# Manual Tècnic Aplicació Java & Oracle

Borja Montoro

Eric Escrich

Brahian Monsalve

Aleix Sastre

16/05/2022

## INTRODUCCIÓ

El nostre projecte consisteix en un joc programat amb java on hem

## CONTINGUT TÈCNIC

### BBDD

#### DIAGRAMA DE CHEN

Per començar amb la part de “BDD” vam iniciar creant el diagrama de “CHEN” on vam indicar les entitats “User\_PW”, “Planet”, “Ship”, “Defense”, “Enemie”, “Battle”, “Defense\_Unite”, “Attack\_Unite”.

Per l’entitat “User\_PW” vam posar els seus atributs corresponents que es poden veure a la imatge, aquesta entitat l’hem relacionat amb “Enemie” perquè cada usuari té adjudicat un “Enemie”. També té relació amb “Planet” perquè cada usuari té un planeta. Per últim te relació amb la taula “Battle” perquè totes les batalles que té un usuari queden registrades.

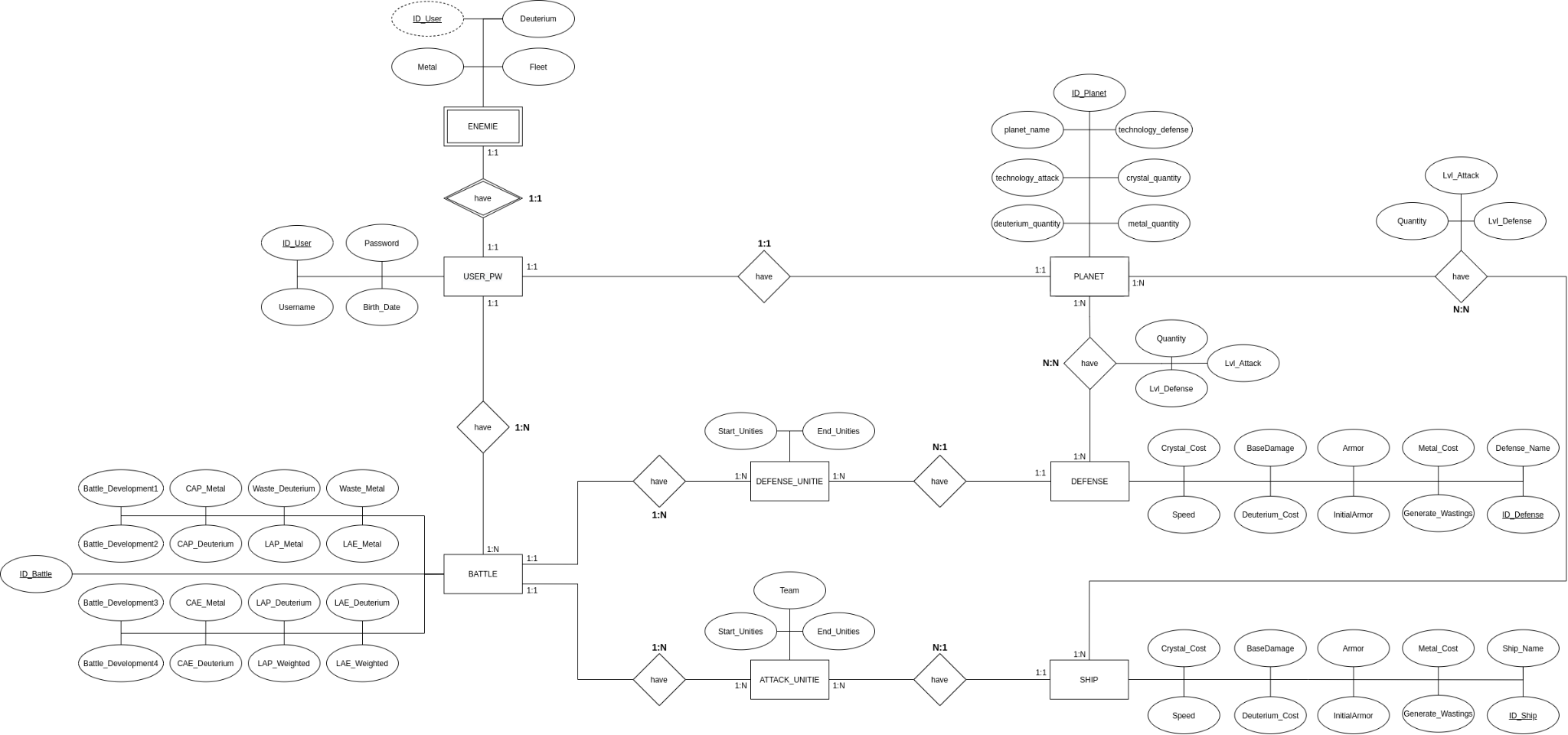
L’entitat “Enemie” té els atributs indicats en la imatge i té la relació amb “User\_PW” que està indicada aquesta entitat és heretada de “User\_PW”.

L’entitat “Planet” té els atributs indicats en la imatge i a més té les relacions amb “Ship” i “Defense” on el “Planet” pot contenir diversos “Ships” i diversos “Defense”. En la relació que hi ha entre “Planet”, “Ship” i “Planet”, “Defense” al ser una relació “N:N”. En la nostra futura base de dades es crearà una taula en la qual hem d'indicar els atributs “quantity”, “lvl\_defense”, “lvl\_attack”.

L’entitat “Battle” té els atributs indicats en la imatge i té una relació amb “User\_PW” que ja hem indicat anteriorment. També té relació amb “Defense\_Unitie”, “Attack\_Unite” on cada batalla pot tenir diverses tropes de “ship” i de “defense”.

L’entitat “Defense\_Unite” té els atributs que es veuen i té la relació explicada anteriorment amb “Battle” i té una relació amb “Defense” on una defensa pot estar en diverses Unites.

L’entitat “Attack\_Unite” té els atributs que es veuen i té la relació detallada anteriorment amb “Battle” i té una relació amb “Ship” on un “ship” pot estar en diverses “Unites”.



#### 

#### MODEL RELACIONAL

Després d’haver acabat amb el diagrama de “CHEN” vam passar a fer el model Relacional on tots els atributs poden tenir diverses qualitats i a més comencen a veure les “primary keys” i les “foreign keys”.

Primer vam començar amb la taula “User\_PW” on vam indicar el seu atribut “ID\_User” que és la “primary key” de la taula, també vam posar tots els atributs “not null” i amb el tipus de variable corresponent per cada atribut, també vam posar en l’atribut username que havia de ser unique.

La següent taula que és la “Enemie” no té “primary key” pròpia, ja que és una taula heretada, llavors la seva “primary key” és “ID\_User” que ve heretat de “user\_pw” i els atributs tots eren “not null” i amb el seu tipus de variable per cada atribut.

La taula “Planet” té la seva pròpia “primary key” que és “ID\_Planet” i a més també té una “foreign key” que prové de “User\_PW” aquesta és “ID\_user” això sorgeix a partir de la relació que hi ha entre aquestes dues taules i la necessitat d'indicar el planeta de quin usuari és. Els atributs a més de no ser “not null” i amb el seu tipus de variable també vam indicar que el “Planet\_Name” era “unique”.

La taula “Ship” té la seva propia “primary key” que és “ID\_Ship” i té totes les variables en “not null” menys “armor” que aquesta està com “null” perquè podria arribar a no tenir dades, les seves relacions amb “Planet” serà explicada en un altra part.

La taula “Defense” té la seva propia “primary key” que és “ID\_Defense” i té totes les variables en “not null” menys armor que està posada com “null” perquè podria arribar a no tenir dades.

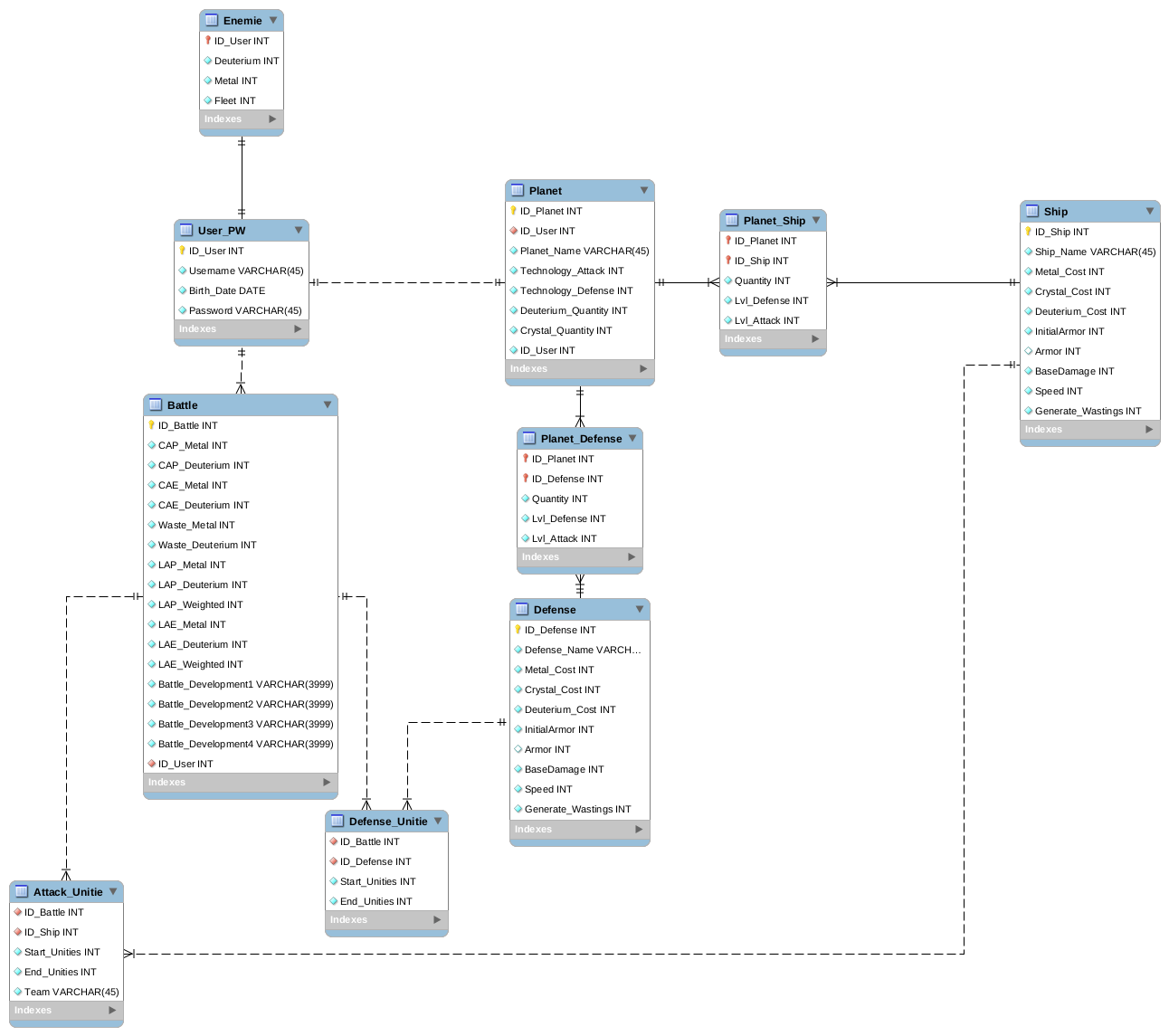
La taula “Defense” i “Ship” tenen una relació “N:N” amb “Planet” això el que provoca és que s’ha de crear una taula entre “Defense/Ship” i “Planet” en aquestes taules es passen com “primary key” i “foreign key” la “primary key” de cadascuna de les taules i a més hem posat 3 atributs que són necessaris per utilitzar-los en la nostra aplicació.

La taula “Battle” té la seva pròpia “primary key” que és “ID\_Battle” però a més té la “foreign key” que prové de “User\_PW” perquè necessitarem saber que una batalla està adjudicada a un usuari. En els atributs “battle\_development” pot arribar a confusió i és que hi han 4 perquè s’ha de guardar un “String/Varchar” que pot arribar a ser molt llarg i el que fem és dividir-lo en 4 perquè es pugui guardar sense problemes perquè la variable tipus “Varchar” només pot guardar fins a 3999 caràcters.

La taula “Attack\_Unitie” no té “primary key” perquè és una taula que hem de fer ús per guardar diverses vegades les tropes de les batalles i ha de guardar moltes vegades tropes que poden repetir-se i això faria que col·lapses la taula. Aquesta taula està relacionada amb “Battle” on passa la seva “primary key (ID\_Battle)” com “foreign key” a “Attack\_Unitie”.

També rep la “ID\_Ship” de la taula “ship” com “foreign key” aquestes relacions estan fetes perquè utilitzarem la taula per indicar que en una batalla específica comença amb nombre específic d’un tipus de tropa i acaba amb un altre número i això ens servirà per a després fer el report de manera correcta. Així mateix, està indicat l'atribut “team” perquè les tropes de “ship” les poden tenir tant el jugador com el “enemie”.

