Bilkent University

Senior Design Project

Cerebro

Project URL: *kyorgancioglu.github.io/CerebroApp/*

High-Level Design Report

Project Members:

Kaan Yorgancıoğlu 21302439
Dilara Ercan 21201256
Özgür Taşoluk 21301674
Nihat Atay 21102292
Furkan Salih Taşkale 21300878

Supervisor: Selim Aksoy
Jury Members: Ercüment Çicek, Uğur Güdükbay

Dec 30, 2016

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Table of Content

Table of Figures

1. Introduction

1.1 Purpose of the System

Everyday thousands of new media are published in many different forms. Although there are some applications that help identify media by several ways, most of the mainstream apps fail to provide more than minimal information. Also most of them focus on a single type of media.

It is common that consumers come across a media product such as albums, movies, books in a shop and want to learn more about it. But there is no single easy way to access all the relative information in one place. Our solution to this problem is “Cerebro”. Cerebro is a mobile app for users to take advantage of. It is a platform that searches the media and puts together desired information by gathering them from trusted third party sources.

Cerebro uses the camera of the smartphone to capture the cover art of media such as book covers, album covers, movie posters, etc. It then searches the database for any matches and gathers all the information from the internet such as the artist, author or the director of the media, casts, release date, customer reviews, brief summary and etc. and displays them in a tidy smart way. This would allow users to quickly find information from their favorite sources, all at once.

1.2 Design Goals

1.2.1 Usability

Cerebro should be easy to use and understandable for the user. User who uses programs like Snapchat and Shazam can easily use this program without any user guide.

1.2.2 Reliability

In any case of failure, Cerebro should give meaningful error message to user and developer to fix system quickly. Also, software should be able to store old data in case of system failure.

1.2.3 Security

Personal data of the user should be protected and should not be shared by third party applications.

1.2.4 Performance

The system should be efficient and give fast response for user request. When user scans any book, movie or music album, waiting time for server answer should not be more than 3 seconds. Also, Cerebro will use cache and internal storage of Android devices, since mobile phones doesn't have too much battery power to use, our application should not waste too much battery power when it runs.

1.2.5 Portability

The user should be able to install this application on different Android versions.

1.2.6 Supportability

The user should be able to send any suggestions and feedbacks using applications itself.

1.3 Definitions

Django: Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design[1].

Android OS: Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets [2].

Python: Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java[3].

Amazon Web Services: Amazon Web Services (AWS), a subsidiary of Amazon.com, offers a suite of cloud-computing services that make up an on-demand computing platform[4].

API: Application programming interface

1.4 Overview

Our application is a mobile application that will inform user about books, movies and albums by scanning cover of the media. By using this application, user can get quick and reliable information about these medias. Cerebro's GUI is completely customizable according to user's choice. User can choose what information will be shown about media and it will allow user to get only necessary information about specific media.

2. Proposed Software Architecture

2.1 Overview

Cerebro is a mobile application for Android OS in which users can find information about books, films and music albums by searching with the cover image of an item. The application consists of two major parts: client and server sides. Client side, also known as frontend, will be the mobile application. In the application user will use camera to scan the cover image of an item, then application will send the image to the server part of the project. Server side will be responsible for processing the image that the frontend of the application sent. It will run necessary operations for recognition of the image. After completing analyzing the queried image, it will start a search to find the given image in the database of the application. When searching is finished, server side create a response and send it to client side. With given information, mobile application will be updated.

In the client side, we plan to use MVC architecture to organize the structure of the application. In the view part, user will be able to see information about her search and use

camera to search with an image. In the controller part, the application will take care of the interactions from the user. In the model part, the application will contain objects like information page for each type of items, widgets etc.

