Bilkent University

Senior Design Project

Drector

Low-Level Design Report

Shahriyar Mammadli, Nihad Azimli, Burak Özmen, Veysel Alperen Ceylan

Supervisor: Varol Akman

Jury Members: Çiğdem Gündüz Demir, İbrahim Körpeoğlu

Innovative Expert: Mustafa Sakalsız

Project Website: https://drectorapp.github.io/Drector/

Table of Contents

Table of Figures

Table of Tables

1. Introduction

1.1. Object design trade-offs

1.1.1. Platform Compatibility vs. Performance

Drector will be cross-platform application, so that it will be implemented in Ionic 3. However, in non native development environment performance cannot be the same as in native. Ionic 3 eases development procedure by providing us necessary plug-ins for camera and map with highly developed libraries. On the other hand, it would take too much time to accomplish these functionalities in native development.

1.1.2. Memory vs. Time

Drector's image proccessing mode may be implemented in 3 different options, which are server side, streaming and user's device. To increase the timing performance, we will implement image processing in user's device. It will provide us higher performance, however, memory of user's device will be more in use.

1.1.3. Security vs. Cost

Drector collects data from users including their, and drones' information, current and past locations. Thus, it is crucial for our system to ensure security and keep the information of the users secure. To improve security, in server side, we will encrypt the data before sending it to the database. Relatedly, the security introduces monetary, time, and labor cost.

1.1.4. Traditional vs. Innovative

Drector is a profoundly innovative application but also has a traditional interface and features. In recent years, users became familiar with a classic social interface like Instagram so we designed a similar edit profile interface. Also, to open camera mode in the app as in instagram user will swipe right. Reporting the drone flying around will have resemblance to Instagram's user reporting. Additionally, map mode of Drector will be as in Google Maps, where pins will show the drones flying around. With the help of traditional applications, users can learn quickly how to use Drector.

1.2. Interface documentation guidelines

Package	Package of the Class
Class Name	Name of Class
Description	Description of the Class/Service
Attributes	Attribute Name: Attribute Type
Methods	Method name (Parameters): return type

Table 1 Interface Documentation Guidelines

1.3. Engineering standards

Our documentation related to project is based on IEEE report format which is widely used and accepted internationally by engineers. In order to design and represent the description of class interfaces, diagrams, scenarios and use cases, subsystem compositions, and hardware-software components depiction, we used UML engineering standards which are very common in software industry.

1.4. Definitions, acronyms, and abbreviations

- IEEE: The Institute of Electrical and Electronics Engineers [1]
- UML: Unified Modeling Language [2]
- NoSQL: non-SQL(Structured Query Language)
- Android OS: Android Operating System
- GPS: Global Pointing System
- IOS: Iphone Operating System
- GSM: Global System for Mobile Communications
- UI: User interface
- API: Application Programming Interface
- HTTP: Hypertext Transfer Protocol [5]

