Aplicatie pentru inchiriat masini pe Android

Analysis and Design Document

Student: Pocol Alexandru Gabriel

**Group: 30235**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 5

2.1 Conceptual Architecture 5

2.2 Package Design 5

2.3 Component and Deployment Diagrams 6

III. Elaboration – Iteration 1.2 7

1. Design Model 7

1.1 Dynamic Behavior 7

1.2 Class Design 11

2. Data Model 12

3. Unit Testing 12

IV. Elaboration – Iteration 2 13

1. Architectural Design Refinement 13

2. Design Model Refinement 15

V. Construction and Transition 15

1. System Testing 15

2. Future improvements 16

VI. Bibliography 16

# Project Specification

Acest proiect reprezinta o aplicatie de inchiriat masini pentru Android. Aplicatia va avea doua tipuri de utilizatori: utilizatorul normal, care vrea sa inchirieze o masina si utilizatorul care posteaza anunturi.

In functie de tipul de utilizator se pot efectua urmatoarele operatii:

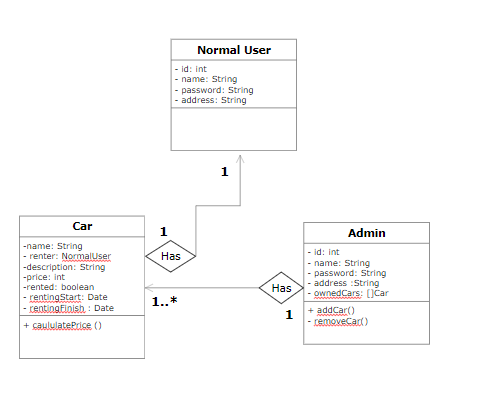
- admin : creare de cont, log in, adaugare de masina, stergere de masina, vizualizare masini inchiriate;

- utilizator normal: creare de cont, log in, vizualizare masini disponibile pentru inchiriere si inchiriere.

Aplicatia va folosi Firebase pentru stocarea datelor, deci pentru functionare este necesara conexiunea la internet.

# Elaboration – Iteration 1.1

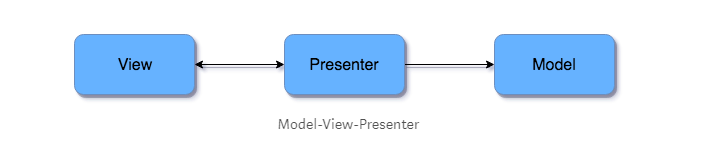
# Domain Model



# Architectural Design

## Conceptual Architecture

Arhitectura aleasa va fi MVP: Model View Presenter.



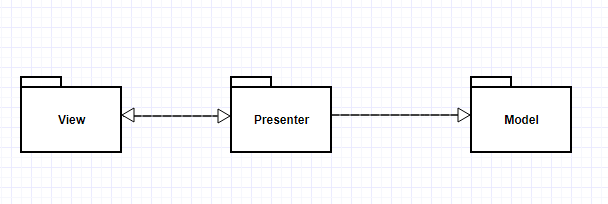
Model reprezinta o interfata responsabila pentru managementul datelor. Responsabilitatea acesteia include: lucrul cu API-uri, baze de date etc.

Presenter reprezinta layer-ul de mijloc dintre Model si View. Acesta este responsabil de logica, interogarea modelului si actualizarea view-ului precum si reactionarea la interactiunile userului si actualizarea modelului.

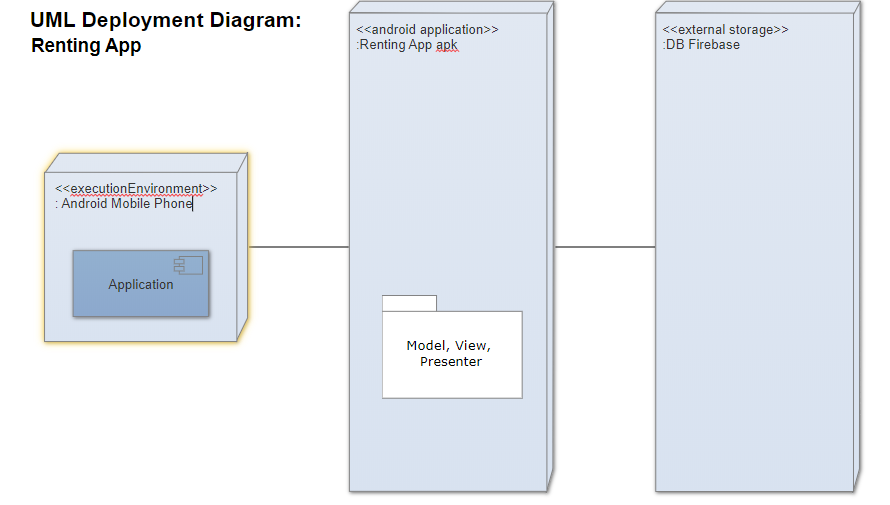
View: reprezinta partea de afisare, interactiunea cu user-ul.