2.2 Subsystem Decomposition

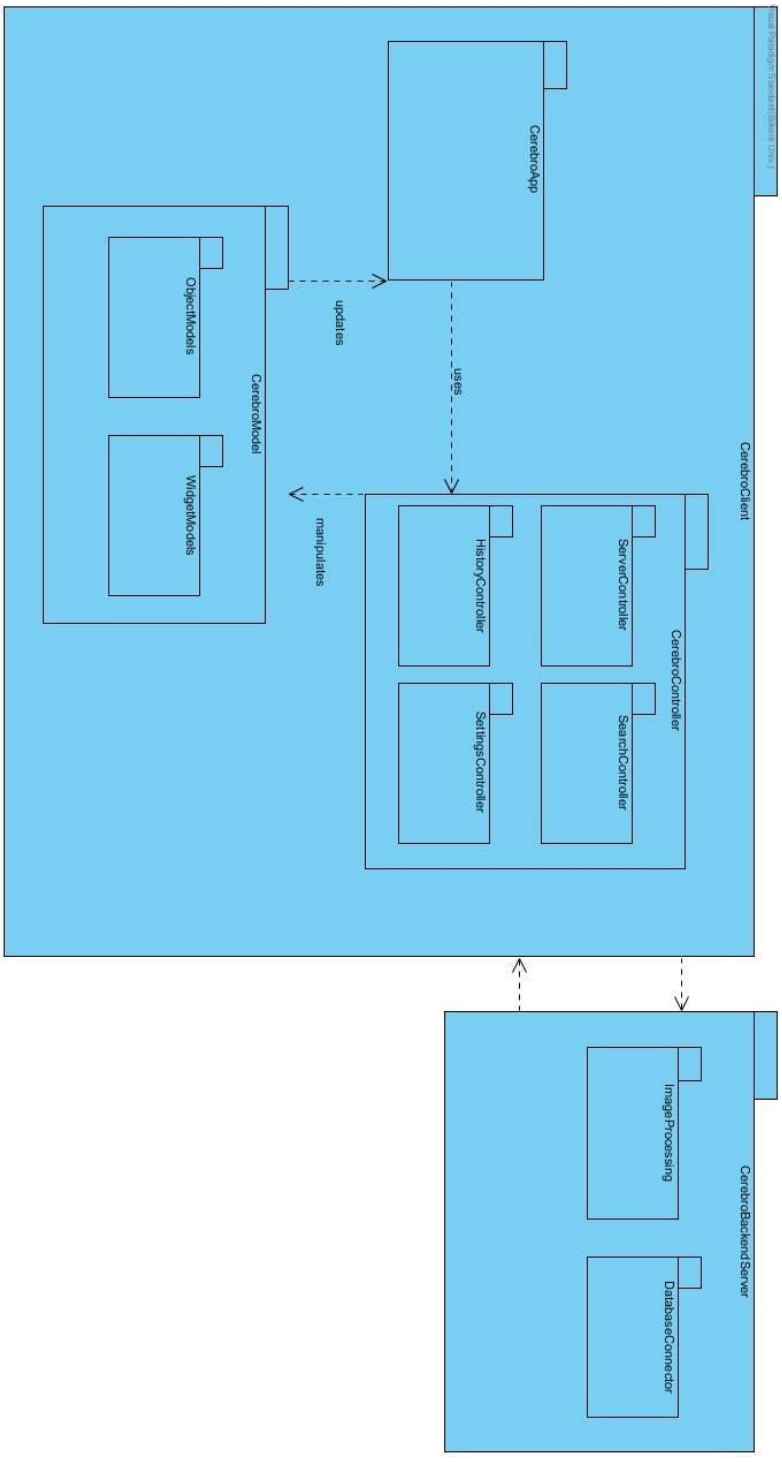

Figure 1: Subsystem Decomposition of Cerebro

Subsystem decomposition is an important subject to be analyzed because it's where the organization of the project is determined. Our project contains an Android application where users make some image request to the server. Also our project contains a server application where users' image queries are processed and database transactions are handled. Because of this request-response cycle to be maintained properly, we have decided to use Client/Server architecture as our main software design pattern. Client part will be the Cerebro android application which is called CerebroClient Subsystem and Server part will be the CerebroBackendServer Subsystem of our project.

Client Subsystem

Cerebro will contain an Android application where users make some image request to the server and server is responsible for processing the image and responding with information that is gathered using the results of the database queries. With respect to needs of the application, we decided to select MVC software design pattern as our architectural choice for the mobile application. MVC provides us to handle different parts of the application separately. As expected we have three main subsystems in our app. For model, corresponding subsystem in our application is CerebroModels Subsystem; for view, we designed CerebroApp Subsystem and lastly, for controller we have CerebroController Subsystem.

In our application we have three main subsystems which are following:

- **CerebroApp Subsystem**: This subsystem is the view side of our application which is the user interface of our application.
- **CerebroController Subsystem**: This subsystem is the main controller subsystem of our project. Main task of this subsystem is handling interactions coming from the user and managing necessary processes.

