Entrega Final Proyecto DBS

Fecha de entrega: 27/11/2025

Samuel Borras Mahecha

Luis Esteban Chaustre Garzón

Esteban Correa Guaqueta

Daniel Prieto Ordoñez

ANDRES OSWALDO CALDERON ROMERO, Ph.D.

PONTIFICIA UNIVERSIDAD JAVERIANA

FACULTAD DE INGENIERÍA

BASES DE DATOS

BOGOTA D.C

2025

Contenido

1. Introducción

El presente reporte corresponde a la Primera Entrega (Parte A: Conceptual y Diseño Lógico) del proyecto "Designing a Telemetry and UX Database for Chocolate-Doom Research". Este proyecto tiene como propósito el diseño de una base de datos relacional que permita almacenar, organizar y analizar datos de telemetría y experiencia de usuario (UX) obtenidos de sesiones de juego del motor modificado Chocolate-Doom.

El objetivo principal es proporcionar una estructura de datos sólida que soporte la integración de información proveniente de múltiples partidas, incluyendo posiciones de jugadores, estadísticas de combate, niveles, mapas y respuestas a instrumentos de experiencia de usuario como PENS, GUESS o BANGS. Esto permitirá al grupo de investigación realizar análisis sobre patrones de movimiento, cooperación y comportamiento de los jugadores, manteniendo al mismo tiempo la calidad, consistencia y ética en el manejo de los datos recolectados. En esta primera fase se abordan tres componentes fundamentales del diseño de la base de datos:

- Definición de supuestos y requisitos, tanto funcionales como no funcionales, junto con las consideraciones éticas relacionadas con la recopilación y manejo de datos personales.
- Elaboración del diagrama entidad-relación (E-R) con sus respectivas cardinalidades, atributos clave y vínculos entre las entidades principales (usuarios, jugadores, partidas, eventos de telemetría, instrumentos y respuestas UX).
- Derivación del esquema relacional, especificando llaves primarias (PK), foráneas (FK) y restricciones, además de justificar el proceso de normalización aplicado para asegurar la integridad y eficiencia de la base de datos.

Esta etapa constituye la base conceptual del sistema, sobre la cual se construirá la implementación y el proceso de ingestión de datos en fases posteriores. Su correcta elaboración garantiza que el diseño cumpla los objetivos de investigación, facilite consultas analíticas avanzadas y respete las normas de ética y privacidad establecidas.

2. Suposiciones y Requerimientos

A continuación, se presentan las suposiciones, requerimientos funcionales y no funcionales que orientan el diseño de la base de datos propuesta para el proyecto. Estas consideraciones establecen el marco técnico y operativo bajo el cual se desarrollará el sistema, definiendo las condiciones iniciales asumidas para su funcionamiento, las capacidades que debe ofrecer para cumplir con los objetivos de investigación, y los criterios de desempeño, integridad y privacidad que deben mantenerse durante su ejecución. En conjunto, estos elementos garantizan que la solución planteada sea coherente, escalable y éticamente responsable en el manejo de los datos de telemetría y experiencia de usuario.

Suposiciones:

- Se asume que los archivos de telemetría se reciben en formato estándar con estructura fija.
- Se asume que cada registro contiene un identificador de partida, jugador y tiempo.
- Se asume que la frecuencia de captura es constante durante toda la sesión.
- Se asume que cada jugador pertenece a un único usuario.
- Se asume que los datos de los cuestionarios UX están completos y validados antes de su carga.
- Se asume que todos los identificadores de las entidades son únicos.
- Se asume que los datos personales se almacenan de forma separada a las respuestas de los cuestionarios UX para preservar el anonimato.
- Se asume que el sistema operará en PostgreSQL como gestor principal.
- Se asume que las consultas analíticas se ejecutan con volúmenes moderados de datos.
- Se asume que el usuario del sistema cuenta con permisos adecuados para ejecutar el script de carga.

Requerimientos funcionales:

- El sistema debe registrar la información de usuarios.
- El sistema debe cargar archivos a las tablas del modelo.
- El sistema debe validar los datos cargados.
- El sistema debe eliminar registros duplicados.
- El sistema debe registrar errores en una tabla de control.
- El sistema debe ejecutar consultas analíticas sobre trayectorias, proximidad, cooperación o desempeño.
- El sistema debe relacionar respuestas UX con datos de telemetría.

Requerimientos no funcionales:

- El sistema debe mantener integridad referencial en todas las tablas.
- El sistema debe garantizar privacidad de los datos personales.
- El sistema debe soportar veinte mil registros de telemetría sin pérdida de rendimiento.
- El sistema debe incluir índices para optimizar las consultas.
- El sistema debe permitir la reconstrucción completa mediante un script de carga.

3. Diagrama Entidad Relación

El siguiente diagrama entidad–relación (E-R) representa, de manera inicial y sujeta a modificaciones, la estructura conceptual y lógica del sistema diseñado para el proyecto “Designing a Telemetry and UX Database for Chocolate-Doom Research”. Este modelo fue desarrollado utilizando la herramienta Visual Paradigm, lo que permitió una representación clara, estructurada y normalizada de las entidades y sus relaciones.

En el diagrama se pueden observar las cardinalidades que definen el tipo de relación entre las entidades (uno a uno, uno a muchos o muchos a muchos), así como los atributos que deben ser únicos para garantizar la integridad de los datos y aquellos que admiten valores nulos cuando la información es opcional o depende del contexto de captura. Asimismo, se destacan las llaves primarias que identifican de manera exclusiva cada registro dentro de una tabla y las llaves foráneas que establecen los vínculos entre las diferentes entidades, asegurando la integridad referencial del modelo. En conjunto, este diagrama constituye la base estructural sobre la cual se construirá el esquema relacional y los procesos de carga, validación y análisis de los datos.

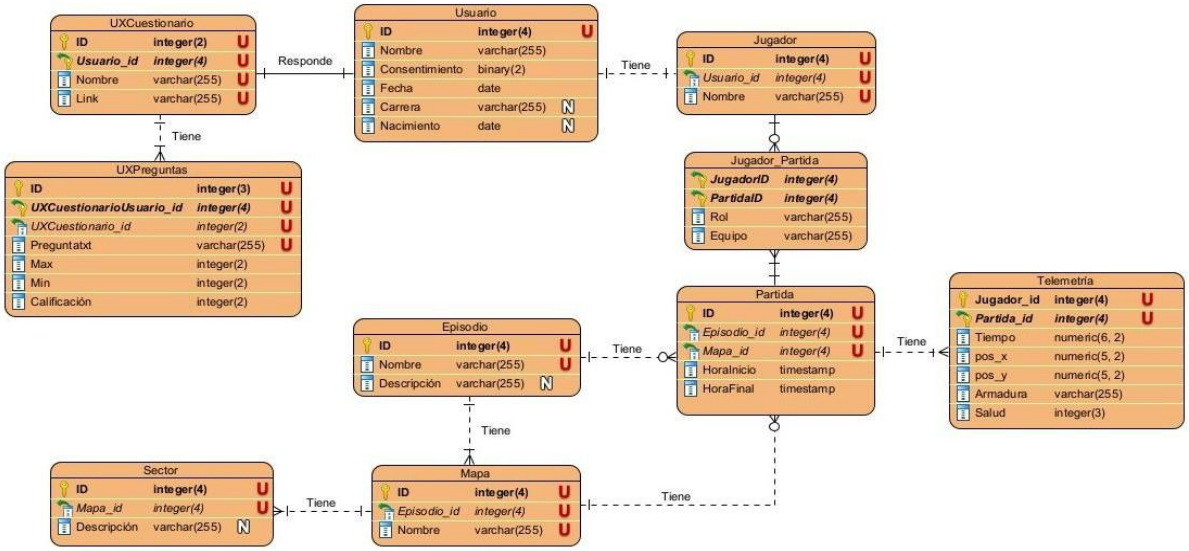

***Fig 1**. Primer Diagrama E-R.*

4. Diccionario de Datos, FKs, PKs

El diccionario de datos presentado a continuación complementa el modelo relacional propuesto para el proyecto. En este documento se especifican, para cada tabla del sistema, los atributos que la componen junto con su respectiva variable, nombre de la variable, tipo de dato, valores permitidos, condición de nulabilidad, y la identificación de si actúan como llave primaria (PK) o llave foránea (FK). Además, se incluye una descripción detallada del propósito y uso de cada campo dentro del modelo. Este nivel de detalle permite comprender de manera precisa la estructura lógica de la base de datos, facilita la implementación en el gestor PostgreSQL y asegura la correcta validación, consistencia e integridad de los datos durante los procesos de carga, análisis y consulta.

