Analysis and Design Document

Student: Vescan Catalin

**Group: 30235**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <04/04/2018> | <1.0 > | <details> | <Vescan Catalin> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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 6

III. Elaboration – Iteration 1.2 6

1. Design Model 6

1.1 Dynamic Behavior 6

1.2 Class Design 8

2. Data Model 9

3. Unit Testing 9

IV. Elaboration – Iteration 2 10

1. Architectural Design Refinement 10

2. Design Model Refinement 11

V. Construction and Transition 12

1. System Testing 12

2. Future improvements 12

VI. Bibliography 12

# Project Specification

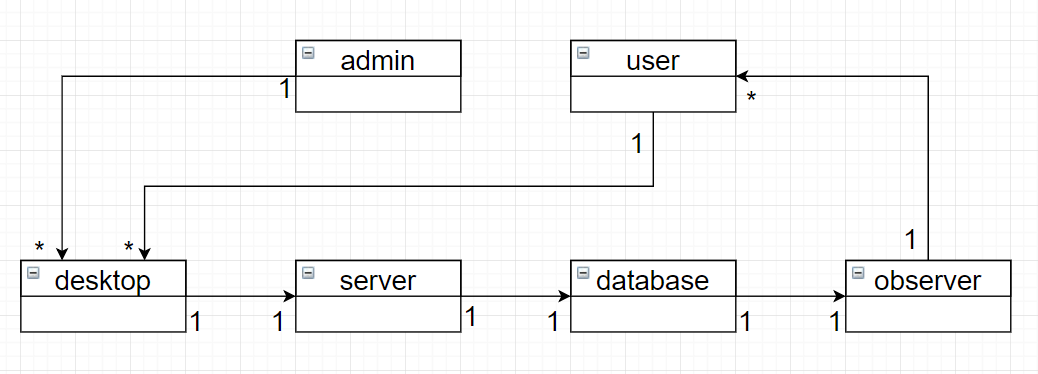
Aceasta aplicatie vine cu scopul de a pastra evidenta in legatura cu numarul de ore lucratoare ale angajatilor unei firme, dar totodata poate contribui si la diferite statistici.

Pentru utilizarea aplicatiei, angajatul trebuie sa deschida aplicatia, unde ii va aparea o fereastra de interfata grafica pentru logare, unde trebuie sa introduca datele de identificare care constau intr-un nume de utilizator care desigur va fi unic si o parola, iar in momentul in care acesta isi incheie activitatea se va deloga.

De asemenea, se poate loga si angajatorul ca si administrator, care beneficiaza de anumite operatii cum ar fi adaugarea sau stergerea unui anumit angajat din baza de date, sau modificarea informatiilor acestora.