    - **ServerController Subsystem**: This subsystem is responsible for connection with server and is in charge of API requests and responses. It maintains data transaction between our client application and server.
    - **SearchController Subsystem**: This subsystem maintains search operations performed by user through our mobile application.
    - **HistoryController Subsystem**: This subsystem takes care of saving and controlling search and view histories for the user.
    - SettingsController Subsystem: This subsystem is in charge of controlling application settings including user interface customization, selection of widgets etc.
- **CerebroModel Subsystem**: This subsystem is the main data storage part of our project. It also maintains objects in the application.
    - **ObjectModels Subsystem**: This subsystem deals with main info page objects for our application. It is responsible for creating and destroying them.
    - **WidgetModels Subsystem**: This subsystem mainly works on widget objects which reside in an info page. It is responsible for creating and destroying them.

Server Subsystem

Our server subsystem is primarily works on processing given images and querying database with the achieved results. We didn't use any particular design pattern for this subsystem. It contains two inner subsystems that are following:

- **ImageProcessing Subsystem**: This subsystem is responsible for analyzing the images and creating identifying information for given image, for instance feature extraction.
- **DatabaseConnector Subsystem**: This subsystem is in charge of connecting the database and managing the database transactions.

2.3 Hardware / Software Mapping

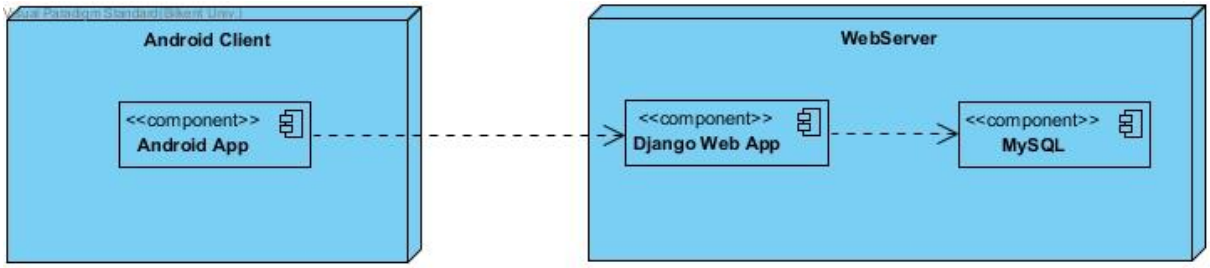

Figure 2: Hardware / Software Mapping of Cerebro

In Cerebro app, the user will access the database server when he/she opens the application. The app will connect the database instantly whether the user performs a search or not, since there will be the categories and suggestions about media on the opening page.

2.4 Persistent Data Management

In order to make an efficient application, we should care about the persistency of our data and consider some essential points. Time efficiency is important since our app will be dynamic and instant. When the user performs a search, the app should show the results in a short time of period. In order to achieve this, the connection between the server and the application has to be quick.

2.5 Access Control and Security

Cerebro app will not require the user to enter access credentials. It will use the device ID to create an initial record when the app launches for the first time. All critical data will be stored locally on the device. Therefore, it will not be possible to restore data such as search history, once the app has been uninstalled. The app will require media, network and camera access of the device.

2.6 Global Software Control

Cerebro is a server-client application. Client is a mobile app that runs on the Android OS. Server is a Python Django app that runs on cloud. Amazon Web services may be used to host Cerebro server.

The mobile app is for the use of end user. When launching the mobile app, the app sends the version data to the server and the server acknowledges the version. If the mobile app is a version that is no longer supported, the server asks for the user to update the app, making the app unavailable for use until it is updated by the user.

When the runs a search, the app puts together an API request whose contents depend on the type of search. It then sends the request to the server, where it is queued and then processed. After the request is processed, the server constructs an AP response and sends it back to the requesting device.

The server might send notifications to users, for promotion purposes. This is done through Android's notification service, not through Cerebro app. Therefore, the app is not required to be running to receive notifications.

2.7 Boundary Conditions

- Initialization: At this stage the mobile app sends the version data and waits for a response before the launch process is completed as mentioned above. It then completes launching as usual if the version is acknowledged, if not it denies user access to the app until user downloads the update.

- Termination: The user can terminate the program anytime by closing the process. All files such as search history are updated at the time of modification. If the user chooses to close the app when a file is being modified, the app waits for the file operation to finish before closing. Instant termination is allowed during all other processes, including but not limited to; sending/receiving network data, waiting for a response and scanning media.

