



UNIVERSIDAD PRIVADA FRANZ TAMAYO

Proyecto hito 5

Manual de usuario Warcraft

Nombre Completo: Nicolas Gonzalo Aguilar Arimoza

Juan Elian Alvarez Vallejos

Diego Rivera Tapia

Asignatura: programación 3 y base de datos

Carrera: INGENIERÍA DE SISTEMAS

Paralelo: PROG (1)

Docente: Ing. William Barra

fecha: 26/06/2020



Introducción

En este proyecto presentamos un simulador para Warcraft creado en inteliej IDEA en el encontramos las diferentes razas tales como humanos, orcos, elfos nocturnos y muertos vivientes.

Cada uno consta con unidades distintas como también héroes, el simulador consta de enfrentar los ejercitos de cada bando y escoger un ganador.

Al escoger un ganador la raza que perdió cambia de nombre a elfos sanguinarios, saqueadores, renegados, orco fel.

Utilizamos una base de datos en data grip para almacenar todas las unidades de cada raza como también daño, armadura, vida, comida y su tipo de clase.

Procesos

- Al momento de iniciar el programa se nos abrirá el menú donde tenemos que escoger la raza



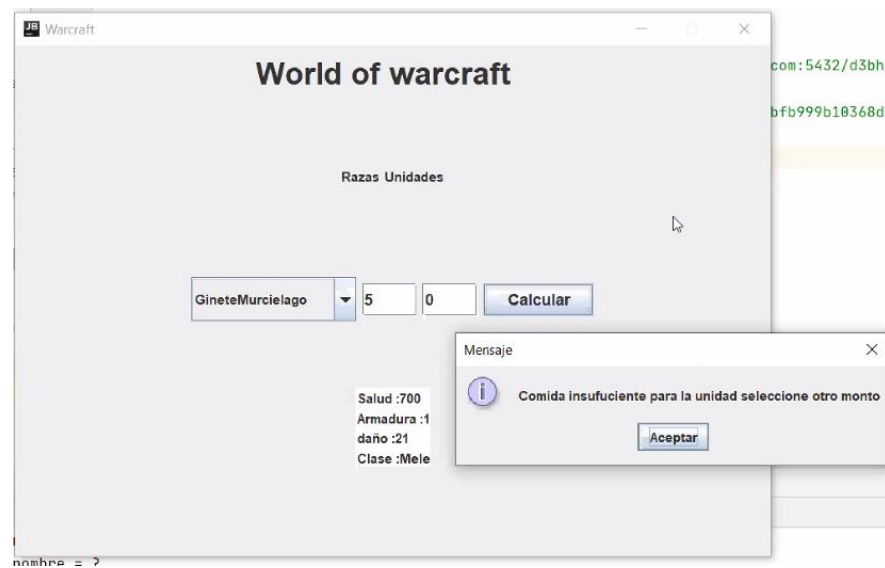
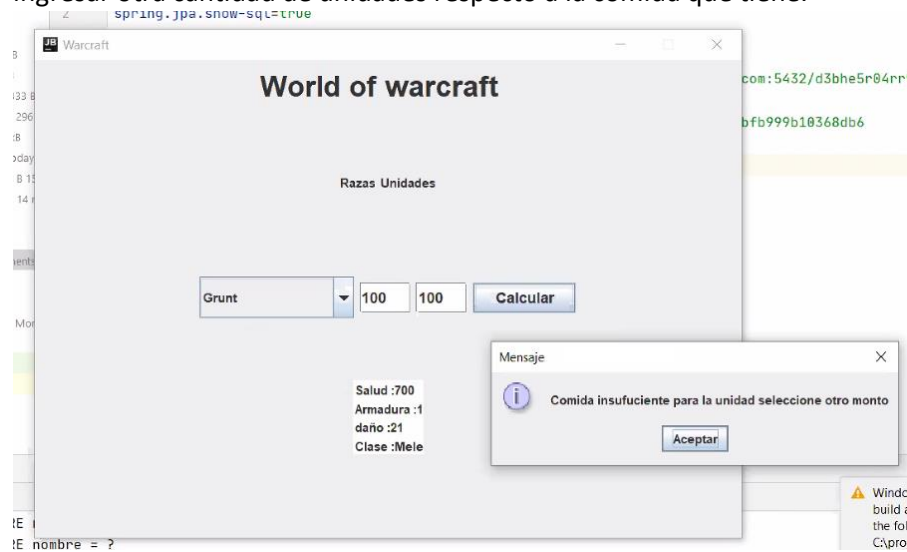
- Al momento de escoger la raza solo podremos seleccionar las unidades de esa raza seleccionada y nos saldrá información de esa unidad.

