TP TACS

Agosto 2020

Introduccion

El objetivo del TP es realizar un juego por turnos.

La aplicación funcionará de modo stand alone, y estará publicada en la nube para ser accedida.

El TP consta de cuatro entregas en las cuales de forma iterativa e incremental se irán agregando funcionalidades a la aplicación.

Recomendaciones Generales

1. Enfocarse en los requerimientos de cada entrega. (Se puede hacer de más pero no de menos)

2. Utilizar al ayudante para validar decisiones de diseño y consultar arquitectura, frameworks, etc.

3. Dividir en forma clara en el equipo las historias de cada entrega para atacarlas en paralelo donde sea posible.

4. Utilizar alguna herramienta para la gestión de tareas (Scrummy, Trello, Issues de Github)

5. Para bajar el riesgo de las futuras entregas aprovechar el tiempo de entregas anteriores para investigar las tecnologías.
6. De ser necesario utilizar al ayudante como facilitador, en cuestiones técnicas y de organización → **El rol del ayudante NO es simplemente el de corregir, sino dar soporte al equipo durante todo el proceso en cuestiones técnicas y metodológicas.**

Juego

El juego debe ser por turnos, su lógica se centrará en 2 o más equipos que luchan por tener el control de una provincia de Argentina. La partida se gana cuando se toma control de todos los municipios de una provincia o un jugador se rinde.

Se deben utilizar las siguientes APIs para obtener la información necesaria:

- Datos GeoRef de argentina: https://apis.datos.gob.ar
- Datos topologicos: https://www.opentopodata.org/
- Imagenes: https://pixabay.com/api/docs/

2. Partidas, se debe:

   a. Generar nuevas partidas seleccionando provincia, cantidad de municipios a usar y usuarios participantes (y por lo tanto buscar usuarios)

   b. Listar todas las partidas, buscar, ordenarlas y filtrar por: fecha, estado

   c. Modificar estado de una partida (en juego, terminada)

   d. Cada partida se da sobre un mapa, el mismo es una provincia de argentina a seleccionar.

3. Acciones

   a. Se deben poder visualizar en un mapa todos los municipios que me pertenecen y los de mis enemigos (diferenciando visualmente a los mismos)

   b. Se deben poder mover gauchos entre municipios que me pertenecen

   c. Se debe poder atacar otro municipio

   d. Se debe poder modificar la especialización del municipio entre producción o defensa

   e. Se debe poder visualizar las estadísticas de producción y defensa de cada municipio. También se debe utilizar una imagen para cada municipio para ver sus detalles

4. Logica

   a. Cada turno es posible cambiar el estado de un municipio entre producción o defensa

   b. Los municipios producen

   $$Gauchos = Redondeo(15^{*}(1 - \frac{AlturaMunicipio - MinAltura}{2^{*}(MaxAltura - MinAltura}))) \text{ si municipio en modo produccion}$$

   $$Gauchos = Redondeo(10^{*}(1 - \frac{AlturaMunicipio - MinAltura}{2^{*}(MaxAltura - MinAltura}))) \text{ si municipio en modo defensa}$$

   $$MaxAltura = \text{Altura del municipio mas alto del mapa}$$

   $$MinAltura = \text{Altura del municipio mas bajo del mapa}$$

   c. Si un municipio ataca, se obtiene el municipio atacado si quedan gauchos después del ataque. Se utilizan las siguientes ecuaciones para saber el estado final del ataque

   $$GauchosAtacantesFinal = RedondeoAbajo(GauchosAtacantes * MultDist - GauchosDefensores * MultAltura * MultDefen$$

   $$GauchosDefensoresFinal = RedondeoArriba(\frac{GauchosDefensores * MultAltura * MultDefensa - GauchosAtacantes * Mul}{MultAltura * MultDefensa}$$

   $$MaxDist = \text{Distancia maxima entre municipios en el mapa}$$

   $$MinDist = \text{Distancia minima entre municipios en el mapa}$$

   $$MultDist = 1 - \frac{DistanciaMunicipios - MinDist}{2^{*}(MaxDist - MinDist)}$$

   $$MultAltura = 1 + \frac{AlturaMunicipioDefensor - MinAltura}{2^{*}(MaxAltura - MinAltura)}$$

   $$MultDefensa = 1.25^{*} \text{ si municipio en modo defensa}$$

   $$MultDefensa = 1 \text{ si municipio en modo produccion}$$

   d. Es posible mover gauchos entre municipios de un mismo jugador, pero tras recibir gauchos el municipio destino queda bloqueado y no puede realizar movimientos hacia otro lugar

   e. Todas los números con * deben ser configurables en el sistema internamente

5. Admin

   a. Se deben proveer estadísticas de cantidad de partidas creadas, en curso, terminadas y canceladas permitiendo seleccionar el rango de fechas

