**DOCUMENTACIÓN TÉCNICA**

**ÍNDICE**

1. Introducción
2. M01 Sistemas Informáticos
3. M02 Bases de Datos
   1. Diseño base de datos
   2. Scripts
4. M03 Programación
5. M05 Entorno de desarrollo
   1. Git Hub
   2. Diagrama de Casos de Uso
   3. Diagrama de Actividades
6. FAQ
7. Conclusión
8. Webgrafía / Bibliografía

**INTRODUCCIÓN**

Planet Wars es un proyecto con el objetivo de poner a prueba nuestros conocimientos acerca de los diferentes módulos del curso que son Sistemas Informáticos (M01), Bases de Datos (M02), Programación (M03) y Entornos de Desarrollo (M05). Antes de iniciar el proyecto, los miembros de este grupo, que somos Alex Pozas, Eduardo Macian, Unai Ricco y Andres Villca, hemos decidido utilizar “trello”, una herramienta que nos ayudaría a organizarnos y saber que parte del proyecto está realizando cada integrante del grupo.

**M01 SISTEMAS INFORMÁTICOS**

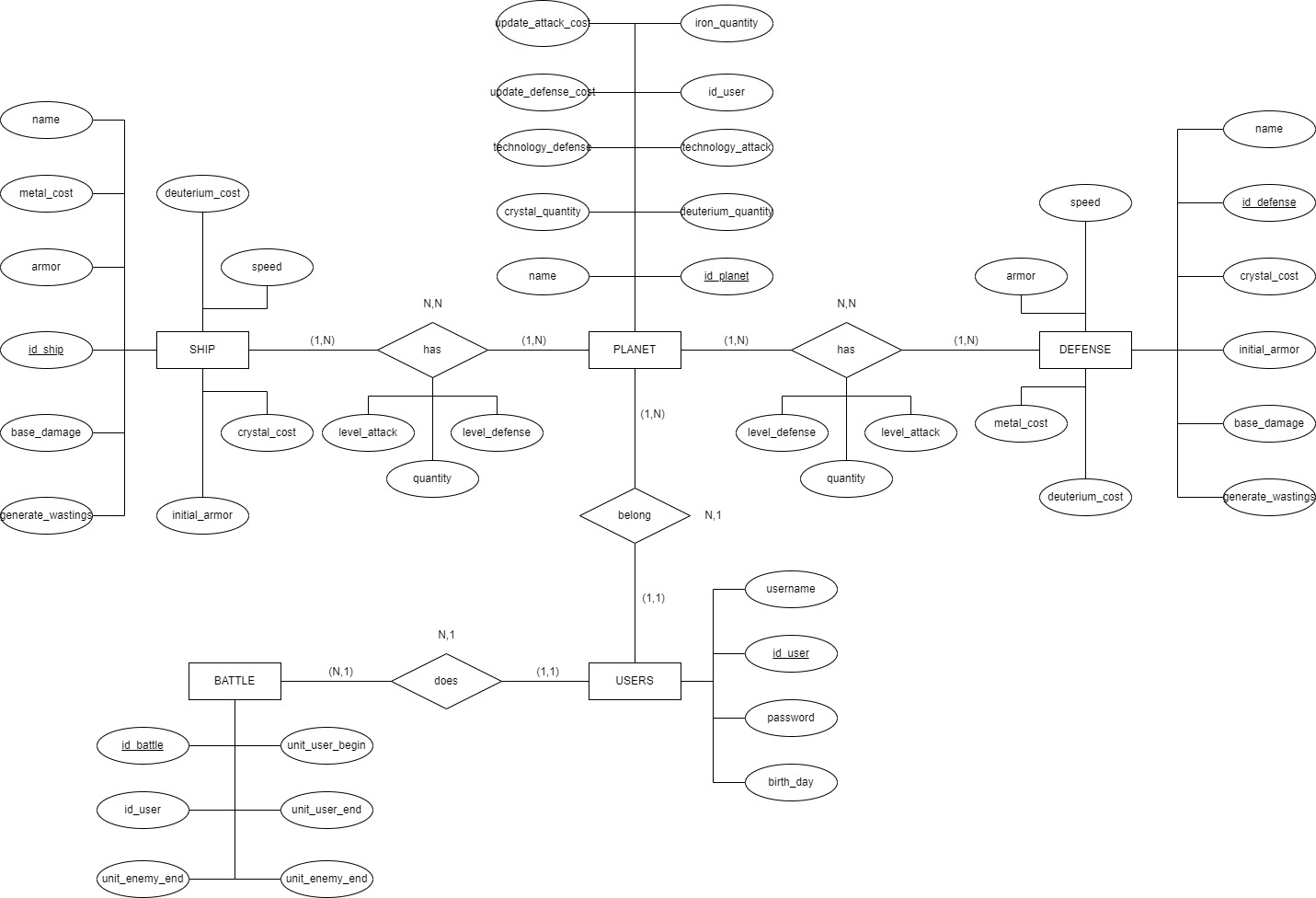
En sistemas informáticos el principal objetivo es crear una documentación técnica del desarrollo del proyecto que documente el proceso y la funcionalidad de cada parte de los módulos.

**M02 BASE DE DATOS**

**+ DISEÑO DE DATOS**

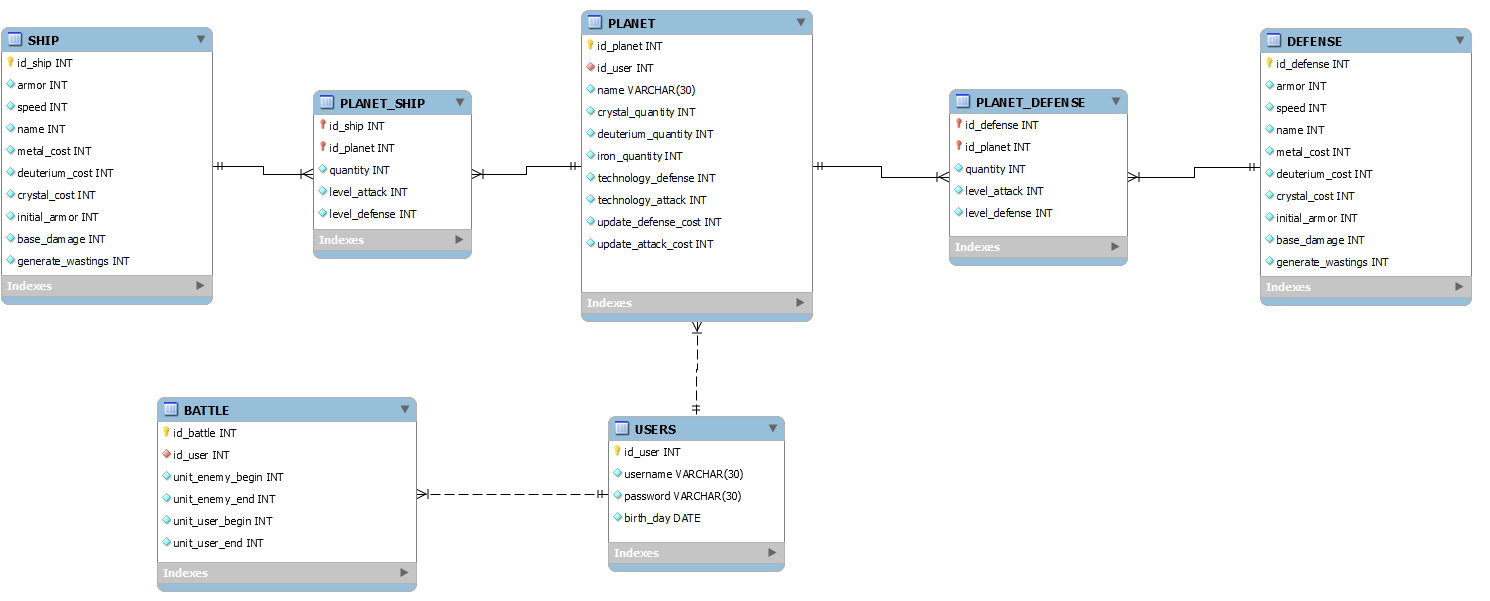
En base de datos nos piden realizar el diseño de un modelo de datos a implementar, unos scripts y el uso de la base de datos.

Diagrama Chen

Para iniciar la parte de M02 tenemos que realizar un diagrama chen, que es útil para modelar los conceptos básicos de las entidades y relaciones, de esta manera se comprende la estructura de la base de datos.