- Failure: The app receives a memory warning from the OS, which is unlikely, the app displays a message and shuts itself down as described in the Termination section. Controlled exceptions do not cause the app to crash. These exceptions include but not limited to; network timeout, network change detection, camera inaccessible. All controlled exceptions display a warning message that instructs the user to fix the issue if there is any fix. Uncontrolled exceptions might cause crashes but these crashes are unlikely to cause permanent damage as files are updated at the time of modification. If the system detects a corrupted file, it displays a message that contains the problematic filename and creates a replacement file. While creating a replacement file, the app sends an API request to the server if necessary. All warning messages also asks the user if they wish to send the error info to the server. This data will be used to improve Cerebro App.

3. Subsystem Services

The Cerebro app consists of two main subsystems. These subsystems are the client subsystem and the server subsystem which are also consisted of several subsystems.

3.1 Client Subsystem

This subsystem has the CerebroApp Subsystem, CerebroController Subsystem and the CerebroModel Subsystem.

3.1.1 CerebroApp Subsystem

CerebroApp subsystem consists of the view part of the program. Containing MainActivity, CameraActivity and SettingsActivity classes to create the user interface of the app.

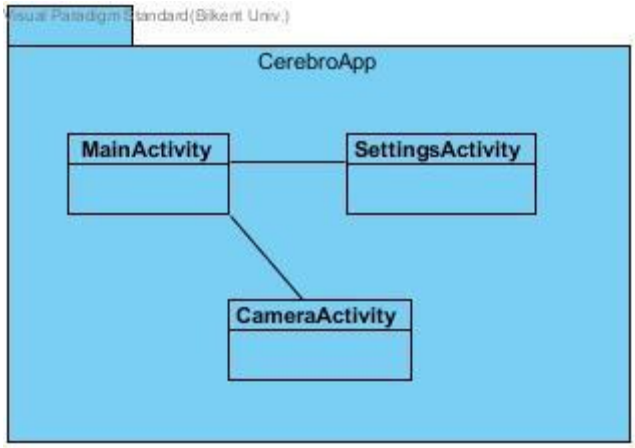

Figure 3: Subsystem of CerebroApp

3.1.2 CerebroController Subsystem

It has several controller subsystems under it.

- **ServerController Subsystem**

  ServerController connects app to the server. It takes API requests and gives the relevant response and errors when needed.

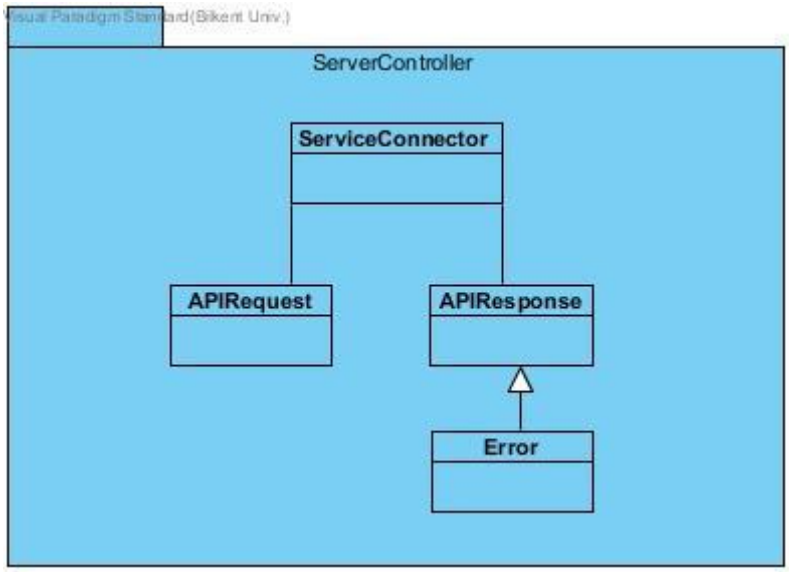

Figure 4: Subsystem of ServerController

- **SearchController Subsystem**

This subsystem is responsible of search operations.

