Architectural design

# Overview

Architectural design is of crucial importance in software engineering because it will have to take account of functional and non-functional requirements, to meet the stakeholders needs and requests and to help not to focus only on standalone elements losing the so called big picture of the system, always adhering to general principles of good quality. An important aspect is in fact to find a good trade-off between the high-level description near to the analysis and the low-level one near to the implementation.

Coming up with good quality design and architecture is mostly a matter of experience and in our field, is also known the importance of the reusability of other’s people work. So, we tried to build our system with various kind of this patterns and known architectural styles.

# Selected architectural styles

Architectural style: Client-Server

It’s the best known and most used architectural style for distributed applications.

We have a well-defined distinction between client and server, they play different roles and also, they are both accessible with a precise interface.

Travlendar+ is a mobile application and will have multiple mobile users, but still the computations will be located in some point where the global view of the system can be seen.

The system must guarantee scalability so, resources in the form of network segments, computers and servers must be added to the network without major interruptions of it.

Said that, we agreed that the best solution for our needs could be the Client-Server style.

* Server is invoked to provide one or a set of services, in our case, for instance, the computation of the best mobility option for the user according to all of his characteristics.
* Clients uses the provided services and initiate the communication through messages (JMS) or remote invocations. they interact directly with end-users using any user interface such as graphical user interface. These are the user logged in to the application.

In the end, we will talk about web clients, which means that in our scenario, they will ask the server to provide services for them and they will not store data locally. Furthermore, the architecture is OS independent, it relies on a central server that will avoid consistency issues between different devices of the same client.

Three-tier architecture

The Client-Server model does not impose any constraint neither about how logical layers (presentation, application/business logic, data) have to be distributed among the deployment units nor about the number of tiers (physical deployment units) has to be designed.

We taught that the best decision for us could have been the three-tier architecture, that’s composed of:

* First tier: Presentation layer (NON C’E’ PROPRIO NESSUNA LOGICA NEL CLIENT?)
* Second tier: Application/business layer. Process and executes both new requests of the clients and possible variations on older and not yet completed requests. It will collect information from the users and store them into the databases, but also it will provide to them answers for their requests. This tier will do both the visualisation (web) and the processing (business) portion.
* Third tier: Data layer. Dedicated to the storage of information in persistent memory.

This architecture permits us to achieve one of the design principles, that says to decouple where possible, in fact we have a solid distinction between logic and data, and also between logic and presentation.

Basic architectural pattern: Stateless components

…