1 Proyecto

1.1 Infonmacion

1.1.1 Objetivo

El objetivo del projecto es crear una copia local del ogame, en la que se uno pueda crear su propio universo configurable, y progrmar los bots que pueblen en universo. Ademas algunas reglas fueron modificadas con el objetivo de hacer el juego mas dinamico y variado.

1.1.2 Cambios

- El maximo de planetas por jugador es 8.
- El cambio de nombre de los planetas y lunas se hace el la vision general directamente.
- No exiesten las alianzas.
- 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.
- Se cambio la vista de fleet movment.
- Un jugador puede atacarce a si mismo.
- La luna tiene edificios nuevos, la hacen un objeto mas deseale.
- 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 hac 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.

1.1.3 Juego original

1.2 Objetos Importantes

1.2.1 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: maximo porcentage de probabilidad de crear una luna
rapidFire: booleano que esta en true si esta activado el fuego rapido en el universo
repearDefenses: booleano que esta en true si se reparan las defensas tras un combate

1.2.2 player

El objeto que guarda toda la informacion de un jugados, su compocicion es la siguiente:
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 ataques
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 atcar a una vaca

sendLarge: cantidad de large cargos que se envia al atcar a una vaca
dark: cantidad de materia oscura del jugador
messagesCant: cantidad de mensajes sin leer del jugador
messages: lista con todos los mensajes del jugador
movement: lista con todos los movimientos de flota del jugador
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, su estructura es:
- 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

1.2.3 planeta

El objeto que guarda toda la informacion de un planeta
idPlanet: id del planeta
coordinates: objeto con las cordenadas del planeta
- gal: numero de galaxia
- sys: numero de systema dentro de la galaxia
- pos: numero de posicion dentro del sistema

coordinatesCod: string con el codigo de las cordenadas 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, no cambia
- max: temperaturaNormal maxima
- min: temperaturaNormal 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 recurso
- 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 active es false esta variable es 0
- crystal: cantidad de cristal en los escombros, si active es false esta variable es 0

1.2.4 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 lunar
- phalanx: nivel del edificio lunar
- spaceDock: nivel del edificio lunar
- marketplace: nivel del edificio lunar
- lunarSunshade: nivel del edificio lunar
- lunarBeam: nivel del edificio lunar
- jumpGate: nivel del edificio lunar
- moonShield: nivel del edificio lunar

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 lunar
- beam: porcentaje de produccion del edificio lunar

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

1.2.5 vaca

coordinates: cordenadas del planeta
playerName: nombre del jugador
planetName: nombre del planeta
estado: estado de actividad del jugador

1.2.6 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 con el formato metal, crystal, deuterium que indica cuato lleva la flota de cada recurso
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

1.2.7 message

Es el objeto que guarda un mensaje del jugador
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

1.2.8 infoPlanet

Es el objeto que tiene la lista “allCord”
espionage: el nivel de espionage del jugador del planeta
playerName: el nombre del jugador del planeta

1.3 Funciones importantes

1.3.1 updatePlayer(player, f, help = false)

Actualiza la informacion de ’player’(objeto) y guarda el objeto actualizado en la base de datos, despues de actualizar ejecuta la funcion f

1.3.2 createNewPlanet(cord, planetName, playerName, playerTypeNew)

Crea un nuevo objeto de tipo planeta, con las cordenadas ’cord’, de nombre ’ planetaName’, que pertenece al jugador ’playerName’

1.3.3 createNewMoon(newSize)

Crea un nuevo objeto de tipo luna, esa nueva luna tiene de tamanio ’newSize’

1.3.4 addNewPlayer(name, styleGame)

Agrega un jugador nuevo al universo de nombre ’name’ y con ’styleGame’, despues lo guarda en la base de datos

1.3.5 getActualBasicInfo(planet)

Devuelve un objeto con la informacion basica del planeta numero 'planet'. Es usado por todo view con el nombre de 'basic'. Ese objeto se compone de la siguiente manera:

name: nombre del universo
speed: velocidad de produccion del universo
speedFleet: velocidad de flota del universo
donutGalaxy: bool que dice si las galaxias son circulares o no
donutSystem: bool que dice si los sistemas son circulares o no
playerName: nombre del jugador que tiene la cuenta abierta
highscore: posicion del jugador abierto
resources: objeto que dice los recursos que tiene el planeta o luna en la que se esta
add:
dark: cantidad de materia oscura que tiene el jugador abierto
messagesNoRead: cantidad de mensajes sin leer que tiene el jugador abierto
classObjResources: objeto que dicta los colores que tiene cada recurso(si el almacen esta lleno o no)
cantPlanets: cantidad de planetas colonizados del jugador abierto
maxPlanets: cantidad maxima de planetas que puede tener el jugador abierto
numPlanet: el index de planeta en la lista de planetas del jugador abierto
planets: lista con todos los planetas(objetos) del jugador abierto
moon: booleano que dice si estas en una luna o no
format: funcion que formatea un numero a un formato lindo

1.3.6 resourcesSetting(planet)

Devuelve el objeto 'info' del view ResourceSetings. El objeto tiene el siguiente formato: basic: objeto que dice los ingresos basicos de metal y de cristal en ese universo

values: objetos con numeros de 0 al 10 que dicta los porcentajes de cada mina
buildings: objeto con los niveles de cada mina del planeta
solarSatelite: cantidad de satelites solares en ese planeta
mines: objeto que dice cuanto produce por hora cada mina
energy: objeto que dice cuanta energia producen los satelites solares(satellite:), la planta de fusion(fusion:) y la planta de energia solar(solar:)
maxEnergy: objeto que dice cuanto es el consumo de energia requerido para cada mina
usageEnergy: objeto que dice cuanto de enrgia puede usar cada mina
resourcesHour: objeto que dice cuanto ganas por hora de cada recurso
storage: objeto que dice cuanto es el almacenamiento maximo de cada recurso
plasma: nivel de la tecnologia de plasma del jugador