# Elaboration – Iteration 1.1

# Domain Model



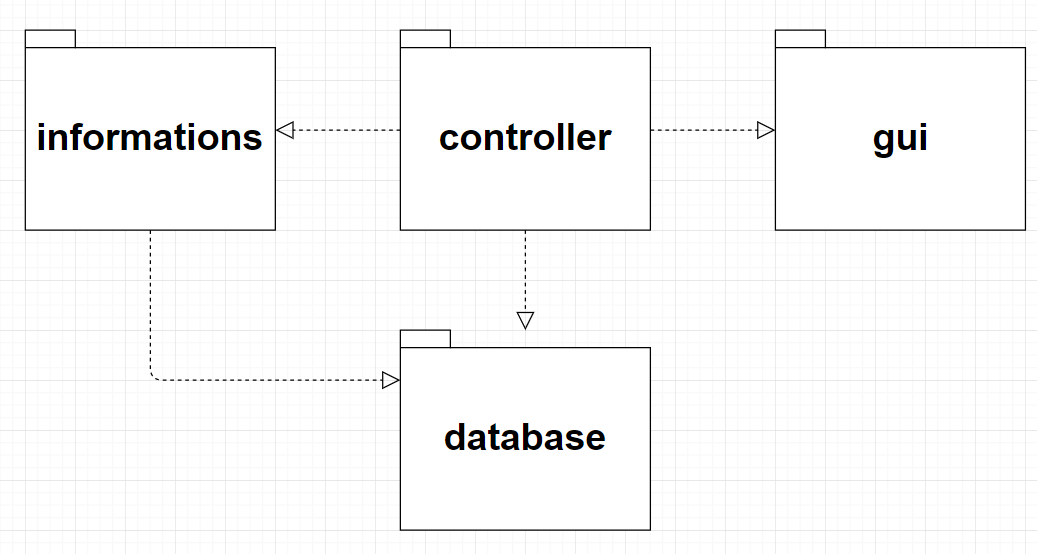
# Architectural Design

## Conceptual Architecture

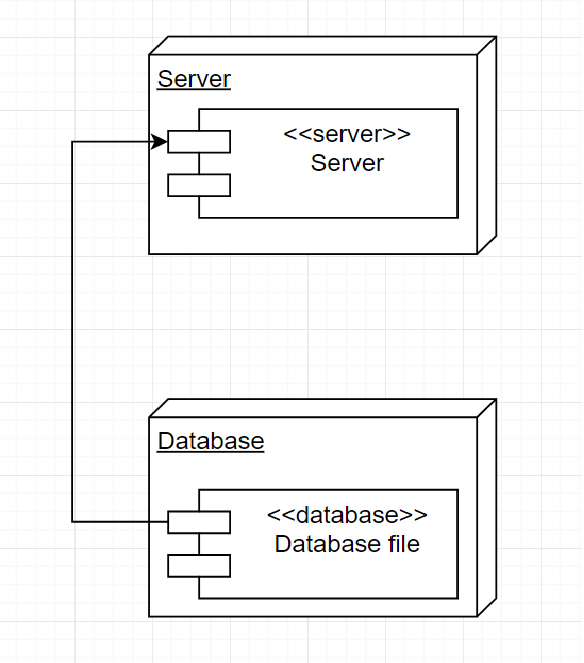
Pentru implementarea acestui proiect am folosit ca pattern arhitectural MVC (Model View Controller), care are ca scop impartirea si organizarea claselor, iar ca pattern de desing am folosit Observer, care are rolul de a trimite notificari atunci cand se schimba informatiile. De asemenea, in proiect am folosit ca pattern creational Factory Pattern care implementeaza mecanisme de creare a obiectelor, iar ca pattern structural am folosit Proxy Pattern care actioneaza asupra interfetei.

Pentru conexiunea la baza de date am folosit ca framework Hibernate.