El diagrama chen tiene 5 entidades, y cada entidad tiene sus propios atributos. Las entidades SHIP y DEFENSE tienen los mismos atributos. La relación entre SHIP y PLANET tienen una relación con cardinalidad N:N ya que un planeta puede tener diferentes naves y una nave puede pertenecer a diferentes planetas. Lo mismo pasa con la relación entre PLANET y DEFENSE. En cuanto a la cardinalidad entre PLANET Y USERS es N:1 porque un usuario puede pertenecer a diferentes planetas en cambio un planeta solo puede pertenecer a un usuario. Y por último la relación entre BATTLE y USERS, que tiene una cardinalidad de N:1 debido a que un usuario puede realizar muchas batallas, pero una batalla solo puede ser realizada por un usuario.

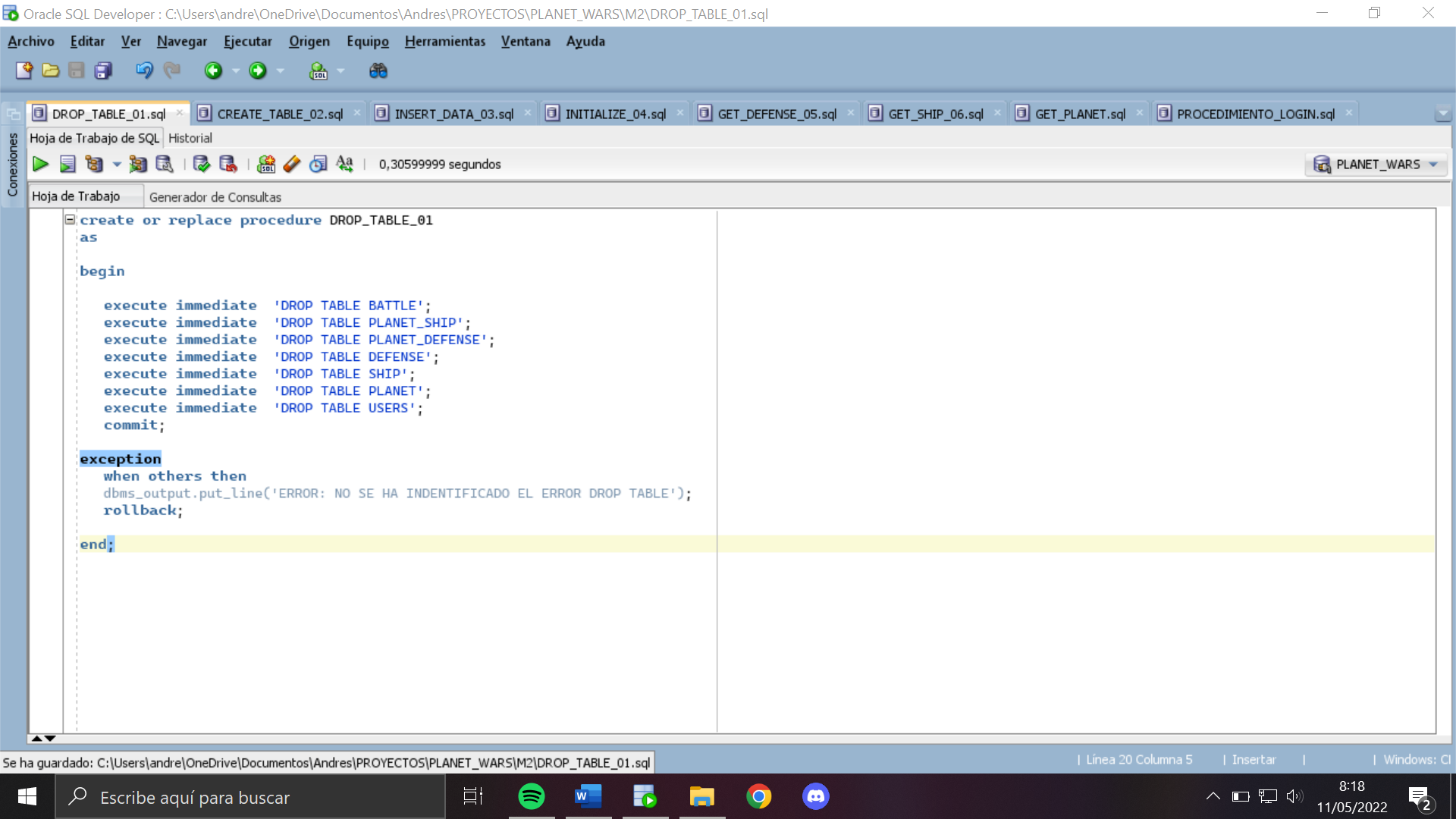
Modelo Relacional

A partir del diagrama chen tenemos que crear un modelo relacional, que proporciona una forma estándar de representar y consultar datos.

