OwnGame

Dario Turco

18 March 2020

Tabla de contenidos

1 Introduccion

En el siguiente projecto voy a crear un clon del juego Ogame, el mismo es un juego de estrategia en tiempo real con tematica sobre naves espaciales y tecnologias futuristicas. En este cada jugador empieza con un planeta y a medida que crece puede crear una flota de naves espaciales, la cual puede utilizar para luchar por recursos con otros jugadores y conquistar nuevos planetas. El juego original es propiedad de Gameforge y este projecto solo tiene fines educativos.

2 Objetivo

El objetivo de este projecto es el de poder introducir bots de manera facil al juego, arreglar algunos problemas que el juego original presenta y por ultimo, poder mejorar el rendimiento general del juego.

En cuanto a los bots, esto se hace con una `API` con la cual un script externo puede interactuar con el servidor, luego el como funcionan los bots es esta acargo de la persona que creo su script. El funcionamiento de la `API` esta explicado en la seccion "**API para bots**".

En cuanto a los problemas que necesitan arreglo en la version original, la mayoria tiene que ver con la performance del servidor y de los item premium que estan inclinando el juego a favor de la gente que mas dinero pone. En resumen, la siguente lista indica la lista de los cambios mas significativos que se realisaron con respecto a la version original, todos estos cambios tienen como objetivo solucionar algun problema encontrado:

- El cambio de nombre de los planetas y lunas se hace el la vision general directamente.
- No existen las alianzas.
- Se elimino todo lo que tenga que ver con la materia oscura y las funcionalidades premuim.
- Se cambiaron misiones del tutorial para que tenga ma sentido y ayuden mas al jugador nuevo.
- Se pueden desplegar naves a planetas de otros jugadores.
- Un jugador puede atacarce a si mismo.
- Existe el edificio lunar, sun shade, el cual puede bajar la temperatura del planeta.
- Existe el edificio lunar, lunar beam, el cual puede subir la temperatura del planeta.
- Existe el edificio lunar, market place, en el cual se puede cambiar recursos y ganar ofertas.
- Existe el edificio lunar, lunar shield, el cual hace mas dificil de destruir una luna.
- El spacial dock ahora es un edicio lunar.
- Se elimino la opcion de ataque ACS attack y ASC defense.
- Se cambio la funcion del deposito de alianza, se necesita para tener un comercio en la luna.
- Cada misil ocupa solo un lugar en el silo, sin importar de que tipo sea.
- Las naves gastan mucho menos deuterio al moverse.
- Pueden cancelarse las construcciones de hangar que estan en la cola.
- Los misiles se usan como una nave de ataque mas.
- Se puede enviar recicladores y ataques con cargueros desde la galaxia o la vista de mensajes.

Finalmente para mejorar el rendimiento del servidor rehice todo el codigo del servidor y del cliente. De todas maneras es obligatoria ya que no cuento con el codigo original del servidor (aunqe si del cliente web). Con esto espero que al poder entender la totalidad del codigo sea mas facil hacerle modificacion y cambios a la hora de aplicar una posible mejora. Ademas esto tiene otro efecto positivo el cual es que, al crear todo el juego en javascripts con librerias especificas, voy a aprender su funcionamiento de manera practica, lo cual se alinea con la idea de usar este projecto para aprender sobre desarroyo web.

3 Tecnologias usadas

Todo el codigo del servidor esta todo hecho en *JavaScript* y corre sobre Node. Para atender las peticiones de la `API` y del cliente web usa *Express* y el motor de renderisado es *Pug*. Esto es para mentener el diseño simple y minimalista ya que con esos modulos me alcanza para poder realizar todo el proyecto.

En la parte de seguridad la pagina web no tiene ningun tipo de medida, cualquiera se puede conectar con cualquier usuario y hacen lo que quiera. Esto es ai porque la idea de todo esto es que bots jueguen atravez de la `API` y la interfaz web sirve para que alguien pueda monitorear todo el universo o un bot en espesifico. Por otro lado la `API` si tiene un sistema de autenticacion con constraseña, el cual tiene cierto nivel de seguridad ya que se usa la libreria *passport.js* sin embargo el objetivo de este projecto no es preveer una `API` segura, esta autenticacion esta porque es mas comodo identificar a los clientes que hablan con la `API`.

