MyTaxiService Design Document

Giorgio Pea - Andrea Sessa

What this document is about?

- High level components of MyTaxiService and their interactions
- The architecture of MyTaxiService: overview, description and motivations
- Some algs designed for MyTaxiService

Some terms and concepts used in this document are defined in the RASD so we haven't defined them again here

Main Components

MVC

Controller Components:

→ **Request Receiver**

Manages all the remote requests from Users, Mtaxies and Administrators by invoking services on the Request Manager

→ **Request Manager**

- Manages all kind of requests coming from the Request Receiver
- Manages a situation of non fair distribution of mtaxies in the city's zones

Main Components

MVC

→ **Dispatcher**

Dispatches async and sync messages from MyTaxiService(B) to users, administrators, mtaxies

→ **Queue manager**

- Organizes mtaxies in zone per zone queues and manages these queues.
- Checks periodically if the distribution of mtaxies in the cities zones is fair or not

Main Components - 1

MVC

Model Components:

→ **Data Manager**
Manages the interaction with the database system

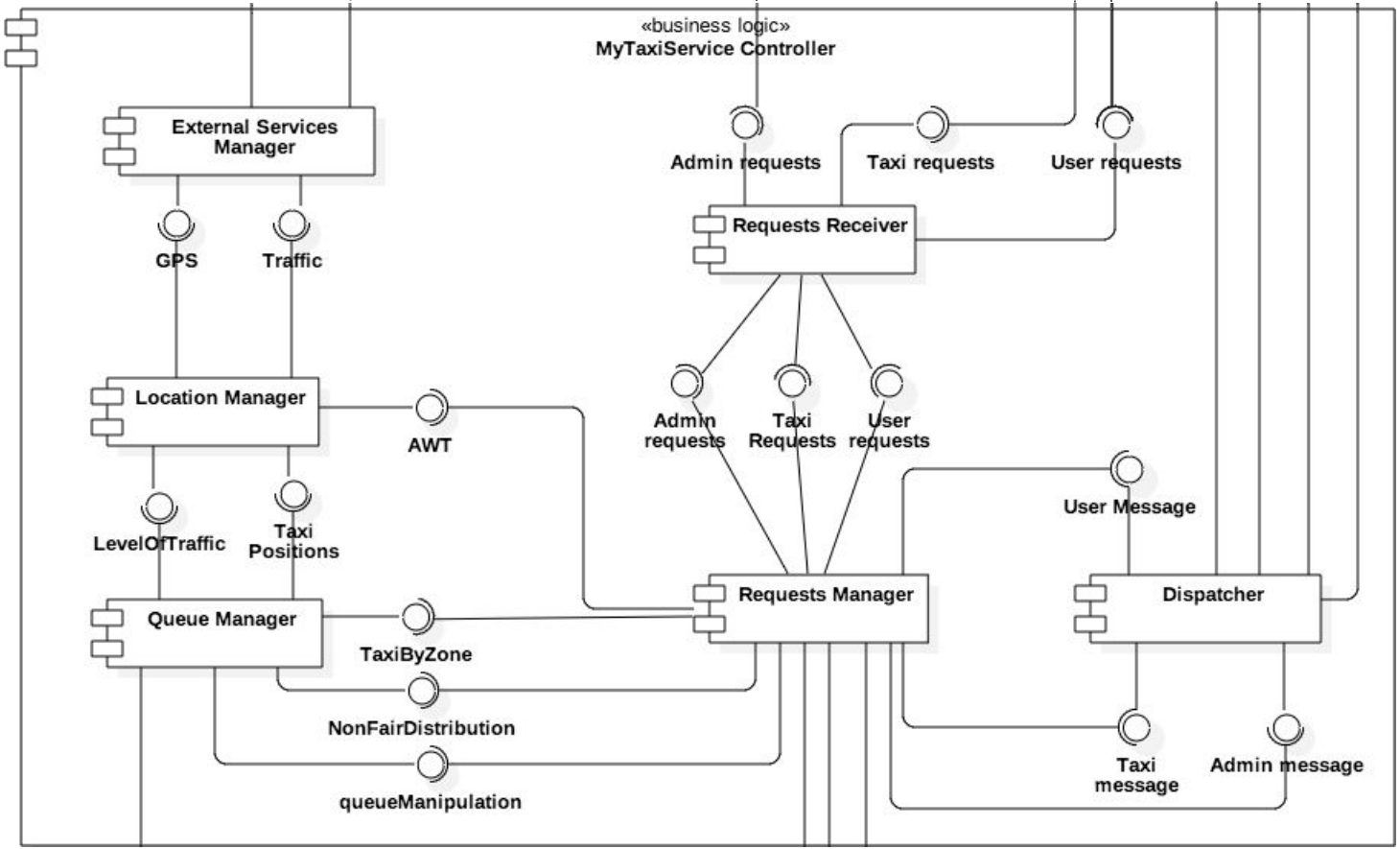
View Components:

