



# Feria de empresas

[Final DDS 20230926.pdf](#)

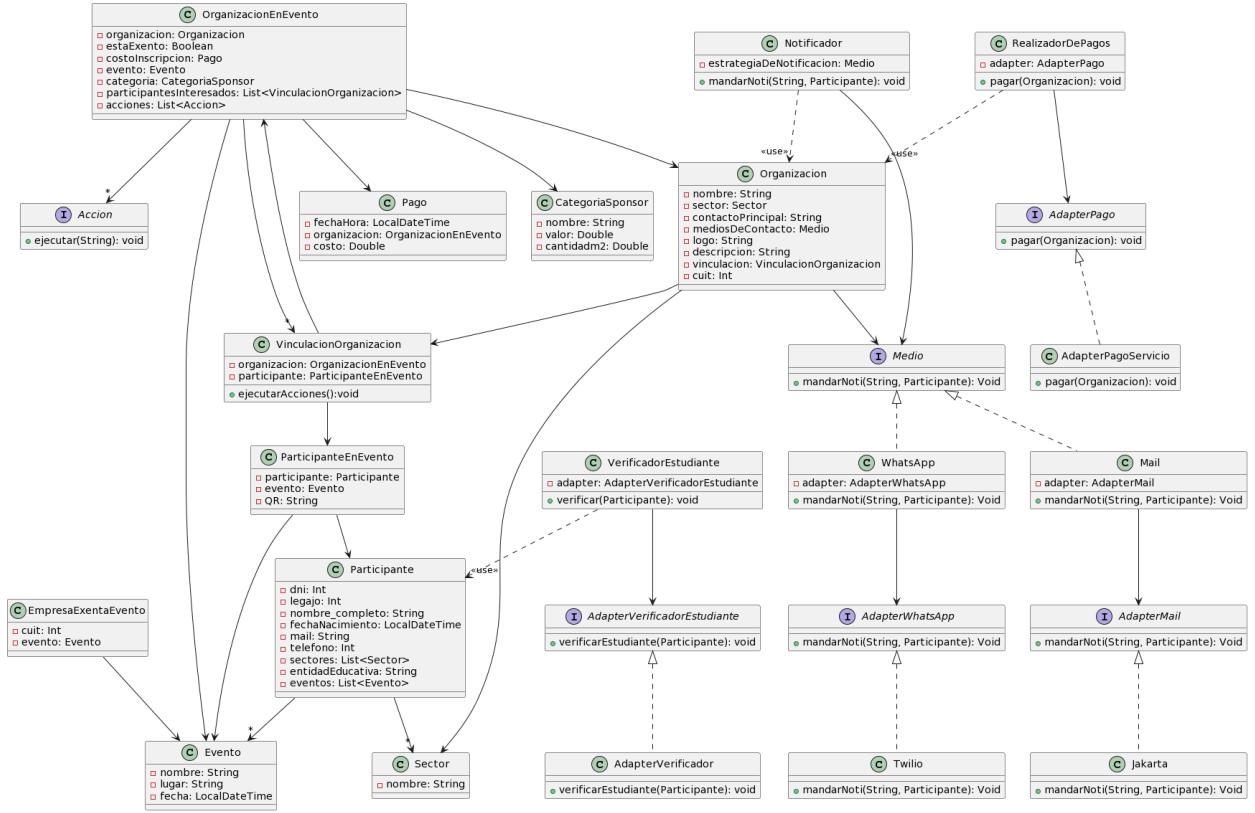
**Clase de consulta:**

[https://drive.google.com/drive/folders/1atpOTp5Ehot8d4YvVb1IOIA\\_FsgtJvEA](https://drive.google.com/drive/folders/1atpOTp5Ehot8d4YvVb1IOIA_FsgtJvEA)

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

## Modelado de objetos

## Diagrama de clases



## Decisiones de diseño

Consideramos que el presente sistema se basa en un multievento, por ello, agregamos la entidad Evento, así como también el ParticipanteEnEvento y OrganizacionEnEvento, de esta manera las organizaciones y los participantes pueden formar parte de mas de un evento, y nos permite mantener la trazabilidad de los mismos. A su vez, para representar las empresas exentas de pagar la inscripción al evento se utiliza una clase llamada EmpresaExentaEvento que puede identificar a las respectivas entidades por su cuit. Respecto a los pagos de las organizaciones consideramos que la clase OrganizacionEnEvento tenga un Pago, el cual contiene una fecha y el costo del pago realizado, dato que esta desnormalizado por consistencia de datos.

