Politehnica University of Bucharest

Computer Science and Engineering Department

**DynamicLib**

**Eftenoiu Alina**

**Focsa Oana-Catalina**

**Iosif Paula-Alexandra**

**SSA**

Bucharest

2014

**Contents**

[1.](#_Toc256000000) **[Duration in days](#_Toc256000000)** [2](#_Toc256000000)

[2.](#_Toc256000001) **[Keywords](#_Toc256000001)** [3](#_Toc256000001)

[3.](#_Toc256000002) **[Project summary](#_Toc256000002)** [3](#_Toc256000002)

[3.1](#_Toc256000003) **[S](#_Toc256000003)**[erver activities diagram 4](#_Toc256000003)

[3.2](#_Toc256000004) **[Client activities diagram](#_Toc256000004)** [4](#_Toc256000004)

[3.3](#_Toc256000005) **[Location service activities diagram](#_Toc256000005)** [5](#_Toc256000005)

[4.](#_Toc256000006) **[Project concept and objectives](#_Toc256000006)** [5](#_Toc256000006)

[5.](#_Toc256000007) **[Work plan](#_Toc256000007)** [5](#_Toc256000007)

[5.1](#_Toc256000008) **[Work Breakdown Structure](#_Toc256000008)** [5](#_Toc256000008)

[5.2](#_Toc256000009) **[Gantt chart at the level of tasks](#_Toc256000009)** [5](#_Toc256000009)

[5.3](#_Toc256000010) **[Work package list](#_Toc256000010)** [5](#_Toc256000010)

[5.4](#_Toc256000011) **[Deliverables list](#_Toc256000011)** [6](#_Toc256000011)

[5.5](#_Toc256000012) **[Description of each work package](#_Toc256000012)** [6](#_Toc256000012)

[5.5.1](#_Toc256000013) ***[Work package no1: Plan](#_Toc256000013)*** [6](#_Toc256000013)

[5.5.2](#_Toc256000014) ***[Work package no2: Analyze](#_Toc256000014)*** [6](#_Toc256000014)

[5.5.3](#_Toc256000015) ***[Work package no3: Design](#_Toc256000015)*** [7](#_Toc256000015)

[5.5.4](#_Toc256000016) ***[Work package no4: Code](#_Toc256000016)*** [7](#_Toc256000016)

[5.5.5](#_Toc256000017) ***[Work package no5: Test](#_Toc256000017)*** [8](#_Toc256000017)

[6.](#_Toc256000018) **[Risk management](#_Toc256000018)** [8](#_Toc256000018)

[6.1](#_Toc256000019) **[Risk identification](#_Toc256000019)** [8](#_Toc256000019)

[6.2](#_Toc256000020) **[Probability / impact evaluation](#_Toc256000020)** [9](#_Toc256000020)

[6.3](#_Toc256000021) **[Responses to risks](#_Toc256000021)** [9](#_Toc256000021)

# **Duration in days**

The project had a duration of **20 days**.

# **Keywords**

ADT – Android Developer Tools

API – Application Programming Interface

IDE – Integrated Development Environment

PHP - (Personal Home Page) *PHP: Hypertext Preprocessor*

REST - REpresentational State Transfer

SDK – Software Development Kit

URL – Uniform Resource Locator

XML - Extensible Markup Language

# **Project summary**

The project consists of implementing a dynamic library in order to find the nearest one which has the book requested by the user. The search is looking in Humanitas, Carturesti and Diverta libraries.

The server is a PHP application which authenticates and registers a user with a specific password and username , searches a book name and then stores a particular history for each user search and a global book rating into a database.

