UTN
Facultad Regional Córdoba

TUP

Tecnicatura Universitaria en Programación

PROGRAMACION III
FRONT END - BACK END

Trabajo Práctico Integrador:
TEG

TPI
2° Año – 3° Cuatrimestre

Índice

Trabajo Práctico Integrador

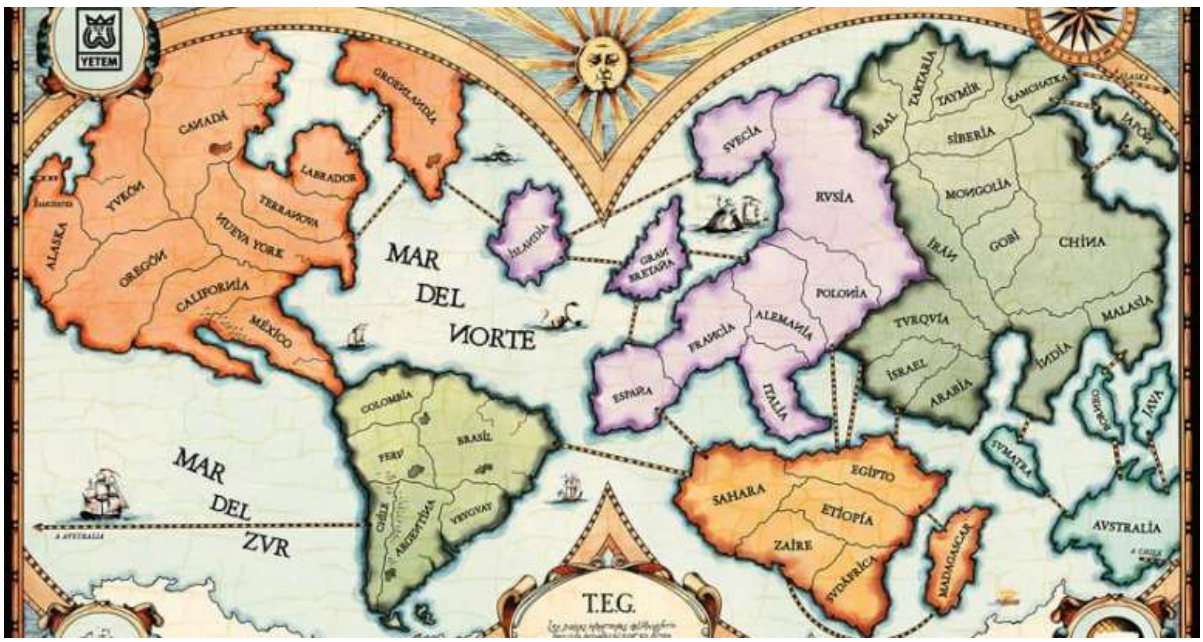
Introducción

El **TEG** (Táctica y Estrategia de la Guerra) es un clásico juego de mesa de estrategia que desafía a los jugadores a conquistar el mundo mediante la planificación, la negociación y la toma de decisiones tácticas. Inspirado en juegos de guerra y geopolítica, el TEG combina azar y estrategia, obligando a los jugadores a gestionar recursos, evaluar riesgos y anticipar movimientos enemigos para lograr sus objetivos de dominación global. El juego propone un conflicto bélico que tiene lugar sobre un mapa-tablero dividido en 50 países.

En este contexto, como parte de la materia **Programación III**, se presenta el desafío de desarrollar una versión digital del TEG, implementando tanto el **front-end** como el **back-end** del juego. Este proyecto no solo pondrá a prueba los conocimientos adquiridos en programación, sino que también requerirá una comprensión profunda de las reglas y mecánicas del juego, además de habilidades en diseño de interfaces, manejo de datos y lógica de negocio.

Los estudiantes deberán aplicar conceptos clave de desarrollo de software, tales como arquitectura cliente-servidor, comunicación a través de APIs, gestión del estado del juego y sincronización de eventos en una aplicación interactiva y funcional.

Este desafío ofrece una oportunidad única para integrar conocimientos teóricos con su aplicación práctica en un proyecto realista, fomentando la creatividad y el trabajo en equipo.



Objetivos

El objetivo principal de este trabajo integrador es aplicar los conocimientos adquiridos durante las clases de Programación III para desarrollar el juego TEG completamente funcional en **Java con Spring Boot** (**Back-End**) y **Javascript o Angular** (**Front-End**). Esto implica diseñar e implementar un programa que permita de 2 a 6 jugadores jugar entre sí al TEG, siguiendo todas las reglas del juego y permitiendo las funcionalidades esenciales del juego, como la conquista de nuevos territorios mediante las batallas, el desplazamiento de ejércitos y la dinámica general de cada turno.