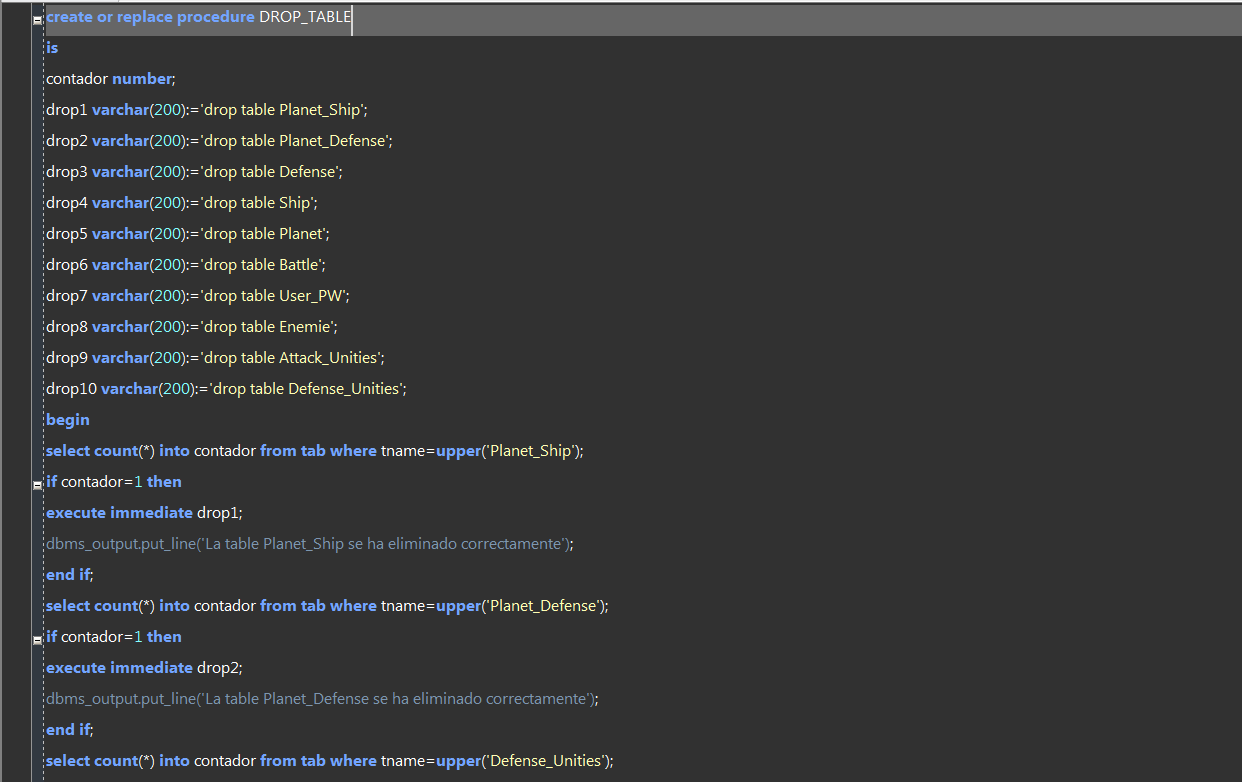
La taula “Defense\_Unitie” és molt semblant a la taula “Attack\_Unitie” el que canvia respecte a l’altra taula és que en comptes de rebre la “ID\_Ship” com “foreign key” aquesta rep la “ID\_Defense” com “foreign key” i també que no té l'atribut “team” ja que les tropes de “defense” només són fetes servir per l’usuari.



#### DROP TABLE

Aquest procedure consisteix a executar un seguit d'instruccions per eliminar totes les taules de la base de dades. Fem comprovacions per saber si existeix la taula, és a dir: fem un count() i fiquem el resultat dins d’una variable “contador”. Si “contador” és igual a 1, executem la instrucció d'esborrar aquesta taula, si “contador” és 0, passem a la següent comprovació i així fins a fer la comprovació de totes les taules.

Per a executar les instruccions, com és un procediment i no poder executar directament la funció “drop table”, les hem de ficar dins de variables i fer un “execute immediate” perquè funcioni correctament.

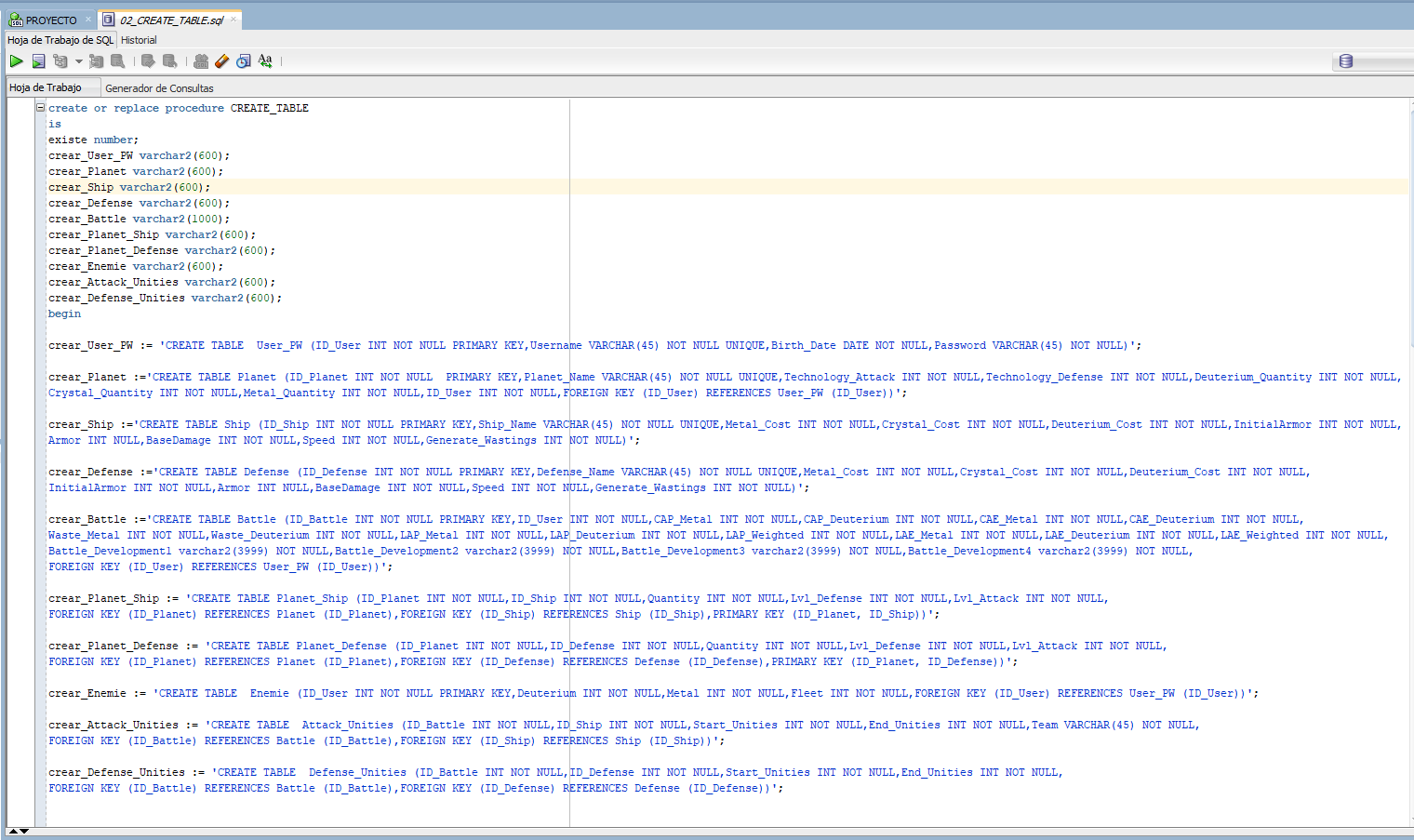




#### 

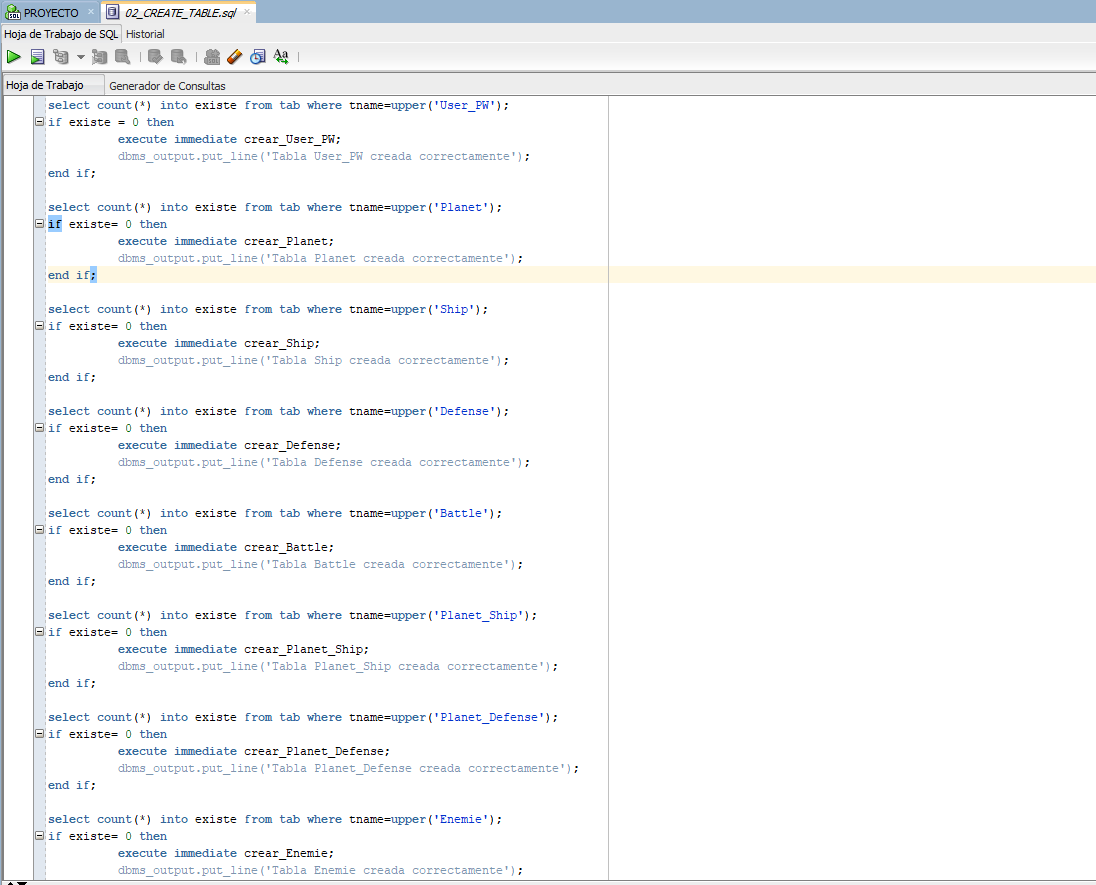
#### CREATE TABLE

Aquest “procedure” consisteix a executar un seguit d’instruccions per crear totes les taules de la base de dades. Primer comencem creant unes variables per introduir el text que hem d’utilitzar per crear cada taula, ja que si utilitzem el “create table” directament al “procedure” ens indica un error i l'haurem d'executar més tard amb una funció que té oracle que es el “execute immediate”.

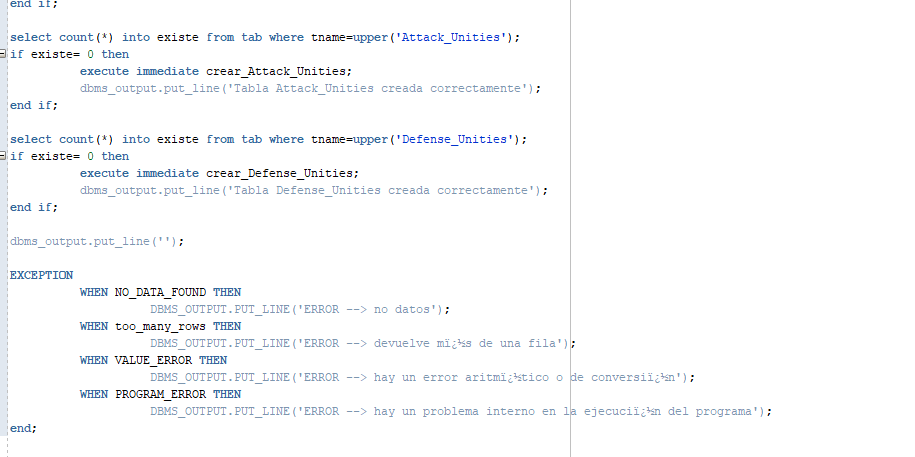


Després d'haver creat les variables amb el seu text per crear les base de dades fem unes comprovacions on executem unes consultes que el que fan és contar quantes vegades surt el nom de la taula indicada en la base de dades que tenim llavors si aquest “count” ens retorna un 0 significa que la taula no existeix i, per tant, entra al “if” que hem indicat i executa la funció “execute immediate” amb el nom de la variable que contindrà totes les dades per crear la taula de manera correcta.

Aquestes comprovacions es fan per controlar que si les taules estan creades no s’executi un altra vegada. Perquè si executar un “create table” d'una taula que ja existeix donarà error i no podrà continuar el programa.

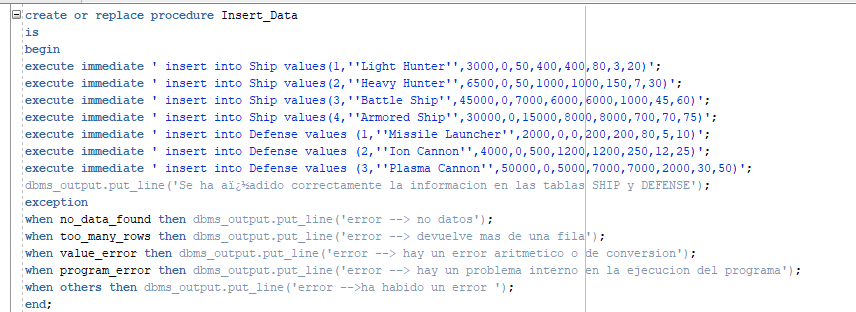


Després de fer aquestes comprovacions també posem en el nostre apartat d'excepcions les excepcions bàsiques perquè en el cas que doni un error bastant comú en la base de dades podem corregir-lo. Aquests errors són: una consulta no troba cap informació, una consulta retorna moltes files, error amb algun valor aritmètic o alguna conversió i per últim el programa té un problema intern en l’execució.