→ **GUIs:**
Manages the user(general meaning of the term) interaction

→ **Communication Managers:**
Manages the dispatching of requests to and the responses from MyTaxiService(B)

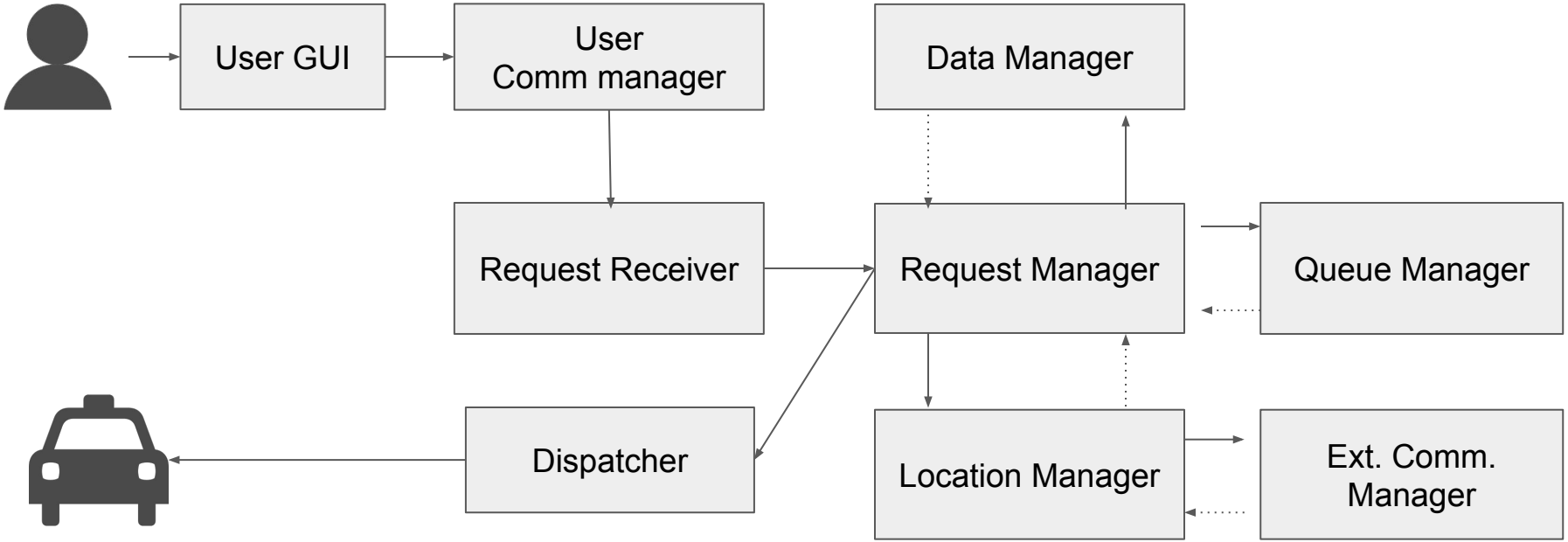
External Data Source

View components



Model components

Component Interactions - Ride Request



Architecture

Architecture

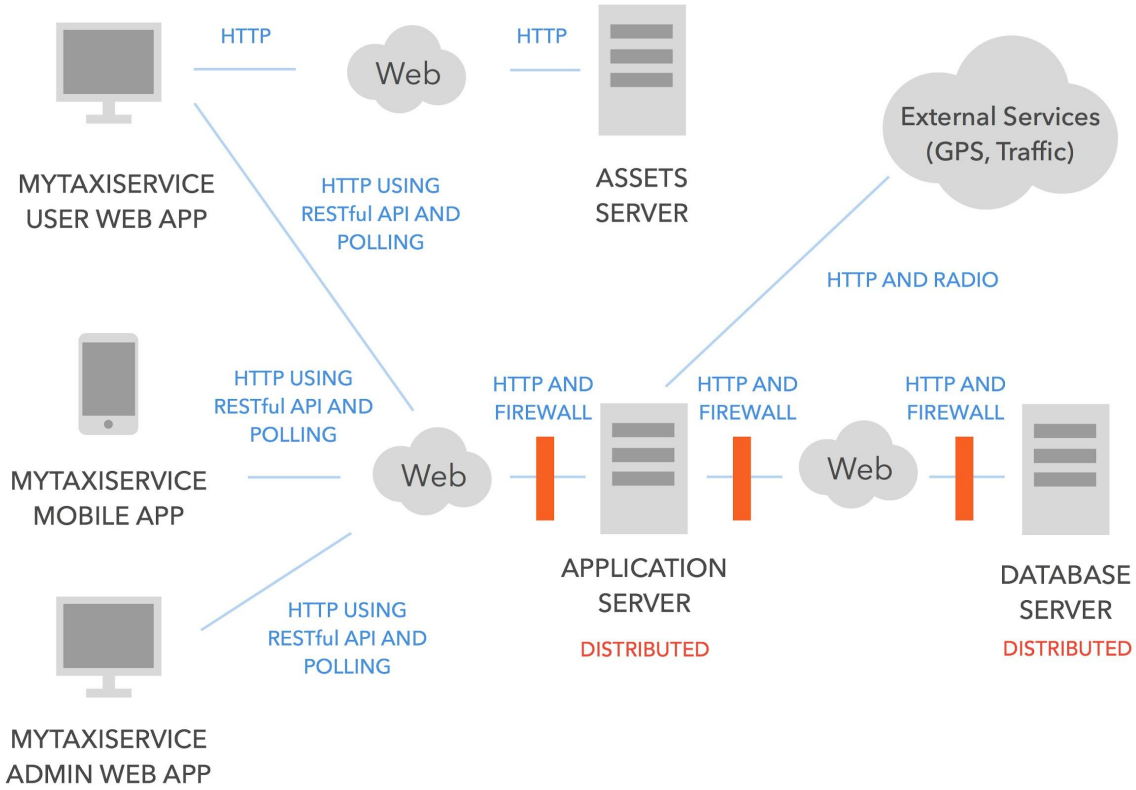
Preliminary considerations:

- The number of requests and of users could be very high
- In the future MyTaxiService can be made available for other cities
- The number of requests can have a very high variance

3 TIER

Architecture - 1

CLIENT SERVER



Architecture - 2

View tier:
Computers and smartphone used to access MyTaxiService websites and mobile application **+** MYT devices

Controller tier:
A distributed application server that implements the bussiness logic of MyTaxiService

Model tier:
A distributed database server that manages the persistent data of MyTaxiService

Architecture - 3

Motivations:

→ **Scalability**

The application server and db server can be further distributed without the need of redesign the whole system

→ **Reliability**

Failures can be isolated easily and db replications can be implemented without impacting too much on the design of the whole system

Architecture - 4

→ **Flexibility**

The chosen architecture can easily be enriched with a load balancer that distributes and activates nodes in relation to the number of incoming requests

→ **Security**

Since the chosen architecture divides strictly business logic and data and since the application server and the db server are isolated from the web using firewalls, malicious attacks can be prevented and an high level of security granted

Algorithms Design

Mtaxies Distribution Checker

