**4.A**

We have chosen to use a top-down approach, so we will start from a higher level of architectural design view and then we will analyse the system deeper.

We have chosen a four-tier architecture in order to keep a logical separation among data level, business level, communication level and client level.

|  |  |
| --- | --- |
| **Tier** | **Layer** |
| Client | Terminal/Browser |
| Web | Application Server |
| Business |
| Data | DBMS |

* **Client Tier** is the level used by the end-user and it is located in the terminal mobile app or in the web browser.
* **Web Tier** is the level that receives communication requests from client and communicates with “Business Tier” to formulate responses. It is located in the application server.
* **Business Tier** is the logical applicative level of the software. It analyse data received from “Data Tier” and it provides info’s to the “Web Tier”. It is located in the application server.
* **Data Tier** is the “Model” in the “Model-View-Control” pattern and it communicates with “Business Tier”. It contains all the software data and is located in a DBMS.

**4.B**

These are the main components that we have identified for each tier:

*Client Tier*

* **GUI**It’s the graphic user interface of the mobile application.
* **GPS Reader**

IT reads GPS information from the API’s of the operative system.

* **Messenger**

It sends and receives messages from Web Tier Central Messenger and handles every message and operation of the Client Tier, in fact it communicates with GPS Reader and GUI.

* ***Browser***It communicates with Web Tier Central Messenger and it represent the User Interface alternative to mobile app .

*Web Tier*

* **Central Messenger**

It sends and receives messages from Client Tier and it handles every message and operation among Business Tier components.

It also elaborates and sends pages to the Browser.

*Business Tier*

* **Sign Up Manager**It manages the registration function and it operates with “Taxi and Taxi driver Database MS” and “Passenger Database MS”.
* **Log In Manager**

It manages the Log In function and it operates with “Taxi and Taxi driver Database MS” and “Passenger Database MS.

* **Coverage Manager**

It controls and manages continuously the coverage of taxi in the city. It operates with “Taxi and Taxi driver Database MS” and “City Zones Database”.

* **Request Gatherer**

It collects the requests and reservations and it writes them in “Requests Database MS”.

* **Request Allocator**

It reads from “Requests Database MS” and it allocates requests to taxi drivers. So it also operates with “Taxi and Taxi driver Database MS” and “City Zones Database MS”.

* **Queue Handler**

It manages the allocation of taxi drivers in the correct city zone queue. It operates with “Taxi and Taxi driver Database MS” and “ City Zones Database MS”.

*Data Tier*

* **Requests Database MS**

It’s the database that contains all reservation and request information.

* **City Zones Database MS**

It’s the database that contains all zones information and queues.

* **Taxi and Taxi driver Database MS**

It’s the database that contains all taxi drivers information.

* **Passenger Database MS**

It’s the database that contains the registered passengers information.

**INSERIRE IMMAGINE (components’ interaction)**

**4.C**

For each component we identified these modules:

Client Tier

**Central Messenger**

- A module that interfaces with GUI

- A module that interfaces with GPS Reader

- A module that interfaces with Server Central Messenger

- A module that handles and forward messages throw modules.

**GPS Reader**

- A module that interfaces with Central Messenger in order to receive requests and send information

- A module that reads GPS information from Mobile API’s:

**GUI**

- A module that interfaces with Central Messenger in order to receive and send messages

- A module that shows navigation pages

Business Tier

**Sign Up Manager**

- A “Communication module” that interfaces with Central Messenger.

- An “Analysis module” that, reading form “Data module”, analyses the correctness of data and it eventually confirms or rejects.

- A “Data module” that stores data in the correct database and read needed information.

**Log In Manager**

- A “Communication module” that interfaces with Central Messenger.

- An “Analysis module” that, reading form “Data module”, analyses the correctness of data and it eventually confirms or rejects.

- A “Data module” that stores data in the correct database and read needed information

**Coverage Manager**

- A “Data module” that interfaces with databases making DBMS queries.

- An “Analysis module” that analyses information received from “Data module” and calculates the best coverage.

- A “Communication module” that interfaces with Central Messenger in order to send messages to Taxi Drivers selected by “Analysis module”.

**Request Gatherer**

- A “Communication module” that receives requests messages from Central Messenger.

- An “Analysis Module” that analyses data integrity and requests’ information received from “Communication module” and eventually it sends to “Data Module” .

- A “Data Module” that stores information interfacing with databases.

**Request Allocator**

- A “Data Module” that, periodically reading from databases, controls if there are requests to fulfil and updates taxi drivers’ information.

- An “Analysis module” that choses taxi diver to allocate to the requests.

- A “Communication module” that interfaces with Central Messenger in order to send notifications to taxi drivers.

**Queue Handler**

- A “Communication module” that receives availability messages from Central Messenger.

- A “Data module” that store and updates information in database.

**Central Messenger**

- A module that interfaces with Queue handler.

- A module that interfaces with Request Allocator.

- A module that interfaces with Request Gatherer.

- A module that interfaces with Coverage Manager.