Por otro lado, al ser mucho el volumen de informacion manejado por el juego, el seridor utiliza una base de datos interna proveida por *MongoDB*. Ademas usa *cookie-parser*, *morgan*, *dotenv*, *nodemon* en modo debug entre otras librerias secundarias.

4 Funcionamiento

Este projecto pretende tener un codigo autodocumentado, si bien en algunas partes eso no esta tan claro, creo que es la mejor manera de documentar como funciona la apliccion. Sin embargo en esta seccion voy a explicar brevemente como funciona, a grandes rasgos, la aplicacion. Este analisis no pretende ser exaustivo, para mas informacion sobre el codigo, lo mejor es mirarlo uno mismo.

El funcionamiento de la aplicacion consiste en un tener dispobles las rutas que se proveen al cliente y cada una, de ser una pagina se envia la misma con los datos correspondientes y en caso de ser una ruta de la `API` se ejecutan las acciones correspondientes.

4.1 Objetos importantes

Algun informacion clave del juego se guarda en objetos de *JavaScript* con ciertas propiedades, a continuacion voy a exponer los mas importantes:

Universo: Objeto con la informacion principal del universo.

- *name*: Nombre del universo.
- *inicio*:
- *maxGalaxies*: Numero maximo de galaxias.
- *donutGalaxy*: Si las galaxias son circulares.
- *donutSystem*: Si los sistemas solares son circlares.
- *speed*: Velocidad del universo.
- *speedFleet*: Velocidad de las flotas del universo.
- *fleetDebris*: Porcentage de 0 a 100 de flota a escombros.
- *defenceDebris*: Porcentage de 0 a 100 de defensa a escombros.
- *maxMoon*: Maxima probabilidad de crear una luna.
- *rapidFire*: Booleano que esta en true si esta activado el fuego rapido en el universo.

Player: El objeto que guarda toda la informacion de un jugador.

- *id*: Id en la base de datos del jugador.
- *name*: Nombre del jugador, es importante que no existan dos jugadores con el mismo nombre.
- *pass*: Contrasenia para identificarse al usar la `API`.
- *styleGame*:
- *planets*: Lista que guarda los objetos de los planetas del jugador.
- *maxPlanets*: Cantidad maxima de planetas que puede tenes el jugador.
- *highscore*: Posicion en el ranking del universo.
- *lastHighscore*:.
- *puntos*: Puntos del jugador(cada 1000 unidades gastadas de cada recurso se da 1 punto.
- *puntosAcum*: Cantidad de recursos que se gasto pero no alcazo a llegar a 1000 para convertirce en 1 punto.
- *vacas*: Lista de jugadores que el jugador puede modificar, son para hacer mas rapidos los ataque.
- *sendEspionage*: Cantidad de sondas de espionaje que se envia al apretar espiar en la vista de la galaxia.
- *sendSmall*: Cantidad de small cargos que se envia al atacar a una vaca.
- *sendLarge*: Cantidad de large cargos que se envia al atacar a una vaca.
- *dark*: Cantidad de materia oscura del jugador .
- *messagesCant*: Cantidad de mensajes sin leer del jugado.
- *messages*: Lista con todos los mensajes del jugado.
- *movement*: Lista con todos los movimientos de flota del jugado.
- *researchConstrucction*: Si no se investiga nada es un bool que esta en false si no es un objeto con el formato.
  - *metal*: Metal que requirio la investigacion.
  - *crystal*: Cristal que requirio la investigacion.
  - *deuterium*: Deuterio que requirio la investigacion.
  - *item*: Nombre de la investigacion.
  - *planet*: Numero de planeta que inicio la investigacion.
  - *init*: Tiempo en que se incio la investigacion.
  - *time*: Duracion de la investigacion.
- *tutorial*: Lista de booleanos que dicen cual tutorial fue completado y cual no.
- *research*: Objeto con los niveles de todas las tecnologias del jugador.
  - *energy*: Nivel de la tecnologia de enrgia.
  - *laser*: Nivel de la tecnologia laser.
  - *ion*: Nivel de la tecnologia ionica.
  - *hyperspace*: Nivel de la tecnologia de hyperespacio.
  - *plasma*: Nivel de la tecnologia de plasma.
  - *espionage*: Nivel de la tecnologia de espionage.
  - *computer*: Nivel de la tecnologia de computacion.
  - *astrophysics*: Nivel de la tecnologia astrifisica.
  - *intergalactic*: Nivel de la tecnologia intergalactica.

