**SELECTED ARCHITECTURAL STYLES AND PATTERNS – CLIENTS AND SERVERS pt1**

The Client-Server style is used for all the requests done by the various

clients connected to the Web Server of PowerEnjoy. The User

Application can use a standardized

Client-Server protocol via HTTPS that follows the principle of a

RESTful Service. The Administrator application it's connected via RPC

to the Application Server and can perform more critical requests.

R

**SELECTED ARCHITECTURAL STYLES AND PATTERNS – CLIENTS AND SERVERS pt2**

A TCP/ IP protocol is used to established connections between

the Car System, the DB Server and the Application Server.

The other requests inherent to the GMaps and the PayPal servers are managed with a HTTPS Protocol

**SELECTED ARCHITECTURAL STYLES – CONCLUSION**

The Client-Server style and Point to Point bidirectional messaging

system are used to implement properly the MVC pattern in this three

Layers, three Tiers system. ???

**ALGORITHMS**

What will follow are slides containing algorithms (written in pseudocode) that explain how

The final payment works and how to find a car in a certain zone.

**BILL ALGORITHM**

Below is represented the \textbf{Bill' algorithm}. Once the driver has stopped the car and exits the car, the system starts checking the state of sensors, the position of the car towards the position of the nearest safe area and last but not least the state of the battery ( the driver has 5 minutes to eventually charge the battery and receive the discount).The events generated and their consequences are discussed in the following table.

\textbf{[Legend]}

\begin{itemize}

\item D: Driver;

\item S: System;

\item C: Car;

\item B: Battery;

\item LoP: List of passengers;

\item SA: Safe area;

\end{itemize}

\algoTab{

D exits C &

\begin{algorithmic}

$S.startChecking()$

\end{algorithmic}\\

\hline

Check the distance between the SA and the current position &

\begin{algorithmic}

\If {$sA.nearest()-D.currPos()\geq 3$}

\State $D.applyTax()$

\Else

\If {$i+k\leq maxval$}

\State $D.applyDiscount()$

\EndIf

\EndIf

\end{algorithmic}\\

\hline

Check the number of passengers &

\begin{algorithmic}

\If {$LoP.size()\geq 2$}

\State $D.applyDiscount()$

\EndIf

\end{algorithmic}\\

\hline

Check the battery state &

\begin{algorithmic}

\If {$B.getState()\leq 20$}

\State $D.applyTax()$

\EndIf

\If {$B.getState()\geq 50$}

\State $D.applyDiscount()$

\EndIf

\end{algorithmic}\\

\hline

D ends the rent &

\begin{algorithmic}

\State $C.status \gets Ready$

\end{algorithmic}\\

\hline

D has 5 minutes to charge the car and take a discount &

\begin{algorithmic}

\State $oldState \gets B.getState()$

\State $wait(5)$

\If {$B.getState()\geq oldState$}

\State $D.applyDiscount()$

\EndIf

\end{algorithmic}\\

**GEOLOCATION’S ALGORITHM – EXPLANATION**

A less heavy weight solution has been found: this solution expects every

Zone of the City to be divided in several convex Polygons, for instance

Triangles, that have interesting properties for our application. In

PowerEnJoy, Zones of regular shape are intended to be designed, and

therefore the number of Triangles in which a Zone should be decomposed

is very limited. So, such a ow is followed:

1 Obtain GPS Data from the selected Car

2 For each Zone, check if the the Point that the Longitude and

Latitude from GPS Data identify is contained inside any Triangle in

which the Zone is divided. If it is so, then the Zone is found. If that's

not the case, then another Zone could contain the given Point. If no

Zone contains the Point, then we can assume that the Point refers

to GPS Data that identify a geographical point outside of the City.

The computation of the Point in Triangle test is simple and e\_cient (e.g.

using barycentric coordinates).