Usuario						
Variable	Nombre Variable	Tipo de Dato	Valores Permitidos	Nulo	PK/FK	Descripción
ID Usuario	ID	Integer	0001-9999	No	PK	Valor numérico asignado a un usuario
Nombre Usuario	Nombre	Varchar	Texto Libre	No		Nombre del usuario
Fecha Nacimiento	Nacimiento	mm/dd/yyyy	01-12/01-31/1990 - now()	Si		Edad del Usuario
Carrera	Carrera	Varchar	Lista de Carreras	Si		Carrera que estudia el usuario
Consentimiento	Consentimiento	Booleano	Si/No	No		Consentimiento del usuario para tratar sus datos
Fecha Creación	Fecha	mm/dd/yy	07/10/25 - now()	No		Fecha de creacion del usuario en la base de datos
Jugador						
Variable	Nombre Variable	Tipo de Dato	Valores Permitidos	Nulo	PK/FK	Descripción
ID Jugador	ID	Integer	0001-9999	No	PK	Valor numérico asignado a un Jugador
Nombre Jugador	Nombre	Varchar	Texto Libre	No		Nombre creado para el jugador
ID del Usuario	Usuario_id	Integer	0001-9999	No	FK	Id del usuario al que pertenece el jugador
Partida						
Variable	Nombre Variable	Tipo de Dato	Valores Permitidos	Nulo	PK/FK	Descripción
ID Partida	ID	Integer	0001-9999	No	PK	Valor numérico asignado a una partida
Hora Inicio	HoraInicio	hh:mm	00-23:00-59	No		Hora de incio de una partida
Hora Final	HoraFinal	hh:mm	00-23:00-59	No		Hora en que termina la partida
ID del episodio	Episodio_id	Integer	0001-9999	No	FK	Id del episodio en el que se jugó
ID del mapa	Mapa_id	Integer	0001-9999	No	FK	Id del mapa en el que se jugó
Episodio						Jugador
Variable	Nombre Variable	Tipo de Dato	Valores Permitidos	Nulo	PK/FK	Descripción
ID Episodio	ID	Integer	0001-9999	No	PK	Valor numérico asignado a un episodio
Nombre Episodio	Nombre	Varchar	Lista de Episodios	No		Nombre del episodio
Descripción	Descripción	Varchar	Texto Libre	Si		Descripción del episodio
Mapa						
Variable	Nombre Variable	Tipo de Dato	Valores Permitidos	Nulo	PK/FK	Descripción
ID Mapa	ID	Integer	0001-9999	No	PK	Valor numérico asignado a un mapa
Nombre Mapa	Nombre	Varchar	Lista de Mapas	No		Nombre del mapa
ID del episodio	Episodio_id	Integer	0001-9999	No	FK	Id del episodio al que pertenece el mapa
Sector						
Variable	Nombre Variable	Tipo de Dato	Valores Permitidos	Nulo	PK/FK	Descripción
ID Sector	ID	Integer	0001-9999	No	PK	Valor numérico adignado a un sector
Descripción	Descripción	Varchar	Texto Libre	Si		Descripción del sector
ID del mapa	Mapa_id	Integer	0001-9999	No	FK	Id del mapa al que pertenece el sector
UXCuestionario						
Variable	Nombre Variable	Tipo de Dato	Valores Permitidos	Nulo	PK/FK	Descripción
ID Cuestionario	ID	Integer	001-003	No	PK	Valor numérico asignado a los cuestionarios UX
ID del Usuario	Usuario_id	Integer	0001-9999	No	PK/FK	Id del usuario que contesta el cuestionario
Nombre Cuestionario	Nombre	Varchar	Lista de Cuestionarios	No		Nombre del cuestionario
Link Cuestionario	Link	Varchar	Direcciones URL	No		Link en el que se encuentra el cuestionario
UXPreguntas						
Variable	Nombre Variable	Tipo de Dato	Valores Permitidos	Nulo	PK/FK	Descripción
ID Pregunta	ID	Integer	001-999	No	PK	Valor numérico asignado a una pregunta del cuestionario
ID Cuestionario	UXCuestionario_id	Integer	001-003	No	FK	Id del cuestionario donde esta la pregunta
ID del Usuario	Usuario_id	Integer	0001-9999	No	PK/FK	Id del usuario que responde la pregunta
Pregunta	Preguntatxt	Varchar	Texto Libre	No		Pregunta que se realiza al usuario
Escala Máxima	Max	Integer	5,10	No		Valor máximo con el que se puede calificar la pregunta
Escala Mínimo	Min	Integer	0	No		Valor mínimo con el que se puede calificar la pregunta
Calificación	Calificación	Integer	[Min,Max]	No		Calificación con la que el usuario respondio la pregunta
Telemetría						
Variable	Nombre Variable	Tipo de Dato	Valores Permitidos	Nulo	PK/FK	Descripción
ID del Jugador	Jugador_id	Integer	0001-9999	No	PK/FK	Id del jugador que esta jugando una partida
ID de la Partida	Partida_id	Integer	0001-9999	No	PK/FK	Id de la partida que se juega
Tiempo	Tiempo	Numeric	0001-9999	No		Cuanto tiempo ha transcurrido en la partida
Posición en eje x	pos_x	Numeric	001-999	No		Posición en eje x que tiene el jugador
Posición en eje y	pos_y	Numeric	001-999	No		Posición en eje y que tiene el jugador
Armadura	Armadura	Varchar	Lista de Armaduras	No		Con que tipo de armadura esta jugando el jugador
Salud	Salud	Integer	000-200	No		Valor de la salud del jugador
Jugador_Partida						
Variable	Nombre Variable	Tipo de Dato	Valores Permitidos	Nulo	PK/FK	Descripción
ID del Jugador	JugadorID	Integer	0001-9999	No	PK/FK	Id del jugador que esta jugando una partida
ID de la Partida	PartidaID	Integer	0001-9999	No	PK/FK	Id de la partida que se juega
Rol	Rol	Varchar	Lista de Roles	No		Cual es el rol del jugador en la partida
Equipo	Equipo	Varchar	Texto Libre	No		A qué equipo pertenece el jugador

Tabla 1. *Diccionario de datos.*

5. Justificación de la Normalización hasta 3NF

El esquema propuesto de manera inicial para la base de datos cumplía satisfactoriamente con los criterios de las tres primeras formas normales. Sin embargo, se realizaron cambios en algunas relaciones y atributos con el fin de reducir la complejidad del sistema, el diagrama Entidad-Relación que nace de estas modificaciones se presenta a continuación:

