Email Integration Application

Analysis and Design Document

Student: Bolba Raluca Maria

**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 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

# Project Specification

*[Present the project specification]*

# Elaboration – Iteration 1.1

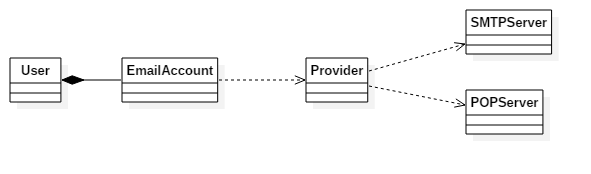
# Domain Model

*[Define the domain model and create the conceptual class diagrams]*

Principalele concepte întâlnite în cadrul acestei aplicații și care vor fi parte a domeniului sunt : utilizatorul, sau clientul care va folosi aplicația, și care va trebui să aibe un nume de utilizator și o parolă pentru a fi recunoscut de sistem și pentru a putea folosi funcțiile acesteia, conturi de email, care vor fi deținute de utilizatori (unul sau mai multe) și care au un tip, furnizori, sau *provideri*, de mail, iar în cazul acestei aplicații îi vom consider pe cei mai comuni, și anume : Yahoo, Gmail și Hotmail. Pentru a efectua anumite operații precum trimitere de email, citire email, trebuie să existe o comunicare cu serverele de email prin așa numitele servere SMTP (Simple Mail Transfer Protocol) și POP (Post Office Protocol). Acestea au configurații diferite pentru fiecare tip de provider în parte, setări cum ar fi numele, numărul portului, dacă se utilizează SSL (Secure Sockets Layer) etc.

Este necesat de asemenea un centru de control sau de administrare a aplicației, care să țină evidența utilizatorilor aplicației și a adreselor de email a acestora care le vor folosi în cadrul aplicației. Tot în acest modul se vor procesa comenzile de adăugare de cont de utilizator, sau de adăugare de adresă de email, caz în care trebuie să i se atribuie adresei și un provider, cu serverele SMTP și POP corespunzătoare astfel încât să se poată efectua operațiile de citire, scriere, ștergere email.

**Conceptual class diagram :**



# Architectural Design

## Conceptual Architecture

*[Define the system’s conceptual architecture; use an architectural style and pattern - highlight its use and motivate your choice.]*

Pattern-ul architectural pe care am ales să îl folosesc pentru această aplicație este pattern-ul MVC (Model View Controller) care, așa cum sugerează numerele, structurează aplicația în 3 categorii : Model, View și Controller. Am ales acest pattern pentru ca fiecare componenta a aplicației (de model, vizualizare sau controller) să fie cât delimitată de restul componentelor.

Partea de *Model* este responsabilă de păstrarea stării curente a aplicației sau a unui set de date. Pe lângă date, care sunt reprezentări ale conceptelor utilizate în aplicație, mai conține și logica care se aplica pe aceste date. Modelul nu are nici o informație cu privire la felul în care sunt reprezentate datele către utilizatori într-un mod grafic. Acesta oferă în schimb Controller-ului metode prin care starea curentă a aplicației poate fi accesată sau modificată.

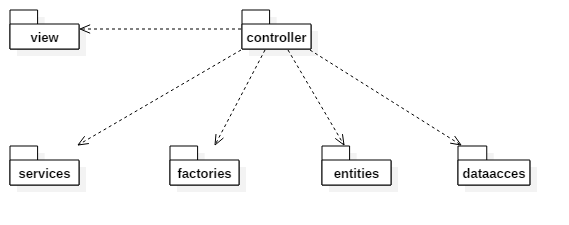
Partea de *View* se ocupă de reprezentarea într-o manieră grafică a datelor primite de la model. Pot exista de exemplu mai multe moduri de reprezentare pentru același set de date dintre care Controller-ul poate alege unul.

Partea de *Controller* este partea ce se ocupă de procesarea cererilor venite de la utilizator, în acest caz ascultătorii de evenimente pentru componentele din interfața grafică, iar datele primite de la utilizator sunt eventual modificate și transmise către partea de model pentru a modifica starea curentă a sistemului. Această parte mai are rolul de a păstra modelul și view-ul sincronizate.

