Documentación técnica detallada para la implementación del modelo Entidad-Relación Extendido (EER) del proyecto **FanHub** de la fase 1 utilizando **Mermaid**, como primer borrador.

Este documento sirve como guía de estilos y sintaxis para asegurar que el diagrama visual represente fielmente las reglas de negocio y la teoría de bases de datos vista en clases, utilizando una aproximación a la notación de Chen del Modelo E/R - Extendido.

Documentación de Diseño: Modelo E/R Extendido con Mermaid.js

1. Introducción y Filosofía del Diseño

El objetivo de esta implementación es representar el esquema conceptual de la base de datos de **FanHub** utilizando "Diagram-as-Code". Dado que Mermaid.js está optimizado para flujos de trabajo, hemos adaptado sus estilos (CSS) y nodos para emular la **Notación de Chen**, utilizada en el modelado conceptual de bases de datos utilizando tecnologías actuales.

Principios Clave:

- **Visualización Semántica:** Los colores no son decorativos; agrupan las entidades por dominios de negocio (Usuarios, Dinero, Contenido, Social).
- **Conexiones Adireccionales:** A diferencia de los diagramas de flujo, en un modelo E/R las relaciones son asociaciones, no flujos de datos. Por ello, **se evita el uso de flechas (-->) salvo que queramos una implementación del modelo E/R simple**. Se utilizarán exclusivamente enlaces simples (---) para denotar conexión entre entidad y relación.
- **Jerarquía Visual:** Se distingue claramente entre elementos fuertes (independientes) y débiles (dependientes de existencia/identificación), tal como la notación del Modelo E/R-E con rombos simples o doble lineados respectivamente.

---

2. Paleta de Colores y Agrupación Temática

Para facilitar la lectura del diagrama extenso, se han definido clases CSS (classDef) que agrupan las entidades por su contexto funcional dentro de FanHub (solo para diseño visual).

Definición de Clases

Fragmento de código Mermaid

```
%% CLASES DE DOMINIO (ENTIDADES)
classDef usuario fill:#FFCCBC,stroke:#E64A19,stroke-width:2px,color:black
classDef creador fill:#FFAB91,stroke:#E64A19,stroke-width:2px,color:black
classDef finanzas fill:#C8E6C9,stroke:#388E3C,stroke-width:2px,color:black
classDef contenido fill:#B3E5FC,stroke:#0277BD,stroke-width:2px,color:black
classDef social fill:#E1BEE7,stroke:#7B1FA2,stroke-width:2px,color:black
```

Explicación de Uso:

1. **Dominio Usuario (:::usuario, :::creador):** Tonos Anaranjados.
   - Representa a los actores del sistema. Se usa para la entidad Usuario y su subtipo **Creador**[2]. El borde naranja oscuro (#E64A19) denota importancia crítica.

2. **Dominio Finanzas (:::finanzas):** Tonos Verdes.
   - Asociado al dinero y transacciones. Se aplica a Tarjeta, Suscripción, Nivel y **Factura**[3].

3. **Dominio Contenido (:::contenido):** Tonos Azules.
   - Representa los activos digitales. Aplica a Publicación y sus subtipos (Video, Encuesta, etc.)[4].

4. **Dominio Social (:::social):** Tonos Púrpuras.
   - Representa la gamificación e interacción. Aplica a Comentario, Reacción e **Insignia**[5].

---

3. Implementación de Entidades

3.1 Entidades Fuertes

Son aquellas que tienen identidad propia (su PK no depende de nadie). Se representan con rectángulos estándar.

- **Sintaxis Mermaid:** NombreEntidad[NOMBRE]:::clase
- **Ejemplo:** Usuario[USUARIO]:::usuario

3.2 Entidades Débiles

Son aquellas cuya existencia depende de una entidad fuerte (ej. **Factura** depende de **Suscripción** [6], OpciónVoto depende de **Encuesta** [7]).