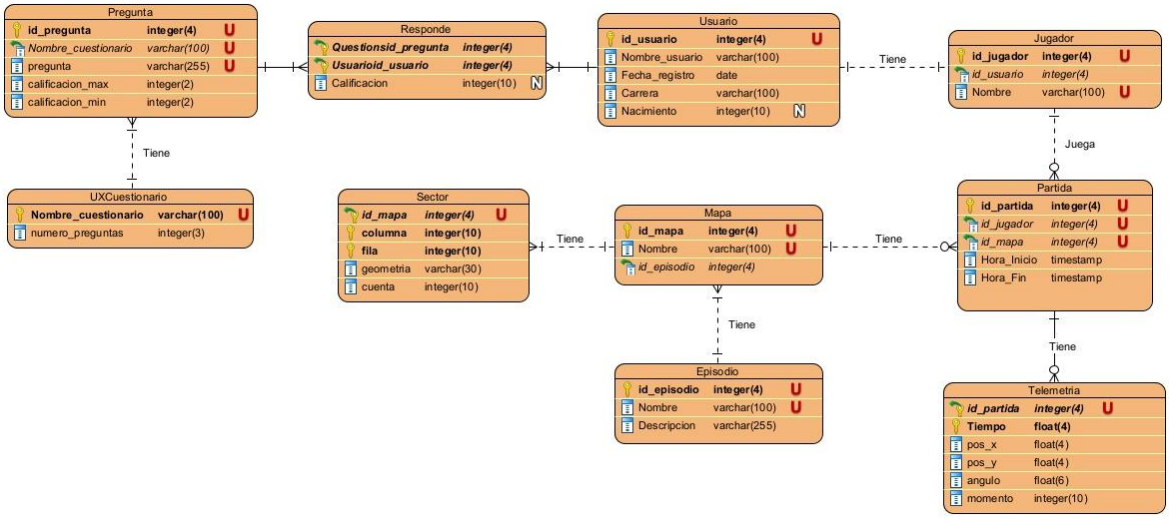

***Fig 2**. Segundo Diagrama E-R.*

Para este planteamiento de diseño también se cumplen las primeras tres formas normales. En Primera Forma Normal (1NF), todas las tablas cuentan con valores atómicos, no se identifican grupos repetitivos y se establece una clave primaria claramente definida en cada entidad. La tabla Sector, por ejemplo, utiliza una clave compuesta (id_mapa, columna, fila) que garantiza la unicidad de cada sector dentro de un mapa, evitando redundancias.

Para la Segunda Forma Normal (2NF), todas las tablas que emplean claves compuestas tienen dependencias funcionales completas. En el caso ya mencionado de Sector, ningún atributo depende parcialmente de la clave compuesta, sino que todos se derivan de la combinación completa de los campos que la conforman. El resto de entidades usan claves simples, por lo que al estar ya en 1NF pasan automáticamente a 2NF.

En relación con la Tercera Forma Normal (3NF), no se observan dependencias transitivas que comprometan la estructura. Por ejemplo, en Jugador el nombre del jugador depende directamente del identificador del jugador y no del usuario, aun cuando exista una relación uno a uno entre ambas entidades. Asimismo, la entidad Pregunta vincula correctamente el cuestionario mediante una clave foránea sin introducir atributos que dependan indirectamente de otra entidad.

6. Implementación de DDL en PostgreSQL

El proceso de construcción de la base de datos *doomProject* partió del diseño conceptual representado en el diagrama entidad-relación, el cual permitió identificar de manera clara las entidades principales, sus atributos y las relaciones existentes entre ellas. Dicho modelo se utilizó como base para elaborar un diccionario de datos, en el que se definieron de forma estructurada los nombres de las tablas, los tipos de datos, las claves primarias, las claves foráneas, las restricciones de unicidad y los dominios permitidos para cada atributo. Este diccionario constituyó la guía técnica para garantizar la correcta correspondencia entre el modelo lógico y su implementación física en el sistema gestor de bases de datos PostgreSQL.

En el diagrama, las entidades Usuario, Jugador, Episodio, Mapa, Sector, Partida, Telemetría, UXCuestionario, Pregunta y Responde se interconectan mediante relaciones uno a uno, uno a muchos y muchos a muchos, reflejando la estructura jerárquica y funcional del proyecto. Por ejemplo, un Usuario puede tener un Jugador asociado, un Jugador participa en múltiples Partidas, y cada Partida genera datos de Telemetría. Asimismo, la parte relacionada con la experiencia del usuario se organiza mediante Cuestionarios compuestos por Preguntas, que a su vez son respondidas por diferentes Usuarios, estableciendo una relación de tipo muchos a muchos materializada en la tabla Responde.

Por su parte, el diccionario de datos permitió estandarizar la nomenclatura de los atributos y definir sus tipos conforme a las necesidades del sistema: se emplearon tipos SERIAL para las claves primarias autoincrementales, VARCHAR para los campos textuales de longitud variable, y FLOAT para la información numérica de precisión variable en la telemetría. Además, se especificaron las reglas de integridad referencial que garantizan la coherencia entre las entidades dependientes, como las claves foráneas entre Mapa y Episodio, o entre Pregunta y UXCuestionario.

Con base en esa documentación, se elaboró el DDL que materializó la estructura lógica del modelo en el entorno físico de PostgreSQL. Cada tabla del DDL se corresponde directamente con una entidad del diagrama y respeta las especificaciones consignadas en el diccionario de datos. Las relaciones entre entidades fueron implementadas mediante sentencias FOREIGN KEY, asegurando la integridad y consistencia del conjunto de datos.

En conclusión, la implementación del DDL fue el resultado de un proceso sistemático de traducción del modelo conceptual a un modelo físico, en el que el diagrama entidad-relación definió la estructura general del sistema, el diccionario de datos precisó los detalles técnicos de cada componente, y el DDL consolidó ambos elementos en una base de datos funcional, eficiente y coherente con los principios del diseño relacional.

En conclusión, la implementación del DDL fue el resultado de un proceso sistemático de traducción del modelo conceptual a un modelo físico, en el que el diagrama entidad-relación definió la estructura general del sistema, el diccionario de datos precisó los detalles técnicos de cada componente, y el DDL consolidó ambos elementos en una base de datos funcional, eficiente y coherente con los principios del diseño relacional. Asimismo, se

adjunta en el entregable un respaldo completo de la base de datos, con el fin de garantizar la trazabilidad del desarrollo, la posibilidad de verificación de su correcta implementación y la facilidad para su futura restauración o despliegue en diferentes entornos de trabajo.

7. Población de las relaciones UXCuestionario y Pregunta

El proceso de población de las relaciones UXCuestionario y Pregunta se realizó tomando como referencia el instrumento GUESS, ampliamente utilizado para la evaluación de experiencias de usuario en entornos interactivos. En primera instancia, se registró en la tabla *UXCuestionario* una única entrada correspondiente al instrumento aplicado, identificada con el nombre "GUESS" y con el valor 18 en el campo *numero_preguntas*, representando el total de ítems que lo componen. Esta inserción permitió establecer el vínculo entre el cuestionario y las preguntas que conforman su estructura, sirviendo como punto de referencia para la posterior carga de datos en la relación *Pregunta*.

Posteriormente, se procedió a poblar la tabla *Pregunta* con las 18 preguntas del instrumento GUESS, como se observa en la Figura 3, asignando a cada una un identificador incremental generado automáticamente mediante la propiedad *SERIAL* del campo *id_pregunta*. Cada pregunta fue registrada asociada al cuestionario “GUESS”, con los valores 1 y 7 en los campos *calificacion_min* y *calificacion_max*, respectivamente. Estos valores representan la escala de respuesta tipo Likert utilizada en el instrumento, donde 1 corresponde a "Strongly Disagree" y 7 corresponde a "Strongly Agree", reflejando la intensidad del acuerdo del usuario con cada afirmación. De esta manera, la estructura de datos quedó preparada para almacenar las respuestas de los usuarios en la tabla *Responde*, asegurando coherencia con el modelo teórico del cuestionario y consistencia con los principios del diseño relacional previamente establecidos.

