FoodGram

Analysis and Design Document

Student:Jakab Gyöngyi Anikó

**Group:30233**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 23/04/2019 | 1.0 | First Version | Jakab Gyongyi Aniko |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 5

2.3 Component and Deployment Diagrams 5

III. Elaboration – Iteration 1.2 6

1. Design Model 6

1.1 Dynamic Behavior 6

1.2 Class Design 6

2. Data Model 7

3. Unit Testing 7

IV. Elaboration – Iteration 2 8

1. Architectural Design Refinement 8

2. Design Model Refinement 8

V. Construction and Transition 8

1. System Testing 8

2. Future improvements 8

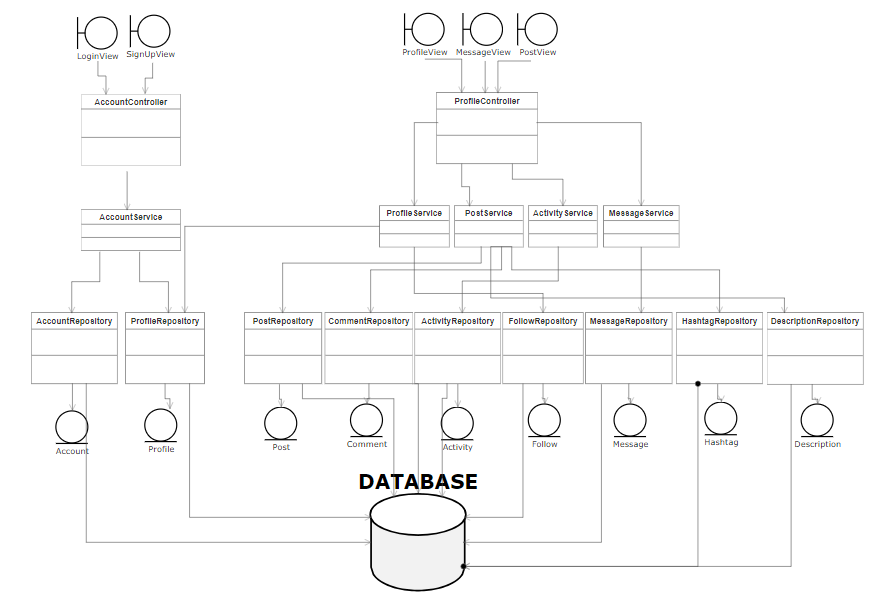
VI. Bibliography 8

# Project Specification

Foodgram is a web application for a community of people that love making and sharing food recipes. This application will make possible for them to share their taste with others or even communicate with their followers.

# Elaboration – Iteration 1.1

# Domain Model

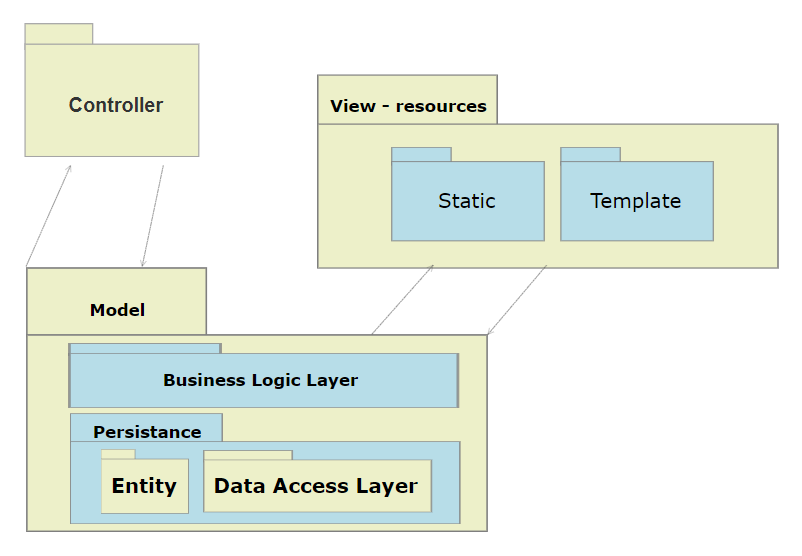


# Architectural Design

## 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. Because of using Spring Boot, the application is separated in element of the Model-View-Controller pattern, however keeping the logical separation of the layered architecture.

