TPA Quinta Entrega
Equipo 4

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

Alcance

La entrega comprende:

- Se deberán persistir las entidades del modelo planteado. Para ello se debe utilizar un ORM.
- Se deberá implementar el servicio que tuviera asignado el grupo (Servicio 1: fusión de comunidades)

API – Fusión de Comunidades

Diseño del Módulo de Sugerencias y Gestiones de Fusiones; evaluación de criterios de similitud entre comunidades; solicitudes de API.

Herramientas

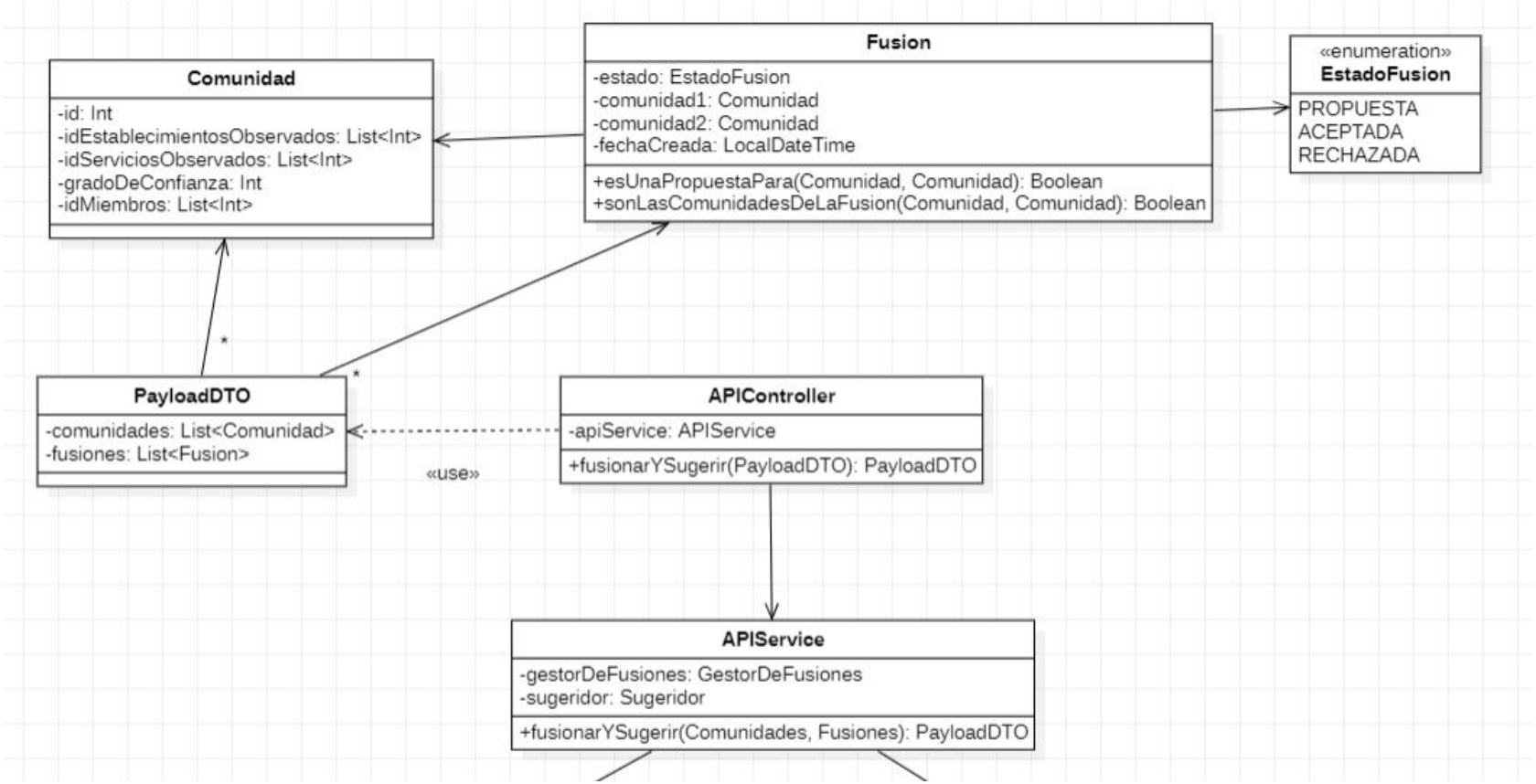
Creador de aplicación: SpringBoot -> Java

