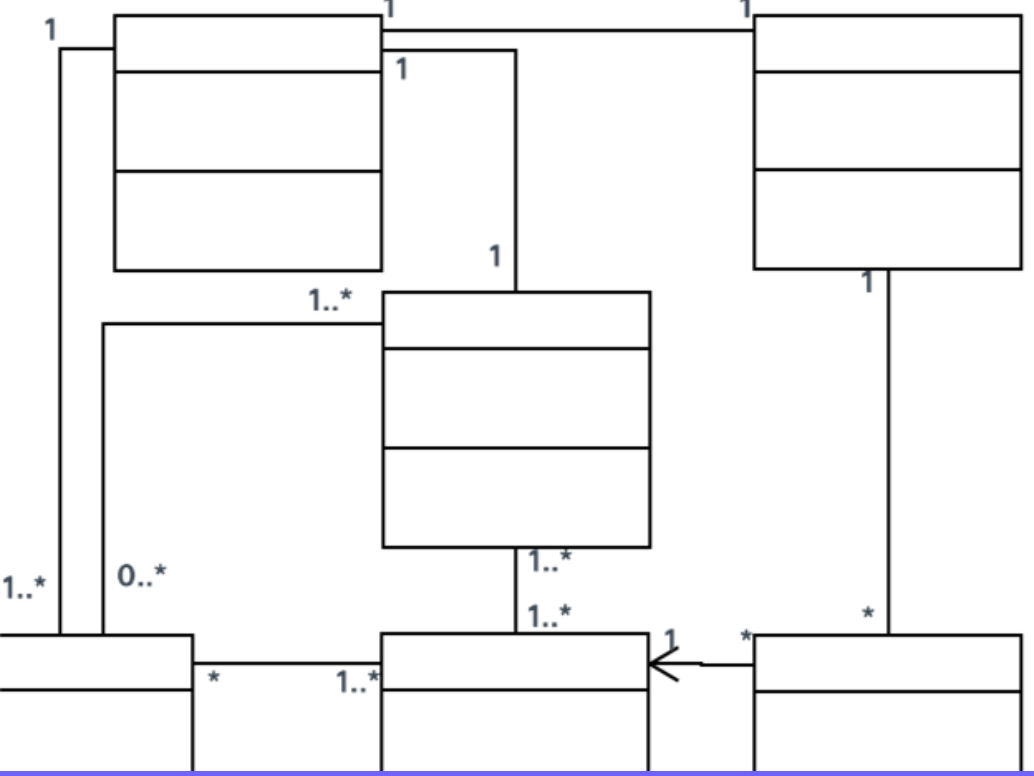
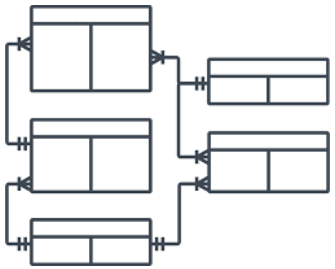




Diseño de Sistemas

Ride Me – Segundo parcial 2023

UTN.BA
DPTO. INGENIERÍA EN SISTEMAS DE INFORMACIÓN
CÁTEDRA DISEÑO DE SISTEMAS

Ride Me - Segundo parcial 2023

¡IMPORTANTE!

El presente documento representa solamente una serie de consideraciones generales del parcial, no es una propuesta de solución, ya que el mismo carece de modelos y justificaciones, fundamentales para la asignatura.

Ride Me - Segundo parcial 2023

Contexto general

En esta oportunidad, el Gobierno de la Ciudad de Buenos Aires nos ha solicitado el diseño y el desarrollo de Ride Me, una solución pensada para administrar las bicicletas públicas que están disponibles para utilizar en toda la ciudad. El Sistema permitirá, entre otras cosas, marcar el retiro de una bicicleta, marcar su devolución, pagar el costo de utilización (en caso de haber), reportar fallas.

Modelado de Datos

A continuación, listaremos las consideraciones más relevantes por cada requerimiento.

1. Que los ciclistas se registren en la plataforma

- Se debe modelar la entidad **Ciclista** con los campos mínimos necesarios indicados en el enunciado.
- Resulta de suma importancia saber la nacionalidad del ciclista, o al menos si es Argentino o Extranjero.