2. Packages

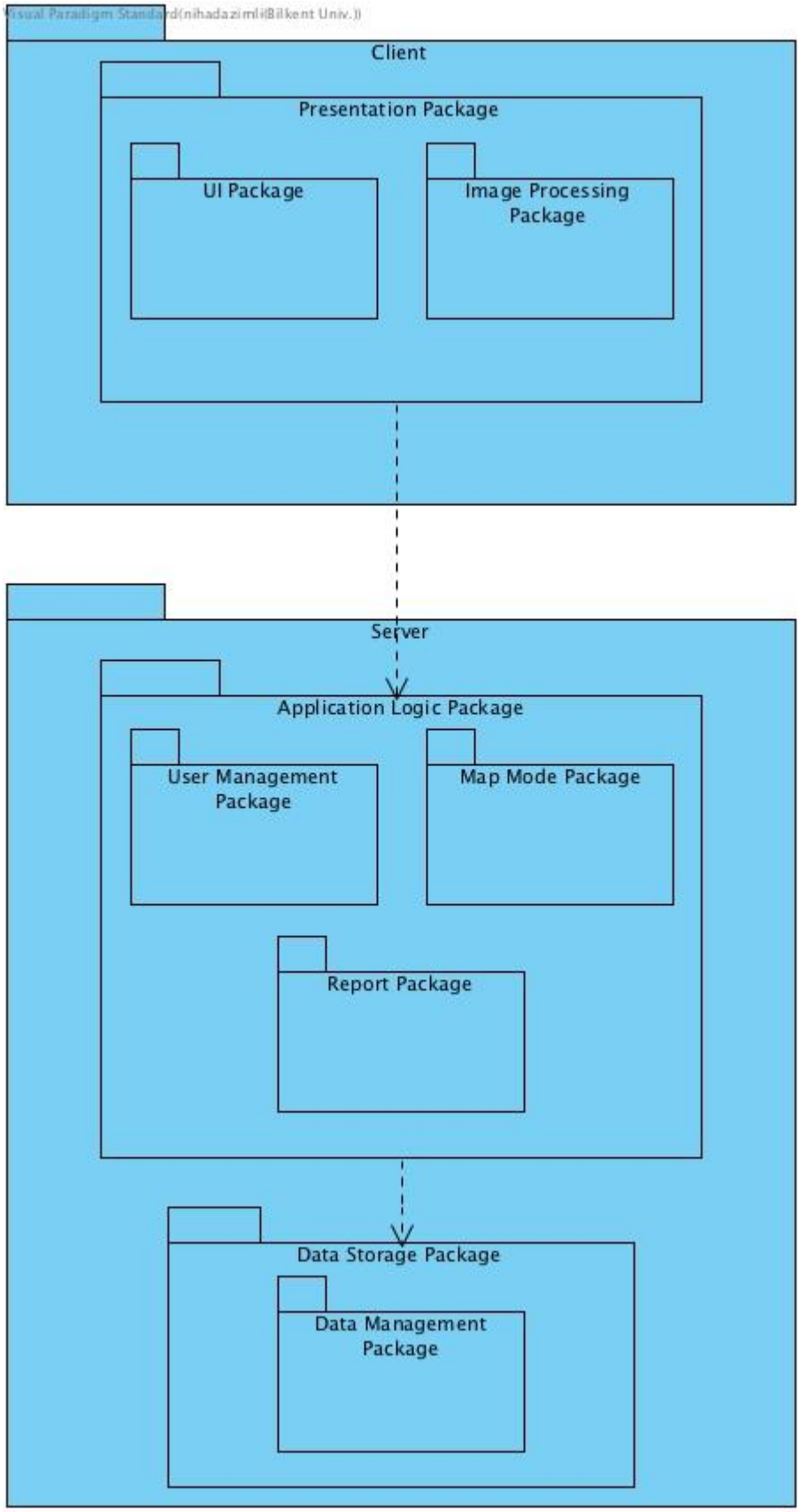

Figure 1 Packages and Dependencies Overview

In subsystem decomposition, the project is decomposed into three components. On the client side, presentation layer reside. Data storage layer and application logic layer lies under the server side. Using this decomposition, the project is divided into three main packages as well: Presentation package, Application Logic Package and Data Storage Package. Presentation package consist of two packages namely UI Package and Image Processing Package. Application Logic Package consists of three packages namely User Management Package, Map Mode Package and Report Package. Data Storage Package consist of only one package namely Data Management Package.

2.1.Presentation Package

Presentation package consist of two packages.Presentation package depends on application logic package because UI is responsible to show most of the entities to the user, allow user to change some of them. In order to do that, Presentation package need access to invoke methods of Application Logic Package and gather required information to display.

2.1.1.UI Package

UI package contains classes for user interfaces. These classes for different application pages such as login page, sign up page, forgot password page, map view page, camera view page, information page, report page, profile page, and add drone page.

2.1.2.Image Processing Package

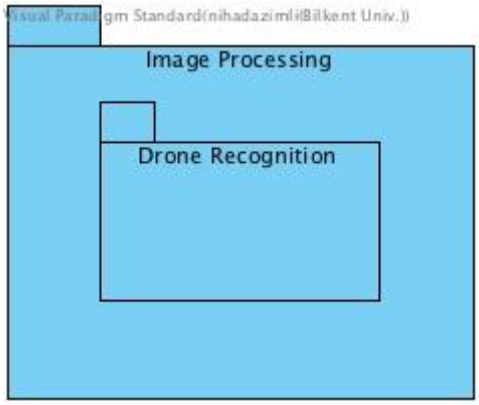

Figure 2 Image Processing Package and Content

Image processing package contains Drone Recognition Class. In this class camera view is processed and managed, and the detection of drone with image processing is done.

2.2. Application Logic Package

This package contains the general logic files such as user management system, map mode system and report system. This package depends on the data storage package since required user information are stored within the remote server. Data storage package methods invoke required methods to retrieve information from database and hand it over to application layer.