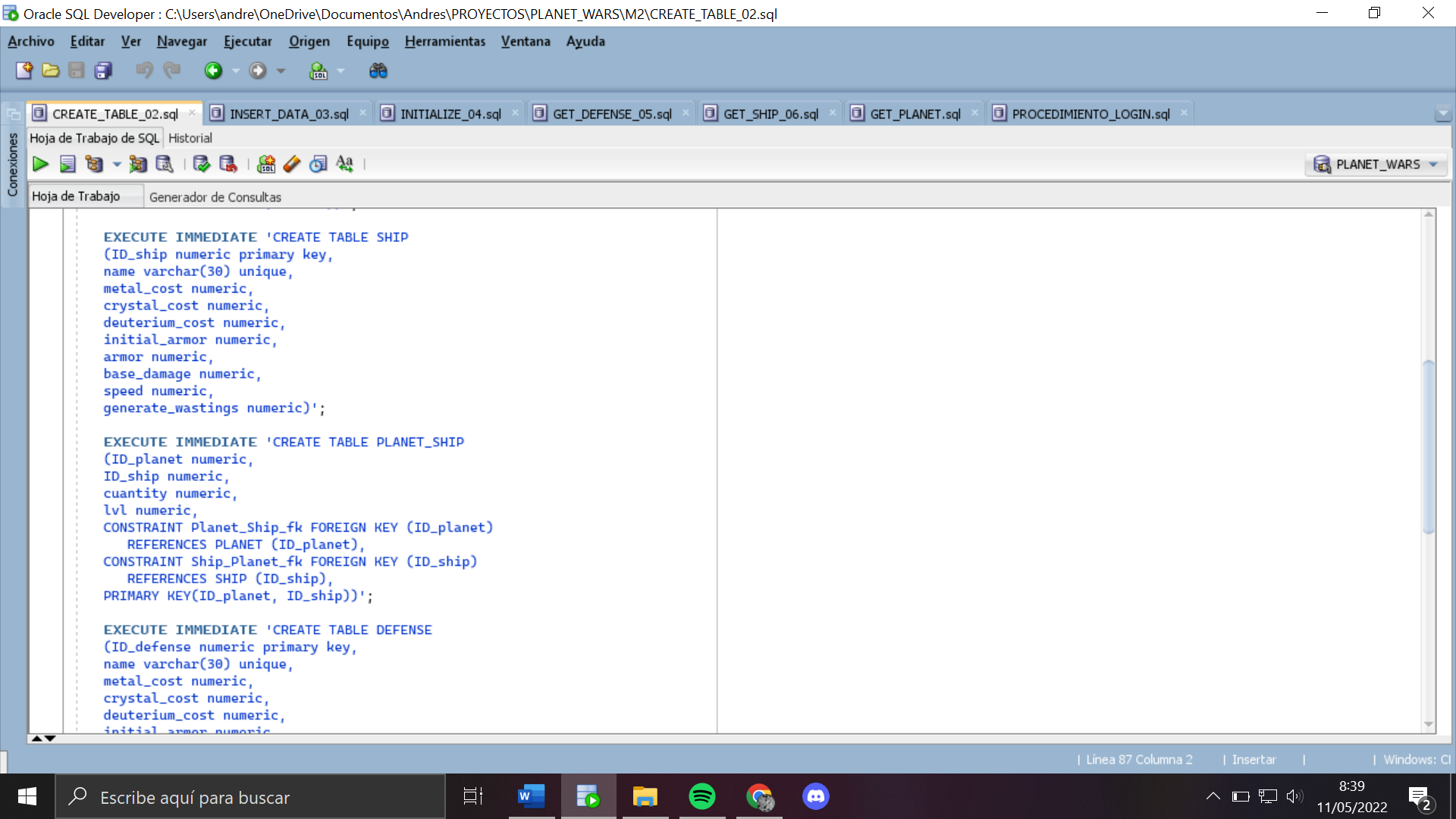
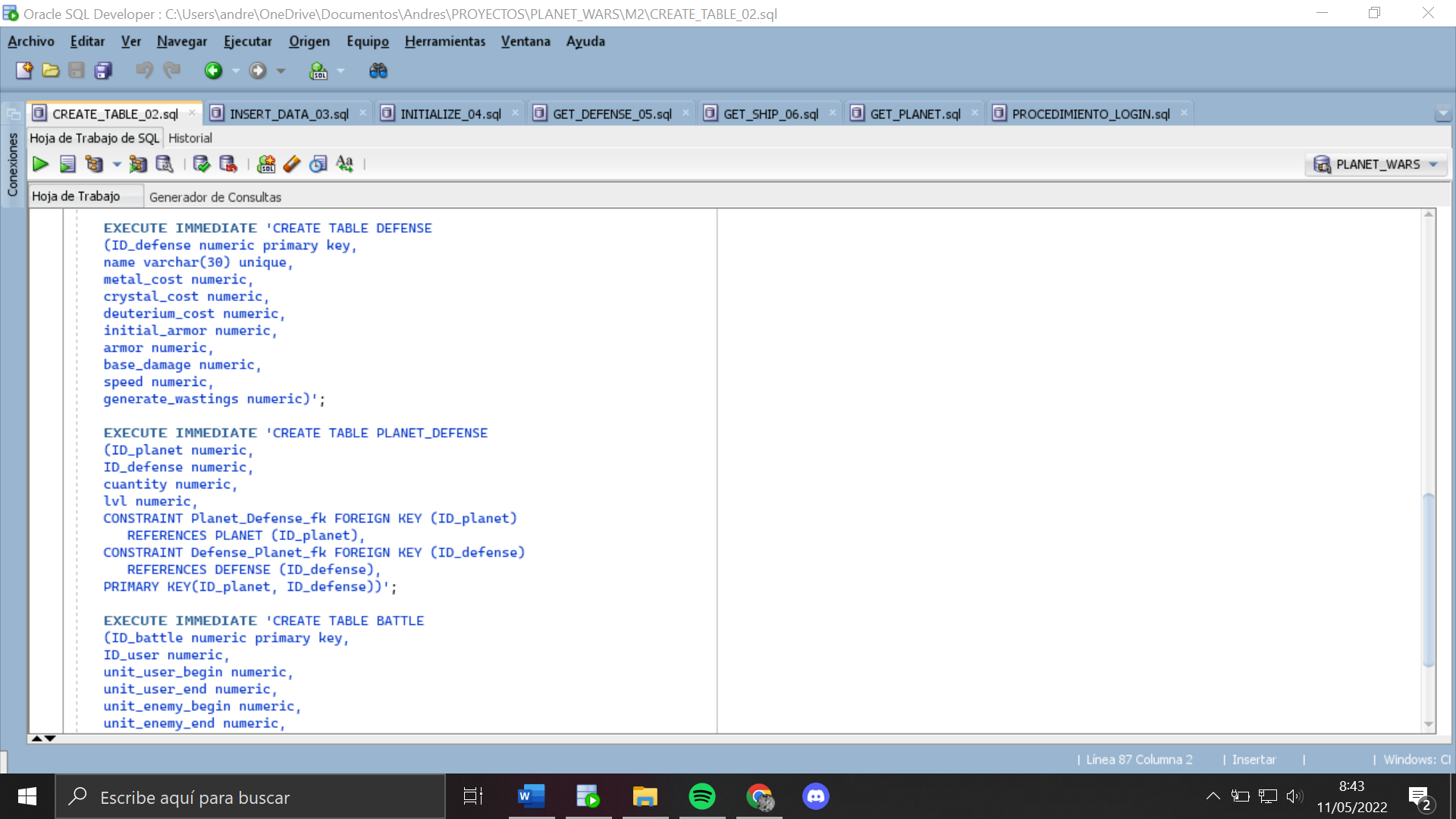
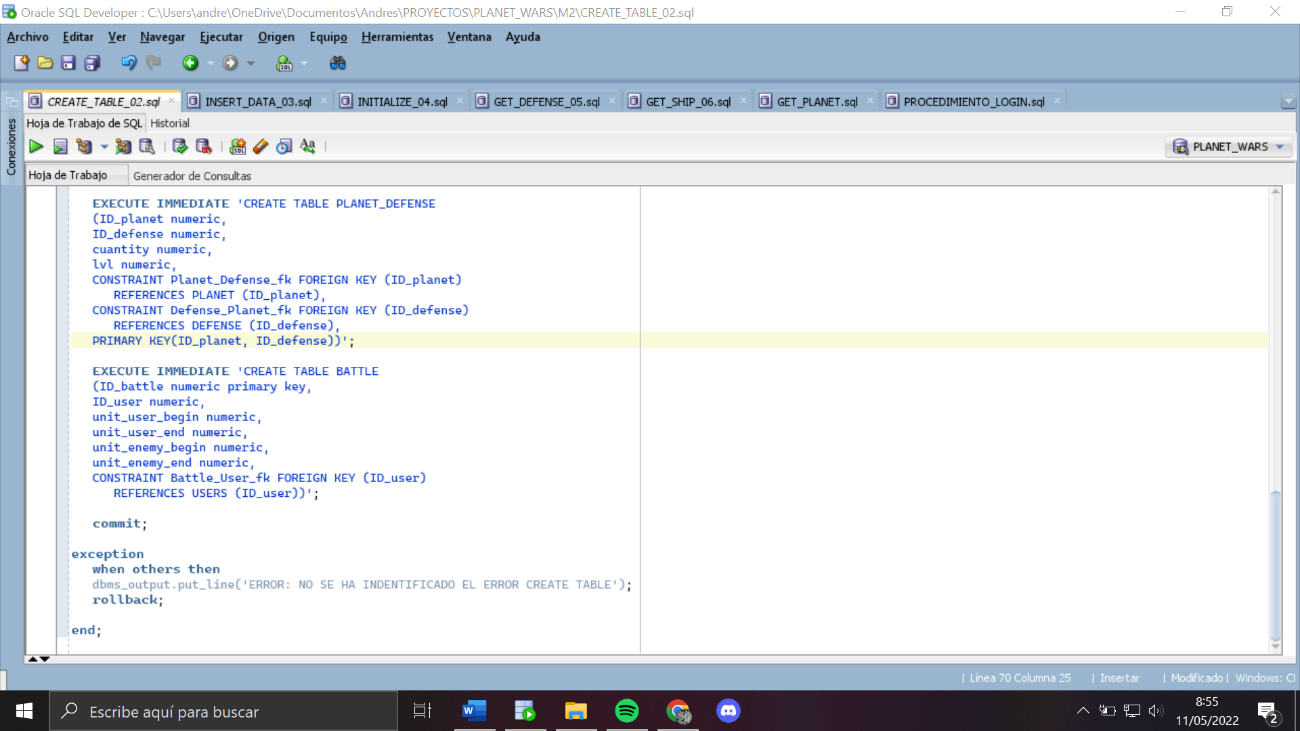
Nuestro modelo relacional cuenta con 7 tablas a diferencia del diagrama chen que cuenta con 5. Las dos tablas que augmenta en este modelo son debido a la cardinalidad N:N que vimos anteriormente. La tabla PLANET\_SHIP recibe la primary key de SHIP y PLANET además de tener sus propios atributos. Lo mismo pasa con PLANET\_DEFENSE.

**+ SCRIPTS**

Son procedimientos plsql con nombre, que pueden recibir parámetros y/o entregar datos. Pueden ser invocados mediante el nombre y según si se ha creado con parámetros o no, tendremos que añadirlo.

