



# Elgakk

[Final DDS 20221210.pdf](#)

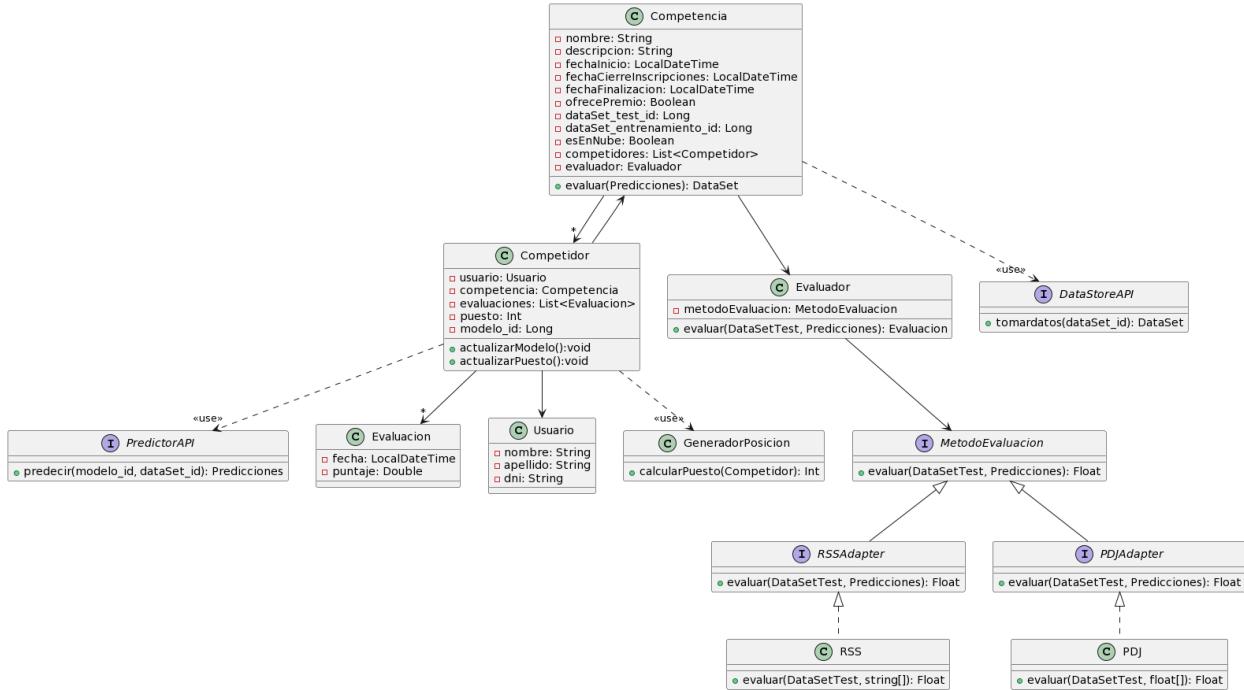
**Clase de consulta:**

[https://drive.google.com/drive/folders/1pdfTlqmxis\\_fehLNyXDcsdf4u3yyqhAH](https://drive.google.com/drive/folders/1pdfTlqmxis_fehLNyXDcsdf4u3yyqhAH)

**Hecho por:** Facundo Piaggio, Chabela Lamas

## Modelado de objetos

### Diagrama de clases



## Decisiones de diseño

Para comenzar, se utilizan atributos como el `modelo_id` o `dataset_id` ya que es información que no encuentra directamente guardada en nuestro modelo sino que tenemos la referencia. Por ejemplo, la información del `dataSet` se encuentra en el `DataStore` que nos comunicamos con ella a través de la API “`DataStoreAPI`”. Cabe aclarar que, las dos APIs del modelo pueden ser representadas como Singleton (no lo representamos en el diagrama pero puede ser una alternativa, sino que las entidades que lo llaman tengan guardada una instancia). Nosotros optamos por una instancia de las apis en la capa de controladores.