El juego de TEG desarrollado también debe incluir características avanzadas que demuestren la comprensión y aplicación de los conceptos de programación aprendidos, como la implementación de patrones de diseño, la gestión eficiente de la memoria, la modularidad y la reutilización de código, la manipulación de estructuras de datos complejas, la implementación de algoritmos de búsqueda y evaluación de jugadas. También deben aplicarse los conocimientos adquiridos sobre la gestión del código fuente, su versionado y manipulación.

El desarrollo del juego de TEG en Java para back-end y Javascript para front-end requerirá la aplicación de conocimientos de programación orientada a objetos, funcional y prototípica respectivamente, manejo de estructuras de datos, algoritmos de búsqueda, manejo de interfaces gráficas de usuario, pruebas unitarias, y el uso de herramientas como Spring Boot, Maven, JUnit, Mockito, Angular, entre otros.

Este proyecto no solo permitirá a los estudiantes aplicar los conceptos y técnicas aprendidas en el aula, sino que también los desafiará a enfrentarse a situaciones reales de desarrollo de software, como la planificación, diseño, implementación, pruebas y entrega de una aplicación de software completa y funcional. Además, el uso de herramientas de control de versiones y la colaboración en un entorno de desarrollo colaborativo como GitHub les brindará una experiencia valiosa en el uso de herramientas profesionales ampliamente utilizadas en la industria.

Adicionalmente, los alumnos deberán desarrollar **Bots** (jugadores virtuales) que permitan a una o varias personas humanas sumar estos para completar los miembros de una partida. Estos **Bots**, deberán implementar distintos perfiles que serán elegidos al momento de iniciar una partida. **Este punto es opcional**, **les permitirá aprender e implementar patrones de diseño más complejos y algoritmos como "Dijkstra" que permite encontrar el camino más corto entre dos puntos de una red.**

Requerimientos Funcionales.

La siguiente lista de requerimientos debe cumplirse para que el trabajo se considere completo y correcto, **a excepción de los requerimientos opcionales, que serán considerados como parte de la nota normativa para aspirar a una posible promoción**.

1) **Reglas del juego:**
   a) Implementación de las reglas básicas del TEG, como:
      i) Reparto de países y objetivos.
      ii) Objetivo común.
      iii) Objetivos secretos (Ocupación y Destrucción).
      iv) Dos vueltas para agregar ejércitos.
      v) Iniciación de hostilidades (Ataque, Reagrupar, solicitar Tarjeta de Paises)
      vi) Continuación del juego (Incorporar ejércitos antes de las hostilidades)
      vii) Finalización del juego (Por objetivo Común o Secreto)
   b) Verificación del orden de ejecución o no de los pasos de un turno.
   c) Validación de ataques legales basada en las reglas del juego, incluyendo la verificación de la cantidad de ejércitos y a que países, la detección de movimientos ilegales (como la cantidad mínima de ejércitos para atacar y defender o los países habilitados).
   d) Canje de tarjetas con sus reglas.
   e) **Opcional**: la implementación de reglas especiales como:
      i) Pactos (Pactos entre países, no agresión, zona internacional)

2) **Gestión del juego:**
   a) Gestión de la partida de TEG, incluyendo el inicio de una nueva partida, la gestión de los turnos de los jugadores y su orden, ataques, la detección de la finalización de la partida.
   b) Gestión de los diferentes estados del juego, como el estado inicial, de juego en curso, y la finalización del juego con la victoria de uno de los jugadores.
   c) El juego debe mantener un historial de los eventos realizados en cada partida.
   d) Implementar las reglas para los casos de partidas de 2 o 6 jugadores.
   e) **Opcional**: Implementar un chat de comunicación para los jugadores de cada partida.
   f) **Opcional**: Comunicación durante el juego (“Vale todo” y “Fair Play”). Implementar al inicio de la partida la configuración de las reglas de comunicación y controlar que se cumplan las reglas implementando un modelo de denuncia y votación entre los jugadores para saldar si se ha roto la regla.
   g) **Opcional**: Gestión del historial de movimientos, permitiendo a los jugadores revisar los movimientos anteriores y retroceder en el historial.
   h) **Opcional**: Gestión de múltiples partidas, permitiendo que varios juegos de TEG se puedan llevar a cabo de manera concurrente o consecutiva.