## Package Design



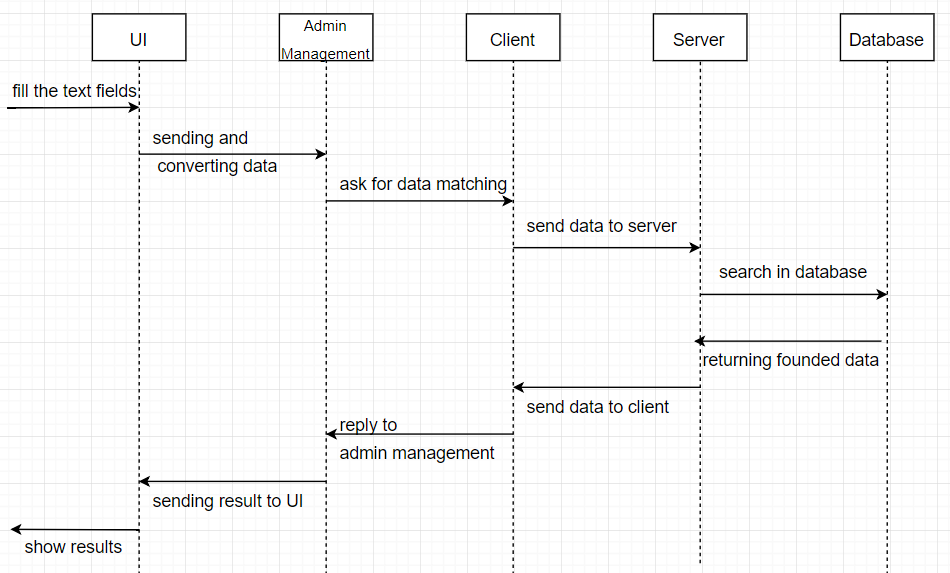
## Component and Deployment Diagrams



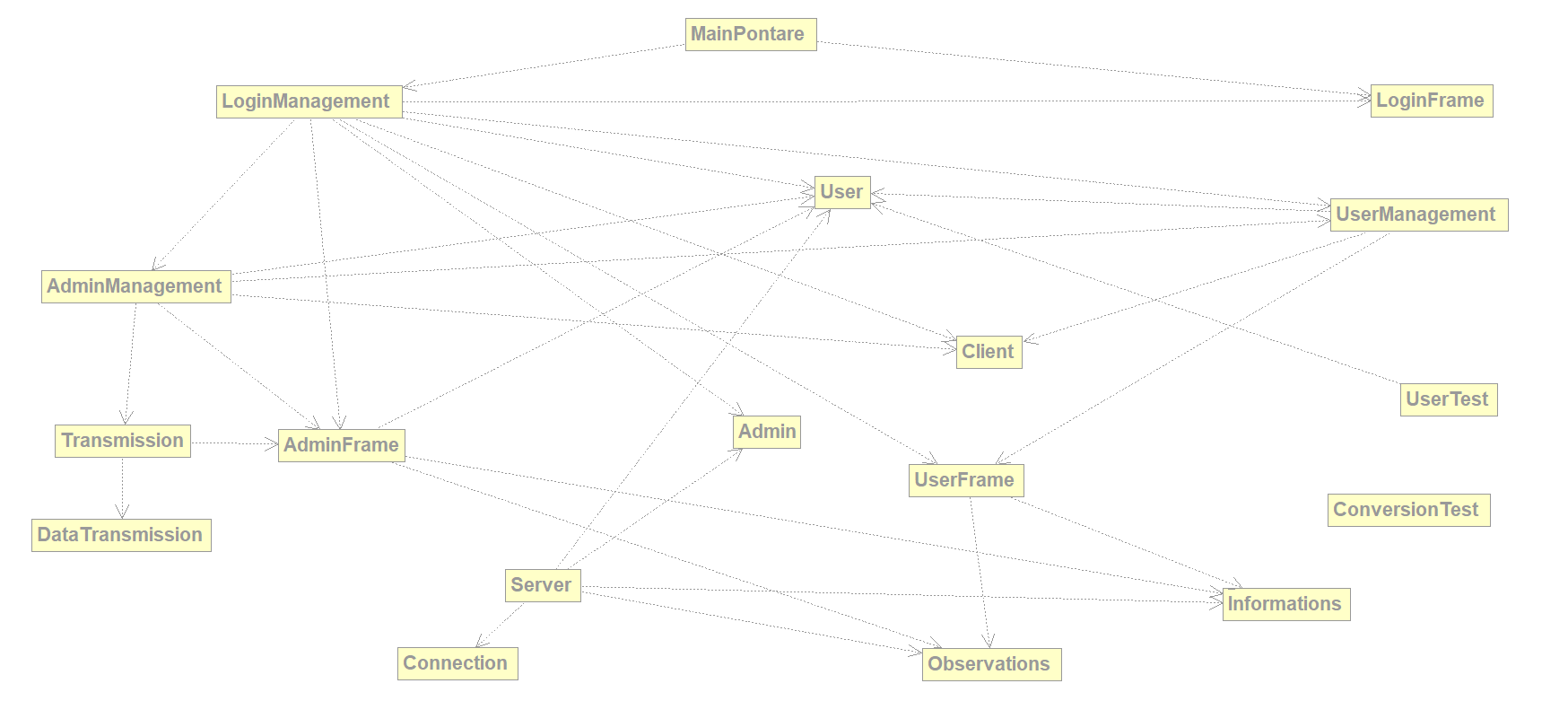