The screenshot shows a web application titled "World of Warcraft" with a subtitle "Razas Unidades". It features two dropdown menus: the first is set to "Orcos" and the second to "Grunt". To the right of these is a text input field containing the number "100" and a "Calcular" button. Below the input fields, a tooltip displays the following statistics: "Salud :700", "Armadura :1", "daño :21", and "Clase :Mele".

- Al momento de seleccionar la cantidad en el texto oprimimos calcular y disminuirá la cantidad de comida limite

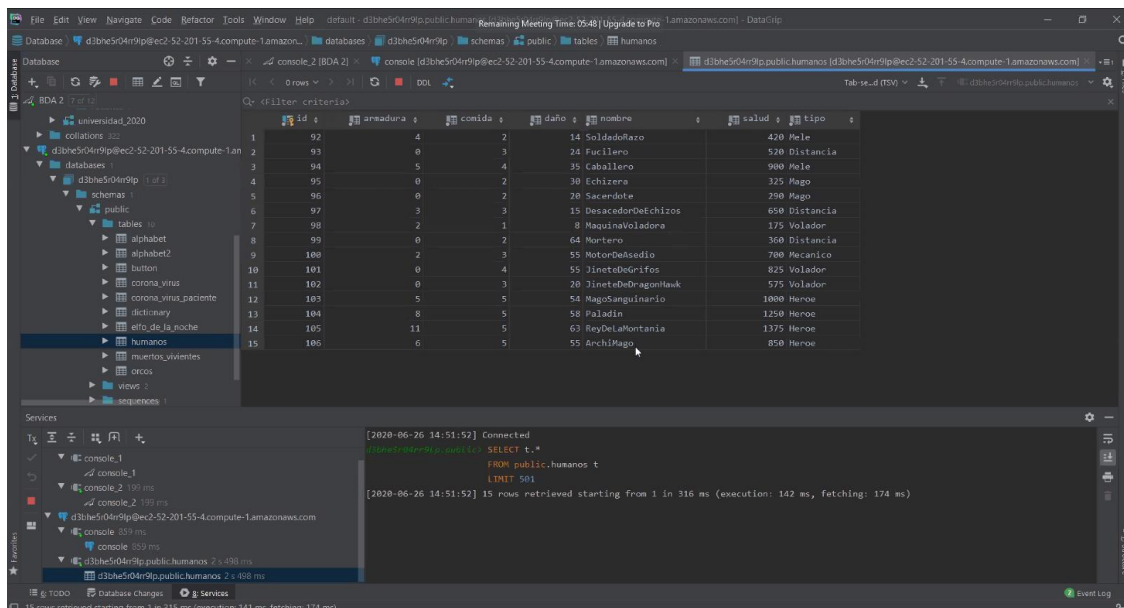
The screenshot shows the same "World of Warcraft" interface. The first dropdown menu is now set to "Demoledor". The second dropdown menu is empty. The text input field now contains the number "3", and the "Calcular" button is highlighted. A tooltip displays the following statistics: "Salud :325", "Armadura :2", "daño :102", and "Clase :Mecanico".

- El programa viene con un limite que es cuando el contador llega a 0 o no alcanza la comida para la cantidad ingresada, el programa le mandara un mensaje diciéndole que tiene q ingresar otra cantidad de unidades respecto a la comida que tiene.