- **Representación E/R:** Doble rectángulo.
- **Implementación Mermaid:** Se simula usando un borde grueso (stroke-width:3px) y

discontinuo (stroke-dasharray: 3 3) y un nodo de subrutina [[ ]].

- **Código:**
  Fragmento de código Mermaid
  ```
  classDef debil fill:#FFCDD2,stroke:#D32F2F,stroke-width:3px,stroke-dasharray: 3 3,color:black

  %% Uso
  Factura[[FACTURA]]:::debil
  ```

4. Implementación de Atributos

Mermaid no tiene "óvalos" nativos perfectos conectados, por lo que usamos la forma stadium (([text])) y circle ((( ))) con estilos específicos.

4.1 Atributos Simples y Claves Primarias (PK)

- **Atributo Simple:** Forma de estadio/ovalo, fondo claros para legibilidad.
- **Primary Key (PK):** Se diferencia por un borde negro sólido (stroke:#000) y texto en negrita. Además, **el texto debe incluir la etiqueta HTML <u> para el subrayado**, norma estándar en E/R.

Fragmento de código Mermaid

```
%% ESTILOS
classDef atributo fill:#f4e6fb,stroke:#555,stroke-width:2px,color:black,shape:stadium
classDef atributoPK
fill:#d6b4fc,stroke:#000,stroke-width:2px,color:black,shape:stadium,font-weight:bold

%% IMPLEMENTACIÓN
attr_id([<u>id_usuario</u>]):::atributoPK
attr_mail([email]):::atributo

%% CONEXIÓN (Sin flechas, adireccionales)
Usuario --- attr_id
Usuario --- attr_mail
```

4.2 Atributos Multivalorados

Son atributos que pueden tener múltiples valores para una misma entidad (ej. "Enlaces de Redes Sociales, Telefonos, Direcciones, etc." de un **Usuario** [8]).

- **Representación E/R:** Doble óvalo.
- **Implementación Mermaid:** Usamos la forma de círculo doble ((( ))) combinada con un estilo de línea discontinua (stroke-dasharray: 5 5) para generar el efecto visual de "doble

borde" o diferenciación clara.

- **Código:**
  Fragmento de código
  classDef atributoMulti fill:#f4e6fb,stroke:#555,stroke-width:2px,color:black,shape:circle,
  stroke-dasharray: 21 5

  attr_social(((Redes
Sociales)))::atributoMulti
  Creador --- attr_social

5. Implementación de Relaciones y Cardinalidad

Las relaciones se representan mediante rombos. Es crucial distinguir entre relaciones normales y relaciones de identificación (para entidades débiles).

NOTA: No se usa -->. Para este modelo siempre --- a excepción de que se quiera extender esta notación al modelo relacional, para efectos de simplicidad, se deja a implementación posterior; no es el caso de esta implementación. La dirección se infiere por la lectura lógica o la posición, pero en el modelo conceptual, la línea representa asociación bidireccional.

5.1 Relaciones Fuertes (Estándar)

- **Forma:** Rombo { }.
- **Color:** Amarillo/Dorado (#FBC02D).
- **Código:**
  Fragmento de código Mermaid
  classDef relacion
  fill:#FFF9C4,stroke:#FBC02D,stroke-width:2px,color:black,shape:diamond
  rel_publica{Publica}:::relacion

5.2 Relaciones Débiles (por Identificación o por Existencia)

Conectan una entidad débil con su fuerte.

- **Forma:** Rombo con borde discontinuo (stroke-dasharray: 5 5) y color anaranjado intenso (#EF6C00) para denotar dependencia estricta. Debido a que la notación de Mermaid para los flowchart no exite un rombo de línea doble, para esta primera implementación se tiene este borde discontinuo, esto se tomará solo para un primer borrador del código.
- **Código:**
  Fragmento de código
  classDef relacionDebil fill:#FFF3E0,
  stroke:#EF6C00,stroke-width:3px,color:black,shape:diamond,stroke-dasharray: 5 5

  rel_genera{Genera}:::relacionDebil

5.3 Cardinalidad

La cardinalidad (1:1, 1:N, M:N) se coloca como **etiqueta sobre la línea de conexión**.