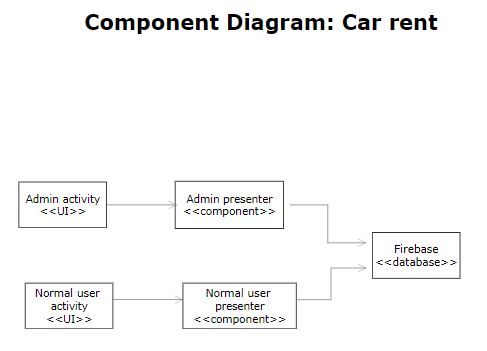
Acest pattern permite testarea mai usoara a aplicatiei deoarece, fiind o platforma foarte complexa eliminandu-se logica din partea de view devine astfel mai usor de testat. De asemena presenterul devine independent de framework.

## Package Design



## Component and Deployment Diagrams





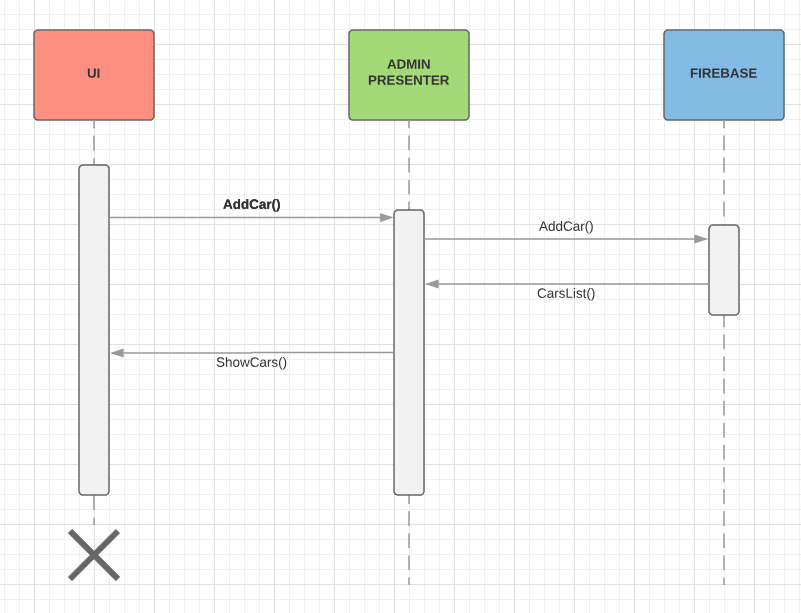
# Elaboration – Iteration 1.2

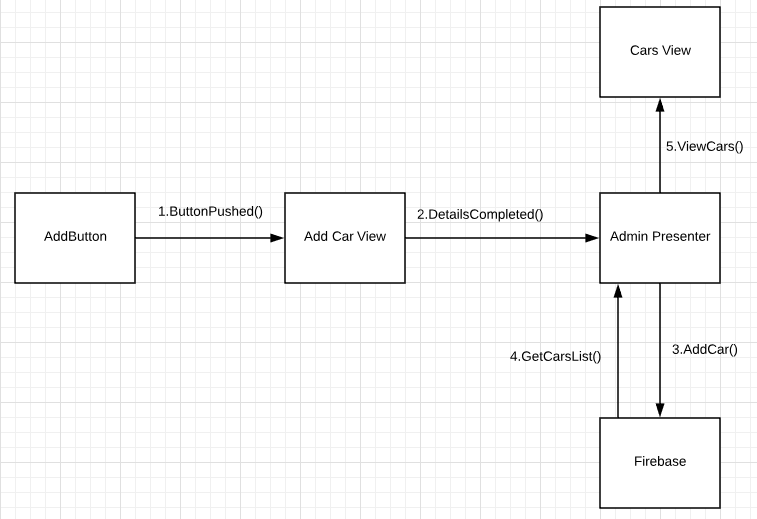
# Design Model

## Dynamic Behavior

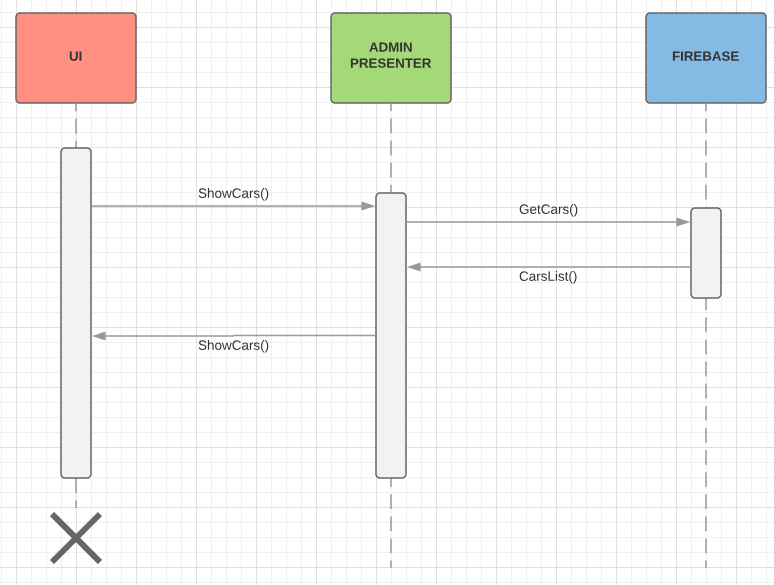
*[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]*