## Package Design



## Component and Deployment Diagrams

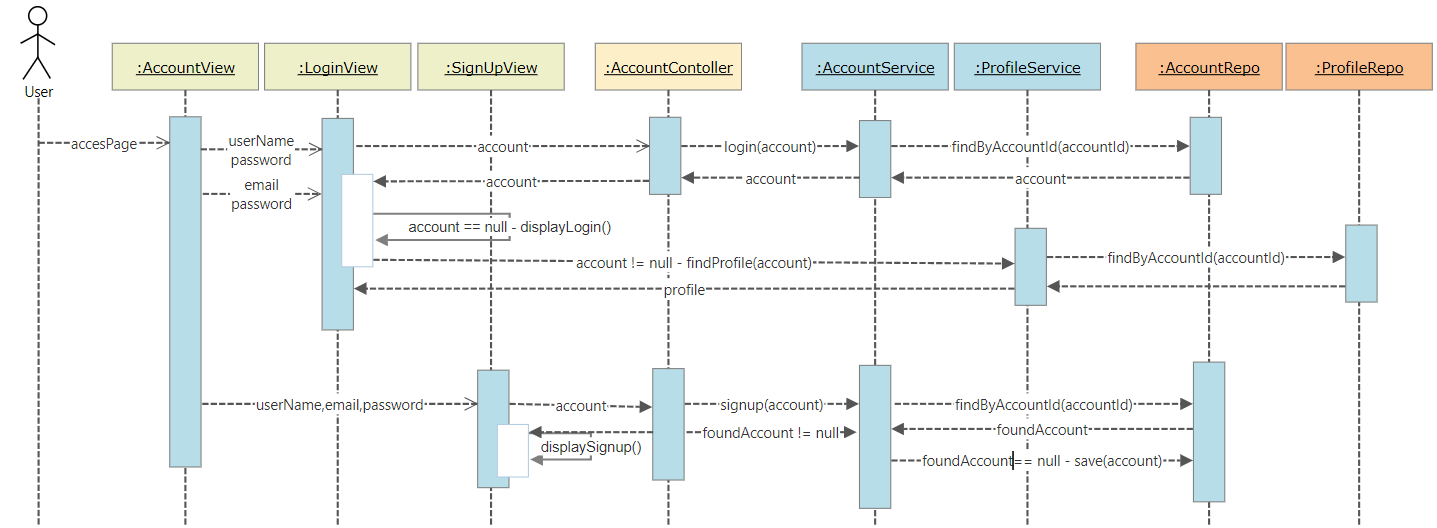
# 

# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior

In the following section it will be presented the account management diagram, meaning login and signup operations and a presentation of communication between service – business logic and data access objects.

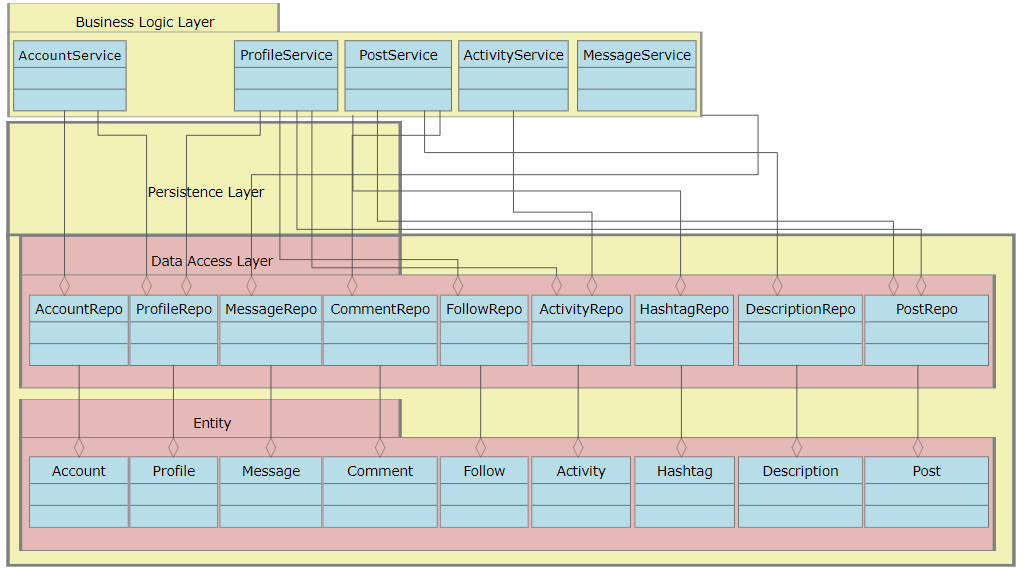


## Class Design

Because of the logical separation of this project, separate class diagrams will be created for each package, the connection between them being presented in the package diagram of previous section.

