TRABAJO FIN DE GRADO

INGENIERÍA EN INGENIERÍA INFORMÁTICA

Software para el diseño de rutas con puntos de interés

Software para el diseño de rutas con puntos de interés en dispositivos móviles

Autor
Alberto Armijo Ruiz

Directores
David Pelta

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

29 de abril de 2018

Software para el diseño de rutas con puntos de interés: Software para el diseño de rutas con puntos de interés en dispositivos móviles

Alberto Armijo Ruiz

Palabras clave: Sistema recomendador, Touring Trip Design Problem, Turismo, Dispositivo móvil.

Resumen

El objetivo de este trabajo ha sido documentar los aspectos más significativos del diseño, desarrollo y despliegue de un sistema recomendador de turismo para la generación de rutas de puntos de interés en ciudades que se ejecute en dispositivos móviles.
Para ello, fueron esenciales los conocimientos y la experiencia en programación adquirida a lo largo del grado, en especial la aportada por las asignaturas "Metaheurística" y "Programación de dispositivos móviles"; las cuales el autor cursó durante el tercer y cuarto curso del Grado en Ingeniería Informática de la Universidad de Granada.
El sistema resultante es una sistema recomendador de turismo que consta de una aplicación y, a través de la interacción con diferentes servidores para obtener la información sobre los puntos de interés y un algoritmo para obtener rutas integrado en la app, obtiene y dibuja diferentes rutas entre las cuales el usuario puede elegir. Además, el usuario puede utilizar filtros para generar diferentes rutas, dependiendo de sus gustos o necesidades.

Software for the desing of routes with points of interest:Software for the desing of routes with points of interest on mobile devices.

Alberto Armijo Ruiz

Keywords: Recommender system, Touring Trip Design Problem, Tourism, Mobile device.

Abstract

The objective of this work was to compile the most significant aspects of design, development and deployment of a tourism recommender system for the creation of POI's routes in cities executed in mobile devices.

The knowledge and experience acquired during the degree were essential, especially those from the subjects "Metaheurísticas" and "Programación de dispositivos móviles"; which the autor attended during the third and fourth year of the "Grado en Ingeniería Informática" of the University of Granada.

The resulting system is a tourism system recommender with an app that, through the interaction with differents servers used for getting information about the POI and an algorithm for obtaining routes integrated in the app, gets and draws differents routes among which the user can choose. The user can also use filters to obtain different routes, depending on his needs and his likings.

Yo, **Alberto Armijo Ruiz**, alumno de la titulación Grado en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 26256219V, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Nombre Apellido1 Apellido2

Granada a 24 de Abril de 2018 .

D. **Nombre Apellido1 Apellido2 (tutor1)**, Profesor del Área de XXXX del Departamento YYYY de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Título del proyecto, Subtítulo del proyecto***, ha sido realizado bajo su supervisión por **Nombre Apellido1 Apellido2 (alumno)**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a X de mes de 201 .

Los directores:

Nombre Apellido1 Apellido2 (tutor1) **Nombre Apellido1 Apellido2 (tutor2)**

Índice general

Capítulo 1

Introducción

1.1. Motivación

1.2. Antecedentes y estado del arte

Capítulo 2

Especificación de Requisitos

2.1. Requisitos

En este apartado se resumirán los requisitos funcionales y no funcionales del proyecto, dividiéndolos en los requisitos de la interfaz de usuario y los requisitos internos de la aplicación.

2.1.1. Requisitos de la interfaz del usuario

Requisitos funcionales

- Mapa de la ciudad elegida por el usuario.
- Formulario de generación de rutas.
  - Alojamiento: hostales y hoteles.
  - Preferencias del usuario: museos, miradores, puntos históricos.
- Especificación de las rutas: dibujado de los nodos de las rutas.
- Información sobre los marcadores.

Requisitos no funcionales

- Interfaz: la interfaz deberá ser atractiva, ligera y lo más intuitiva posible.

2.1.2. Requisitos de la aplicación

Requisitos funcionales

- Recepción de peticiones del cliente.
- Cálculo y retorno de rutas óptima.
- Publicación de lista de alojamientos.