2.2.1. User Management Package

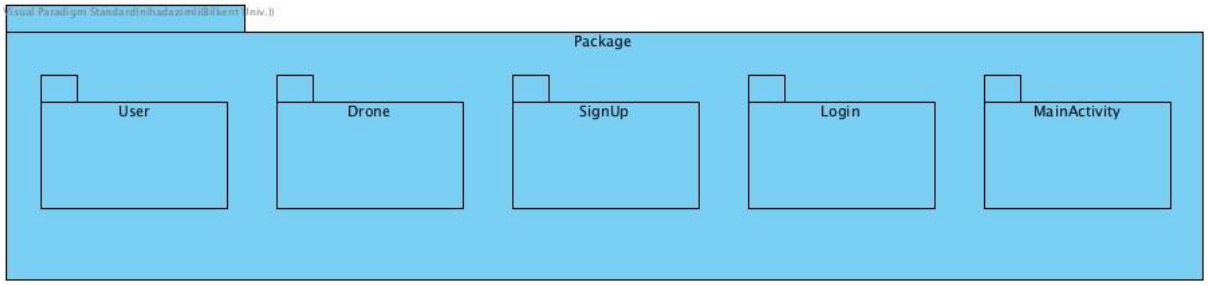

Figure 3 User Managemeng Package and Content

This package handles user management. It contains the classes namely User, Drone, Singup, Login, MainActivity. Login, signup activities, changing and showing information about user, and adding drones of the system are done inside these classes. This package requires data storage package to retrieve user and drone information, insert a new user and drone or update a current user or drone.

2.2.2. Map Mode Package

Figure 4 Map Mode Package and Content

This package handles map mode and contains MapActivity class. Placing drones' location in map consistently is processed and managed and selection of the drones from map are done inside this class. This package requires data storage package to retrieve user's and drones' locations.

2.2.3.Report Package

Figure 5 Report Package and Content

This package handles Report mode. It contains Report Class. The reporting selected drones to the officials and administrators is processed and managed inside this class. This package requires data storage package to retrieve user and drone information, and insert a new report to the data storage.

2.3.Data Storage Package

2.3.1.Data Management Package

User data is stored in a remote server database. This package contains classes to handle database information retrieval, insertion and update on all of users and drones data, reports and other necessary data. This package is the lowest level package, therefore it does not depend on any other package, this package stores the data collected from upper layer and the outer system. Only requirement is an internet connection to retrieve data from remote server.

3. Class Interfaces

Package	User Management
Class Name	User
Description	User class with two different types which are drone owner and ordinary user.
Attributes	**-name:** String

	-surname: String **-email:** String **-password:** String **-owner:** boolean **-location: double[]**
Methods	**+userExists():** boolean **+createUser():** boolean **+deleteUser():** boolean **+changeUserRole():** boolean **+editProfile():** boolean **+changePassword():** boolean

Table 2 User Class Interface

Package	User Management
Class Name	Drone
Description	Drone class
Attributes	**-ownerName:** String **-ownerSurname:** String **-droneId:** int **-reportNumber:** int **-droneType:** String **-location[]:** double
Methods	**+addDrone():** boolean **+reportDrone():** boolean **+changeOwner():** boolean

Table 3 Drone Class Interface

Package	User Management
Class Name	SignUp
Description	SignUp class which controls user sign up activity
Attributes	**-email:** String **-name:** String **-surname:** String **-password:** String
Methods	**+createUser():** boolean

Table 4 SignUp Class Interface

Package	User Management
Class Name	MainActivity
Description	Controls main activities
Attributes	
Methods	

Table 5 MainActivity Class Interface

Package	Image Processing
Class Name	DroneRecognition
Description	Class that recognizes drones
Attributes	
Methods	**+findDrone():** Drone

Table 6 DroneRecognition Class Interface

Package	Report
Class Name	Report
Description	Class that manages reports
Attributes	
Methods	**+report():** boolean

Table 7 Report Class Interface

Package	Map
Class Name	MapActivity
Description	Class that finds nearby drones' information
Attributes	
Methods	**+findNearbyDrones():** Drone[]

Table 8 MapActivity Class Interface

Package	Database Management
Class Name	DatabaseAccess
Description	It is a façade class that provides relation between Data Management Package and the all other packages. It supplies database operations.
Attributes	**-StringDB**_url **-DatabaseHelper** instance **-Connection** connect **-Statement** state -**List<String>** db_credentials
Methods	**+getInstance():** DatabaseHelper