id_pregunta	nombre_cuestionario	pregunta	calificacion_max	calificacion_min
1	GUESS	I find the controls of the game to be straightforward.	7	1
2	GUESS	I find the game's interface to be easy to navigate.	7	1
3	GUESS	I am captivated by the game's story from the beginning.	7	1
4	GUESS	I enjoy the fantasy or story provided by the game.	7	1
5	GUESS	I feel detached from the outside world while playing the game.	7	1
6	GUESS	I do not care to check events that are happening in the real world during the game.	7	1
7	GUESS	I think the game is fun.	7	1
8	GUESS	I feel bored while playing the game.	7	1
9	GUESS	I feel the game allows me to be imaginative.	7	1
10	GUESS	I feel creative while playing the game.	7	1
11	GUESS	I enjoy the sound effects in the game.	7	1
12	GUESS	I feel the game's audio (e.g., sound effects, music) enhances my gaming experience.	7	1
13	GUESS	I am very focused on my own performance while playing the game.	7	1
14	GUESS	I want to do as well as possible during the game.	7	1
15	GUESS	I find the game supports social interaction (e.g., chat) between players.	7	1
16	GUESS	I like to play this game with other players.	7	1
17	GUESS	I enjoy the game's graphics.	7	1
18	GUESS	I think the game is visually appealing.	7	1

(18 rows)

***Fig 3**. Tuplas de la Tabla Pregunta.*

8. Inserción de Datos Restantes y Relación Telemetría

El proceso de población de las tablas Usuario, Jugador, Responde, Mapa y Episodio se llevó a cabo mediante la inserción manual de los registros, utilizando sentencias SQL del tipo INSERT INTO. Dado que estas relaciones no manejan grandes volúmenes de datos, su llenado fue realizado sin inconvenientes, permitiendo mantener un control preciso sobre la

integridad de los valores insertados. En la tabla Usuario se registraron los datos básicos de identificación de los jugadores, mientras que en Jugador se vinculó cada usuario con su respectiva identidad dentro del sistema de juego, a manera de ejemplo las relaciones se muestran en las Figuras 4 y 5, respectivamente. De igual forma, las tablas Mapa y Episodio fueron pobladas con los nombres y descripciones de los niveles disponibles en el entorno, manteniendo la coherencia entre las claves foráneas y las relaciones establecidas en el diagrama entidad-relación.

```
doomproject=# select * from usuario;
 id_usuario |  nombre_usuario  | fecha_registro |     carrera
------------+------------------+----------------+-----------------
          1 | EstebanChaustre  | 2025-11-06     | IngSistemas
          2 | SamuelBorras     | 2025-11-06     | IngSistemas
          3 | DanielPrieto     | 2025-11-06     | IngSistemas
          4 | JulioTriana      | 2025-11-07     | Negocios
          5 | JuanMateus       | 2025-11-07     | Administracion
          6 | AlejandraPerdomo | 2025-11-08     | DisenoIndustrial
(6 rows)
```

***Fig 4**. Tuplas de la Tabla Usuario.*

```
doomproject=# select * from jugador;
 id_jugador | id_usuario |  nombre
------------+------------+----------
        123 |          1 | EstebiPRO
        456 |          2 | Borrium
        789 |          3 | Danidih
        444 |          4 | Jeelian
        100 |          5 | Juanmatro
        126 |          6 | Alepero
(6 rows)
```

***Fig 5**. Tuplas de la Tabla Jugador.*

Posteriormente, cada usuario participó en una sesión de juego y, al finalizar, diligenció el cuestionario GUESS. Las calificaciones proporcionadas en este instrumento se registraron en la tabla Responde, respetando el formato definido en el modelo relacional. En cada registro se asoció el identificador del usuario con el de la pregunta correspondiente, junto con la calificación otorgada. Este proceso permitió consolidar la información de la experiencia del usuario de manera ordenada y normalizada, garantizando que las respuestas se vincularan correctamente al cuestionario y al usuario que las emitió.

Por otro lado, el proceso de inserción de datos para las tablas Partida y Telemetría fue considerablemente más complejo y automatizado. Cuando un jugador iniciaba una nueva partida, el archivo modificado proporcionado por el profesor comenzaba a captar los datos de telemetría de forma automática durante toda la sesión. El primer registro obtenido se consideró como la hora de inicio y el último como la hora de finalización de la partida, valores

que fueron posteriormente almacenados en la tabla Partida, como se ve reflejado en la Figura 6. De esta manera, se logró mantener la trazabilidad temporal de cada sesión de juego, vinculando al jugador con su actividad en un mapa determinado.

```
doomproject=# select * from partida;
 id_partida | id_jugador | id_mapa |     hora_inicio     |      hora_fin
------------+------------+---------+---------------------+---------------------
          1 |        456 |       5 | 2025-11-08 01:37:41 | 2025-11-08 03:05:11
          2 |        126 |       3 | 2025-11-08 14:32:13 | 2025-11-08 14:47:52
          3 |        100 |       3 | 2025-11-08 16:12:20 | 2025-11-08 16:24:11
          4 |        123 |       7 | 2025-11-08 16:52:42 | 2025-11-08 18:46:28
          5 |        444 |       5 | 2025-11-08 19:10:40 | 2025-11-08 20:25:40
          6 |        789 |       7 | 2025-11-08 21:03:55 | 2025-11-08 22:35:08
(6 rows)
```

***Fig 6**. Tuplas de la Tabla Partida.*

El llenado de la tabla Telemetría requirió un procedimiento más elaborado, debido al gran volumen de datos generados por el sistema. Para ello, se desarrolló un script en lenguaje C++, con apoyo de herramientas de inteligencia artificial, que automatizó la conversión del formato original a uno compatible con las sentencias SQL de inserción. El código leía los datos desde un archivo de entrada con formato:

timestamp tic x y z angle momx momy momz

Y los transformaba al formato deseado:

```
(id_partida, 'timestamp', tic, x, y, z, angle, momx, momy),
```

Eliminando la última columna y agregando comillas al valor de tipo timestamp.

El uso de esta herramienta resultó fundamental, ya que permitió convertir miles de registros de telemetría de forma rápida y precisa, generando automáticamente las tuplas en el formato adecuado para los INSERTS dentro de PostgreSQL. Sin esta automatización, el proceso habría sido sumamente tedioso y propenso a errores humanos, pues habría requerido modificar manualmente cada línea del archivo fuente, que alcanzo más de veinte mil tuplas, como se muestra en la consulta ejecutada en la Figura 7. En consecuencia, el script desarrollado facilitó de manera significativa la población masiva y estructurada de la tabla Telemetría, asegurando la coherencia y eficiencia del proceso de carga de datos.

```
doomproject=# select count (*) from telemetria;
 count
-------
 21706
(1 row)
```

***Fig 7**. Conteo de Tuplas de la Tabla Telemetría.*

Finalizado el proceso de inserción de todas las tuplas en las tablas correspondientes, se generó una copia de respaldo de la base de datos utilizando la documentación oficial de PostgreSQL y el comando:

```
pg_dump doomproject > doomproject.sql
```

Este procedimiento permitió obtener un volcado completo con la estructura y los datos del proyecto, garantizando su preservación y posibilidad de restauración futura. El archivo resultante, doomproject.sql, se encuentra adjunto dentro del archivo comprimido .zip entregado junto con el presente trabajo. Por su parte, otros elementos relevantes como el script programado en C++ y los datos de telemetría por partida se encuentran disponibles en el repositorio de GitHub, el cual fue compartido previamente con el profesor para su revisión.

