TP por parejas #1
Diseño de Sistemas UTN
Cuidándonos

Punto 1 – Arquitectura:

1. Ambas son buenas alternativas, sin embargo en cuanto a **mantenibilidad** el proceso propio será mucho más fácil de adaptar, ya que tenemos la libertad de cambiarlo según nuestras necesidades, a diferencia de una API cuyas configuraciones pueden no abarcar lo necesario para nuestra implementación. En cuanto a **disponibilidad** la cola de mensaje es superior, ya que utiliza una API cuyos servidores son muy robustos dado que son de Google.

2.

a) Selección de respuestas correctas

- ☐ La aplicación deberá ser nativa en su totalidad.
- ☑ ~~La aplicación deberá tener una capa de visualización que corra sobre el sistema operativo del smartphone y la lógica de negocio implementada en un servidor en la nube.~~
- ☐ El cálculo de la distancia deberá hacerla la propia aplicación.
- ☑ ~~El cálculo de la distancia deberá ser delegada a un componente de terceros (tal cual lo presenta el dominio actualmente).~~
- ☐ El dominio podría ser implementado en una base de datos no relacional
- ☐ La aplicación podría tener persistencia políglota

b) Justifique sus elecciones del ítem anterior según el atributo de calidad **performance** y cualquiera de los siguientes que elija que crea que aplican al dominio:

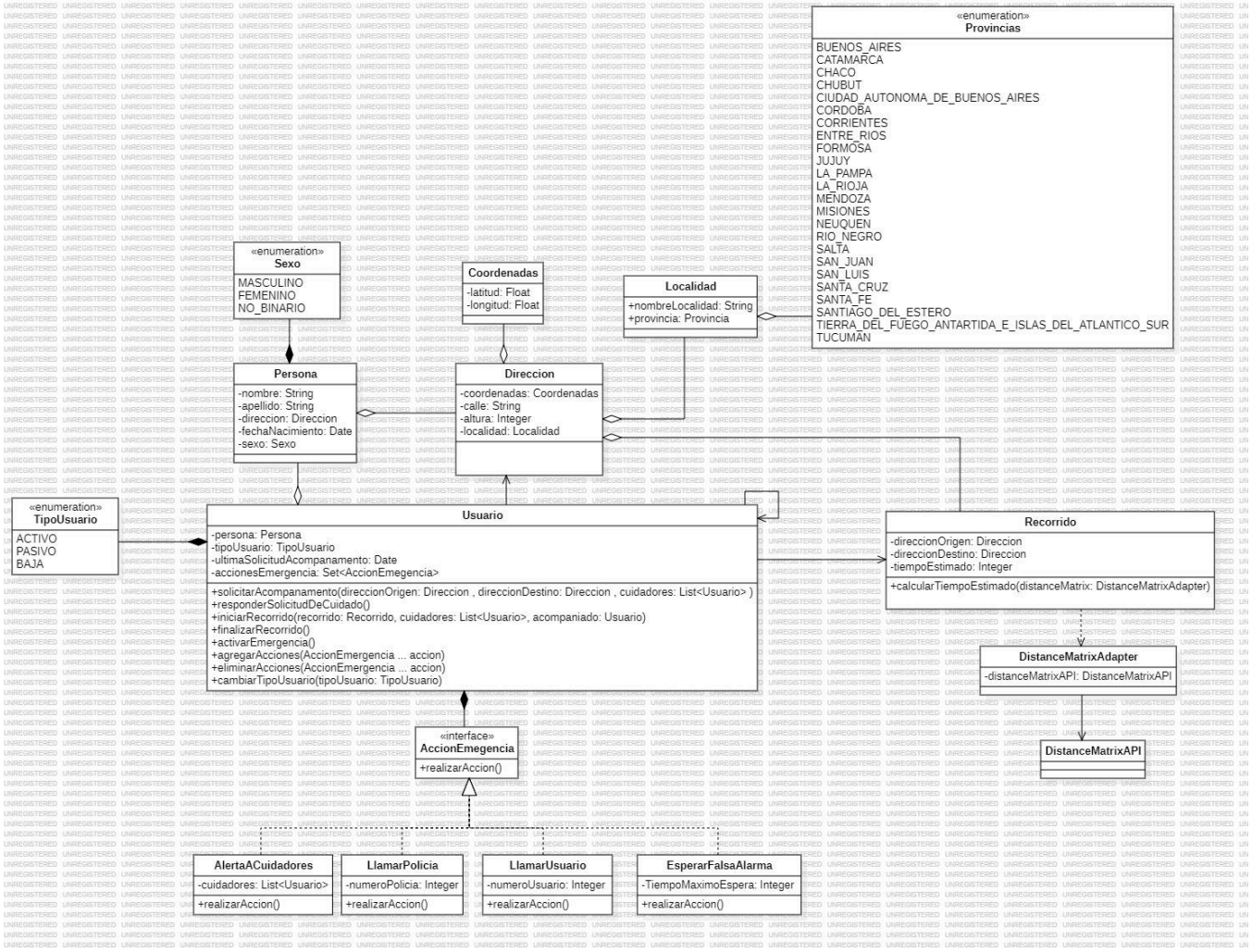
- ☐ Seguridad
- ☑ ~~Portabilidad~~
- ☐ Funcionalidad
- ☐ Usabilidad
- ☑ ~~Eficiencia~~
- ☑ ~~Madurez~~

En cuanto a **performance**, creemos que la alternativa de tener la capa de visualización que corra separada de la lógica de negocios hará que las aplicaciones funcionen mejor independientemente de las capacidades de cada smartphone.
En cuanto a **portabilidad**, la mejor opción es que no sean nativas en su totalidad, ya que, de esta manera solo se debe adaptar la capa visual para los distintos sistemas operativos.

