- Agrega lo que aprendiste en rondas anteriores: Mantén un historial que ayudará a la IA
- Simplifique los árboles de decisión: reduzca las opciones de decisión de la IA para centrarse en diferentes perspectivas
- Limitar los tokens de respuesta: los mensajes deben definir claramente el formato de respuesta para reducir la cantidad de tokens

16.6. Prioridades de desarrollo para garantizar la jugabilidad

1. Visualización básica del mapa y movimientos
2. Sistema de turnos funcionales
3. Construcción mínima en las ciudades
4. Reclutamiento de unidades
5. Sistema de recursos simplificado
6. Sistema básico de combate
7. Árbol tecnológico simplificado
8. Oponente básico de IA
9. Verificación de las condiciones de victoria
10. Sistema de guardado y carga de juegos

Limítese a implementar estas funcionalidades en la primera fase del proyecto, y una vez que esté funcional, agregue cualquier extensión o complejidad adicional.