Colnexus Reporte Final

Daniel Alejandro Duarte Duarte

Mónica Castro Benítez

Juan Felipe Vela Jiménez

José Alejandro Contreras Obregón

Prof. Andrés Oswaldo Calderón Romero

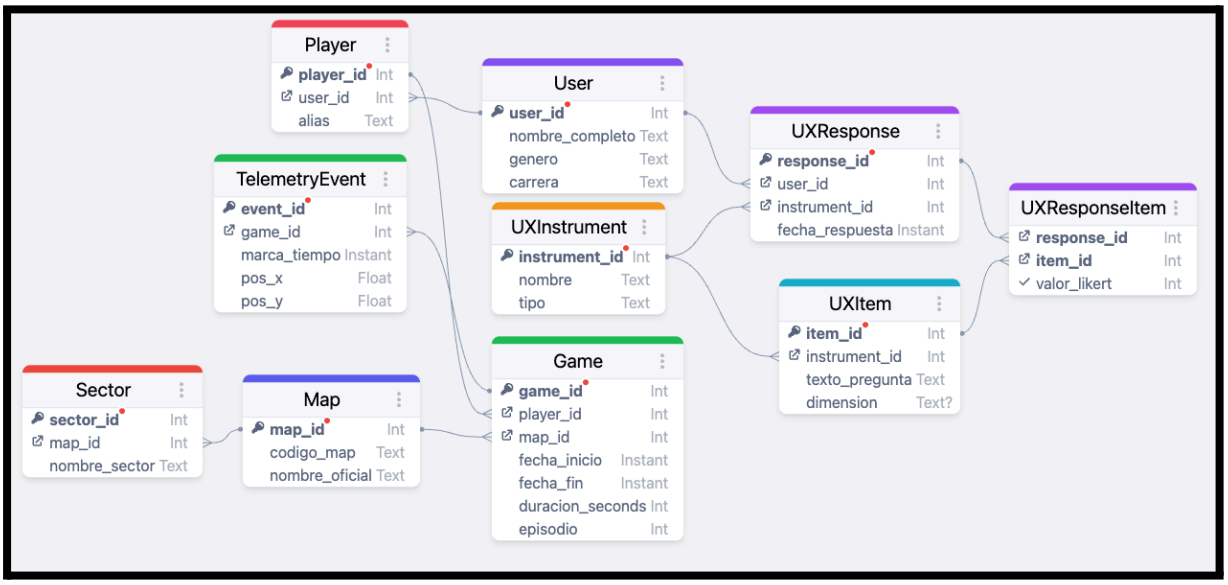
25 de Noviembre de 2025

Introducción

El presente informe detalla el diseño conceptual y lógico para un sistema de base de datos (BD) destinado a recopilar datos de telemetría de juego y experiencia de usuario (UX) generados por el puerto fuente *Chocolate-Doom*. Este proyecto se enmarca dentro de un esfuerzo de investigación para analizar las dinámicas de juego y el rendimiento en un entorno deliberadamente diseñado para emular la tecnología de los años 90 y generar una propuesta que atienda la consultoría la cual nos fue presentada.

1. Diseño de DB

1.1 Diagrama E-R



1.2 Esquema Relacional

Player(<u>player_id</u>, user_id, alias)
User(<u>user_id</u>, nombre_completo, genero, carrera)
UXResponse(<u>response_id</u>, user_id, instrument_id, fecha_respuesta)
UXResponseItem(<u>response_id</u>, item_id, valor_likert)
UXItem(*item_id*, instrument_id, texto_pregunta)
UXInstrument(*instrument_id*, nombre, tipo)
TelemetryEvent(*event_id*, game_id, marca_tiempo, pos_x, pos_y)
Game(*game_id*, player_id, map_id, fecha_inicio, fecha_fin, duracion_seconds, episodios)
Map(*map_id*, codigo_map, nombre_oficial)

Sector(*sector_id*, map_id, nombre_sector)

1.3 Raciocinio

Separación entre telemetría y sesiones:

- GameSession actúa como tabla padre que agrupa todos los *ticks* de telemetría de una partida, lo que facilita consultas por sesión (duración, nivel jugado, jugador).
- TelemetryTick almacena datos por “tic” del juego (posición, acción, número de tick), lo que permite un análisis muy granular de comportamiento durante la partida.

