Cuarta Entrega TPA

Grupo 19 • 13/09/2023

Requerimientos

- **Desarrollo de API**: servicio de fusión.
- **Diagrama de clases**: actualización.
- **Modelo de datos**: DER.

Desarrollo de API

API: Propuestas de Fusión

Entrada:

```
[
  {
    "idOrganizacion": 0,
    "establecimientos": [
      0
    ],
    "servicios": [
      0
    ],
    "gradoConfianza": 0,
    "miembros": [
      0
    ],
    "ultIntentosFusion": [
      {
        "idOrganizacion": 0,
        "fechaIntento": "2023-09-11T22:48:24.165Z"
      }
    ]
  }
]
```

API: Propuestas de Fusión

Respuesta (200):


```
[
  {
    "idOrganizacion1": 0,
    "idOrganizacion2": 0
  }
]
```


Respuestas error:


```
{
  "tipoError": "string",
  "mensaje": "string"
}
```

API: Aceptar Fusión

Entrada:

```
{
  "organizacion1": {
    "idOrganizacion": 0,
    "establecimientos": [
      0
    ],
    "servicios": [
      0
    ],
    "gradoConfianza": 0,
    "miembros": [
      0
    ],
    "ultIntentosFusion": [
      {
        "idOrganizacion": 0,
        "fechaIntento": "2023-09-11T22:52:59.157Z"
      }
    ]
  },
```

```
  "organizacion2": {
    "idOrganizacion": 0,
    "establecimientos": [
      0
    ],
    "servicios": [
      0
    ],
    "gradoConfianza": 0,
    "miembros": [
      0
    ],
    "ultIntentosFusion": [
      {
        "idOrganizacion": 0,
        "fechaIntento": "2023-09-11T22:52:59.157Z"
      }
    ]
  }
}
```

API: Aceptar Fusión

Respuesta (200):

```
{
  "idOrganizacion1": 0,
  "idOrganizacion2": 0,
  "organizacionFusionada": {
    "idOrganizacion": 0,
    "establecimientos": [
      0
    ],
    "servicios": [
      0
    ],
    "gradoConfianza": 0,
    "miembros": [
      0
    ],
    "ultIntentosFusion": [
      {
        "idOrganizacion": 0,
        "fechaIntento": "2023-09-11T22:52:59.158Z"
      }
    ]
  }
}
```

API: Rechazar Fusión

Entrada:

```
{
  "organizacion1": {
    "idOrganizacion": 0,
    "establecimientos": [
      0
    ],
    "servicios": [
      0
    ],
    "gradoConfianza": 0,
    "miembros": [
      0
    ],
    "ultIntentosFusion": [
      {
        "idOrganizacion": 0,
        "fechaIntento": "2023-09-11T22:52:59.157Z"
      }
    ]
  },
```

```
  "organizacion2": {
    "idOrganizacion": 0,
    "establecimientos": [
      0
    ],
    "servicios": [
      0
    ],
    "gradoConfianza": 0,
    "miembros": [
      0
    ],
    "ultIntentosFusion": [
      {
        "idOrganizacion": 0,
        "fechaIntento": "2023-09-11T22:52:59.157Z"
      }
    ]
  }
}
```