Base de datos información

Todo esta controlado desde la base de datos de data grip que ingresamos por código, cada unidad como la creación de las tablas esta realizada por código y se ve de esta manera:

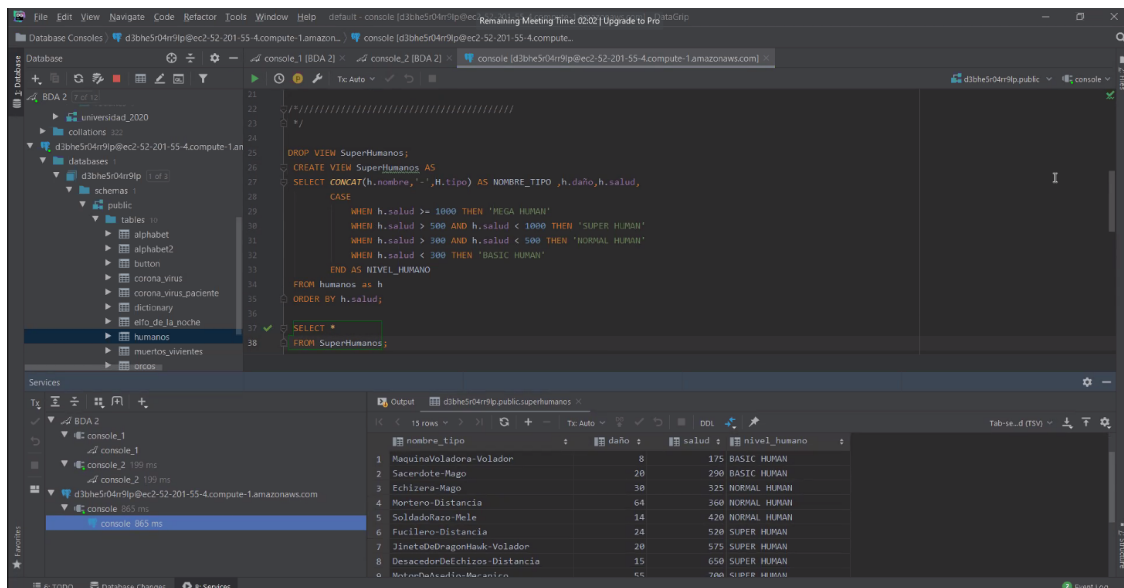


The screenshot shows the DataGrip interface with a SQL query executed in the console. The query selects data from the 'humanos' table, ordered by 'id'. The results are displayed in a table with the following columns: id, armadura, comida, daño, nombre, salud, and tipo.

id	armadura	comida	daño	nombre	salud	tipo
92	4	2	14	SoldadoRazo	420	Mele
93	0	3	24	Fucilero	520	Distancia
94	5	4	35	Caballero	900	Mele
95	0	2	30	Echizera	325	Mago
96	0	2	20	Sacerdote	290	Mago
97	3	3	15	DesacadorDeElchizos	650	Distancia
98	2	1	8	RequinalVoladora	175	Volador
99	0	2	64	Mortero	360	Distancia
100	2	3	55	MotorDeAsedio	700	Mecanico
101	0	4	55	JineteDeGrifos	825	Volador
102	0	3	20	JineteDeDragonHawk	575	Volador
103	5	5	54	MagoSanguinario	1000	Heroe
104	8	5	58	Paladin	1250	Heroe
105	11	5	63	ReyDeLaMontania	1375	Heroe
106	6	5	55	ArchiMago	850	Heroe

Toda la base de datos esta controlado por vistas triggers como también procedimientos de almacenado

Una de las primeras vistas que utilizamos es para diferenciar que unidad tiene mas poder que otras y las diferenciamos como mega humano super humano ect.



The screenshot shows the DataGrip interface with a SQL query executed in the console. The query creates a view named 'SuperHumanos' based on the 'humanos' table, categorizing units into 'MEGA HUMAN', 'SUPER HUMAN', 'NORMAL HUMAN', or 'BASIC HUMAN' based on their 'salud' (health) and 'daño' (damage) values. The results are displayed in a table with the following columns: nombre_tipo, daño, salud, and nivel_humano.

nombre_tipo	daño	salud	nivel_humano
RequinalVoladora-Volador	8	175	BASIC HUMAN
Sacerdote-Mago	20	290	BASIC HUMAN
Echizera-Mago	30	325	NORMAL HUMAN
Mortero-Distancia	64	360	NORMAL HUMAN
SoldadoRazo-Mele	14	420	NORMAL HUMAN
Fucilero-Distancia	24	520	SUPER HUMAN
JineteDeDragonHawk-Volador	20	575	SUPER HUMAN
DesacadorDeElchizos-Distancia	15	650	SUPER HUMAN
MotorDeAsedio-Mecanico	55	700	SUPER HUMAN

En esta vista controlamos las clases de cada uno como se puede observar tenemos distancia, mele, mecánico etc.