Modelado de niveles (GameLevel)

- Tener una tabla GameLevel (o similar) para los niveles del juego permite normalizar la información de los niveles y evitar duplicación de datos de nivel en cada sesión.
- Además, facilita extender la información del nivel (por ejemplo, nombre, dificultad, diseño) sin repetirlo en cada sesión.

Instrumento UX (cuestionario PENS)

- UXItem representa preguntas o ítems del cuestionario, categorizados por dimensión (autonomía, competencia, etc.), lo que permite mantener la estructura del instrumento en la base de datos.
- UXResponse almacena las respuestas de los jugadores a cada ítem para cada sesión, vinculando la experiencia subjetiva (cuestionario) con la sesión de juego real. El score se guarda y está posiblemente validado con un CHECK para asegurar que esté en un rango válido (por ejemplo, 1–7).

Integridad referencial

- Las claves foráneas (session_id, item_id, level_id) aseguran que los datos de telemetría y respuestas UX siempre estén relacionados con una sesión válida y con ítems válidos del cuestionario.
- Esto previene entradas huérfanas (“orphan rows”) y garantiza consistencia entre tablas.

Flexibilidad y escalabilidad

- Al separar los ticks en su propia tabla, el esquema puede escalar para manejar grandes volúmenes de datos sin sobrecargar la tabla de sesiones.
- También permite construir consultas complejas: por ejemplo, cruzar la posición o tipo de acción en ciertos “ticks” con las puntuaciones del cuestionario para explorar correlaciones entre comportamiento de juego y experiencia del usuario.

Análisis relacional enriquecido

- Tener tanto datos objetivos (telemetría) como subjetivos (respuestas PENS) en la base de datos permite análisis de tipo “¿cómo influye la experiencia de juego en el comportamiento?” o “¿existen patrones de acción para jugadores que reportan más autonomía o competencia?”.
- Permite usar SQL para agregaciones, filtrados, métricas por sesión o por jugador, posibilitando también la generación de dashboards o reportes.

2. DDL Constraints Appendix

A continuación, una serie de fragmentos del archivo db_struct.sql que define tablas, claves primarias y foráneas a modo de ejemplificación para los DDL Constraints:

Figura 1: *Creación de tablas Inicial*

```sql
CREATE TABLE "User" (
  user_id SERIAL PRIMARY KEY,
  nombre_completo TEXT NOT NULL,
  genero TEXT NOT NULL,
  carrera TEXT NOT NULL
);

CREATE TABLE Player (
  player_id SERIAL PRIMARY KEY,
  user_id INTEGER NOT NULL REFERENCES "User"(user_id),
  alias TEXT NOT NULL
);

CREATE TABLE Map (
  map_id SERIAL PRIMARY KEY,
  codigo_map VARCHAR(50) NOT NULL,
  nombre_oficial VARCHAR(200) NOT NULL
);

CREATE TABLE Game (
  game_id SERIAL PRIMARY KEY,
  player_id INTEGER NOT NULL REFERENCES Player(player_id),
  map_id INTEGER NOT NULL REFERENCES Map(map_id),
  fecha_inicio TIMESTAMP NOT NULL,
  fecha_fin TIMESTAMP NOT NULL,
  duracion_seconds INTEGER NOT NULL,
  episodio INTEGER NOT NULL
);
```

