## Cuidándonos: Justificación

Nuestra idea fue crear una única clase “Persona “tanto para transeúnte y cuidador, ya que el usuario debería elegir como comportarse. De esta manera descartamos la idea de dos clases “Transeúnte” y “Cuidador” hereden de la clase “Persona”, ya que no se puede en tiempo de ejecución dejar de ser un objeto para pasar a ser otro.

Se decidió crear la clase “Viaje” en la cual definimos quien es el transeúnte y su cuidador o cuidadores. Tenemos además la dirección de origen y los distintos destinos, que en primera instancia solo nos habla de un solo trayecto sin paradas por lo cual fuimos por dejarlo como un atributo destino que tenga referencia a una dirección, pero en el punto 2 nos indica extender la funcionalidad para que haya varias paradas en el trayecto entonces agregamos la clase “Destino” con una dirección, tiempoDetenido en ese punto, un orden para seguir y un estadoDeLlegada para saber si llego bien a destino. Para las notificaciones se creó una clase “Notificador” que será el encargado de enviarlas, a la cual “Viaje” tiene un atributo para referenciarla, además de otro para habilitar cuando se podrá enviar notificaciones. Los métodos para calcular tiempo estimado tanto total como por sección que deberán recibir por parámetro la una instancia de” CalculadorDeTiempo “. Se creo esta clase, que tiene un método “tiempoAproximadoEntre” que recibe como parámetro dos Direcciones y devuelve un entero que representa al tiempo en minutos que se tarda en recorrer la distancia entre esas dos direcciones. Ahora bien, si bien la API de Google permite calcular la distancia entre dos puntos (dados por dos direcciones), no permite calcular el tiempo estimado en recorrer dicha distancia. Para eso suponemos que la velocidad promedio que la tenemos por archivo de configuración. Se calcula el tiempo estimado asi: $T = \frac{Distancia}{VelocidadPromedio}$ .

***” La distancia (en metros) entre dos direcciones será calculada por “Distance Matrix API” de Google, cuyo sistema nos brinda una interface REST”:***

No sabemos cómo acoplarnos a ese sistema, pero sí sabemos qué resuelve ese sistema. Como no sabemos exactamente cómo nos vamos a integrar a esa API, delegamos esa responsabilidad en un ADAPTER.

Entonces, la clase CalculadorDeTiempo se encargará de llamar al “AdapterCalculadorDistancia” para preguntarle la distancia en metros. Luego, el método “tiempoAproximadoEntre” obtendrá, con el dato de la distancia en metros recientemente calculado, el tiempo en minutos

***“El sistema va a reaccionar frente a este incidente según lo que haya configurado el usuario (enviar un mensaje de alerta a sus cuidadores; realizar una llamada automática a la policía; realizar una llamada al celular del usuario; esperar N minutos para ver si es una falsa alarma -los minutos deben ser parametrizables). Se debe considerar que pueden surgir nuevas formas de reaccionar frente a un incidente y que el usuario puede cambiar esta configuración cuantas veces quiera”:***

Para configurar las estrategias de reaccionar se pensó en un STRATEGY, porque es una estrategia que puede configurar manualmente el transeúnte, asumiendo que se pueda usar sólo una estrategia al mismo tiempo.

## Código:

```
public int duracionEstimadaTotal (CalculadorDeTiempo calculador) {
    destino.sort((destino1, destino2) -> Integer.compare(destino1.getorden(),
destino2.getorden()))
    int tiempoTotal = 0;

        Destino primerDestino = destinos.get(0);
        tiempoTotal += calculador.tiempoAproximadoEntre(origen,
primerDestino.direccion);


    for (int i = 1; i < destinos.size(); i++) {
        Destino destinoAnterior = destinos.get(i - 1);
        Destino destinoActual = destinos.get(i);
        tiempoTotal += demoraEstimadaPorSeccion(calculador,
destinoAnterior.direccion, destinoActual);
    }

    return tiempoTotal;
}

public int demoraEstimadaPorSeccion(CalculadorDeTiempo calculador, Direccion
origen1, Destino destino) {
    return calculador.tiempoAproximadoEntre(origen1, destino.direccion) +
destino.gettiempoDetenido();


}
```