Ejemplos cálculos

$$AlturaMunicipio = 1500$$
$$MaxAltura = 2000$$
$$MinAltura = 1000$$
$$Gauchos = Redondeo(15(1 - \frac{1500 - 1000}{2(2000 - 1000)})) = 11 \text{ si municipio en modo produccion}$$
$$Gauchos = Redondeo(10(1 - \frac{1500 - 1000}{2(2000 - 1000)})) = 7.5 \text{ si municipio en modo defensa}$$

$$MultDefensa = 1$$
$$AlturaMunicipioDefensor = 1500$$
$$MaxAltura = 2000$$
$$MinAltura = 1000$$
$$MultAltura = 1 + \frac{1500 - 1000}{2(2000 - 1000)} = 1.25$$
$$MaxDist = 20$$
$$MinDist = 10$$
$$MultDist = 1 - \frac{15 - 10}{2(20 - 10)} = 0.75$$
$$GauchosAtacantes = 100$$
$$GauchosDefensores = 100$$
$$GauchosAtacantesFinal = RedondeoAbajo(100 * 0.75 - 100 * 1.25 * 1) = -50 \texttt{ (mueren todos los gauchos atacantes)}$$
$$GauchosDefensoresFinal = RedondeoArriba(\frac{100 * 1.25 * 1 - 100 * 0.75}{1.25 * 1}) = 40 \texttt{ (sobreviven 40 gauchos defensores)}$$

$$MultDefensa = 1.25$$
$$AlturaMunicipioDefensor = 1500$$
$$MaxAltura = 2000$$
$$MinAltura = 1000$$
$$MultAltura = 1 + \frac{1500 - 1000}{2(2000 - 1000)} = 1.25$$
$$MaxDist = 20$$
$$MinDist = 10$$
$$MultDist = 1 - \frac{15 - 10}{2(20 - 10)} = 0.75$$
$$GauchosAtacantes = 100$$
$$GauchosDefensores = 100$$
$$GauchosAtacantesFinal = RedondeoAbajo(250 * 0.75 - 100 * 1.25 * 1.25) = 31 \texttt{ (sobreviven 31 gauchos atacantes)}$$
$$GauchosDefensoresFinal = RedondeoArriba(\frac{100 * 1.25 * 1.25 - 100 * 0.75}{1.25 * 1.25}) = -20 \texttt{ (mueren todos los gauchos defensores)}$$

Requerimientos no funcionales

Los requerimientos no funcionales no solo son importantes para aprobar el TP sino que están directamente relacionados con la filosofía y objetivos de la materia. **La calidad no se negocia.**

Tecnicos

- Se debe utilizar Github/GitLab como SCM.
- El nivel de cobertura de tests debe ser superior al 70%.
- Todos los métodos no triviales deben tener su correspondiente doc (ej: javadoc) explicando su función, forma de uso y cualquier otra información relevante.
- **Se debe incluir en el README.me/Wiki como levantar la aplicación y cualquier decisión respecto del código o las soluciones utilizadas.**
- La aplicación debe ser capaz de correrse desde Maven/Gradle/SBT/Node, el comando a correr debe iniciar la aplicación dentro de un Docker container.
- Se debe usar docker-compose para definir el conjunto de aplicacion + db + network de modo tal que se pueda correr todo con un solo comando. Esto es obligatorio para modo local, si en la nube se va a

(manejo de contraseñas, recursos externos, etc.)

1. Las claves deben ser guardadas de forma correcta.

2. En caso de existir, las API keys NUNCA debe ser expuestas al usuario

● La aplicación debe soportar un load test, se utilizara alguna tool como Vegeta, Wrk, etc

● La aplicación debe ser portable, requiriendo solamente de Gradle/Maven/SBT/Node + Docker para su prueba y evaluación.

UX

● Si bien se espera algo sencillo, la aplicación debe tener un frontend amigable. Recomendamos seguir los lineamientos de https://material.io/

● Utilizar algun framework CSS (Bootstrap, etc)

Condiciones para promoción

1. Utilizar alguna forma de CI (Travis, Jenkins, Circle CI, etc)

2. La aplicación debe integrar monitoreo (Datadog, NewRelic), con métricas de CPU, Memoria, Disco, latencia y requests por endpoint

3. Permitir al crear una partida configurar las variables de las ecuaciones con 3 “modos de juego”

4. Permitir crear partidas de hasta 4 jugadores

5. Recibir un email cuando es el turno de jugar de un usuario y el mismo no lo utilizó por más de un minuto

6. Entregas en fecha

Entregas

● Las entregas deberán realizarse el día pactado antes de las 19 Hs. con un tag llamado Entrega_XX correspondiente al número de entrega.

● Las entregas se realizarán indicando el link al repositorio y el tag para la entrega.

● Todo retraso en una entrega que no haya sido correctamente comunicado y justificado tendrá como penalización el agregado de nuevos requisitos para la aprobación final del TP.

Entrega 1

Esqueleto de la aplicación WEB.
Se debe definir un primer approach hacia los recursos y URLs REST que se utilizarán para cumplir con las historias propuestas. Para esta entrega no es necesario que las historias funcionen sino que los recursos devuelvan respuestas ficticias estáticas. No es necesario tener un frontend en esta instancia. Para esta entrega si es necesario que la app corra dentro de Docker.

Entrega 2

Se debe definir el comportamiento de los principales servicios relacionados al dominio y cumplir con la funcionalidad persistiendo en memoria. La integración principal con las API externas de datos GeoRef y Topológicos debe estar disponible

Entrega 3

Segunda versión del front end: user stories 3 y 4. debe ser posible jugar una partida.

Integración con API de imágenes.

Entrega 4

Persistencia utilizado una base de datos. Se debe modificar la aplicación para que en vez de almacenar los datos en memoria, la misma lo haga utilizando alguna base a definir por el equipo.

Tercera version del front end: user story 5, pantalla de administrador

Entrega 5

Para esta entrega la aplicación debe estar deployada en la nube, el deploy en la nube debe ser utilizando Docker containers.

Entrega Final

Entrega final con detalles pulidos de funcionalidades faltantes y correcciones finales