Figura: *Creación de tablas Inicial (continuación)*

```sql
CREATE TABLE Sector (
  sector_id SERIAL PRIMARY KEY,
  map_id INTEGER NOT NULL REFERENCES Map(map_id),
  nombre_sector VARCHAR(150) NOT NULL
);

CREATE TABLE TelemetryEvent (
  event_id SERIAL PRIMARY KEY,
  game_id INTEGER NOT NULL REFERENCES Game(game_id),
  marca_tiempo TIMESTAMP NOT NULL,
  pos_x NUMERIC NOT NULL,
  pos_y NUMERIC NOT NULL
);

CREATE TABLE UXInstrument (
  instrument_id SERIAL PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  tipo VARCHAR(50) NOT NULL
);

CREATE TABLE UXItem (
  item_id SERIAL PRIMARY KEY,
  instrument_id INTEGER NOT NULL REFERENCES UXInstrument(instrument_id),
  texto_pregunta TEXT NOT NULL,
  dimension VARCHAR(100)
);

CREATE TABLE UXResponse (
  response_id SERIAL PRIMARY KEY,
  user_id INTEGER NOT NULL REFERENCES "User"(user_id),
  instrument_id INTEGER NOT NULL REFERENCES UXInstrument(instrument_id),
  fecha_respuesta TIMESTAMP NOT NULL,
  respuestas_json TEXT DEFAULT '{}'
);

CREATE TABLE UXResponseItem (
  response_id INTEGER NOT NULL REFERENCES UXResponse(response_id),
  item_id INTEGER NOT NULL REFERENCES UXItem(item_id),
  valor_likert INTEGER NOT NULL CHECK (valor_likert BETWEEN 1 AND 7),
  PRIMARY KEY (response_id, item_id)
);
```

3. Descripción ETL

El proceso ETL implementado para ColNexus sigue las recomendaciones del enunciado oficial del proyecto, las cuales establecen que la ingestión debe seguir un flujo *staging* → *validación* → *carga* en tablas core. En este repositorio, dicho proceso se encuentra implementado principalmente en el archivo *loader_script.cxx* dentro de la carpeta Code.

A continuación se describe el funcionamiento detallado del pipeline ETL, basado exclusivamente en los archivos del repositorio:

3.1 Extract (Extracción de datos crudos)

El motor modificado de Chocolate-Doom genera un archivo de telemetría por sesión de juego utilizando la redirección estándar:

```
chocolate-doom -iwad DOOM.WAD > Data/NombreJugador_IDPartida.txt
```

El archivo es un **log secuencial por tic** que contiene información mínima utilizada por el loader:

- Timestamp del motor (formato YYYY-MM-DD HH:MM:SS)
- Tic del juego
- Posición X
- Posición Y

Adicionalmente, el log incluye líneas meta como:

```
When: 2025-11-22 14:35:10 Episode: 1 Map: 1
```

Las cuales permiten al sistema:

- Identificar la fecha de inicio
- Determinar episodio
- Determinar el mapa
- Calcular la duración total de la partida.

El archivo crudo se deposita en la carpeta `Data/`.

3.2 Transform (Validación, limpieza y normalización)

El archivo *loader_script.cxx* implementa la etapa de transformación de la siguiente forma:

a. Detección de metadatos de sesión

El script utiliza expresiones regulares *(std::regex)* para extraer:

- Fecha de inicio
- Episodio jugado
- Número de mapa
- Primer y último timestamp
- Duración calculada en segundos

b. Lectura y validación línea por línea

Cada línea de telemetría se compara con el patrón:

```
YYYY-MM-DD HH:MM:SS   tic   pos_x   pos_y
```

Se valida que:

- El timestamp sea correcto
- El tic sea numérico
- Las coordenadas estén en rango válido
- La fila no esté vacía ni truncada

c. Normalización

El ETL convierte:

- timestamps tipo “2025-11-22T14:35:12” → “2025-11-22 14:35:12”
- Coordenadas extraídas del motor → valores numéricos SQL
- Nombres del jugador → alias definidos en la ejecución del programa.

d. Construcción incremental del SQL de carga

El loader no carga directamente en la base de datos. Genera un archivo estándar: *Code/data_loader.sql*

Este archivo contiene:

1. INSERT INTO User
2. INSERT INTO Player
3. INSERT INTO Map
4. INSERT INTO Game
5. INSERT INTO Sector (placeholder único para el mapa actual)
6. INSERT múltiples en TelemetryEvent
7. INSERT de UXInstrument y UXResponse (estructura vacía por defecto)

La estructura generada respeta las claves foráneas definidas en *db_struct.sql*.