Documentación: Swagger

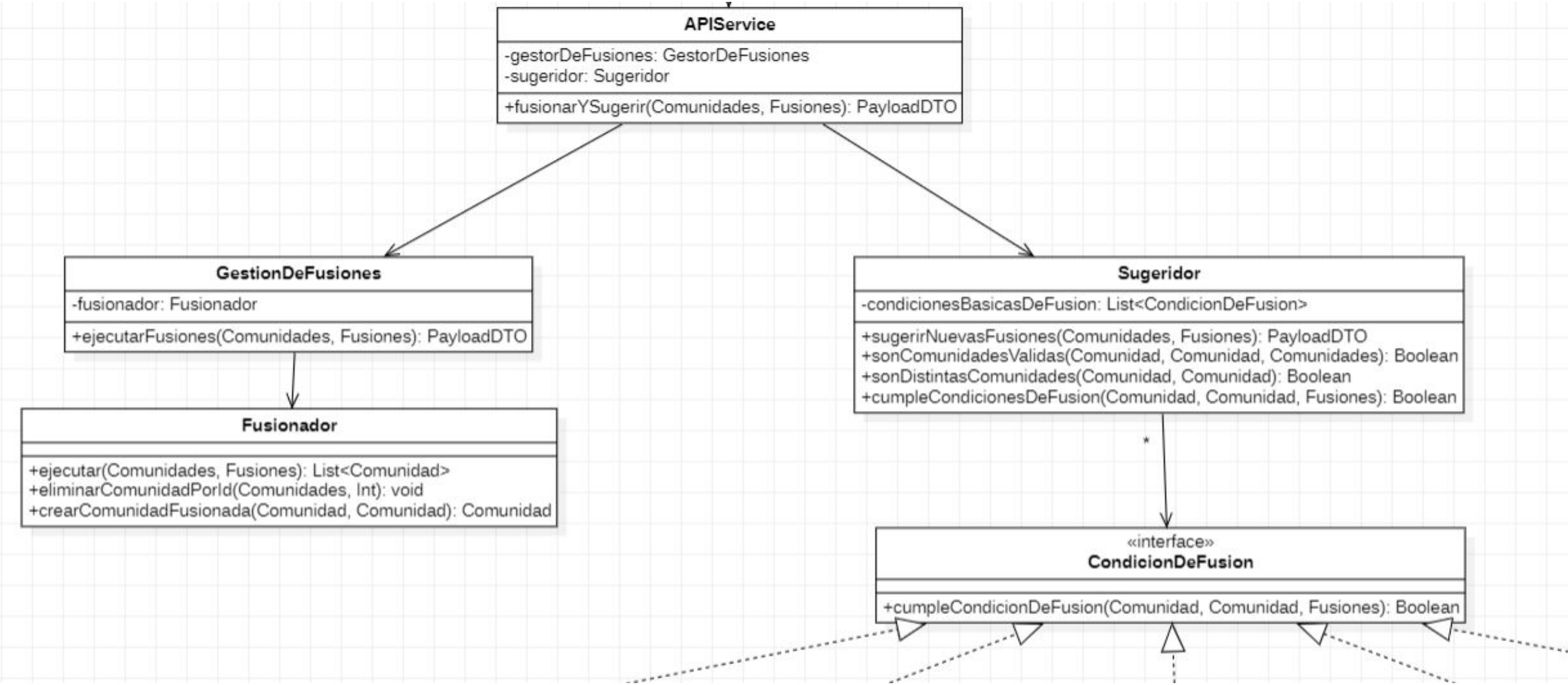
Plataforma API: Postman

API Controller – Diseño general

APIController recibe y devuelve las request y llama al servicio de la APIService la cual separa la lógica de las fusiones y las sugerencias. Se decidió que sea de esta manera para mejorar la testeabilidad y reducir el acoplamiento. De esta manera, aumenta la cohesión, siguiente uno de los principios SOLID llamado “Single Responsibility”.