## Package Design

*[Create a package diagram]*

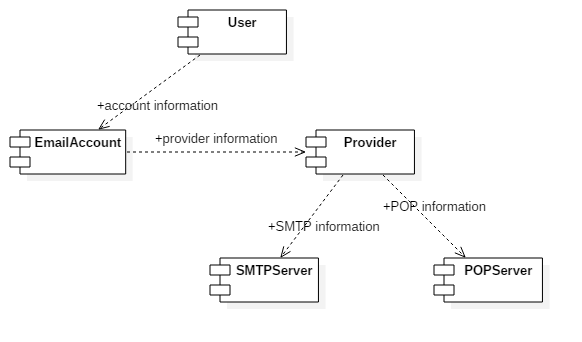
Pachetele corespund pattern-ului architectural MVC conform următoarelor: partea de View corespunde pachetului *view*,partea de Controller corespunde pachetului *controller* iar partea de Model este formată din pachetele *entities, dataaccess, services și factories.*

**

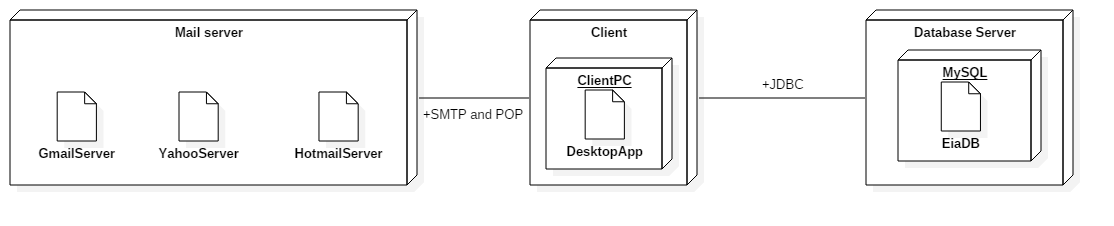