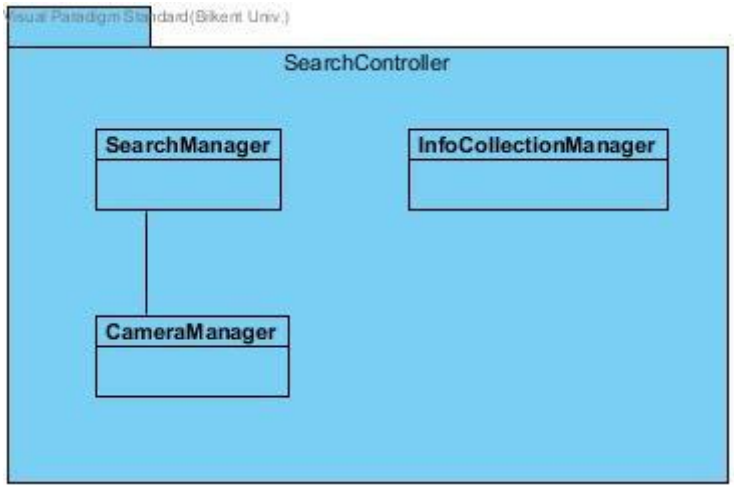

Figure 5: Subsystem of SearchController

- **HistoryController Subsystem**

This subsystem controls the storing of the search history of the user with the HistoryManager. And views it on demand.

Figure 6: Subsystem of HistoryController

- **SettingsController Subsystem**

Settings controller subsystem has the SettingsManager which updates the info that is requested to the InfoPageCustomizer and gathers the needed widgets from the WidgetFactory through the InfoPageFactory.

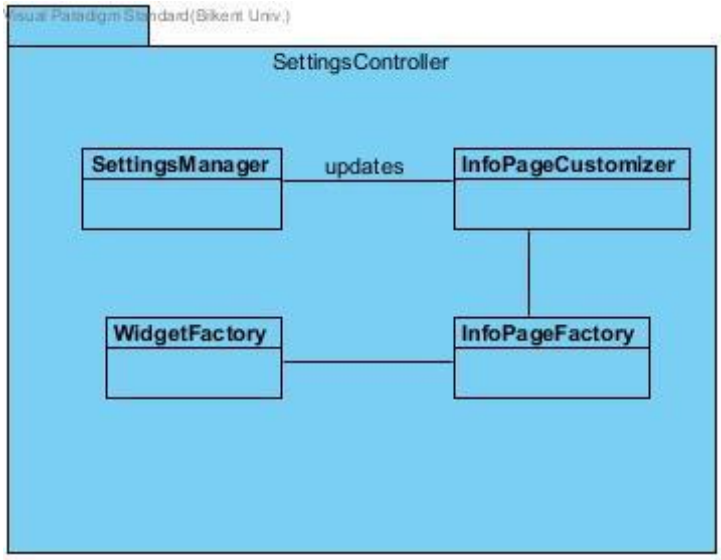

Figure 7: Subsystem of SettingsController

3.1.3 CerebroModel Subsystem

It consists of the object classes of the program and controls them.

- **ObjectModels Subsystem**

  This subsystem is responsible of the main info page objects. It creates and destroys objects such as Book, Film and Music under the Info Page class.

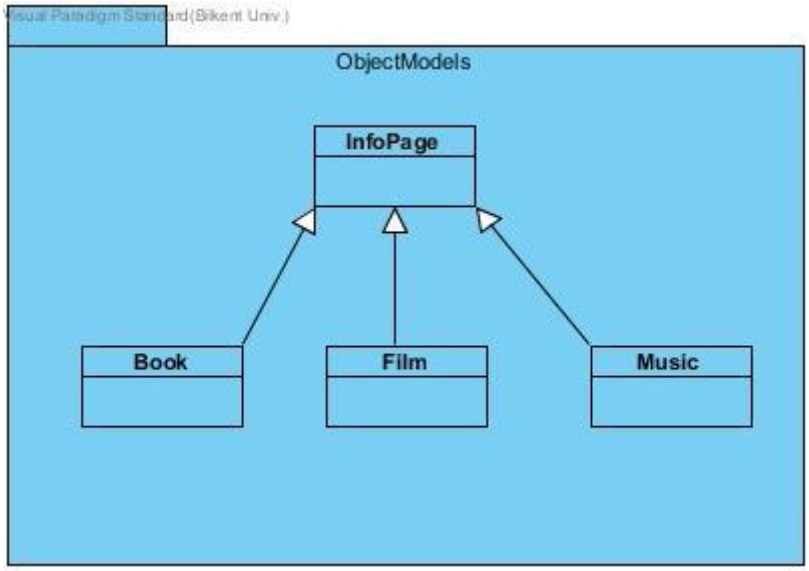

Figure 8: Subsystem of ObjectModels

- **WidgetModels Subsystem**

  WidgetModels Subsystem has the main widget objects such as image, reviewcomment, similar, purchase, summary classes. It controls creation and deletion of these objects, according to the object that is scanned.