# Elaboration – Iteration 1.2

# Design Model

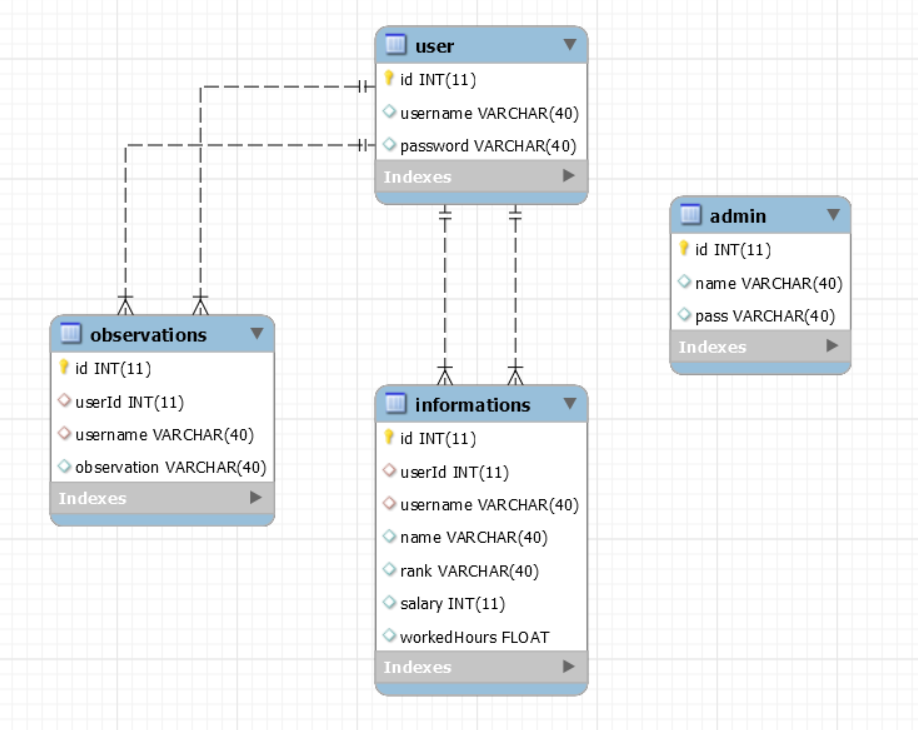
## Dynamic Behavior



## Class Design



# Data Model

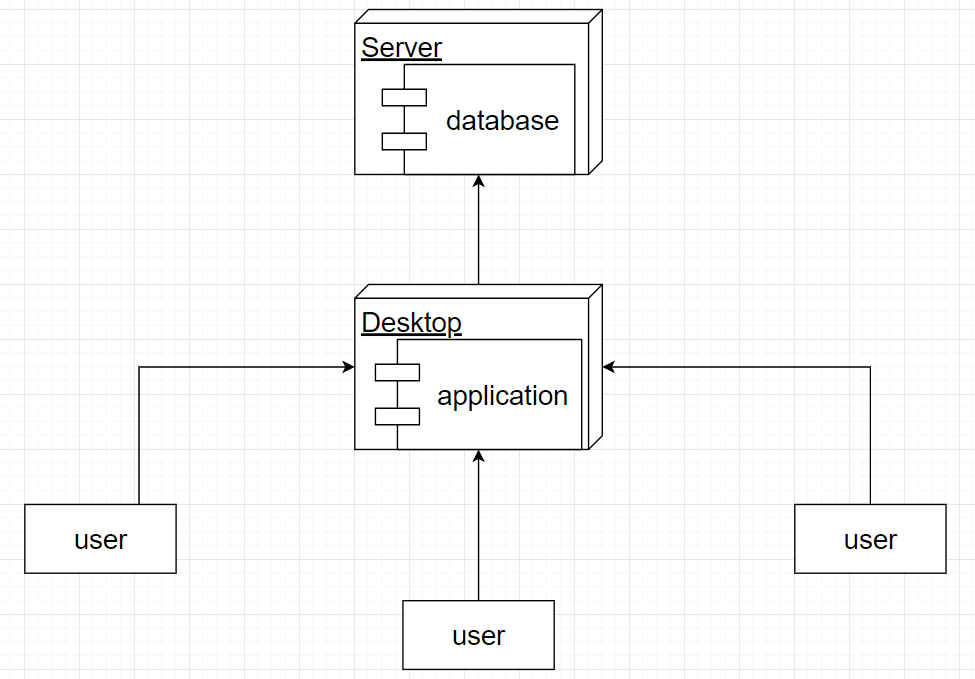
