Un caso particular...

//Métodos de pago en MeLi

IT BOARDING

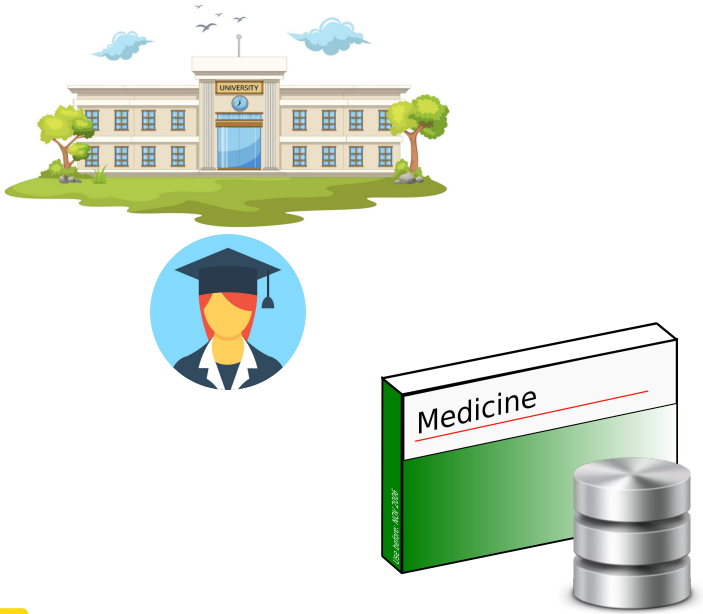
BOOTCAMP

Repensando la Capa de Persistencia

- El mundo de los pagos está viviendo muchos **cambios** y en **MeLi** se está repensando la capa de persistencia.

- Actualmente está basada en **BDs relacionales**, pero se busca realizar un cambio a no relacionales basadas en el concepto de **Key Value**.

¿Por qué guardamos transacciones críticas de esta manera?



- Una de las primeras razones es la **influencia académica**. En la universidad es común introducir al mundo de las bases de datos relacionales y al proceso de **normalización de la información**.

- Además, por sus características generales, las **BD Relacionales** son un *"remedio de amplio espectro"* que nos permite resolver **amplia gama de problemas** (como en este caso, almacenar pagos).

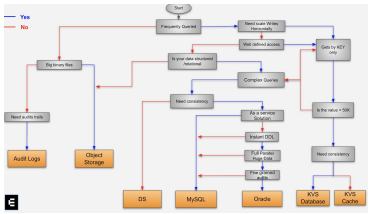
¿Por qué guardamos transacciones críticas de esta manera?

- Si suponemos un profesional que empezó a trabajar en MeLi, el mismo comenzará de a poco a hacer sus APIs y a emplear sus conocimientos para modelar su capa de persistencia y sus entidades...

Usando la herramienta correcta...

- Este profesional tomó sus conocimientos y validado por un árbol de decisión realizado con otro/a compañero/a, escogió la herramienta “correcta” para resolver el problema.