API: Rechazar Fusión

Respuesta (200):

```
Fusión rechazada correctamente.
```

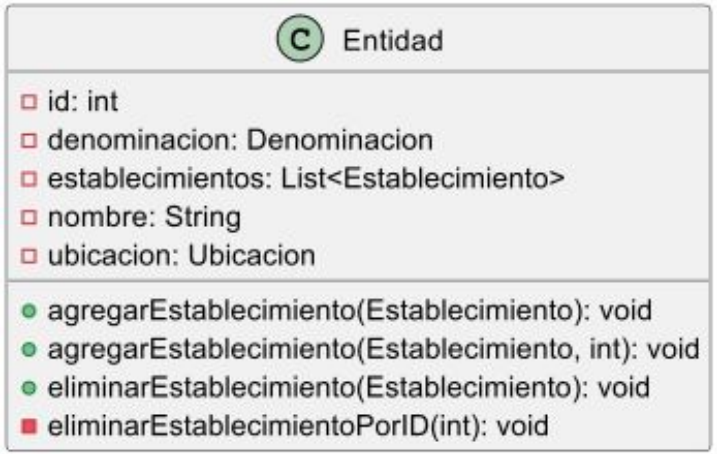
API: Fusión de Comunidades

Acceso a documentación

Diagrama de Clases

Diagrama de Clases

Agregado de IDs en entidades persistentes



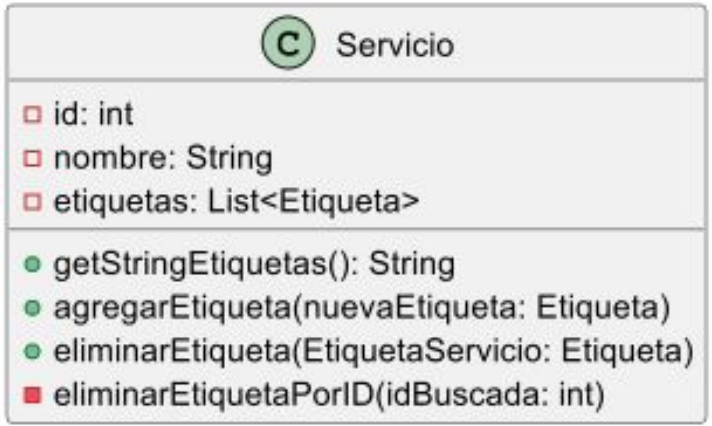
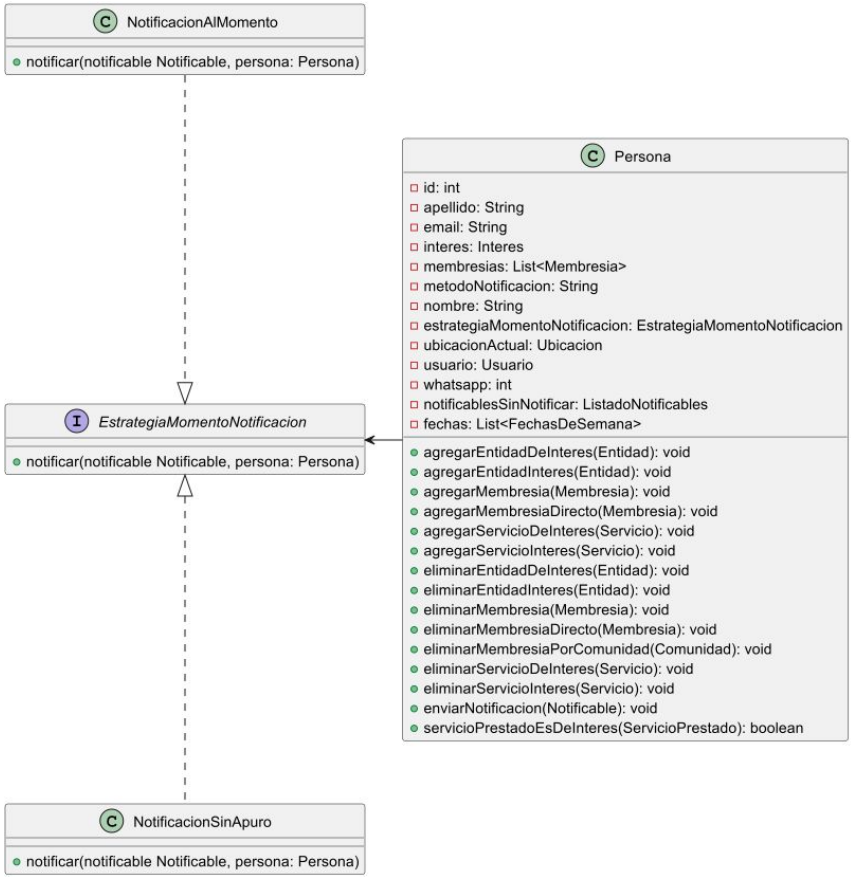