1.3.7 moonSetting(planet)

Devuelve el objeto 'info' del view MoonBuildings. El objeto tiene el siguiente formato:

buildings: objeto que dice los niveles de todos los edificio de la luna
values: objeto que da los porcentajes del lunar beam y del sunshade. Su formato es:
    sunshade: numero del 0 al 10
    beam: numero del 0 al 10
fleets: objeto con la cantida de cada nave en la luna
cuanticTime: numero que dice cuanto tiempo falta para poder usar el salto cuantico
cuanticMoonsCord: lista que tiene las cordenadas de todas las lunas a la que se puede hacer un salto cuantico
campos: objeto que tiene los campos maximos y campos usados de la luna. Tiene el formato:
    campos: campos actualmente usados de la luna
    camposMax: campos maximo de la luna

1.3.8 overviewActualInfo(planet)

Devuelve el objeto 'info' del view Overview. El objeto tiene el siguiente formato:

diameter: tamanio del planeta o luna
type: tipo del planeta
temperature: objeto con la temperatura efectiva del planeta
camposMax: campos maximos del planeta o luna
campos: campos usados en el planeta o luna
cantPlayers: cantidad de jugadores en todo el universo
points: cantidad de puntos del jugador

1.3.9 buildingsActualInfo(planet)

Devuelve el objeto 'info' del view Resources y Facilities. Es un objeto que tiene los niveles de todos los edificons del planeta o luna, si es un planeta ademas da el numero de satelites solares del planeta.

1.3.10 navesInfo: function()

Entrega el objeto info de la pagina Fleet. El objeto tiene la forma:

fleets: objeto con la cantidad de naves del planeta/luna

speed: lista con la velocidades base de todas las naves
misil: cantidad de misiles en el planeta, si esta en una luna siempre es 0
expeditions: cantdad de expediciones que se estan aciendo actualmente
maxExpeditions: cantidad maxima de espediciones
slot: catidad de flotas volando actualmente
maxSlot: cantidad maxima de posibles flotas volando
vacas: lista de vacas del jugador

1.3.11 fleetInfo(planet, moon)

Entrega un objeto que por cada nave que s pueda enviar en una flota retorna un objeto con la siguiente forma:
speed: velocidad de la nave con respecto a las tecnologias del jugador
carga: capacidad de carga de una nave de ese tipo
consumo: consumo de deuterio de una sola nave

1.3.12 galaxyInfo(planet)

Entrega el objeto info de la pagina Galaxy. El objeto tiene la forma:
espionage: cantidad de sondas de espionage en el planeta/luna
recycler: cantadad de recicladores en el planeta/luna
misil: cantadad de misiles interplanetarios en el planeta
slot: cantidad de flotas volando
maxSlot: cantidad de flotas maximas volando

1.3.13 systemInfo(res, gal, sys)

Dada una galaxia y un systema solar retorna un objeto con todos los planetas en ese sistema.
El objeto esta compuesto por 15 claves que son 'pos + i' donde i es un numero entre 0 y 15 inclusive.
Si no hay nadie en la posicion 'pos + i' se guarda con esa clave el objeto active: false, si hay un planeta se guarda un objeto con la forma:
active: booleano que dice si hay planeta o no en esa posicion
player: nombre del jugador duenio del planeta de esa posicion
type: tipo de planeta
color: color del planeta
name: nombre del planeta
moon: booleano que dice si hay luna en ese planeta
moonName: nombre de la luna
moonSize: tamanio de la luna
debris: booleano que dice si hay escombros en esa posicion
metalDebris: cantidad de metal en los escombros
crystalDebris: cantidad de cristal en los escombros
estado: estado del jugador duenio del planeta en esa posicion

1.3.14 updateResourcesData(f, planet, obj = null)

1.3.15 updateResourcesDataMoon(f, planet, obj)

1.3.16 contPoint(player)

Cuenta todos los puntos de un jugador y los guarda en la base de datos.

1.4 BOTS

Cada bot va a ser controlado por un proceso externo al servidor. Este proceso puede comunicarse con el servidor medieante una API, la cual hay que hacer. Para conectarse y desconectarse al server va a haber una funcion "Conect" y otra "Disconect". Estas funciones guardan el objeto player del bot en una lista para tenerlo accesible. Si el bot sabe que no se va a conectar, se desconecta para que el servidor no use tanta memoria. Y si no se conecta en un tiempo el server lo saca automaticamente. Tambien puede ofreser una funcion para que si alguien ataca ese bot el server le avise que lo estan atacando incluso si el bot no esta conectado. Los bots no van a tener seguridad, o sea un bot puede hacerse pasar por otro bot, sin embargo voy a suponer que ninguno lo va a hacer. Esto se podria solucionar con un sistema se contrasenas a la hora de conectarse.

1.5 Otros

1.5.1 Comandos para lanzar el servidor

'service mongod start' Arranca el servicio de mongo
'npm run start-dev' Arranca el servidor

1.5.2 dataset

Para pasar un dato por html se puede usar la propiedad data-val='valor' y en javascript leerla con item.dataset.val

1.5.3 idea para el servidor

Modularisar el script universe.js. Que contenga una lista de los scripts de cada jugador donde cada script de jugador contiene los algoritmos del jugador y el objeto player del propio jugador y cuando se cambia algo con una funcion de universe.js se pase ese objeto jugador desde el script.js.

1.5.4 idea para la base de datos

Modularizar los accesos a la base de datos, que se hagan todos desde un script llamado, base.js.