3.3 Load (Carga en tablas core de PostgreSQL)

Una vez generado *data_loader.sql*, la carga se ejecuta manualmente:

```
psql -U usuario -d colnexus -f Code/db_struct.sql

psql -U usuario -d colnexus -f Code/ux_instrument_pens.sql

psql -U usuario -d colnexus -f Code/data_loader.sql

psql -U usuario -d colnexus -f Code/index_creation.sql
```

En orden:

1. *db_struct.sql* crea el esquema relacional (User, Player, Game, Map, Sector, TelemetryEvent, UX…).
2. *ux_instrument_pens.sql* inserta los 21 ítems del cuestionario PENS.
3. *data_loader.sql* ingesta la sesión de telemetría procesada.
4. *index_creation.sql* crea los índices recomendados para análisis:
    - índices compuestos *(game_id, player_id, tic)*
    - índices por sector/mapa
    - índice espacial GiST basado en *(pos_x, pos_y)*

Este orden garantiza integridad referencial y rendimiento óptimo.

3.4 Ejemplo de Telemetría Cruda (TSV Original del Juego)

(Basado estrictamente en el formato reconocido por loader_script.cxx — no se inventan campos que el script no procesa.)

```
When: 2025-11-04 19:49:25 Episode: 1 Map: 5
timestamp        tic     x       y       z       angle   momx    momy
2025-11-04 19:49:25      171     -224    -624    0       85.78   -1      -2      0
2025-11-04 19:49:27      206     -192    -594    0       187.03  -1      0       0
2025-11-04 19:49:28      241     -281    -627    0       2.81    2       2       0
2025-11-04 19:49:29      276     -26     -590    0       104.06  1       1       0
2025-11-04 19:49:30      311     -103    -576    0       97.03   -12     -2      0
2025-11-04 19:49:31      346     -376    -230    0       91.41   -3      15      0
2025-11-04 19:49:32      381     -212    -270    -9      82.97   12      -12     0
When: 2025-11-04 19:49:33 Episode: 1 Map: 1
timestamp        tic     x       y       z       angle   momx    momy
2025-11-04 19:49:35      417     1055    -3538   0       91.76   -1      6       0
2025-11-04 19:49:36      452     963     -3282   -16     167.34  -6      6       0
2025-11-04 19:49:37      487     697     -3226   0       179.65  -9      -1      0
2025-11-04 19:49:38      522     399     -3225   -8      184.92  -9      -1      0
2025-11-04 19:49:39      557     107     -3234   104     186.68  -9      -1      0
2025-11-04 19:49:40      592     -184    -3240   128     177.89  -9      0       0
2025-11-04 19:49:41      648     -288    -3247   104     348.40  4       -2      0
2025-11-04 19:49:42      683     -42     -3265   104     25.31   7       2       0
2025-11-04 19:49:43      718     188     -3108   50      51.68   5       5       0
2025-11-04 19:49:44      753     171     -3151   -8      32.34   -3      -4      0
2025-11-04 19:49:45      788     239     -3100   -8      111.45  -1      2       0
2025-11-04 19:49:46      823     103     -3066   -8      234.49  -4      -5      0
2025-11-04 19:49:47      858     144     -3153   -8      357.54  2       -2      0
2025-11-04 19:49:48      893     239     -3088   -8      76.64   0       4       0
2025-11-04 19:49:49      928     205     -3025   -8      87.19   -1      0       0
```

Características del ejemplo:

- Coincide exactamente con las expresiones regulares del loader.
- Contiene timestamp → tic → pos_x → pos_y.
- No incluye valores que el loader no reconoce (z, angle, health, ammo, etc.).
- Representa un tramo realista de 9 tics consecutivos.

4. Ejecuciones

4.1 Queries y Resultados

1. Duración Promedio de las Sesiones de Juego por Mapa.

Código SQL	SELECT m.map_id, m.nombre_oficial, ROUND(AVG(g.duracion_seconds)::numeric, 2) AS duracion_promedio_segundos FROM Game g JOIN Map m ON g.map_id = m.map_id GROUP BY m.map_id, m.nombre_oficial ORDER BY duracion_promedio_segundos DESC;
Resultado Impreso	

