FastDelivery: Online Catering Service

Analysis and Design Document

Student: Virtic Andrei

**Group: 30227**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 07/04/2017 | 1.0 | Project Specification, Elaboration-Iteration | Virtic Andrei |
| 26/05/2017 | 1.1 | Elaboration Iteration 2 | Virtic Andrei |
|  |  |  |  |
|  |  |  |  |

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

Proiectul meu consta in crearea unei aplicatii client-server, desktop de catering. Aplicatia va pune la dispozitia clientilor, feluri de mancare diferite, deserturi, bauturi si sosuri .Actorii care vor interacţiona cu sistemul sunt: Administrator, Client, Vizitator, Furnizor.

Administratorul poate adauga/modifica/sterge conturi utilizatori, gestiona meniul, gestiona comenzi, modifica pagina principala.

Clientul poate face log in/log out, poate modifica datele contului , vizualiza meniul, adauga felurile de mancare disponibile in cosul de cumparaturi si finaliza comanda.

Vizitatorul poate doar vizualiza meniul disponibil si sa creeze un nou cont, devenind astfel client.

Delivery poate vizualiza si gestiona comenzile plasate de catre clienti.

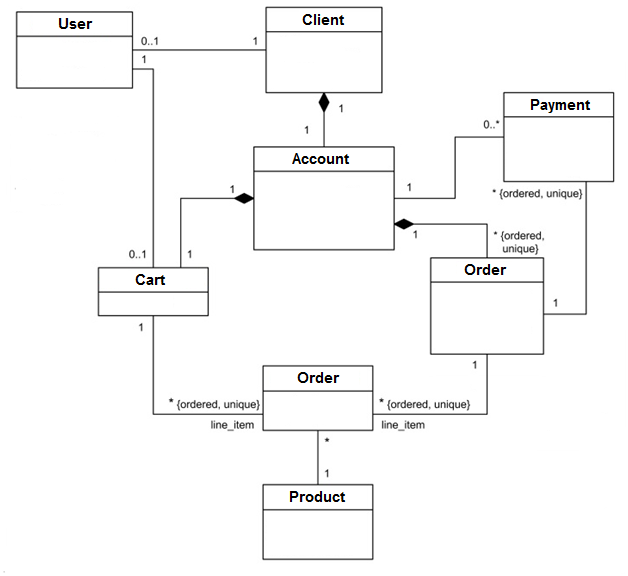
Aplicatia online de catering permite tuturor clientilor sa isi creeze un cont, cu ajutorul caruia o sa poata adauga diferite feluri de mancare in cosul de cumparaturi, mai apoi sa le comande daca doresc.

Site-ul pune la dispozitia oricarui utilizator un motor de cautare pe baza numelui unui fel de mancare. Pentru o cautare mai eficienta in partea stanga a paginii se afla cateva filtre care pot fi aplicate.

# Elaboration – Iteration 1.1

# Domain Model

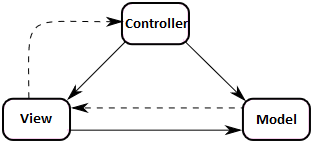
Diagrama de clase:



# Architectural Design

## Conceptual Architecture

Sistemul foloseste modelul de arhitectura Model View Controller. Modelul manipulează operațiunile logice și de utilizare a informației. Componentei View îi corespunde reprezentarea grafică, sau mai bine zis, exprimarea ultimei forme a datelor: interfața grafică ce interacționează cu utilizatorul final. Rolul său este de a evidenția informația obținută până ce ea ajunge la controller. Cu ajutorul Controller-ului putem controla accesul la aplicația noastră. Pot fi fișiere, scripts sau programe, in general orice tip de informație permisă de interfață. În acest fel putem diversifica conținutul nostru de o formă dinamică și statică, în același timp.