**GEOLOCATION’S ALGORITHM - CODE**

\subsection{Geolocation' Algorithm}

Below is represented the \textbf{Geolocation' algorithm}. Let's begin with the premise that we are imagine to build the algorithm with an object oriented Language and we're providing a pseudo-code. This algorithm checks if the given point is inside this Triangle. Infact thanks to a theorem about convex polygons, we can check if a point P is inside a given convex polygon (i.e. if the given vector associated to the point P is a convex combination of the polygon vertices). We can calculate if such coefficients exists solving a vector equation:

\begin{itemize}

\item P = \mathrm{d}x \* P1 + \mathrm{d}y \* P2 + \mathrm{d}z \* P3.

\item P = \mathrm{d}x\*P1+ \mathrm{d}y\*P2+(1 - \mathrm{d}x - \mathrm{d}y )\*P3.

\item P - P3 = \mathrm{d}x\*(P1 - P3)+\mathrm{d}y\*(P2 - P3).

\end{itemize}

This equation can then be split into two scalar linear equations in the x and y components. The system is solved using Cramer's rule and then it is checked that alpha1 and alpha2 (and alpha3) found by solving the system satisfy the constraints.

\textbf{[Legend]}

\begin{itemize}

\item P: class Point;

\item T: class Triangle;

\item Z: class Zone;

\item A: class Area;

\item a,b,c,d,e,f: Double(or Float) values;

\end{itemize}

\algoTab{

Declare variables that will allow to solve the linear equation system thanks to Cramer's method &

\begin{algorithmic}

\State $a\gets p1.getX()-p3.getX()$

\State $b\gets p2.getX()-p3.getX()$

\State $c\gets p1.getY()-p3.getY()$

\State $d\gets p2.getY()-p3.getY()$

\State $e\gets p.getX()-p3.getX()$

\State $f\gets p.getX()-p3.getY()$

\end{algorithmic}\\

\hline

Calculate the determinant to check the solution of the system &

\begin{algorithmic}

\State $tContains()\*$

\State $d\gets a\*d-b\*c$

\If {$d == 0$}

\State \Return $false$

\EndIf

\State $\mathrm{d}Px \gets e\*d-f\*b$

\State $\mathrm{d}Py \gets a\*f-c\*e$

\State $\mathrm{d}Z \gets 1- \mathrm{d}Px - \mathrm{d}Py$

\State $\mathrm{d}X \gets \mathrm{d}Px /d$

\State $\mathrm{d}Y \gets \mathrm{d}Py /d$

\end{algorithmic}\\

\hline

Check the results and draw conclusion &

\begin{algorithmic}

\If {$\mathrm{d}x\leq 0 || \mathrm{d}y\leq0 || \mathrm{d}z\leq0$}

\State \Return $false$

\EndIf

\State \Return $true$

\end{algorithmic}\\

\hline

Instance the Zone Class and set the Zone as a set of Triangles. Then check if a point (in our case our position) is contained in the triangle &

\begin{algorithmic}

\State $zContains()$

\ForAll {Triangle t : triangles}

\If {t.tContains(p)}

\State \Return $true$

\EndIf

\State \Return $false$

\EndFor

\end{algorithmic}\\

\hline

Instance the Area Class and set the Area as a set of Zone. Then check if a point is contained in the zone. This allows us to determine in which zone is our point &

\begin{algorithmic}

\State $Point(latitude,longitude)$

\ForAll { Zone z : zones}

\If {z.zContains(p)}

\State \Return $z$

\EndIf

\State \Return $null$

\EndFor

\end{algorithmic}\\

**GUI DESIGN**

In this section we provide the most important and meaningful mockups

for every class of screens we have designed. In particular we identified

three classes of graphical user interfaces:

.car info

.reservation

.end rent

**MOCKUPS**

In the following slides are shown sequence of graphical states that the

application has to render in order to perform a resevation.

The first one car info interface shows the status of a car, its position and

Other parameters. When a User clicks on BOOK button he reserves the car.

The Reservation mockup shows instead the status of the Reservation and gives the possibility

Of delete the reservation or to check the remaining time.

The last screen shows the end of a rent, screen that appears when a User has exit and close the car and at least 5 minutes are elapsed