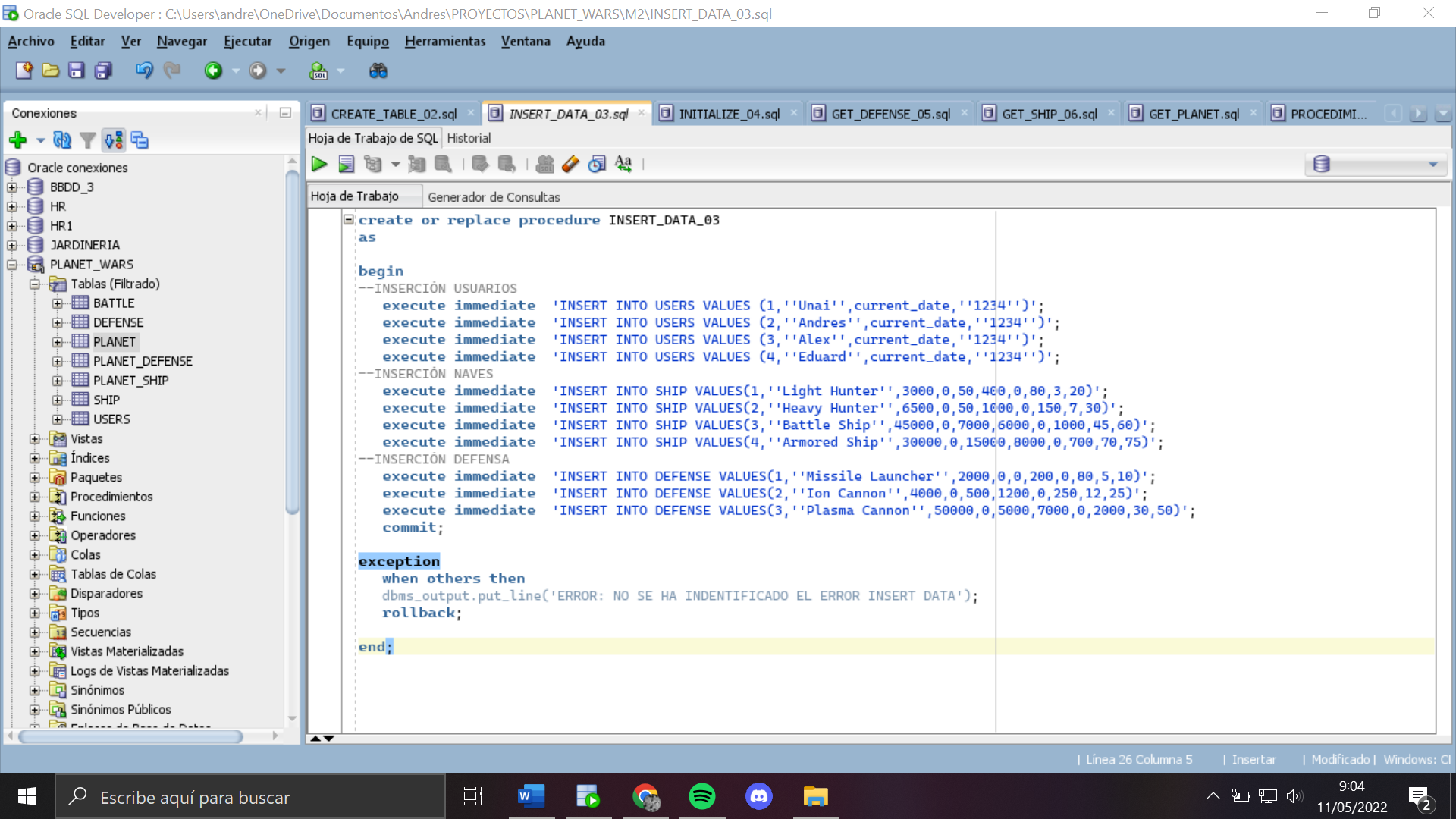
Drop Table

Este procedimiento tiene el objetivo de borrar todas las tablas de la base de datos siempre que existan, y lo hará de forma ordenada ya que de esta manera evitaremos errores. Se usará cuando el programa inicie o cuando el usuario le dé al botón “reset”. Implementa un “commit” que realiza un cambio permanente en la base de datos, y un “rollback”, que deshace los cambios que la ejecución ha realizado.

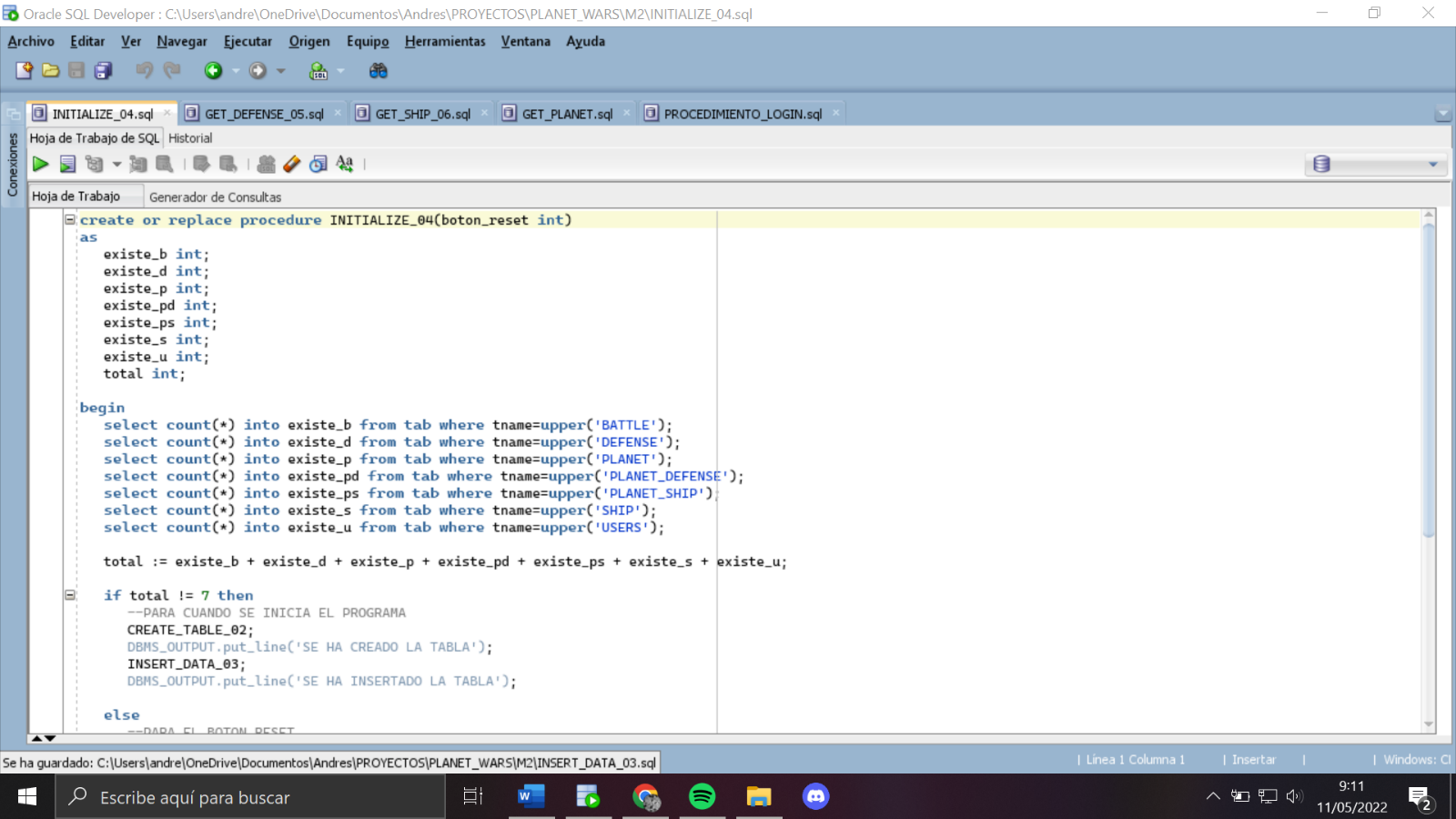
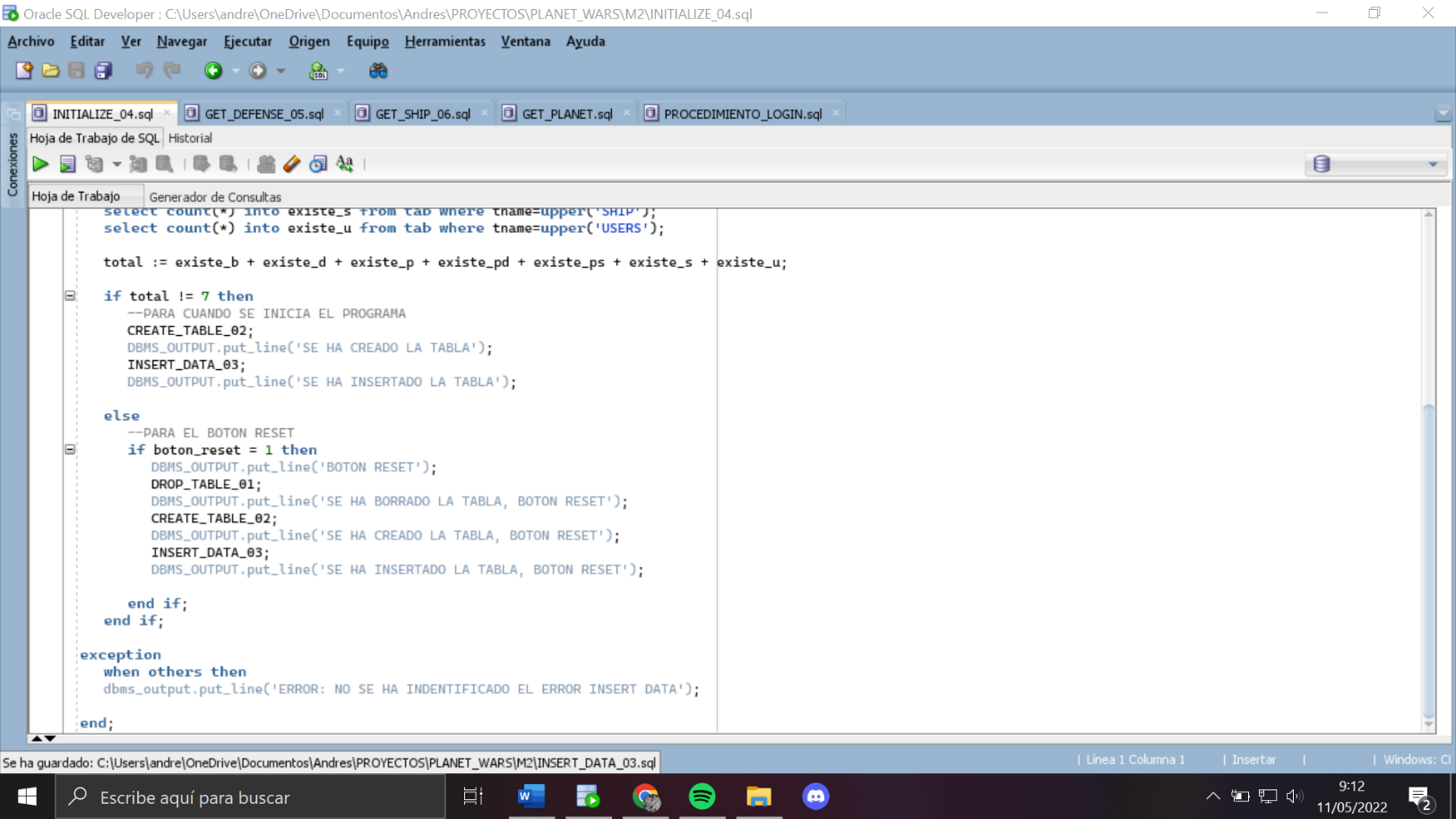
Create Table

