portada

indice

*Description of the implementation strategies, problems and Solutions*

* *Structure:*

The project has been divided into different packages that contain all the classes and resources needed by the application.

The packages we have are: domain, GUI, domain and DB inside of src Package, and data package that contains more packages inside. We are just going to have a look to the different parts.

* Domain

This Packaged contains all the classes related to the Business layer of our application, it means, the main functionality.

This Project consists on client/Server architecture, so we decided to divide this actual Packaged into other three:

* + Server: contains the classes of the daemon servers and the request servers
  + Util: it contains the class “SocketManager” used in the session’s lab, which are going to help us to communicate the client with the servers.
* DB

In this Packaged the only one class we can find is the database manager that is going to deal with every aspect related to the database.

* GUI:

It contains all the classes of the windows of the application, which are in fact the client of it.

* *Implementation*

Talking about the implementation, just begin with the servers:

The application has been implemented with two servers. The aim of the first server, called RequestServer, is to response to the request of the clients, it means, it’s going to provide answers to all the commands coming from the clients. On the other hand, the purpose of the second Server, GridRequestServer, is to provide only one service, although it has four commands.

The service this Server is going to provide is to obtain GPS coordinates of a picture that has been taken in the client, when the GPS system is switched off.

In addition to this two servers, other two has been implemented too, the daemon servers, whose purpose is to create different child servers, which are going to offer service to the clients.

The way they have been implemented is simple; they follow a state transition diagram (which will be represented in next section). Every time the server receives a request from the client, analyses the command, splitting the request line into different strings, to do an action or another. **//apartir de aqui digamos que es en sucio**

Furthermore, it has been said that there could be as many clients as we want, that’s why, a Server deamon has been implemented for each Server which attends the requests of the clients. This way, when a client ask for a service, the deamon Server catch that petition, but instead of process itself the request, it delegates in a “request Server” which is going to provide a response to the client. Summing up the aim of the deamon servers is to create “subservers” in order to attend the clients’ petitions.

Data:

As almost every application, ours works with data which could be stored in a text file, in an xml file or just in a database. We have decided to use an Access database, to stored and manipulate all the information referred to aour application: vehicles, sensors, measurements… andall the relation hmong them.

This database is going to be used exclusive by the servers, clients are never going to know that there’s a database. It has been decided this way, because if we talk about a client/Server architecture, it doesn’t have sense that the client has access to the information it’s asking for to the Server. However, in our application we have followed this approach in part just to simplify things. It will be explained later in problems section.

Moving to the field of the database implementation, we decided to create a class whichdélas with all aspects of the information stored in it. This class is called DBM (Data Base Management). It has typical methods such as connect (constructor) and disconnect, and all the methods needed by the Server to obtain information.

*State transition diagram requestServer*

CONNECT ip

[ip OK]

PASS password

[password OK]

USER username

[username OK]

USER username

[usernameWrong]

USER

[no username]

PASS

[No password]

PASS password

[passwordwrong]

CONNECT

[no ip]

CONNECT ip

Iipwrong]

LISTSENSOR

GPSON, GPSOFF

GET\_PIC

HISTORYLOG sensorId

ON sensorId

OFF sensroId

GET\_CURVALUEsensorId

RETURN

GET\_LOC

QUIT

*State transition diagram requestGridServer*

PASS password

[password OK]

USER username

[username OK]

USER username

[usernameWrong]

USER

[no username]

PASS

[No password]

PASS password

[passwordwrong]

GET\_COOR

QUIT

*User’s manual*

* Set-up process

In this section we will explain the steps any user has to follow, in order to deploy this application using the integrated development environment (IDE) “Eclipse”.

As we have mention in a previous section, we have two server and a client that can be launch as many times as we want to create different clients, so if we want to run the application, we have to launch first the servers, and finally the client.

These are the step to be taken:

1. Launch the GridServer class. This class can be found in the package: src/domain/server/GridServer.java
2. Launch the other server, CarServer which is in the route: src/domain/server/CarServer.java
3. Launch the client class. This class is in the GUI package, because in deed the client is an interface. The class that we must run is ClientGUI and it’s located in the following path: src/GUI/ClientGUI.java

* Screenshots

AQUI LOS PANTALLAZOS ITZI

* Communication log (client-requestServer-requestGridServer)
* **Client:** USER Itzi
  + **Server:** 201 OK Welcome Itzi
* **Client:** PASS 12345
  + **Server:** 202 OK Welcome to the system
* **Client:** CONNECT 127.0.0.1
  + **Server:** 214 OK Connection with the server established
* **Client:** LISTSENSOR
  + **Server:** 112 OK start of sensor list… 212 OK end of sensor list
* **Client:** ON 2
  + **Server:** 418 ERROR Sensor already activated
* **Client:** HISTORYLOG 3
  + **Server:** 113 OK Start of measurement list Ups, there aren't measurements for this sensor
* **Client:** GET\_CURVALUE 2
  + **Server:** 114 OK 10/4/2010; 19:49:51; 14º 34' 56.89''- 45º 56' 34.89''; 76%
* **Client:** RETURN
  + **Server:** Coming back to the main menu...
* **Client:** GPSOFF
  + **Server:** 420 ERROR GPS already deactivated
* **Client:** GET\_PIC
  + **Server:** 206 OK Loading image [96997 bytes]… photo transmitted
* **Client:** GET\_LOC
  + **Server:** USER Itzi
    - **Server2:** 201 OK Welcome Itzi
  + **Server:** PASS 12345
    - **Server2:** 202 OK Welcome to the system
  + **Server:** GET\_COOR 1
    - **Server2:** 114 Ok 14º 34' 56.89''- 45º 56' 34.89''
  + **Server:** QUIT
    - **Server2:** 222 Ok Bye
  + **Server:** 114 Ok 14º 34' 56.89''- 45º 56' 34.89''
* **Client:** QUIT
  + **Server:** 208 OK Bye

**Black:** requests from client to requestServer

**Red:** answer from requestServer to client

**Blue:** request from requestServer to requestGridServer

**Green:** answer from requestGridServer to requestServer