3) <u>Gestión de jugadores y partidas</u>:

a) Gestión de múltiples jugadores, permitiendo de 2 a 6 jugadores humanos compitan entre sí en la misma computadora.
b) Gestión de los diferentes estados de los jugadores, en juego o eliminados.
c) Creación de "bots" con modelos con diferentes niveles de dificultad para completar una cantidad de jugadores que no supere nunca los 6. Los perfiles serán:
   i) **Novato**
      (1) *<u>Reglas generales</u>:*
         (a) No tiene una estrategia definida a largo plazo.
         (b) Ataca siempre que sea posible, sin evaluar consecuencias.
         (c) No prioriza defensa ni bloqueos.
      (2) *<u>Fase de ataque</u>:*
         (a) Si tiene más tropas que un país enemigo adyacente, ataca automáticamente.
         (b) Si tiene múltiples opciones de ataque, elige una al azar.
         (c) Si logra conquistar un territorio, mueve la mitad de sus tropas allí.
         (d) Si no puede atacar, pasa de turno sin reagrupar.
      (3) *<u>Fase de refuerzos</u>*:
         (a) Coloca tropas al azar en cualquiera de sus territorios.
         (b) No intenta consolidar fronteras ni reforzar posiciones estratégicas.
      (4) *<u>Fase de reagrupación</u>*:
         (a) Mueve tropas al azar entre territorios sin seguir una estrategia clara.
   ii) **Balanceado**
      (1) *<u>Reglas generales</u>*:
         (a) Evalúa probabilidades antes de atacar.
         (b) Reparte tropas de manera más inteligente.
         (c) Intenta completar su objetivo, pero sin priorizarlo al 100%.
      (2) *<u>Fase de ataque</u>*:
         (a) Si puede atacar un país con al menos el doble de tropas que el enemigo, ataca.
         (b) Si hay varios países atacables, prioriza los que sean parte de su objetivo.
         (c) Si no tiene ataques favorables, pasa sin atacar.
         (d) Mueve tropas a los territorios conquistados, pero deja al menos 2 tropas de reserva en el original.
      (3) *<u>Fase de refuerzos</u>*:
         (a) Refuerza primero los territorios fronterizos más expuestos, los que tienen menos tropas que sus vecinos.
         (b) Si tiene tropas extra, las coloca en países clave para cumplir su misión.
      (4) *<u>Fase de reagrupación</u>*:
         (a) Mueve tropas hacia la frontera si tiene un bloque seguro de territorios.
         (b) Si un territorio aislado tiene tropas en exceso, mueve tropas hacia otro territorio propio.

iii) **Experto**

(1) *Reglas generales*:

(a) Evalúa con precisión riesgos y oportunidades.

(b) Ataca solo cuando es altamente favorable.

(c) Distribuye y reagrupar tropas para bloquear oponentes y cumplir objetivos.

(2) *Fase de ataque*:

(a) Evalúa probabilidad de éxito (ejemplo: solo ataca si tiene al menos 3 veces las tropas del enemigo).

(b) Prioriza atacar territorios que lo acerquen a cumplir su misión.

(c) Si no hay ataques estratégicamente beneficiosos, no ataca y refuerza en su turno.

(d) Redistribuye tropas para no dejar territorios débiles después de atacar.

(3) *Fase de refuerzos*:

(a) Refuerza territorios clave para su misión sin descuidar defensas.

(b) Si detecta que un jugador está cerca de ganar, coloca tropas para bloquearlo.

(4) *Fase de reagrupación*:

(a) Reagrupa tropas para consolidar defensas y abrir caminos para futuros ataques.

(b) Si detecta que otro jugador está intentando ganar, redistribuye tropas para evitarlo.

4) Guardado y carga de partidas:

a) El juego debe guardar el estado del mismo y todos sus elementos automáticamente después de cada turno de cada jugador por si alguna eventualidad sucede. Dicha partida puede ser retomada siempre que haya al menos 1 jugador humano para hacerlo, si después de 5 minutos no se presentan el resto de los jugadores, la partida termina sin que haya ganador.

b) El juego debe permitir a los jugadores guardar y cargar partidas en curso para poder reanudarlas en otro momento. Esto implica implementar un sistema de guardado y carga de partidas que sea confiable y seguro, y que mantenga la integridad de los datos del juego. Todos los jugadores deben acordar dejar de jugar para poder pausar y guardar la partida. Lo mismo debe suceder para retomar la partida, todos los jugadores deben retomarla para que pueda continuar el juego, si no sucede, el juego continuará pausado.

c) La interfaz debe incluir botones y opciones para guardar el progreso de la partida y cargar partidas guardadas.

d) Asegurar que las partidas guardadas mantengan todos los elementos del juego de forma correcta, incluyendo las posiciones de los ejércitos, los territorios ocupados y el estado de los jugadores.