  - *graviton*: Nivel de la tecnologia de graviton.
  - *combustion*: Nivel del motor de conbustion.
  - *impulse*: Nivel del motor de impulso.
  - *hyperspacedrive*: Nivel del motor de hyper espacio.
  - *weapons*: Nivel de la tecnologia militar.
  - *shielding*: Nivel de la tecnologia de defensa.
  - *armour*: Nivel de la tecnologia de blindaje.
- *lastVisit*: Numero que determina en que instante de tiempo se actualizaron los datos de ese jugador por ultima vez.
- *type*: Tipo de jugador en la vista de galaxia.
- *hazards*: Lista con las movements que estan atacando al jugado.

Planeta: El objeto que guarda toda la informacion de un planet.

- *idPlanet*: Id del planeta.
- *coordinates*: Objeto con las cordenadas del planeta.
- *coordinatesCod*: String con el codigo de las cordenadas, en el formato "galaxy_system".
- *player*: Nombre del jugador al que pertenece el planeta.
- *playerType*: Tipo de jugador en la vista de galaxia.
- *name*: Nombre del planeta.
- *type*: Tipo del planeta, numero del 1 al 7.
- *color*: Color del planeta, numero del 1 al 10.
- *temperature*: Objeto que guarda la temperatura maxima y minima del planeta con modificaciones.
  - *max*: Temperatura maxima.
  - *min*: Temperatura minima.
- *temperatureNormal*: Objeto que guarda la temperatura maxima y minima sin modificaciones.
  - *max*: Temperatura normal maxima.
  - *min*: Temperatura normal minima.
- *camposMax*: Campos maximos del planeta.
- *campos*: Campos que se estan usando en el planeta.
- *buildingConstrucction*: Booleano que es true cuando se esta construllendo un edificio.
- *shipConstrucction*: Lista de naves que se estan construllendo en el hangar.
- *resources*: Objeto que guarda la cantidad que hay en el planeta de cada recurso.
  - *metal*: Cantidad de metal.
  - *crystal*: Cantidad de cristal.
  - *deuterium*: Cantidad de deuterio.
  - *energy*: Cantidad de energia que se produce.
- *resourcesAdd*: Objeto que guarda la cantidad que se produce por hora de cada recurs.
  - *metal*: Produccion de metal por hora.
  - *crystal*: Produccion de metal por hora.
  - *deuterium*: Produccion de metal por hora.
  - *energy*: Cantidad de energia que sobra o falta.
- *resourcesPercentage*: Numero del 0 al 10 que regula la producion y consumo de energia de las minas.
  - *metal*: Porcentaje de produccion de la mina de metal.
  - *crystal*: Porcentaje de produccion de la mina de cristal.
  - *deuterium*: Porcentaje de produccion de la mina de deuterio.
  - *energy*: Porcentaje de produccion del reactor de fusion.
- *buildings*: Objeto que guarda el nivel de todos los edificion del planeta.
  - *metalMine*: Nivel del edificio.
  - *crystalMine*: Nivel del edificio.
  - *deuteriumMine*: Nivel del edificio.
  - *solarPlant*: Nivel del edificio.
  - *fusionReactor*: Nivel del edificio.
  - *metalStorage*: Nivel del edificio.
  - *crystalStorage*: Nivel del edificio.
  - *deuteriumStorage*: Nivel del edificio.
  - *robotFactory*: Nivel del edificio.
  - *shipyard*: Nivel del edificio.
  - *researchLab*: Nivel del edificio.
  - *alliance*: Nivel del edificio.
  - *silo*: Nivel del edificio.
  - *naniteFactory*: Nivel del edificio.
  - *terraformer*: Nivel del edificio.
- *fleet*: Objeto que guarda la cantidad de cada nave que hay en el planeta.
  - *lightFighter*: Cantidad de esa nave.
  - *heavyFighter*: Cantidad de esa nave.
  - *cruiser*: Cantidad de esa nave.
  - *battleship*: Cantidad de esa nave.
  - *battlecruiser*: Cantidad de esa nave.

