Employee management application.

Analysis and Design Document

Student:Filip Andrei

**Group:30239**

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

Designing and implementing a client-server application for a company administration which

can be used by workers, supervisors by an administrator.

Supervisors can perform the following operations:

* View the tasks that need to be done
* Assign ratings to a worker
* Give a worker a pay rise
* Assign tasks to a worker

Workers can perform the following operations:

* View their own task
* View their rating (given by the supervisor)

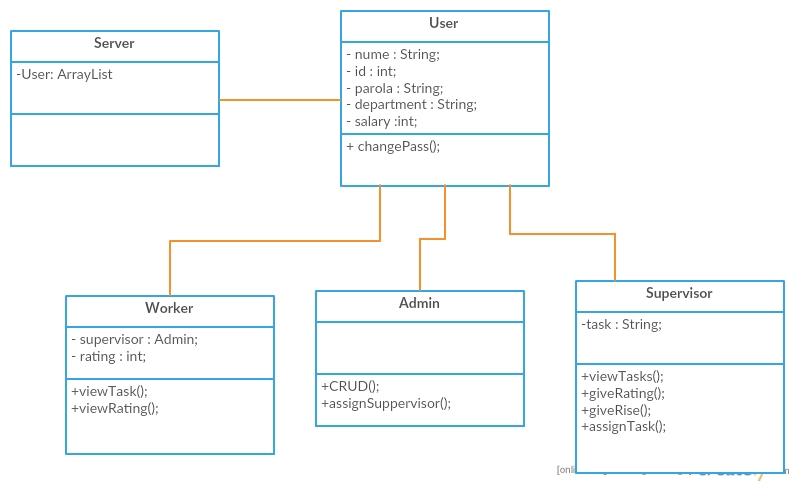
Administrators can perform the following operations:

* CRUD operations on workers, supervisors
* Assign supervisor to worker

The system will notify the worker when he receives a new rating/pay rise/new supervisor/new task.