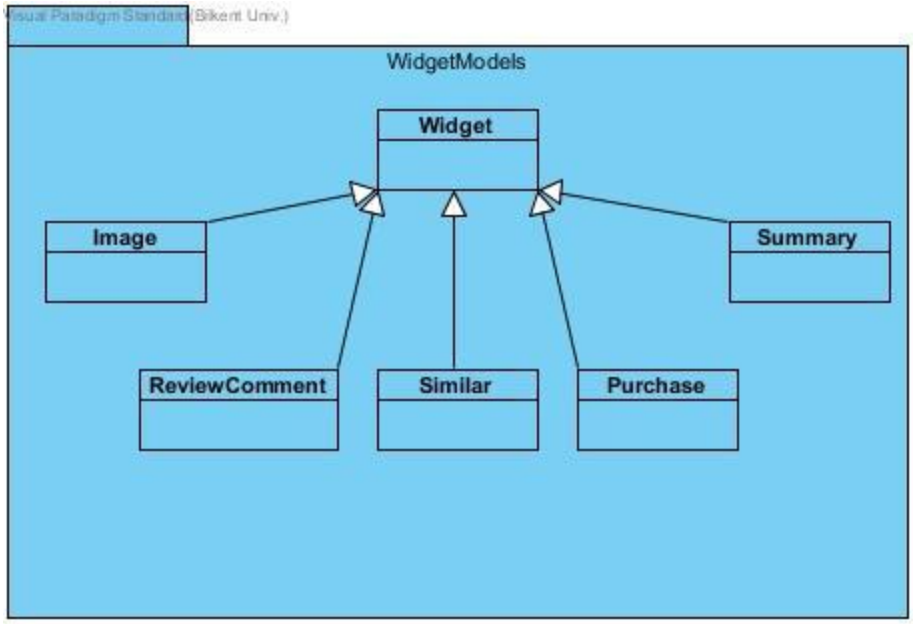

Figure 9: Subsystem of WidgetModels

3.2 Server Subsystem

Server subsystem handles the server part of the Cerebro app. It is a combination of two subsystems that are ImageProcessing Subsystem and Database connector subsystem.

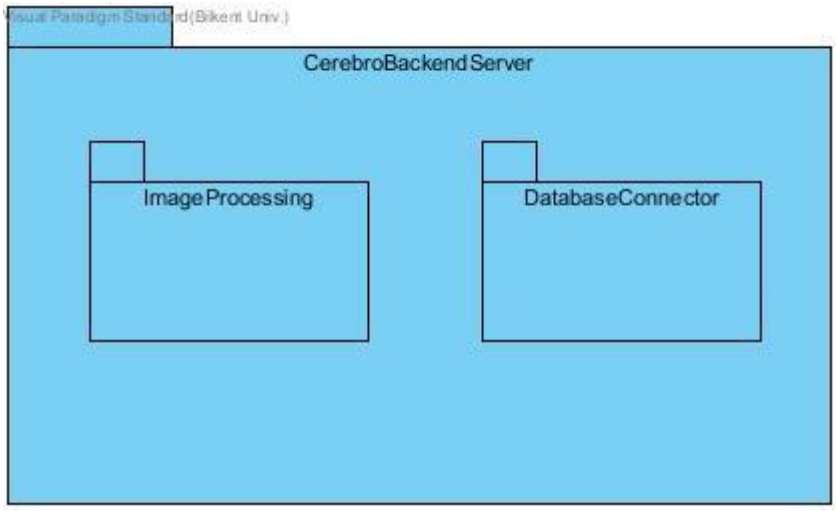

Figure 10: Subsystem of Server

3.2.1 ImageProcessing Subsystem

Image Processing subsystem analyses the images that are scanned through the camera of the device and creates identifying information of the object for its information to be gathered.

3.2.2 DatabaseConnector Subsystem

Database Connector subsystem connects the database to the app and controls the data transactions from data to the app on demand.

4. References

[1] Django, Django Framework
https://www.djangoproject.com (Accessed on 27 Dec 2016)

[2] Wikipedia, Android (operating system)
https://en.wikipedia.org/wiki/Android_(operating_system) (Accessed on 27 Dec 2016)

[3] Wikipedia, Python (programming language)
https://en.wikipedia.org/wiki/Python_(programming_language) (Accessed on 27 Dec 2016)

[4] Wikipedia, Amazon Web Services
https://en.wikipedia.org/wiki/Amazon_Web_Services (Accessed on 27 Dec 2016)