9. Ejecución de Queries Analíticas

Con el fin de comprobar la funcionalidad de la base de datos realizada, y de obtener algunos datos relevantes en relación a la telemetría, duración de las partidas y satisfacción de los usuarios se ejecutaron las siguientes consultas:

9.1. Consulta 1

La consulta 1 tiene como propósito obtener la duración promedio de las partidas asociadas a cada mapa dentro del entorno del videojuego. Esta métrica es fundamental porque permite identificar patrones de comportamiento de los jugadores, así como evaluar si determinados mapas presentan una mayor complejidad, tiempos de resolución más largos o menor eficiencia por parte de los usuarios. El querie propuesto y los resultados de la consulta se muestran a continuación en las Figuras 8 y 9, respectivamente.

```
1.
SELECT
    p.id_mapa,
    m.Nombre AS nombre_mapa,
    AVG(p.Hora_fin - p.Hora_inicio) AS duracion_promedio
FROM Partida p
JOIN Mapa m ON m.id_mapa = p.id_mapa
GROUP BY
    p.id_mapa,
    m.Nombre
ORDER BY
    p.id_mapa;
```

***Fig 8**. Querie Ejecutado para la Consulta 1.*

```
 id_mapa |    nombre_mapa    | duracion_promedio
---------+-------------------+-------------------
       3 | Toxin Refinery    | 00:13:45
       5 | Phobos Lab        | 01:21:15
       7 | Computer Station  | 01:42:29.5
(3 rows)
```

***Fig 9**. Resultado Consulta 1.*

9.2. Consulta 3

La consulta 3 tiene como finalidad analizar el comportamiento espacial de los jugadores identificando, para cada uno de ellos, tanto la distancia mínima como la máxima recorrida en sus partidas. Para ello, se utilizan los datos de posiciones registradas en la tabla de telemetría, calculando la distancia total recorrida por el jugador durante cada partida a partir de los cambios sucesivos en las coordenadas espaciales (pos_x, pos_y, pos_z). El querie ejecutado y los resultados obtenidos se muestran en las Figuras 10 y 11, respectivamente.

```
3.
WITH trayectorias AS (
    SELECT
        p.id_jugador,
        t1.id_partida,
        SQRT(
            POWER(t2.pos_x - t1.pos_x, 2) +
            POWER(t2.pos_y - t1.pos_y, 2) +
            POWER(t2.pos_z - t1.pos_z, 2)
        ) AS dist
    FROM Telemetria t1
    JOIN Telemetria t2
        ON t1.id_partida = t2.id_partida
       AND t2.tiempo = (
            SELECT MIN(t3.tiempo)
            FROM Telemetria t3
            WHERE t3.id_partida = t1.id_partida
              AND t3.tiempo > t1.tiempo
       )
    JOIN Partida p
        ON p.id_partida = t1.id_partida
)
SELECT
    id_jugador,
    CAST(SUM(dist) AS DECIMAL(10, 2)) AS trayecto_total,
    CAST(MIN(dist) AS DECIMAL(10, 2)) AS tramo_minimo,
    CAST(MAX(dist) AS DECIMAL(10, 2)) AS tramo_maximo
FROM trayectorias
GROUP BY
    id_jugador;
```

***Fig 10**. Querie Ejecutado para la Consulta 3.*

id_jugador	trayecto_total	tramo_minimo	tramo_maximo
789	1029237.21	1.00	5410.03
456	987951.26	1.00	5410.03
444	845041.92	1.00	5410.03
123	1282425.48	1.00	5410.03
126	176733.16	1.00	5406.43
100	132694.53	1.41	5410.03

(6 rows)

***Fig 11**. Resultado Consulta 3.*

Los resultados obtenidos muestran que, aunque la distancia total recorrida por jugador varía considerablemente, los valores de los tramos mínimo y máximo son prácticamente idénticos entre todos los jugadores. Este comportamiento sugiere que los movimientos extremadamente cortos y los saltos máximos son determinados más por la estructura del mapa y las limitaciones del motor del juego que por las diferencias individuales de los jugadores**.** En particular, los tramos mínimos probablemente corresponden a movimientos prácticamente estacionarios entre tics consecutivos, mientras que los tramos máximos representan desplazamientos grandes que podrían ser resultado de teletransportes, saltos de o errores de captura de la telemetría.

9.3. Consulta 4

La consulta 4 busca listar las respuestas de encuestas de experiencia de usuario (UX) únicamente de los jugadores cuya duración de trayectoria en el juego es superior al promedio, lo que permite enfocar el análisis en los usuarios más comprometidos. Esto es importante porque estos jugadores suelen ser los más activos y leales, por lo que sus respuestas ofrecen información valiosa para mejorar la experiencia y la retención dentro del juego. La consulta une la tabla de respuestas UX con la tabla de trayectorias de los jugadores, y filtra aquellos cuya duración excede el promedio general, mostrando así únicamente las opiniones de los usuarios con trayectorias más largas, optimizando el análisis y facilitando decisiones estratégicas de diseño. La consulta ejecutada y parte del resultado se muestran en las Figuras 12 y 13, respectivamente.

```
4.
WITH promedio AS (
    SELECT
        AVG(EXTRACT(EPOCH FROM (hora_fin - hora_inicio))) AS duracion_promedio
    FROM Partida
)
SELECT
    u.id_usuario,
    u.nombre_usuario,
    r.id_pregunta,
    r.calificacion
FROM Partida p
JOIN Jugador j ON j.id_jugador = p.id_jugador
JOIN Usuario u ON u.id_usuario = j.id_usuario
JOIN Responde r ON r.id_usuario = u.id_usuario
WHERE EXTRACT(EPOCH FROM (p.hora_fin - p.hora_inicio)) >
      (SELECT duracion_promedio FROM promedio)
ORDER BY
    u.id_usuario,
    r.id_pregunta;
```

***Fig 12**. Querie Ejecutado para la Consulta 4.*

id_usuario	nombre_usuario	id_pregunta	calificacion
1	EstebanChaustre	1	6
1	EstebanChaustre	2	7
1	EstebanChaustre	3	5
1	EstebanChaustre	4	7
1	EstebanChaustre	5	6
1	EstebanChaustre	6	7
1	EstebanChaustre	7	4
1	EstebanChaustre	8	5
1	EstebanChaustre	9	7
1	EstebanChaustre	10	7
1	EstebanChaustre	11	7
1	EstebanChaustre	12	6
1	EstebanChaustre	13	6
1	EstebanChaustre	14	5
1	EstebanChaustre	15	4
1	EstebanChaustre	16	7
1	EstebanChaustre	17	7
1	EstebanChaustre	18	6
2	SamuelBorras	1	7
2	SamuelBorras	2	7
2	SamuelBorras	3	7
2	SamuelBorras	4	6
2	SamuelBorras	5	5
2	SamuelBorras	6	3
2	SamuelBorras	7	5
2	SamuelBorras	8	6
2	SamuelBorras	9	6
2	SamuelBorras	10	7
2	SamuelBorras	11	7
2	SamuelBorras	12	6
2	SamuelBorras	13	7
2	SamuelBorras	14	5

***Fig 13**. Resultado Consulta 4.*

9.4. Consulta 5

La consulta 5 busca identificar, para cada episodio y cada mapa, cuál es el sector o área que recibe la mayor cantidad de visitas por parte de los jugadores. Esto es importante porque permite a los diseñadores del juego entender los patrones de movimiento y concentración de los jugadores, detectar áreas muy populares o descuidadas, y optimizar el diseño del mapa para mejorar la experiencia de juego y la estrategia del usuario. La consulta normalmente agrupa los datos de visitas por sector, episodio y mapa, cuenta la cantidad de visitas por cada combinación y luego selecciona el sector con el mayor número de visitas en cada caso, proporcionando información clave sobre los puntos calientes o "hotspots" dentro del juego. La consulta ejecutada y parte del resultado se muestran en las Figuras 14 y 15, respectivamente.

```
5.
WITH conteo AS (
    SELECT
        e.id_episodio,
        m.id_mapa,
        (t.pos_x / 250) AS sector_x,
        (t.pos_y / 250) AS sector_y,
        COUNT(*) AS visitas
    FROM Telemetria t
    JOIN Partida p ON p.id_partida = t.id_partida
    JOIN Mapa m ON m.id_mapa = p.id_mapa
    JOIN Episodio e ON e.id_episodio = m.id_episodio
    GROUP BY
        e.id_episodio,
        m.id_mapa,
        sector_x,
        sector_y
),
ranking AS (
    SELECT
        *,
        ROW_NUMBER() OVER (
            PARTITION BY id_mapa
            ORDER BY visitas DESC
        ) AS rn
    FROM conteo
)
SELECT
    id_episodio,
    id_mapa,
    sector_x,
    sector_y,
    visitas
FROM ranking
ORDER BY
    id_mapa,
    visitas DESC;
```