#### INSERT DATA

En aquest “procedure” busquem introduir les dades principals a les taules “ship” i “defense”, per fer això utilitzem la funció “execute immediate” perquè el “procedure” es pugui crear sense que doni errors encara que no existeix la base de dades. També hem inserit els errors comuns. Que hem indicat anteriorment.



#### 

#### INITIALIZE

En aquest “procedure” el que farem serà executar els “procedures” de “drop table”, “create table” i insert data en el cas que sigui la primera vegada que s’inicia l'aplicació o en el cas que en Java es pressioni un botó que reiniciaria tota la base de dades. En aquest “procedure” rebem una variable que l’hem anomenat “reset\_buton”. Per això vam crear consultes per contar les taules que existien en la base de dades i així poder comprovar si era la primera vegada que s’iniciava l'aplicació. Per això vam usar una variable contado”.



Després de comprovar si la base de dades és la primera vegada que es comença entra al “if” en el cas que el “contador” sigui diferent de 10 que són totes les taules que hi ha a la base de dades. També entrarà en el cas que en Java es pressioni el botó de “reset” que llavors això retornarà al procediment una variable en “true” perquè així si la nostra variable és “true” entri al “if” independentment del que s'ha contat a les nostres comprovacions. Si entra en aquest “if” s'executa primer el “procedure” de “drop”, després el de “create” i per últim el d'insert data. En el cas que no entri en cap moment al “if” s'executa una excepció on indicarà que la base de dades ja existeix. A més també tenim les excepcions comunes d'oracle.



#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### REGISTRO

Aquest “procedure” el que farà serà registrar un usuari, és a dir, demanar les dades a l'usuari en l’aplicació Java i passar-les a la base de dades per registrar-lo. Per fer això en aquest procediment tenim 3 variables d’entrada i una variable de sortida. Les variables d’entrada son “usu” on estarà el “username”, “pass” que serà la contrasenya, “cumple” que serà la seva data de naixement. La variable de sortida serà “boleano” que és una variable que conté un 0 o 1 depenent de si l'usuari s’ha introduït o no.

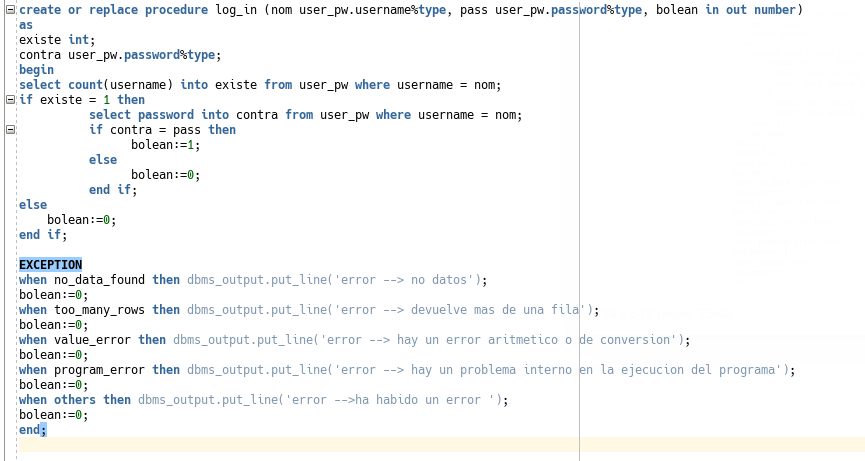
Aquí fem les comprovacions amb una consulta utilitzant el “count” per a comprovar si existeix l'usuari. En el cas que existeixi l'usuari es ficaria en una excepció, que es diu “existe”, on informaria que existeix l'usuari ja i a més donaria a la variable “boleano” el valor 0 perquè no s’hauria completat el procés. En el cas que no existís entraria en el “else” on trauríem el major “id” de la taula “user” per a poder introduir l'usuari amb un “id” diferent dels que existeix. També comprovem que si és el primer usuari a registrar-se li donem el primer valor. Després d’això inserim l'usuari a la respectiva taula amb les seves dades i també introduïm un “enemie” a la taula “enemie” que tindrà la “id” de l‘usuari que s’acaba de crear. Això es fa per adjudicar un “enemie” a un usuari ràpidament. Si el procés a sortir bé la variable “boleano” en tindrà 1.

Per últim, controlem les excepcions bàsiques on a més d’ensenyar l’error que és retornar-me la variable “boleano” en 0 perquè el procés no haurà sortit bé.



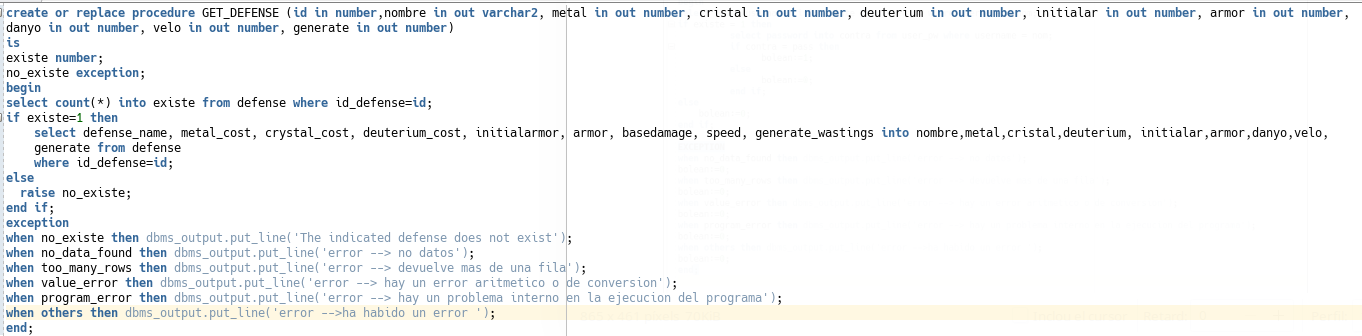
#### LOGIN

Aquest “procedure” rebrà les dades de l’aplicació Java que les utilitzarà per saber si pot entrar a jugar o no per això tendra que iniciar amb un usuari que ja existeixi. Rebrem el “username” i la contrasenya per a saber si podem entrar. Fem un “count” del “username” a la base de dades i en el cas que existeixi entraria i comprovaria si la contrasenya és igual a la que té l’usuari. Si tot coincideix el “procedure” l’hi donarà a la variable “bolean” el valor 1 i, per tant, en Java se sabrà que pot entrar. En el cas que en algun moment no coincideixi alguna cosa o es produeixi algun error la variable serà 0 i en Java no deixarà entrar i donarà com a invàlid el “username” i/o el “password”.



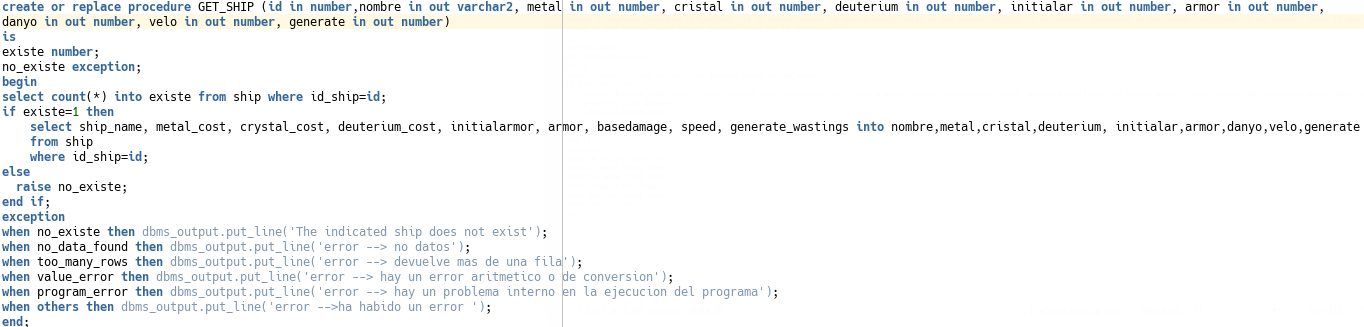
#### GET\_DEFENSE

Aquest “procedure” el que fa és demanar en Java un “id” que serà el “id” de la defensa i amb aquest “id” buscar quina és la defensa que coincideix i així passar-li a les variables de sortida les dades que rebran en l’aplicació.



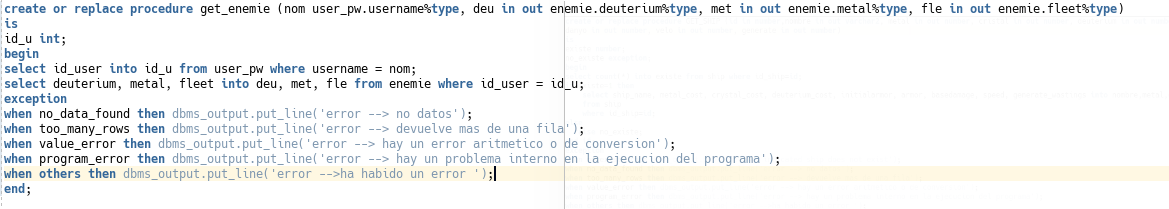
#### GET\_SHIP

Aquest “procedure” fa el mateix que el “get\_defense” explicat anteriorment, però amb la diferència de què en comptes de la “id” ser de “defense” serà de “ship” i haurà de buscar en la taula “ship” les respectives dades de la tropa.



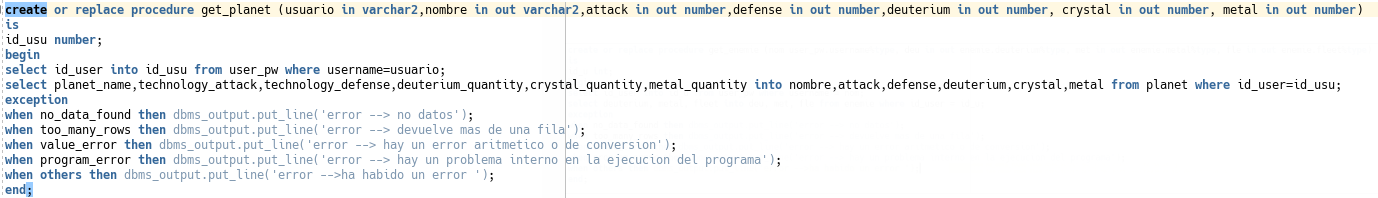
#### GET\_ENEMIE

El que fem en aquest procediment és demanar un usuari per poder comprovar quin enemic té vinculat per això treure'm la “id” del usuari amb el nom, després de treure la “id” de l'usuari ja podrem treure la informació del “enemie” que té l’usuari i es passaran a les variables de sortida.



#### GET\_PLANET

El que fa aquest procediment és demanar el nom de l'usuari per poder treure la “id” que l’hi correspon a l'usuari i així poder treure la informació del planeta que té adjudicat l'usuari. La informació del planeta és retornada en les variables de sortida.



#### 

#### GET\_BATTLE

Aquí el que fem és demanar el nom de l’usuari per poder treure la “id” de l’usuari i així poder veure les batalles que a jugat un usuari. També demanem la batalla que vol veure, perquè en aquest procediment ja ordenem tot directament i d’aquesta manera l’usuari pot veure exactament la batalla que demani. Totes les variables seran utilitzades per mostrar un report a Java.



#### 

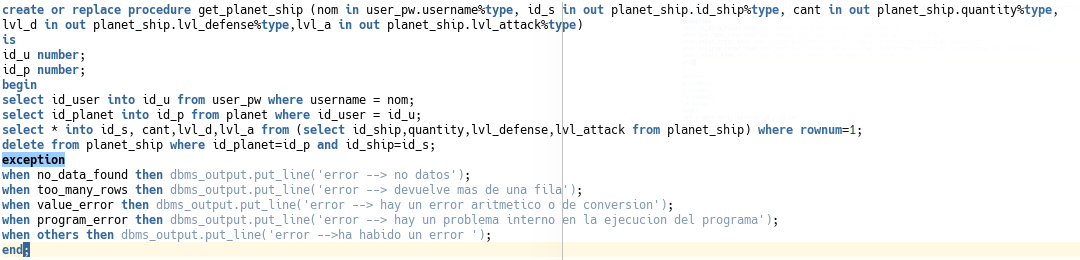
#### GET\_PLANET\_DEFENSE

En aquest procediment demanem també l'usuari per treure la “id” d’aquest i d’aquesta manera podrem anar traient la informació de la taula “planet\_defense” d’un en un i també anirem eliminant el que retornem d’aquesta manera retornarem la informació necessària i a més anirem deixant buida la taula perquè a la següent batalla podem inserir noves dades de nou.

