Parking Management

Analysis and Design Document

Student: Aionitoaie Mihai  
 Bercean Andrei

**Group: 30233**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 24/April/19 | 1.0 | Iteration 1.1 | Bercean Andrei |
| 6/May/19 | 1.1 | Iteration 1.2 | Bercean Andrei |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 5

II. Elaboration – Iteration 1.1 5

1. Domain Model 5

2. Architectural Design 6

2.1 Conceptual Architecture 6

2.2 Package Design 7

2.3 Component and Deployment Diagrams 7

III. Elaboration – Iteration 1.2 7

1. Design Model 7

1.1 Dynamic Behavior 7

1.2 Class Design 8

2. Data Model 9

3. Unit Testing 9

IV. Elaboration – Iteration 2 9

1. Architectural Design Refinement 9

2. Design Model Refinement 11

V. Construction and Transition 12

1. System Testing 12

2. Future improvements 12

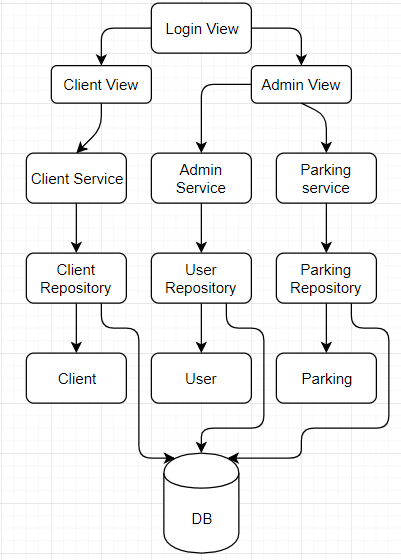
VI. Bibliography 12

# Project Specification

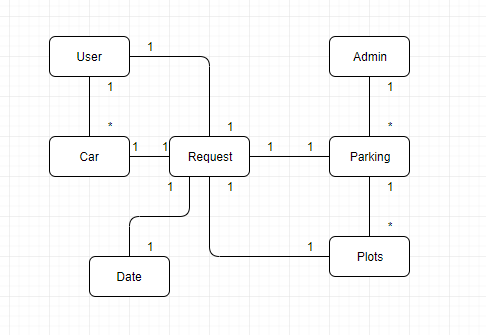
The Parking Management application manages a parking facility with multiple levels. Parking spots can be reserved as well as take as walk-in. A user will need an account and all the accounts will be managed by an admin account.

# Elaboration – Iteration 1.1

# Domain Model



conceptual class diagrams



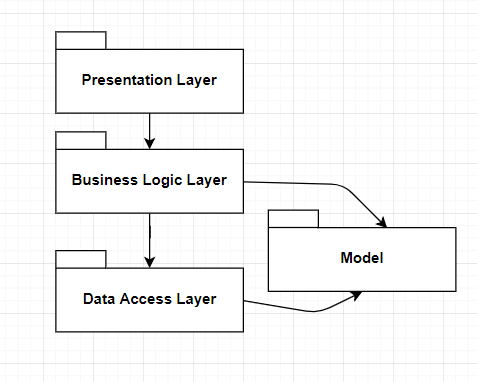
domain model

# Architectural Design

## Conceptual Architecture

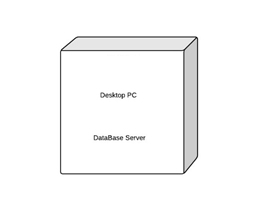
The application will be designed in accordance with the Layered architectural pattern as it can be seen. The pattern was chosen for its intuitive way of separating the code as well as making changes to logic layers or data access be more self-contained and not propagate into other layers.

## Package Design



## Component and Deployment Diagrams

Deployed locally on a single device, with no web component.