  - *bomber*: Cantidad de esa nave.
  - *destroyer*: Cantidad de esa nave.
  - *deathstar*: Cantidad de esa nave.
  - *smallCargo*: Cantidad de esa nave.
  - *largeCargo*: Cantidad de esa nave.
  - *colony*: Cantidad de esa nave.
  - *recycler*: Cantidad de esa nave.
  - *espionageProbe*: Cantidad de esa nave.
  - *solarSatellite*: Cantidad de esa nave.
- *defense*: Objeto que guarda la cantidad de cada defensa que hay en el planeta.
  - *rocketLauncher*: Cantidad de esa defensa.
  - *lightLaser*: Cantidad de esa defensa.
  - *heavyLaser*: Cantidad de esa defensa.
  - *gauss*: Cantidad de esa defensa.
  - *ion*: Cantidad de esa defensa.
  - *plasma*: Cantidad de esa defensa.
  - *smallShield*: Cantidad de esa defensa.
  - *largeShield*: Cantidad de esa defensa.
  - *antiballisticMissile*: Cantidad de esa defensa.
  - *interplanetaryMissile*: Cantidad de esa defensa.
- *moon*: Objeto que guarda los datos de la luna.
- *debris*: Objeto que guarda la informacion de los escombros de ese planeta.
  - *active*: Bool que guarda si hay escombros activos o no.
  - *metal*: Cantidad de metal en los escombros, si el escombro no esta activo esta variable vale 0.
  - *crystal*: Cantidad de cristal en los escombros, si el escombro no esta activo esta variable vale 0.

Moon: Es el objeto que guarda la informacion de la luna, tiene dos campos que estan siempre disponibles exista la luna o no, los cuales son active y size.

- *active*: Boolenao que si es true entonces ese planeta tiene luna.
- *size*: Size de la luna. Si active es falso size es igual a 0.
- *name*: Nombre de la luna.
- *camposMax*: Campos maximos de la luna.
- *campos*: Campos que se estan usando en la luna.
- *type*: Tipo de luna.
- *resources*: Objeto que dice cuantos recursos hay en la luna.
- *buildingConstrucction*: Bool que dice si se esta contrullendo un edificio.
- *buildings*: Objeto que dice el nivel de cada edificio en la luna.
  - *lunarBase*: Nivel del edificio luna.
  - *phalanx*: Nivel del edificio luna.
  - *spaceDock*: Nivel del edificio luna.
  - *marketplace*: Nivel del edificio luna.
  - *lunarSunshade*: Nivel del edificio luna.
  - *lunarBeam*: Nivel del edificio luna.
  - *jumpGate*: Nivel del edificio luna.
  - *moonShield*: Nivel del edificio luna.
- *values*: Numero del 0 al 10 que representa el porcentaje de actividad del sun shade y del lunar beam.
  - *sunshade*: Porcentaje de produccion del edificio luna.
  - *beam*: Porcentaje de produccion del edificio luna.
- *cuantic*: Entero que indica el instante en el que se va a poder usar el salto cuantico otra vez.
- *fleet*: Objeto que dice cuantas naves hay en la luna, la compocicion es similar al objeto fleet de planeta.

Fleet: Es el objeto que guarda una flota o movement que viaja de un lugar a otro.

