Chocolate Doom

Proyecto: Entrega #2

Ivan Santiago Lastra Romero.
Ana Maria Murcia Gomez.
Martin Sanmiguel Delgado.
Lucas Fuentes Sanchez.

Pontificia Universidad Javeriana.

Facultad de Ingeniería.

Bases de Datos.

Profesor: Andrés Oswaldo Calderón.

10 de noviembre de 2025

Índice:

1. Introducción de continuación de proyecto:

Las bases de datos son una de las herramientas más importantes en la Ingeniería de Sistemas, ya que permiten almacenar, organizar y manipular información de manera estructurada, consistente y eficiente. En este campo, el modelo relacional ha sido el más robusto y empleado para asegurar la integridad y consistencia de los datos, al representarlos en tablas enlazadas por medio de claves y restricciones. Además, el lenguaje SQL y los diagramas Entidad-Relación (E-R) ofrecen las herramientas conceptuales y prácticas para modelar, implementar y manipular tales sistemas de datos.

En la primera parte del proyecto Chocolate Doom se modeló conceptual y lógicamente la base de datos a partir de los datos recogidos de una versión alterada del juego Doom. Esta adaptación hizo posible guardar datos de telemetría del jugador en tiempo real (tics), como por ejemplo momentum x,y dirección en x,y,z, ángulo , etc. Además, se estructuró la base para guardar las tablas requeridas bajo el modelo relacional con cada especificación y además el diccionario con datos especificando los datos utilizados

En esta segunda parte, se crea físicamente el modelo diseñado, creando la base de datos en PostgreSQL a partir del diagrama E-R normalizado hasta la 3FN, donde se modificó levemente para establecer un modelo más simple contemplando los comentarios hechos dado el tiempo y la relación con los datos. Luego, se importaron a la base de datos los archivos .tsv con más de 23.000 tuplas de telemetría recopiladas en sesiones de juego por los miembros del equipo, donde se repartieron de manera equitativa para establecer un conocimiento del juego y proceder a llenar las tuplas requeridas en la base de datos. Además, se les administró el instrumento PENS a los cuatro participantes reflejando el ámbito de evaluación de UX en la base de datos, cuyos resultados se plasmaron en las tablas correspondientes, respetando las reglas de integridad y relaciones previamente establecidas en el esquema, esto será tratado de forma organizada en el documento de reporte estableciendo paso a paso todos los elementos desarrollados.

2. Contenido.

En el contenido se tratará todo lo relacionado al paso a paso que se realizó para cumplir con los 4 requerimientos establecidos en el enunciado, desde la recolección de datos de Chocolate Doom en archivos .tsv que se subieron al GitHub; para más detalle, se sugiere hacer clic aquí para dirigirse al GitHub con los elementos establecidos. Con esto se analizó cada dato crudo de telemetría disponible con los datos esperados y elementos especificados. Cabe aclarar que, al lado de cada archivo, hay elementos con la letra del nombre de la persona que lo realizó y todas las tuplas almacenadas con detalle de mapas o elementos disponibles según el plan.

Con esto en cuenta, se hizo un análisis de telemetría donde se analizan los datos crudos y su relación con la normalización hecha y el nuevo modelo E-R que se deriva de la primera entrega y el utilizado para crear las tablas. Siguiendo con el detalle de la encuesta realizada por parte del grupo, que es descrita como PENS, la cual se incluyó y detalló cada puntaje en el reporte y en la base de datos.

Finalmente, se detalla al final el proceso de creación de tablas en PostgreSQL, llenado de cada una de las tablas dependiendo de los datos disponibles con jugadores, usuarios y elementos disponibles, y el llenado de las 23k tuplas generadas en los archivos .tsv mediante la creación de una tabla temporal que almacena los datos derivados de las tuplas generadas y luego, mediante inserts, se establecen en la tabla destinada a obtener los datos de telemetría que se relaciona con su jugador y juego respectivamente, para el futuro análisis de los datos obtenidos y las futuras búsquedas a realizar dentro de la base de datos.

2.1: Recolección de datos de Chocolate Doom.

Para la recolección de datos como primer paso se definió el uso de los primeros 4 mapas ("Entryway", "Underhalls", "The Gauntlet", "The Focus") del primer episodio de Chocolate Doom 2. Para el proceso en sí se utilizó una máquina virtual de la empresa Oracle(VirtualBox) con un sistema operativo Ubuntu, con el objetivo de facilitar el proceso de captura de la información generada durante el juego. Primero, se descargó desde el repositorio de GitHub proporcionado por el profesor, el cual contenía los archivos del juego Chocolate Doom debidamente modificados. Posteriormente, mediante el comando mostrado en la Figura 1, se ejecutó el videojuego en una ventana reducida de 1300x1100 píxeles. Durante la partida, los datos de posición y eventos fueron almacenados automáticamente en un archivo con formato TSV nombrado “telemetria#.tvs”, donde el # identifica la sesión de juego correspondiente.

Cada tupla generada del juego representa el estado del jugador en cada tic/instante de tiempo. Como se ve en la Figura #1, los datos generados en la duración de la partida anuncian primero el número de episodio y de mapa, y luego forman 9 columnas(una donde no se especifica dato o elemento, pero que se tendrá en cuenta para meterlo a la base de datos), de las cuales tendremos en cuenta las 8 siguientes con su debido elemento conectado:

-Timestamp, tiempo en el que se está jugando la partida, año, mes y día.

-Tic, instante temporal en el que se modelan los datos de juego, 1 tic es aproximadamente 1.35 segundos.

-X, coordenada X del jugador en el mapa.