Diagrama de Clases

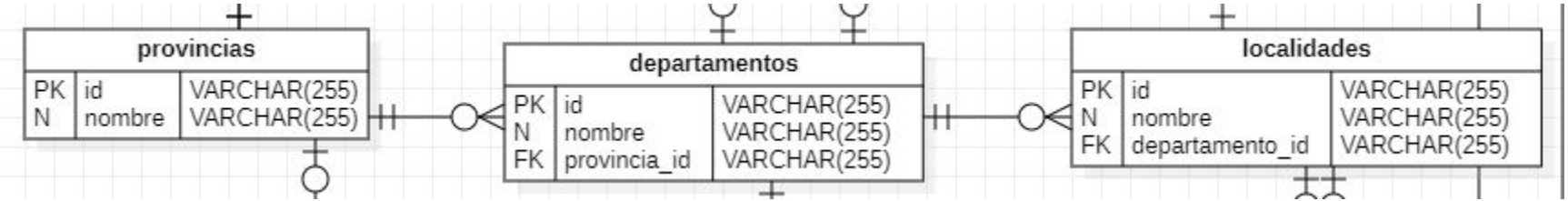
Modificación para EstrategiaMomentoNotificacion stateless



Modelo de Datos

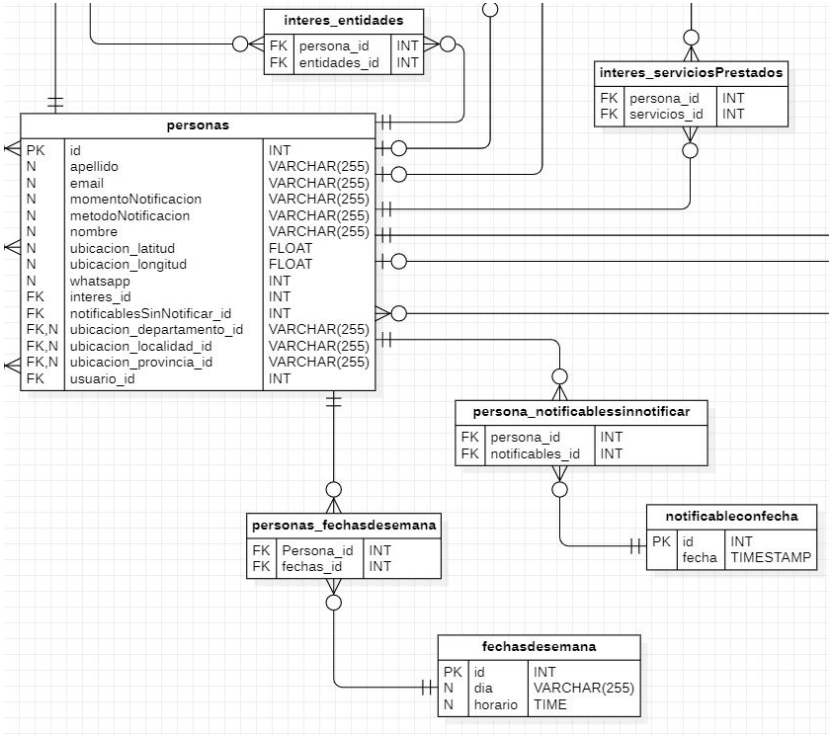
DER - Decisiones importantes

Provincia - Departamento - Localidad



DER - Interés y ListadoNotificables

```
class Persona{
  @Embedded
  private ListadoNotificables  notificablesSinNotificar ;

  @Embedded
  private Interes interes;
...
}

class Interes{
  @ManyToMany
  @JoinTable(name = "interes entidades" )
  private List<Entidad> entidades = new ArrayList<>();

  @ManyToMany
  @JoinTable(name = "interes serviciosPrestados" )
  private List<ServicioPrestado>  servicios = new ArrayList<>();
...
}

class ListadoNotificables{
  @ManyToMany
  @JoinTable(name = "persona notificablesSinNotificar" )
  private List<NotificableConFecha>  notificables;
...
}
```