+



Escenario

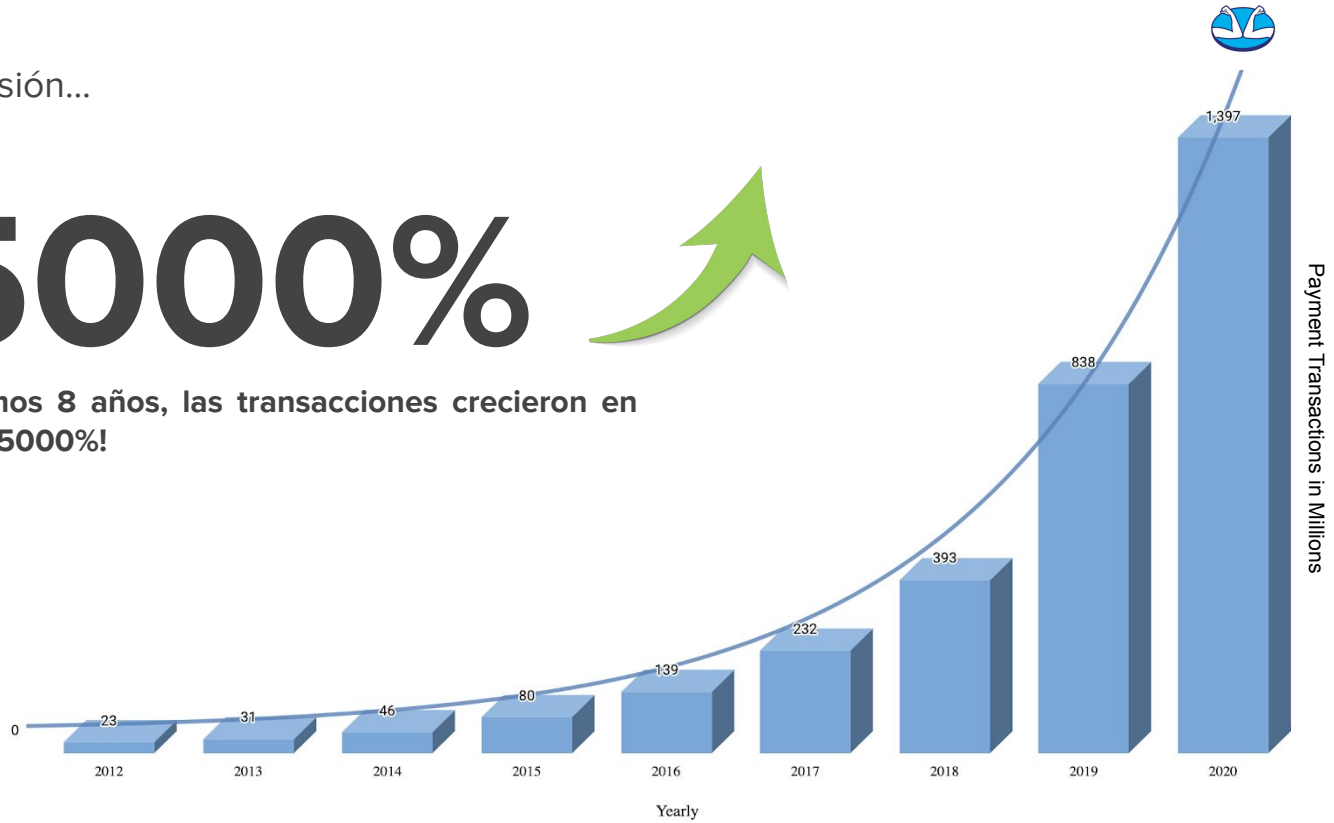
BD Relacional

¡Los pagos crecieron exponencialmente!

- Y con esta decisión...

+5000%

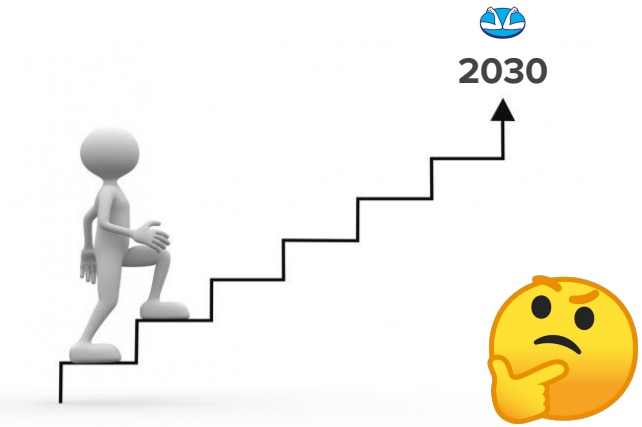
En los últimos 8 años, las transacciones crecieron en más de un ¡5000%!



Sin embargo....

- Lo que nos trajo hasta aquí

- No es precisamente lo que nos permitirá seguir creciendo



¿Qué significa esto?

- Este crecimiento no viene sin efectos secundarios
- El tráfico aumentó en GRAN medida a nivel de transacciones
- Todo esto trae grandes inconvenientes tanto de performance como de escalabilidad

¿Qué decisión debería ser tomada?

//Conclusiones

IT BOARDING

BOOTCAMP

Usar la “herramienta correcta” no siempre es suficiente

- Cuando la **“herramienta correcta”** empieza a no ser suficiente para acompañar el **crecimiento del negocio**, es hora de tomar otra decisión...

Puntos de pánico (Pain Points)

- **High Availability Issues**

Las **BD relacionales** actuales no están preparadas para brindar una robusta alta disponibilidad, ya que **no son sistemas distribuidos (sino centralizados)** lo que puede llevar a algún tipo de **Single Point of Failure (SPOF).**

- **Unable to scale horizontally**

Las **BD relacionales** tampoco escalan horizontalmente (a nivel escritura) y esto es un **problema grave** en un **ecosistema que crece exponencialmente**.

Puntos de pánico (Pain Points)

- **Modeling JSON objects in a normalized relational database lacks flexibility**

Modelar objetos **JSON** complejos sobre esquemas normalizados no brinda demasiada flexibilidad ya que cada cambio en un objeto implica un **DML** a la **base de datos**.

- **Database connections pools are complex to configure**

Los **pools de conexiones** suelen ser **muy difíciles de configurar**, sobre todo al intentar optimizar al máximo los accesos a la **base de datos.**

¿Por qué elegir NoSQL?

//Bases de datos No Relacionales

IT BOARDING

BOOTCAMP

¿Por qué NoSQL?

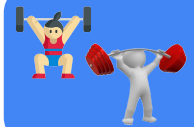
- **High Availability**

Las **BD No Relacionales** tienen la capacidad de ser **altamente disponibles**, ya que en caso de tener algún inconveniente, se degrada la información en forma parcial.

- **Horizontal Scalability**

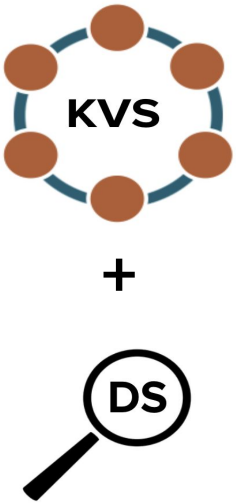
Permiten el crecimiento horizontal tanto en escrituras como en lecturas de forma **consistente**.