- *ships*: Objeto con la catidad de cada nave en la flota.
- *moon*: Boolenao que esta en true si la flota salio de una luna y falso si salio desde un planeta.
- *coorDesde*: Coordenadas simplificadas del lugar de partida de la flota.
- *coorHasta*: Coordenadas simplificadas del lugar de llegada de la flota.
- *destination*: Numero que indica si se el destino es un 1 = planeta, 2 = luna, 3 = debris.
- *resources*: Objeto que indica cuatos recursos lleva la flota.
- *speed*: Numero del 1 al 10 que indica a que velocidad va la flota.
- *mission*: Numero del 0 al 8 que indica que numero de mission ejecuta esta flota.
- *time*: Tiempo en que empezo el viaje.
- *duracion*: Cuanto tarda el viaje.
- *llegada*: Tiempo en que la flota llega.
- *ida*: Bool si dice si es un viaje de ida o de vuelta.
- *desdeColor*: Color del planeta de salida.
- *desdeType*: Tipo del planeta de salida.
- *desdeName*: Nombre del planeta de salida.
- *hastaColor*: Color del planeta de llegda.
- *hastaType*: Tipo del planeta de llegada.

Message: Es el objeto que guarda un mensaje del jugado.

- *date*: String con la fecha de cuando se envio el mensaje.
- *type*: Tipo de mensaje.
  - Si es 1 es un reporte de batalla.
  - Si es 2 es un reporte de espionaje.
  - Si es 3 es un mensaje de texto informativo.
  - Si es 4 es un mensaje de la categoria "otros".
- *title*: Titulo del mensaje.
- *text*: Texto del mensaje en caso de ser de tipo.
- *data*: Objeto con informacion extra del mesaje.

4.2 `API` para los bots

La aplicacion provee una `API` para que scripts externos puedan controlar un BOT en el universo. Para esto, el scipt externo debe saber el username y password del BOT, para crear estos se debe hacer directamente desde el servidor.

En este apartado voy a enumerar brevemente los puntos de entrada que provee la `API`. En el codigo, en la carpeta bots, ahi un archivo con casos de uso de estas funciones que se pueden usar a modo de ejemplo y que esta mas detallado el funcionamiento de la `API`. Todas los puntos de entrada de la `API` usan el metodo POST y son los siguientes:

- *login*: Sirve para loguearse y obtener un token id que estara activo por un tiempo.
- *logout*: Sirve para desloguearse y dar de baja el token id.
- *changePlanetName*: Cambia el nombre de un planeta.
- *abandonPlanet*: Abandona un planeta, se necesecita tener mas de un planeta para poder usarlo.
- *infoPlayer*: Devuelve el objeto con la informacion de un jugador.
- *infoUniverso*: Devuelve el objeto con la informacion del universo.
- *infoGalaxy*: Dada una galaxia y un sistema solar, devuelve la informacion sobre esa posicion del universo.
- *changeResourcesOptions*: Cambiar los porcentajes de uso de las minas de un planeta o edificios de una luna.
- *infoBuildings*: Devuelve el objeto con la informacion sobre la construccion de los edificios de un planeta.
- *buildingRequest*: Envia una solicitud de construccion de un edificio de un planeta.
- *cancelBuilding*: Cancela la construccion de un edificio de un planeta.
- *infoBuildingsMoon*: Devuelve el objeto con la informacion sobre la construccion de los edificios de un luna.
- *buildingRequestMoon*: Envia una solicitud de construccion de un edificio de una luna.
- *cancelBuildingMoon*: Cancela la construccion de un edificio de una luna.
- *infoShips*: Devuelve el objeto con la informacion sobre la construccion de las naves.
- *buildShips*: Envia una solicitud para construir naves en un planeta.
- *cancelShips*: Cancela una solicitud de construccion de naves en un planeta.
- *infoDefenses*: Devuelve el objeto con la informacion sobre la construccion de las defensas.
- *buildDefenses*: Envia una solicitud para construir defensas en un planeta.
- *cancelDefenses*: Cancela una solicitud de construccion de defensas en un planeta.
- *infoResearch*: Devuelve el objeto con la informacion sobre el desarrollo de las investigaciones.
- *buildResearch*: Envia una solicitud para realizar una investigacion.
- *cancelResearch*: Cancela una investigacion en curso.
- *sendFleet*: Envia una flota.
- *returnFleet*: Ordena a una flota regrasar al planeta sin cumplir su mision.
- *isSomeoneAttackingMe*:
- *readMessage*: Devuelve un objeto con todos los mensajes del jugador.
- *deleteMessage*: Elimina un mensaje de un jugador.
- *changeOptions*: Cambia las opciones automaticas.
- *showTechnology*: Devuelve un objeto con los requisitos para construir cada cosa en el juego.
- *showHighscore*: Devuelve un objeto con el ranking de jugadores del universo.
- *searchPlayer*: Dado un jugador busca su informacion publica.
- *seeRewards*: Devuelve un objeto con la informacion sobre las remcompensas.
- *updateReward*: Actualiza las recompensas completadas.
- *usePhalanx*: Permite usar el sensor phalanx de una luna.
- *useJumpGate*: Permite usar el salto cuantico de una luna.
- *useMarketMoon*: Permite usar el mercado lunar de una luna.
- *changePassword*: Cambia la contrasena de un usuario.