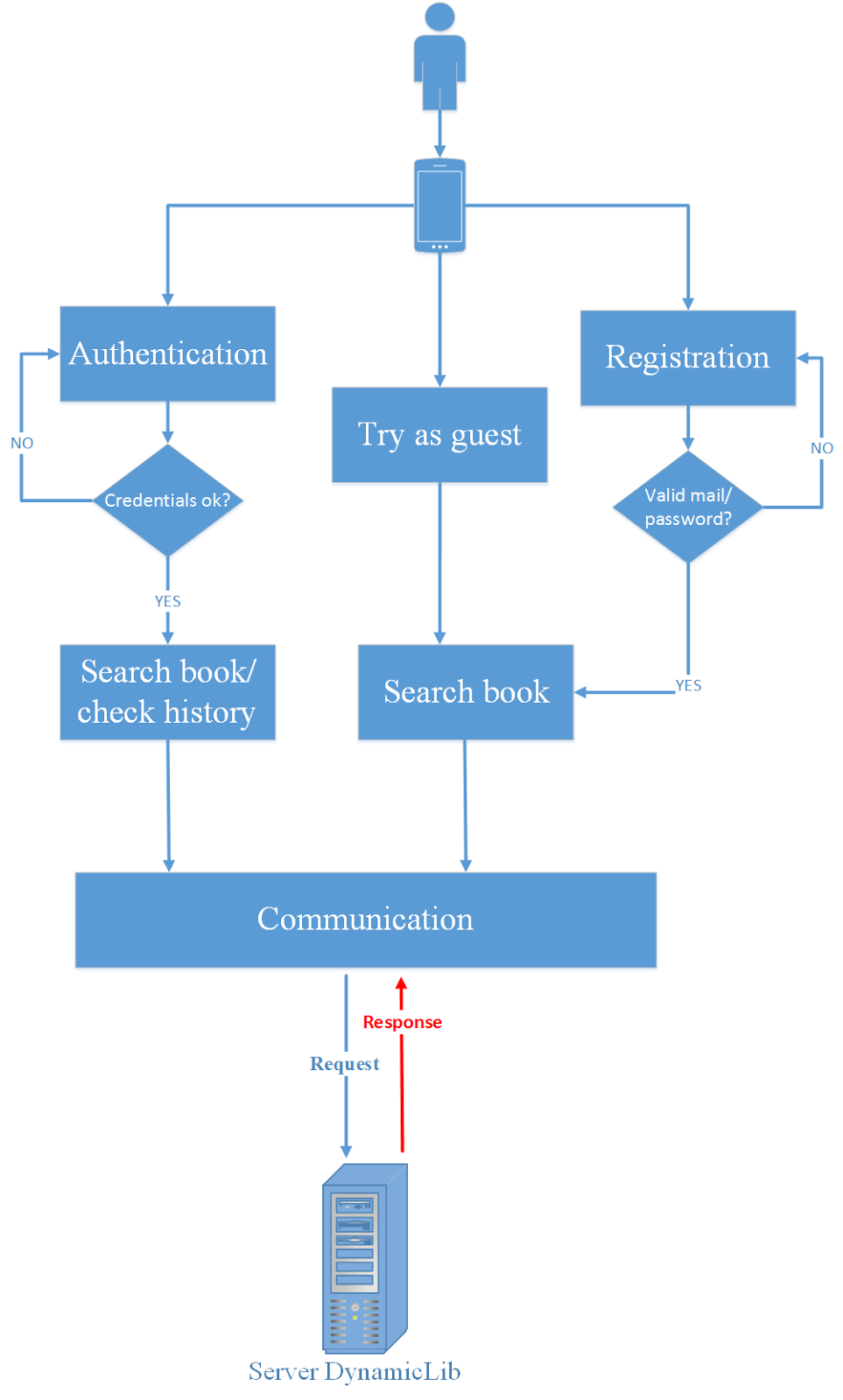
The client is an Android application written in Android 4.1.2, SDK revision 16. It has a simple and friendly interface and consists of two parts. The first part is the communication with the server and the second one is getting user location and calculate nearest library.

The user can easily login using an username and a password or create a free account by introducing the credentials requested: username, password, email. The user also has the option to try out the application as a guest. If the user is logged in, he also has access to a list of previous searches. After he selects one of these options, the user will encounter a new page where, by simply introducing the name of the book he wants to search, will receive informations about the book in every of the 3 libraries: availability, price and rating. Based on these informations, the user will decide which book suits him the best and locate it.