Requisitos no funcionales

- Envío de peticiones de puntos de interés y alojamientos. (Overpass y OSM)
- Importación de información sobre puntos de interés y alojamientos.
- Envío de peticiones a servidor de cálculo de matrices de tiempos.
- Importación de matriz de tiempos.

2.2. Diagramas de casos de uso

2.2.1. Interfaz de usuario

2.2.2. Aplicación móvil

2.3. Fuente de los datos

2.3.1. Open Street Map

2.3.2. Overpass API

2.3.3. Open Source Routing Machine

Capítulo 3

Planificación

3.1. Objetivos

Los objetivos concretos que persigue este proyecto son los siguientes:

1. Estudio del estado del arte de los sistemas de planificación de rutas con recomendaciones, y de los sistemas de información sobre puntos de interés en las ciudades.
2. Definición y diseño de modelos y herramientas de recomendación de itinerarios ajustados a las necesidades y preferencias de los turistas.
3. Validación de modelos y métodos, empleando pruebas y datos de puntos de interés en ciudades de interés turístico.
4. Obtención de un prototipo software integrado en una aplicación para dispositivos móviles.
5. Creación de documentación técnica, entregables y memoria final del proyecto.

Capítulo 4

Análisis

En este capítulo se explican los modelos teóricos y algorítmicos empleados para el diseño de las rutas y el desarrollo del sistema recomendador.

4.1. Diseño de rutas turísticas

Los problemas de diseño de rutas turísticas (Tourist Trip Design Problems, TTDP) consisten seleccionar los puntos de interés (point of interest, POI) a visitar por un turista atendiendo a sus restricciones y al beneficio o grado de satisfacción que produce su visita. El turista dispone en su estancia de un cierto número de días para organizar las visitas a los puntos de interés mediante rutas de duración limitada. Se parte de un conjunto de puntos disponibles a visitar de los que se conoce, el beneficio o grado de satisfacción, la duración de la visita y el intervalo de tiempo en el que puede realizarse. El beneficio total que intenta maximizar el turista es la suma de los beneficios obtenidos en cada visita.
Los elementos que forman parte del modelo son:

- Un conjunto de puntos de interés (POI), asociado a un índice $i, i = 1, 2, ..., n$ y con los siguientes atributos:
  - Una puntuación o beneficio s_i.
  - Un tiempo de duración de la visita r_i.
  - Un intervalo de tiempo $[e_i, l_i]$ dentro del que se puede realizar la visita.
- Un punto de partida de cada una de las rutas denotado por $i = 0$.
- Los tiempos de recorrido entre los pares de puntos $t_{ij}, i = 0, 1, ..., n$.
- Un tiempo máximo T_{max} de duración total de la ruta del día considerando los tiempos de viaje, visita y espera en los POI.
- Una función objetivo $max \sum_{i=1}^{n} s_i y_i^k$
  En la que y_i corresponde a una variable de visita para cada uno de los puntos de

interés $i, i = 1, ..., n$. Dicha variable es binaria, teniendo como valor 1 si el punto de interés i es visitado en la ruta y un 0 si no se visita.

El problema de optimización coincide con el Team Orienteering Problem with Time Windows (TOPTW) que se ha estudiado en la literatura científica con algunas modificaciones. Se consideran puntos de interés museos, miradores, catedrales, mezquitas, etc... de la ciudad que el usuario elija.

4.2. Heurísticas del diseño

Para dar soluciones de alta calidad al problema de optimización planteado se ha elegido la metaheurística constructiva GRASP (Greedy Randomized Adaptive Search Procedure) que comprende dos fases: una fase contructiva y otra de búsqueda local. En la fase constructiva se genera una solución partiendo de una ruta vacía a la que se va añadiendo puntos de interés desde una Lista Restringida de Candidatos (Restricted Candidate List, RCL en inglés) de forma aleatoria hasta que no se puedan añadir nuevos puntos a la ruta. En la fase de búsqueda local se reemplaza la solución obtenida en la parte constructiva por la mejor de sus soluciones vecinas si existe mejora. Estas dos fases se ejecutan un cierto número de iteraciones.