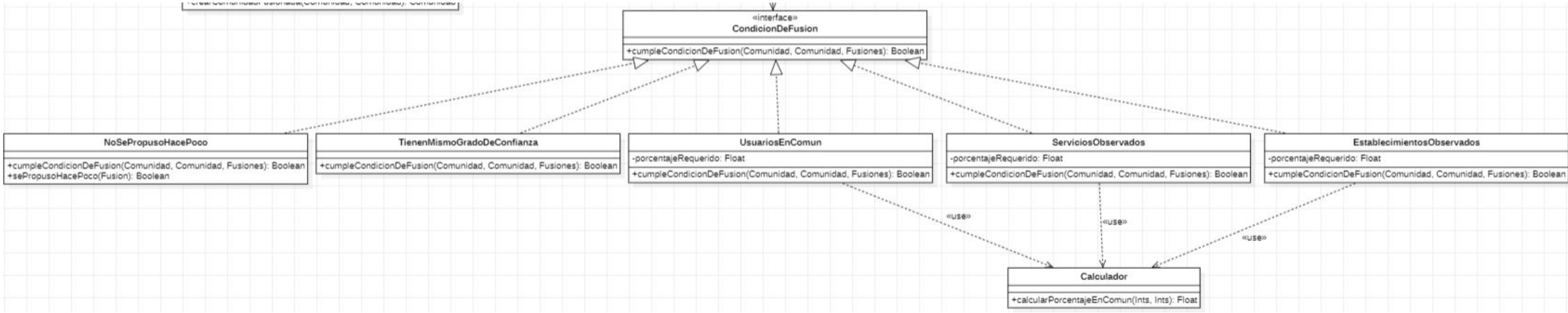


API Service – Diseño general



Condiciones de Fusión – Diseño general

Podemos identificar la aplicación del Patrón Command en el método "cumpleCondicionDeFusion". En este contexto, descompusimos las diversas validaciones en clases que implementan la interfaz "CondiciónDeFusión". La razón detrás de esta elección radica en la necesidad de configurar dinámicamente una serie de acciones que debe llevar a cabo un objeto de la clase "Sugeridor" en el momento de la ejecución. El Patrón Command nos proporciona la flexibilidad necesaria para lograr este objetivo de manera eficiente y organizada.



Links importantes

Repositorio:
https://github.com/Chabelamas/API_Fusion_Servicio

Documentación:
https://app.swaggerhub.com/apis/LSANGRONI/API_Fusion_Comunidades/1.0.0

Persistencia

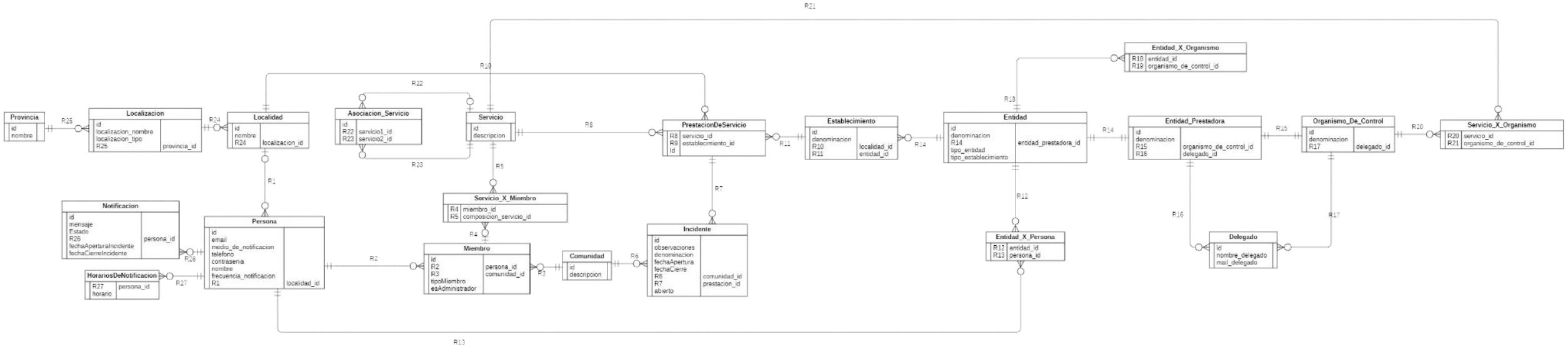
Persistencia de entidades del modelo planteado, utilizando un ORM.