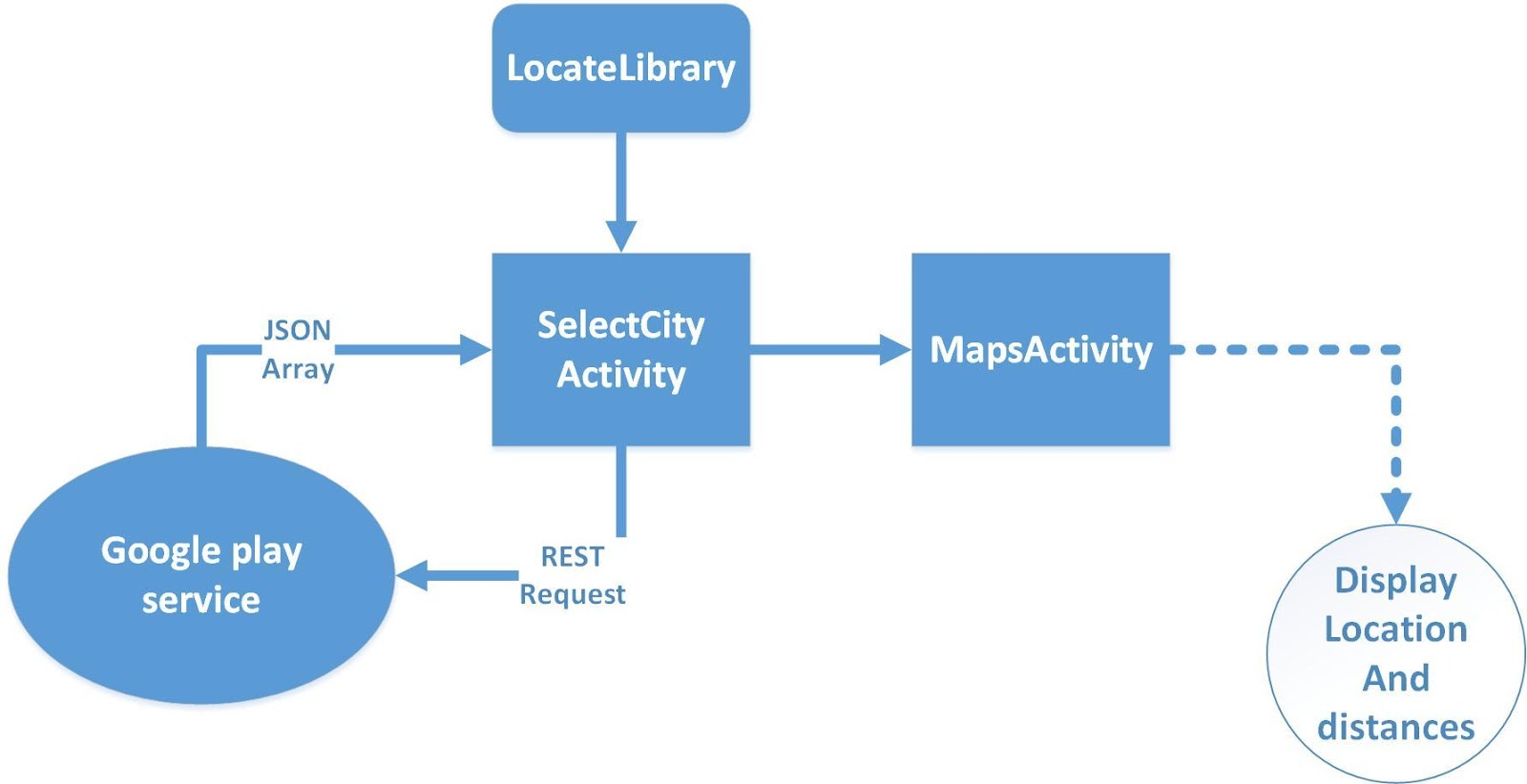
When the user wants to find the nearest library, a list of towns where the libraries are available appears. The user taps on a town and a search on Google Maps is made for all the three libraries. The search is transparent to the user and looks like this: „***Carturesti, Bucuresti***”. The user only receives the nearest library for each bookstore along with the address and distance.

## **S**erver activities diagram

## **Client activities diagram**



## **Location service activities diagram**



# **Project concept and objectives**

The objective of this project is to help users find the books they are searching for fast and easy. In this way, they will save precious time and effort.

# **Work plan**

## **Work Breakdown Structure**

## **Gantt chart at the level of tasks**

[server]

## **Work package list**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **WORK PACKAGE NO.** | **WORK PACKAGE TITLE** | **RESPONSIBLE PERSON** | **PERSON DAYS** | **START DATE** | **END DATE** |
| 1. | Plan | All team | 1 | 18 nov 2014 | 18 nov 2014 |
| 2. | Analyze | Alina  Oana  Paula | 3  5  2 | 20 nov 2014  19 nov 2014  19 nov 2014 | 23 nov 2014  23 nov 2014  20 nov 2014 |
| 3. | Design | Alina  Oana  Paula | 7  6  4 | 24 nov  25 nov  21 nov 2014 | 1 dec 2014  1 dec 2014  24 nov 2014 |
| 4 | Cod | Alina  Oana  Paula | 7  14  17 | 4 dec 2014  28 nov  25 nov 2014 | 11 dec 2014  11 dec 2014  11 dec 2014 |
| 5. | Test | Alina  Oana  Paula | 2  5  3 | 12dec 2014  10dec2014  12dec 2014 | 14 dec 2014  14 dec 2014  14 dec 2014 |

## **Deliverables list**

|  |  |  |  |
| --- | --- | --- | --- |
| **DEL. NO.** | **DELIVERABLE NAME** | **WORK PACKAGE** | **DATE** |
| 1.  2.  3. | DeliverLib  SearchBookLib  LocateLib | 5  5  5 | 15 dec 2014  15 dec 2014  15 dec 2014 |

## **Description of each work package**

### ***Work package no1: Plan***

We all met in order to split the tasks and decide how the project will look like. The objectives were established and the work distributed.

### ***Work package no2: Analyze***

**Server**

I started by looking for some parser examples in PHP and for various operations with a database. I also read some articles regarding them.