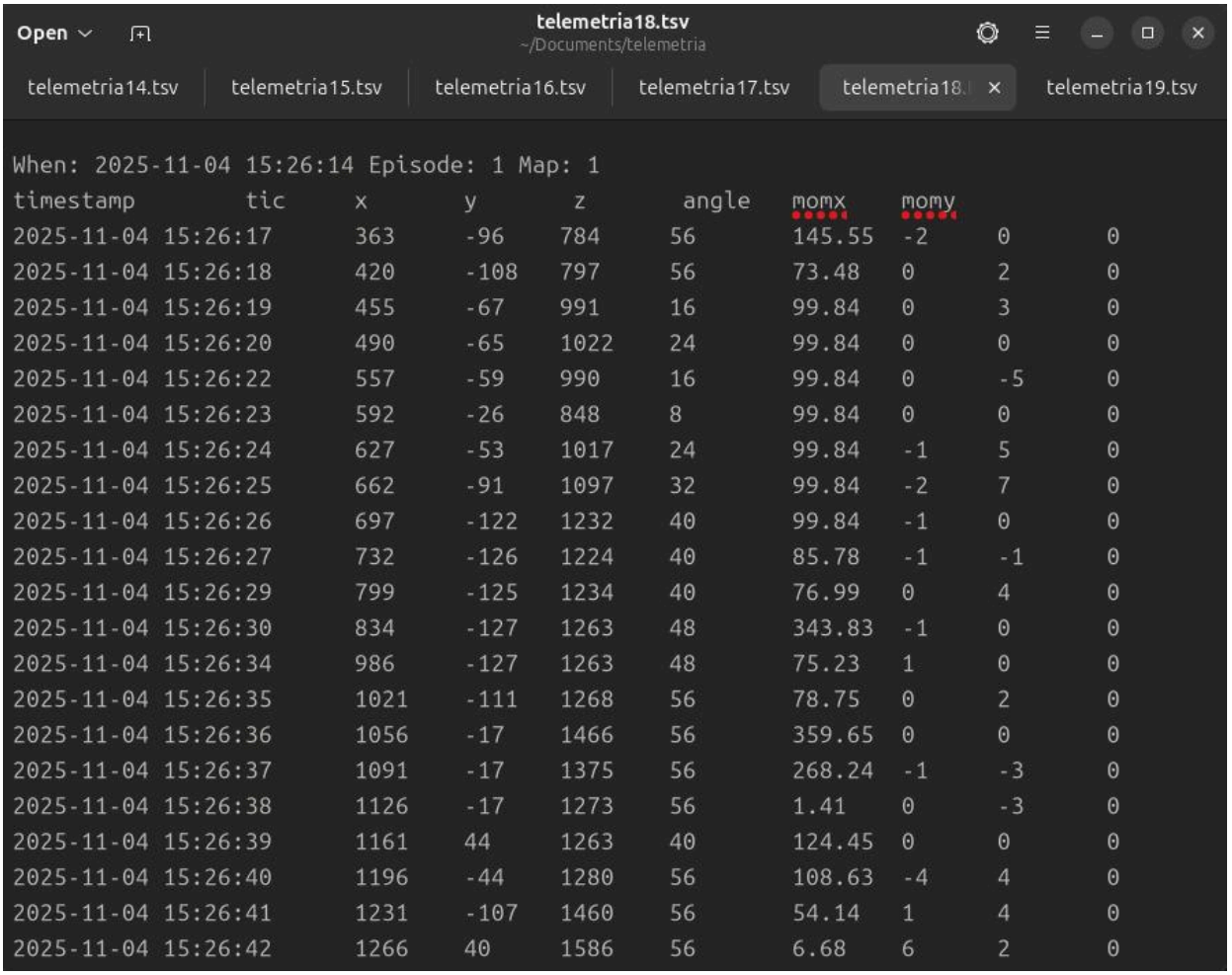
-Y, coordenada Y del jugador en el mapa.

-Z, coordenada Z/altura del jugador en el mapa (esta se da debido a que la manera en la que el juego está diseñado los niveles están posicionados por altura, uno encima del otro).

-Ángulo, ángulo de orientación del jugador, es decir la posición de la cámara o hacia donde mira el usuario en el juego , medido desde 0° a 360°.

-Momx, se refiere al momentum en x, es decir, velocidad o impulso del jugador sobre el eje X y cuánto se mueve horizontalmente. Positivo o negativo según dirección(adelante/atrás).

-Momy, se refiere al momentum en y, es decir, velocidad o impulso del jugador sobre el eje Y y cuanto se mueve verticalmente. Positivo o negativo según dirección(arriba/abajo).



When: 2025-11-04 15:26:14 Episode: 1 Map: 1

timestamp	tic	x	y	z	angle	momx	momy		
2025-11-04	15:26:17	363	-96	784	56	145.55	-2	0	0
2025-11-04	15:26:18	420	-108	797	56	73.48	0	2	0
2025-11-04	15:26:19	455	-67	991	16	99.84	0	3	0
2025-11-04	15:26:20	490	-65	1022	24	99.84	0	0	0
2025-11-04	15:26:22	557	-59	990	16	99.84	0	-5	0
2025-11-04	15:26:23	592	-26	848	8	99.84	0	0	0
2025-11-04	15:26:24	627	-53	1017	24	99.84	-1	5	0
2025-11-04	15:26:25	662	-91	1097	32	99.84	-2	7	0
2025-11-04	15:26:26	697	-122	1232	40	99.84	-1	0	0
2025-11-04	15:26:27	732	-126	1224	40	85.78	-1	-1	0
2025-11-04	15:26:29	799	-125	1234	40	76.99	0	4	0
2025-11-04	15:26:30	834	-127	1263	48	343.83	-1	0	0
2025-11-04	15:26:34	986	-127	1263	48	75.23	1	0	0
2025-11-04	15:26:35	1021	-111	1268	56	78.75	0	2	0
2025-11-04	15:26:36	1056	-17	1466	56	359.65	0	0	0
2025-11-04	15:26:37	1091	-17	1375	56	268.24	-1	-3	0
2025-11-04	15:26:38	1126	-17	1273	56	1.41	0	-3	0
2025-11-04	15:26:39	1161	44	1263	40	124.45	0	0	0
2025-11-04	15:26:40	1196	-44	1280	56	108.63	-4	4	0
2025-11-04	15:26:41	1231	-107	1460	56	54.14	1	4	0
2025-11-04	15:26:42	1266	40	1586	56	6.68	6	2	0

Figura #1: Datos crudos de telemetría.