***Fig 14**. Querie Ejecutado para la Consulta 5.*

id_episodio	id_mapa	sector_x	sector_y	visitas
1	3	0	0	76
1	3	1	0	54
1	3	-6	4	43
1	3	-4	-8	41
1	3	0	3	36
1	3	-6	3	35
1	3	1	4	34
1	3	2	3	33
1	3	-7	-10	31
1	3	-5	1	30
1	3	2	4	27
1	3	-8	-8	25
1	3	-6	1	25
1	3	-7	-8	24
1	3	-3	3	23
1	3	0	1	23
1	3	-4	3	22
1	3	-7	-9	21
1	3	-1	-3	20
1	3	0	-8	20

***Fig 15**. Resultado Consulta 5.*

9.5. Consulta 7

La consulta 7 busca calcular el puntaje promedio de experiencia de usuario (UX) únicamente entre los jugadores cuya trayectoria dentro del juego es de las más cortas, agrupando estos resultados por episodio. Esto es importante porque permite identificar cómo perciben el juego aquellos usuarios que abandonan rápido o tienen poca participación, lo cual puede revelar problemas tempranos de experiencia, dificultad o diseño que afectan la retención. La consulta generalmente identifica primero a los jugadores con menor duración, los relaciona con sus puntajes UX y luego promedia dichos puntajes por cada episodio, ofreciendo así una visión crítica sobre qué tan bien está funcionando el juego para los jugadores con menor compromiso. La consulta ejecutada y su resultado se muestran en las Figuras 16 y 17, respectivamente.

```sql
WITH siguiente AS (
    SELECT t1.*, (
        SELECT t2.tiempo
        FROM Telemetria t2
        WHERE t2.id_partida = t1.id_partida
          AND t2.tiempo > t1.tiempo
        ORDER BY t2.tiempo
        LIMIT 1
    ) AS tiempo_sig
    FROM Telemetria t1
),
distancias AS (
    SELECT
        e.id_episodio, p.id_jugador,
        SQRT(
            POWER(t2.pos_x - t1.pos_x, 2) +
            POWER(t2.pos_y - t1.pos_y, 2) +
            POWER(t2.pos_z - t1.pos_z, 2)
        ) AS dist
    FROM siguiente t1
    JOIN Telemetria t2
        ON t2.id_partida = t1.id_partida
       AND t2.tiempo = t1.tiempo_sig
    JOIN Partida p ON p.id_partida = t1.id_partida
    JOIN Mapa m   ON m.id_mapa = p.id_mapa
    JOIN Episodio e ON e.id_episodio = m.id_episodio
),
minimos AS (
    SELECT id_episodio, id_jugador, SUM(dist) AS total_dist
    FROM distancias
    GROUP BY id_episodio, id_jugador
),
ganadores AS (
    SELECT m1.id_episodio, m1.id_jugador
    FROM minimos m1
    WHERE m1.total_dist = (
        SELECT MIN(m2.total_dist)
        FROM minimos m2
        WHERE m2.id_episodio = m1.id_episodio
    )
)
SELECT
    g.id_episodio, u.id_usuario, u.nombre_usuario, AVG(r.calificacion) AS promedio_calificacion
FROM ganadores g
JOIN Jugador j ON j.id_jugador = g.id_jugador
JOIN Usuario u ON u.id_usuario = j.id_usuario
JOIN Responde r ON r.id_usuario = u.id_usuario
GROUP BY g.id_episodio, u.id_usuario, u.nombre_usuario
ORDER BY g.id_episodio, u.id_usuario;
```

***Fig 16**. Querie Ejecutado para la Consulta 7.*

```
 id_episodio | id_usuario | nombre_usuario | promedio_calificacion
-------------+------------+----------------+-----------------------
           1 |          5 | JuanMateus     |    5.2222222222222222
(1 row)
```

***Fig 17**. Resultado Consulta 7.*

10. Implementación de Índices en las Consultas

En el proyecto se implementaron algunos índices con el propósito de optimizar el rendimiento de las consultas analíticas más exigentes, especialmente aquellas que requerían recorrer grandes volúmenes de datos o realizar uniones y filtros frecuentes. Los índices permiten acelerar significativamente la búsqueda y recuperación de información al evitar escaneos completos de tablas, lo cual es fundamental en bases de datos orientadas tanto al análisis como al procesamiento eficiente de información. Su incorporación no solo mejora la velocidad de ejecución de las consultas, sino que también contribuye a un diseño más robusto y escalable, demostrando la importancia de considerar estrategias de indexación adecuadas

como parte esencial de la arquitectura de cualquier sistema de datos.

10.1. Índice 1, para Consulta 5

Se creó el índice *idx_partida_id* ejecutando:

```
Ceate Index idx_partida_id_mapa On Partida (id_mapa);
```

Con el objetivo de optimizar una de las consultas analíticas más exigentes del proyecto, la cual requiere agrupar información de telemetría por episodios, mapas y sectores. Este índice resulta especialmente útil porque permite acelerar el proceso de búsqueda y emparejamiento entre la tabla Partida y la tabla Mapa, reduciendo la necesidad de realizar escaneos completos y permitiendo que el motor de la base de datos localice rápidamente todas las partidas asociadas a un mapa específico. Gracias a esta indexación, las operaciones de JOIN y agrupación se ejecutan de forma más eficiente, disminuyendo considerablemente los tiempos de respuesta de la consulta. La Figura 18 presenta la diferencia en los tiempos obtenidos luego de ejecutar varias veces el *Explain Analyze* antes y después de crear el índice.

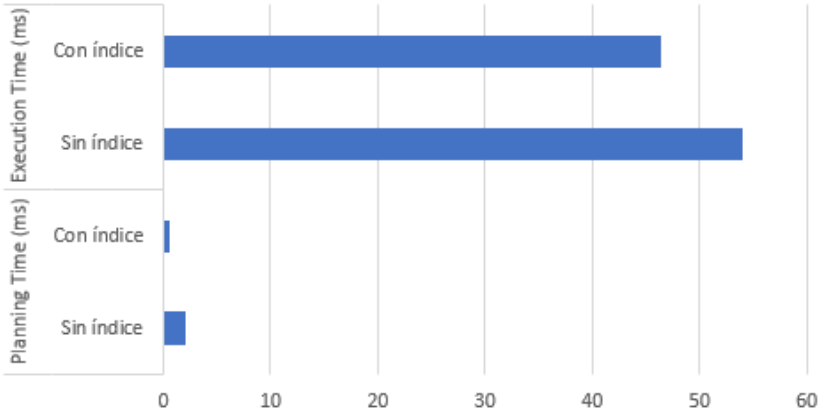

***Fig 18**. Consulta 5 Antes y Después del índice.*

Se observó una mejora importante en el promedio del tiempo de planeación, pasando de un promedio de 1.995 ms a 0.586 ms, lo que evidencia que el optimizador de PostgreSQL pudo identificar rutas de ejecución más eficientes gracias a la disponibilidad del índice. En cuanto al promedio del tiempo de ejecución, este disminuyó de 53.998 ms a 46.42 ms, reflejando una mejora moderada pero significativa considerando el volumen de datos procesado en la consulta.

10.2. Índice 2, para Consulta 4

Se creó el índice *idx_partida_id_jugador* ejecutando:

```
Create Index idx_partida_id_jugador On Partida (id_jugador);
```

Con el objetivo de optimizar la consulta 4 que relaciona partidas con jugadores, usuarios y sus respuestas, y que además filtra por la duración de las partidas por encima del promedio. Este índice acelera la localización de las partidas asociadas a un jugador concreto, reduciendo la necesidad de escanear toda la tabla Partida durante las operaciones de JOIN hacia Jugador y Usuario y las posteriores uniones con Responde. Al permitir accesos más directos por id_jugador, el motor de la base de datos puede escoger rutas de ejecución con menos coste, lo que disminuye la cantidad de filas que deben procesarse en etapas posteriores de la consulta y mejora la eficiencia global del procesamiento. La Figura 19 presenta la diferencia en los tiempos obtenidos luego de ejecutar varias veces el *Explain Analyze* antes y después de crear el índice.