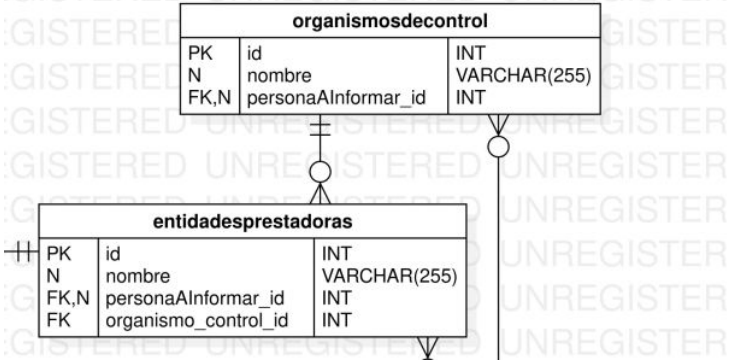
DER - Persona

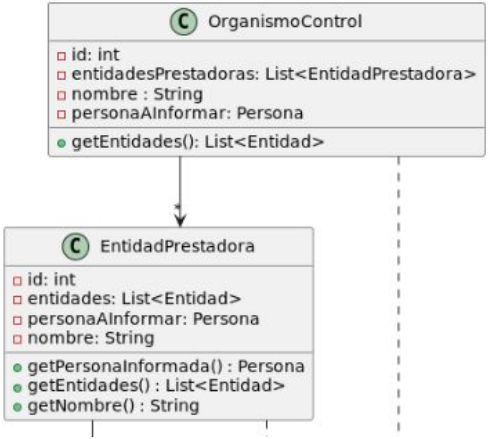
```
class Persona{
  @Column(name = "metodoNotificacion")
  private String metodoNotificacion;

  @Convert(converter = EstrategiaMomentoNotificacionConverter. class)
  @Column(name = "momentoNotificacion")
  private EstrategiaMomentoNotificacion
estrategiaMomentoNotificacion;

  @Embedded
  private Ubicacion ultimaUbicacion;
...
}

class Ubicacion{
  // Embede la ubicacion latitud, ubicacion longitud,
  // ubicacion departamento id, ubicacion_localidad_id
  // y ubicacion_provincia_id
...
}

class EstrategiaMomentoNotificacionConverter {
  // Relaciona "alMomento" con clase  NotificacionAlMomento
  // Relaciona "sinApuro" con clase  NotificacionSinApuro
...
}
```

personas		
PK	id	INT
	apellido	VARCHAR(255)
N	email	VARCHAR(255)
	momentoNotificacion	VARCHAR(255)
	metodoNotificacion	VARCHAR(255)
	nombre	VARCHAR(255)
N	ubicacion_latitud	FLOAT
N	ubicacion_longitud	FLOAT
N	whatsapp	INT
FK	interes_id	INT
FK	notificablesSinNotificar_id	INT
FK,N	ubicacion_departamento_id	VARCHAR(255)
FK,N	ubicacion_localidad_id	VARCHAR(255)
FK,N	ubicacion_provincia_id	VARCHAR(255)
FK	usuario_id	INT

Relaciones con Cascadas

- Se aplica **cascada** en algunas relaciones para simplificar las inserciones, actualizaciones y eliminación de entidades relacionadas.