A continuación, en la Tabla #1 se ve reflejada la distribución de datos y de archivos .tsv, en la cual cada participante del grupo ayudó a recolectar para establecer la cantidad de tuplas del juego. Se recolectaron poco más de 23k tuplas de información cruda de estadísticas, cumpliendo el requerimiento de 20k establecido, con más de 6 juegos, donde en total se establecieron 23 juegos distintos hasta el mapa 4. Como se establece anteriormente, se muestran las tuplas de cada jugador generadas y sus archivos correspondientes.

	Ana	Martin	Ivan	Lucas
Nombre de archivo	-telemetria1A.tsv -telemetria10.tsv hasta telemetria18.tsv	telemetriaM.tsv -telemetria1M.tsv	-telemetria.tsv -telemetria2.tsv hasta telemetria9.tsv	-telemetria K.tsv -telemetria 2K.tsv
Tuplas	10,698	2678	2278	7605
Total Tuplas	23,259			

Tabla #1. Distribución de recolección de datos.

Para ejecutar en la terminal el juego y su almacenamiento en archivos .tsv, los cuales posteriormente fueron subidos al repositorio en GitHub de forma homogénea, en la Figura #2 se muestra el comando utilizado en Ubuntu como ejemplo. Este comando permite iniciar el juego y comenzar la generación automática de tuplas de telemetría, tal como se refleja en la Figura #1. De esta manera se establece una forma consistente de jugar, registrar y exportar los datos crudos que posteriormente serían utilizados para la estructuración y carga dentro de la base de datos.

```
ana@ana-VirtualBox:~/trajectory_doom/chocolate-doom$ src/chocolate-doom -iwad ../misc/OriginalW
ADs/DOOM2.WAD -geometry 1300x1100 2>&1 | tee -a ~/Documents/telemetria/telemetria19.tsv
```

Figura #2: comando utilizado para jugar y recolectar los datos de telemetría.

Para transferir los datos de telemetría de las partidas a la base de datos desarrollada en PostgreSQL, primero se definió el esquema de la tabla; se creó una tabla con 9 columnas, las cuales se asociaron con su tipo de dato más apropiado.

De esta forma, la base de datos queda preparada para recibir la información en un formato estructurado, como el de los archivos .tsv en los que se recolectó los datos.

El método más rápido y eficiente para cargar grandes volúmenes de datos en PostgreSQL es haciendo uso del comando copy con la unión de todos los .tsv en un archivo unico llamado all_merged.tsv donde se muestra todo el dato crudo desde el entorno de administración psql, el cual permite importar directamente un archivo localmente hacia la base de datos con la tabla temporal a ser especificada más adelante, así como la máquina local, a una tabla. Por medio de este comando fue que los archivos guardados de manera local fueron insertados en la tabla temporal/base de datos.

Durante la carga, se realizaron tareas de limpieza y validación de los datos. Entre ellas, se eliminaron los espacios en blanco, se convirtieron los valores no numéricos en nulos, se revisaron los rangos de coordenadas y velocidades, y se aseguró que las marcas de tiempo estuvieran en el formato adecuado, igualmente se eliminaron datos basura como errores que interfieren en la lectura de las tuplas por parte de psql en la tabla temporal. Aunque la telemetría pertenece a distintas partidas y jugadores, en este caso no se utilizaron llaves para diferenciarlos entre ellos, ya que se consideró innecesario para el desarrollo del ejercicio.

2.2: Análisis de Telemetría.

La telemetría se refiere a la recolección automática de datos a partir de sensores o procesos en tiempo real, con el fin de medir y registrar diferentes variables de un sistema. En este contexto, el movimiento del jugador en Doom 2, en cuyo caso los datos telemétricos provienen de la ejecución de las partidas, posiciones del jugador, coordenadas del entorno, velocidad, momentum, dirección del movimiento, tiempo transcurrido, entre otros parámetros que describen en gran detalle el comportamiento del jugador dentro del juego.

Una base de datos relacional, como PostgreSQL, ofrece la infraestructura necesaria para organizar y estructurar los datos de telemetría. En lugar de guardar los datos en archivos de texto o planillas, una base de datos permite representarlos mediante tablas interrelacionadas, las cuales garantizan integridad, consistencia y rendimiento en las consultas. Adicionalmente, el modelo relacional facilita que los datos puedan ser asociados con otras entidades del proyecto, como jugadores, niveles, armas o eventos específicos del juego. De esta forma, la telemetría no solo se almacena, sino que se convierte en información útil para análisis posteriores sobre la misma información.

Aplicar la normalización a los datos telemétricos de una base de datos tiene un impacto muy positivo, ya que reduce la redundancia, cosa que disminuye el tamaño total de los datos. De igual forma, mejora la integridad, al usar claves foráneas por las que se puedan relacionar múltiples tablas evitando errores como referencias inexistentes, o más comúnmente la redundancia de datos. Por último, facilita las consultas analíticas, ya que con un esquema bien normalizado se pueden realizar uniones entre tablas para obtener informes detallados sobre cualquier información a la que se desee acceder.

Con los datos establecidos se puede analizar la homogeneidad y coherencia de la información obtenida, tanto en términos de la duración y comportamiento de cada sesión de juego como de los elementos propios del estilo y desempeño de cada jugador (movimiento, interacción con el entorno, progresión por el mapa, uso de recursos, entre otros). Esto se evidencia de manera implícita en la Figura #1 y se encuentra detallado en cada uno de los archivos .tsv presentes en el repositorio de GitHub asociado al proyecto.

La existencia de esta información cruda y estructurada es fundamental, ya que constituye la base para los procesos posteriores de análisis dentro de la base de datos relacional. A partir de estas tuplas almacenadas, es posible formular consultas y construir vistas que permitan visualizar patrones de juego, relaciones entre variables, métricas de rendimiento y aspectos conductuales.