Respecto a las acciones que pueden realizar las organizaciones con sus interesados, consideramos que son una interfaz con un método ejecutar(String), donde cada clase que implemente dicha interfaz tendrá la implementación propiamente dicha del metodo ejecutar(string). Esto nos proporciona mayor mantenibilidad, cohesion y testeabilidad. Además para persistir dichas acciones debemos usar un Converter y la annotation @ElementCollection, de tal manera que el ORM lo mapee como una tabla con los nombres de las acciones y la referencia a la organizacion a la que pertenezca.

Por otra parte, se modela la clase CategriaSponsor y Sector para que se puedan instanciar y utilizar una referencia en objetos de las clases como OrganizacionEnEvento, Organización y Participante. Por consiguiente, se cumple con uno de los requerimientos que es la modificación de los valores de las categorías. Cabe aclarar que, si la organización no desea tener un sponsor en un evento, el campo puede quedar en null.

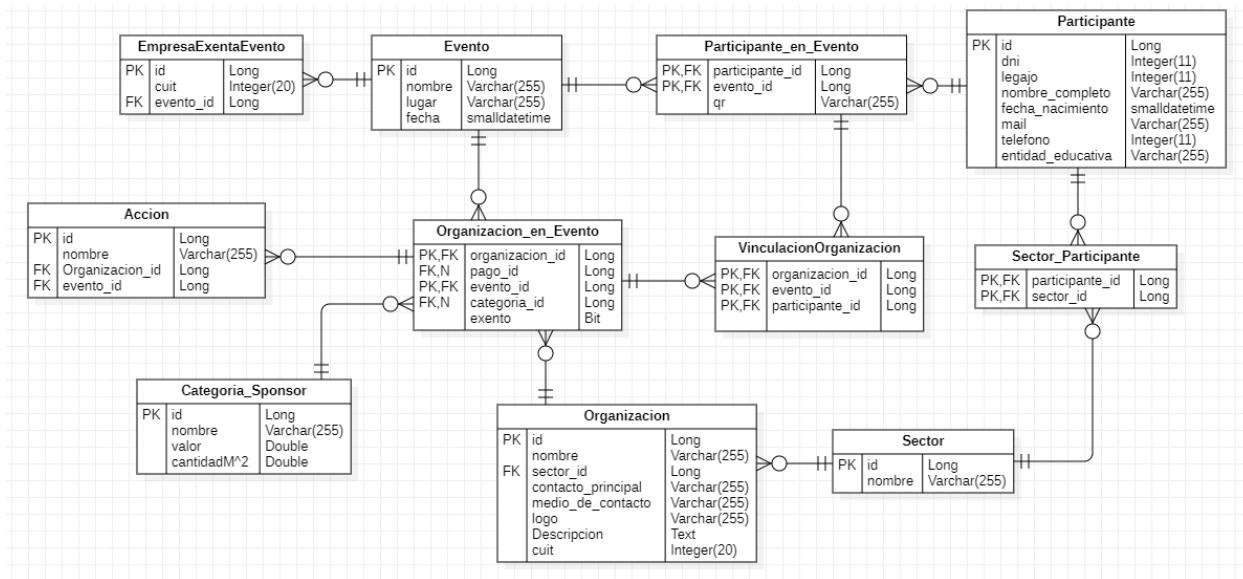
De la misma manera, para los sectores de interés del participante consideramos una clase Sector con un nombre, de tal manera que se los pueda dar de alta y baja de forma flexible.

Por lo que respecta a los medios de notificación y el envío de las mismas, se utiliza un componente que permita la ejecución de dicha actividad. Se implementa un patrón adapter así como también para la verificación de los datos del estudiante ya que ganamos cohesión en las clases adaptadas, y delegamos la responsabilidad en objetos particulares, aumentando de esta manera la mantenibilidad al no tener acoplamiento entre la clase adaptada y el cliente.