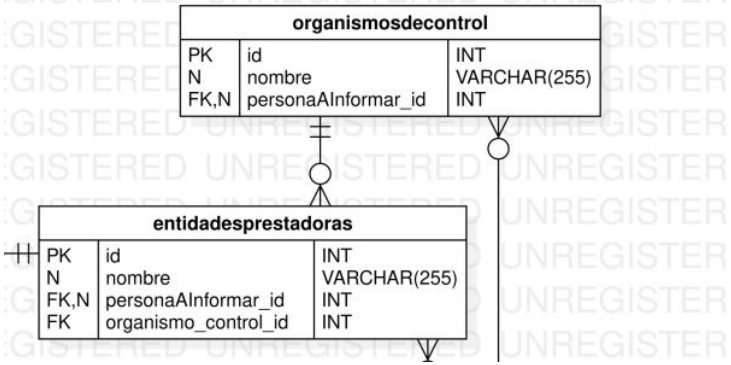
Relaciones con Cascadas

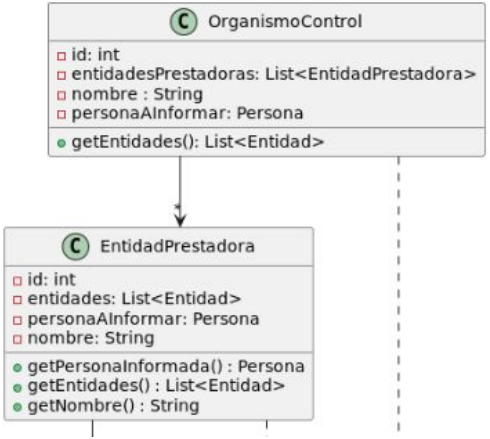
```java
@OneToMany(cascade = { CascadeType.PERSIST, CascadeType.MERGE, CascadeType.REMOVE })
@JoinColumn(name = "organismo_control_id", referencedColumnName = "id")
private List<EntidadPrestadora> entidadesPrestadoras;
```

- Se aplicó en algunas relaciones **OneToMany** que tenemos en el dominio: **OrganismoControl** a **EntidadPrestadora**, **EntidadPrestadora** a **Entidad**, **Entidad** a **Establecimiento**, **Comunidad** a **Membresia**, entre otros.
- Dichas relaciones representan composiciones entre una clase y otra. Donde la supresión de una clase contenedora implica la supresión de la clase contenida o componente. Por ejemplo si se elimina una instancia de **OrganismoControl** , se debería eliminar también las instancias de **EntidadPrestadora** asociadas.

Relaciones con Cascadas

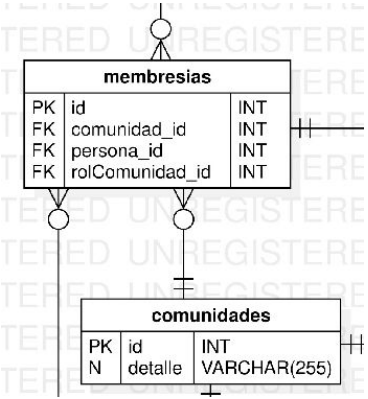
```java
@OneToMany(cascade = { CascadeType.PERSIST, CascadeType.MERGE })
@JoinColumn(name = "organismo_control_id", referencedColumnName = "id")
private List<EntidadPrestadora> entidadesPrestadoras;
```

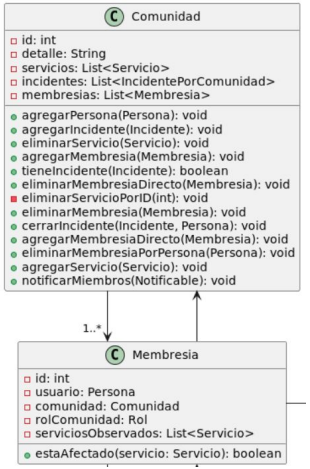




Relaciones con Cascadas

```java
@OneToMany(mappedBy = "comunidad", cascade = { CascadeType.PERSIST, CascadeType.MERGE, CascadeType.REMOVE })
private List<Membresia> membresias = new ArrayList<>();
```





- **Ventajas**
  - No es necesario escribir código para insertar dichas entidades “hijas” relacionadas. Ya que se calcula automáticamente.
- **Desventajas**
  - inserts/updates/deletes se generan automáticamente esto puede generar problemas de performance al trabajar con colecciones enormes de entidades hijas.
  - CascadeType.REMOVE + ManyToMany: puede llegar a generar situaciones de remoción de entidades no esperadas. En este caso se recomienda hacer remoción manual.

Gracias.