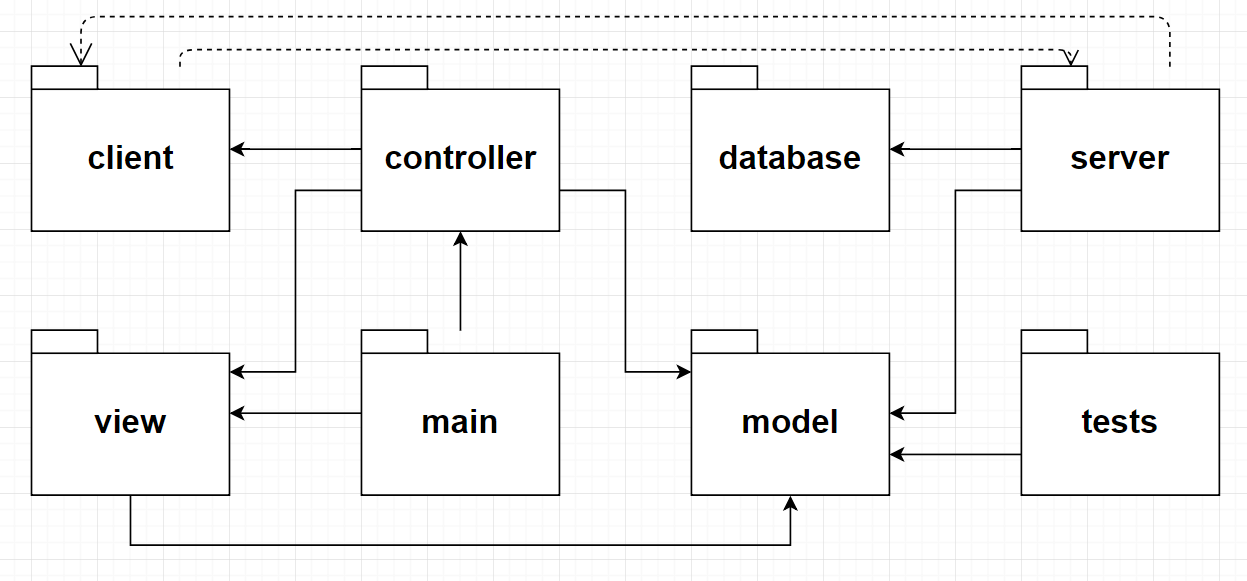


# Unit Testing

Pentru testarea functiilor utilizate in proiect am folosit JUnitTest.