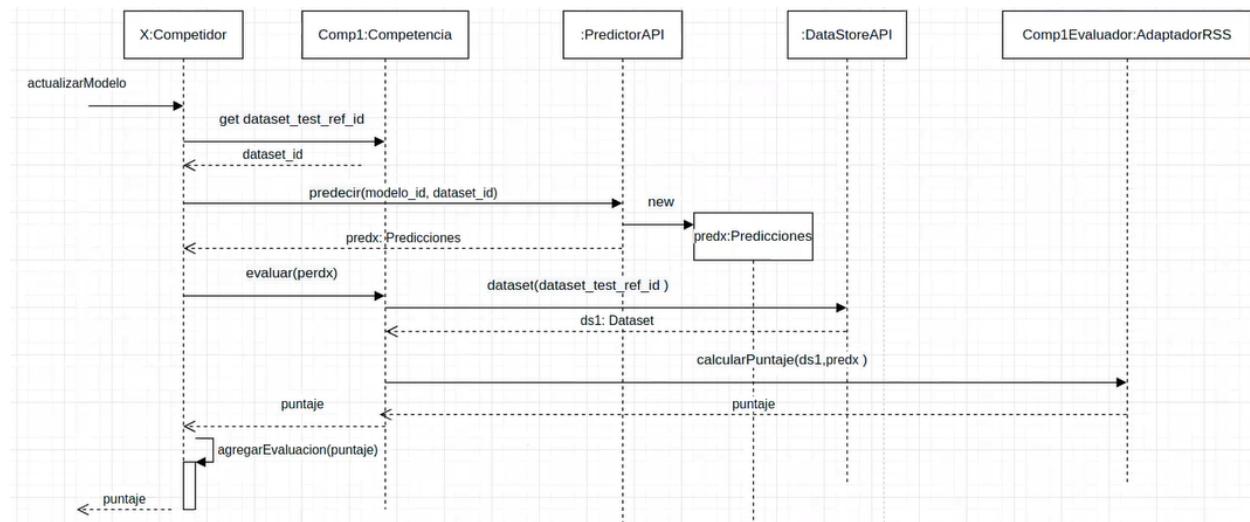
De la misma manera, un competidor representa la instancia del usuario en una competencia ya que éste puede estar en varias al mismo tiempo. A su vez, en la competencia existe un atributo que es llamado “`esEnNube`”, un boolean que representa el método de entregable presentado.

Para mantener el dominio de la manera más cohesiva y menos acoplada posible, realizamos dos clases: `Evaluador` y `GeneradorDePuesto`. Por consiguiente, ganamos testeabilidad ya que es más fácil encontrar fallas en la ejecución de las clases. Asimismo, se usa un Patrón Adapter del método de evaluación con el fin de delegar la responsabilidad en objetos particulares, aumentando de esta manera la mantenibilidad y la cohesión al no tener acoplamiento entre la clase adaptada y el cliente.

Por último, consideramos que el puesto de los competidores puede ser actualizado con una CronTask que ejecute el proceso cada cierto tiempo.

Explique con un diagrama de secuencia, pseudocódigo o prosa el caso de uso: el competidor Y sube nuevos resultados para la competencia 3.