2. Jugadores con la Proximidad Promedio más Alta (más cercanos).

Código SQL	WITH paired_events AS (SELECT g1.player_id AS player_a, g2.player_id AS player_b, g1.map_id, SQRT(POWER(te1.pos_x - te2.pos_x, 2) + POWER(te1.pos_y - te2.pos_y, 2)) AS distancia FROM TelemetryEvent te1 JOIN Game g1 ON te1.game_id = g1.game_id JOIN TelemetryEvent te2 ON te1.game_id <> te2.game_id JOIN Game g2 ON te2.game_id = g2.game_id WHERE g1.map_id = g2.map_id AND g1.player_id < g2.player_id) SELECT

	player_a, player_b, ROUND(AVG(distancia)::numeric, 2) AS promedio_proximidad FROM paired_events WHERE distancia <= 10 GROUP BY player_a, player_b ORDER BY promedio_proximidad ASC;
Resultado Impreso	

3. Distancias de Trayectoria Mínima y Máxima por Jugador.

Código SQL	WITH distancia_tramos AS (SELECT g.player_id, te.game_id, SQRT(POWER(LEAD(te.pos_x) OVER (PARTITION BY te.game_id ORDER BY te.marca_tiempo) - te.pos_x, 2) + POWER(LEAD(te.pos_y) OVER (PARTITION BY te.game_id ORDER BY te.marca_tiempo) - te.pos_y, 2)) AS distancia_parcial FROM TelemetryEvent te JOIN Game g ON te.game_id = g.game_id), trayectorias AS (SELECT player_id, game_id, SUM(distancia_parcial) AS distancia_total FROM distancia_tramos GROUP BY player_id, game_id) SELECT player_id, ROUND(distancia_total::numeric, 2) AS distancia_total FROM trayectorias ORDER BY distancia_total DESC;
Resultado Impreso	

4. Respuestas UX para Jugadores con Duración de Trayectoria Superior al Promedio.

Código SQL	WITH duraciones AS (SELECT g.player_id, g.game_id, EXTRACT(EPOCH FROM (MAX(te.marca_tiempo) - MIN(te.marca_tiempo))) AS duracion_segundos

	FROM TelemetryEvent te JOIN Game g ON te.game_id = g.game_id GROUP BY g.player_id, g.game_id), promedio AS (SELECT AVG(duracion_segundos) AS duracion_promedio FROM duraciones) SELECT d.player_id, ROUND(d.duracion_segundos::numeric, 2) AS duracion_total, ui.dimension, AVG(uri.valor_likert) AS promedio_ux FROM duraciones d JOIN promedio p ON d.duracion_segundos > p.duracion_promedio JOIN UXResponse ux ON ux.user_id = d.player_id JOIN UXResponseItem uri ON uri.response_id = ux.response_id JOIN UXItem ui ON uri.item_id = ui.item_id GROUP BY d.player_id, duracion_total, ui.dimension ORDER BY duracion_total DESC;
Resultado Impreso	

5. Sector Más Visitado (*Hotspot*) por Episodio y Mapa.

Código SQL	WITH sectorized AS (SELECT g.episodio, g.map_id, FLOOR(te.pos_x / 250) AS sector_x, FLOOR(te.pos_y / 250) AS sector_y FROM TelemetryEvent te JOIN Game g ON te.game_id = g.game_id) SELECT episodio, map_id, sector_x, sector_y, COUNT(*) AS visitas FROM sectorized GROUP BY episodio, map_id, sector_x, sector_y ORDER BY visitas DESC LIMIT 10;
Resultado Impreso	

6. Número de Tics Donde los Jugadores Coincidieron en un Sector.

Código SQL	WITH sectorized AS (

	SELECT g.player_id, te.game_id, FLOOR(te.pos_x / 250) AS sector_x, FLOOR(te.pos_y / 250) AS sector_y FROM TelemetryEvent te JOIN Game g ON te.game_id = g.game_id) SELECT a.player_id AS jugador_a, b.player_id AS jugador_b, a.sector_x, a.sector_y, COUNT(*) AS coincidencias_sector FROM sectorized a JOIN sectorized b ON a.sector_x = b.sector_x AND a.sector_y = b.sector_y AND a.player_id < b.player_id GROUP BY jugador_a, jugador_b, a.sector_x, a.sector_y ORDER BY coincidencias_sector DESC;
Resultado Impreso	