2. El registro de uso de bicicletas, así como también la visualización de viajes realizados por cada ciclista (no entiempo real).

- Se debe modelar la entidad **Bicicleta** y la entidad **Viaje**.
- La entidad viaje es la que debería contener relaciones (FKs) hacia: la bicicleta, el ciclista, la estación origen, la estación destino, fecha y hora de partida, fecha y hora de llegada, KMs recorridos, entre otras cosas (se mencionarán más adelante).

Modelado de Datos

3. Gestionar a los administradores

- Se debe modelar la entidad ***Administrador*** (empleado del Gobierno).
- No olvidarse e guardar la comuna principal y las comunas secundarias. Pueden estar normalizadas o desnormalizadas.

4. Gestionar las bicicletas por parte de los administradores

- Reafirma que es súper necesario el modelado de la entidad Bicicleta, por si en el requerimiento 2 no resultó lo suficientemente convincente.

5. Gestionar las estaciones por parte de los administradores

- Reafirma que es súper necesario el modelado de la entidad Estación, también mencionada en el req. 2.
- Resulta de suma importancia no olvidar de guardar las bicicletas que están estacionadas en el momento.

Modelado de Datos

6. Asegurar la trazabilidad y auditoría sobre la utilización de las bicicletas

- Reafirma la importancia de las entidades mencionadas en el req. 2.
- Si me olvido de colocar estación origen y destino => Mal.
- Si me olvido de hacer referencia al ciclista => Mal.
- Si me olvido de hacer referencia a la bicicleta => Mal.
- Si me olvido de poner fechas/horas => Mal.

7. Que los administradores gestionen los valores de los costos de utilización

- Se debe modelar una entidad "**Precio servicio**" (o nombre similar), cuyas columnas podrían ser un ID, un campo "detalle" y un campo "valor". Ejemplo de los dos registros básicos: "costo base" y "costo adicional por KM recorrido".
- Además, se le podría agregar un campo "fecha actualización" o similar, para saber cuándo se actualizaron los registros.
- Si modelamos los precios en archivos de configuración => Mal.

Modelado de Datos

8. Que tanto los administradores como los ciclistas reporten fallas sobre las bicicletas

- Se debe modelar la entidad **Falla** (o nombre similar).
- La falla debe poder contener varias fotos. Recordar que, al poder ser varias, se transformará en una relación One To Many entre Falla y FallaFoto.
- Además, se deberá modelar una entidad "**MotivoFalla**", cuyos campos serán "id" y "nombre/descripción".
- La falla debe contener referencia a un motivoFalla, nullable, pues si la persona no selecciona ninguno (o mejor dicho, selecciona "otros"), deberá escribir a mano el motivo.
- A raíz de lo anterior, es necesario un campo de tipo "text" "otraRazon" sobre la entidad Falla.

Modelado de Datos

9. Gestionar la facturación y cobro por utilización de las bicicletas, a ciclistas extranjeros.

- Reafirma que resulta de suma importancia saber la nacionalidad del ciclista (o al menos saber si es argentino o no).
- Además, para poder realizar el cobro mensual es necesario guardar el costo de cada viaje (el base + el extra). Resulta de suma importancia que estos valores estén desnormalizados, pues si cambian los costos en un momento posterior al viaje, no debería alterarse la facturación realizada.
- Es necesario modelar la entidad "**Factura**" (o "Cobro", o similar), mínimamente con fecha/hora de confección, ciclista al que se le está cobrando, y relación a todos los viajes involucrados.

Modelado de Datos

A continuación, listaremos las consideraciones más relevantes que se tuvieron que haber tenido en cuenta en base a los otros requerimientos especificados.

"Será necesario mostrar "la cantidad de viajes realizados", en el listado de "ciclistas" y en el panel de bienvenida de un ciclista"

- Es necesario desnormalizar la cantidad de viajes realizados por cada ciclista, ya que es un dato altamente consultado.

"Será necesario mostrar el promedio de kms recorridos por cada ciclista, en el listado de "ciclistas" y en el panel de bienvenida de un ciclista"

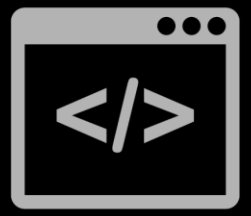
- Es necesario desnormalizar el promedio de kms recorridos por cada ciclista, ya que es un dato altamente consultado.