Gracias a este diseño, se pueden realizar análisis complejos sin comprometer la integridad ni eficiencia del sistema, se pueden hacer consultas de todo tipo y el rendimiento no se verá afectado de ninguna manera.

2.3: PENS, Encuesta UX.

Después de haber jugado DOOM 2, se realizó un formulario llamado PENS v1.6 – Subscale Scoring, que nació del modelo PENS por sus siglas en inglés Player Experience of Need Satisfaction, este modelo nace por la investigación de Ryan, Rigby y Przybylski en 2006 que aplican SDT, the Self Determination Theory, que básicamente es teoría de la psicología sobre la motivación humana, que fue seleccionado.

De esta investigación surgió el instrumento PENS (Player Experience of Need Satisfaction), diseñado para medir cuánto un videojuego satisface esas tres necesidades durante la

experiencia de juego:

- Competence: es un campo en donde se mide si el jugador se sintió eficaz, hábil y capaz de superar los desafíos del juego.
- Autonomy: es la sección que mide si el jugador se sintió con la libertad para tomar decisiones y actuar de forma voluntaria dentro del juego.
- Relatedness: es la parte en donde se evalúa si el jugador siente cercanía, amistad o conexión con otros jugadores o personajes del juego.
- Presence / Immersion: aquí se evalúa si el jugador se siente dentro del mundo del juego, emocional y mentalmente.
- Intuitive Controls: en esta sección se evalúa qué tan fácil y natural se sienten los controles del juago

Este test tiene el objetivo de evaluar cuánto disfruta el jugador el juego y así medir su nivel de satisfacción y de compromiso (Player Experience Of Needs Satisfaction (PENS) – selfdeterminationtheory.org, s. f.).

Así, este formulario fue aplicado a cada uno de los integrantes del grupo con el fin de evaluar qué tanto se disfrutó la experiencia de juego en DOOM. Cada afirmación del instrumento se calificó en una escala de 1 a 7, donde 1 corresponde a estar totalmente en desacuerdo y 7 a estar totalmente de acuerdo.

Posteriormente, estas respuestas fueron registradas y cargadas en la base de datos en PostgreSQL, con el objetivo de permitir su análisis y comparación con los datos de telemetría obtenidos durante las sesiones de juego. De esta manera, se logrará observar posibles relaciones entre la experiencia subjetiva del jugador y su comportamiento dentro del juego.

A continuación, en la Figura #3 se presenta la tabla con los resultados del instrumento PENS correspondientes a cada integrante, los cuales fueron asociados y almacenados de forma estructurada en la base de datos, relacionando cada respuesta con su respectivo ítem.

Subescala	Ítem	Pregunta	Ana	Lucas	Iván	Martin
Competence	1	I feel competent at the game.	6	4	4	7
Competence	2	I feel very capable and effective when playing.	6	4	4	7
Competence	3	My ability to play the game is well matched with the game's challenges.	7	3	3	7
Autonomy	1	The game provides me with interesting options and choices.	5	6	6	7
Autonomy	2	The game lets you do interesting things.	4	6	6	7
Autonomy	3	I experienced a lot of freedom in the game.	5	5	5	7
Relatedness	1	I find the relationships I form in this game fulfilling.	6	4	4	4
Relatedness	2	I find the relationships I form in this game important.	5	5	5	4
Relatedness	3	I don't feel close to other players.	7	3	3	6
Presence/Immersion	1	When playing the game, I feel transported to another time and place.	7	4	4	7
Presence/Immersion	2	Exploring the game world feels like taking an actual trip to a new place.	7	5	5	7
Presence/Immersion	3	When moving through the game world I feel as if I am actually there.	6	4	4	5
Presence/Immersion	4	I am not impacted emotionally by events in the game.	5	5	5	2
Presence/Immersion	5	The game was emotionally engaging.	3	3	3	7
Presence/Immersion	6	I experience feelings as deeply in the game as I have in real life.	6	4	4	3
Presence/Immersion	7	When playing the game I feel as if I was part of the story.	7	3	3	3
Presence/Immersion	8	When I accomplished something in the game I experienced genuine pride.	7	5	5	7
Presence/Immersion	9	I had reactions to events and characters in the game as if they were real.	6	6	6	5
Intuitive Controls	1	Learning the game controls was easy.	7	7	7	7
Intuitive Controls	2	The game controls are intuitive.	7	5	5	7
Intuitive Controls	3	When I wanted to do something in the game, it was easy to remember the corresponding control.	7	6	6	7

Figura #3: Resultados de la encuesta UX PENS de Chocolate Doom.

Cabe aclarar que en la base de datos se realizó el llenado con los datos correspondientes mostrados en la Figura #3. Sin embargo, en la misma base se encuentran definidos los tres

instrumentos PENS, BANGS y GUESS, de manera que, si en el futuro se desea aplicar o completar alguno de los dos restantes, ya existe el espacio y la estructura necesaria para almacenar dicha información.

En este caso, se diligenciaron únicamente las respuestas del instrumento PENS, ajustándose directamente a sus subescalas e ítems. Por lo tanto, en la base de datos se registran los puntajes correspondientes a este instrumento, mientras que se deja explícito que las encuestas BANGS y GUESS no han sido diligenciadas, permaneciendo sus campos disponibles para un llenado posterior.

2.4: Creación de Tablas y Llenado de datos.

Finalmente, en este capítulo se establece, a partir del diagrama Entidad–Relación, la creación del DDL y de las tablas correspondientes en PostgreSQL, incluyendo todas las tablas acordadas y los elementos ajustados después de constatar la necesidad de desnormalizar ciertos componentes y excluir entidades que no debían ser consideradas. Además, se realizaron unificaciones de elementos en función de la estructura real de la telemetría y se corrigieron ciclos presentes en el modelo E–R.

De igual forma, se definieron nuevas claves foráneas con el fin de dejar explícita la forma en que se relacionan las entidades dentro de la base de datos implementada en PostgreSQL. Esto permitió consolidar la estructura final a partir de los datos disponibles de entidades como *juego*, *usuario*, *jugador*, *instrumento_ux*, entre otras, que componen el esquema relacional con sus componentes respectivos.