### Caso de uso: Evaluar competidor

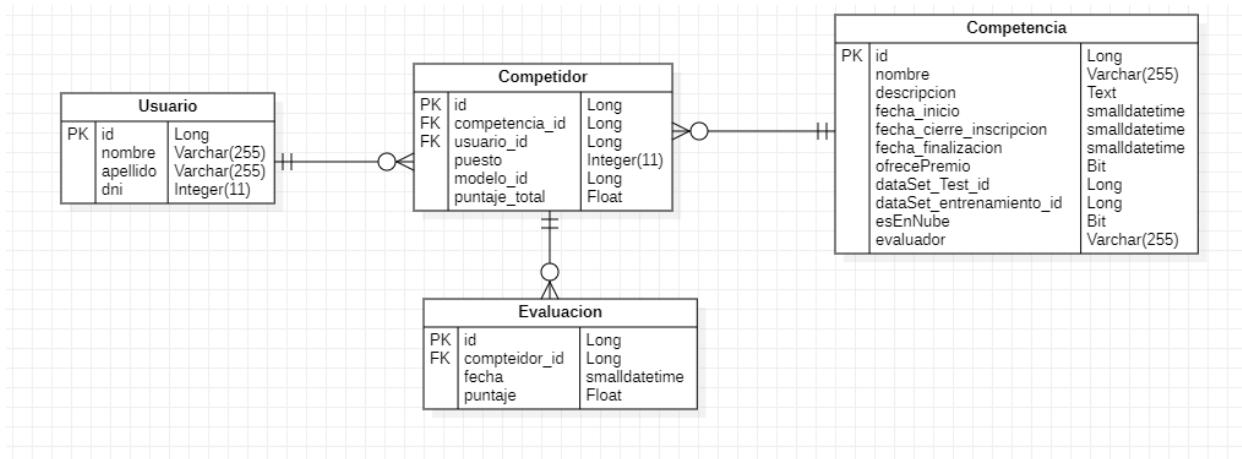


Explique con un diagrama de secuencia, pseudocódigo o prosa el caso de uso: el competidor X dice que su notebook tiene un nuevo modelo para la competencia

Es parecido al anterior nada más que el modelo lo ingresa el competidor y se generan las predicciones desde cero, comparándolas con los tests de la competencia dada

## Persistencia

### Modelo de datos



## Decisiones de diseño

Realizamos un @Embeded de la clase Evaluador en Competencia que contiene un converter del Método Evaluación.

## Entidades necesarias a persistir

- Competidor
- Competencia
- Usuario
- Evaluación

## Impedance mismatch

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

Respecto al impedance mismatch de Identidad, tuvimos que agregar un campo "id" (clave subrogada) para identificar únicamente a cada una de nuestras entidades.

## Modelo desnormalizado

Si, se desnormaliza el puntaje total del competidor por performance para que cuando se calcule el puesto no se deba realizar la suma de cada evaluación individual. A su vez, el puntaje total se actualiza por un trigger de insert de la Tabla Evaluación.

Explicar cómo se almacenan los datos para el caso de uso:  
“Generar Tabla de posiciones para la competencia X”

