

La universidad te lleva a Catar

[Final DDS 20220723.pdf](#)

Clase de consulta:

https://drive.google.com/drive/folders/1atpOTp5Ehot8d4YvVb1IOIA_FsgtJvEA

Hecho por: Facundo Piaggio, Chabela Lamas

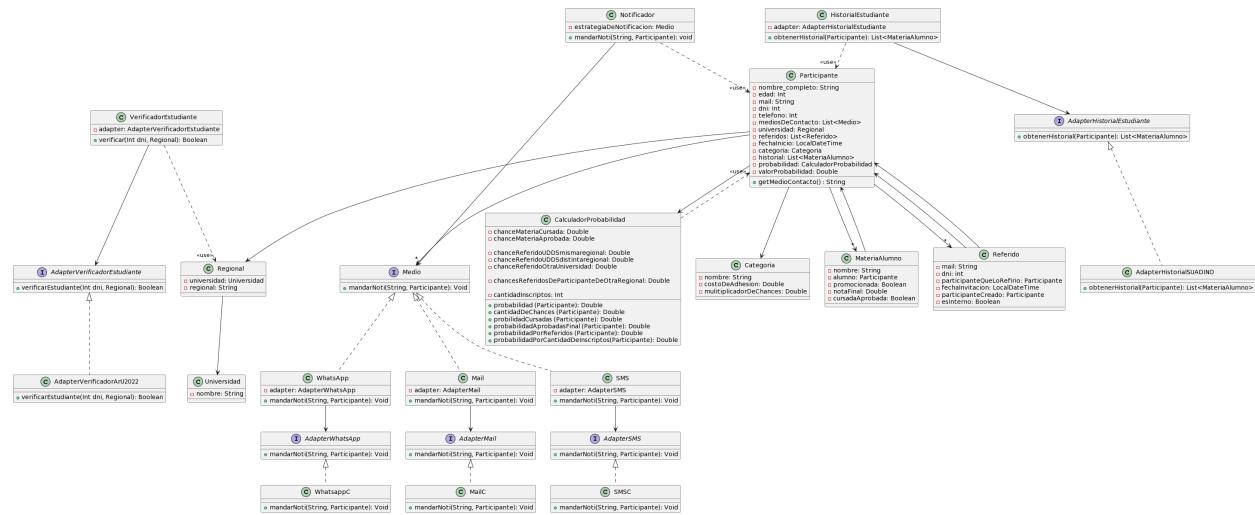
Modelado de objetos

Responder

- a. ¿Cuál es el objetivo del sistema? ¿Quiénes son los actores?
- b. ¿Cuál es el alcance desde el punto de vista a quién está dirigido?
- c. ¿Es un proyecto a largo plazo?

- a. El sistema tiene como objetivo permitir la inserción e invitación de diversos alumnos al sorteo de la facultad UDDS por el viaje a Catar y realizar el calculo de probabilidad de ganar de los diversos participantes. Los actores son: administradores del sorteo y alumnos de diversas facultades.
 - b. El alcance desde el punto de vista de a quién esta dirigido es a nivel nacional ya que es todos los alumnos de las universidades del país
 - c. Sería un proyecto a corto plazo ya que solo se va a utilizar este sistema hasta que se sortee el viaje entre los alumnos.

Diagrama de clases



Decisiones de diseño

En primer lugar, colocamos una clase Universidad para que los participantes utilicen una referencia a un objeto de la misma, es decir que se pueda instanciar diversas universidades para que puedan ser utilizadas por los estudiantes. Cabe aclarar que, en la capa de presentación, al tenerlo como una clase podemos mostrarlo en un posible desplegable y no haría falta integrarnos con una API que nos otorgue un listado de facultades. De la misma manera, se añade un boolean “esInterno” en la regional en pos de saber si es de la UDD y si se puede acceder al historial académico.

Respecto a los participantes y referidos, consideramos incluir una fechalinicio en el participante que representa la fecha en la cual ingreso al sorteo, y además una

fechalInvitacion con el participante que lo invitó, en el referido, que es cuando un participante envía una invitación a otro. En el momento en el que el referido acepta dicha invitación, se convierte en un participante (sumando la referencia al mismo en su atributo participanteCreado), dando de alta la fechalinicio. Así conseguimos mantener la trazabilidad de los participantes, sus referidos y como estos se convierten (o no) en otros participantes.

Se utiliza el Patrón Adapter para representar los medios de comunicación para el envío de notificaciones, para integrarnos con la API REST del SUADINO y del verificador ArU2022.

De esta manera, 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.

Por otra parte, en la clase CalculadorDeProbabilidad decidimos añadir como atributos los valores de las chances de las diversas situaciones mencionadas en la consigna, ya sea como la de las materias aprobadas, promocionadas, entre otras. Así como también sumamos en la clase Categoría atributos como costoDeAdhesión o multiplicadorDeChances. Por consiguiente, en ambos casos, se gana mantenibilidad y flexibilidad debido a que el producto software puede ser modificado o alterado de manera más efectiva y eficiente. Además, el método de probabilidad(Participante): Double, hará uso del resto de métodos de calculo de probabilidades según algún criterio. Tenerlo delegado de esta manera nos ayuda a que el código de la funcionalidad sea mas mantenible y mas testeable, ya que se podría probar cada método en particular.