- **Adaptive Consistency**



Brindan flexibilidad para recuperar información consistente o eventualmente consistente según el caso.

¿En qué mejorarían nuestras capacidades?



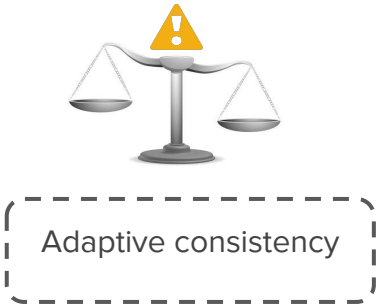
- La combinación de **Fury** + el **KVS** (Key Value Store) + **DS** (Document Search) cumple con las capacidades necesarias para escalar las apps de **MeLi** entre ellas:
  - **Alta disponibilidad** (High availability) ✔
  - **Escalabilidad Horizontal** (Horizontal Scalability) ✔
  - **Consistencia adaptable** (Adaptative Consistence) }

Pero existe un inconveniente....

- Como todo escenario de la vida, es **prácticamente imposible** tener absolutamente **todas las características necesarias**, por lo cual es importante encontrar un **“balance”** priorizando lo que se necesita en mayor medida.

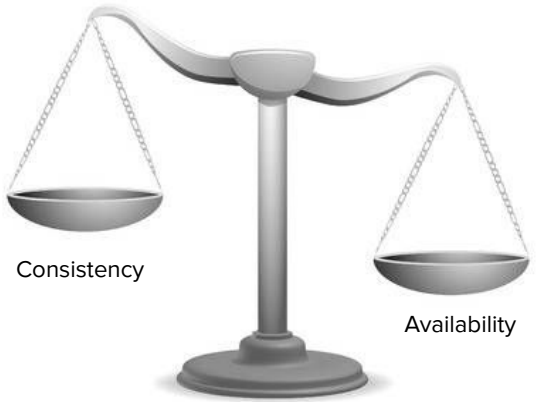
Compensando...

- La compensación en la elección de **KVS** + **DS** se encuentra en la replicación asincrónica existente entre los dos.
- **KVS** ofrece consistencia fuerte con el acceso exclusivo por key
- **DS** complementa todo el espectro de consultas que **KVS** no ofrece
- Sin embargo, la replicación asincrónica no hace posible la consistencia fuerte entre ambos



Siempre hay un “balance”

- El dilema está en si priorizar la **disponibilidad** o la **consistencia**.
- La balanza se inclina hacia la **disponibilidad**, dado que es necesario escalar para acompañar el c**recimiento del negocio**.
- Sin embargo, hay formas de **minimizar el impacto** que esta decisión pueda traer consigo.



De Relacional a No-Relacional

//Conceptos claves para esta migración

IT BOARDING

BOOTCAMP

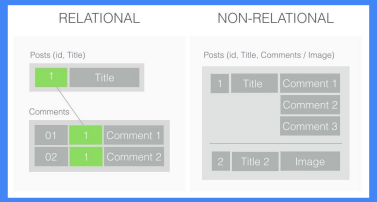
Aspectos Claves

- **Crea una lista de TODOS tus patrones de acceso**

- Definir cuáles de estos son **críticos para la aplicación**, ya sea por **performance o consistencia**.
- Direccionar al **KVS** la mayor cantidad de **accesos críticos** identificados.
- El resto de los accesos que requieran de consultas más complejas pueden utilizar **almacenamientos secundarios** (DS, Audtis, etc)

Aspectos Claves

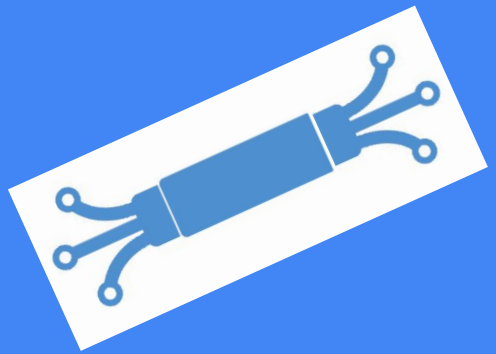
- **Desnormaliza las tablas normalizadas**



- Identificar las claves principales (ID/PK).
- Todas las relaciones basadas en una clave externa (FK) deben almacenarse bajo la misma clave principal.
- Representar las relaciones one-to-one con maps y relaciones one-to-many con listas de maps.

Aspectos Claves

- **Habilitar el feed del KVS y utilizalo para replicar datos**



- **Hacia Almacenamientos secundarios:**
  - Document Search
  - Audits Logs
  - Other Relational Databases
- **Hacia Sistemas de eventos:**
  - Enviar eventos a BigQueue
- **Hacia cualquier otro requerimiento aplicativo**

// En Conclusión...

“Usar la herramienta que parece correcta para el trabajo... No siempre es suficiente”.

Gracias

IT BOARDING

BOOTCAMP