El punto de entrada *login* da, en su respuesta, un string llamado *id* el cual se puede usar para autentificarse en otros puntos de entrada sin necesidad de usar la contrasena de nuevo.

4.3 Comandos importantes

Para lanzar el servidor es importante tener la base de datos corriendo en el sistema, para esto una vez installado *mongoDB*, en linux, solo hay que usar el siguiente comando `service mongod start`.

Para correr el servidor se puede hacer en modo debuger o en modo release, para correrlo en el primer modo se tiene que usar el comando `npm run start-dev` y para el segundo `npm run start`.

5 Resultado Final

5.1 Resultado

El resultado final es un juego el cual se puede personalizar mucho y al tener la opcion de llenar el universo con distintos bots, cada uno con sus particularidades, uno puede tener muchas estrategias que probar y muchos desafios que superar.

En cuanto al conocimiento adquirido en la realisacion del projecto, pase de no saber nada sobre express y base de datos a poder manejarme bien y entender lo que estaba haciendo. Tambien aprendi como manejar el desarroyo y diseno del software en un projecto grande, en ese sentido me gustaria cambiar algunas cosas para dejar un codigo aun mas claro.

5.2 Galeria

El apartado grafico del projeto es el mismo que el del juego original, salvo por algunos pequenos casos donde no se consiguieron los assets originales o se cambio la interfaz para mejorar algun aspecto. A continuacion voy a presentar algunas imagenes correspondiente a las pantallas mas importantes del nuevo juego:

5.3 Futuras mejoras

En el futuro me gustaria mejorar todo el codigo del lado del servidor y del cliente para hacerlo mas claro y quitar algunas incongruencias que tiene, como el usar objetos que representan lo mismo pero que difieren en algunos nombres de algunas propiedades, tal como pasa con los misilies o en los objetos del cliente.

Tambien me gustaria revisar el archivo que contiene todo el codigo *css* de la aplicacion y me gustaria agregar algunas funcionalidades a la `API` de los bots. Algo que estaria bueno agregar seria mejorar el edificio lunar *market-place* para que tenga promociones diarias. Ademas estaria bueno tener un conjunto de test para poder asegurarse que cada modificacion hecha al codigo siga con la correctud del codigo.

Finalmente una vez terminada la aplicacion me encantaria poder encontrar un buen algoritmo que permita jugar Ogame a un bot de manera totalmente autonoma. Para esto es ideal esta plataforma, ya que probee un espacio ideal en donde distintos algoritmos puede competir, ser comparados y ajustados. Es por eso que estaria bueno, en un futuro, tener un projecto que use este trabajo para, justamente, comparar distintos algoritmos y encontrar el mas eficiente.