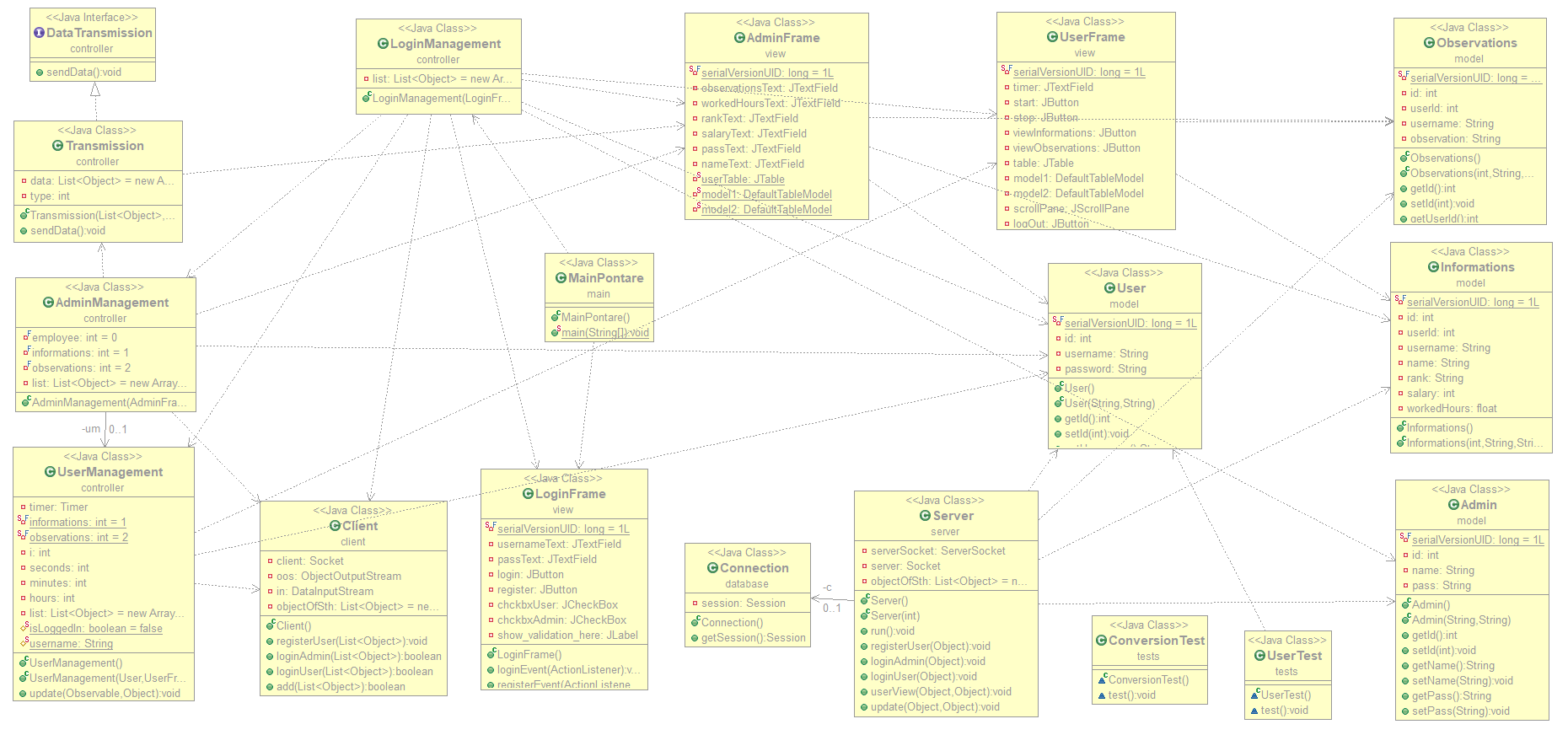
# Elaboration – Iteration 2

# Architectural Design Refinement



In aceasta sectiune am adaugat diagramele de componente si relatiile dintre pachete, obtinute in urma adaugarii si efectuarii testelor de rigoare.

# Design Model Refinement



# Construction and Transition

# System Testing

Asupra bazei de date s-au efectuat teste pentru verificarea conexiuni, mai exact daca s-a efectuat cu succes o conextiune la baza de date, iar apoi se afiseaza un mesaj de succes, in caz contrat se va afisa un mesaj de eroare, dar s-au efectuat si teste pentru verificarea validitatii datelor din baza de date.

# Future improvements

Aplicatia poate fi dezvoltata astfel incat sa li se acorde anumite beneficii angajatilor, precum o sectiune in care sa comunice in timp real cu ceilalti angajati, sau chiar in cazul in care doresc sa lucreze de acasa.

De asemenea, aplicatia se poate dezvolta pentru a raspunde mult mai repede si mai prompt actiunilor cerute, si sa fie mai dezvoltata din punct de vedere al securitatii.

# Bibliography

<https://www.google.ro/search?rlz=1C1AVFC_enRO781RO781&tbm=isch&q=model+view+controller&chips=q:model+view+controller,online_chips:mvc+pattern&sa=X&ved=0ahUKEwjsh4uwp6DaAhWCIpoKHZKfC6AQ4lYIJygB&biw=1536&bih=759&dpr=1.25#imgrc=rBSncauPecDYKM>:

<https://stackoverflow.com/>

<https://www.youtube.com/>