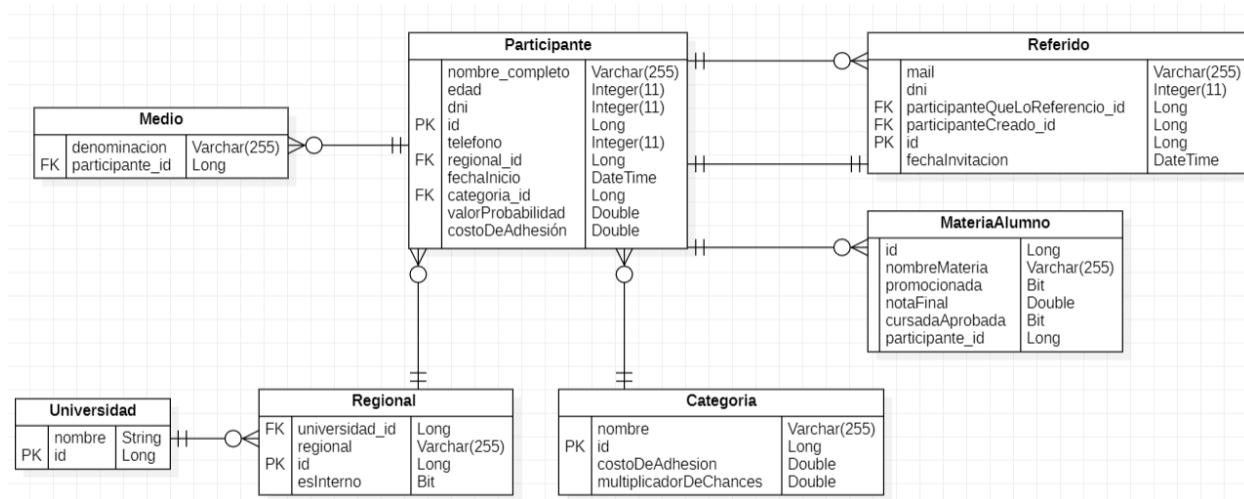
Por último, decidimos sumar una clase Materia Alumno para poder representar los valores otorgados por la API. Se convierte en la misma clase del historial el JSON que manda como repuesta el SUADINO. La clase MateriaAlumno cuenta con todos los atributos necesarios para poder utilizarlos en el CalculadorDeProbabilidad. Asimismo, en la clase Participante, añadimos un atributo valorProbabilidad para poder guardar el valor brindado por el calculador y se podría tener una CronTask, por ejemplo, que lo vaya actualizando cada cierto tiempo, teniendo desnormalizado este valor ganamos performance (eficiencia en el uso de los recursos) ya que no haría falta calcular el valor cada vez que un participante lo consulte.

Respecto al cambio de categoría de un participante, este se realizará en un método del controller, que cuando un participante, que por ejemplo haya iniciado como “Bronce” si

quiere adherirse a “Oro”, al controller le llegará el participante que solicita el cambio y la categoría, de tal forma que pueda buscar al participante y setear el atributo. De la misma manera, cada vez que invita a un referido nuevo, en esta capa de controller, se pude verificar si se debe realizar el cambio de categoría.

Persistencia

Modelo de datos



Decisiones de diseño

En primer lugar, decidimos no persistir la clase CalculadorDeProbabilidades ya que sus atributos se pueden encontrar en un archivo de Configuración. No obstante, por una cuestión de performance decidimos desnormalizar el valor de la probabilidad en el participante tal como habíamos mencionado a nivel objetos. Por el mismo motivo, desnormalizamos también el costo de adhesión, así se puede mantener el histórico del precio y no se pierde la consistencia de datos.

De la misma manera, optamos por dejar relacionado al referido con el participante que surgió luego de su aceptación para la incorporación al sorteo y con el participante que lo invitó, con el fin de especificar la relación que menciona la consigna “una persona solamente puede ser referida por un único participante” y para marcar la trazabilidad (por ello también sumamos las fechas en esta entidad como en la de participante).

Respecto a los medios (que a nivel de objetos representan comportamiento) decidimos modelarlos como una tabla en la cual se guarda la referencia al cual pertenecen y la denominación del medio en si. De esta forma, a nivel objetos, sabría que interfaz usar a la hora de usar un medio u otro. Esto se logra usando @ElementCollection, un AttributeConverter y @Convert para “mapear” dicha interfaz como un varchar.

Entidades necesarias a persistir

- Participante
- Referido
- Medio
- MateriaAlumno
- Categoría
- Regional

Impedance mismatch

Tipo de dato objetos	Tipo de dato relacional
LocalDateTime	DATETIME
String	VARCHAR(255)
Int	INTEGER(11)
Boolean	BIT

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

Modelo desnормalizado

Nuestro modelo no se encuentra normalizado. Como explicamos previamente, decidimos desnormalizar el costoDeAdhesión por consistencia de datos, para mantener el histórico del precio (entre el actual y al que se adhirieron los participantes) y además,

el valor de la probabilidad por performance, de manera tal que cada vez que un participante quiera consultar su probabilidad lo pueda hacer sin que se tenga que calcular en el sistema (se refresca con una Cron Task cada cierto tiempo).