## Component and Deployment Diagrams

*[Create the component and deployment diagrams.]*

*Component diagram :*

**

*Deployment diagram :*



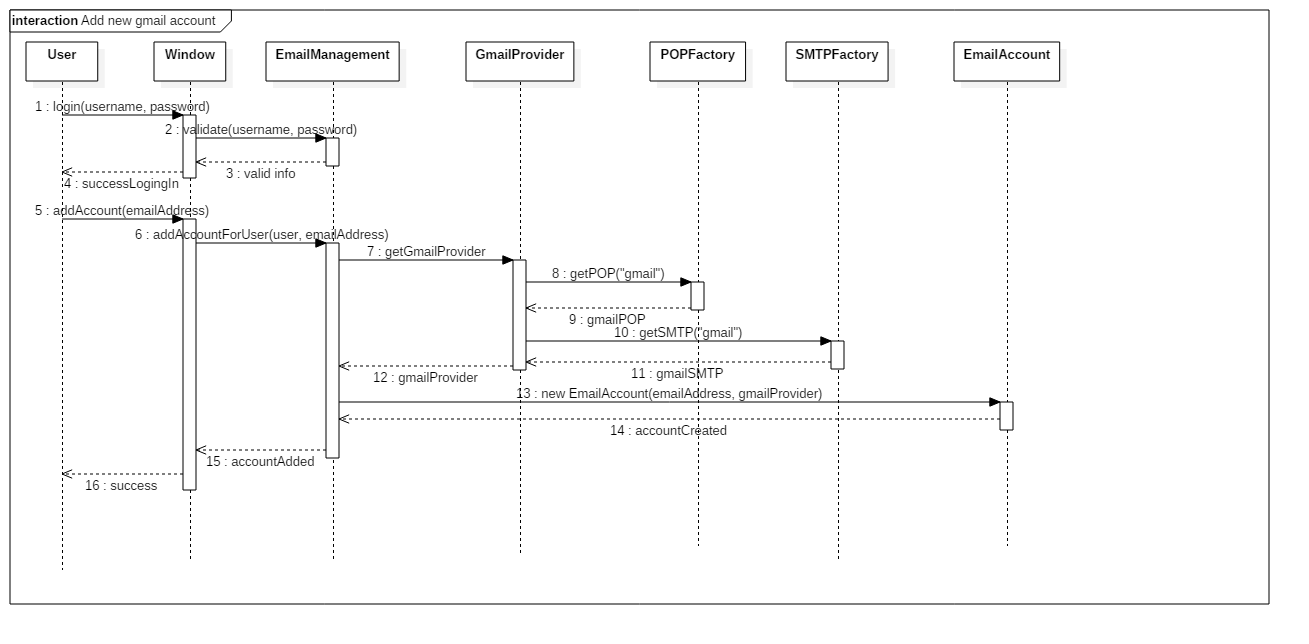
# Elaboration – Iteration 1.2

# Design Model

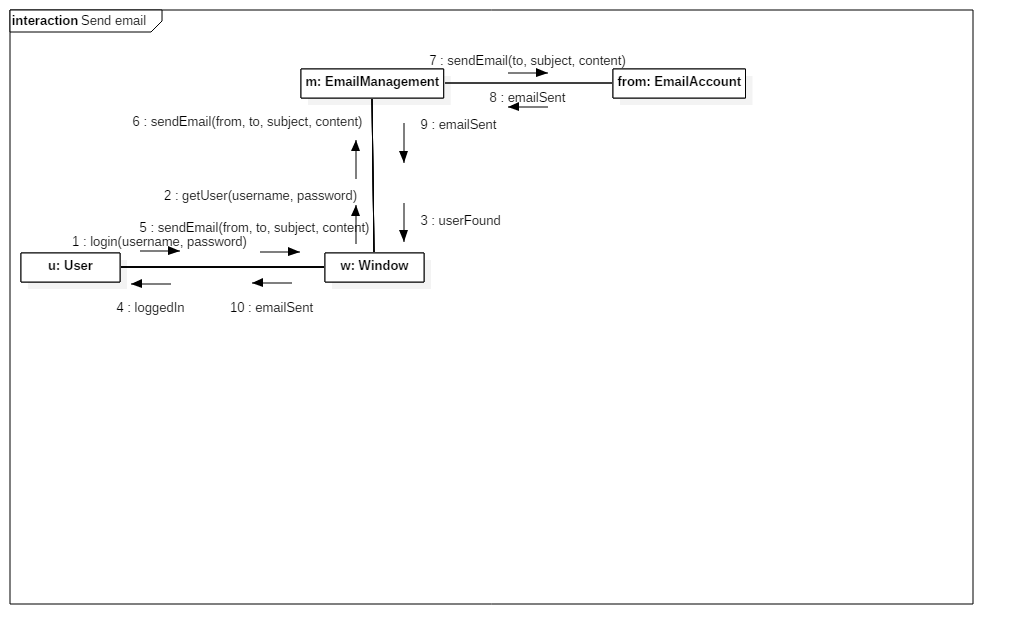
## Dynamic Behavior

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

Diagramă de secvenţă pentru adăugarea de către un utilizator a unei noi adrese de email de tip Gmail:

**

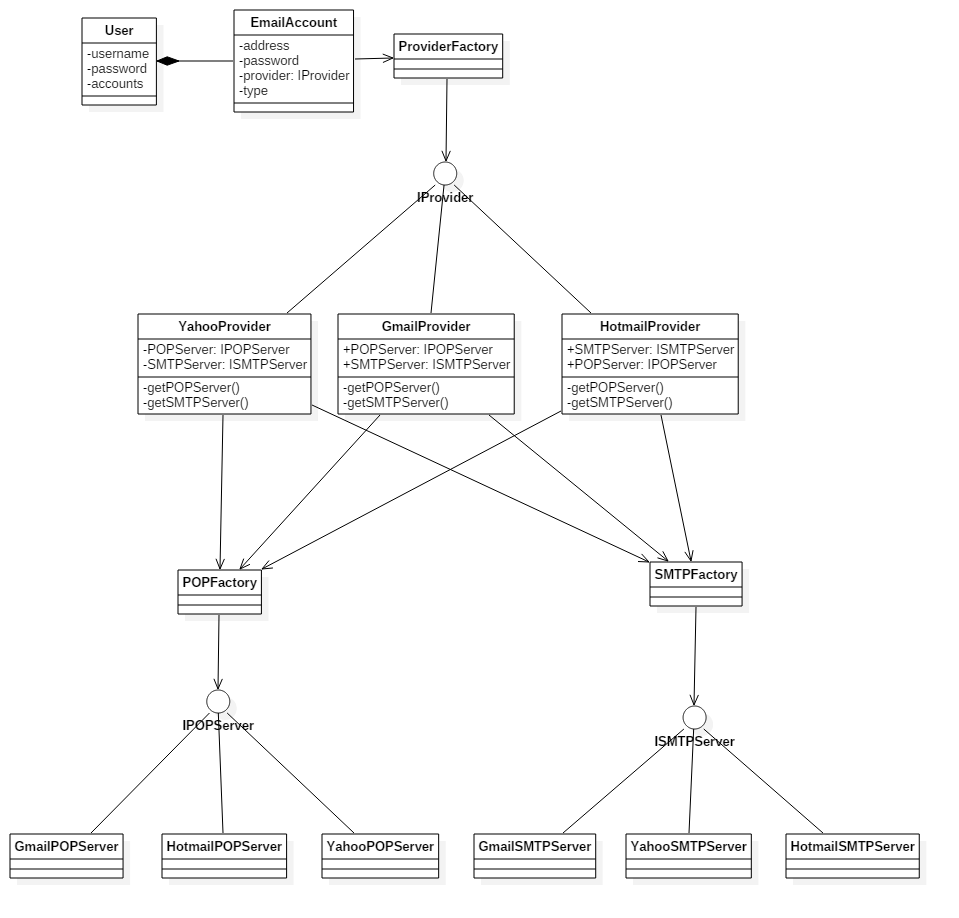
Diagramă de comunicare pentru operația de trimitere email :

**

## Class Design

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

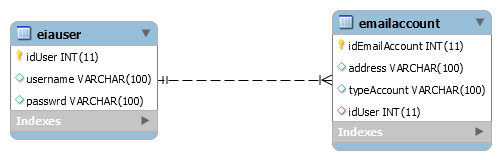
Pentru această aplicație am decis să utilizez design pattern-ul *Factory* pentru crearea unor obiecte de tip *Provider* întrucât acesta poate fi de trei tipuri : Gmail, Yahoo și Hotmail (însă se pot adăuga și alte tipuri de furnizori pentru aplicație)*, SMTPServer* și *POPServer*, întrucât acestea au attribute particulare pentru cei trei tipuri de furnizori precizați*.* Astfel fiecare obiect de tip *Provider* va avea un obiect de tip server SMTP și unul de tip POP în funcție de tipul furnizorului, iar crearea acestor obiecte va reveni în funcția claselor *SMTPFactory și POPFactory*, *Provider* netrebuind să fie conștient de acest proces, el doar folosindu-le în anumite situații.

**

# Data Model

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

În cadrul aplicației voi folosi o bază de date pentru a stoca informații despre conturile utilizatorilor aplicației și despre conturile de email ale acestora. Așadar baza de date constă din două tabele, *EiaUser* care conține informații despre utilizatorii aplicației, informații precum numele de utilizator și parola și tabela *EmailAccount* care stochează date despre adrese de email folosite de utilizator în cadrul aplicației. Informațiile stocate în *EmailAccount* sunt adresa de email, tipul de cont de email (Yahoo, Gmail, Hotmail) și o cheie străina la tabela *EiaUser* care reține id-ul utilizatorului care deține adresa respectivă.

**

# Unit Testing

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

# 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