



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 RODDY BARRA

fecha: 26/06/2020



Introducción

En este proyecto presentamos un simulador para Warcraft creado en intelleg 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



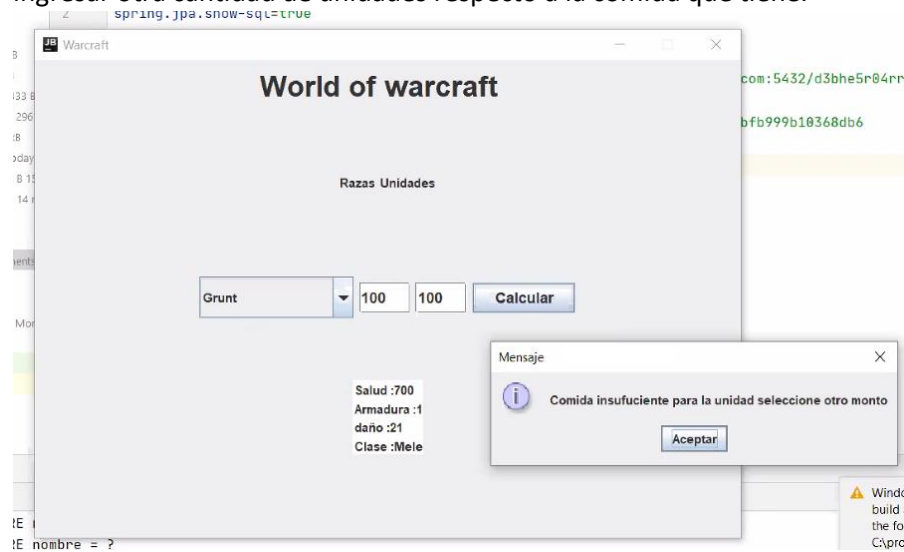
- Al momento de escoger la raza solo podremos seleccionar las unidades de esa raza seleccionada y nos saldrá información de esa unidad.

The screenshot shows a web application window 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

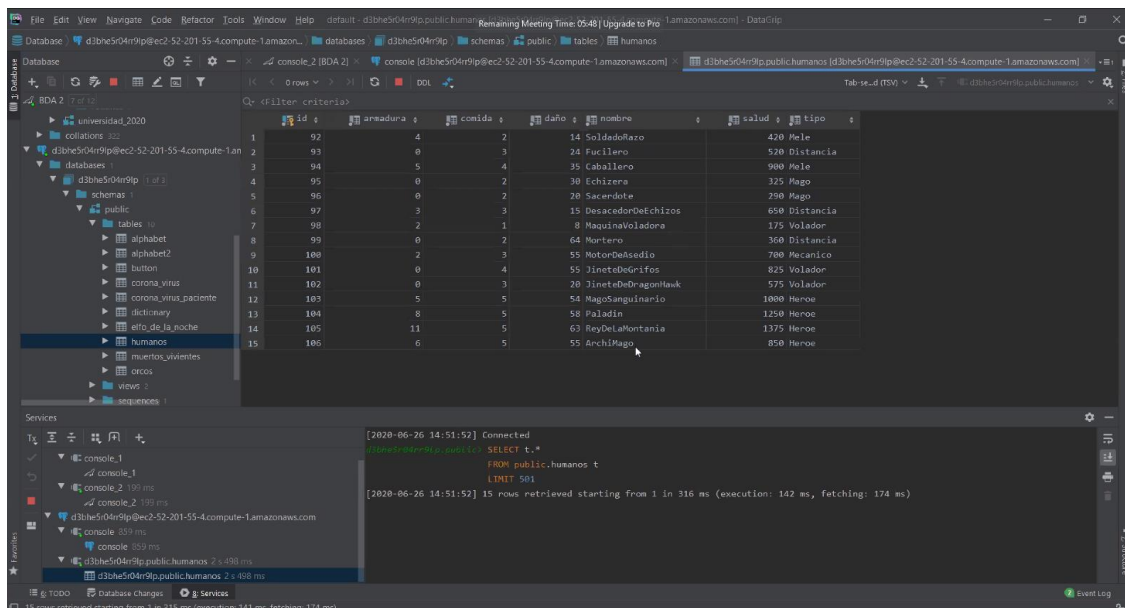
The screenshot shows the same "World of Warcraft" application window. In this state, the first dropdown menu is set to "Demoledor". The second dropdown menu is empty. The text input field now contains the number "3", and the "Calcular" button is highlighted. A tooltip below the input fields shows the updated 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:

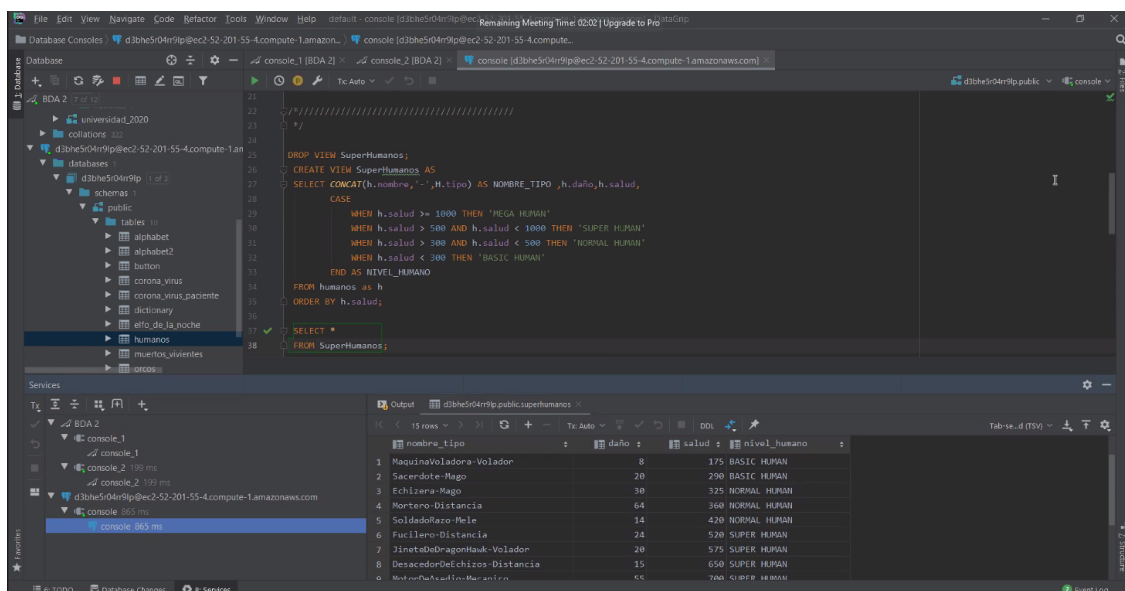


The screenshot shows the DataGrip interface with a SQL query executed in the console. The query selects data from the 'public.humanos' table, limited to 50 rows. The results are displayed in a table with the following columns: id, armadura, comida, daño, nombre, salud, and tipo.

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	DesacedorDeIchizos	650	Distancia
98	2	1	8	MaquinaVoladora	175	Volador
99	0	2	64	Montero	360	Distancia
100	2	3	55	MotorDeSedio	700	Mecanico
101	0	4	55	JineteDeGifos	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', and 'BASIC HUMAN' based on their 'salud' (health) 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
MaquinaVoladora-Volador	8	175	BASIC HUMAN
Sacerdote-Mago	20	290	BASIC HUMAN
Echizera-Mago	30	325	NORMAL HUMAN
Montero-Distancia	64	360	NORMAL HUMAN
SoldadoRazo-Mele	14	420	NORMAL HUMAN
Fucilero-Distancia	24	520	SUPER HUMAN
JineteDeDragonHawk-Volador	20	575	SUPER HUMAN
DesacedorDeIchizos-Distancia	15	650	SUPER HUMAN
MotorDeSedio-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;

```

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 an output window at the bottom right.

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;