# Elaboration – Iteration 1.1

# Domain Model

**

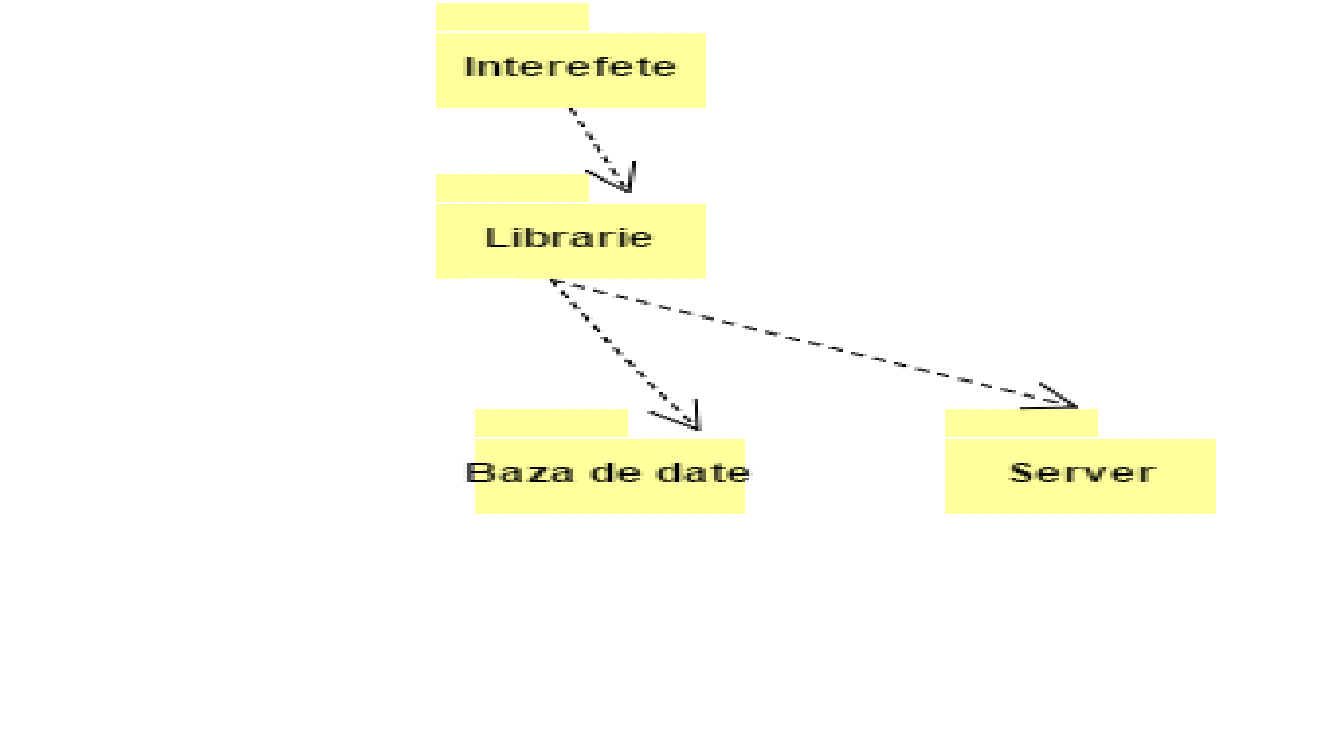
# Architectural Design

## Conceptual Architecture

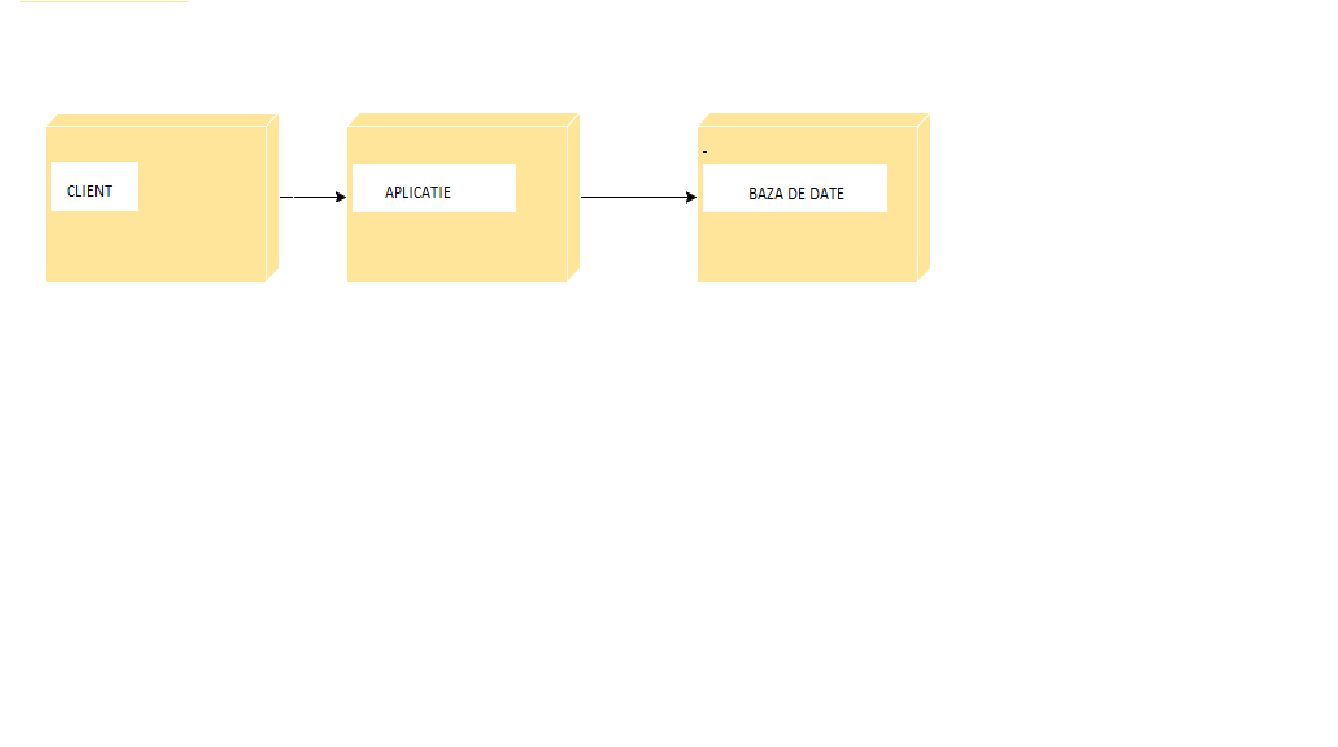
Aplicatia este bazata pe stilul architectural Client-server. Stilul se bazeaza pe utilizarea unui server ce ofera servicii clientilor. Fluxul de control al serverului este independent de cel al clientilor, cu exceptia sincronizarii gestiunii actiunilor ce necesita modificarea datelor aflate pe server. Stilul permite utilizarea optima a unui volum mare de date.

Stilul Multi-Layer este de asemenea folosit. Sistemul presupune mai multe nivele de abstractizare. Cererile merg “in jos “, nivelele de sus primind notificari. Acest stil este benefic pentru reutilizarea straturilor, si delimitarea clara a sarcinilor fiecarui nivel.

## 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]*

## 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.]*

# 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