5) Gestión de tiempos:

a) El juego debe implementar la gestión del tiempo de inactividad, incluyendo la posibilidad de establecer límites de tiempo por turno. Si un jugador no hace ningún movimiento después de cierto tiempo, su turno termina, sus tropas no se reagrupan y no se hacen canjes.

b) Si el tiempo de inactividad se da durante el paso de asignación de ejércitos, el sistema asignará aleatoriamente sus ejércitos en sus países.

c) Visualización clara del tiempo disponible para cada jugador, con un contador regresivo que indique el tiempo restante en su turno.

6) Interfaz de Usuario:

a) Implementar una interfaz de usuario interactiva que permita a los jugadores gestionar y visualizar el estado de la partida en tiempo real (reparto de países, visualización de objetivos, control de ejércitos, etc.).

b) La visualización del mapa del juego debe ser clara y permitir al usuario interactuar de manera intuitiva, con un diseño que destaque los territorios, las fronteras, los países ocupados y las zonas de ataque.

c) Implementar botones y controles para realizar acciones como "Reagrupar", "Atacar", "Solicitar Tarjeta de País", "Verificación de Objetivos", "Ver Historial", "Finalizar Partida", etc.

d) La interfaz debe permitir la selección de la cantidad de jugadores y configuración inicial antes de iniciar la partida.

e) Incluir notificaciones visuales para indicar acciones o eventos importantes (como ataques exitosos, cambios de turno, objetivos alcanzados, o fin de partida).

Requerimientos No Funcionales

La siguiente lista de requerimientos debe cumplirse para que el trabajo se considere completo y correcto, **a excepción de los requerimientos opcionales, que serán considerados como parte de la nota normativa**.

1) **Rendimiento y optimización:**
   a) El juego debe tener un rendimiento óptimo, con una respuesta rápida a las interacciones del usuario, incluso en condiciones de carga de trabajo intensa.
   b) Se debe realizar una optimización adecuada del código y del uso de recursos, considerando la eficiencia en la ejecución del juego y la gestión de memoria.
   c) El frontend debe ser eficiente en la renderización del mapa de juego y las acciones interactivas, con tiempos de respuesta acordes al juego.
   d) Optimizar la carga de los recursos, como imágenes, gráficos y otros elementos visuales (mapas, íconos, etc.), utilizando formatos eficientes como WebP y SVG.
   e) Las animaciones y transiciones deben ser suaves y no afectar la experiencia del usuario. Los movimientos de los ejércitos, ataques y reagrupaciones deben ser fluidos.

2) **Calidad del código y buenas prácticas de programación:**
   a) El código debe seguir las convenciones de codificación de **Java (en versión 17)** y mantener una alta calidad en términos de legibilidad, mantenibilidad y escalabilidad.
   b) Se debe aplicar el principio de responsabilidad única (Single Responsibility Principle) y otros principios de diseño **SOLID** para garantizar una arquitectura de software robusta y modular.
   c) Se debe utilizar **Maven** como herramienta de gestión de dependencias y construcción del proyecto, siguiendo las mejores prácticas de estructura de proyectos y gestión de versiones.
   d) El código frontend debe estar estructurado siguiendo las mejores prácticas de Angular o Javascript, utilizando TypeScript con tipado estricto y una arquitectura mvc(model-view-controller).
   e) Debe seguir la guía de estilo oficial de Angular, asegurando que los componentes sean reutilizables, eficientes y fáciles de mantener.
   f) La interfaz debe ser completamente funcional y usable en dispositivos de diferentes tamaños (móviles, tabletas y escritorios) y ser accesible desde los navegadores más comunes como Chrome, Firefox, Safari y Edge.
   g) El diseño debe adaptarse correctamente a la orientación vertical y horizontal de los dispositivos móviles.

3) **Pruebas unitarias y cobertura de código:**
   a) Se deben implementar pruebas unitarias utilizando **JUnit** y **Mockito** para garantizar la calidad del código y su correcto funcionamiento.
   b) Se debe lograr una cobertura de código con **JaCoCo** de **al menos 80%**, asegurando que las pruebas cubran la mayoría de las funcionalidades del juego.

4) **Usabilidad y accesibilidad:**
   a) La interfaz de usuario debe ser intuitiva y fácil de usar, con una experiencia de usuario agradable.

5) **Diseño de software y patrones de diseño:**
   a) Se debe aplicar un enfoque de diseño de software modular y bien estructurado, utilizando patrones de diseño apropiados para mejorar la calidad y mantenibilidad del código.

