SafeStreets
DD document

Dario Miceli Pranio
Pierriccardo Olivieri

Academic year: 2019 - 2020

POLITECNICO
MILANO 1863

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to provide a functional description of *SafeStreets* application.

1.2 Scope

SafeStreets is a service that aims to provide *Users* with the possibility to notify *Authorities* when traffic violations occur, and in particular parking violations. The application's goal is achieved by allowing *Users* to share photo, position, date, time and type of violation and by enabling *Authorities* to request them.

SafeStreets requires the *Users* to create an account to access its services, the functionalities unlocked after registration depend on the type of account created.
If a *User* creates an account as *Citizen*, he/she must provide name, surname and a fiscal code in order to prove that he/she is a real person. Furthermore, he must provide an email with which he will be uniquely identified and a password. Once the account has been activated, *User* can finally start to report parking violations and can also see statistics of the streets or the areas with the highest frequency of violations.

On the other hand, an officer will create an account as *Authority* and he will need to provide his name, surname, work's Matricola, a password and as for *Citizen*, will be uniquely identified by an email. Once the Matricola has been verified and the account has been activated, the officer can retrieve the potential parking violations sent by *Citizen* that have not been taken into account yet by other officers, analyze them and, if it is the right case, generates traffic tickets. *Authorities*, can see the same statistics of the *Citizen* and can also see statistics about vehicles' license plate that commit the most violations.

1.3 Definitions, acronyms, abbreviations

1.3.1 Definitions

- *Users*: can be either *Citizen* or *Authority*
- *traffic violation*: generic violation that can occur in a street
- *parking violation*: a violation caused by a bad parking
- *violation*: general violation, identity both traffic or parking violation
- *unsafe areas*: areas with an high rate of violations

1.3.2 Acronyms

Table with all acronyms used in document.

ACRONYM	COMPLETE NAME
DD	Design Document
GPS	Global Positioning Systems
S2B	Software To Be
GDPR	General Data Protection Regulation
FC	Fiscal Code
UC	Use Case

1.3.3 Abbreviations

- **Rn**: n-th Requirement

1.4 Revision History

1.5 Reference documents

- ISO/IEC/IEEE 29148: https://www.iso.org/standard/45171.html
- Specification Document: "SafeStreets Mandatory Project Assignement"

1.6 Document Structure

2 Architectural Design

2.1 Overview: High level components and their interaction

In this graphical representation of *SafeStreets* we describe at an high level the main interaction between the components that are involved. Focusing on client side *SafeStreets* provides a Client application, in this view case we don't make a distinciton between the *Users* type, this is primarily devoted to the fact that the requests are similar. We identify then a 3-tier architecture, composed by the client, a proxy in the middle to manage properly all the requests and finally a back end part in which the *System* store the information (in a DBMS) and generate statistics thanks to the data submitted by the *Citizen* and confirmed by *Authorities*. In order to do this we need an application server. Since we will authenticate the *Users* through a confirmation email, and we give the possibility to the *Authority* to receive data via mail, the *System* needs also a Mailing System. For notify the *Users* with some relevant informations *System* will use a notification System. Finally we also need some service in the client like GPS Service and Maps Service.

2.2 Component view

***SafeStreets* Client Application**

***SafeStreets* Business Logic**

This component describes core logic of *SafeStreets* application. It is a stateless module that lies between the Client application and the central Database. Now we will describe each component:

- **Network Manager**
  It handles all incoming messages exchanged between Clients and Server. It does two foundamental things; it sends all incoming messages from client application to the correct handler in Business Logic and on the contrary it collects all the outgoing messages and sends to the corresponding Client Application.

- **Authentication Manager**
  The Authentication Manager exposes all methods related to the access to the platform. It handles the phase of *User* registration and login and in order to do this it has to continuosly interact with the database through the Data Storage Interface. It also checks all the costraints in order to garantee creation of correct account. Furthermore it sends email to all the *Users* that have been registered by accessing the Mailing *System*'s Interface.

- **Citizen Manager**
  The *Citizen* manager handles all functionalities related to a *Citizen* Account. In order to do this it has to continuosly interact with database through the Data Storage Interface. It also allows *Citizen* to update his setting.

- **Authority Manager**
  The *Authority* manager handles all functionalities related to an *Authority* Account. In order to do this it has to continuosly interact with database through the Data Storage Interface. It also allows *Authority* to update his setting.

- **Data Manager**
  This component is the main gateway between the Client Applications and the

central Database. It handles the import of new data associated to a *User*'s account and its retrieval.

- **Send Report Manager**
  The Send Report Manager handles the creation and the management of a Send Report performed by *Citizen* and directed to *System*. Inserting is possible thanks to Data Storage Manager. Furthermore, in order to avoid alteration, the Report is encrypted by using Privacy Manager.

- **Retrieve Report Manager**
  The Retrieve Report Manager handles the management of a Retrieve Report performed by *Authority*. Retrieving is possible by querying the central Database through the Data Storage Interface.

- **Statistics Manager**
  The Statistics Manager handles the management of a Retrieve Statistics performed by *User*. Retrieving is possible by querying the central Database through the Data Storage Interface.

- **Privacy Manager**
  The Privacy Manager exposes methods to encrypt sensitive data. This operation is important in order to avoid data leaks and data alteration.

- **Data Storage Manager**
  The Data Storage Manager provides all methods to interact with the central Database such as data retrieval, storage and update.

External interfaces
Some of the described components in our *System* are also dedicated to communicating with external services through specific interfaces. It is essential that the communication works properly in order to fulfill application's functionalities.

2.3 Deployement view

The architecture choosed for *SafeStreets* is a multi-tier architecture. Below all the nodes involved will be described.

Smarthphone
Represents the *User's* smarthphone in which the client will run. This is the client machine that will run *SafeStreets* application.

Firewall
(senti dario per parti inerenti alla sicurezza)Necessary to provide protection between local network and world network, so not-allowed third parties will not be able to access data.

Application Server Is the central unit of *SafeStreet* all the other components refer to this. It contains all the logic and provide bidirectional access to Database i.e. manages data acquisition and requests. Since for *SafeStreet* we need to focus on security, is a though decision to have only one main unit.

Proxy *System* will use a proxy to receive requests. This solution is more scalable for the eventuality of further improvements and guarantee a better stability of the *System*.

Database
A Database is necessary in order to store all the informations about personal data, registration and data submitted by *Citizen* and retrieve information in order, for instance, to build statistics.