Herramientas

Gestor para la base de datos

Framework de mapeo Objeto-relacional:

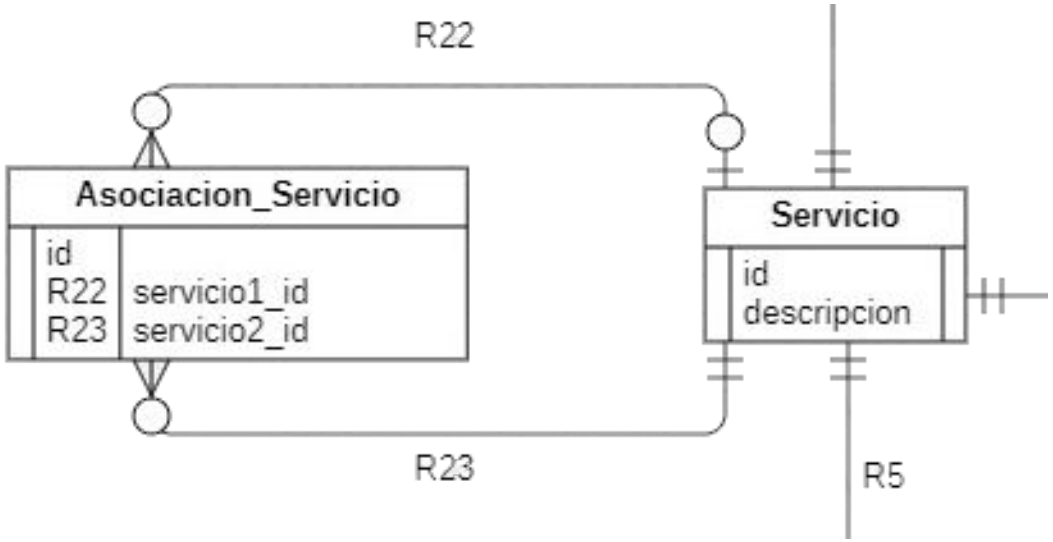
DER – Diseño general



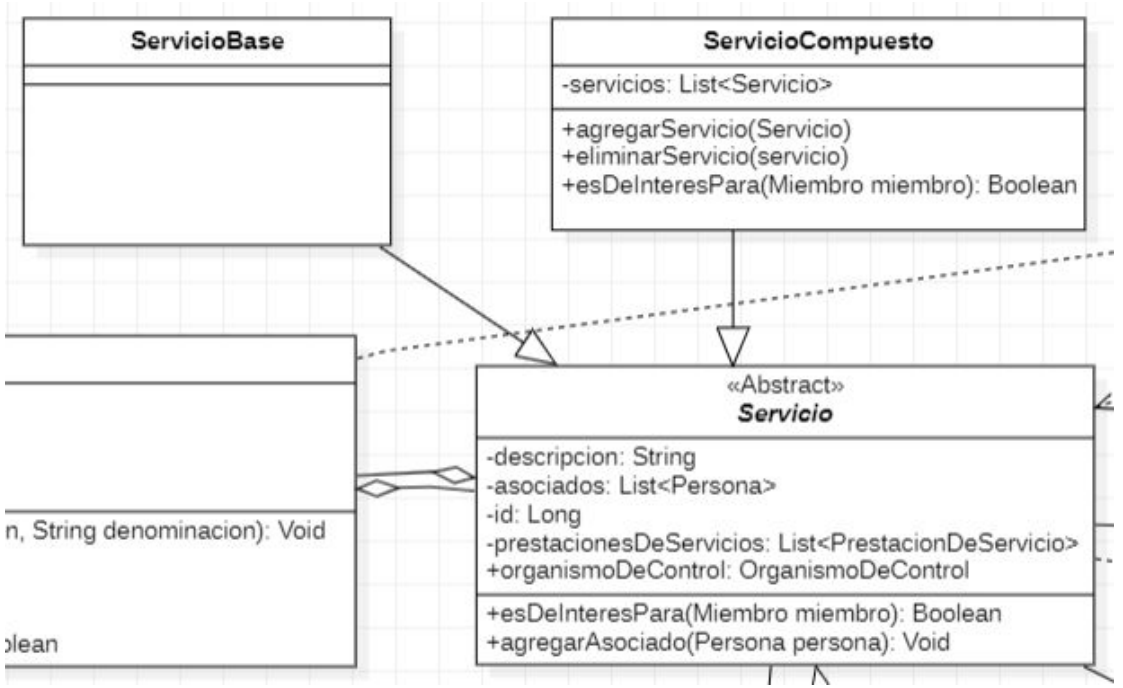
Decisiones de diseño

Modelado de los servicios

DER

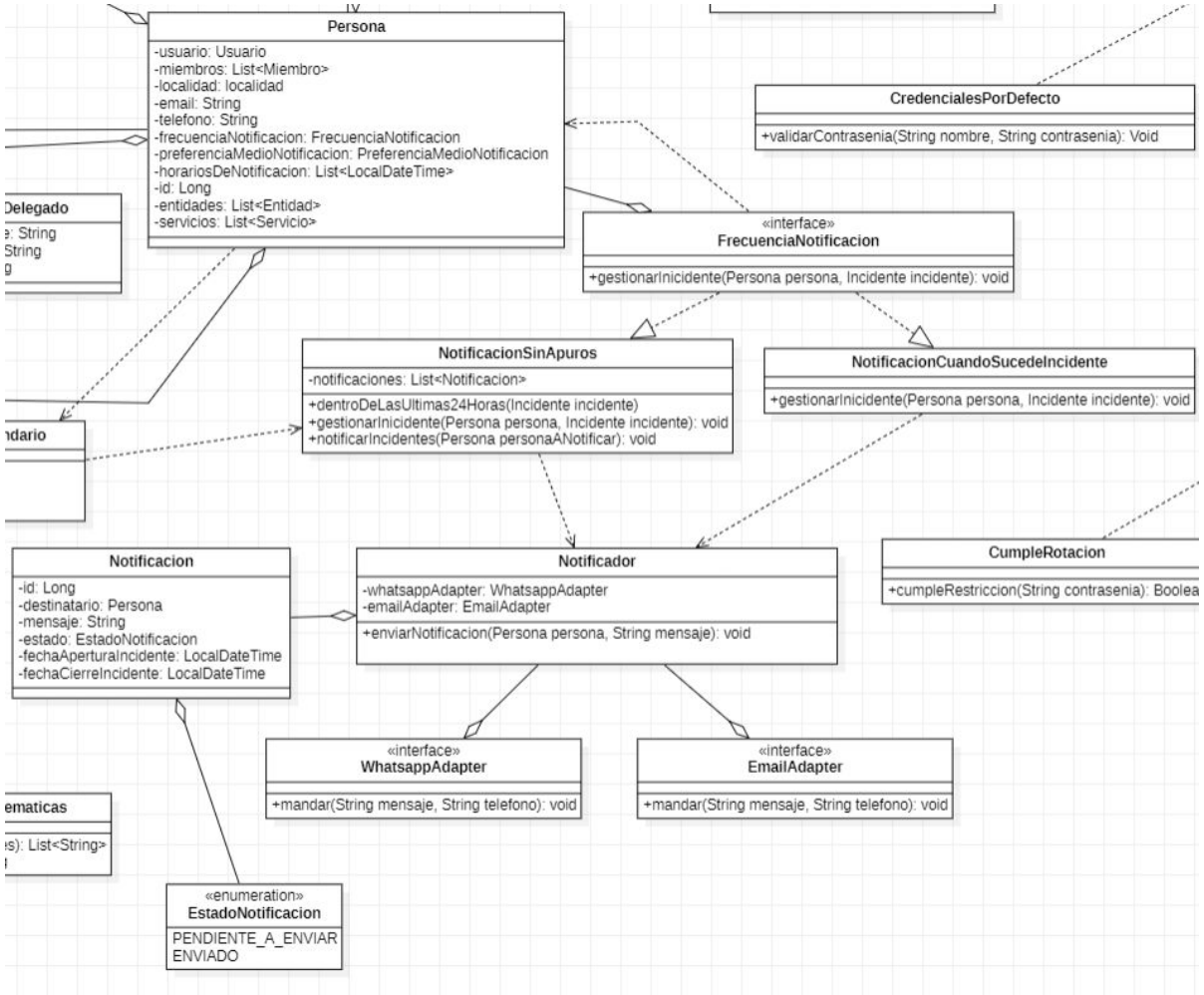


DC



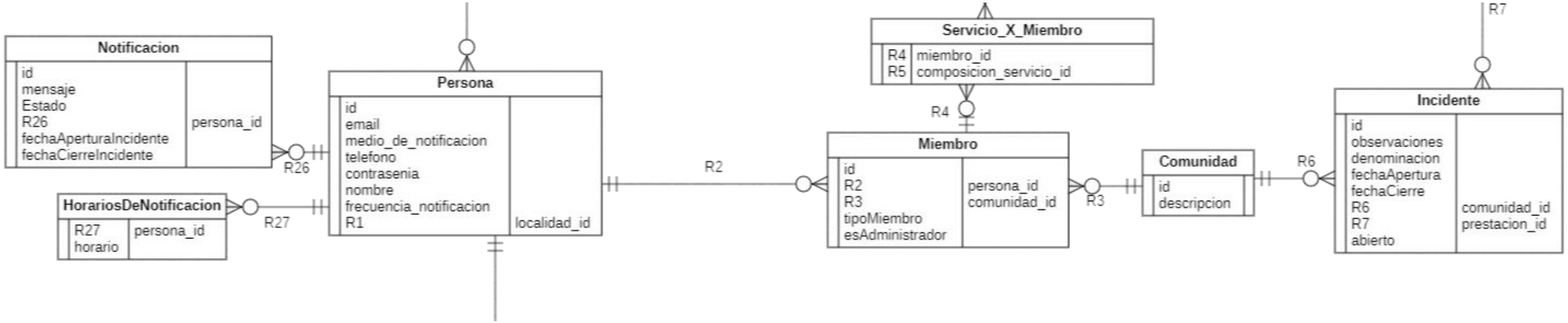
Decisiones de diseño

Modelado de las frecuencias de notificación



Decisiones de diseño

Modelado de las frecuencias de notificación



Converter para Frecuencia de Notificación

Decisiones de diseño

```java
@Converter(autoApply = true)
public class FrecuenciaDeNotificacionAttributeConverter implements
    AttributeConverter<FrecuenciaNotificacion, String>
{
