Decisiones de Diseño

Integrantes: Matias Sicher y Lautaro Petronacci

TP: Actividad Cuidandonos - De a pares

Fecha: 01/05/2024

Justificaciones y explicaciones de Diseño

* **Clase Persona**
  + Decidimos que cada Persona que usa la app, además de conocer sus propios atributos, tenga un TipoPersona que podrá variar entre Cuidado o Cuidador.  
    Esto lo hicimos para mantener separados los datos genéricos de una persona como los datosPersonales (llevado a otra clase por una cuestión de comodidad en la clase Persona), de los datos relacionados al rol asignado en el momento determinado (cuidador o cuidado).  
    Consideramos también que tener una herencia en el tipoPersona para los dos roles actuales (que suponemos en el futuro pueden agregarse más) debido a que, aunque manejen datos casi idénticos entre sí, tienen comportamientos muy distintos con los mismos.  
    Además la persona podrá tomar en cualquier momento el rol de cuidador o cuidado, pero no al mismo tiempo.
* **Incidentes:**
  + Se utilizó el patrón de diseño “Strategy”, para que la persona pueda cambiar el modo en el que la app va a actuar ante un incidente. El estilo elegido fue el de tener una interfaz como estrategia principal, ya que no tiene estado interno propio en sí.
  + Se decidió que la Persona conozca y cambie la estrategia para que pueda cambiar de TipoPersona sin tener que elegir una estrategia cada vez que una persona Cuidador vuelva a ser Cuidado.
  + La implementación de este método fue elegida debido a que resuelve rápida y polimórficamente el requerimiento de posible agregado y eliminado de estrategias y además, porque todas implementan la misma acción (ejecutar la estrategia elegida) pero cada una tiene un comportamiento diferente.
* **Viaje:**
  + Cada persona tendrá asignado un viaje (sea activa o pasiva). Los diferentes comportamientos con respecto al viaje serán manejados en las clases que implementan a tipoPersona.  
    El viaje será el encargado de bloquear las notificaciones del transeúnte una vez comience. Por eso decidimos que tanto la persona como el viaje se conozcan entre sí.  
    Esa clase tendrá a su vez una clase Destino.la cual tiene una lista de objetos Parada dentro. Decidimos crear una lista para garantizar la unicidad y multiplicidad de destinos por viaje. Por un tema de inicialización para el método calcular demora, se agregó una posicionInicial como atributo.
  + Dentro de cada parada se especifica además el tiempo de demora y la dirección de esa parada.
* **Cálculo de Demora Aproximado:**
  + El cálculo de la demora lo dividimos en dos. Un método es para calcular la demora total aproximada, reutilizando el segundo método que es para calcular específicamente entre dos paradas. De esta manera se pueden calcular las diferentes demoras reutilizando la mayoría del código.
  + Además, el calcularDemoraEntre es el método que integra la API Rest con “Distance Matrix API” de Google. No realizamos la implementación pero quedó explicitado en el pseudo-código adjunto.