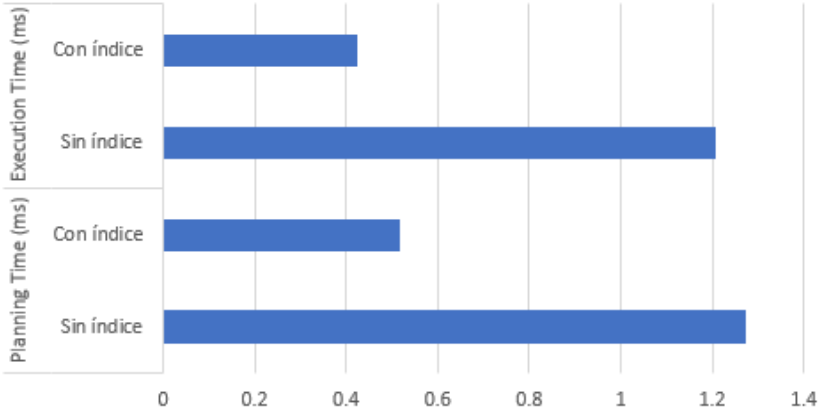

***Fig 19**. Consulta 4 Antes y Después del índice.*

Se observó una reducción clara en los promedios de tiempo. El tiempo de planeación pasó de 1.273 ms a 0.519 ms, y el tiempo de ejecución promedio disminuyó de 1.207 ms a 0.425 ms. Estas mejoras indican que el optimizador encontró planes más eficientes gracias al índice y que la ejecución también requirió menos trabajo efectivo al aprovechar búsquedas indexadas en las uniones. Cabe resaltar que, aunque la consulta realiza cálculos sobre la duración de las partidas, el beneficio proviene de procesar menos filas y de acelerar los JOIN centrales.

10.3. Índice 3, para Consulta 3

Se creó el índice *idx_telemetria_posicion* ejecutando:

```
CREATE INDEX idx_telemetria_posicion ON Telemetria
    (id_partida, pos_x, pos_y, pos_z);
```

Con el propósito de optimizar la consulta 3, probablemente la consulta más pesada del proyecto, enfocada en calcular las trayectorias recorridas por los jugadores a partir de diferencias sucesivas en coordenadas espaciales. Este índice resulta especialmente útil debido a que la consulta realiza múltiples uniones dentro de la misma tabla Telemetria para

encontrar posiciones consecutivas, además de filtrar constantemente por id_partida. Al indexar conjuntamente el identificador de partida y las coordenadas, el motor de la base puede ubicar con mayor rapidez los registros relevantes y reducir escaneos sobre una tabla particularmente grande. Como resultado, las operaciones de JOIN entre puntos consecutivos y los cálculos de distancias se ejecutan con mayor eficiencia, disminuyendo el costo de la consulta. La Figura 20 presenta la diferencia en los tiempos obtenidos luego de ejecutar varias veces el *Explain Analyze* antes y después de crear el índice.

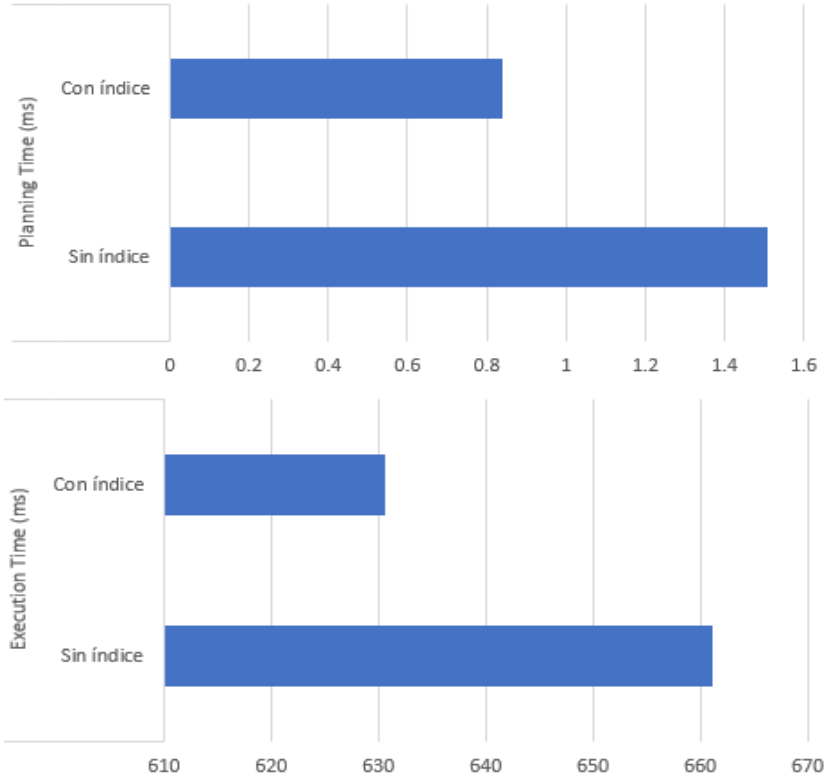

***Fig 20**. Consulta 3 Antes y Después del índice.*

Se observó una mejora medible en los tiempos, el tiempo de planeación pasó de 1.508 ms a 0.839 ms, mientras que el de ejecución pasó de 661.077 ms a 630.682 ms. A pesar de que la mejora en ejecución es moderada, refleja la utilidad del índice al reducir el volumen de datos que deben analizarse en cada unión. Sin embargo, la consulta sigue siendo costosa por naturaleza, debido a que requiere un self-join sobre Telemetria para encontrar el siguiente punto por registro, además de ejecutar una subconsulta y múltiples cálculos geométricos para obtener las distancias. Estas operaciones son inherentemente pesadas y difíciles de optimizar, ya que dependen del procesamiento fila por fila, por ello, aunque el índice mejora el rendimiento, la complejidad del análisis de trayectorias hace que la consulta siga teniendo tiempos de ejecución elevados.

11. Creación de Vistas y Vista Materializada

En el proyecto se implementaron dos vistas y una vista materializada con el fin de optimizar la organización y consulta de la información dentro de la base de datos. Las vistas funcionan como tablas virtuales que almacenan consultas predefinidas, permitiendo simplificar análisis complejos y ofrecer una capa adicional de abstracción sin duplicar datos. Por su parte, la vista materializada almacena físicamente el resultado de una consulta, lo que acelera significativamente el acceso a información que se utiliza con frecuencia o requiere cálculos pesados. En conjunto, estas estructuras mejoran el rendimiento, facilitan la mantenibilidad del sistema y permiten que las consultas críticas se ejecuten de manera más eficiente.

11.1.1. Vista 1: Duración promedio de las partidas por mapa

La Vista 1 se crea con el propósito de facilitar el análisis del comportamiento de los jugadores en cada entorno del juego, permitiendo identificar rápidamente cuáles mapas generan partidas más largas o más cortas. Esta vista ofrece la ventaja de simplificar una consulta que puede ser costosa y repetitiva, evitando cálculos manuales cada vez que se requiere evaluar la eficiencia o complejidad de un mapa. Además, centraliza la información en una estructura clara y accesible, lo que mejora la toma de decisiones sobre balance y diseño, y agiliza la generación de reportes dentro del proyecto. El código ejecutado en PostgreSQL para la creación de esta vista y lo que se desea visualizar se muestra en las Figuras 21 y 22, respectivamente.

```
1. /
CREATE VIEW vw_duracion_promedio_mapa AS
SELECT
    p.id_mapa,
    m.Nombre AS nombre_mapa,
    AVG(EXTRACT(EPOCH FROM (p.Hora_fin - p.Hora_inicio))) AS duracion_promedio_seg
FROM Partida p
JOIN Mapa m ON m.id_mapa = p.id_mapa
GROUP BY p.id_mapa, m.Nombre;
```