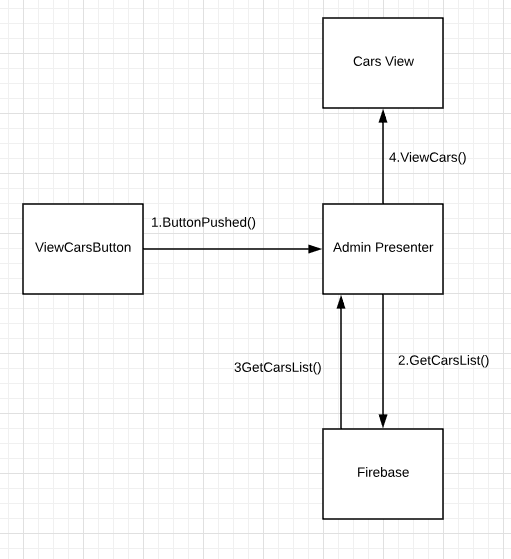
**Add car**





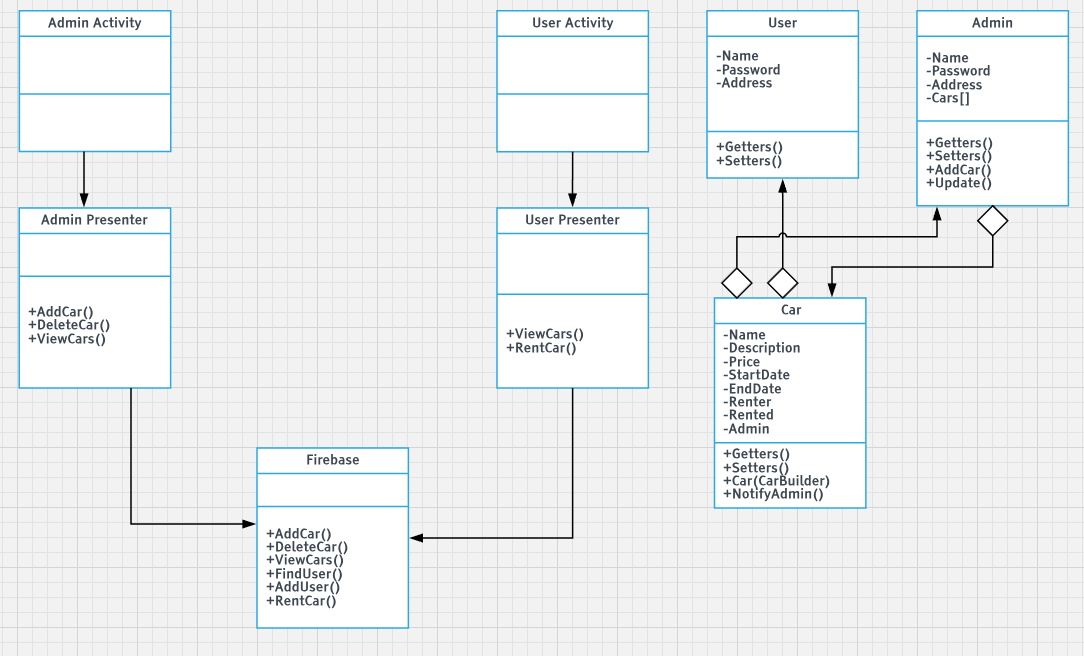
**View Cars**





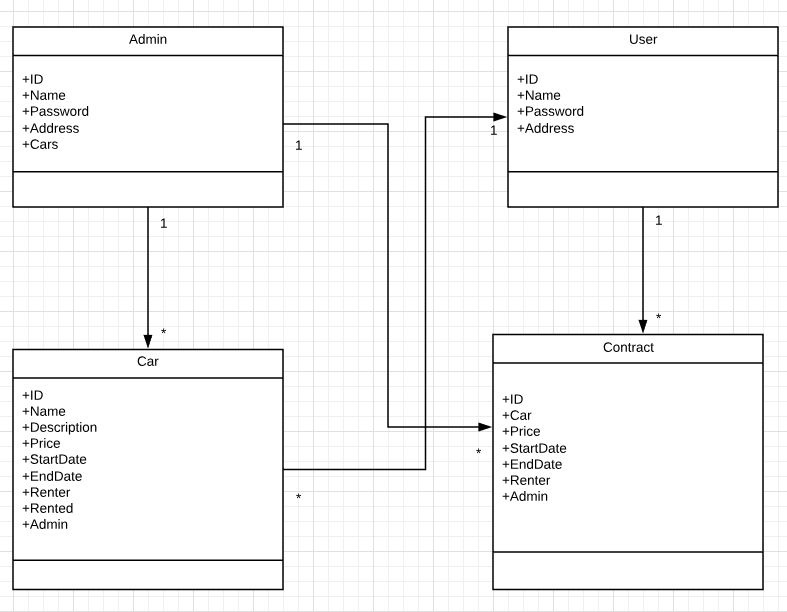
## Class Design

*[Create the UML class diagram; apply GoF patterns and motivate your choice]*



# Data Model

*[Create the data model for the system.]*



# Unit Testing

*[Present the used testing methods and the associated test case scenarios.]*

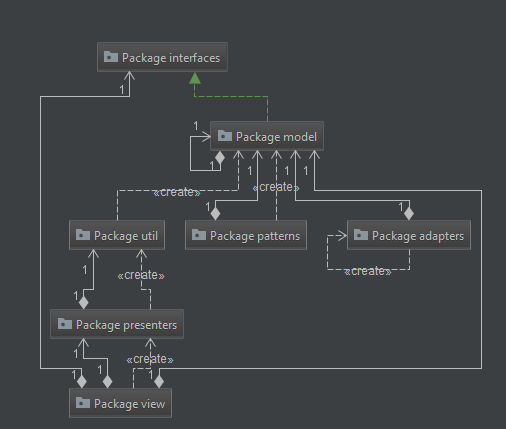
Pentru testarea unitara se va folosi Mockito impreuna cu Junit. Mockito este un framework pentru testare automata care permite creearea de obiecte de test dublu.