```

Output Window:

```

[2020-06-26 13:26:49] completed in 119 ms
[2020-06-26 13:26:58] completed in 180 ms

```

operation	stamp	userid	nombre	cantidad
1 I	2020-06-26 12:34:58	root@localhost	Elian	300
2 I	2020-06-26 13:29:16	root@localhost	Sergio	500

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

```

DROP PROCEDURE insertTableElfos;
CREATE PROCEDURE insertTableElfos(
  IN operation CHAR(1),
  IN NombreTipoBefore TEXT,
  IN NombreTipoAfter TEXT
)
BEGIN
  INSERT INTO auditoria_elfosactualizador(operation, stamp, userid, hostname,nombreTipoBefore,nombreTipoAfter)
  SELECT operation, now(), user(), @@hostname,NombreTipoBefore,NombreTipoAfter;
end;

DROP TRIGGER Elfos_Modificado;
CREATE TRIGGER Elfos_Modificado
AFTER UPDATE ON elfo_de_la_noche
FOR EACH ROW
BEGIN
  CALL insertTableElfos( operation 'U', NombreTipoBefore: CONCAT(OLD.nombre,' ',OLD.tipo), NombreTipoAfter: CONCAT(NEW.nombre,' ',NEW.tipo));
end;

```

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

operation	stamp	userid	hostname	nombreTipoBefore	nombreTipoAfter
1 U	2020-06-26 13:10:37	root@localhost	LAPTOP-Q305ICRX	Arquera:Mele	Arquera:Distancia
2 U	2020-06-26 13:10:43	root@localhost	LAPTOP-Q305ICRX	Cazadora:Mecanico	Cazadora:Distancia
3 U	2020-06-26 13:30:41	root@localhost	LAPTOP-Q305ICRX	Cazadora:Distancia	Cazadora:Guerrera

El ultimo procedimiento almacenado realiza una eliminación de los datos para guardar lo que se elimino y cuando se elimino

```

1289 CREATE PROCEDURE insertTableMuertos_Vivientes(
1290     IN operation CHAR(1),
1291     IN NombreTipoBefore TEXT,
1292     IN NombreTipoAfter TEXT
1293 )
1294 BEGIN
1295     INSERT INTO auditoria_muertosvivos(operation, stamp, userid, hostname,nombreTipoBefore,nombreTipoAfter)
1296     SELECT operation, now(), user(), @@hostname,NombreTipoBefore,NombreTipoAfter;
1297 END;
1298
1299 DROP TRIGGER MuertosVivos_Borrador;
1300
1301 CREATE TRIGGER MuertosVivos_Borrador
1302 AFTER DELETE ON muertos_vivos
1303 FOR EACH ROW
1304 BEGIN
1305     DECLARE res TEXT DEFAULT 'Empty Value - Delete Action';
1306     CALL insertTableMuertos_Vivientes( operation: 'D', NombreTipoBefore: CONCAT(OLD.nombre,' ',OLD.tipo), NombreTipoAfter: res);
1307 END;
1308
1309
1310
1311
1312

```

Services

console_2 199 ms

[2020-06-26 13:26:58] completed in 180 ms

Como resultado nos dará que unidad se eliminó y cuando

Query (Filter criteria)

operation	stamp	userid	hostname	nombreTipoBefore	nombreTipoAfter
D	2020-06-26 13:27:12	root@localhost	LAPTOP-Q305ICR8	Necrofago:Mele	Empty Value - Delete Action
D	2020-06-26 13:31:38	root@localhost	LAPTOP-Q305ICR8	Gargola:Volador	Empty Value - Delete Action

Services

console_2 199 ms

[2020-06-26 15:22:21] completed in 1 ms

ongdbaiifinal: SELECT t.* FROM ongdbaiifinal.auditoria_muertosvivos t LIMIT 501

[2020-06-26 15:22:21] 2 rows retrieved starting from 1 in 57 ms (execution: 4 ms, fetching: 53 ms)