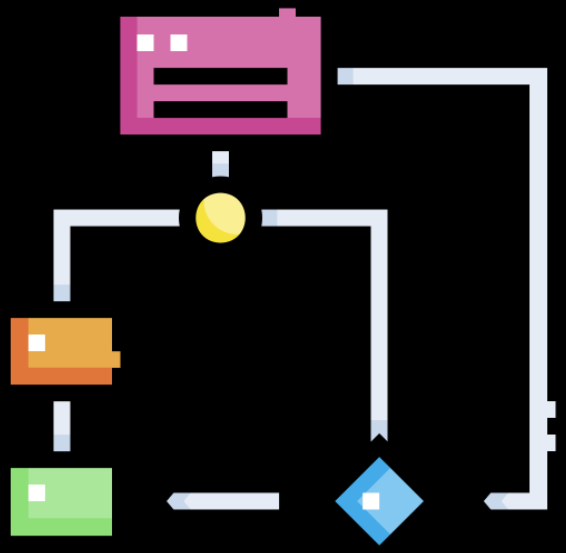
Modelado de Datos

"Será necesario mostrar la cantidad de fallas reportadas por cada una de las bicicletas, en el listado de bicicletas por estación."

- Es necesario desnormalizar la cantidad de fallas reportadas por bicicletas, ya que es un dato altamente consultado.

Arquitectura





Arquitectura

A continuación, listamos las consideraciones más relevantes a alto nivel por cada punto.

1. "Entendiendo que nuestro Sistema debe poseer una capa de presentación gráfica, ¿qué tipo de interfaz propondría implementar para los ciclistas? (…) se poseen 4 meses en total (…)"

- Se debería descartar la opción de app mobile nativa pues es la que más tiempo llevaría desarrollar, pues habría que hacer mínimamente 2 desarrollos (uno para cada SO más relevante en el mercado)
- De las restantes, cualquiera podría ser válida (importa muchísimo la justificación).

Arquitectura

2. "¿Mantendría la elección de la capa de presentación visual del punto anterior para todas las tareas que tienen que realizar los administradores (empleados del Gobierno de la Ciudad)? (...)"

- Las tareas a realizar por los empleados del Gobierno son más de Backoffice, por lo que lo ideal sería la utilización del Sistema a través de una PC.

Arquitectura

3. "Teniendo en cuenta que es necesario que no se demore más de 20 milisegundos en el proceso de reconocimiento facial de un ciclista, y que los procesos de fichadas de retiro y devolución no deben involucrar más de 45 segundos en total; mencione de qué forma abordaría estos requerimientos, justificando adecuadamente y detallando el mecanismo completo, además de considerar que las estaciones podrían sufrir cortes de conexión a internet en ciertos casos".

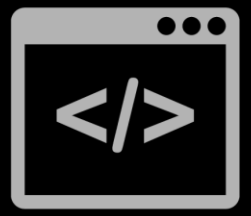
- De forma muy resumida: realizar el proceso de reconocimiento facial de forma local en cada tótem, de forma sincrónica (no debería otorgar a una bici a alguien que no esté seguro si está registrado o no en la plataforma). Para ello, es necesario sincronizar cada tótem con un servidor central cada cierto tiempo.

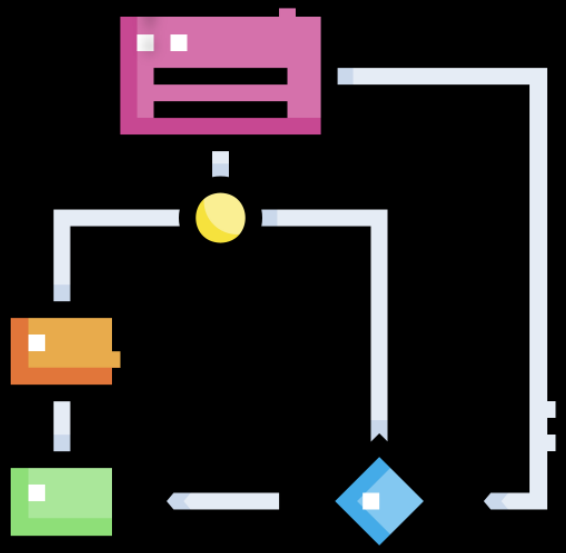
Arquitectura

4. "Para poder realizar el cobro por uso del Servicio, a los ciclistas, debemos integrarnos con RiuPagos. Este Sistema ofrece la posibilidad de integración sincrónica, mediante llamada a una API REST, o asincrónica mediante el mecanismo de WebHook. Considerando que los cobros hay que efectuarlos de forma mensual, ¿qué mecanismo escogería? (…)"