Ahora se mostrará el pseudocódigo del algoritmo, y de cada una de las fases en inglés.

Algorithm 1 Pseudocódigo algoritmo GRASP

```
function GRASP(maxIterations,sizeRCL)
    readInput()
    for i = 1 to maxIterations do
        solution ← GRASPConstructPhase(sizeRCL)
        solution ← LocalSearch(solution)
        if solution ≥ bestSolution then
            bestSolution ← solution
        end if
    end for
    return bestSolution
end function
```

Para la parte constructiva del algoritmo, primero crearemos una lista de candidatos (CL), la cual tiene un tamaño igual al de la lista de puntos de interés que queden disponibles. Dicha lista de candidatos contiene la posición dentro de la lista de puntos de interés y la puntuación que tiene dicho punto de interés según las preferencias elegidas por el usuario. La forma de puntuar a cada punto de interés se hace de la siguiente manera:

$\frac{1}{time\_between(poi_{i-1}, poi_i)} \sum_{j=1}^{n} c_j$

Donde time_between(x,y) representa el tiempo que hay entre el punto de interés x y el

y. poi_{i-1} representa el último punto de interés que hemos incluido en la solución y poi_i al punto de interés que estamos evaluando; c_j es cada uno de las preferencias posibles sobre los puntos de interés, es decir, si se ha seleccionado museos como preferencia y el punto de interés i es un museo, c_j devolverá uno, es caso contrario es cero.
Una vez hemos calculado la lista de candidatos, se crea la lista restringida de candidatos con los puntos de interés mejor valorados, esta es de tamaño sizeRCL. Una vez hemos obtenido la lista de candidatos restringida, elegimos uno de los puntos de interés de forma aleatoria y lo introducimos dentro de la solución. Este proceso se repite hasta que no se puedan introducir más puntos de interés dentro de la solución. El pseudocódigo de este algoritmo es el siguiente.

Algorithm 2 Pseudocódigo algoritmo GRASPConstructPhase

```
function GRASPConstructPhase(sizeRCL)
    solution.add(first_node)
    while is posible to visit POIs do
        for each poi in POIs do
            value ← f(poi)
            cl.add([value, i])
        end for
        Create rcl with the top sizeRCL duos in cl
        Select a random duo from rcl.
        Update solution adding the poi in the position i.
    end while
    return solution
end function
```

Para la fase de optimización se utilizará el algoritmo de búsqueda local, dicho algoritmo busca la mejor solución posible entre la solución actual y los vecinos de esta. Una solución vecina es aquella que intercambie dos puntos de interés dentro de la solución, por ejemplo, el segundo por el cuarto. El procedimiento es el siguiente, se generan todos los posibles vecinos de la solución, y por cada uno de ellos se comprueba si la valoración de dicha solución vecina es mejor que la mejor solución actual; finalmente se devuelve la mejor solución encontrada. El pseudocódigo del algoritmo es el siguiente.

Algorithm 3 Pseudocódigo algoritmo LocalSearch

```
function LocalSearch(solution)
    for all neighbor_solution of solution do
        if neighbor_solution ≥ solution then
            solution ← neighbor_solution
        end if
    end for
    return solution
end function
```

Capítulo 5

Diseño

5.1. Diagrama de clases

En este apartado se mostrarán los diagramas de clases de los elementos más importantes de la aplicación.