**Client analysis:**

I started by looking for some of the native applications for Android existing or in development which are similar and I read some articles.

**Location analysis:**

I started by looking for LocationServices API provided by Android and Google and I read a few articles and forum about how to get your current location on Android and how to search the map for other addresses.

### ***Work package no3: Design***

**Server Design:**

I established the structure of the application and how the operations with the database are made and the conditions which determine them.

**Client Design:**

I established the structure of the application, how the pages are linked and the GUI of the application.

**Location Design:**

I searched for Android interfaces, activities, ListView and all of the APIs needed to implement this. I draw a draft diagram of activities and decide how the application will look like.

### ***Work package no4: Code***

**Server Code:**

The architecture uses some modules: authentication, register, history, rating and search which communicate with the database in various ways.

When trying to register, the client must provide a valid username, password and e-mail.

When trying to authenticate, the client must provide a valid username and password, already registered in the database.

The user cannot login again if it is already logged in, it must logout first.

When searching for a bookname, the client must also provide a valid username and password for adding the search result to a history and the bookname into a comon table from the database for furtherrating.

When trying to rate a book, a client must provide the bookname and a value for rating, but also a valid username and password because only authenticated users are allowed to rate a book.

For guest access, the server has a user guest which is easy accessible from client side with username guest, password guest.

The architecture is simple, connecting the modules above to the database and taking queries according to the information needed.

The results obtained from queries are then provided in an XML format.

**Client Code:**

The architecture used by the two entities that communicate is REST (oriented resources), based on common language XML, the engine that makes it possible to transfer data via the Internet and HTTP common transport protocol. For implementing the functionality of the application, first of all it was necessary to establish the communication between client and server. For this, I used serialized objects Message type using HTTPConnection and URLConnection.

The basic functional unit of the system is Android Activity: the activity present in the flow of operations that make up the application. Each of the classes used in the application corresponds to an activity. Basically, as the application accesses resources, new pages are created. Elements on the pages (buttons, lists, TextView's) are described in the XML pages associated.

**Location Code:**

I wrote the code and continued to document about the little stuffs that escaped on Analyze step.

I used Android 4.1.1, SDK 16 revision(4.1.2) and ADT 23.0. I wrote the code in Eclipse Luna with Android and Google SDKs.

I used google play services for getting the location and HTTPGet/HTTPPost to acquire a JSON with the specific coordinates and address of a given string to search on Google Maps.

From Android I used Button, ListView, TextView and other perspectives to create a user-friendly interface. The application check network connection and location service status.

I made a top 3 libraries (one of each bookstore) in order from the nearest one to the farest.

### ***Work package no5: Test***

**Server Testing:**

I tested the application in Google Chrome, by simply providing various queries and examining results in XML format.

**Client Testing:**

When testing the application, I initially used a virtual Android device (AVD) that successfuly shapes a real device. Subsequently, the client application was installed on a mobile device smartphone HTC running Android OS version 4.1.1

Given the fact that the application is composed of multiple modules with different functionalities, I first conducted independent testing of components. Subsequently, I integrated all of these components to make up a functional system. To detect errors, I used the utility DDMS (Dalvik Debug Monitor).

**Location Testing:**

I tested the application on HTC ONE S running Android 4.1.1 and Nexus 7 running Android 4.4.1. The application had good results on both devices.

# **Risk management**

## **Risk identification**

* If the components used, such as frameworks and other software modules, run with full privileges, an attack can facilitate serious data loss or server takeover.
* If the server redirects users to other pages and websites, without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.
* If the client cannot connect to the server, the application won’t be functional.
* If no Internet connection is established, the location cannot be provided.
* If the location service is off when running the application, it exists the risk that the program will consider the last location registered on device and this could lead to wrong output.
* If the webhost we are currently using is not available, the server is not functional. We are not currently providing a local database for the application.

## **Probability / impact evaluation**

|  |  |  |
| --- | --- | --- |
| Risk summary | Probability of event | Impact |
| Deficient teamwork  Lack of qualification  Technology risks  Complex tasks  Using components with vulnerabilities  Unvalidated redirects  No connectivity to server  Server crashes  No Internet connectivity  Location service off | Low  Low  Low  High  High  Low  Low  Low  Medium  Medium | Medium  Medium  High  High  High  High  Very High  High  Very High  High |

## **Responses to risks**

* Deficient teamwork - Teambuilding
* Lack of qualification - Early training
* Technology risks - Check how these performed in the past and we will know they will work.
* Complex tasks - Decompose complex tasks into smaller tasks.
* Using components with vulnerabilities - Inform about the components before using them.
* No Internet connectivity - Be sure acces to Internet is granted.

Location service off - Activate location service before using the application.