A continuación, en la Figura #4 se presenta el diagrama Entidad–Relación modificado y revisado, donde se observan las tablas definitivas y las relaciones que estructuran la base de datos.

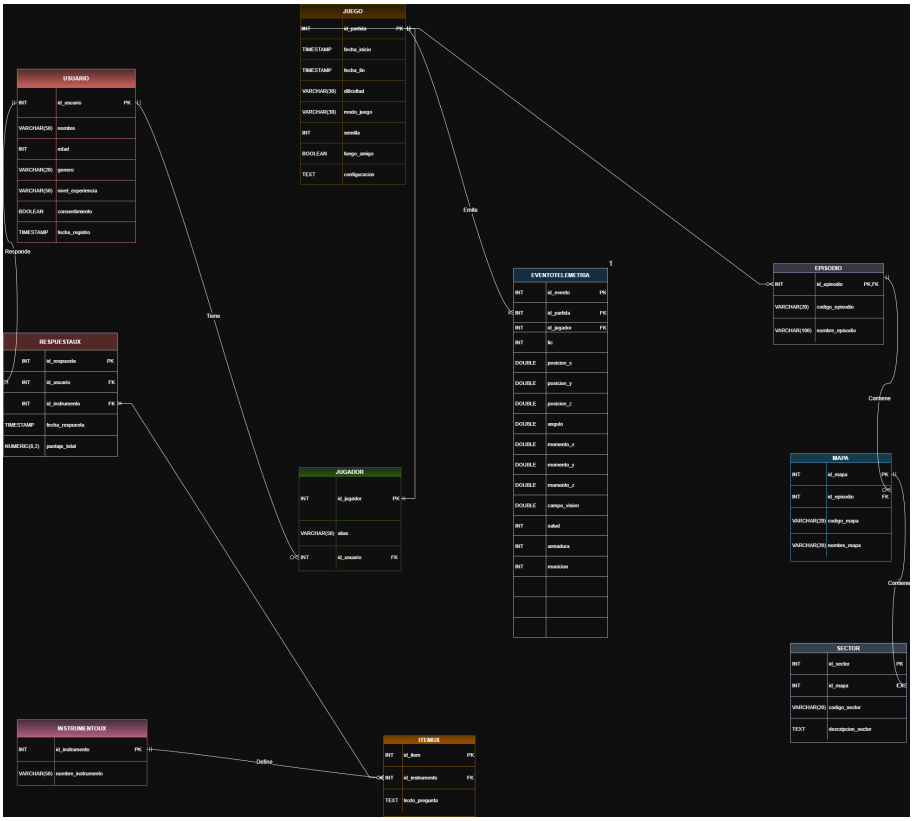


Figura #4: Diagrama Entidad-Relación utilizada.

Con este diagrama se realizaron y crearon las tablas principales que intervienen en la base de datos, con sus relaciones específicas definidas por sus claves foráneas y las restricciones asociadas que se encuentran especificadas en la entrega #1 de Chocolate Doom. Del mismo modo, se establecen en cada entidad sus atributos con sus respectivos tipos de datos y los elementos asociados

En este diagrama Entidad–Relación se deja claro que cada una de las tablas posee una clave primaria, ya sea simple o compuesta, y que el modelo fue desnormalizado con respecto al diseño inicial presentado en la entrega #1, según los datos reales obtenidos y las aclaraciones establecidas anteriormente.

A continuación, se presenta el proceso de creación e inserción de datos para la tabla usuario. En esta etapa se tuvieron en cuenta las especificaciones definidas previamente en el modelo Entidad–Relación y en el diccionario de datos, aplicando las restricciones correspondientes y asignando los atributos definidos para cada usuario. Para esta tabla se insertaron los cuatro integrantes del grupo, cada uno registrado con sus datos personales y características asociadas al proyecto.

Es importante señalar que para la implementación de la base de datos se creó un *schema* llamado doom, el cual permite organizar de manera estructurada todos los objetos del proyecto dentro de PostgreSQL, evitando interferencias con otros esquemas o estructuras existentes en la misma instancia de la base de datos. Con el fin de garantizar que las operaciones realizadas (creación de tablas, inserciones, consultas y demás operaciones) se ejecutarán correctamente dentro del esquema correspondiente, se utilizó de manera permanente el siguiente comando: SET search_path = doom, public;

```
CREATE TABLE usuario (
  id_usuario       INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  nombre           VARCHAR(120) NOT NULL,
  edad             INT,
 género            VARCHAR(2),
nivel_experiencia VARCHAR(50),
consentimiento    BOOLEAN NOT NULL DEFAULT TRUE,
fecha_registro    TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
CONSTRAINT ck_usuario_edad_nonneg CHECK (edad IS NULL OR edad >= 0);
);
INSERT INTO doom.usuario (nombre, edad, genero, nivel_experiencia, consentimiento)
VALUES
('Iván Lastra', 19, 'M', 'intermedio', TRUE),
 ('Ana Murcia', 19, 'F', 'novato',     TRUE),
 ('Lucas Fuentes', 18, 'M', 'experto',    TRUE),
 ('Martin Sanmiguel',   19, 'F', 'intermedio', TRUE);
```

Continuando con la creación de las tablas, a continuación se presenta la creación e inserción de datos de la tabla jugador, la cual se deriva de la tabla usuario asociando a cada usuario un alias dentro del

contexto del videojuego. Esta tabla establece una relación mediante el uso de los identificadores correspondientes, lo que permite mantener la trazabilidad entre el usuario real y su representación dentro del juego.

```
CREATE TABLE jugador (
id_jugador  INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
alias       VARCHAR(60) NOT NULL UNIQUE,
id_usuario  INTEGER REFERENCES usuario(id_usuario) ON DELETE SET NULL
);

INSERT INTO doom.jugador (alias, id_usuario)
SELECT 'Ivancho',  (SELECT id_usuario FROM doom.usuario WHERE nombre='Iván Lastra')
UNION ALL
SELECT 'AnitaLaMasBonita', (SELECT id_usuario FROM doom.usuario WHERE nombre='Ana
Murcia')
UNION ALL
SELECT 'Kukitas',(SELECT id_usuario FROM doom.usuario WHERE nombre='Lucas Fuentes')
UNION ALL
SELECT 'MartinsitoFlansito',   (SELECT id_usuario FROM doom.usuario WHERE
nombre='Martin Sanmiguel')
ON CONFLICT (alias) DO NOTHING;
```