The screenshot shows a database IDE with a project tree on the left, a SQL editor in the center, and an output window at the bottom right.

SQL Editor:

```

DROP VIEW CONTACLASE;
CREATE VIEW CONTACLASE AS
SELECT count(o.tipo) AS CANTIDAD,o.tipo,
CASE
WHEN o.tipo = 'Distancia' THEN 'CLASE 1'
WHEN o.tipo = 'Mele' THEN 'CLASE 2'
WHEN o.tipo = 'Mecanico' THEN 'CLASE 3'
WHEN o.tipo = 'Mago' THEN 'CLASE 4'
WHEN o.tipo = 'Hecce' THEN 'CLASE 5'
WHEN o.tipo = 'Volador' THEN 'CLASE 6'
END AS CLASE
FROM orcos AS o
GROUP BY o.tipo
ORDER BY CLASE;
SELECT *
FROM CONTACLASE;

```

Output Window:

cantidad	tipo	clase
2	Distancia	CLASE 1
3	Mele	CLASE 2
1	Mecanico	CLASE 3
3	Mago	CLASE 4
4	Heroe	CLASE 5
2	Volador	CLASE 6

El resultado nos saldrá la clase 1 es distancia y así diferenciamos en cada raza a q clase corresponde cada uno.

En este trigger controlamos el guardado de cada partida de quien gana como también quien pierde

The screenshot shows a database IDE with a project tree on the left, a SQL editor in the center, and a services window at the bottom.

SQL Editor:

```

DROP TRIGGER CalculaGanador;
CREATE TRIGGER CalculaGanador
BEFORE INSERT ON partidas
FOR EACH ROW
BEGIN
IF (NEW.cantidad1 > NEW.cantidad2)
then
INSERT INTO auditoria_Razascaldas(operation, stamp, userid,nombre,cantidad)
SELECT 'I',now(),user(),NEW.jugador1,NEW.cantidad1;
else
INSERT INTO auditoria_Razascaldas(operation, stamp, userid,nombre,cantidad)
SELECT 'I',now(),user(),NEW.jugador2,NEW.cantidad2;
end if;
end;
DROP table elfo_de_la_noche;

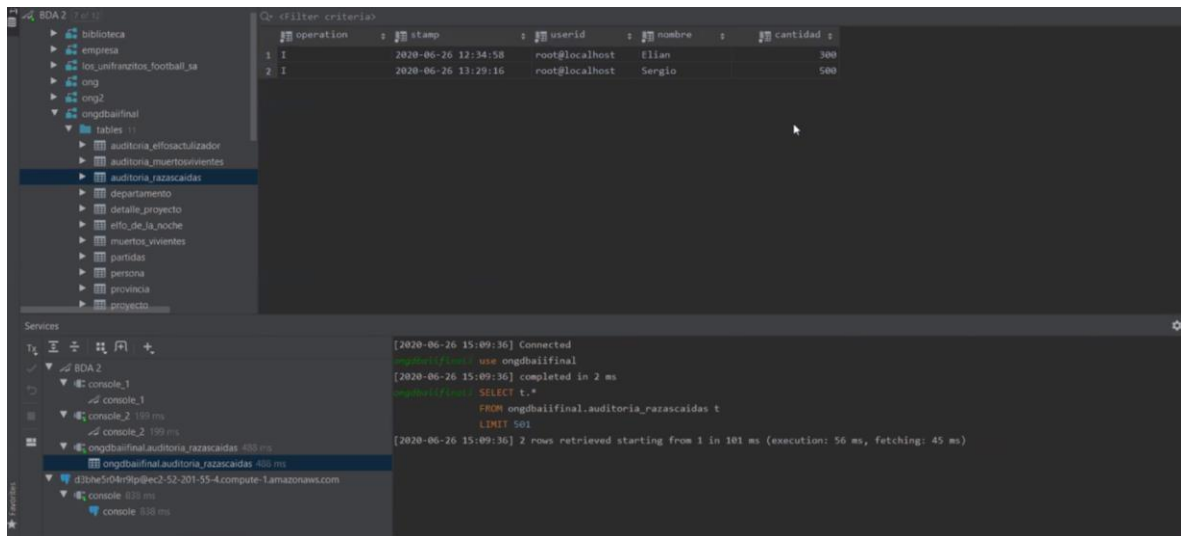
```