  no usages  irinaperezg
  @Override
  public String convertToDatabaseColumn(FrecuenciaNotificacion frecuenciaNotificacion) {
    String frecuencia = "";
    switch(frecuenciaNotificacion.getClass().getName()) {
      case "NotificacionCuandoSucedeIncidente": frecuencia = "Cuando sucede"; break;
      case "NotificacionSinApuros": frecuencia = "Sin apuros"; break;
    }
    return frecuencia;
  }

  no usages  irinaperezg
  @Override
  public FrecuenciaNotificacion convertToEntityAttribute(String frecuencia) {
    FrecuenciaNotificacion frecuenciaNotificacion = null;
    if(Objects.equals(frecuencia, b: "Cuando sucede"))
      frecuenciaNotificacion = new NotificacionCuandoSucedeIncidente();
    if(Objects.equals(frecuencia, b: "Sin apuros"))
      frecuenciaNotificacion = new NotificacionSinApuros();
    return frecuenciaNotificacion;
  }
}
```

Modelado de las fechas → Converter

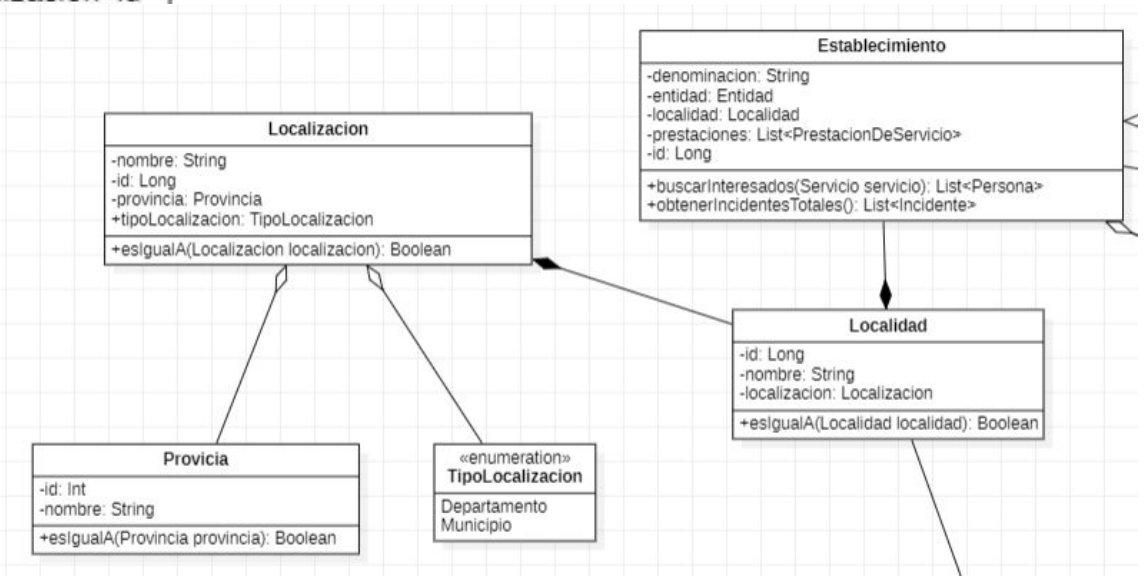

```java
@Converter(autoApply = true)
public class LocalDateAttributeConverter implements AttributeConverter<LocalDate, Date> {

    no usages  irinaperezg
    @Override
    public Date convertToDatabaseColumn(LocalDate localDate) {
        return localDate == null? null: Date.valueOf(localDate);
    }

    no usages  irinaperezg
    @Override
    public LocalDate convertToEntityAttribute(Date date) {
        return date == null? null : date.toLocalDate();
    }
}
```