Este procedimiento es útil para la creación de tablas en caso de que no existan. Este y el anterior servirán para un procedimiento posterior. Se utilizará cuando el programa inicie o cuando el usuario le dé al botón “reset”. Tiene también un “rollback” y un “commit”.

Insert Data

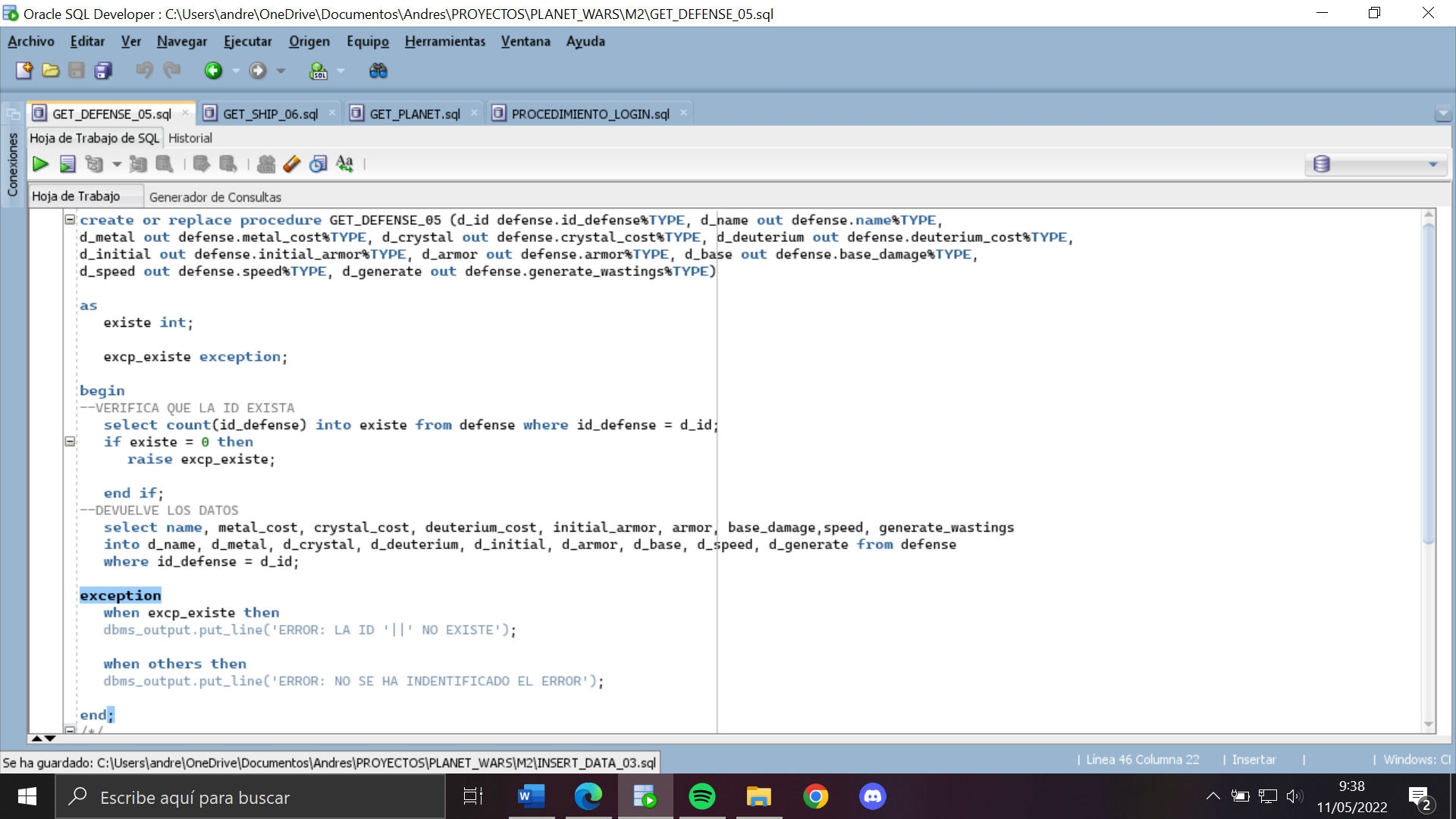


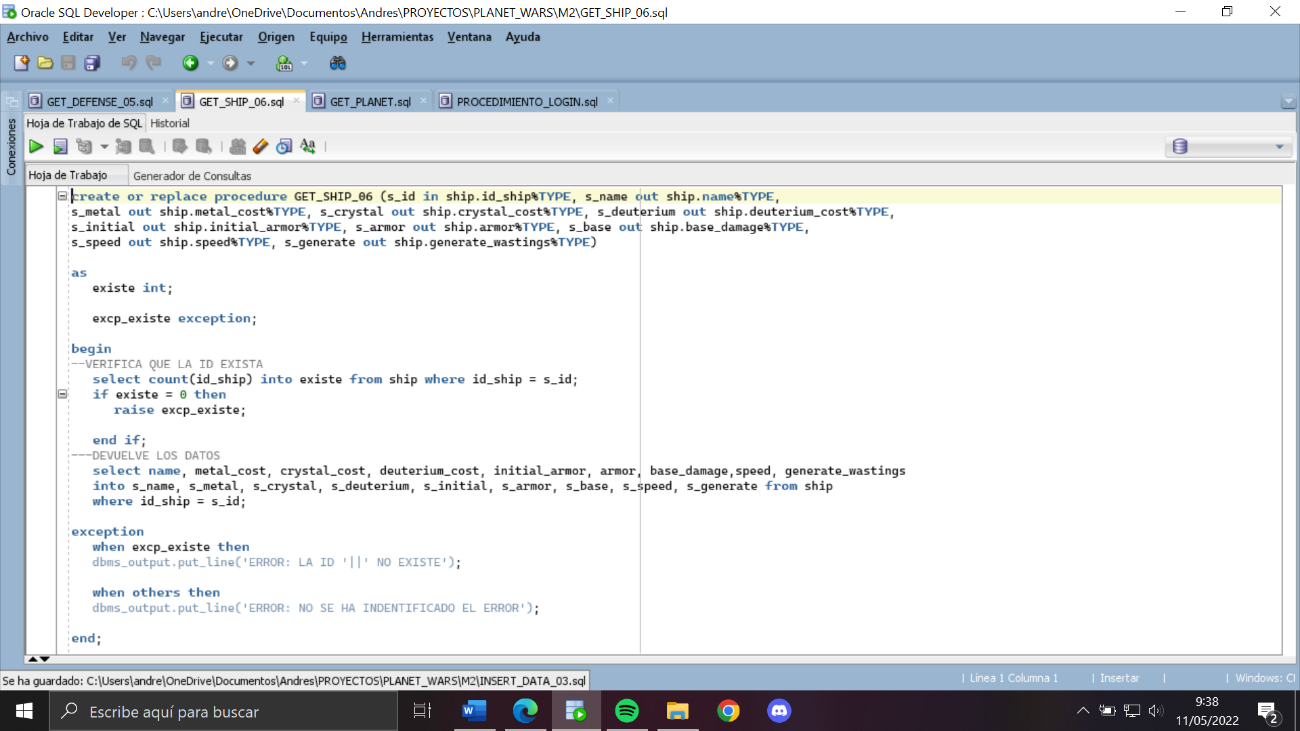
Este procedimiento se utiliza para insertar datos en las tablas. Será útil en un procedimiento posterior. En nuestro caso, aparte de insertar los datos pedidos, hemos creado un usuario para cada miembro, de esta manera ya tendremos un usuario para iniciar sesión en el juego. Se usará cuando el programa inicie o cuando el usuario le dé al botón “reset”. En las únicas tablas en las cuales hemos insertado datos son, USERS, SHIPS y DEFENSE.