	+**connectDatabase():** void +**createAccount(User u)**: boolean +**isAccountAvailable(String email):** boolean +**verifyAccount(Stringemail,String password):** boolean +**getAllDrones():** List<Drone> +**getDrone(int id):** Drone +**addDrone(Drone dr):** boolean +**updateDrone(Drone dr):** boolean +**updateAccount(User u):** boolean +**updatePassword():** boolean +**createReport(Report r):** boolean +**listReports(<optional>Drone d, <optional> User u):** List<Drone>

Table 9 DatabaseAccess Class Interface

4. Design Patterns

4.1. Façade Design Pattern

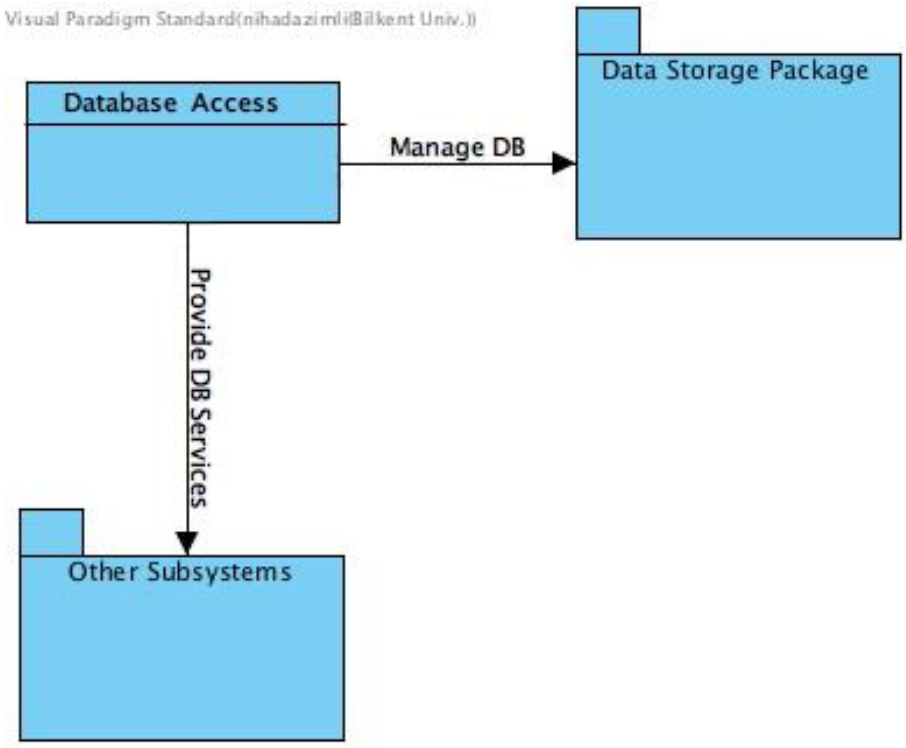

Figure 6 Façade Design Pattern

In order to manage database system properly, we used Facade Design Pattern [6]. We will implement Database Access class which will manage all the tasks about Data Storage Package and provide database services to all other subsystems.

5. Glossary

- Three-tier Architecture: Software architecture pattern for Client-Server applications.
- Server: The part of the system responsible from logical operations, scheduling, and data management
- Client: The part of the system the users interact with
- Request: Task identification sent to the server with required information.
- Response: Results of the request given to the client application.
- Query: Information request sent to the database.
- Angular: is a Typescript-based open-source front-end web application platform led by the Angular Team at Google and by a community of individuals and corporations.

6. References

[1]IEEE, "IEEE Standards Association," [Online]. Available: http://standards.ieee.org/. [Accessed 17 February 2017].

[2]UML, "Unified Modeling Language," [Online]. Available: http://www.uml.org/. [Accessed 13 February 2017].

[3]Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13- 047110-0.

[4]"Ionic Framework", Ionic Framework, 2017. [Online]. Available: https://ionicframework.com/docs/. [Accessed: 22- Dec- 2017].

[5]IETF, "RFC 2616 - Hypertext Transfer Protocol," IETF.org, 1999. [Online]. Available: https://tools.ietf.org/html/rfc2616. [Accessed 20 February 2017].

[6]Facade Design Pattern in Java - JournalDev", JournalDev, 2018. [Online]. Available: https://www.journaldev.com/1557/facade-design-pattern-in-java. [Accessed: 10- Feb- 2018].

[7] "Chapter 3: Architectural Patterns and Styles", Msdn.microsoft.com, 2018. [Online]. Available: https://msdn.microsoft.com/en-us/library/ee658117.aspx#ClientServerStyle. [Accessed: 10- Feb- 2018].