#### 

#### GET\_PLANET\_SHIP

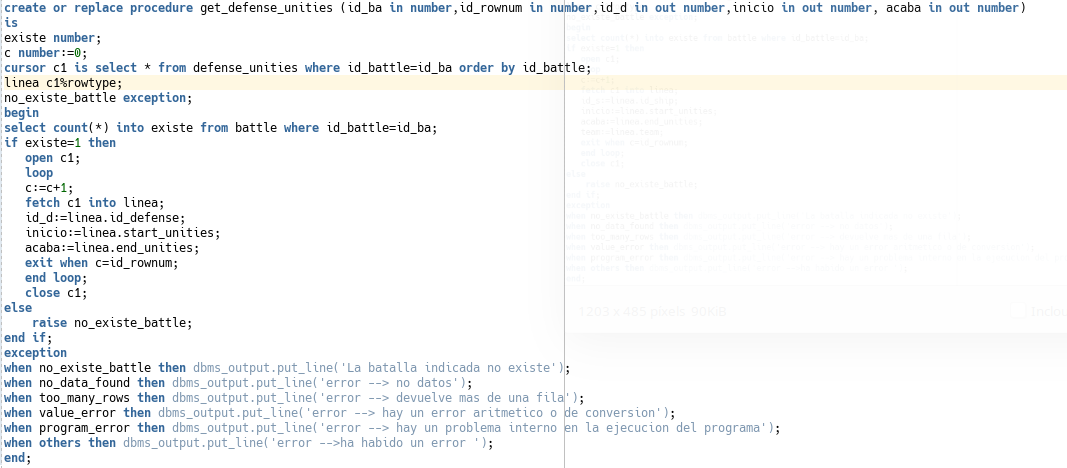
En aquest procediment demanem també l'usuari per treure la “id” d’aquest i d’aquesta manera podrem anar traient la informació de la taula “planet\_ship” d’un en un i també anirem eliminant el que retornem d’aquesta manera retornarem la informació necessària i a més anirem deixant buida la taula perquè a la següent batalla podem inserir noves dades de nou.



#### 

#### GET\_DEFENSE\_UNITIES

Aquest procediment rep un “id” de la batalla amb el que es podrà buscar les tropes que hi ha en una batalla i també la fila que ens interessa buscar per a poder anar buscant totes les tropes d'una en una i així poder agafar les dades ordenadament i retornar-les en les variables de sortida.



#### GET\_ATTACK\_UNITIES

Aquest procediment fa la mateixa funció que el “get\_defense\_unities” amb la diferència que en aquest cas las tropes es busquen a “attack\_unities” en comptes de “defense\_unities”.

#### 

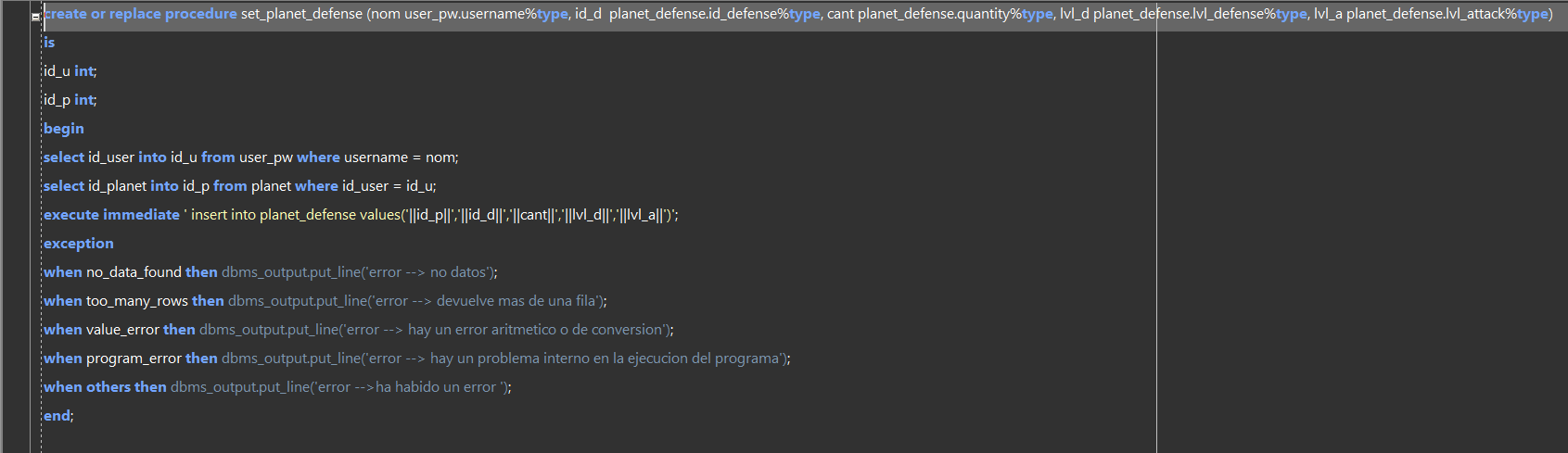
#### SET PLANET SHIP

A aquest procediment consisteix a introduir noves dades a la taula “PLANET\_SHIP”, per tant, li hem de passar un “id\_planet”, un “id\_ship”, la quantitat de naus (“quantity”), i els seus nivells d’atac i defensa (“lvl\_attack”, “lvl\_defense”). Com al joc no sabem quin és l’”id\_planet” del planeta del jugador (ja que només consta a la base de dades), el procediment demana el nom d’usuari (que es queda emmagatzemat a una variable quan el jugador inicia sessió), pel fet que a partir del nom podem aconseguir l’”id\_user” i amb l’”id\_user” podem aconseguir l’”id\_planet” (això és possible perquè cada jugador només té assignat un planeta). La resta de dades les passa directament el joc i el procediment simplement fa un “execute immediate” per afegir-les a la taula “PLANET\_SHIP”.



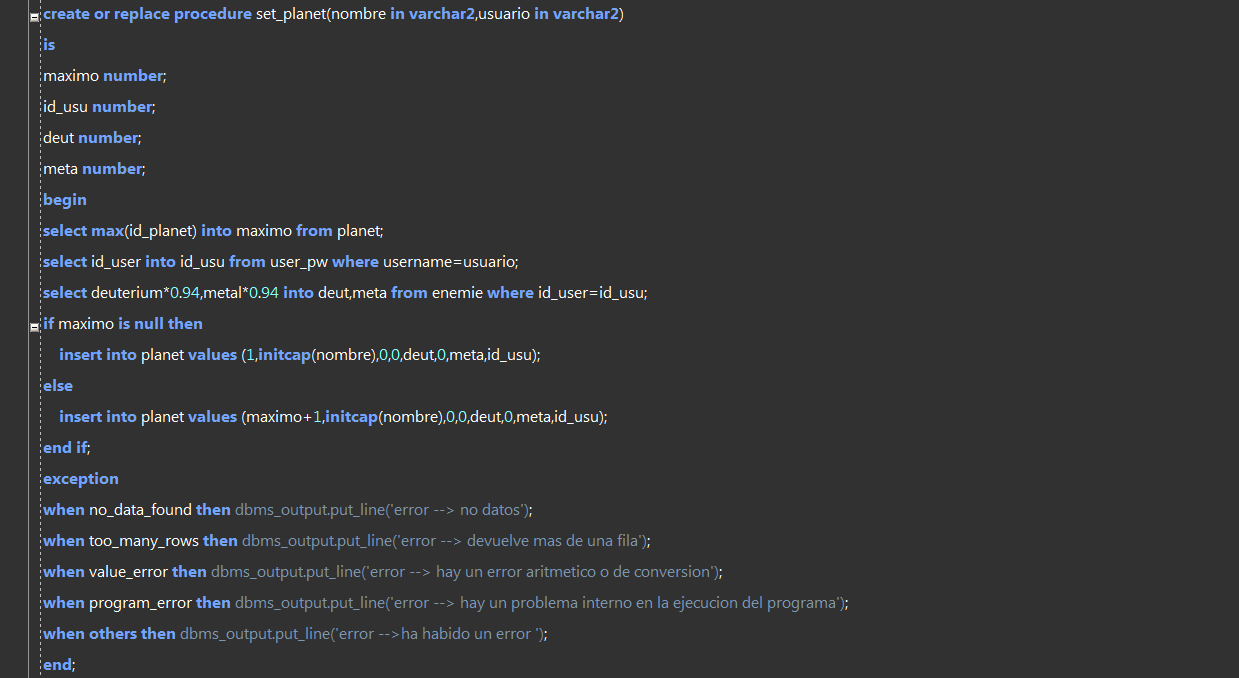
#### SET PLANET DEFENSE

Aquest procediment fa exactament el mateix que el **set planet ship** amb la diferència que en lloc d’inserir les dades a la taula “PLANET\_SHIP”, les insereix a la taula “PLANET\_DEFENSE”. També s’ha de tindre en compte que li passarem “id\_defense” en lloc de “id\_ship”.



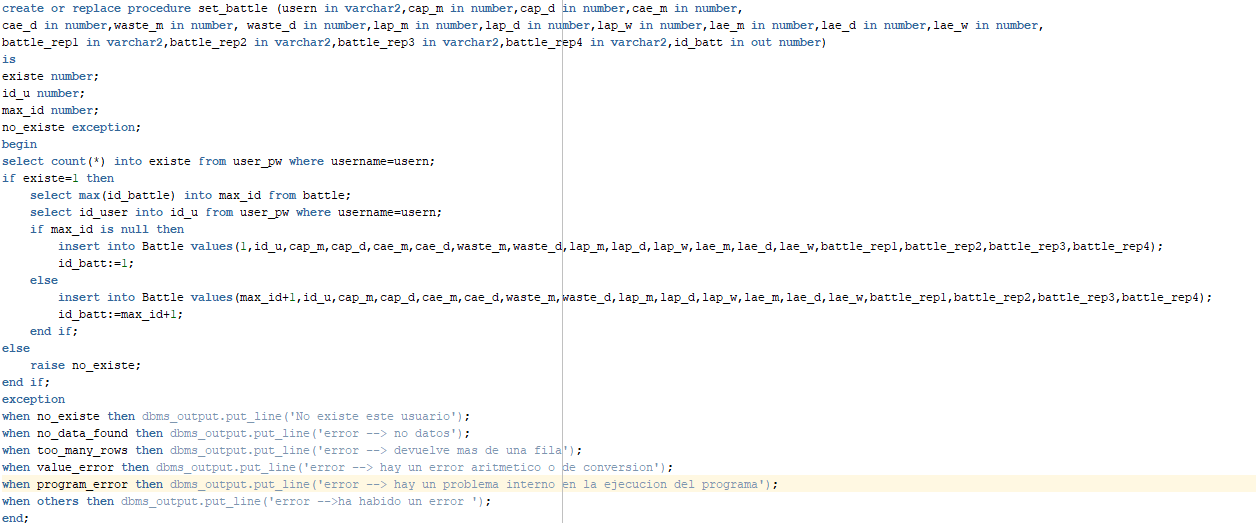
#### SET PLANET

Aquest procediment s’executa a l'hora de crear un planeta per a l’usuari. Li introduïm el nom del planeta que passa l’usuari i el nom d’usuari. Per assignar un “id\_planet” buscarem a la taula “PLANET” quin és l’id màxim i li sumarem 1, en cas que no hi hagi cap planeta creat li assignarem l’id 1. Trèiem l’”id\_user” a partir del nom d’usuari com hem fet a altres procediments. El deuterium i el metall del planeta s’extreu a partir del que té l’enemic (l’usuari té un 94% dels recursos que té l’enemic), és a dir, fem un select a la taula “ENEMIE” per saber quant metall i deuterium té i multipliquem aquest valor per 0,94. El resultat d’aquesta multiplicació s’emmagatzema dins de variables per despres fer l’insert. Un cop ja tenim totes les dades inicials necessàries les afegim a la taula “PLANET” amb un insert. El “technology\_attack”, el “technology\_defense” i el cristall s’afegeixen amb un 0, ja que en crear el planeta les tecnologies són de nivell 0 i comences sense cristall.



#### SET BATTLE

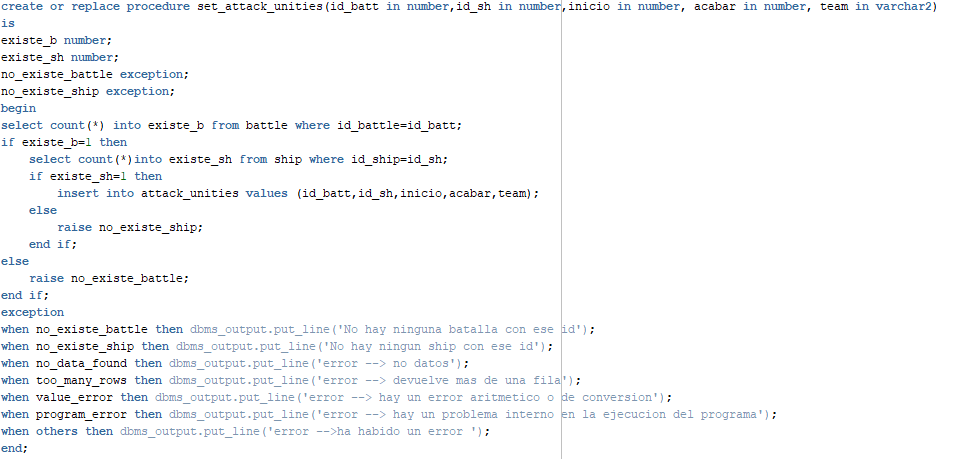
Aquest procediment consisteix a afegir a la taula “BATTLE” tot el registre d’una batalla. Des de l’aplicació de Java es cridarà a aquest procediment passant-li el nom d’usuari i tota la informació del registre de batalla, i el programa esperarà una variable de sortida que és l’”id\_battle” (hem de tornar l’”id\_battle” perquè la necessiten per executar els procediments “set\_defense\_unites” i “set\_attack\_unites”). Primer de tot comprovem si el nom d’usuari introduït existeix, si existeix fem un “select max(id\_battle)” per saber quin és l’id de batalla més gran i sumar-li 1. En cas que no existís el nom d’usuari saltaria una excepció, i si encara no existeix cap id\_battle li donem el valor d’id 1. Un cop ho tenim pot a punt, inserim totes les dades a la taula “BATTLE” i retornem l’id\_battle.