Initialize

Los tres procedimientos anteriores son invocados por este, que en el momento el cual el programa inicie, verificará que tenga tablas o no. En caso de tener tablas, este procedimiento no hará nada, pero si no tiene iniciará los procedimientos Drop Table, Create Table y Insert Table, en ese orden. El juego tendrá a disposición un botón “reset” que se usará para reiniciar los datos predeterminados del juego, eso quiere decir que ejecutará el procedimiento Initialize.

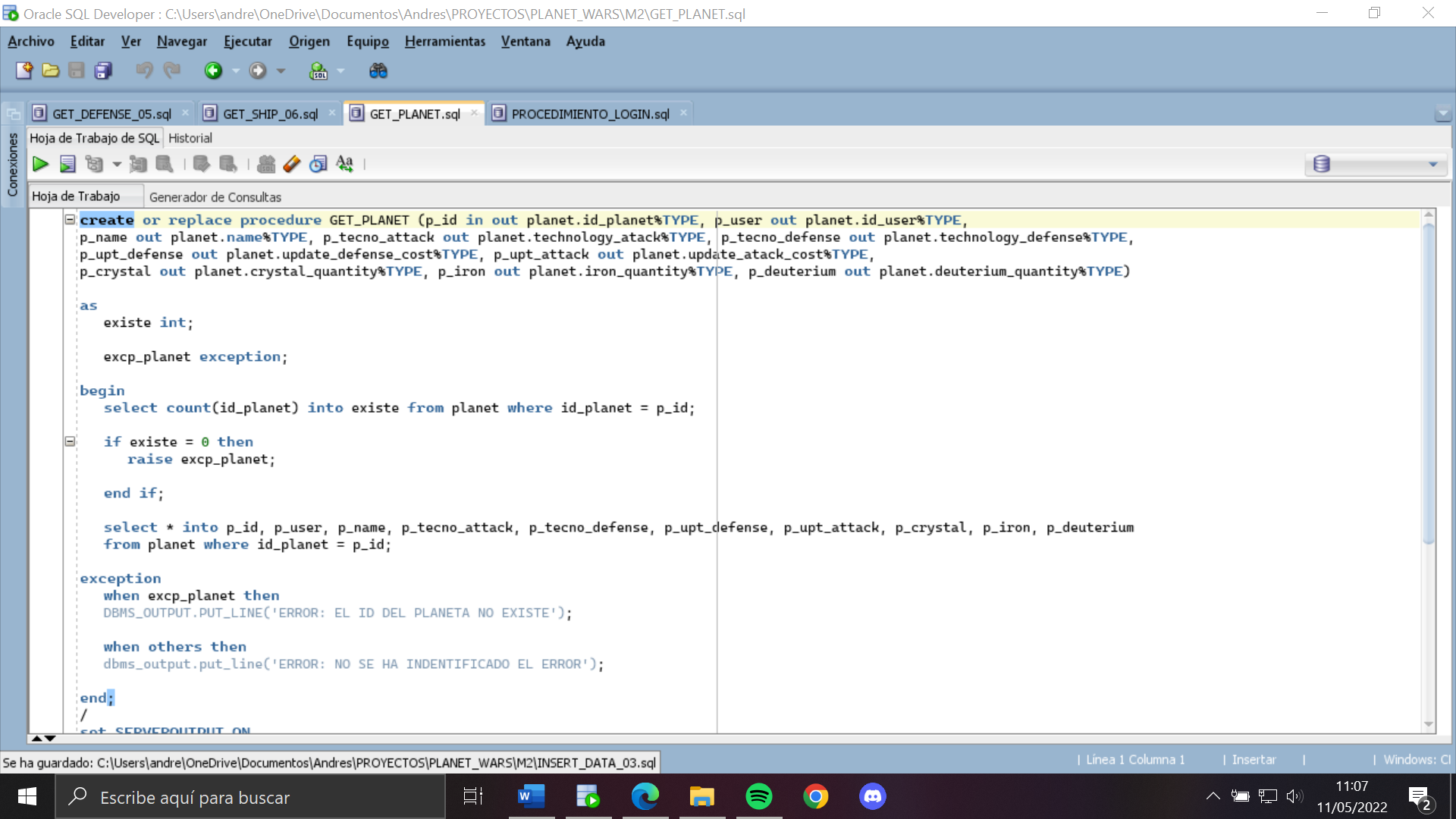
Get Defense y Get Ship





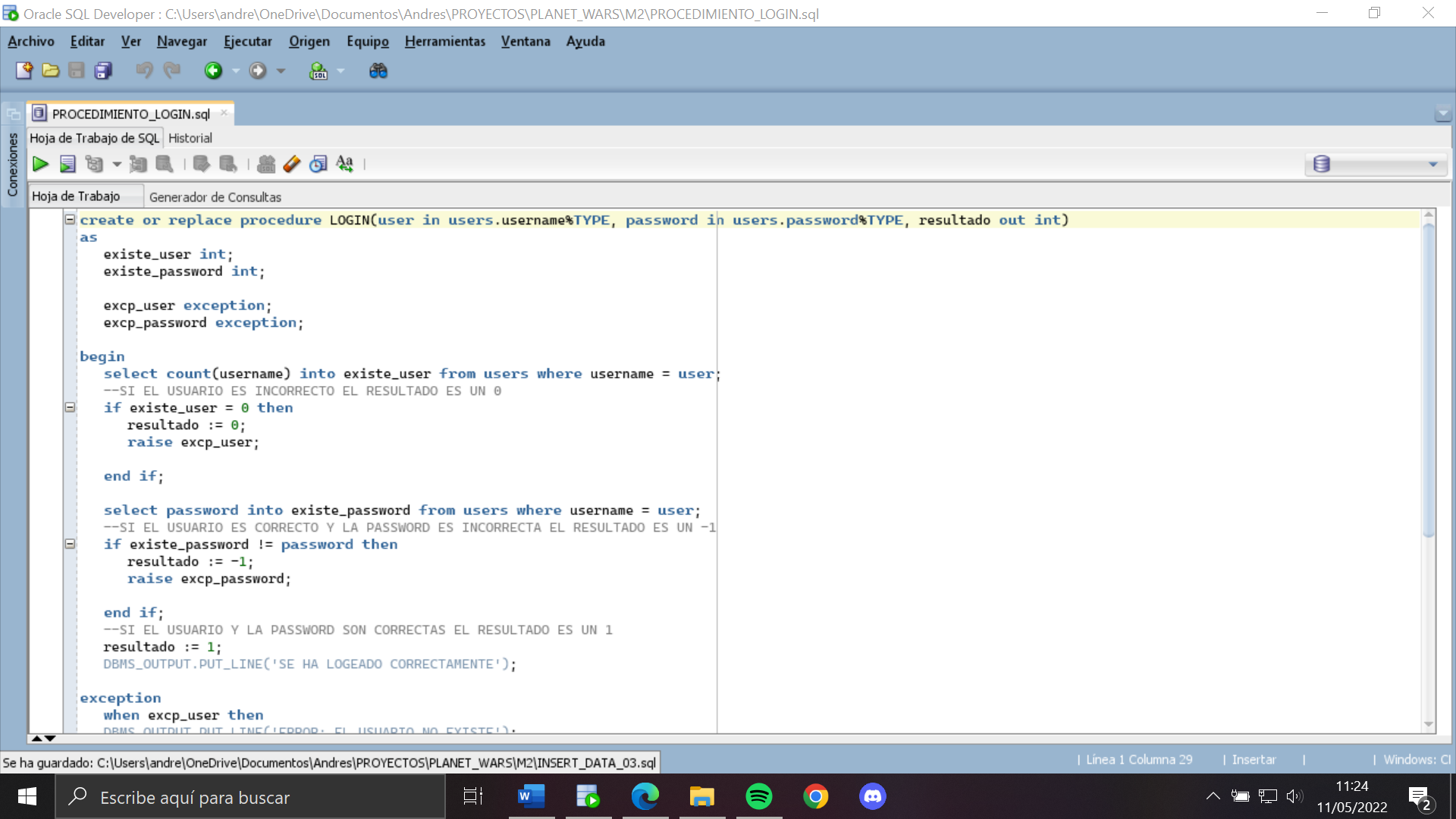
Estos dos procedimientos tienen el mismo uso, que es entregar al programa información sobre la tabla, pero para saber perfectamente que parte de la información quiere, el procedimiento recibirá un id. Primero comprobaremos que el id exista, en caso de que no, lanzaremos una excepción, por lo contrario, el procedimiento entregará la información que tenga ese id. La única diferencia entre estos dos procedimientos es la tabla del cual obtendrán la información, uno lo hará de la tabla DEFENSE y el otro de SHIP