# Elaboration – Iteration 2

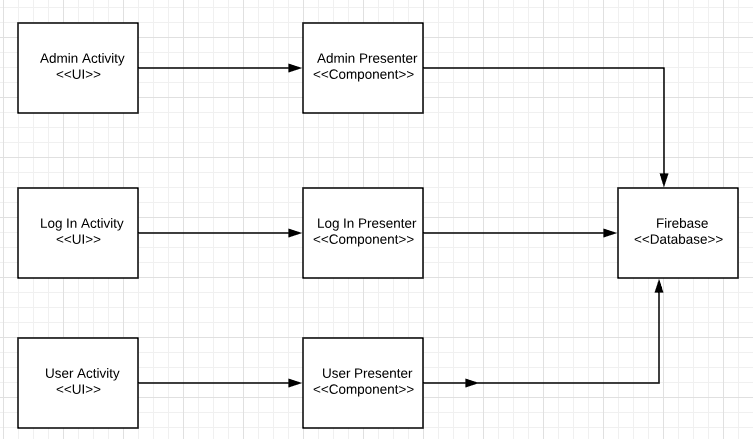
# Architectural Design Refinement

*[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]*

**Package design:**



**Component diagram:**



# Design Model Refinement

## *[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]*

Android SDK: este un framework de dezvoltare al aplicatiilor pentru dispozitive cu sistem de operare Android. Aplicatiile pot fi scrise in Java sau Kotlin. Contine unelte precum : debugger, librariii, emulator etc..

Firebase este o platforma web de dezvotare care ofera functionalitati precum: baza de date, autentificare, stocare, analiza etc..

