1. **System Vision**

*Business context*

The system aims to ease the search of a car sale for Republic of Moldova. There are no other systems alike on the moldavian market.

*Business opportunities*

Searching for a car for sale on the internet may take quite a lot of time. First a buyer needs a list of web sources, then comes the searching and of course data filtering. These are time consuming actions. Our solution will free the user from all of these actions.

*Problem*

At the moment there is no centralized service for searching a car for sale.

*Concerns*

Cars byers; people interested in car market prices

*Consequences*

A lot of time is lost; some very rich web sources may be skipped

*Successful solution*

Quick and effective searches, a rich centralized database

*Task of the system*

Centralized ads from multiple web sources

1. **Stakeholders**

|  |  |  |
| --- | --- | --- |
| Stakeholder | Role | Description |
| Project Manager | Represents the top managers | Analyzes performance, development schedule and dictates high level software functionality |
| Team Leaders | Represents the architecture rules. | Decides the system architecture and programming languages, modules and protocols |
| Developers | Represents the system specific details | Develop system functionality |
| Server | Represents the requirements of the server maintenance | Assures maintenance and solves possible errors |
| Supervisor | Represents the requirements of the server maintenance | Assures maintenance and solves possible errors |
| Buyer | Represents the requirements of the potential clients. | Evaluates the correspondence between the requirements and the actual system. |

1. **Functional Requirements**

*Search*

Users should be able to search a car by its brand name and model.

*Filter responses*

Users should be able to add filters or search using filters.

*Filter web sources*

Users should be able to see the web sources list, disable some of them or porpose new sources.

*View details*

Users should be able to view details about the car for sale.

*View advanced details*

Users should be able to view all the details about a car model from some official sources.

*View history*

Users should be able to see a history of his searches.

Third Party Requirements:

*Provide Data*

Web sources should not change the way they are providing data.

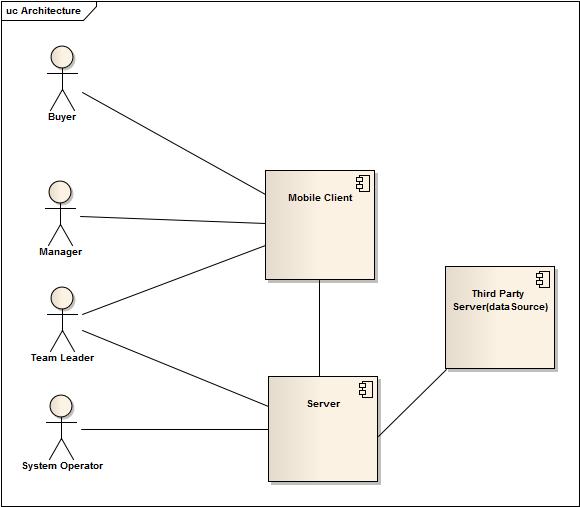
*Run non-stop*

Web sources should work properly and non-stop.

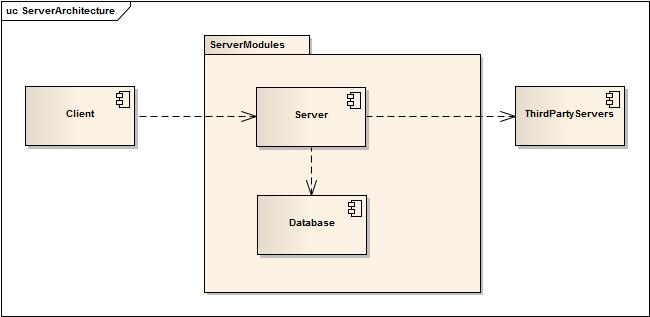
1. **Non-functional Requirements**

|  |  |
| --- | --- |
| Non-functional requirement | Description |
| Extensibility | New data source can be added to the system. |
| Responsiveness | A search must be resolved in less than 1 second |
| Time To Market | 3 months at most. |
| Interoperability | The client connection and querry protocols must be universal for iOS and Android and other platforms to be able to connect. |
| Usability | The system must react to inputs in less than 250ms , either with a real or a mocked answer thus removing the feeling of non-responsiveness. |
| Accessibility | The system must be online 24/7. |

1. **Architectural sketch**



1. **Functional blocks**



1. **Architectural Choice**

Our application architecture compliant with the following principles:

1.Layered architecture – this architecture separates the responsibilities within the system

Client/Server – we have a big advantage even over the car selling websites. The user doesn’t have to navigate through the site in order to find a car, which means more traffic used, more money spent. The server will do all the searching, filtering and will just send the answer to the user.

Thin client – the client will be a thin application that will not spend user’s phone battery for filtering, searching or data analyzing. The server does all the hard work.

SOA – our system is not bounded to iOS or Android devices. Knowing the protocol, you can make a request from whatever platform, using whatever programming language.

n-tier architecture – The client application doesn’t have direct access to the database, everything goes through a server which will make any additional analyzes and just after that will send a request to the database.

1. **Architectural Principles**

The sketches above are very general and we can say nothing about complying or not with some principles. Although we can analyze the system

* Loose coupling – in our system every module is coupled to the others as weak as it possible
* High Cohesion – as it is noted on the previous principle, the modules aims to be as weak coupled as possible
* Design for change – our system tries as much as possible to be designed for change
* Separation of concerns – every module has its own responsibility, it relies only on the others modules result, not on the internal structure
* Information hiding – the client application doesn’t have a direct access to the database. It gets only the information it needs from the server.
* Liskov substitution – the system doesn’t have some subtypes that could substitute the basetypes.
* Interface segregation – the client application provides the same view for all the users
* Modularity – as it was noticed above, every module has its own responsibility
* Convention over configuration – the system tries to be standardized, but the main goal is effectiveness.

1. **Architectural Sketch**