- **Sintaxis:** Entidad -- "Cardinalidad" --- Relacion
- **Ejemplo Completo:**
  Fragmento de código
  %% Un Creador publica muchas (N) Publicaciones
  Creador -- "1" --- rel_publica
  rel_publica -- "N" --- Publicacion

6. Resumen del Código Base

Copia y pega este bloque al inicio de tu archivo Mermaid para habilitar todos los estilos documentados anteriormente y comenzar una implementación con todo lo anterior.

Nota: NO hay forma de representar los subtipos solapado y disjunto de una entidad, por lo que tomará esta herramienta como una primera versión para su posterior implementación usando draw.io, dia-installer.de, miro.com o algún software de edición de imágenes como Photoshop/Illustrator que permita trabajar con una exportación del borrador en png, svg, etc.

Fragmento de código Mermaid

```
flowchart TD
    %% --- DEFINICIÓN DE CLASES Y ESTILOS —
    %% 1 ENTIDADES GENERALES
    classDef entidad fill:#FFCCBC,stroke:#E64A19,stroke-width:2px,color:black
    classDef personalizada fill:#FFCCBC,stroke:#E64A19,stroke-width:2px,color:black
    %% 1.1 ENTIDADES POR TEMÁTICA
    classDef usuario fill:#FFCCBC,stroke:#E64A19,stroke-width:2px,color:black
    classDef creador fill:#FFAB91,stroke:#E64A19,stroke-width:2px,color:black
    classDef finanzas fill:#C8E6C9,stroke:#388E3C,stroke-width:2px,color:black
    classDef contenido fill:#B3E5FC,stroke:#0277BD,stroke-width:2px,color:black
    classDef social fill:#E1BEE7,stroke:#7B1FA2,stroke-width:2px,color:black
    %% 2. ENTIDADES DÉBILES (Doble borde simulado)
    classDef debil fill:#FFCDD2,stroke:#D32F2F,stroke-width:3px,
stroke-dasharray: 3 3,color:black
    %% 3. RELACIONES
    %% Fuerte (Rombo Sólido)
    classDef relacion fill:#FFF9C4,stroke:#FBC02D,stroke-width:2px,color:black,shape:diamond
    %% Débil (Rombo Punteado)
    classDef relacionDebil
```

```
fill:#FFF3E0,stroke:#EF6C00,stroke-width:3px,color:black,shape:diamond,
stroke-dasharray: 5 5

    %% 4. ATRIBUTOS
    %% Normal
    classDef atributo fill:#f4e6fb,stroke:#555,stroke-width:2px,color:black,shape:stadium
    %% Primary Key (Subrayado visual y borde negro)
    classDef atributoPK
fill:#d6b4fc,stroke:#000,stroke-width:2px,color:black,shape:stadium,font-weight:bold
    %% Multivalorado (Círculo punteado/doble simulado)
    classDef atributoMulti fill:#f4e6fb,stroke:#555,stroke-width:2px,color:black,shape:circle,
stroke-dasharray: 21 5

    %% ============================================================
    %% EJEMPLO DE USO CORRECTO (IMPLEMENTACIÓN APROXIMADA)
    %% ============================================================

    %% Definición de Entidades (nodos)
    Entidad[NOMBRE]:::entidad
    Attr1([<u>id</u>]):::atributoPK
    Attr2([nombre]):::atributo

    Rel{Realiza}:::relacion

    Accion[ACCIÓN]:::personalizada

    %% Conexiones con Cardinalidad
    Entidad --- Attr1
    Entidad --- Attr2

    Entidad -- "1" --- Rel
    Rel -- "N" --- Accion
```