- Lo cierto es que podría ser cualquiera de los dos, ya que una vez al mes deberíamos ejecutar un proceso (Cron) que "barra" todos las facturas no cobradas y realizar el cobro a través de este Sistema. Lo ideal podría ser que cuando ejecuta esta tarea, la misma no se quede "colgada" esperando la finalización de los cobros de forma secuencial sincrónica, por lo que podríamos ir por un WebHook.

Persistencia



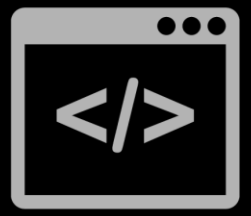


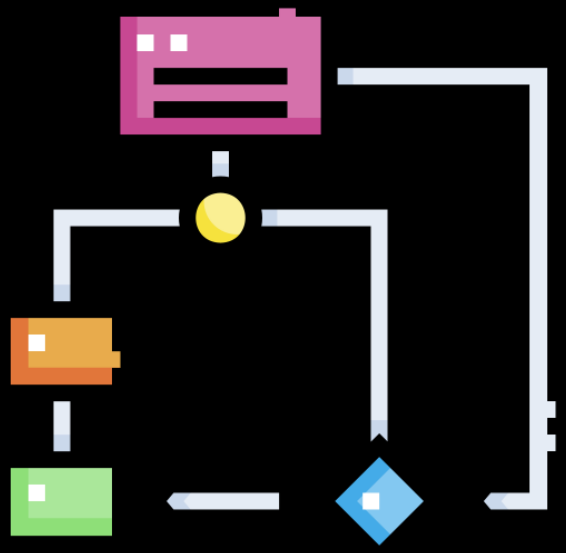
Persistencia

A continuación, dejamos las consideraciones más relevantes a tener en cuenta para el dominio presentado:

- La colección de Contactos del Deportista debe ser mapeada como un ElementCollection.
- Por lo anterior, a nivel diagrama se debe graficar una nueva tabla “deportista_contacto” (por ejemplo), que contenga únicamente dos campos: “detalle” (donde se guarda el valor del String) y “deportista_id” (FK a deportista).
- La motivación NO DEBE SER PERSISTIDA mediante una estrategia de mapeo de herencia, ya que son clases stateless únicamente de comportamiento.
- Por lo anterior, se debería haber guardado una referencia a la motivación escogida por el deportista mediante un CONVERTER.
- Respecto a la relación recursiva del ejercicio, podría ser considerada una relación OneToMany o bien ManyToMany. Dependiendo lo considerado, no solamente cambia las anotaciones del mapeo, sino también a nivel gráfico. Si es ManyToMany se necesita tabla “intermedia”, con dos FKs a ejercicio. Si es OneToMany, se necesita una columna “ejercicio_id” en la tabla ejercicio, FK a la misma entidad.

Teoría





UNIFIED MODELING LANGUAGE™

Teoría

1. Eager Loading es mejor que Lazy loading como mecanismo de Hidratación => ***FALSO***. Ninguna estrategia es mejor que la otra.
2. Un token es generado por el cliente y almacenado en el servidor => ***FALSO***. No se genera en el cliente ni se almacena en el server.
3. GraphQL es un mecanismo de integración sincrónico => ***FALSO***. Puede ser sincrónico o asincrónico.
4. Un servidor de bases de datos relacional es un ejemplo de componente para mí arquitectura => ***FALSO***. Es un ejemplo de Nodo.
5. El principal componente de una arquitectura del tipo SOA es el api Gateway => ***FALSO***. Para SOA es ESB, para Microservicios es API Gateway.

Dos motivos para normalizar:

- Eliminar redundancias.
- Consistencia de datos.

Un protocolo para la utilización de cola de mensajes

- AMQP

Gracias