7. Puntuación UX Promedio para Jugadores con la Trayectoria Más Corta por Episodio.

Código SQL	WITH eventos_ordenados AS (SELECT g.episodio, g.player_id, te.pos_x, te.pos_y, LEAD(te.pos_x) OVER (PARTITION BY g.game_id ORDER BY te.marca_tiempo) AS next_x, LEAD(te.pos_y) OVER (PARTITION BY g.game_id ORDER BY te.marca_tiempo) AS next_y FROM telemetryevent te JOIN game g ON te.game_id = g.game_id), distancias AS (SELECT episodio, player_id, SUM(SQRT(POWER(next_x - pos_x, 2) + POWER(next_y - pos_y, 2))) AS total_distancia FROM eventos_ordenados WHERE next_x IS NOT NULL GROUP BY episodio, player_id

	), jugadores_validos AS (SELECT DISTINCT r.user_id FROM uxresponse r JOIN uxresponseitem ri ON ri.response_id = r.response_id), minimos AS (SELECT DISTINCT ON (d.episodio) d.episodio, d.player_id, d.total_distancia FROM distancias d JOIN jugadores_validos j ON j.user_id = d.player_id ORDER BY d.episodio, d.total_distancia ASC) SELECT m.episodio, m.player_id, ROUND(AVG(ri.valor_likert),2) AS promedio_ux FROM minimos m JOIN uxresponse r ON r.user_id = m.player_id JOIN uxresponseitem ri ON ri.response_id = r.response_id GROUP BY m.episodio, m.player_id ORDER BY m.episodio;
Resultado Impreso	

8. Distancia Total Recorrida y Velocidad Promedio por Jugador, Analizando Todas las Partidas.

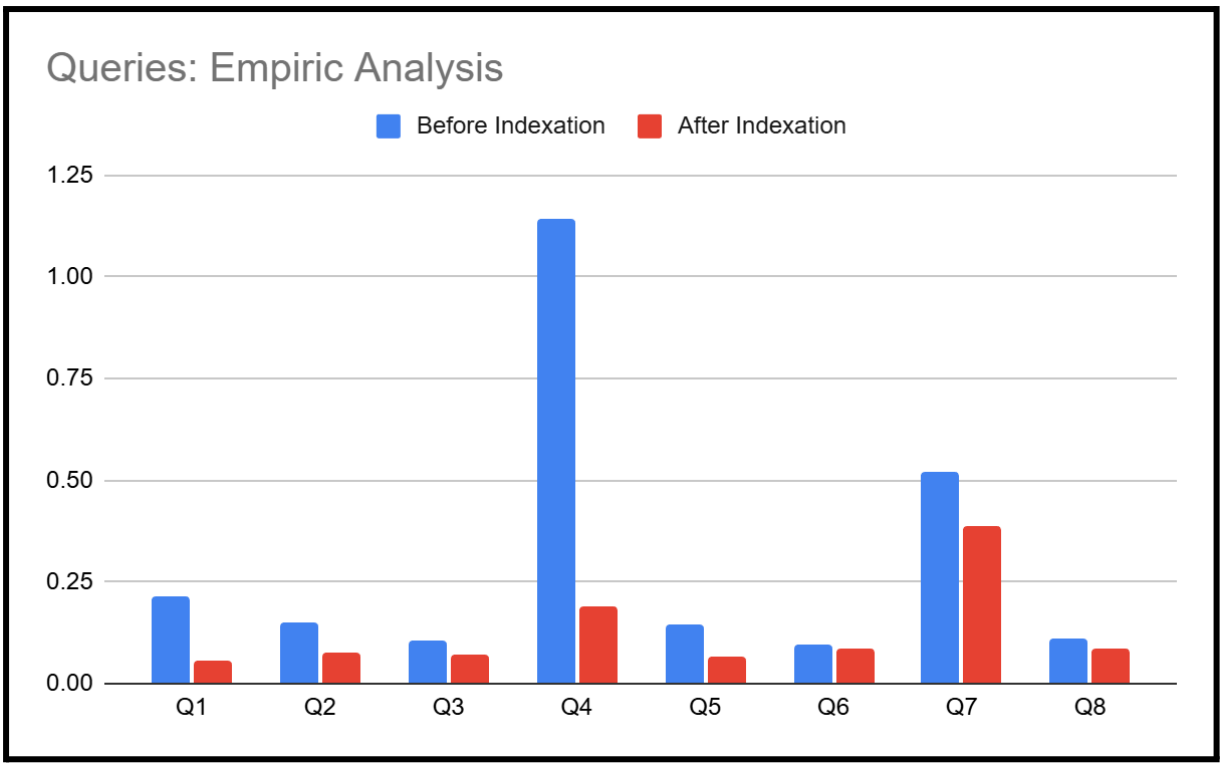
Código SQL	WITH puntos AS (SELECT g.game_id, g.player_id, g.episodio, te.marca_tiempo, te.pos_x, te.pos_y, LEAD(te.pos_x) OVER (PARTITION BY g.game_id ORDER BY te.marca_tiempo) AS next_x, LEAD(te.pos_y) OVER (PARTITION BY g.game_id ORDER BY te.marca_tiempo) AS next_y FROM TelemetryEvent te JOIN Game g ON te.game_id = g.game_id), distancias AS (SELECT episodio, player_id, marca_tiempo, SQRT(POWER(next_x - pos_x, 2) + POWER(next_y - pos_y,