#### 

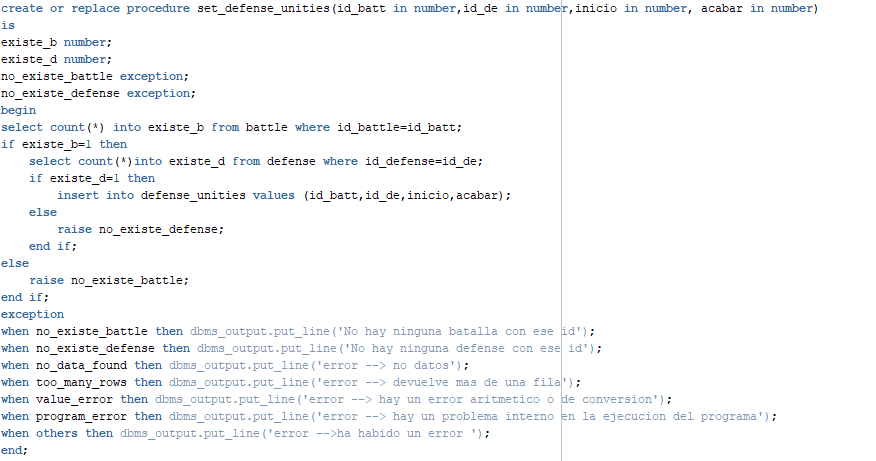
#### SET ATTACK UNITIES

A aquest procediment li passem un id\_battle (que l’aconseguim amb el procediment “set\_battle”), un id\_ship, la quantitat de tropes en començar la batalla, la quantitat de tropes en acabar la batalla i l’equip al qual pretenen (planeta o enemic). Primerament, comprovem si existeix l'”id\_battle” i l’”id\_ship”, si no existeixen saltaran excepcions. En cas que existeixin, s’afegiran les dades a la taula “ATTACK\_UNITIES”.



#### SET DEFENSE UNITIES

Aquest procediment es casi igual a l’anterior amb la diferència que no necessita que li passem l’equip (ja que l’equip enemic només ataca) i que en lloc de passar-li l’”id\_ship” li passem l’”id\_defense”.



#### UPDATE\_PLANET

En aquest procediment el que es fa és passar-li el nom del “username”, el “technology attack”, el “technology defense”, el “deuterium” i el “metal”. Amb el “username” el que fem és treure el “ID\_user” per a poder veure quin planeta està relacionat amb l'usuari i així ja podrem passar a l’actualització de les dades de “Planet”.

#### 

#### UPDATE\_ENEMIE

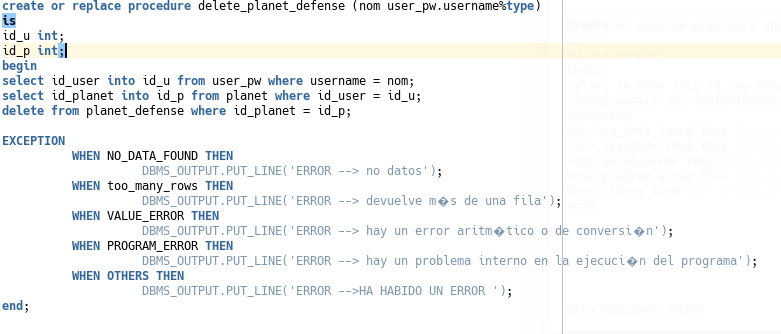
En aquest procediment rebrem una variable que en aquest cas és el “username” que en serà útil per poder treure el “ID\_User” que l’utilitzarem per trobar l'enemic relacionat amb l’usuari i així l’actualitzarem. L'enemic a l'haver d’actualitzar-se en cada batalla trucaria a aquest procediment per a poder sumar-li un 6% als materials que ja tenia.

#### 

#### 

#### DELETE\_PLANET\_DEFENSE

Aquest procediment s’executaria al moment en què un usuari acabi una partida perquè la taula “planet\_defense” quedi completament buida, perquè si un altre usuari vol jugar serà necessari que tingui aquesta taula buida per a poder introduir temporalment les tropes d'aquella batalla.



#### DELETE\_PLANET\_SHIP

Aquest procediment s’executaria al moment en què un usuari acabi una partida perquè la taula “planet\_ship” quedi completament buida, perquè si un altre usuari vol jugar serà necessari que tingui aquesta taula buida per a poder introduir temporalment les tropes d'aquella batalla.

#### 

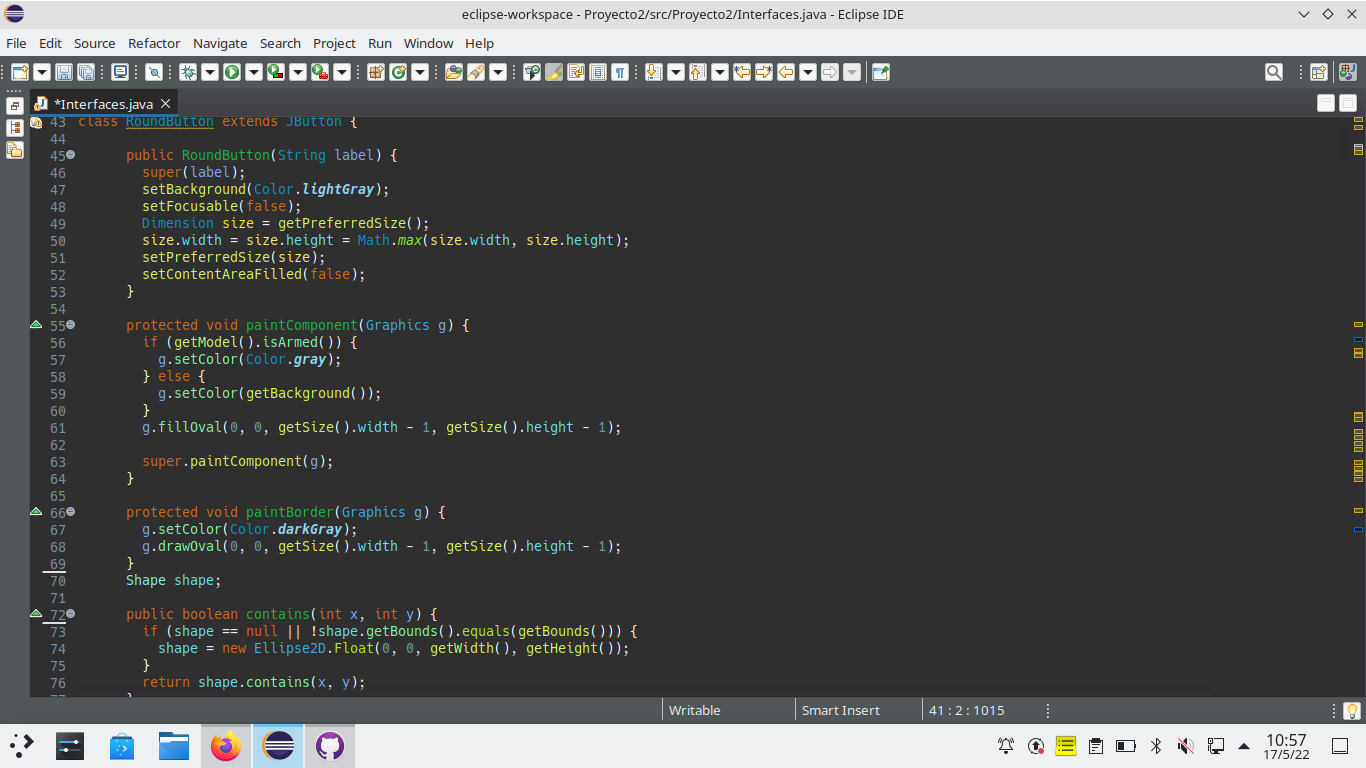
#### 

### PROGRAMACIÓ

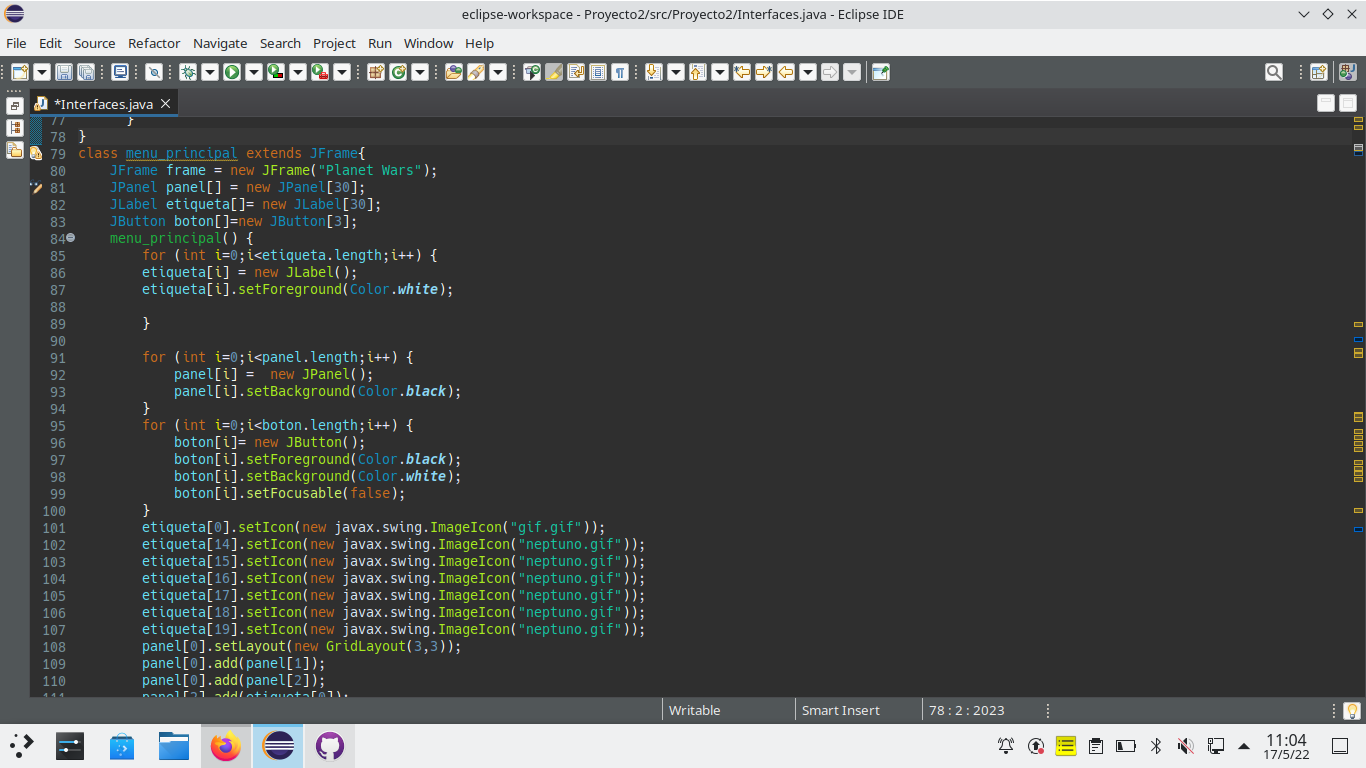
Classe Interficies:

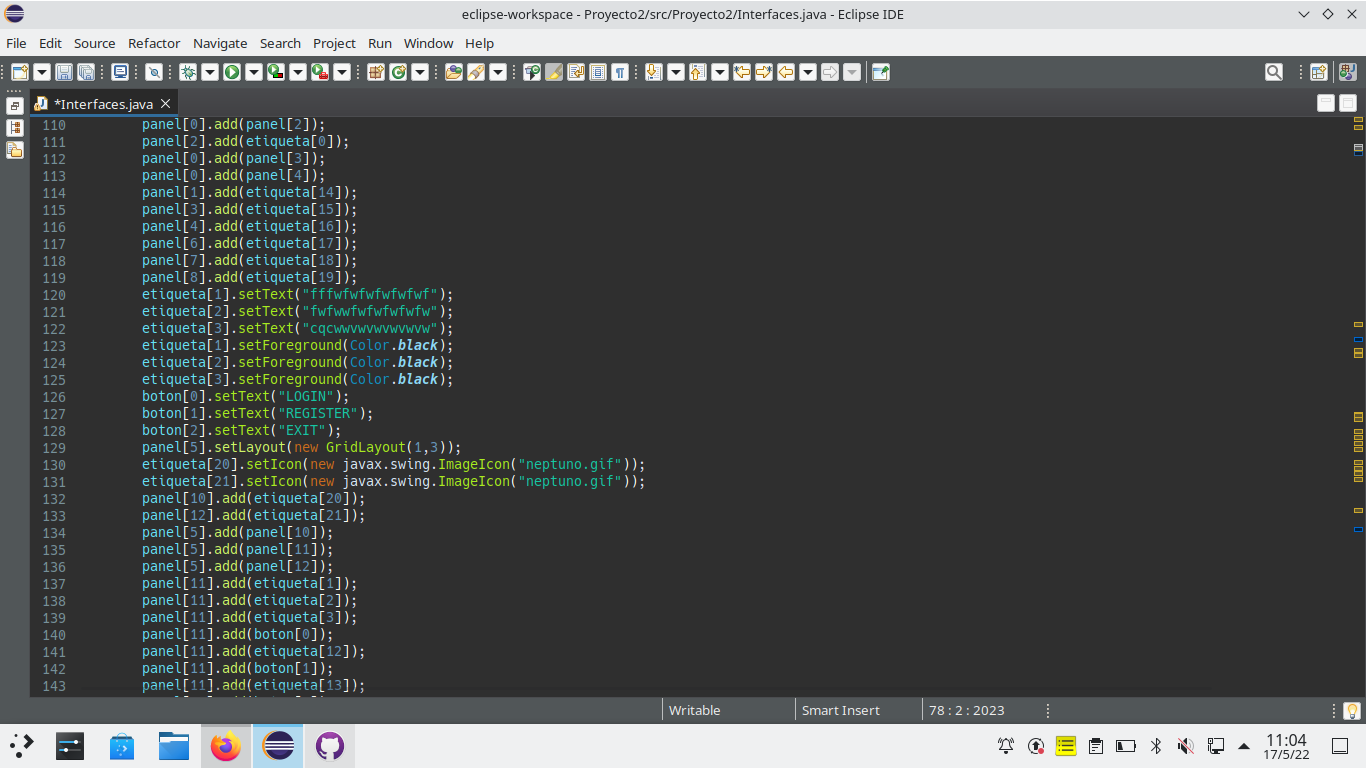
Hem creat la clase interfícies, on hem creat les diferents classes que contenen les interfícies que utilitzarem durant el projecte, també conté una classe per modelar botons.