Del mismo modo, se presenta a continuación la creación e inserción de datos para la tabla episodio, en la cual se definen los episodios disponibles en el videojuego junto con sus atributos correspondientes. Esta tabla se encuentra relacionada con la tabla juego, permitiendo identificar a qué episodio pertenece cada partida almacenada en la base de datos.

```
CREATE TABLE episodio (
 id_episodio      INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
 codigo_episodio  VARCHAR(20)  NOT NULL UNIQUE,
nombre_episodio  VARCHAR(120) NOT NULL
);

INSERT INTO doom.episodio (codigo_episodio, nombre_episodio)
VALUES ('1','Episode 1')
ON CONFLICT (codigo_episodio) DO NOTHING;
```

Con esto, se procede a la creación de la tabla *juego*, entidad característica del proyecto, en la cual se registran las partidas realizadas junto con sus atributos y configuraciones correspondientes. Esta tabla almacena información relevante como la dificultad, el modo de juego, la semilla utilizada, el episodio asociado y las condiciones de la partida, consolidando así los datos necesarios para el análisis posterior de la experiencia y desempeño de los jugadores.

```
CREATE TABLE juego (
 id_partida     INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
```

```
fecha_inicio   TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
fecha_fin      TIMESTAMP,
dificultad     VARCHAR(30),
modo_juego     VARCHAR(40),
 semilla        BIGINT,
 limite_tiempo  BOOLEAN DEFAULT FALSE,
 configuracion  TEXT,
id_episodio    INTEGER NOT NULL REFERENCES episodio(id_episodio) ON DELETE
RESTRICT,
CONSTRAINT ck_juego_tiempo CHECK (fecha_fin IS NULL OR fecha_fin >= fecha_inicio)
);
ALTER TABLE juego
 ALTER COLUMN modo_juego SET DEFAULT 'singleplayer',
 ALTER COLUMN configuracion SET DEFAULT 'default';

WITH ep AS (
SELECT id_episodio
 FROM episodio
  WHERE codigo_episodio = '1'
)
INSERT INTO juego (id_episodio, dificultad, fuego_amigo, semilla)
SELECT ep.id_episodio, v.dificultad, FALSE, v.semilla
FROM ep
CROSS JOIN (
 VALUES
 ('I"m Too Young To Die', 1101),
('Hey, Not Too Rough',    1202),
 ('Hurt Me Plenty',        1303),
 ('Ultra-Violence',        1404),
  ('Nightmare!',           1505),

('Hurt Me Plenty',        2301),
('Ultra-Violence',        2402),
 ('Hey, Not Too Rough',    2203),
 ('Nightmare!',           2504),
 ('I"m Too Young To Die', 2105),

('Ultra-Violence',        3401),
 ('I"m Too Young To Die',  3102),
('Hurt Me Plenty',        3303),
('Nightmare!',           3504),
('Hey, Not Too Rough',    3205),

('Nightmare!',           4501),
```

```
 ('Ultra-Violence',        4402),
 ('Hurt Me Plenty',        4303),
('Hey, Not Too Rough',    4204),
('I"m Too Young To Die',  4105),

('Hurt Me Plenty',        5301),
('Nightmare!',            5502),
('Ultra-Violence',        5403)

) AS v(dificultad, semilla)
```

Continuando con la creación y el llenado de datos, se procede con la entidad y tabla mapa, en la cual se registran los diferentes mapas pertenecientes a cada episodio. Esta tabla permite identificar y organizar la estructura espacial del juego, estableciendo la relación entre el episodio correspondiente y los escenarios en los que se desarrollan las partidas.

```
CREATE TABLE mapa (
id_mapa      INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
id_episodio  INTEGER NOT NULL REFERENCES episodio(id_episodio) ON DELETE
RESTRICT,
codigo_mapa  VARCHAR(30)  NOT NULL,
nombre_mapa  VARCHAR(120) NOT NULL,
CONSTRAINT uq_mapa_por_episodio UNIQUE (id_episodio, codigo_mapa)
);


WITH ep AS (SELECT id_episodio FROM doom.episodio WHERE codigo_episodio='1')
INSERT INTO doom.mapa (id_episodio, codigo_mapa, nombre_mapa)
SELECT id_episodio, codigo_mapa, nombre_mapa
FROM (
VALUES
('E1M1', 'Entryway'),
('E1M2', 'Underhalls'),
('E1M3', 'The Gauntlet'),
('E1M4', 'The Focus'),
('E1M5', 'The Waste Tunnels'),
('E1M6', 'The Crusher')
) AS maps(codigo_mapa, nombre_mapa)
JOIN ep ON TRUE

ON CONFLICT (id_episodio, codigo_mapa) DO NOTHING;
```

Prosiguiendo con la construcción del modelo, se llevó a cabo la creación de la tabla *sector*. Sin embargo, al no contar con datos asociados directamente a los sectores dentro de los archivos de telemetría, se decidió no realizar la inserción de datos en esta tabla. No obstante,

la estructura se mantiene definida dentro de la base de datos para permitir su uso en futuras extensiones o análisis que requieran un mayor nivel de detalle espacial dentro de los mapas.

```
CREATE TABLE sector (
 id_sector      INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
id_mapa       INTEGER NOT NULL REFERENCES mapa(id_mapa) ON DELETE CASCADE,
 codigo_sector VARCHAR(40) NOT NULL,
descripcion   TEXT,
CONSTRAINT uq_sector_por_mapa UNIQUE (id_mapa, codigo_sector)
);
```