	2)) AS distancia FROM puntos WHERE next_x IS NOT NULL), resumen AS (SELECT episodio, player_id, SUM(distancia) AS distancia_total, MIN(marca_tiempo) AS inicio, MAX(marca_tiempo) AS fin FROM distancias GROUP BY episodio, player_id) SELECT episodio, player_id, ROUND(distancia_total::numeric, 2) AS distancia_total, EXTRACT(EPOCH FROM (fin - inicio)) AS tiempo_total_segundos, ROUND((distancia_total / NULLIF(EXTRACT(EPOCH FROM (fin - inicio)), 0))::numeric, 2) AS velocidad_promedio FROM resumen ORDER BY episodio, velocidad_promedio DESC;
Resultado Impreso	

4.2 Index Analysis

Query Comparison: Q2 - Q5

	Before Indexation	**After Indexation**
Q1	0.217	0.059
Q2	0.151	0.077
Q3	0.106	0.070
Q4	1.144	0.189
Q5	0.145	0.065
Q6	0.094	0.088
Q7	0.519	0.389
Q8	0.109	0.088



Un análisis de los resultados demuestra conclusivamente la efectividad de la indexación a fin de incrementar la eficiencia y velocidad de la base de datos. La tendencia de decrementó en tiempos de ejecución se mantuvo a través de los ocho queries y se manifestó en una reducción de 58.75 puntos porcentuales. La diferencia fue más pronunciada en el caso del query número 4, con una reducción del 83.47%. Esto se debe a que dicho query trabaja con mayor cantidad de entidades indexadas, haciendo más evidente su impacto.

5. Nota Ética

Este proyecto analiza telemetría de juego proveniente del motor modificado Chocolate-Doom con el objetivo de diseñar e implementar una base de datos relacional que permita la ingestión, validación y análisis de datos de juego por tick, junto con respuestas del cuestionario de experiencia del jugador (PENS: *Player Experience of Need Satisfaction*). De acuerdo con la descripción del repositorio, los datos incluyen archivos de telemetría de sesiones de juego (TSV) y resultados del instrumento PENS suministrados por participantes voluntarios. A continuación se describen las consideraciones éticas aplicables:

1. Naturaleza de los datos: El proyecto utiliza telemetría de juego de DOOM y respuestas del cuestionario PENS. Los datos consisten únicamente en métricas de juego (movimiento, acciones, rendimiento) y valoraciones de experiencia del jugador.

2. Privacidad y anonimato: No se almacena información personal identificable. Cualquier nombre presente en los archivos originales se reemplaza por códigos anónimos. La base de datos no contiene correos, IPs ni datos sensibles.

3. Minimización y propósito: Solo se recopila la información necesaria para el análisis académico del comportamiento de juego. Los datos se utilizan exclusivamente con fines educativos y no para perfilamiento o usos externos.