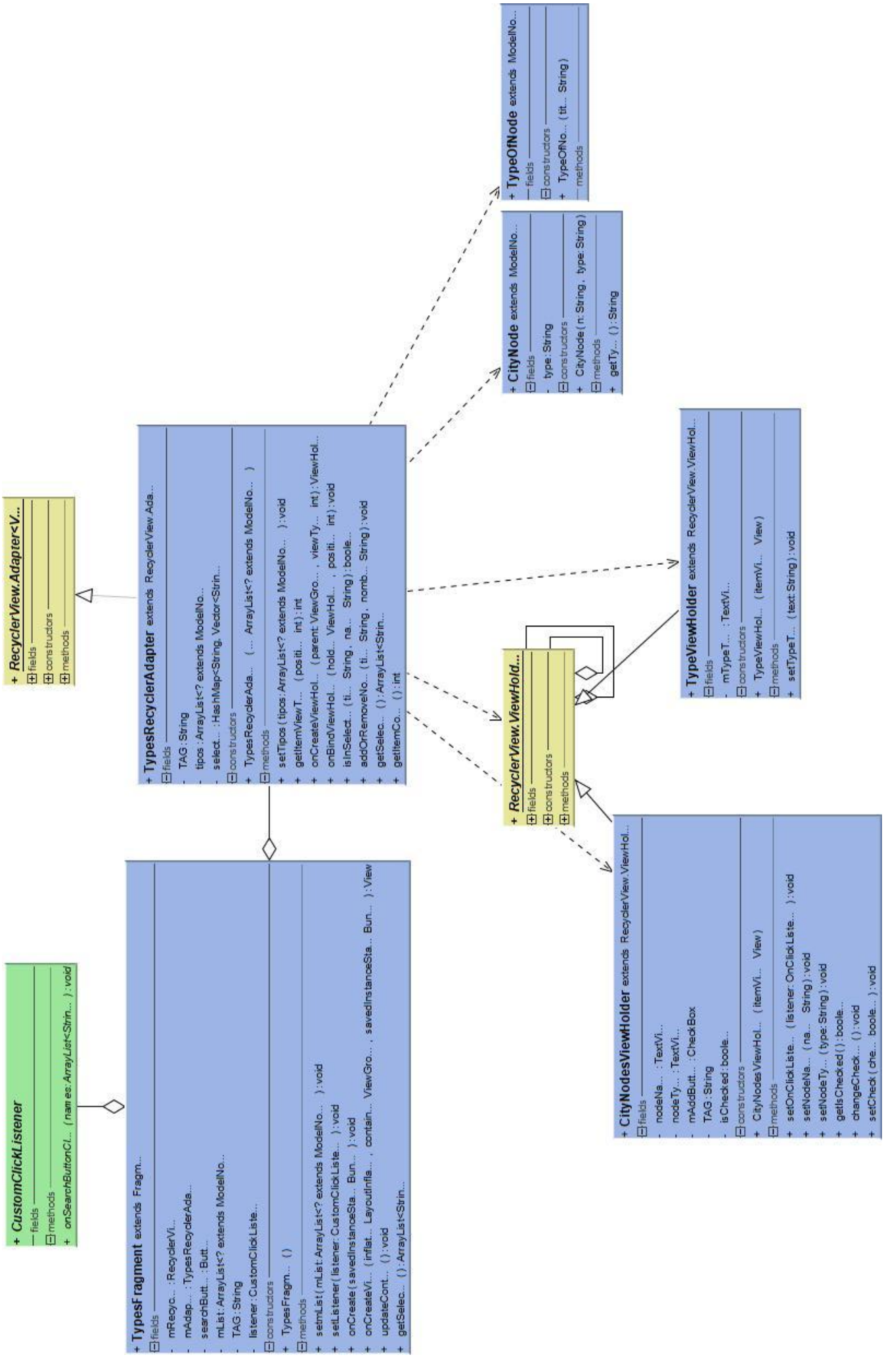

Figura 5.1: Diagrama de clases del fragment

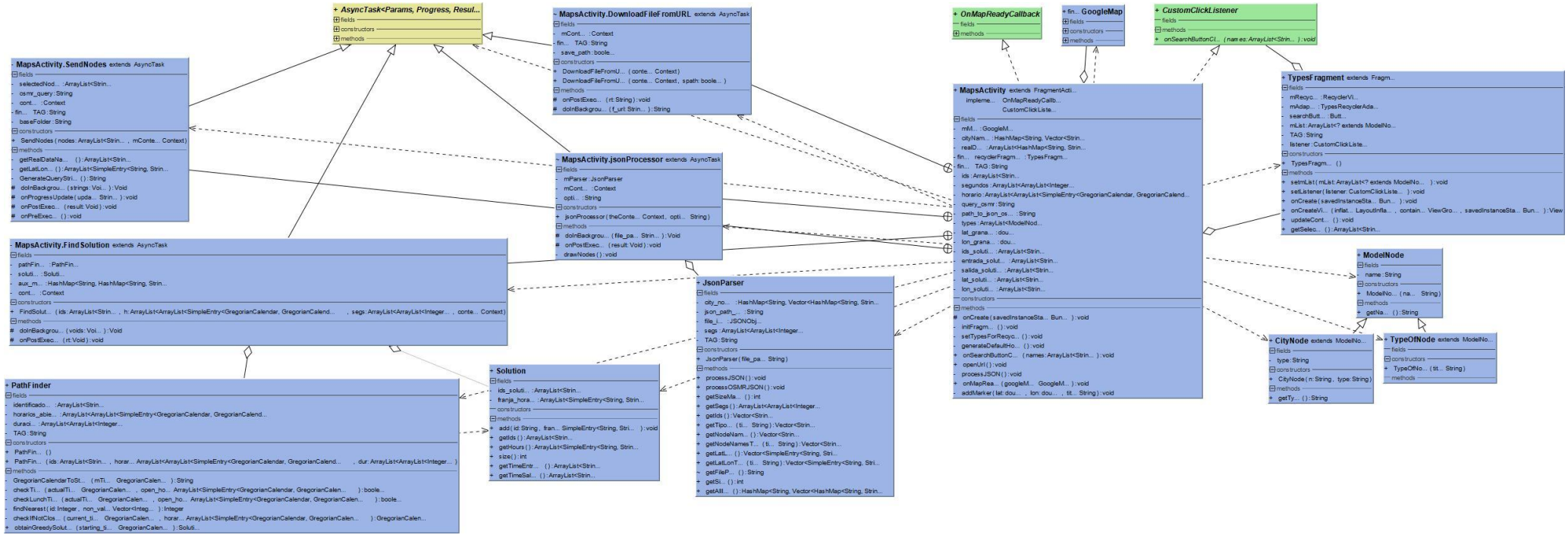