***Fig 21**. SQL Implementado Vista 1.*

```
doomproject=# SELECT *
FROM vw_duracion_promedio_mapa;
 id_mapa |   nombre_mapa    | duracion_promedio_seg
---------+------------------+-----------------------
       7 | Computer Station | 6149.5000000000000000
       3 | Toxin Refinery   |  825.0000000000000000
       5 | Phobos Lab       | 4875.0000000000000000
(3 rows)
```

***Fig 22**. Datos que muestra la Vista 1.*

11.1.2. Vista 2: Partidas de cada jugador

La Vista 2 se diseñó para ofrecer una forma clara y directa de consultar todas las partidas asociadas a cada usuario, permitiendo analizar su actividad, frecuencia de juego y patrones de participación. Esta vista aporta la ventaja de unificar información que normalmente requeriría múltiples uniones o filtros, simplificando significativamente el acceso a los datos y reduciendo la complejidad de las consultas recurrentes. Su creación responde a la necesidad de contar con una herramienta eficiente para evaluar el rendimiento individual, hacer seguimiento al progreso de los jugadores y facilitar el desarrollo de análisis estadísticos dentro del proyecto. El código ejecutado en PostgreSQL para la creación de esta vista y lo que se desea visualizar se muestra en las Figuras 23 y 24, respectivamente.

```
2.
CREATE VIEW VistaPartidasJugador AS
SELECT
    u.id_usuario,
    u.nombre_usuario,
    j.id_jugador,
    j.nombre AS nombre_jugador,
    e.nombre AS episodio,
    m.nombre AS mapa,
    p.id_partida,
    p.hora_inicio,
    p.hora_fin
FROM Partida p
JOIN Jugador j ON p.id_jugador = j.id_jugador
JOIN Usuario u ON j.id_usuario = u.id_usuario
JOIN Mapa m ON p.id_mapa = m.id_mapa
JOIN Episodio e ON m.id_episodio = e.id_episodio;
```

***Fig 23**. SQL Implementado Vista 2.*

id_usuario	nombre_usuario	id_jugador	nombre_jugador	episodio	mapa	id_partida	hora_inicio	hora_fin
1	EstebanChaustre	123	EstebiPRO	Knee-Deep in the Dead	Computer Station	4	2025-11-08 16:52:42	2025-11-08 18:46:28
2	SamuelBorras	456	Borrium	Knee-Deep in the Dead	Phobos Lab	1	2025-11-08 01:37:41	2025-11-08 03:05:11
3	DanielPrieto	789	Danidih	Knee-Deep in the Dead	Computer Station	6	2025-11-08 21:03:55	2025-11-08 22:35:08
4	JulioTriana	444	Jeelian	Knee-Deep in the Dead	Phobos Lab	5	2025-11-08 19:10:40	2025-11-08 20:25:40
5	JuanMateus	100	Juanmatro	Knee-Deep in the Dead	Toxin Refinery	3	2025-11-08 16:12:20	2025-11-08 16:24:11
6	AlejandraPerdomo	126	Alepero	Knee-Deep in the Dead	Toxin Refinery	2	2025-11-08 14:32:13	2025-11-08 14:47:52

(6 rows)

***Fig 24**. Datos que muestra la Vista 2.*

11.2. Vista Materializada: Promedio de respuestas cuestionario por usuario

La Vista Materializada propuesta se implementó para acelerar el acceso a un cálculo que se utiliza con frecuencia y que requiere procesar grandes cantidades de datos, como es el promedio de calificaciones otorgadas por cada jugador en el cuestionario. A diferencia de una vista tradicional, esta vista materializada almacena físicamente los resultados de la consulta, lo que reduce significativamente el tiempo de respuesta en análisis repetitivos o en reportes que se generan de manera constante. Su creación se justificó por la necesidad de optimizar el rendimiento del sistema, garantizar consultas más rápidas y disponer de

información agregada de forma inmediata para la toma de decisiones dentro del proyecto. El código ejecutado en PostgreSQL para la creación de la vista materializada y lo que se desea visualizar se muestra en las Figuras 25 y 26, respectivamente.

```
Materialized View
CREATE MATERIALIZED VIEW MV_PromedioUXPorUsuario AS
SELECT
    u.id_usuario,
    u.nombre_usuario,
    COUNT(r.calificacion) AS total_respuestas,
    AVG(r.calificacion)   AS promedio_calificacion
FROM Usuario u
LEFT JOIN Responde r ON u.id_usuario = r.id_usuario
GROUP BY u.id_usuario, u.nombre_usuario;
```

***Fig 25**. SQL Implementado Vista Materializada.*

```
doomproject=# select * from MV_PromedioUXPorUsuario;
 id_usuario |  nombre_usuario   | total_respuestas | promedio_calificacion
------------+-------------------+------------------+-----------------------
          3 | DanielPrieto      |               18 |    6.1111111111111111
          5 | JuanMateus        |               18 |    5.2222222222222222
          4 | JulioTriana       |               18 |    6.2222222222222222
          6 | AlejandraPerdomo  |               18 |    6.4444444444444444
          2 | SamuelBorras      |               18 |    6.0000000000000000
          1 | EstebanChaustre   |               18 |    6.0555555555555556
(6 rows)
```

***Fig 26**. Datos que muestra la Vista Materializada.*

12. Creación de un Script para recrear el Esquema

Una vez finalizada la creación de las vistas y la vista materializada, se elaboró un script completo para generar la base de datos desde cero y poblarla con datos de ejemplo. Para este proceso se utilizaron los conjuntos de datos completos en todas las tablas, con la excepción de la tabla Telemetría, en la cual solo se insertó la información correspondiente a las primeras dos partidas con el fin de mantener el script más claro, manejable y evitar una extensión excesiva. El desarrollo del script se apoyó en el código SQL construido durante las entregas 2 y 3 del proyecto. Dado su tamaño y propósito técnico, el script no se adjunta directamente en el reporte, sino que fue cargado en el repositorio de GitHub del equipo, al cual el profesor tiene acceso.

13. Conclusiones

El desarrollo del proyecto permitió abordar de manera integral todas las etapas necesarias para la construcción de una base de datos completa y funcional, desde la definición

de requerimientos y la normalización hasta la implementación del esquema, la inserción de datos y la ejecución de consultas analíticas. La base de datos obtenida se ajusta de forma coherente al modelo planteado inicialmente en el diagrama entidad-relación normalizado, lo que demuestra una correcta traducción del diseño conceptual hacia una estructura lógica y física en PostgreSQL. La creación del diccionario de datos, los índices, las vistas y la vista materializada fortaleció la solidez del sistema y optimizó el acceso a la información, logrando un entorno consistente y adecuado para el análisis del comportamiento de los jugadores.

En cuanto a las implicaciones éticas, el manejo de datos de usuarios y de telemetría dentro de un proyecto de bases de datos exige considerar la responsabilidad asociada al tratamiento de información potencialmente sensible. Aunque los datos utilizados en este caso provienen de una versión modificada del juego, el proceso permitió reflexionar sobre la importancia de garantizar la privacidad, la integridad y el uso adecuado de la información en sistemas reales. Este ejercicio también reforzó la necesidad de diseñar estructuras de datos claras y transparentes que eviten manipulaciones incorrectas o interpretaciones erróneas, promoviendo el desarrollo de soluciones tecnológicas conscientes y alineadas con principios éticos.

Por último, uno de los aspectos más interesantes del proyecto fue la extracción de datos de telemetría desde una versión modificada de Doom, lo cual implicó aplicar estrategias creativas para interpretar y adaptar dichos datos al modelo relacional establecido. Aunque este proceso pudo generar ciertas dificultades y resultados atípicos en algunas consultas analíticas, también enriqueció la experiencia al exponer desafíos reales relacionados con la limpieza, integración y manejo de grandes volúmenes de información. A pesar de estas complejidades, el proyecto resultó altamente provechoso para profundizar en conceptos fundamentales de diseño, implementación y administración de bases de datos, aportando una comprensión más sólida y práctica sobre su funcionamiento en contextos reales.