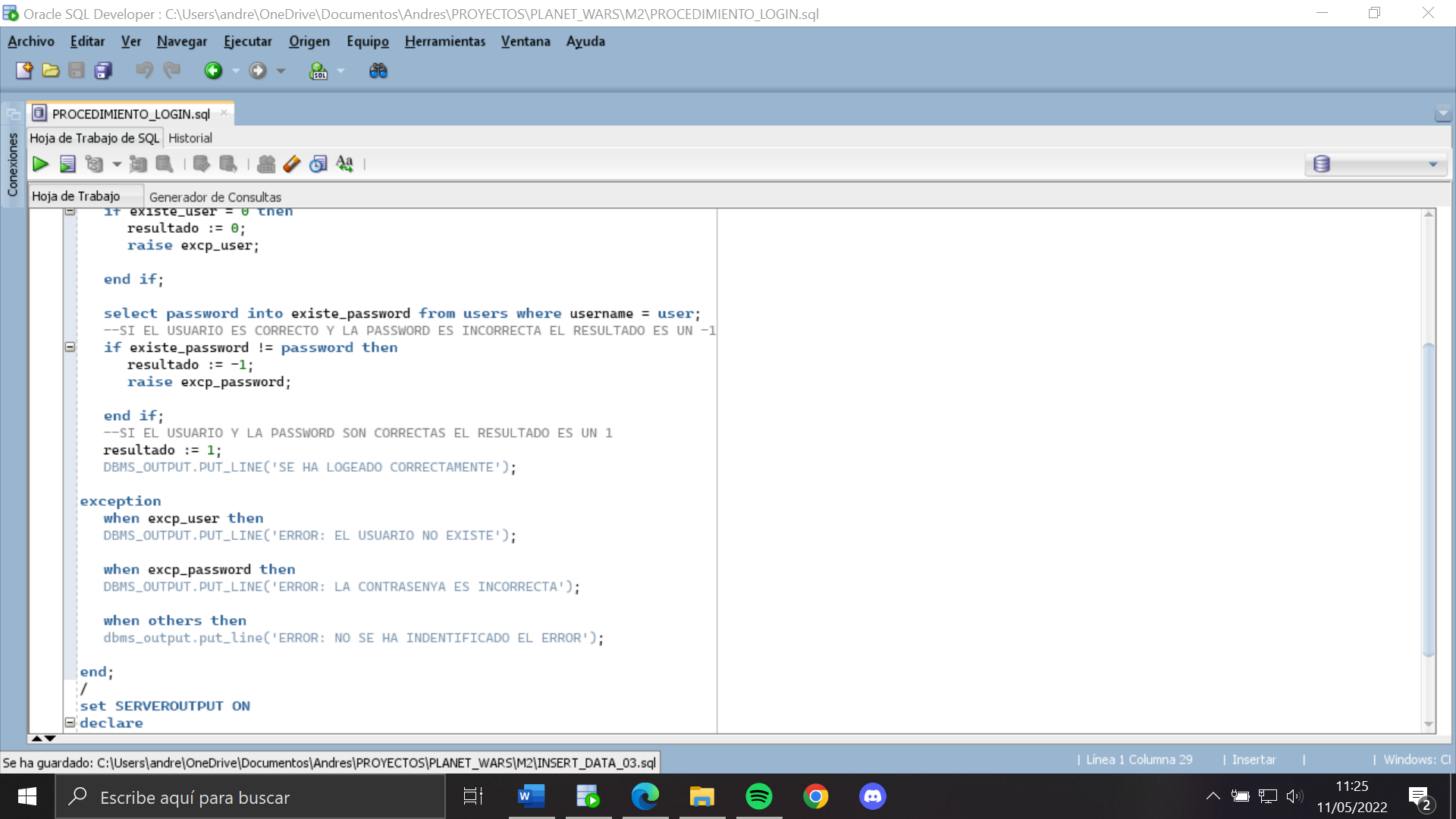
Get Planet



Los anteriores procedimientos eran obligatorios, pero este y los siguientes los hemos creado para que la programación del juego sea más cómoda. Este procedimiento es completamente igual al anterior, eso quiere decir que recibiremos una id y entregaremos información de la tabla PLANET en función a la id.

Procedimiento Login





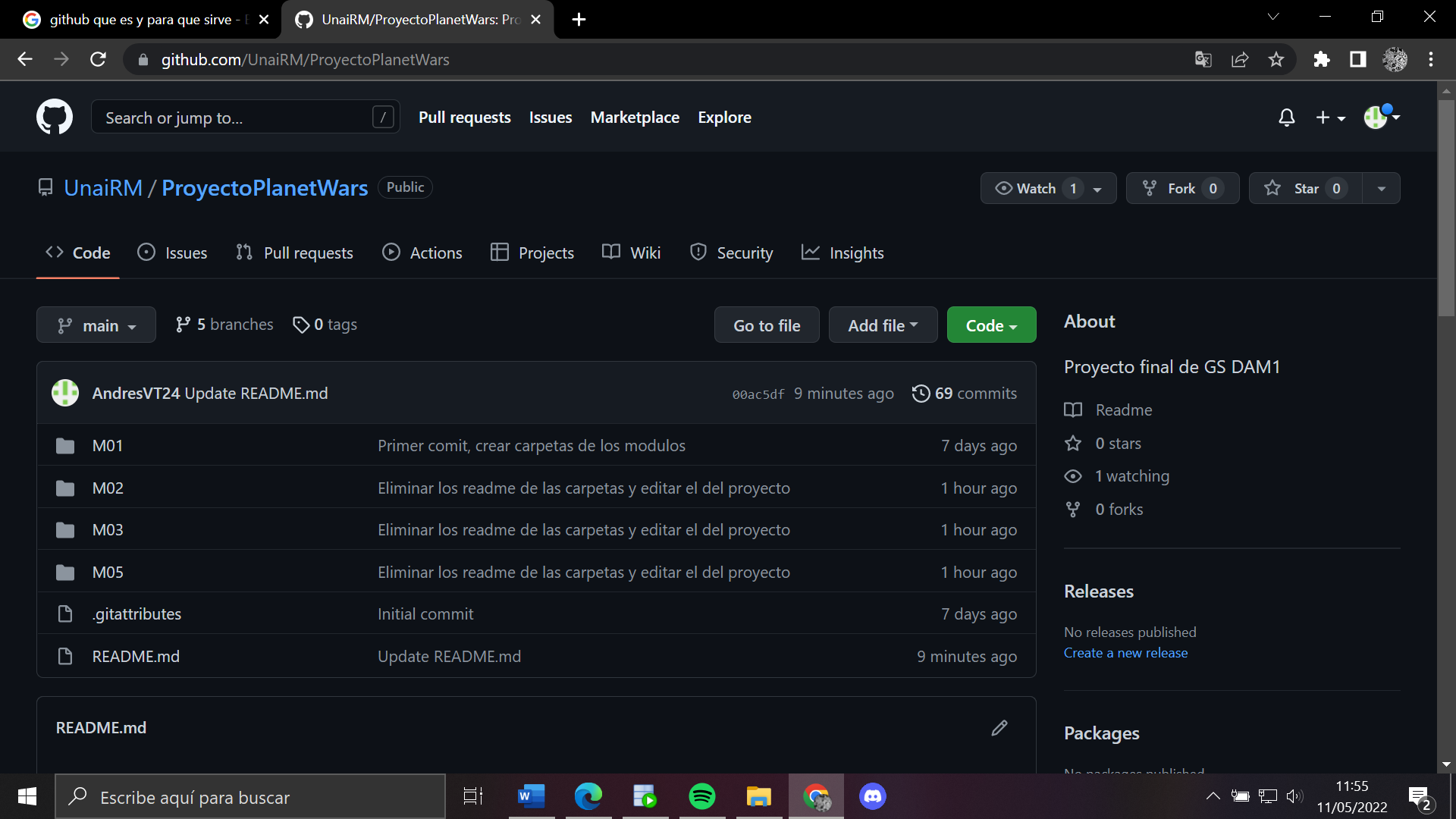
Este procedimiento recibirá dos parámetros, el usuario y la contraseña, y entregará un 1, 0 o -1. El objetivo es comprobar si el usuario existe y si la contraseña es correcta. En caso que el usuario no exista, el procedimiento entregará un 0. Si el usuario existe, pero la contraseña es incorrecta, entregará un -1. Y si el usuario existe y la contraseña es correcta, entregará un 1. En caso que el usuario no existe, pero la contraseña sea correcta al no existir el usuario, que es lo que primero comprobará, retornará un 0.