Figura 5.2: Diagrama de clases de la actividad principal

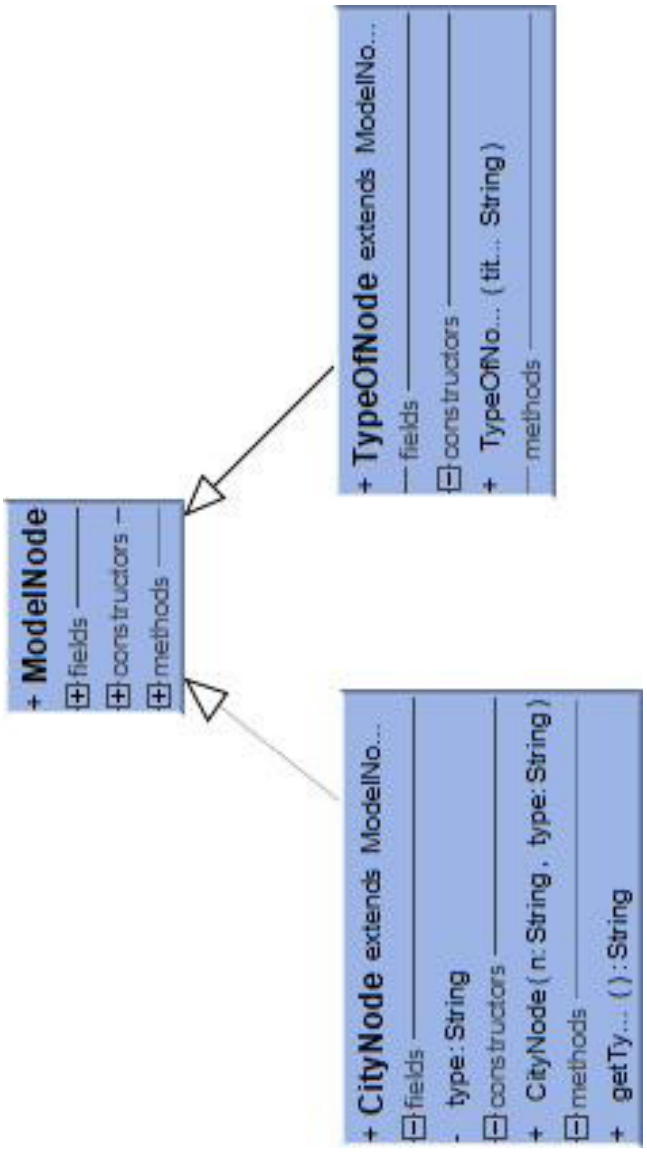

Figura 5.3: Diagrama de clases del paquete models