4. Riesgos y mitigación: El riesgo ético es bajo debido a la naturaleza no sensible de los datos y al uso de pseudonimización. Los resultados del análisis se presentan de forma agregada, sin exponer sesiones individuales de manera identificable.

5. Responsabilidad: El equipo se compromete a manejar todos los datos de forma responsable, garantizando privacidad, transparencia y respeto por los participantes.

6. Diccionario de Datos

Tabla	Atributo	Tipo de dato	PK / FK	Restricciones / Comentarios
User	user_id	SERIAL	PK	Identificador único del usuario.
	nombre_completo	TEXT		No nulo.
	genero	TEXT		Control semántico (masculino, femenino, otro).
	carrera	TEXT		Información académica.
Player	player_id	SERIAL	PK	Identificador del alias del jugador.
	user_id	INTEGER	FK → User(user_id)	Relaciona el alias con el usuario dueño.
	alias	TEXT		Alias único dentro del estudio.
Map	map_id	SERIAL	PK	Identificador del mapa.
	codigo_map	VARCHAR(50)		Código interno del motor del juego.

	nombre_oficial	VARCHAR(200)		Nombre descriptivo.
Sector	sector_id	SERIAL	PK	Identificador del sector.
	map_id	INTEGER	FK → Map(map_id)	Un sector pertenece a un mapa.
	nombre_sector	VARCHAR(150)		Nombre o referencia espacial.
Game	game_id	SERIAL	PK	Identificador de la sesión de juego.
	player_id	INTEGER	FK → Player(player_id)	Jugador que participó.
	map_id	INTEGER	FK → Map(map_id)	Mapa donde se jugó.
	fecha_inicio	TIMESTAMP		No nulo.
	fecha_fin	TIMESTAMP		No nulo.
	duracion_seconds	INTEGER		Validación de duración.
	episodio	INTEGER		Episodio jugado.
TelemetryEvent	event_id	SERIAL	PK	Identificador del evento.
	game_id	INTEGER	FK → Game(game_id)	Evento asociado a una sesión.
	marca_tiempo	TIMESTAMP		Orden temporal.
	pos_x	NUMERIC		Coordenada X.
	pos_y	NUMERIC		Coordenada Y.
UXInstrument	instrument_id	SERIAL	PK	Identificador del instrumento UX.
	nombre	VARCHAR(100)		Nombre del cuestionario.
	tipo	VARCHAR(50)		Tipo o categoría del instrumento.

UXResponse	response_id	SERIAL	PK	Identificador de la respuesta.
	user_id	INTEGER	FK → User(user_id)	Usuario que respondió.
	instrument_id	INTEGER	FK → UXInstrument(instrument_id)	Instrumento aplicado.
	fecha_respuesta	TIMESTAMP		Registro temporal.
UXItem	item_id	SERIAL	PK	Identificador del ítem.
	instrument_id	INTEGER	FK → UXInstrument(instrument_id)	Pertenece a un instrumento.
	texto_pregunta	TEXT		Texto de la pregunta.
	dimension	VARCHAR(100)		Dimensión psicológica.
UXResponseItem	response_id	INTEGER	FK → UXResponse(response_id)	Relaciona respuesta con ítem.
	item_id	INTEGER	FK → UXItem(item_id)	Ítem respondido.
	valor_likert	INTEGER		CHECK (valor_likert BETWEEN 1 AND 7).

Conclusión

ColNexus logró integrar datos de telemetría de DOOM y respuestas del cuestionario PENS dentro de una base de datos relacional diseñada para garantizar consistencia, integridad y capacidad analítica. A través de un proceso ETL propio, los datos crudos fueron limpiados, normalizados y transformados en un conjunto estructurado apto para consultas tanto descriptivas como exploratorias.

El sistema resultante permite vincular métricas objetivas de comportamiento en el juego con evaluaciones subjetivas de la experiencia del jugador, habilitando análisis más completos sobre patrones de interacción y rendimiento. El proyecto demuestra la eficacia de combinar técnicas de ingeniería de datos y modelado relacional en un entorno académico, y deja una base sólida para futuras ampliaciones orientadas al análisis avanzado o la visualización de resultados.