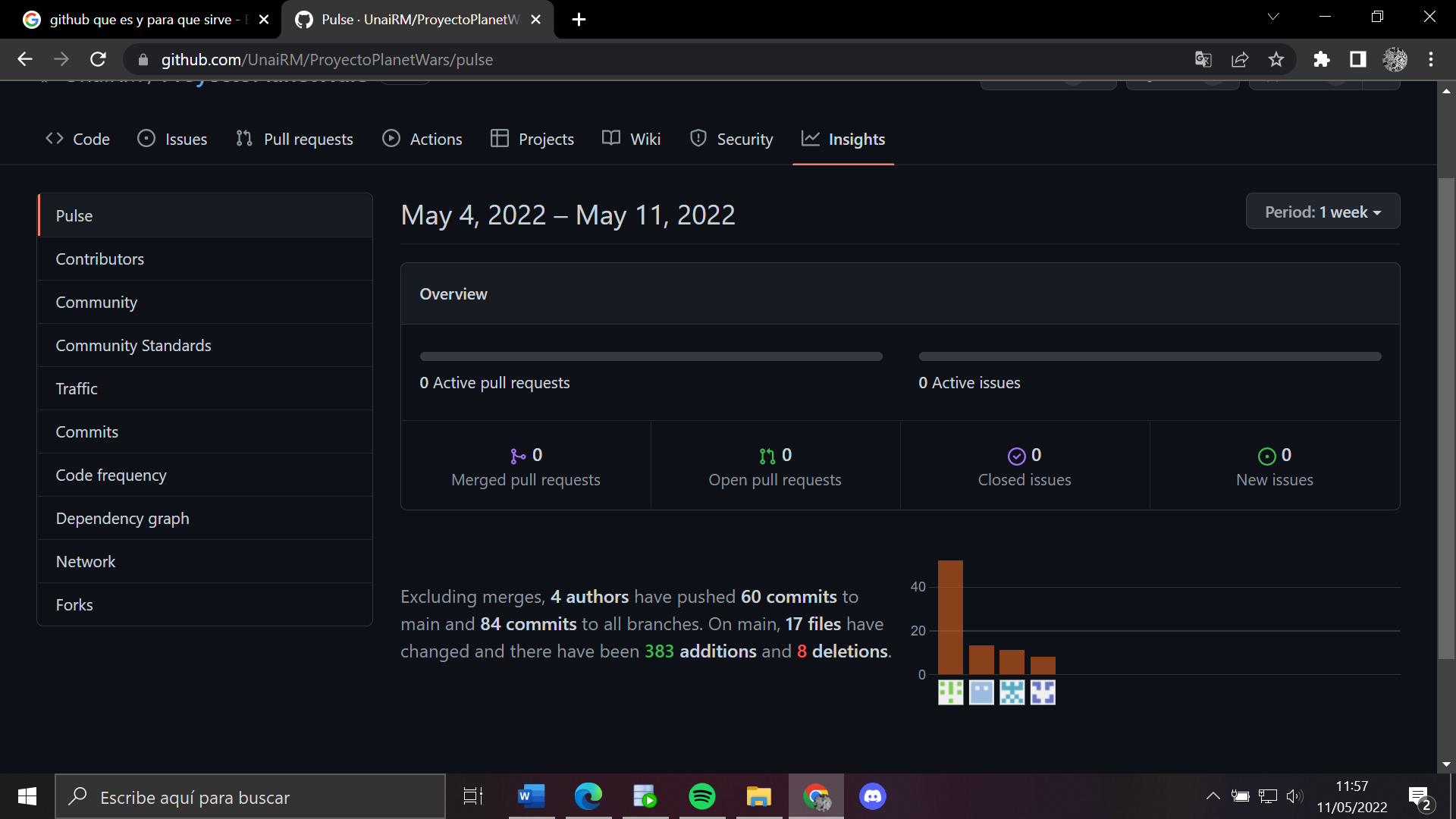
**M03 PROGRAMACIÓN**

+clases

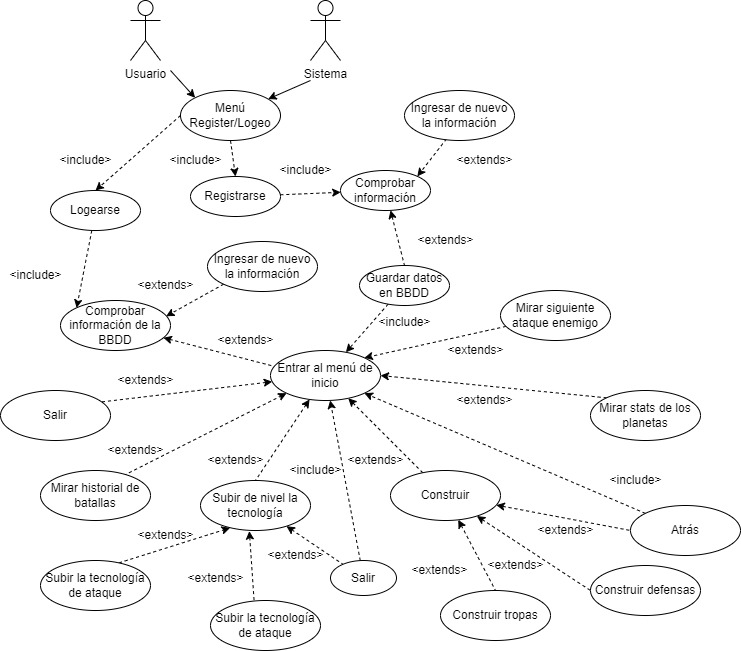
**M05 ENTORNO DE DESARROLLO**

**+ GIT HUB**

Git Hub es un repositorio online gratuito que permite gestionar proyectos y almacenar códigos y otros ficheros. Es útil para organizar y compartir los archivos entre los integrantes del grupo

Principalmente hemos creado 4 carpetas para cada módulo y un fichero llamado “README” que explica brevemente en que consiste el proyecto. A parte de eso, también hemos establecido una “branch” (rama) para cada integrante del grupo. Eso quiere decir que cada vez que una persona inserte un fichero, lo tendrá que hacer en su “branch”, de esta manera la persona tendrá sus propios ficheros.

Cada uno de los miembros del grupo ha realizado más de 5 “commits”. Que son cambios que se llevan acabo y se guardan. Estos cambios los miembros del grupo expresamente, eso quiere decir que no hay un autoguardado.

+ DIAGRAMA DE CASOS DE USO

En el diagrama de casos de uso los actores principales son usuario y sistema. El principal medio de interacción entre los dos actores es el menú de inicio. Para poder iniciar sesión el usuario tiene que haberse registrado anteriormente.

**GLOSARIO**

palabra

**FAQ**

+problemas

**CONCLUSIÓN**

grupo