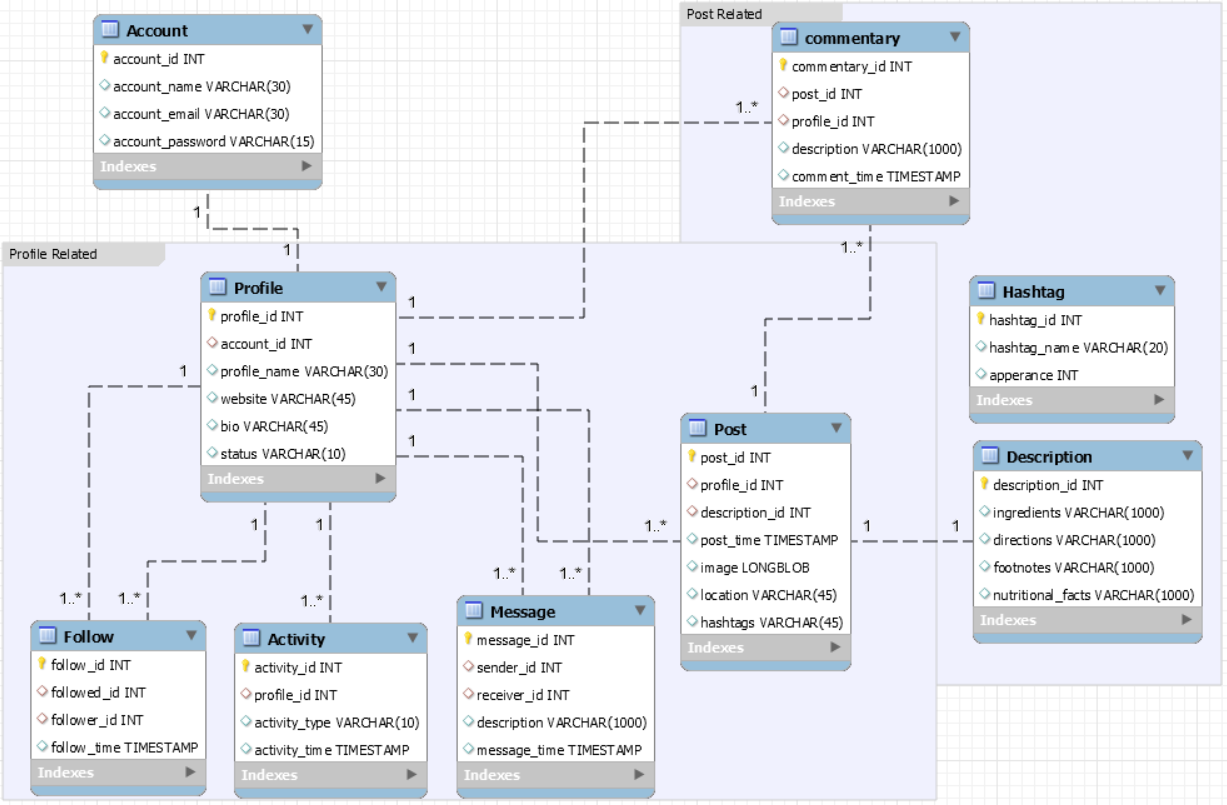
Model package – consisting of: business logic layer and persistence layer: entities and repositories

Controller package – consisting of only 2 controllers – AccountController and ProfileController



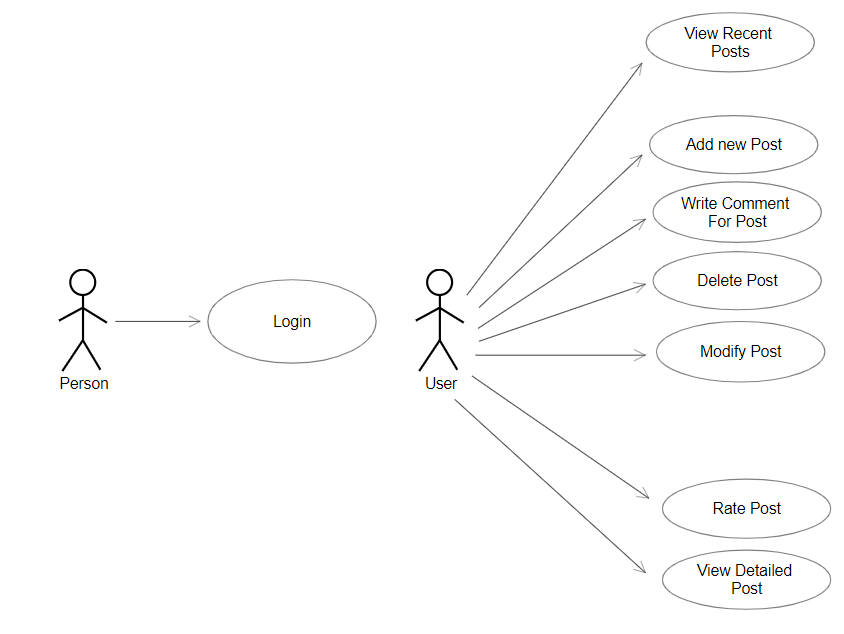
# Data Model

The Data Model of this project represents the entities and relations between them.



# Unit Testing

Testing in case of this project is made using Junit 4, which is integrated in Spring. In the main testing class we injected all repositories and tested them individually for each main operation: save, delete and find by different attribute names. This testing unit is also including some use case testings, for example login and signup operation. All use-case operations need to be tested but there are some of them that require the interface elements too.



# 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.]*

# Design Model Refinement

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

# Construction and Transition

# System Testing

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

# Future improvements

*[Present future improvements for the system]*

# Bibliography