Objective: Redistribute mtaxies in the city's zones according to requests' expectancy

Requests' expectancy for a given zone is estimated using:

- Historical data
- Real time data(last hour)
- Zone's size

*expectedRequests = ((historicalAvg * w1+currentAvg * w2) * zoneWeight())*

Mtaxies Distribution Checker - 1

At each execution the algorithm performs the following actions:

For each zone:

- Computes the number of expected requests
- Subtracts the previous result from the number of mtaxies currently in the zones → *n*

- Zones with *n > 0* have a surplus of taxies

- Zones with *n < 0* have a lack of taxies

The algorithm starts moving taxies from the queue with the lower *n* to the queue with the higher *n*

Mtaxies Distribution Checker - 2

The algorithm terminates when one of the two following condition is satisfied:

- The expected demand of mtaxies has been fullfilled
- There are no more mtaxies to fullfill the expected demand

Mtaxies picker

Objective: Extract a mtaxi to fullfil a user ride request

The algorithm consists of two functions:

checkTaxi(zone, limit)

Returns the first mtaxi in the queue associated with the given zone that is in good traffic conditions and its position from the top of the queue is lower or equal than *limit*. If such a mtaxi does not exists -1 is returned

Mtaxies picker - 1

searchTaxi(zone, limit)

Checks if checkTaxi returns a mtaxi for a given zone and limit.

YES → exits and returns the found mtaxi
NO →searchTaxi invokes checkTaxi for the zones adjacent to the given one

YES → exits and returns the found mtaxi
NO → searchTaxi invokes checkTaxi for the zones adjacent at depth 2 to the given one

YES → exits and returns the found mtaxi
NO → exits and returns -1 meaning that no mtaxies is available to fullfill the ride request

Thank You!

Questions??