      Ejemplos de patrones de diseño que podrían ser aplicados en el desarrollo del juego TEG incluyen el patrón de diseño de Observer para gestionar eventos de usuario y el patrón de diseño de State para gestionar los diferentes estados del juego, como el estado de juego en curso, etc. o Strategy para los perfiles de los bots.

6) **Seguridad:**
   a) Se debe evitar el uso de librerías (dependencias) que poseen vulnerabilidades críticas/high conocidas.
   b) **Opcional**: Se deben implementar medidas de seguridad adecuadas para proteger la integridad del juego y la confidencialidad de los datos del usuario, si los hubiera.
   c) **Opcional**: Se deben evitar vulnerabilidades conocidas, como inyección de código, desbordamiento de búfer y otros ataques comunes.
   d) Implementar validaciones en todas las entradas de usuario para prevenir posibles errores de seguridad (incluso en la interfaz).
   e) No almacenar datos sensibles en localStorage o sessionStorage, y asegurar que las interacciones sean seguras.
   f) Implementar un manejo adecuado de errores en todas las interacciones con el backend (como al guardar el estado de la partida o cargar datos).

7) **Optimización de la Experiencia de Usuario:**
   a) Incluir retroalimentación visual y sonora para cada acción realizada (por ejemplo, al hacer un ataque, seleccionar un objetivo, pasar turno, etc.).
   b) Las transiciones entre los diferentes estados del juego deben ser suaves, como entre la pantalla inicial y el tablero de juego, así como entre los estados de "es tu turno" y "turno del otro jugador".
   c) Proveer atajos de teclado para facilitar la navegación entre las principales funcionalidades del juego, como "Atacar", "Reagrupar", "Ver objetivos", "Pausar", etc.

Es importante que los alumnos consideren estos requerimientos no funcionales como parte integral del desarrollo del juego TEG, para garantizar un software de alta calidad, eficiente y con características profesionales.

Forma de trabajo

Para este proyecto, **los alumnos trabajarán en grupos de 6 o 7 personas**, lo que fomentará el trabajo en equipo y la colaboración entre los miembros del grupo. Cada equipo será responsable de autogestionarse y organizarse de manera eficiente para llevar a cabo todas las etapas del desarrollo del juego, desde el diseño y la implementación hasta las pruebas y la entrega final.

Una parte fundamental de este proyecto será **el uso de herramientas de control de versiones, específicamente Git y GitHub, usando el workflow "GitFlow"** aprendido en clases. Cada equipo deberá utilizar Git como sistema de control de versiones para mantener un registro de los cambios realizados en el código fuente del juego y trabajar de manera colaborativa en un repositorio de GitHub. Además, **se creará un aula virtual (classroom) en GitHub específica para este proyecto**, donde los equipos podrán acceder a los recursos necesarios, realizar seguimiento del progreso del proyecto y entregar sus entregables.

El trabajo en equipo y la colaboración serán elementos clave en el desarrollo del proyecto. Los equipos deberán dividirse las tareas de manera equitativa, asignando responsabilidades a cada miembro en función de sus habilidades y conocimientos. Además, deberán comunicarse de manera efectiva para coordinar y sincronizar sus esfuerzos, asegurándose de que todos los miembros del equipo estén alineados y contribuyan activamente al progreso del proyecto.

Forma y fechas de entrega

La entrega del trabajo se realizará a través del aula virtual (classroom) específica para este proyecto en GitHub. Se establecerá una fecha límite de entrega, la cual deberá ser respetada por todos los equipos. Es importante tener en cuenta que todos los requerimientos funcionales y no funcionales obligatorios deberán estar cumplidos en la fecha de entrega para poder aprobar el proyecto. Cualquier entrega realizada después de la fecha límite será considerada como tardía y estará sujeta a penalizaciones en la evaluación final del proyecto.

La fecha de entrega será fijada hasta el día XXXXX NN de XXXXX del NNNN a las HH:mm Hs.

Evaluación y Calificación:

Es importante destacar que cumplir con los requerimientos establecidos es un criterio mínimo para aprobar el proyecto, pero para obtener una evaluación sobresaliente se valorará también la calidad del código, la eficiencia del diseño y la implementación, la cobertura de pruebas, la claridad de la documentación y la capacidad de trabajo en equipo y colaboración demostrada durante el desarrollo del proyecto.

- Se evaluará la calidad, complejidad y presentación del trabajo como así también la oportunidad de entrega en fecha.
- Este práctico tiene carácter de obligatorio y deberá aprobarse con un 60% (al igual que los parciales) para regularizar la materia.
- La calificación alcanzada por el grupo de trabajo será la calificación individual de cada uno de los integrantes del equipo.