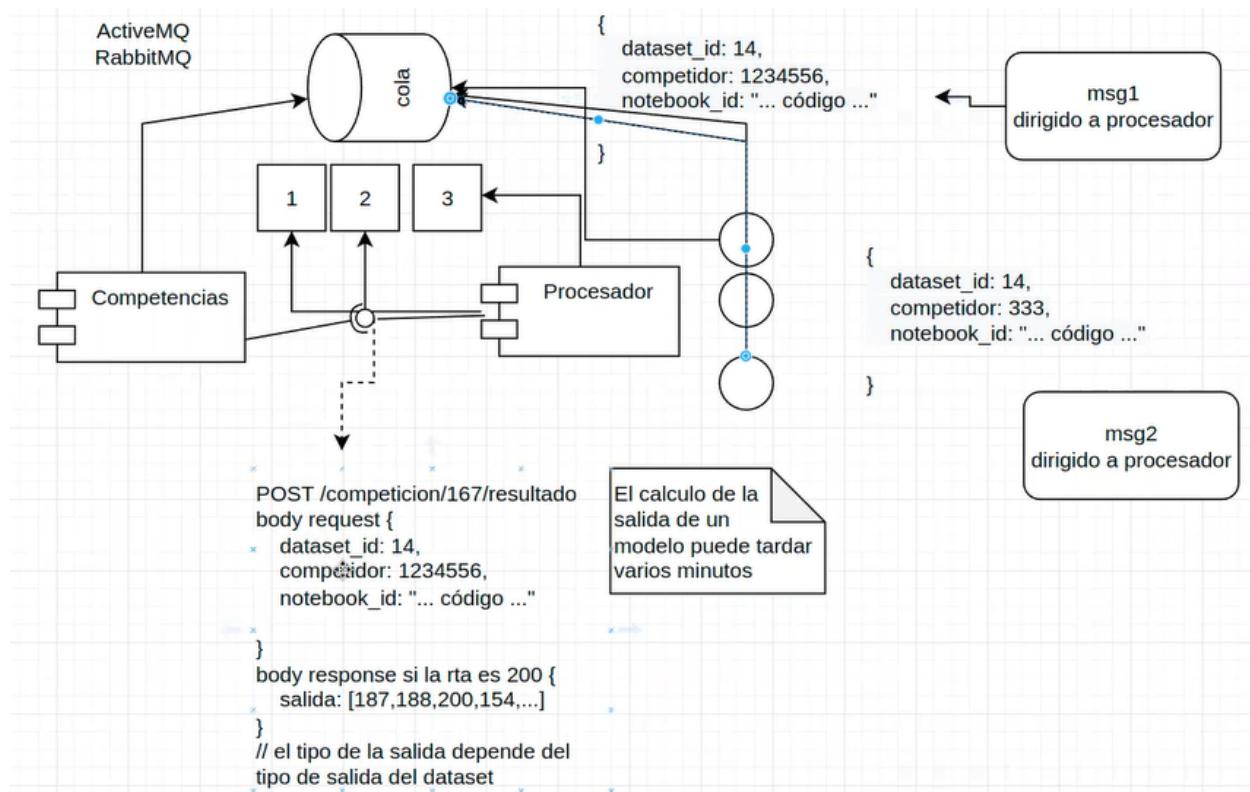
El dato del puesto lo almacenamos en el Competidor y mencionamos que éste se actualiza mediante el uso de una CronTask que implementa la clase GeneradorPuesto que analiza los puntajes de los competidores. En el controller, cuando se intenta ejecutar este caso de uso, se ordena a los competidores por su puesto de mayor a menor, para presentarlos en la vista en el formato de tabla.

## Arquitectura

1. Se reporta que cuando varios competidores suben una nueva implementación de sus modelos al mismo tiempo, se generan grandes demoras en la respuesta y varios procesos dentro del componente Procesador se terminan abruptamente por falta de recursos.
  - a. Modifique la arquitectura para evitar estas demoras (haga el diagrama), explique las modificaciones
  - b. De un ejemplo de cómo funciona la subida del nuevo modelo, use el diagrama anterior para explicar
  - c. A nivel interfaz de usuario, ¿Qué cuidado debería tener si el proceso de carga es largo ?  
(varios minutos)

a. Utilizar una cola de mensajes para realizar el proceso de manera asíncrona con el fin de no saturar el sistema. El procesador (consumidor) es aquel que consume los mensajes ingresados a la cola por el componente de Competencia (productor). Por consiguiente, se crean varias instancias con el propósito de solucionar el tema de las demoras.

b.



- Un competidor trae un nuevo modelo
- "Competencias" lo suma a la cola de mensajes
- Una instancia del "Procesador" consume un mensaje de la cola
- Se suma el modelo al "Ambientes-Notebooks"
- Se buscan los tests de la competencia en cuestión desde el "DataStore"
- El "Procesador" lo envía a "Competencias"
- "Competencias" compara los resultados de los tests y realiza la evaluación del modelo

c. Que aparezca un texto que diga “Se está realizando la carga, esto puede tardar unos minutos”

2. ¿A qué propiedad del sistema ayuda que el componente procesador no tenga estado?

Ayuda a la propiedad de Disponibilidad ya que cualquier instancia del procesador puede atender la solicitudes de la cola de mensajes debido a que todas tienen la misma información, al ser stateless. Es más escalable también ya que puedo hacer todas las instancias que quiera al no guarda información.

3. Un colega suyo propone el uso de una caché en el componente Procesador para el endpoint expuesto. ¿Es una decisión adecuada? Justifique. De ejemplos

No tiene sentido guardar información en la caché ya que en este contexto no plantean todos los usuarios lo mismo entonces resultaría ser inútil.