Arquitectura

A. Considerando las variables que afectan a la calidad según la triple restricción de los proyectos (alcance, tiempo y costo) y sabiendo que es necesario que los participantes puedan interactuar con el Sistema mediante sus dispositivos móviles: ¿optaría por desarrollar una aplicación móvil (híbrida o nativa) u optaría por desarrollar una aplicación Web para que sea accesible a través de los navegadores? De ser afirmativa la segunda propuesta, ¿escogería un cliente liviano o un cliente pesado? Compare ventajas o desventajas utilizando como atributo de calidad la mantenibilidad y la performance.

Optaríamos por desarrollar una aplicación Web que sea accesible a través de los navegadores ya que por cuestiones de tiempo y costo, una aplicación móvil ya sea nativa o híbrida tomaría una mayor cantidad de tiempo (y costo) en realizarse y ya que consideramos que como clientes no sería de nuestra mayor conveniencia bajarnos una app sólo para un sorteo. A su vez, es más conveniente esta implementación web, teniendo en cuenta el ciclo de vida de nuestro aplicativo ya que no es un proyecto a largo plazo. A continuación, se encuentra la siguiente comparativa que nos permite decidir, entre otros aspectos, elegir entre un cliente pesado o liviano:

PERFORMANCE: Consumo de procesador y de memoria RAM del servidor (uso de recursos)

Cliente pesado

Cliente liviano

Tiene gran parte de lógica del lado del cliente y se comunica a través de un JSON con el server, por ende, gasta poco procesador y memoria RAM del servidor

La lógica de renderización de pantallas esta del lado del servidor. Ante las request se envían código de HTML a cada uno de los clientes, generando un mayor uso de recursos del procesador y memoria del servidor

MANTENIBILIDAD: Distribución de la carga de trabajo frente a un cambio que se deba realizar en el sistema

Cliente Pesado

Se consideran dos componentes: la vista (cliente pesado) y el backend. Si el cambio solicitado no requiere agregar/ modificar/ eliminar ningún campo de alguna entidad, solamente habrá carga de trabajo para el equipo responsable del front. Si en cambio, se requiere algún cambio de lo anteriormente mencionado, habrá carga de trabajo en ambos componentes

De esta manera concluimos que un cliente pesado nos otorga mas performance y mantenibilidad. Teniendo en cuenta que buscamos mayor performance en el servidor que en el cliente, dado que nuestra aplicación debería ser capaz de soportar un alto nivel de concurrencia teniendo en cuenta, a su vez, del alcance del proyecto y si bien la mantenibilidad no es un aspecto que perseguimos para este proyecto, gracias a orientar la web a este tipo de cliente, es una ventaja que conseguimos y no esta de más.

Cliente liviano

Se considera un componente principal (arquitectura monolítica). La carga de trabajo estará concentrada en un único equipo, considerando que existe un único equipo que se encarga del mantenimiento del aplicativo.



Eze en la clase de consulta mencionó que se tiraría más por un cliente liviano debido a la cantidad de tiempo que tenemos para realizar el proyecto y que este cliente es más sencillo de desarrollar que el pesado

B. Explique cómo imagina que está diseñado e implementado el Sistema de “ArU2022”, que mediante una llamada a su API REST permite saber si una persona es alumna de una Universidad del país. Recuerde que dicho Sistema se integra con todos los sistemas universitarios del país. Además, mencione qué patrones de integración podrían llegar aplicar en este caso.

El sistema denominado "ArU2022" presenta un punto de acceso mediante el cual, mediante la utilización del método GET, es posible acceder a información acerca de si el DNI de un individuo está asociado a su condición de estudiante en una universidad nacional. La tecnología subyacente en el desarrollo de este sistema puede variar, ya que no se dispone de información concreta sobre su stack tecnológico utilizado; únicamente se expone la interfaz mencionada.

De la misma manera, “ArU2022” podría estar implementado de dos maneras:

Por un lado podría guardarse un listado de todos los alumnos en una base de datos propia (asumiendo que cuando se integra con los sistemas universitarios del país, estos proveen dicho listado), de manera tal que cuando le consultemos por la existencia de un alumno nos responda de manera síncrona si existe o no en su base. Consideramos también que debe existir un tiempo de refresco para su listado de alumnos, ya que en un momento dado, una persona podría no formar parte de una universidad, y luego si. Para ello, se podría hacer uso de una Cron Task.

Por otro lado, podría estar implementado de manera tal que cuando lo llamemos, pregunte de manera paralela a cada una de las universidades si dicha persona forma parte de ella. De esta manera, la respuesta podría tardar mucho tiempo en realizarse, es por ello que la misma podría ser otorgada de manera asíncrona en una dirección de callback. Por consiguiente, se podría utilizar un método de WebHook, ya que se podría evitar un corte por un timeout.