Posteriormente, se hace hincapié en la creación y llenado de la tabla *instrumento_ux*, la cual define los tres instrumentos que pueden ser utilizados en la base de datos. Sin embargo, como se mencionó anteriormente, para esta entrega únicamente se utilizó el primer instrumento, correspondiente a PENS, dejando los demás instrumentos registrados pero sin datos asociados por el momento.

```
CREATE TABLE instrumento_ux (
 id_instrumento      INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
nombre_instrumento  VARCHAR(80) NOT NULL UNIQUE
);

INSERT INTO instrumento_ux (nombre_instrumento) VALUES
 ('PENS'), ('BANGS'), ('GUESS')
ON CONFLICT (nombre_instrumento) DO NOTHING;
```

Con esto, se procede a la creación y llenado de la entidad *item_ux*, la cual define cada una de las preguntas o ítems que componen el instrumento seleccionado. En esta tabla se almacenan los textos de los ítems asociados al instrumento PENS, permitiendo posteriormente relacionar cada respuesta registrada por los usuarios con el ítem correspondiente dentro de la base de datos.

```
CREATE TABLE item_ux (
 id_item         INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
id_instrumento  INTEGER NOT NULL REFERENCES instrumento_ux(id_instrumento) ON
DELETE CASCADE,
texto_pregunta  TEXT NOT NULL
);

WITH pens AS (
SELECT id_instrumento FROM instrumento_ux WHERE nombre_instrumento = 'PENS'
)
INSERT INTO item_ux (id_instrumento, texto_pregunta)
SELECT id_instrumento, txt FROM pens
CROSS JOIN (
 VALUES
```

```
-- Competence (3)
('I feel competent at the game.'),
('I feel very capable and effective when playing.'),
('My ability to play the game is well matched with the game"s challenges.'),

-- Autonomy (3)
 ('The game provides me with interesting options and choices.'),
('The game lets you do interesting things.'),
 ('I experienced a lot of freedom in the game.'),

 -- Relatedness (3)  -- (3) es reverse-score
('I find the relationships I form in this game fulfilling.'),
('I find the relationships I form in this game important.'),
('I don’t feel close to other players.'),     -- (-) reverse-score

 -- Presence / Immersion (9) -- (4) es reverse-score
('When playing the game, I feel transported to another time and place.'),
 ('Exploring the game world feels like taking an actual trip to a new place.'),
 ('When moving through the game world I feel as if I am actually there.'),
 ('I am not impacted emotionally by events in the game.'), -- (-) reverse-score
('The game was emotionally engaging.'),
 ('I experience feelings as deeply in the game as I have in real life.'),
 ('When playing the game I feel as if I was part of the story.'),
('When I accomplished something in the game I experienced genuine pride.'),
('I had reactions to events and characters in the game as if they were real.'),

-- Intuitive Controls (3)
('Learning the game controls was easy.'),
('The game controls are intuitive.'),
('When I wanted to do something in the game, it was easy to remember the corresponding control.')
) q(txt)
```

Con lo anterior, se procedió al llenado de la tabla respuesta_ux, donde se registraron los puntajes correspondientes a cada uno de los integrantes participantes, tal como se muestra en la Figura #3. En esta tabla se almacenan las respuestas asociadas a cada ítem del instrumento PENS, permitiendo identificar de manera clara qué usuario respondió y cuál fue el valor asignado. A continuación, se presenta un ejemplo del proceso de inserción de datos realizado para esta entidad.

```
CREATE TABLE respuesta_ux (
 id_respuesta    INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
id_usuario   INTEGER NOT NULL REFERENCES usuario(id_usuario)
 ON DELETE CASCADE,
 id_instrumento  INTEGER NOT NULL REFERENCES instrumento_ux(id_instrumento) ON
DELETE RESTRICT,
```

```
 fecha_respuesta TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
 puntaje_total   NUMERIC(6,2),
 CONSTRAINT uq_respuesta_unica UNIQUE (id_usuario, id_instrumento, fecha_respuesta)
);

INSERT INTO respuesta_ux (id_usuario, id_instrumento, id_pregunta, puntaje_total)
VALUES (3, 1, 22, 6); – Valores variables dependiendo del usuario con cada usuario.
```

Finalmente, se establece la relación donde se almacenará la telemetría correspondiente a las 23k tuplas de datos generadas durante las partidas. Para poder realizar este proceso de manera adecuada, se creó inicialmente una tabla temporal llamada stg_telemetria, la cual permitió cargar y limpiar los datos crudos extraídos directamente del juego, asegurando que los registros quedaran en un formato secuencial y estructurado.

A continuación, se llevó a cabo la creación tanto de la tabla evento_telemetria, encargada de almacenar los datos definitivos, como de la tabla temporal stg_telemetria. Posteriormente, se realizó la transferencia de los datos desde el archivo consolidado all_merged.tsv hacia la tabla temporal, y desde allí hacia evento_telemetria, garantizando el cumplimiento de las restricciones, relaciones y claves definidas dentro del esquema.

Este proceso permitió dejar la base de datos lista para el análisis posterior, así como para la ejecución de consultas y operaciones analíticas orientadas al estudio del comportamiento de los jugadores y de la dinámica del juego.

```
CREATE TABLE stg_telemetria (

  ts_text   TEXT,

  tic       INTEGER,

  x         DOUBLE PRECISION,

  y         DOUBLE PRECISION,

  z         DOUBLE PRECISION,

  angle     DOUBLE PRECISION,

  momx      DOUBLE PRECISION,

  momy      DOUBLE PRECISION,

  fov_stg   DOUBLE PRECISION – Columna sin significado unicamente para completar datos

);

CREATE TABLE evento_telemetria (
```

```
 id_evento     INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
 id_partida    INTEGER NOT NULL REFERENCES juego(id_partida)   ON DELETE CASCADE,
id_jugador    INTEGER NOT NULL REFERENCES jugador(id_jugador) ON DELETE
CASCADE,
 tic          INTEGER NOT NULL,
posicion_x    DOUBLE PRECISION,
posicion_y    DOUBLE PRECISION,
posicion_z    DOUBLE PRECISION,
angulo        DOUBLE PRECISION,
momentum_x    DOUBLE PRECISION,
momentum_y    DOUBLE PRECISION,
campo_vision  DOUBLE PRECISION,
 salud        INTEGER,
armadura      INTEGER,
 municion     INTEGER,
CONSTRAINT ck_salud_nonneg    CHECK (salud    IS NULL OR salud    >= 0),
CONSTRAINT ck_armadura_nonneg CHECK (armadura IS NULL OR armadura >= 0),
CONSTRAINT ck_municion_nonneg CHECK (municion IS NULL OR municion >= 0)
);
```