# Elaboration – Iteration 1.2

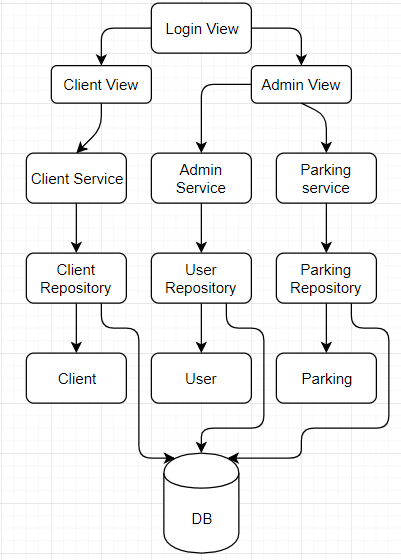
# Design Model

## Dynamic Behavior

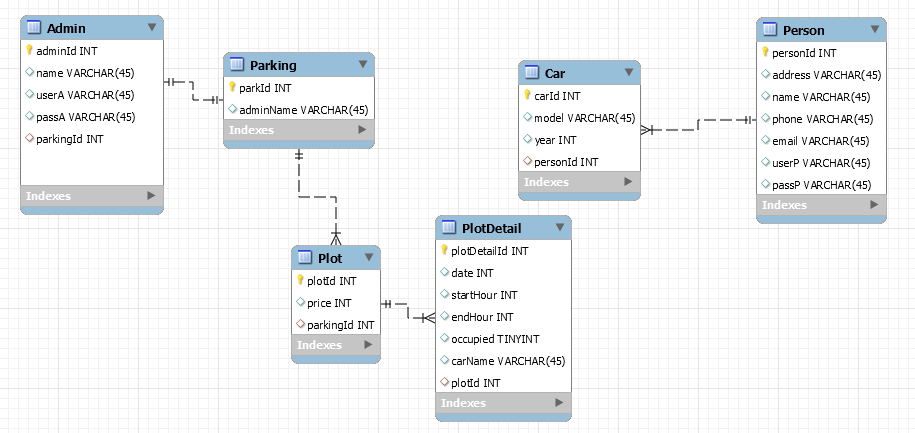
## 

getUserInformation diagrams

## Class Design



# Data Model



# Unit Testing

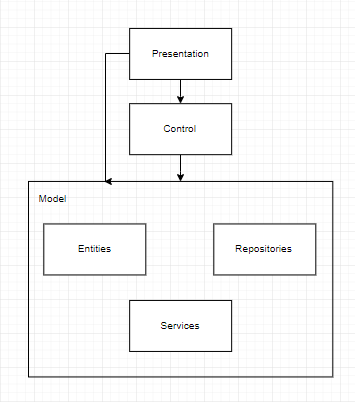
The tests will be done with JUnit on the CRUD operations for each

# Elaboration – Iteration 2

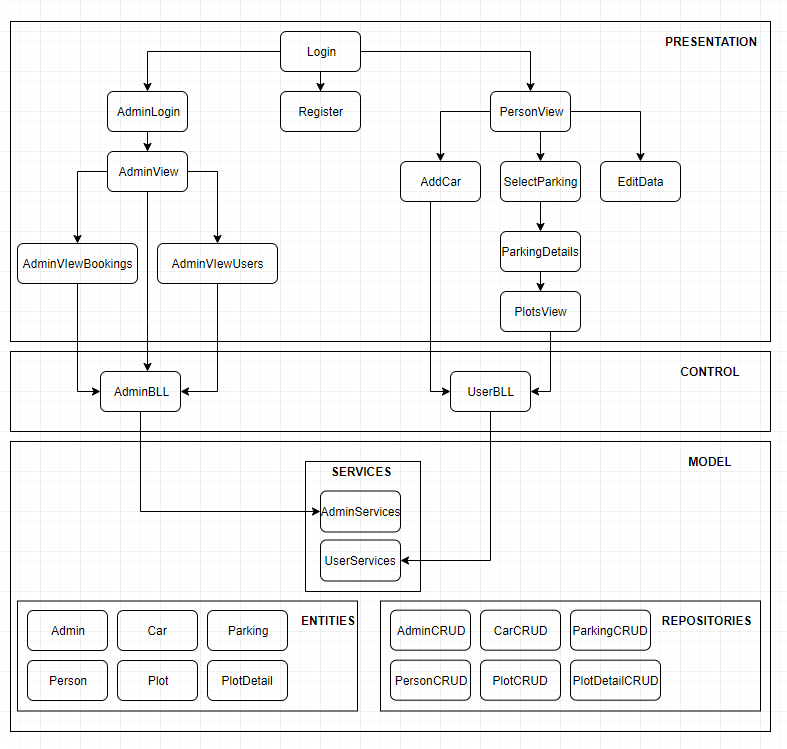
# Architectural Design Refinement

The final package design respects the three principles of package cohesion.

* Reuse – release Equivalence Principle:
  + all the classes in the packages are of the same family and reusable
* Common-reuse Principle:
  + all the classes that are reused together are in the same package
* Common-closure Principle



# Design Model Refinement



# Construction and Transition

# System Testing

JUnit was used for the testing of most user inputs.

# Future improvements

The application could be improved by being deployed on a larger scale, with multiple types of notifications and alerts. For example, if all the plots are full, one could add a parking on a wishlist and be notified when any plot is available. Another improvement would be to be able to cancel the reservation.

# Bibliography

<https://www.slideshare.net/RajivKumar13/uml-39230098>

<http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod>

<https://matthiasnoback.nl/book/principles-of-package-design/>

<https://dzone.com/articles/solid-grasp-and-other-basic-principles-of-object-o>

<https://groups.yahoo.com/neo/groups/domaindrivendesign/conversations/messages/7582>

<https://github.com/utcn-cs-software-design-tudorv-2019/LabInfo>