GRASP:

* Creator : Main Activity – creeaza : Admin Fragment, User Fragment, Details Fragment, Add Fragment, Log In Fragment, Sign In Fragment.
* Polymorphism : User and Admin extend UserInterface.
* High Coesion : Firebase se ocupa doar de partea de date

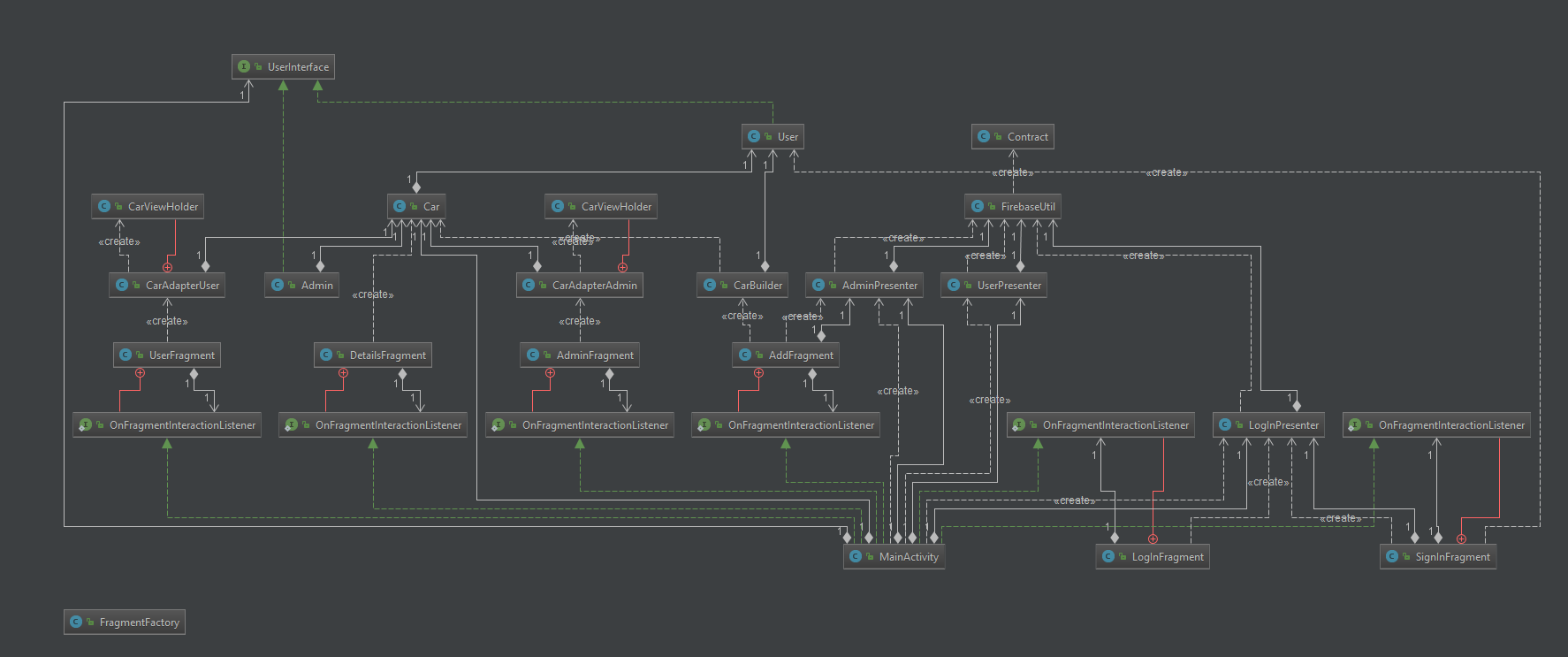
Design Patterns:

-Builder este un design pattern creational folosit la crearea de obiecte complexe acesta permitand crearea obiectelor pas cu pas.

- Adapter este un design pattern structural care actioneaza ca un pod intre doua interfete total diferite si independente.

-Factory este un design pattern creational care permite crearea de obicete distincte folosind o interfata comuna fara a dezvalui logica de creare.

Class Diagram:



# Construction and Transition

# System Testing

*[Describe how you applied integration testing and present the associated test case scenarios.]*

Pentru integration testing se va folosi Roboelectric. Roboelectric este un framework care ajuta la rularea unor teste mai rapide deoarece nu mai este necesar un emulator Android sau un dispozitiv, acesta permite rularea testelor pe un JVM normal.

# Future improvements

*[Present future improvements for the system]*

O dezvoltare ulterioara a aplicatiei ar putea fi monitorizarea in timp real a locatiei masinii prin intermediul aplicatiei de catre utilizatorii de tip admin.

# Bibliography

<https://medium.com/@cervonefrancesco/model-view-presenter-android-guidelines-94970b430ddf>

https://developer.android.com/docs/