En caso de que el verificador no haya encontrado al alumno entre sus facultades, se instanciará el alumno pero dejará los campos en null y el controller le pedirá al usuario que registre los datos faltantes así como también su entidad educativa.

## Persistencia

### Modelo de datos



## Decisiones de diseño

Los medios a nivel objetos son una interfaz, debido a que estas no son persistibles. Para ello, utilizamos un AttributeConverter para poder persistir dicho medio como un nombre en la tabla, de esta manera, cuando haya que hidratar al objeto revisaremos cuál es su nombre y sabremos que medio instanciar.

El logo en la organización representa la ruta a donde esta depositado en el filesystem dicha imagen.

El atributo costoInscripción de OrganizaciónEnEvento, es a nivel datos una referencia al pago, por lo que lo persistimos como pago\_id. En caso de que la organización pudiera estar exenta de pago, este atributo podría quedar en null. A su vez, si la organización no desea ser un sponsor en un evento, el atributo categoriasponsor se coloca también en null.

Respecto a las acciones que pueden realizar las organizaciones con sus interesados, para persistirlas utilizamos un Converter y la annotation @ElementCollection, de tal manera que el ORM lo mapee como una tabla con los nombres de las acciones y la referencia a la organización a la que pertenezca.

## Entidades necesarias a persistir

- EmpresExentaEvento
- Evento

- Organizacion
- OrganizacionEnEvento
- Accion
- Categoria\_Sponsor
- Participante\_en\_Evento
- Participante
- Sector
- VinculacionOrganizacion

Las interfaces, clases stateless de los adapters, clases como Notificador o RealizadorDePagos no deben ser persistidas.

## Impedance mismatch

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

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

Para el presente ejercicio no se desnormalizó ningún dato, ya sea por performance o por consistencia

## Arquitectura

A. ¿Cómo gestionaría el cambio de Pasarela de Pago? ¿Qué atributo de calidad se estaría priorizando?

El proceso de pago se podría realizar de manera asíncrona y a su vez, cada servicio que permita la ejecución del mismo, podría tener un atributo de prioridad que determina el orden en el cual se deberían realizar los cambios. Con esta prioridad se revisará si el medio de pago (del más prioritario, al menos) se encuentra disponible y listo para recibir solicitudes, en caso afirmativo usará ese, y sino utilizará al de la próxima prioridad. Se debería hacer automático el cambio y debería poder ver el estado de cada proveedor. Cada uno debería ser una interfaz con un método polimórfico y con sus respectivos adapters. De la misma manera, el atributo de calidad que se está priorizando es el de Disponibilidad (lo menciona el mismo dominio) se debe priorizar que exista un medio de pago disponible, esto es así dado el modelo de negocio.

B. Considerando que las acciones pueden ser configuradas como asincrónicas indicando una fecha/hora para su envío, ¿Qué cambiaría en su modelo? ¿Con qué componente lo resolvería?

El modelo cambiaría por varias cuestiones, por un lado las acciones stateless se transformarían a stateful (se sumarían atributos llamados “fechaEnvio” y “horaEnvio”) y, de la misma manera, se debería sumar un componente a nivel sistema operativo, una Cron Task que revise, cada cierto tiempo, que tareas tengo que ejecutar.

C. Considerando que se quiere reutilizar un componente preexistente de generación de entradas a eventos, el cual no tiene la posibilidad de escalar, pero es posible realizar este proceso de forma asíncrona, ¿Qué estrategia de integración utilizaría y por qué? Explique cómo realizaría la integración.

Dado que el componente de generación de entradas no puede escalar, pero se lo puede utilizar de forma asíncrona, proponemos integrarnos con él a través de una cola de mensajes, la cual iremos llenando con los nuevos inscriptos, y el componente, por su lado irá consumiendo de dicha cola generando de esta manera las entradas. Esta estrategia se puede implementar dado el contexto ya que se menciona que el modelo del proceso lo permite.



Nosotros lo dejamos en la cola de mensajes y después desde el servicio se toma de la cola de mensajes. Si no permitiera el sistema integrado la forma asíncrona y solo espera un post, por ejemplo, deberíamos generar un código que agarre el evento de la cola de mensajes y ella genere el post requerido