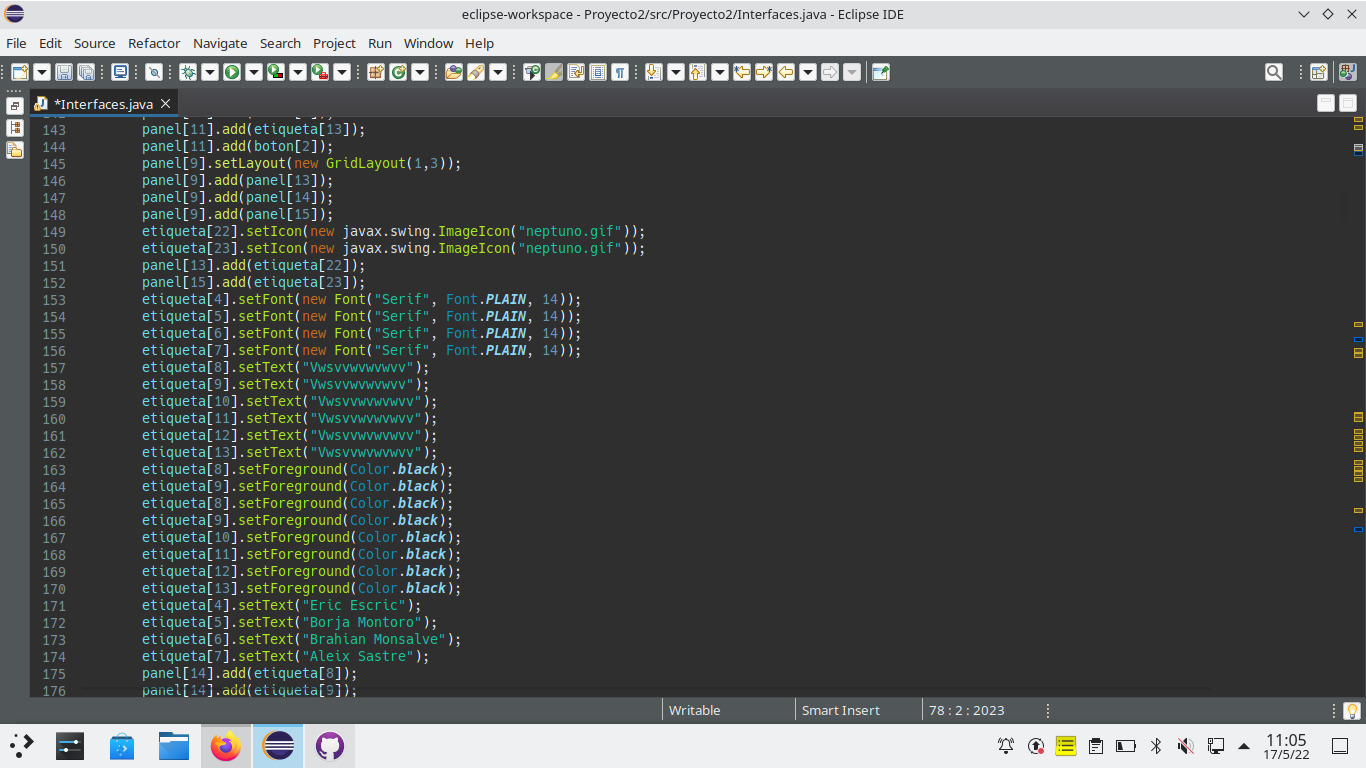
Classe RoundButton: classe que serveix per a fer els botons rodons, la qual l’hi has de passar el nom del botó i et crea el botó redó amb el nom que l’hi passes.

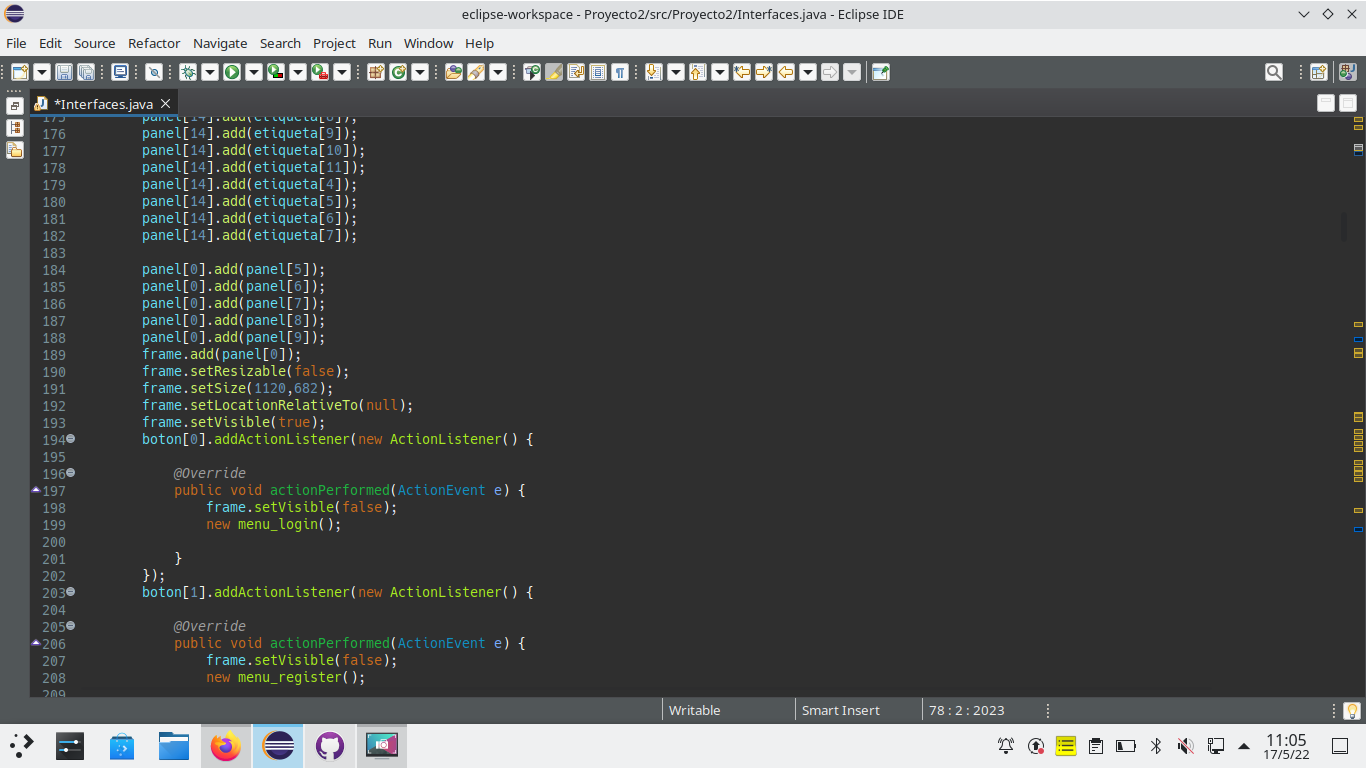


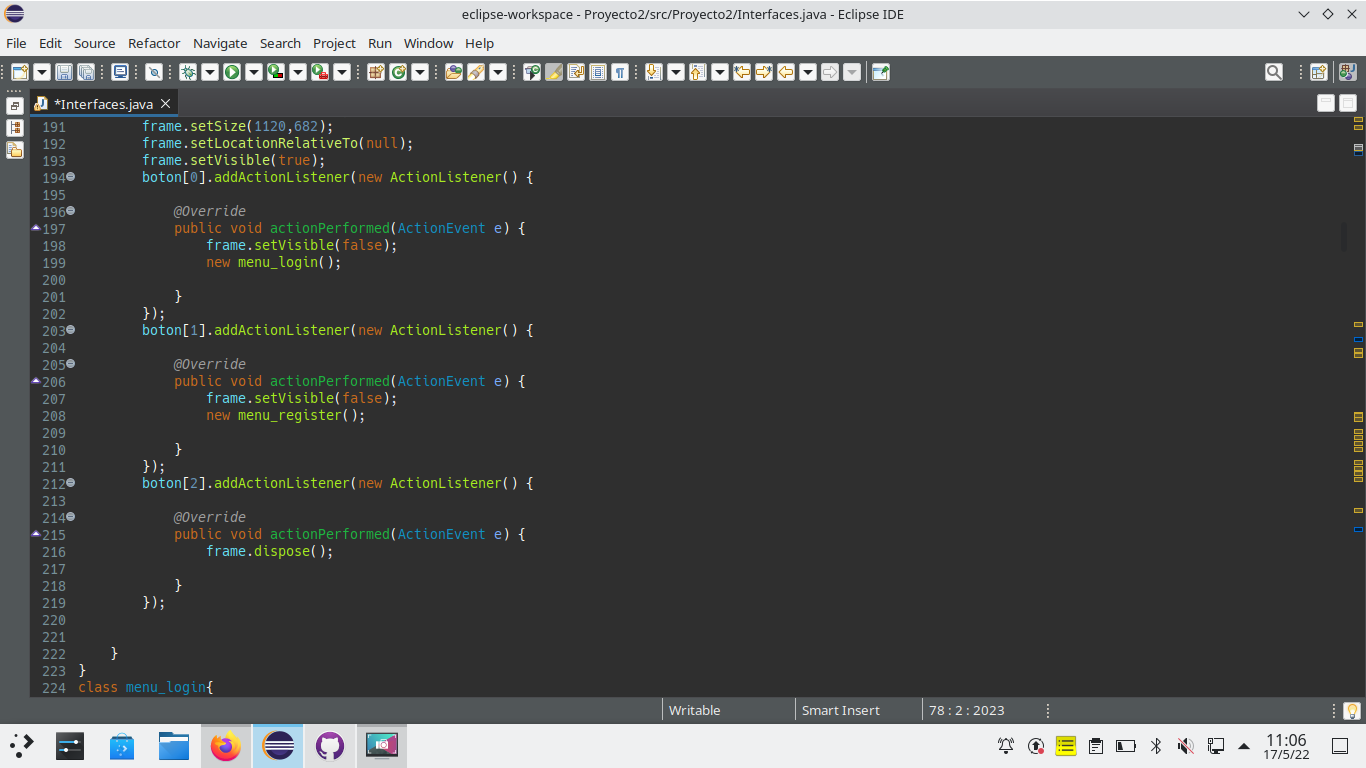
Classe menu\_principal: En aquesta clase com el seu nom indica hem creat la pàgina que surt al començar el programa la qual té tres botons un per entrar al teu usuari, un altre per registrar un nou usuari, i per últim el botó de sortir.