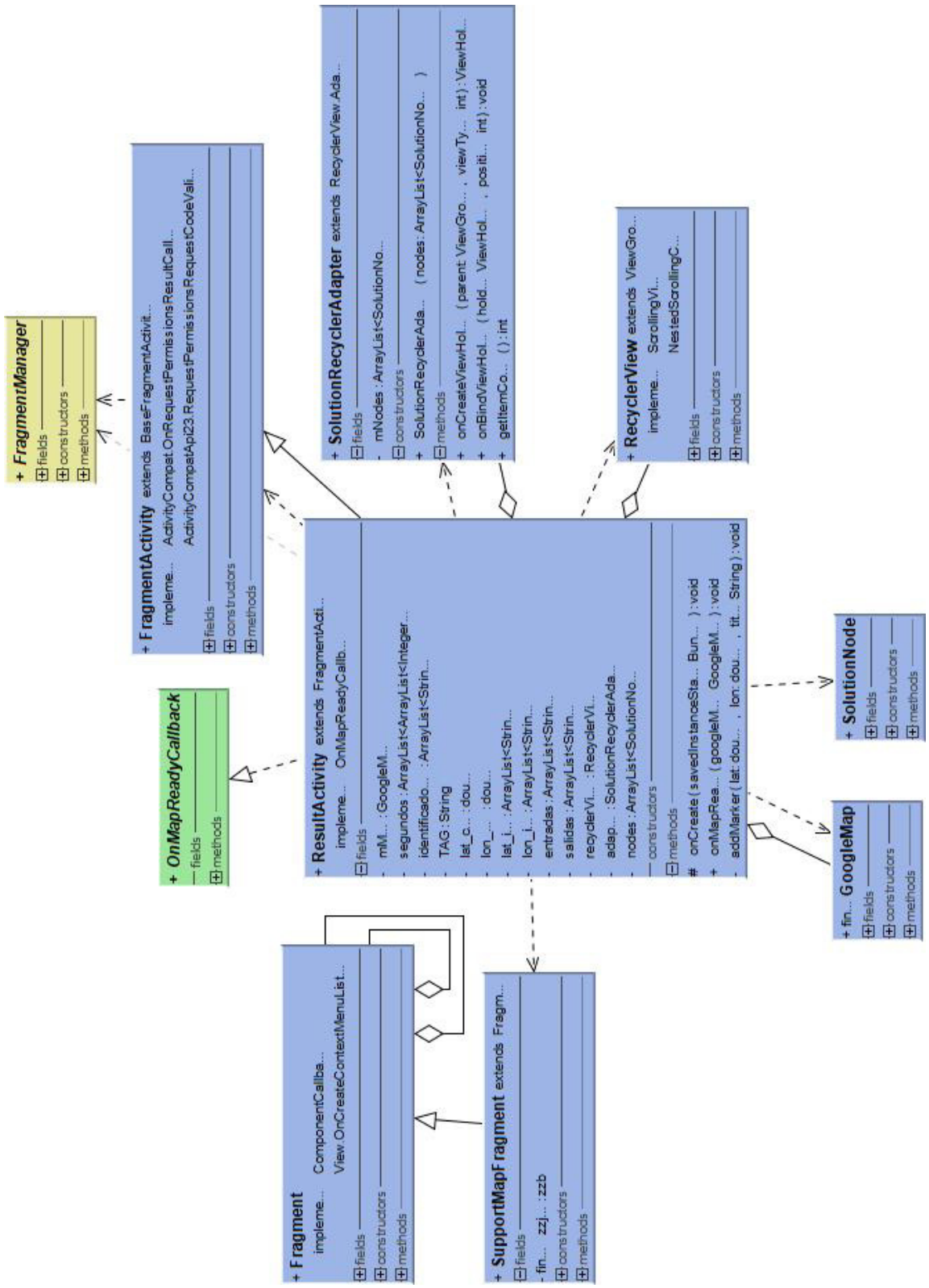

Figura 5.4: Diagrama de clases del activity ResulActivity

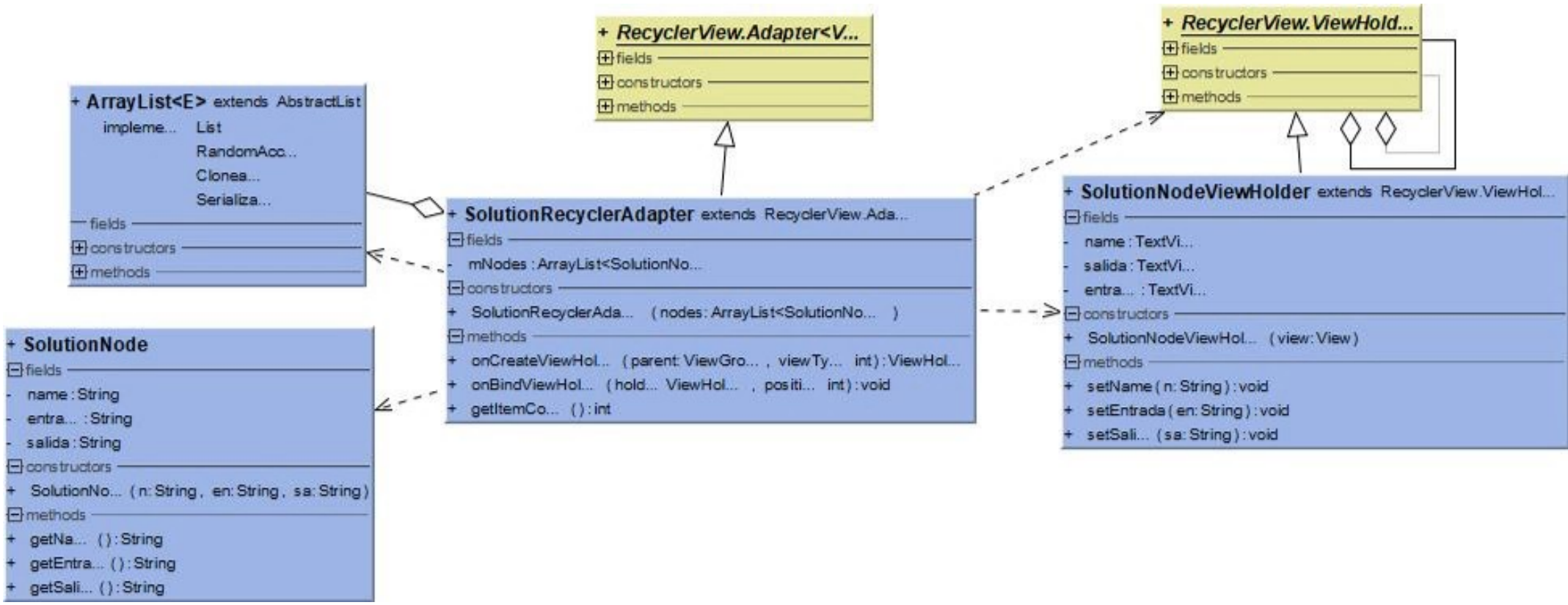

Figura 5.5: Diagrama de clases de la clase SolutionRecycler

Capítulo 6

Implementación

Capítulo 7

Pruebas

Capítulo 8

Conclusiones