Modelado de las localizaciones

Decisiones de diseño

DER

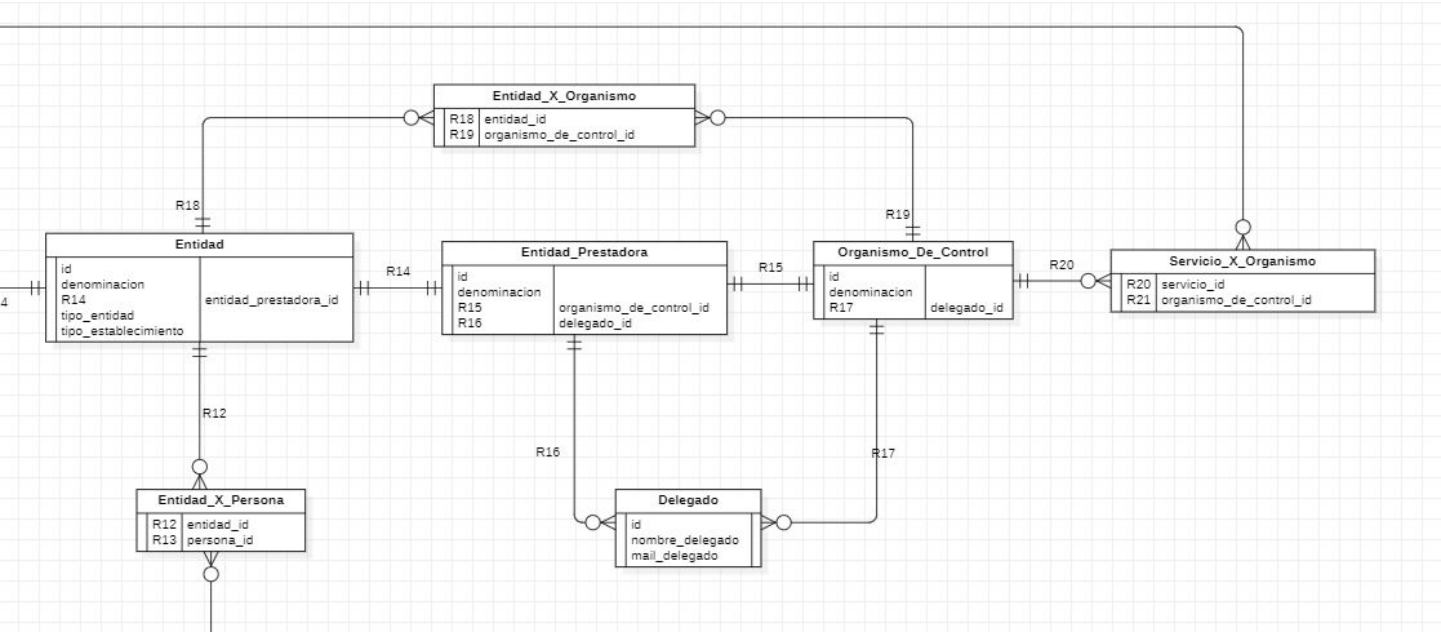
DC



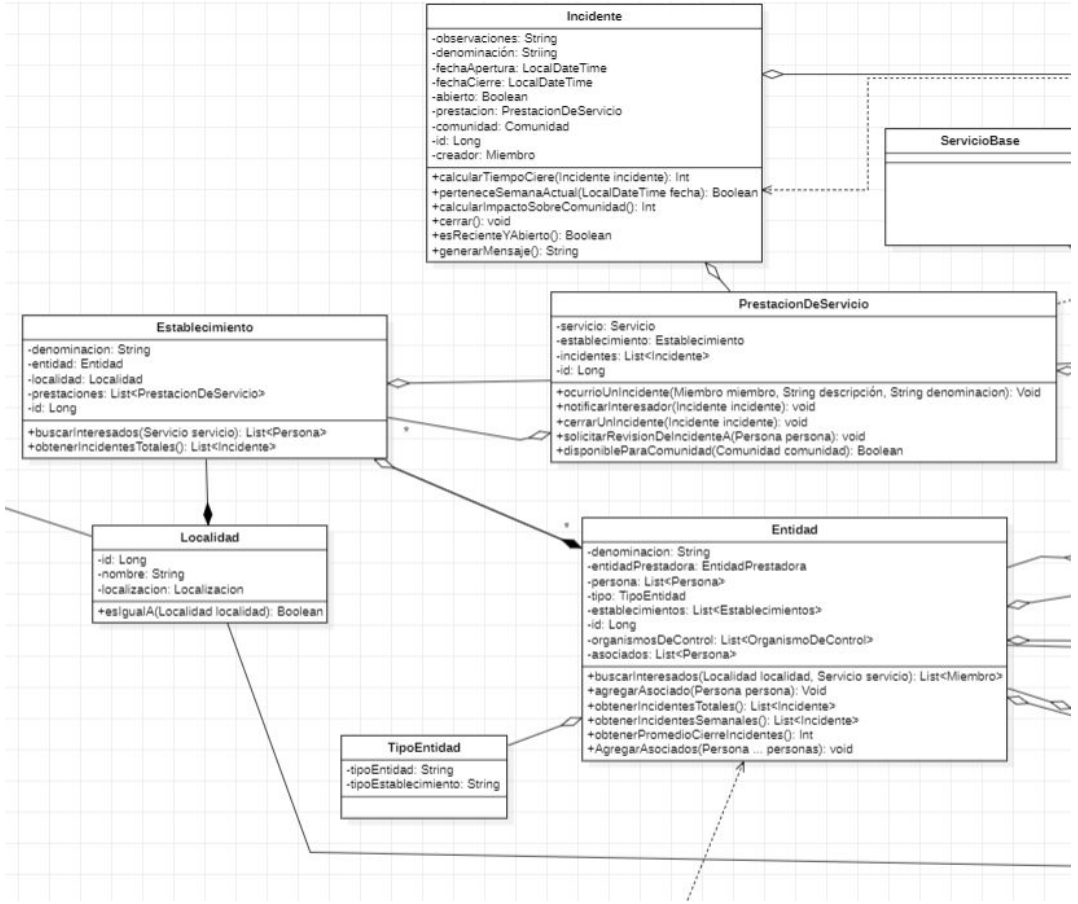
Modelado de las entidades

Decisiones de diseño

DER



DC



Decisiones de diseño

Entidades no persistidas

Aquellas que se encargan de:

- La carga masiva de datos a partir de un archivo CSV
- La generación de los ranking de incidentes
- La generación de informes en formato PDF
- Notificar a las personas (Notificador y los medios de notificación)
- Validaciones de contraseña
- Interactuar con el servicio Georef

Distribución de trabajo en el equipo

Nos dividimos en dos grupos con tareas claramente definidas. Cada grupo se enfocó en cumplir uno de los dos requerimientos esenciales. El primer grupo se encargó de la persistencia de las entidades del modelo que habíamos planteado. Para lograr esto, se empleó un Object-Relational Mapping (ORM), asegurando así una gestión eficiente de la base de datos y garantizando la integridad de los datos almacenados. Mientras tanto, el segundo grupo asumió la responsabilidad de implementar el servicio asignado a nuestro proyecto. Esta parte crítica del desarrollo se abordó asegurando que el servicio cumpliera con los requisitos y funcionalidades especificados. Esta división de tareas permitió una colaboración eficaz y un progreso fluido en nuestro trabajo.

Fin y Gracias

¿Preguntas?

Equipo 4

Integrantes:

- Carriquiry Castro, Joaquin
- Lamas, Chabela María
- Montenegro Aguilar, Gabriel Montenegro
- Pérez Gribnicow, Irina Maia
- Sangroni, Luciano Gabriel