Services Window:

```

[2020-06-26 13:26:49] completed in 119 ms
[2020-06-26 13:26:50] CREATE PROCEDURE InsertarMuestrasVivientes(
IN: operation CHAR(1),
IN: NombreTipoBefore TEXT,
IN: NombreTipoAfter TEXT
)
BEGIN
INSERT INTO auditoria_muertosviviendes(operation, stamp, userid, hostname,nombreTipoBefore,nombreTipoAfter)
SELECT operation, now(), user(), @@hostname,NombreTipoBefore,NombreTipoAfter;
end
[2020-06-26 13:26:58] completed in 180 ms

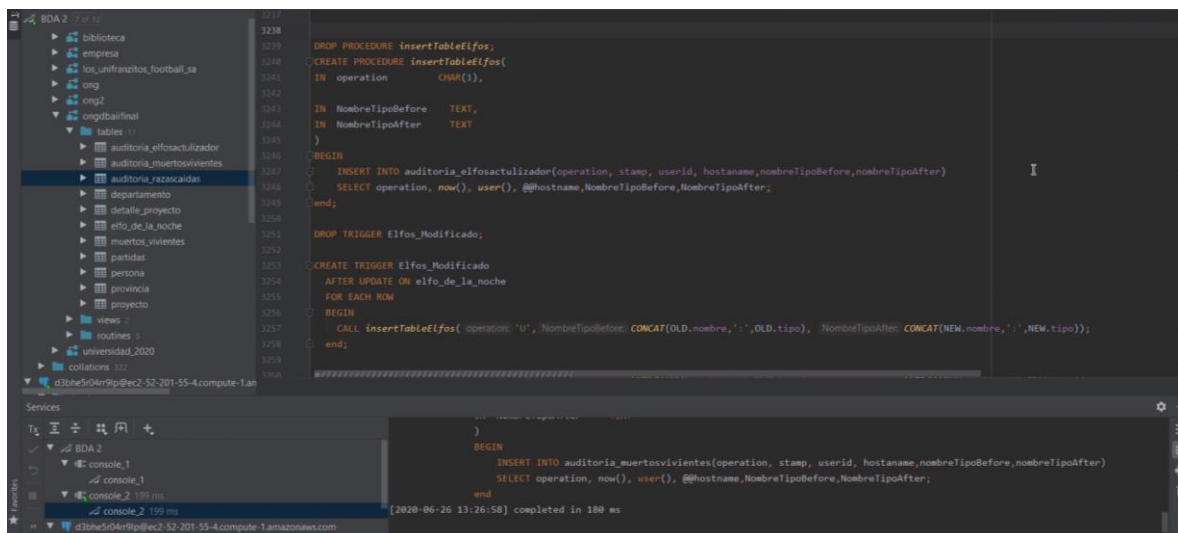
```



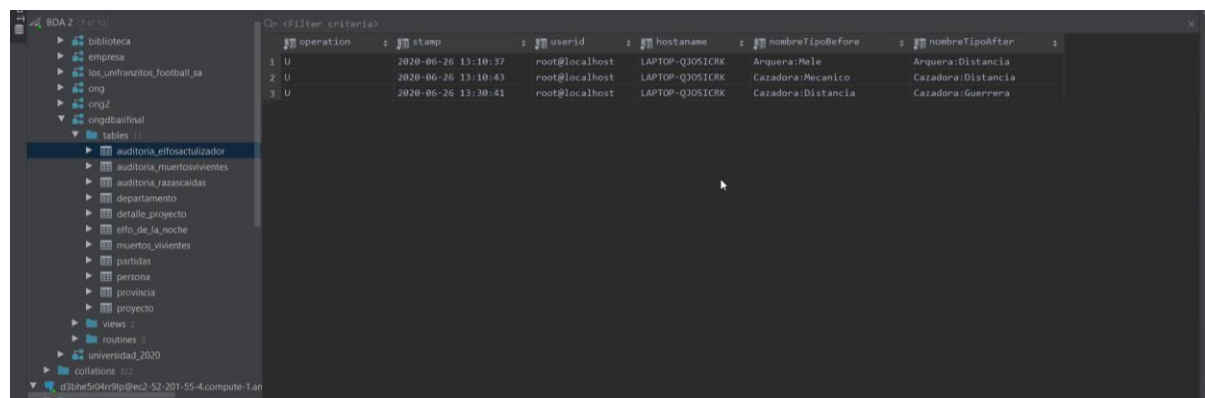
Como resultado nos mostrara el nombre y la cantidad de unidades con las que perdió en la anterior partida al igual que su nombre ingresado