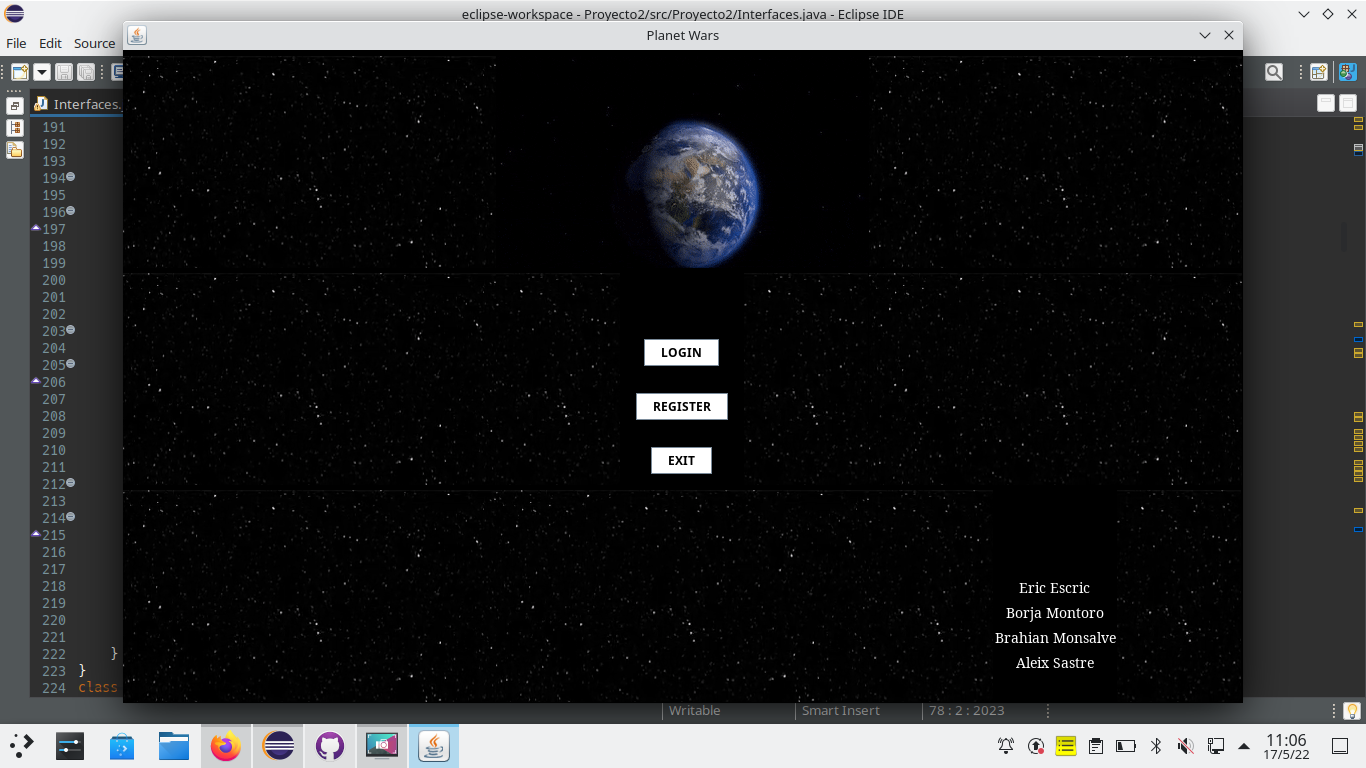




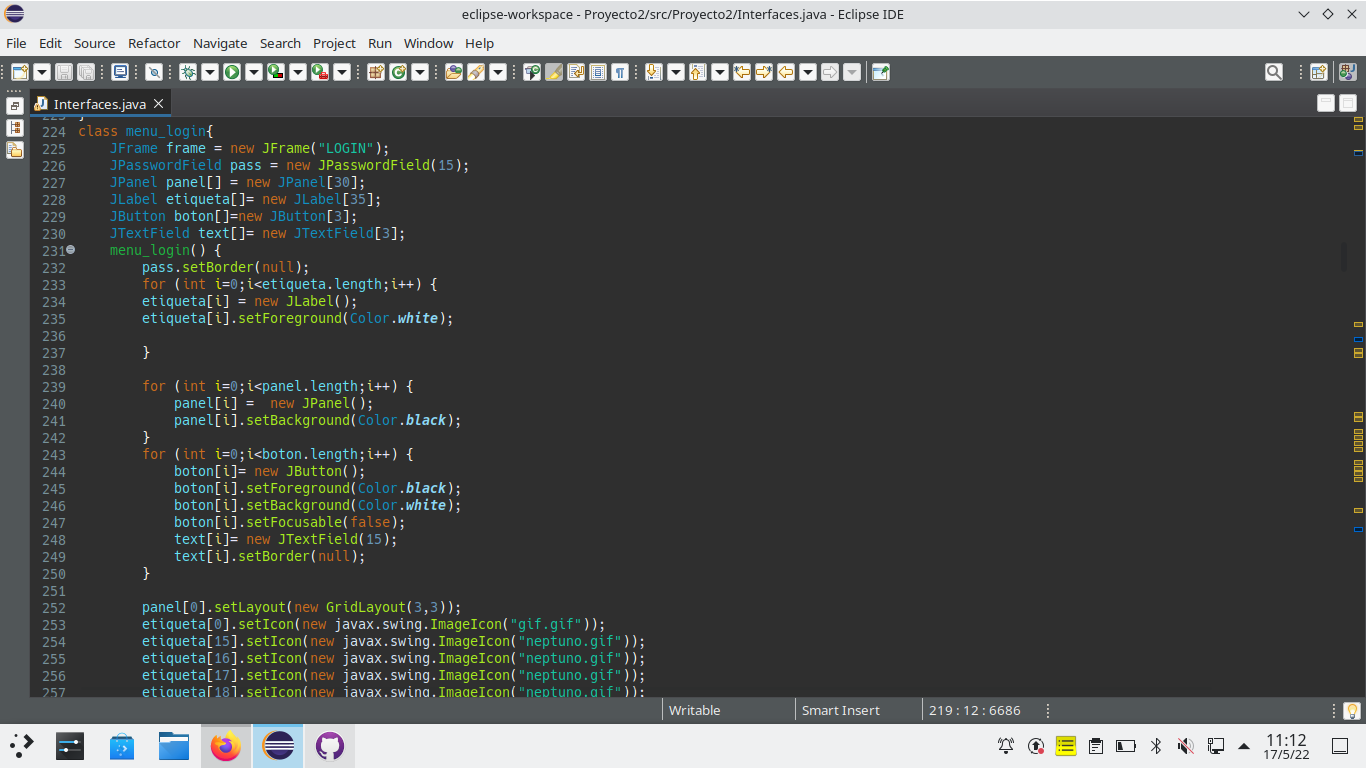


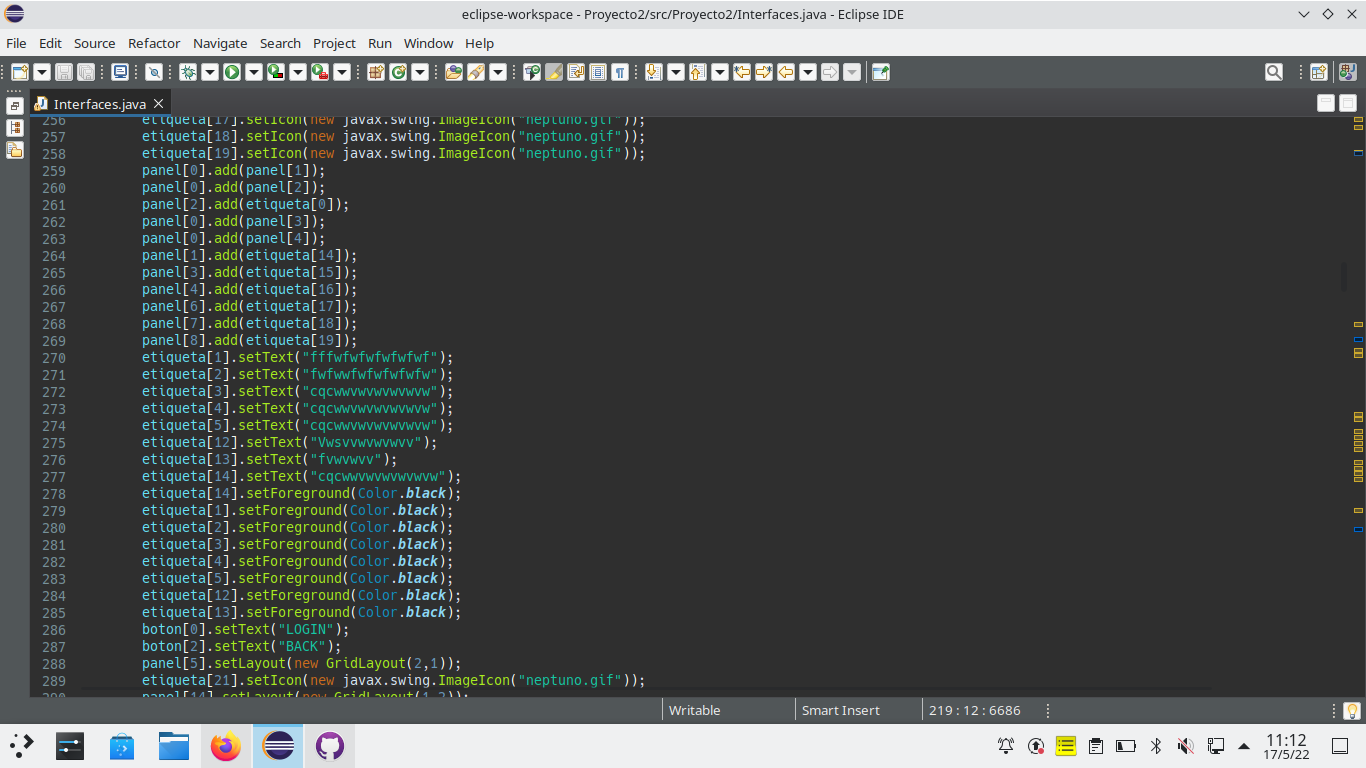


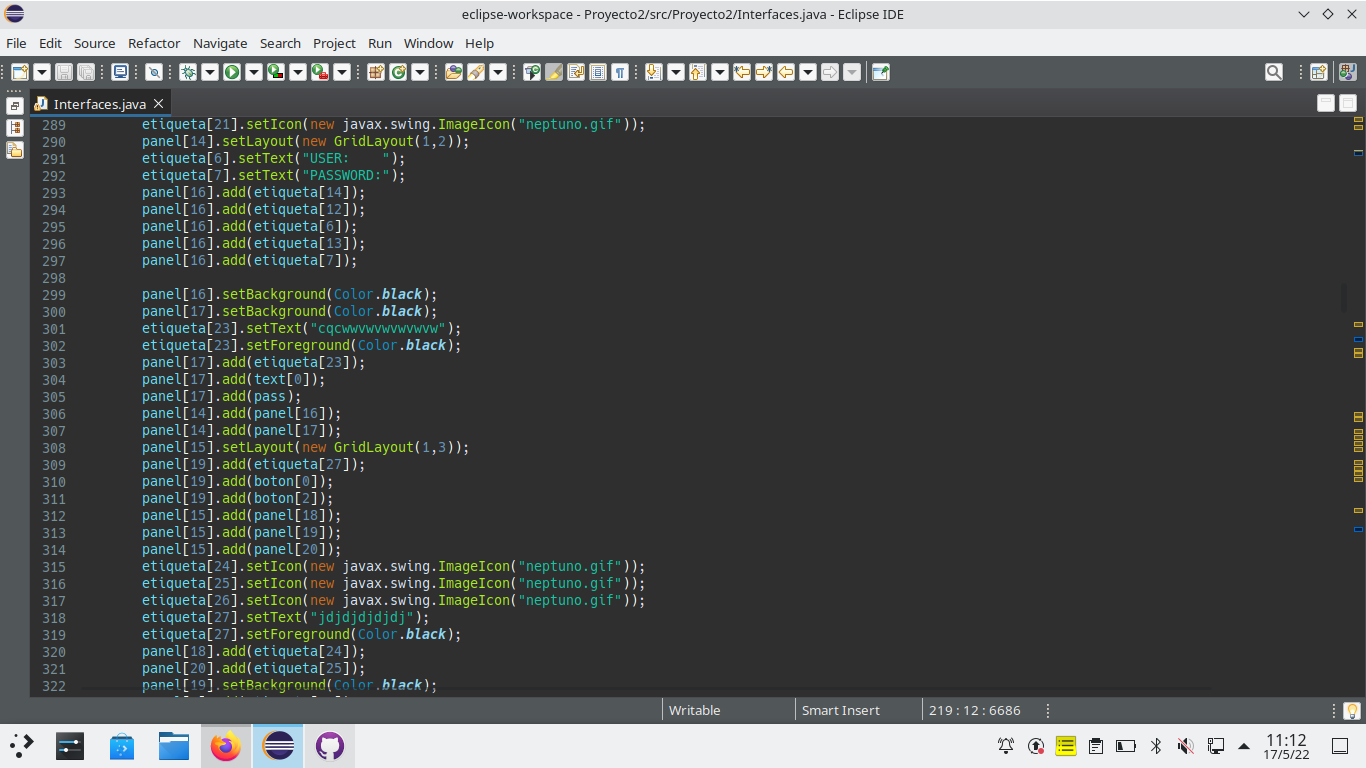


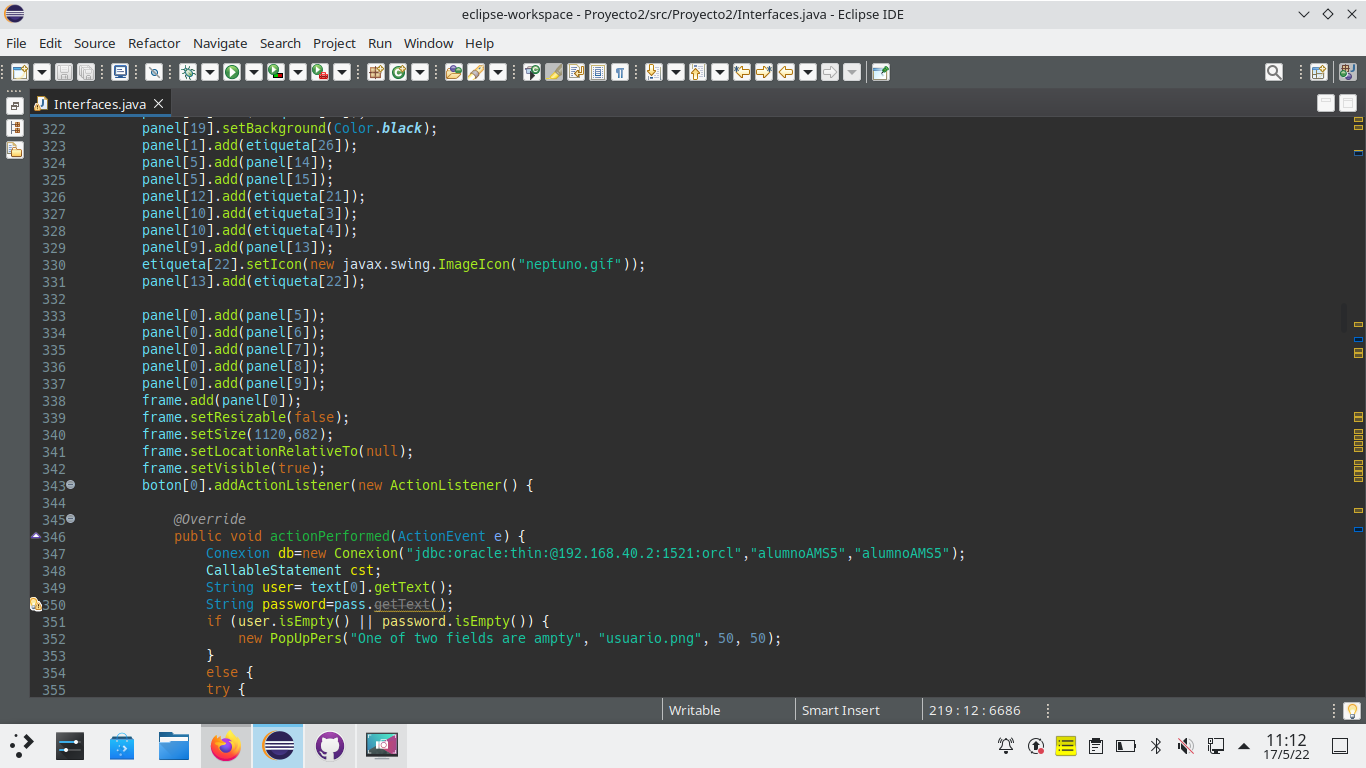


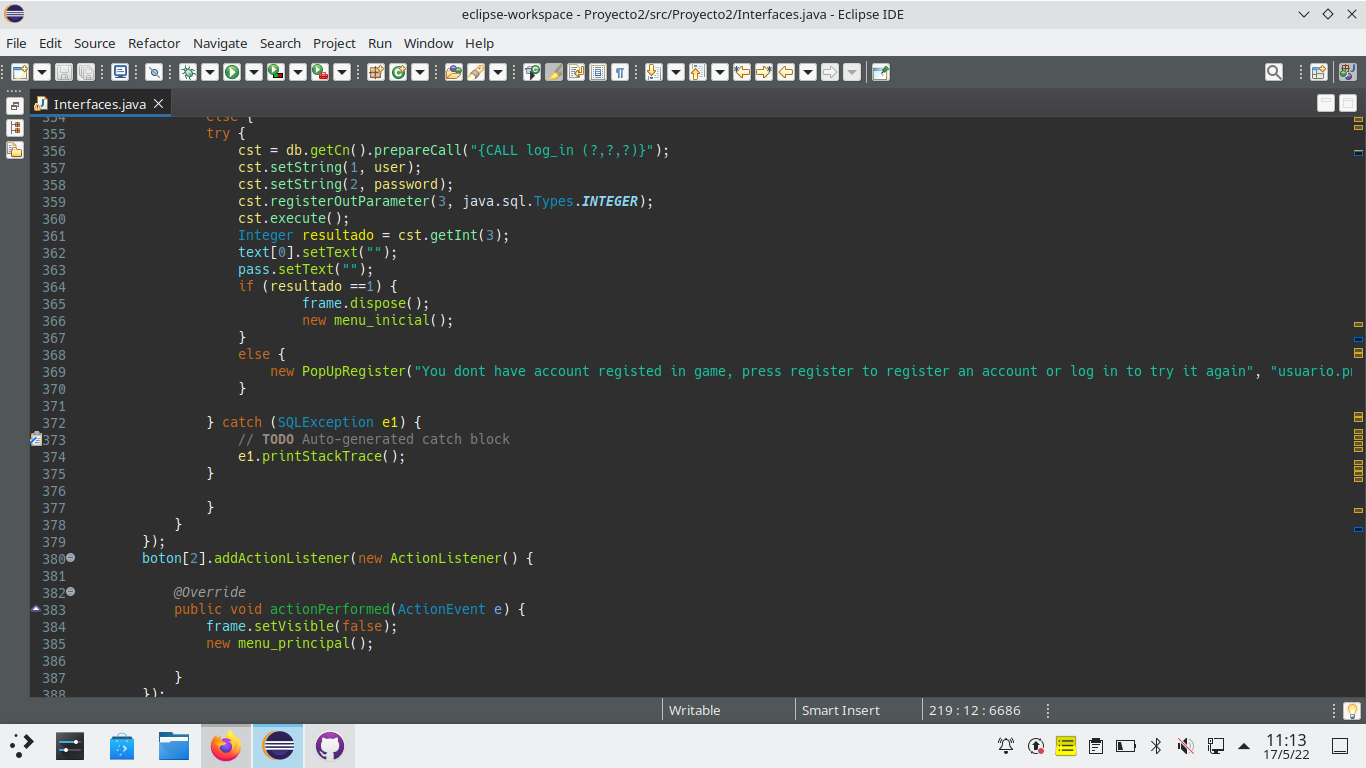
Classe menu\_login: A aquesta classe entra quan l'usuari pitja el botó login, aquesta interfície està dissenyada per als usuaris ja creats, que hauran d'introduir el seu nom d'usuari i la seva contrasenya, conté camps que s'han d'omplir per poder jugar, també conté dos botons un per enviar les dades que posen als camps vuits, i l'altre per sortir al menú principal, on en deixarà tornar a login, register o sortir directament.

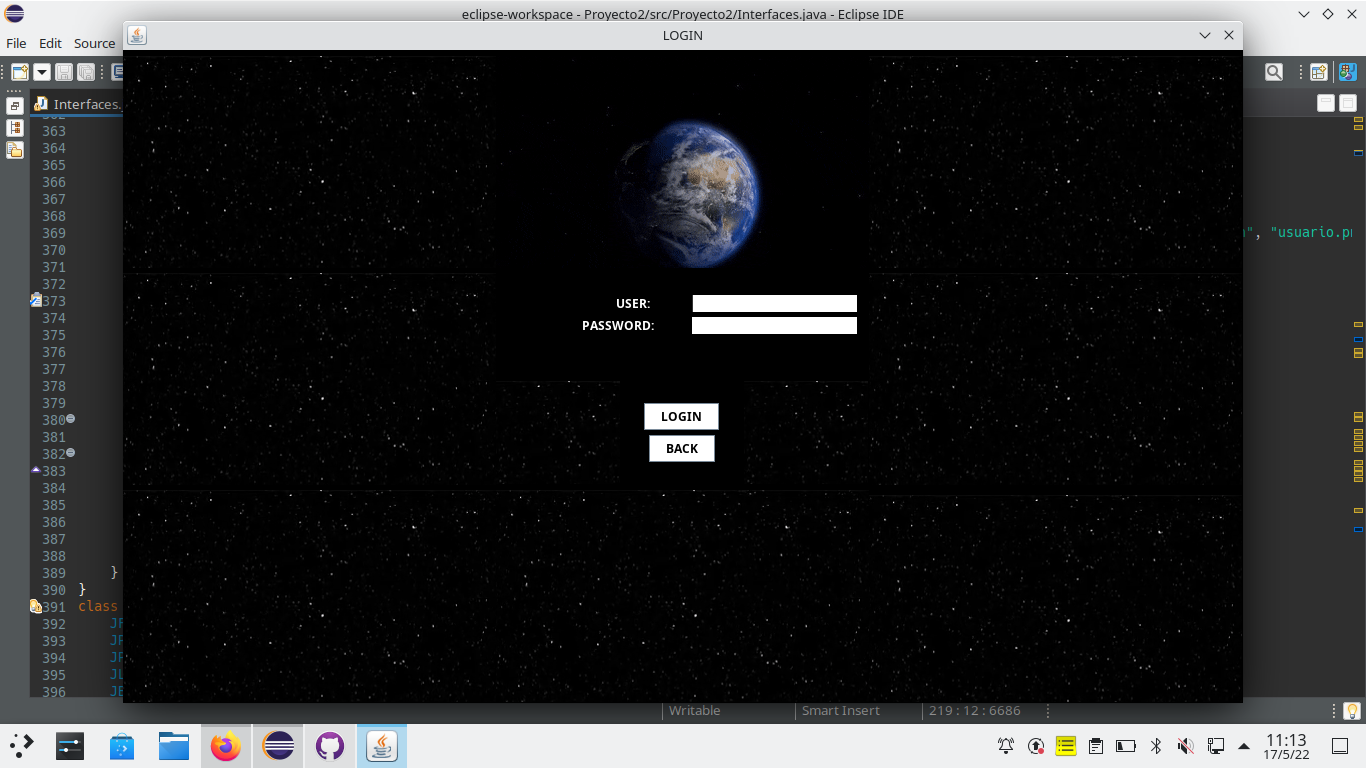












### ENTORNS DE DESENVOLUPAMENT

#### GITHUB

Per començar amb el projecte vam crear un repositori a "GitHub" on vam posar nos tots com a col·laboradors. Després vam començar a crear les "branches" individuals per a cadascun dels que som al grup i així anar pujant com anàvem progressant en el projecte. En finalitzar el projecte i després de fer molts canvis vam pujar tots els canvis al "main" i ja haurem acabat l'apartat de "GitHub".

#### DIAGRAMA CASOS D'ÚS

#### DIAGRAMA D’ACTIVITATS

Nosaltres vam decidir que en el diagrama d'activitats explicaríem la part del "login".

En començar l'aplicació com indiquem entraria a la primera pantalla que seria la d'inici on hi ha 3 botons que són: el botó de "log in", botó de "register" i el botó de "exit".

Si premem el botó de "exit" sortirem de l'aplicació i és tancada definitivament. Si premem el botó "register" li demanarem un "username", un "password" i la seva data de naixement, enviaria les dades a la base de dades perquè es guarden les dades i llavor la base de dades comprovar si ja existeix l'usuari indicat. En el cas que existís retornaria "false" i llavors ens indicaria que haurà d'introduir unes altres dades perquè aquestes ja existeixen. Si no existís es crearia i ja entraria a l'aplicació.

En cas que l'hi prems el botó de "login", demanaria el "username" i la contrasenya llavors anirà a la base de dades i comprovarà si les dades són correctes amb les dades que ja hi han creades si existeix entraria a l'aplicació. Si no existís aniria al menú de registrar en el cas que premi el botó de registrar en el pop up que sortiria o tornaria a demanar les dades si prems el botó de tornar.

### 

## GLOSSARI

| Paraula/Expressió | Definició |
| --- | --- |
| count() | És una funció que hem utilitzat en base de dades que serveix però contar el nombre de files que té￼￼ una columna. També es pot posar una condició. |
| BDD | Són sigles que indican Base de Dades |