- A module that interfaces with Sign Up/Log In manager.

- A module that interfaces with Client Central Messenger.

- A module that interfaces with Client Browser.

- A module that handles and forward messages throw modules.

4.E

We have produced some run-time sequence diagrams of the main operations that can be made with the software:

* New reservation
* Coverage update
* Taxi driver queue update
* Reservation allocation

INSERIRE 4 SEQUENCE DIAGRAMS

4.G

Some style attitudes:

* We designed an event based architecture throw an intense flow of messages among components. We choose this style because our components are very independent and in most of cases they interact asynchronously with each other.
* Our architecture is intrinsically client-server. In particular, “fat-server” (data tier, web tier, business tier) and “thin-client”. Moreover this architecture is confortable to be developed with JEE.

4.H

We have paid particular attention in designing a very modular architecture favoured by having chosen a top-down approach. This disposition has been selected in order to facilitate future extensions.

**7. Traceability**

To ensure requirements traceability we pointed out, for each of the requirements identified in RASD, which components are involved in the developing.

We omitted to write “Central messenger” because it’s involved in every functional requirement and we also omitted to write databases because we have already discus their strict connection with components of the “Business tier”.

1. The system must balance taxi drivers workload by assigning each one to  a first-in, first-out queue in a city zone 🡪 **Queue handler**
2. The system must provide a sign up function for passengers **🡪 Sign-Up Manager, GUI**
3. The system must store all required passenger information 🡪 **Sign-Up Manager**
4. The system must provide a log in function to access all passenger features 🡪 **Log-In manager, GUI**
5. The system must provide a function that allows logged passengers to re- quire a taxi 🡪 **Request Gatherer, GUI**
6. The system must provide a form in which the logged passenger will be able to add trip information for a request for service (starting point, destination point, number of passengers) 🡪 **Request Gatherer, GUI**
7. The system must retrieve GPS information from the logged passenger’s mobile phone 🡪 **GPS Reader**
8. The system must provide a reservation function  🡪 **Request Gatherer, GUI**
9. The system must provide a form in which the logged passenger will be able to add trip information for a reservation (starting point, destination point, number of passengers, leaving time)  🡪 **Request Gatherer, GUI**
10. The system must finalise a reservation two hours before its requested time, making it unchangeable 🡪 **Request Allocator**
11. The system must store a reservation history for each passenger 🡪 **Request Gatherer**
12. The system must provide a function which shows reservation history  **🡪 Request Gatherer, GUI**
13. The system must provide a function to allow logged passengers to modify a reservation up to two hours before the requested time 🡪 **Request Gatherer, GUI**
14. The system must provide a function to allow logged passengers to delete a reservation up to two hours before the requested time 🡪 **Request Gatherer, GUI**
15. The system must analyse a newly-made real time request and send a confirmation with the estimated waiting time if it can be fulfilled 🡪 **Request Gatherer, GUI**
16. The system must analyse a newly-made reservation request and send a confirmation if it can be fulfilled 🡪 **Request Gatherer**
17. The system must notify passenger if his reservation cannot be met 🡪 **Request Allocator**  **, GUI**
18. The system must delete all reservations in a passenger’s history that can- not be met after sending him notice 🡪 **Request Gatherer**
19. The system must provide a sign up function for taxi drivers **🡪 Sign-Up Manager**  **, GUI**
20. The system must store all required taxi driver information **🡪 Sign-Up Manager**
21. The system must check that all taxi drivers trying to sign up own both a valid taxi driver’s license and a proper driver’s license **🡪 Sign-Up Manager**
22. The system must check that the information needed for a taxi driver sign up points to a specific person in the company that has not already registered before confirming **🡪 Sign-Up Manager**
23. The system has to provide a log in function to access all taxi drivers features 🡪 **Log-In manager, GUI**
24. The system must provide a function to allow logged taxi drivers to inform the system of their availability 🡪 **Queue Handler, GUI**
25. The system must notify taxi drivers of an incoming request for service 🡪 **Request Allocator**  **, GUI**
26. The system must provide all necessary information for taxi drivers to carry out requests 🡪 **Request Allocator**  **, GUI**
27. The system must provide a function to allow taxi drivers to confirm that they are going to take care of an assigned request for service 🡪 **Request Allocator**   **, GUI**
28. The system must provide an explicit decline function to allow taxi drivers to notify that they are not going to take care of an assigned request for service 🡪 **Request Allocator**   **, GUI**
29. The system must flag a taxi driver as having declined an assignment if no answer is received within 30 seconds 🡪 **Request Allocator**
30. The system must move a taxi driver that has declined an assignment at the end of their queue 🡪 **Queue Handler**
31. The system must analyse taxi locations and calculate their best possible distribution 🡪 **Coverage Manager**
32. The system must choose which taxi drivers need to be moved to ensure total coverage of the city 🡪 **Coverage Manager**
33. The system must notify taxi drivers in which city zone they have to move, as needed 🡪 **Coverage Manager, GUI**