Ba-Herausfindungen

* Ein und Ausgabepunkt PlannerResource.java in org.opentripplanner.api.resource
  + Wichtige weitere Punkte:
    - getRouter 🡪 Bildung des routers. Requestbasiert. Einer pro request
    - GraphPathFinder 🡪 Erstellt eine Pathfinder mit dem erzeugten Router
    - graphPathFinderEntryPoint 🡪 Findet die zu den Request Parametern passenden Paths.
    - GraphPathtoTripPlanConverter.generatePlan 🡪 wandelt gefundenen Pfad in die vom Webservice benötigte Form um.
* Mögliche Query Parameter in RoutingResponse in org.opentripplanner.routing.core
* RoutingRequest

# Ablauf:

1. Anfrage auf routers/{routeId}/plan
2. Response wird erstellt und Querry Parameter warden in die Response geschreiben
3. Request wird erstellt
   1. Router wird geladen
   2. defaultRoutingRequest wird geklont und dem Request zugewiesen
   3. routerId wird dem request element zugewiesen
   4. Querry Parameter überschreiben die default Parameter falls gesetzt
4. Router wird mit dem Parameter der routerId erstellt
   1. Router ist der Graphservice der für die routerId einen Graphen erstellt und diesen für tieferliegende Opjekte verfügbar macht
5. GraphPathFinder wird mit dem soeben erstellten Router iniziiert
   1. Enthält die Algorithmuslogik.
   2. Wird von uns sicher ersetzt
6. Paths werden mit dem GraphPathFinder gefunden.
   1. Einstiegspunkt über graphPathFinderEntryPoint
   2. Returned path
7. GraphPathToTripPlanConveter konvertiet die Paths in die vom Webservice benötigte Form.
   1. Unser Schlusspunkt am bessten vor oder nach diesem Schritt.
   2. Die rückgabeform unseres Algorithmusses muss die gleiche sein wie die des alten.
8. TripPlan wird in die Response geschrieben.
9. Error Catching
10. Debug Output
11. Reqeust cleanup
12. ElevationMetaData
13. Logging