Canovaccio

Slide 1

- Read the slide.

Slide 2 – UML

- A client is just a user accessing the application, there's no need for authentication
- The client can interact with map with his input devices; you might be familiar with services like Google Maps, OpenStreetMap or Bing Maps
- Clicking with a transit's stop will show the lines that offer a service there and their timetables
- **TIMS** is a live feed of disruptions in the city. It's provided by Transport for London and it contains information related to the most important disruptions on the network:
  - Road works
  - Major accidents: collisions between vehicles, burst water pipes et similia
- We will basically do an **ON-LINE SCRAPING** of TIMS' data, so we view TIMS as an actor in our software system
- The **"statistician"** runs analytics

Slide 3 – Dataset

- We'll merge the **3 DATASETS READ THE SLIDES**

Slide 4 – Classes

- This is a **SIMPLIFIED VIEW** with only the important bits

- The road network is a graph of vertexes and edges, the highway stretches are pieces of network that connect **TWO** intersections, they also carry infos such as
  - Access restrictions on vehicles, pedestrians or animals
  - The speed limit
  - The class of road (motorway, A-road, secondary road, residential street, pedestrian-only path, path for pedal cycles, etc…)
- We can use **geospatial indexes** in MongoDB and Neo4j to look for *WGS84* coordinates in $n\log(n)$ time.
- We **duplicate** data about **active disruptions** to aid the routing algorithm find the optimal path. Edges with **ROAD CLOSURES** are ignored in the search, and edges with **HIGH CONGESTION** are given an higher cost

Slide 5 – Mongo Database

- We save the POIs and their coordinates (with index)
- If a POI is a transit's stop it also contains all its timetables
- Current and historical TIMS' data are stored there
- The heatmap is basically a list of squared areas, each of those squares has an intensity level (eg [0,1]). We can then use the query's result to render a map.

Slide 6 – Rédis

- It is used to **accelerate** response time for the most common routing problems.

Slide 7 – Neo4j

- We think that a flipped representation of the road's network **might** improve traversing performance

- In OpenStreetMap the edges contain the most information!

Slide 8 – Implementation

- We have an inclination in developing a classical **DBMSes, HTTP Server, Web Client** application either in Python or Java.
- A standalone application is not be ruled out
- Most of the heavy lifting will be done by the server
- We can opt for two approaches to render the map
  - send to the client **PNG tiles**, like OpenStreetMap.org and Bing Map
  - send to the client **vectors, points and other stuff** and let it render, like Google Maps