| Diagrama de CHEN | Tipus de diagrama on s'usen diverses figures i serveix per estructurar la nostra base de dades abans d'iniciar la construcció d'aquesta. |
| Heredada | És una entitat/taula que la seva existència depèn d’una altra taula. |
| N:N | Es fa servir en les relacions del model de CHEN i això indica que les taules tenen una relació de molt a molt. Per exemple si les taules fossin departament i empleat una relació N:N seria que en un departament poden contractar molts empleats i que un empleat pot estar contractat en diversos departaments. |
| Primary Key | És l’atribut que indica en una taula el seu valor identificatiu que no es podrà repetir en tota la taula. |
| Foreign Key | És l’atribut que proveeix d'una altra taula, aquest atribut en l’altra taula és una primary key. |
| Not Null/Null | L’atribut indicat si és not null significa que aquest camp no podrà quedar buit, si és null podrà quedar-se sense introduir dades. |
| Unique | És quan a un atribut l’indiquem que només poden contenir una vegada una dada, no es podrà repetir en tota la taula. |
| String/varchar | Tipus de variable on es pot guardar tota mena de caràcters. |
| Bloc Anònimo | Aquesta és la part de base de dades on va gran part de les funcionalitats que usarem ja sigui per a mostrar dades, introduir-les, eliminar-les i/o actualitzar-les. |
| Procedure | També conegut com a procediment s'utilitza per a guardar les funcionalitats d'alguna part de base de dades perquè després a l'hora de fer ús en un bloc anònim sigui més fàcil cridar-li fins i tot en Java sigui més fàcil introduir dades i extreure'ls. |
| Consultes | En base de dades es coneix també com query que serveix per a poder extreure dades de la base de dades fent servir condicions entre altres funcions. |
| ID | Identificador únic |
| GitHub | “GitHub” és una forja per a allotjar projectes utilitzant el sistema de control de versions “Git” |
| Branches | Són apartats que es poden crear en el nostre projecte de GitHub. Aquests apartats seran diferents al “main” ja que si pugem coses al “main” es guardarà en les altres “branches” que haurem creat però si guardem alguna cosa en les “branches” noves no es guardarà en el “main” almenys que ho indiquem. |

## FAQ

Como executar tots els "procedures" que son "get's", "set's", "update", "delete" i el procedure "log in" i "register": Aquests "procedures" necessiten que la base de dades estigui creada per a poder-los executar-los per poder utilitzar-los primer haurem d'haver creat respectivament la base de dades i després executar-los.

## WEBGRAFIA

<https://www.xataka.com/basics/que-github-que-que-le-ofrece-a-desarrolladores>