Creamos un procedimiento almacenado para controlar que cambios ubo en cada unidad.

Una disminución en sus datos como también aumento



Como resultado nos guarda después de realizar el cambio que se cambio la fecha y a que dato cambio cada uno.



The screenshot displays a SQL IDE interface. On the left, a database schema tree shows a database named 'ongdbfinal' with a table 'auditoria_efloscualizador'. The central pane contains a SQL script for creating a procedure and a trigger. The script is as follows:

```

1298 CREATE PROCEDURE insertTablaMuertos_Vivientes(
1299     IN operation CHAR(1),
1300     IN NombreTipoBefore TEXT,
1301     IN NombreTipoAfter TEXT
1302 )
1303 BEGIN
1304     INSERT INTO auditoria_muertosviviendes(operation, stamp, userid, hostname,nombreTipoBefore,nombreTipoAfter)
1305     SELECT operation, now(), user(), @@hostname,NombreTipoBefore,NombreTipoAfter;
1306 end;
1307
1308 DROP TRIGGER MuertosViviendes_Borrador;
1309
1310 CREATE TRIGGER MuertosViviendes_Borrador
1311 AFTER DELETE ON muertos_viviendes
1312 FOR EACH ROW
1313 BEGIN
1314     DECLARE res TEXT DEFAULT 'Empty Value - Delete Action';
1315     CALL insertTablaMuertos_Vivientes(operation: 'D', NombreTipoBefore: CONCAT(OLD.nombre, '-', OLD.tipo), NombreTipoAfter: res);
1316 end;
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332

```

A semi-transparent overlay with the text "Press ESC or double-click to exit full screen mode" is visible over the script. The bottom pane shows a log of database services, including the execution of the script and its completion time: [2020-06-26 13:26:58] completed in 180 ms.

4. BDA2 > 10/10

- ▶ biblioteca
- ▶ empresa
- ▶ los_unifanitos_football_sa
- ▶ ong
- ▶ ong2
- ▶ ongdbaifinal
 - ▶ tables (1)
 - ▶ auditoria_effosactualizador
 - ▶ auditoria_muertosviviendes
 - ▶ auditoria_razascaldas
 - ▶ departamento
 - ▶ detalle_proyecto
 - ▶ elfo_de_la_noche
 - ▶ muertos_vivientes
 - ▶ partidas
 - ▶ persona
 - ▶ provincia
 - ▶ proyecto
 - ▶ views (1)
 - ▶ routines (1)
- ▶ universidad_2020
- ▶ collations (1)

db3he504r9ip@ec2-52-201-55-4.compute-1.amazonaws.com

services

- ▶ console_2 (199 ms)
- ▶ ongdbaifinal.auditoria_muertosviviendes (169 ms)
- ▶ db3he504r9ip@ec2-52-201-55-4.compute-1.amazonaws.com
- ▶ console (338 ms)

Q: filter criteria

	operation	stamp	userid	hostname	nombreTipoBefore	nombreTipoAfter
1	D	2020-06-26 13:27:12	root@localhost	LAPTOP-Q305ICRK	Necrofago:Male	Empty Value - Delete Action
2	D	2020-06-26 13:31:38	root@localhost	LAPTOP-Q305ICRK	Gargola:Volador	Empty Value - Delete Action


```

[2020-06-26 15:22:21] completed in 1 ms
ongdbaifinal: SELECT t.*
FROM ongdbaifinal.auditoria_muertosviviendes t
LIMIT 501
[2020-06-26 15:22:21] 2 rows retrieved starting from 1 in 57 ms (execution: 4 ms, fetching: 53 ms)

```