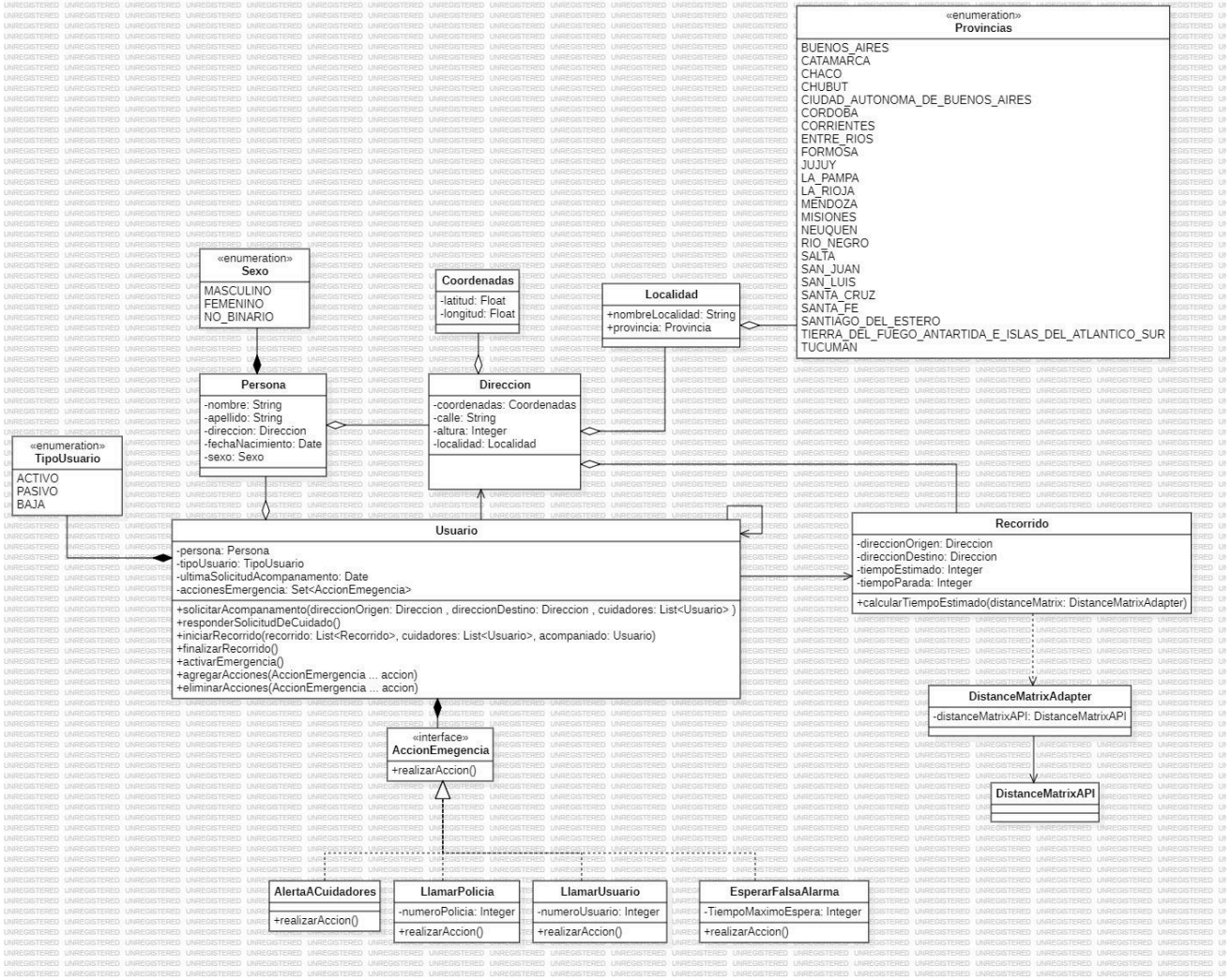
En cuanto a **eficiencia**, nos parece que se debería implementar una base de datos relacional, ya que, no se trabaja con una gran cantidad de datos como para utilizar una base de datos no relacional, o de persistencia políglota.
Y por último, en cuanto a **madurez**, elegimos mantener el cálculo de distancias en un componente de terceros, dado a que estaríamos utilizando una API hecha y testeada por Google, lo cual nos garantiza la certeza de los cálculos necesitados.

Punto 2 - Modelo de Dominio

1.



2.



Patrones utilizados:

- Strategy: se utiliza en AccionEmergencia ya que nos permite tener comportamientos polimórficos entre las distintas clases. Además este nos da facilidad para agregar acciones de emergencia en runtime si es que lo necesitamos.
- Adapter: se utiliza para adaptar las llamadas a la API creada por Google a una interfaz que concuerda con lo requerido en el sistema.