



## Final - Tinder

Hecho por: Centurión, Franco; Idañez, Lucía; Lamas, Chabela

[Final DDS 20230211 - Tinder.pdf](#)

### Dudas con respuestas

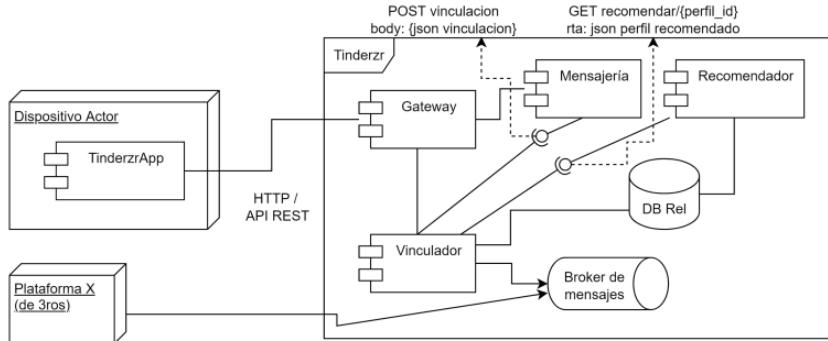
- Está bien poner una entidad intermedia como nombreTabla1\_nombreTabla2? No era que si nos quedaba así, algo no estábamos entendiendo del dominio? → a\_b no implica que directamente ESTE MAL o me esté olvidando una abstracción. Les dije varias veces que: si C es una entidad intermedia SIMPLE, entonces sí puede llamarse a\_b; pero si C es compleja y guarda más cosas, entonces sí probablemente esté representando una abstracción más relevante en mi dominio.
- No sabemos si el punto 1 de arquitectura está bien. La opción ideal sería ir por la propuesta de Master Slave. La caché no sería tan viable porque deberías cachear por cada usuario sus próx. 10, lo que de todas formas te consumiría muchos recursos (e inclusive más!)

### Notas

- Gateway → Microservicio
- Broker de mensajes aplica el patron Publicador Subscriptor que tiene temas, donde un suscriptor puede adherirse. En la cola de mensajes es un solo canal
- Desnormalización → Por consistencia de datos (como fue en el DentalBA con precio de tratamiento) o Performance (tener algo precalculado)
- OJO one to one

# Arquitectura

## Arquitectura



- **Gateway:** recibe todos los pedidos
- **Mensajería:** el Vinculador avisa cuando se forma un vínculo y habilita a este componente para poner en contacto a las partes. Los mensajes se gestionan en este componente.
- **Recomendador:** toma un perfil/usuario e indica cuál es el siguiente perfil a mostrar. En él está toda la lógica para maximizar la probabilidad de vínculos.
- **Vinculador:** Este es nuestro componente, en el cual se gestionan los perfiles, las vinculaciones y se coordinan las búsquedas
- Las **plataformas de 3ros** son los clientes de otras organizaciones, que consumen los mensajes de los vínculos recién establecidos.

1. Se reporta que cuando varios usuarios están buscando (pasando de uno a uno los perfiles), se genera una gran demora en todo el sitio, probablemente relacionado al intenso uso de la Data Base por parte del Recomendador.
  - a. Plantee un cambio en la arquitectura para resolver ese problema
  - b. Explique el funcionamiento del cambio propuesto.
  - c. Brevemente describa el/los mecanismos utilizados.

### RTA:

a.

(Opción 1) Se podría hacer que el Microservicio de "Recomendador" tenga su propia base de datos, logrando de esta manera poseer una menor cantidad de tablas que la DB Rel tiene de por sí. A su vez, se deberían replicar las tablas que requiera este servicio como es la de "Perfil".

Otra opción no excluyente con la anterior sería implementar una Caché para el recomendador, donde se guarden, por ejemplo, los próximos 10 perfiles. De esta manera, se harían muchas menos consultas a la DB.

(Opción 2) Se podrían tener dos bases de datos, exactamente iguales, con la arquitectura Master Slave, donde una de ellas se utilice únicamente para las lecturas y la otra para las escrituras. En este caso, se deben tener en cuenta los algoritmos de replicación de datos y balanceo de cargas.



La opción ideal sería ir por la propuesta de Master Slave. La caché no sería tan viable porque deberías cachear por cada usuario sus próx. 10, lo que de todas formas te consumiría muchos recursos (e inclusive más!).

b.

(Opción 2) Para el caso de la arquitectura master-slave, las escrituras/lecturas, en vez de impactar directamente en la BD, serán redistribuidas primero por un balanceador de cargas. Se podría establecer que en la instancia ESCLAVO leo mientras que en la MAESTRO escribo y leo a la vez. Esta arquitectura opera con, mínimo, dos bases de datos, de esta manera la cantidad de consultas que se procesan en simultáneo aumenta debido a que las consultas se distribuyen en distintas bases de datos y no siempre las atiende la misma base (máster). De esta manera, la cantidad de consultas que se procesan en simultáneo aumenta debido a que las consultas se distribuyen en distintas bases de datos y no siempre las atiende la misma base (máster). En este caso, para el componente Recomendador se podría utilizar una base de datos ESCLAVA ya que ésta únicamente lee.

(Opción 1) El funcionamiento sería el mismo solo que buscaría todos los datos en su base de datos propia, sin distinguir entre la escritura y lectura como en el caso anterior.

c. Las operaciones de escritura serán dirigidas a la BD Master y luego se escribirá en las bases de datos replica (Slave), de esta manera las operaciones de lectura serán dirigidas a cualquier base de datos.

2. Explique cómo el broker de mensajes informa a las plataformas externas con la información correspondiente. ¿Bajo qué circunstancia/s este componente provee tolerancia a fallos?