Insertar telemetria en tabla temporal

```
\copy stg_telemetria (ts_text,tic,x,y,z,angle,momx,momy,fov_stg)
FROM 'C:\Users\ivans\OneDrive\Documentos\Doom\Arepas-De-arroz-pelado-DB\all_merged.tsv'
WITH (FORMAT csv, DELIMITER E'\t', HEADER true);
```

Transferir de tabla

```
SET search_path = doom, public;

WITH stg AS (
SELECT
 s.*,
 ROW_NUMBER() OVER (
ORDER BY COALESCE(s.tic, 0), COALESCE(s.ts_text, '')
) AS r
FROM stg_telemetria s
),

juegos AS (
 SELECT
  j.id_partida,
ROW_NUMBER() OVER (ORDER BY j.id_partida) AS rn
 FROM juego j
),
```

```
jug AS (
SELECT alias, id_jugador
FROM jugador
 WHERE alias IN ('Ivancho','AnitaLaMasBonita','Kukitas','MartinsitoFlansito')
),

asignacion AS (
 SELECT
 g.id_partida,
g.rn,
CASE
 WHEN g.rn BETWEEN  1 AND  6 THEN (SELECT id_jugador FROM jug WHERE
alias='Ivancho')
WHEN g.rn BETWEEN  7 AND 12 THEN (SELECT id_jugador FROM jug WHERE
alias='AnitaLaMasBonita')
WHEN g.rn BETWEEN 13 AND 18 THEN (SELECT id_jugador FROM jug WHERE
alias='Kukitas')
ELSE                        (SELECT id_jugador FROM jug WHERE alias='MartinsitoFlansito')
END AS id_jugador
 FROM juegos g
)
INSERT INTO evento_telemetria (
id_partida, id_jugador, tic,
 posicion_x, posicion_y, posicion_z,
angulo, momentum_x, momentum_y, campo_vision,
salud, armadura, municion
)
SELECT
 a.id_partida,
 a.id_jugador,
 stg.tic,
 stg.x, stg.y, stg.z,
 stg.angle, stg.momx, stg.momy, stg.fov_stg,
 0::int AS salud,
0::int AS armadura,
0::int AS municion
FROM stg
JOIN asignacion a
ON ((stg.r - 1) % 23) + 1 = a.rn;
```

Con esto, los datos quedaron almacenados en la tabla *evento_telemetria* y listos para ser analizados. Para finalizar el DDL de la base de datos, se generó un respaldo completo (*backup*) tanto en formato .sql como en formato binario, los cuales se encuentran adjuntos en este documento y también disponibles en el repositorio de GitHub mencionado en la sección de contenido para su revisión.

De esta manera, se habilitan dos métodos para verificar y restaurar la estructura del DDL en PostgreSQL según sea necesario. Finalmente, estos datos y estructuras permitirán abordar la siguiente fase del proyecto, correspondiente a la construcción de consultas analíticas y vistas, con el fin de realizar el análisis de la información y completar la entrega final, incluyendo el desarrollo del componente adicional propuesto.

3. Conclusiones.

Elaborar este proyecto hizo que se pudiera dar un gran paso para poder crear la base de datos para analizar la telemetría y la experiencia del usuario en el juego Chocolate Doom. Basándose en el modelo de la primera entrega, se revisaron, modificaron y ajustaron las entidades y relaciones planteadas, para lograr un modelo lógico y funcional que se ajustara a las necesidades del sistema. El modelo refleja fielmente los principales dominios descritos en el problema: telemetría, experiencia de usuario (UX) y ETL para ingestión y organización de datos.

El diagrama Entidad-Relación final representa la jerarquía espacial, semántica y funcional requerida, enlazando apropiadamente las entidades usuario, jugador, juego, episodio, mapa y evento_telemetria con las entidades correspondientes a las herramientas UX, en este caso PENS. El modelo se adaptó a través de la normalización y posterior desnormalización parcial, según la naturaleza de los datos reales y eliminando ciclos y redundancias. Estas decisiones dieron como resultado una estructura más limpia, precisa y lista para ser implementada en PostgreSQL

La recolección de datos se llevó a cabo en sesiones de juego de los miembros del grupo, generando más de 23.000 tuplas de telemetría en formato .tsv, superando el mínimo exigido. Para manipular estos datos, se usó un proceso ETL estructurado, creando una tabla temporal (stg_telemetria) para limpiar, consolidar y cargar los datos en la tabla final evento_telemetria, manteniendo la integridad referencial y la consistencia semántica. Adicionalmente, se administró la encuesta PENS a los cuatro jugadores, cuyos resultados se almacenaron en las tablas instrumento_ux, item_ux y respuesta_ux, dejando además estructuradas las tablas para los instrumentos BANGS y GUESS en caso de que se requirieran en un futuro.

La base de datos final se guardó tanto en .sql como en binario, para poder replicarse, revisarse y restaurarse en otros lugares. Esta entrega deja la plataforma lista para la siguiente etapa del proyecto, en la que se crearán consultas analíticas, vistas y visualizaciones para entender el comportamiento de los jugadores, la experiencia de usuario y el juego. Además, la estructura elaborada permitirá resolver la parte complementaria (bonus) de manera explícita y justificada.

Referencias:

- Player Experience of Needs Satisfaction (PENS) – selfdeterminationtheory.org. (s. f.). https://selfdeterminationtheory.org/player-experience-of-needs-satisfaction-pens/