**RTA:** El broker de mensajes facilita que el componente vinculador emita mensajes bajo ciertos tópicos. Al mismo tiempo, las plataformas externas pueden consumir estos mensajes al suscribirse a tópicos específicos relacionados con los cuales los emisores publiquen. De esta forma, cuando el broker detecta que el vinculador emite un mensaje, se encarga de redirigirlo a todos aquellos componentes que expresaron interés en recibirllos.

Este componente provee tolerancia a fallos, ya que, en caso de que la Plataforma no esté disponible, el mensaje sigue "publicado" en el Broker hasta el momento en que se efectúe el envío del mensaje a la Plataforma.

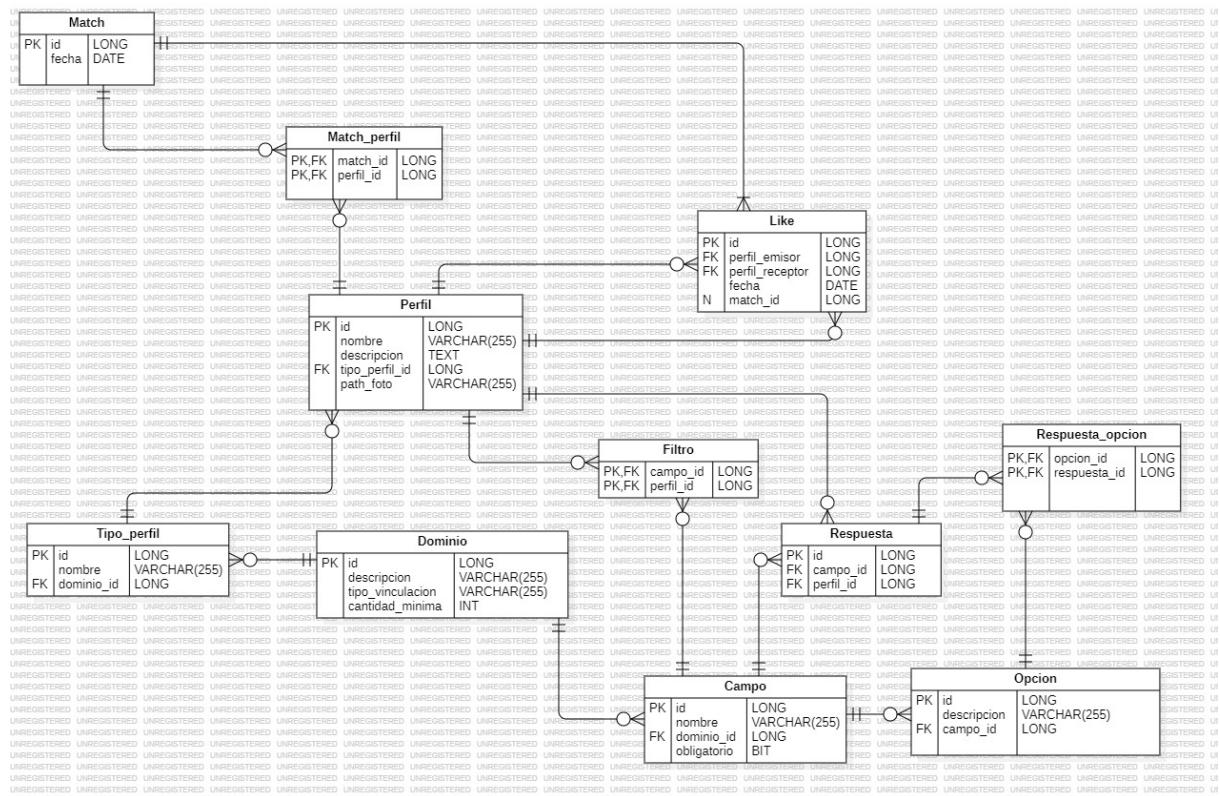
3. Se propone un patrón Adapter para enviar mensajes a los distintos interesados a través del broker de mensajes. ¿Qué opina al respecto? Justifique.

**RTA:** El patrón Adapter saliente para el vinculador no sería de gran conveniencia ya que no tiene sentido que el broker delegue el trabajo en un adapter como se plantea. En este dominio, al ya saber que nos vamos a integrar a un broker, no tiene sentido implementar un adapter porque la estrategia para que nuestro Vinculador notifique siempre va a ser a través del Broker.

# Persistencia

Diseñar el modelo de datos del componente “**Vinculador**” para poder persistir en una base de datos relacional a través de un ORM.

1. Armar la especificación usando un DER físico. Indicando las entidades, sus campos, claves primarias, las foráneas, cardinalidad, modalidad y las restricciones según corresponda.



2. Justificar:

- Qué elementos del modelo es necesario persistir y cuáles no.
- Cómo resolvió los impedance mismatches.
- Las estructuras de datos que deban ser desnormalizadas, si corresponde.

## Elementos necesarios de persistir

- Match

- Perfil
- Dominio
- Like
- Filtro
- Campo
- TipoPerfil
- Valor
- TipoVinculación
- Foto

## Decisión de diseño

- Consideramos que un like está relacionado con un solo match.
- Existe una relación al estilo “Pregunta-Respuesta” con las relaciones entre Dominio (encuesta), Campo (pregunta) y Valor (respuesta). A su vez, es necesario guardar los valores seleccionados por el usuario en su respectivo perfil.

## Impedance mismatch

Tipo de dato objetos	Tipo de dato relacional
LocalDate	SMALLDATETIME
String	VARCHAR(255)
Int	INTEGER(11)

El impedance mismatch de fecha lo resolvemos usando un converter que pase del tipo de dato de objeto LocalDate a SMALLDATETIME del mundo relacional.

## Estructuras desnormalizadas

Ninguna 😎