## Package Design

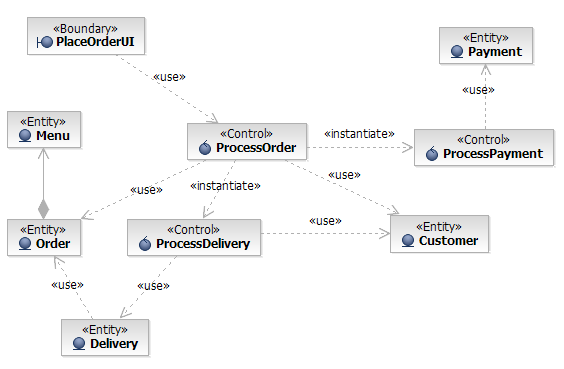
## 

Fiecare pachet are urmatoarele responsabilitati:

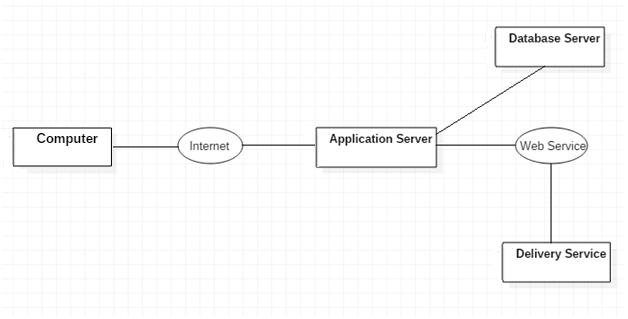
* pachetul Model manipulează datele folosite de aplicatie.
* pachetul View are rolul de a evidenția informația obținută până ce ea ajunge la controller.
* pachetul Control controleaza accesul la aplicația noastră. Pot fi fișiere, scripts sau programe, in general orice tip de informație permisă de interfață.

## Component and Deployment Diagrams

Diagrama de componente:



Deployment diagram:

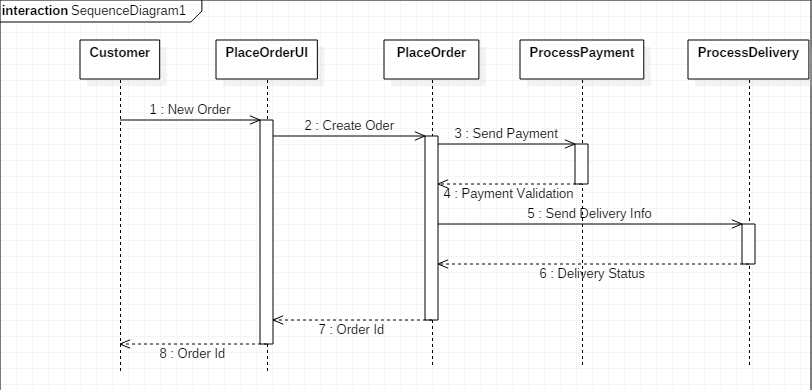


# Elaboration – Iteration 1.2

# Design Model

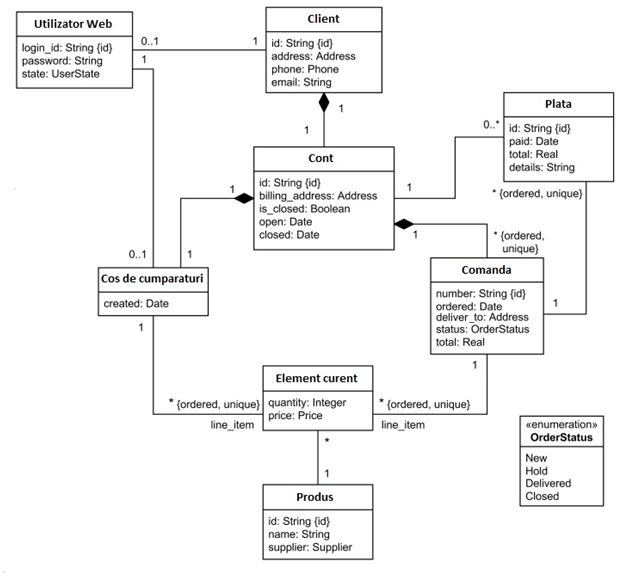
## Dynamic Behavior

Sequence diagram:

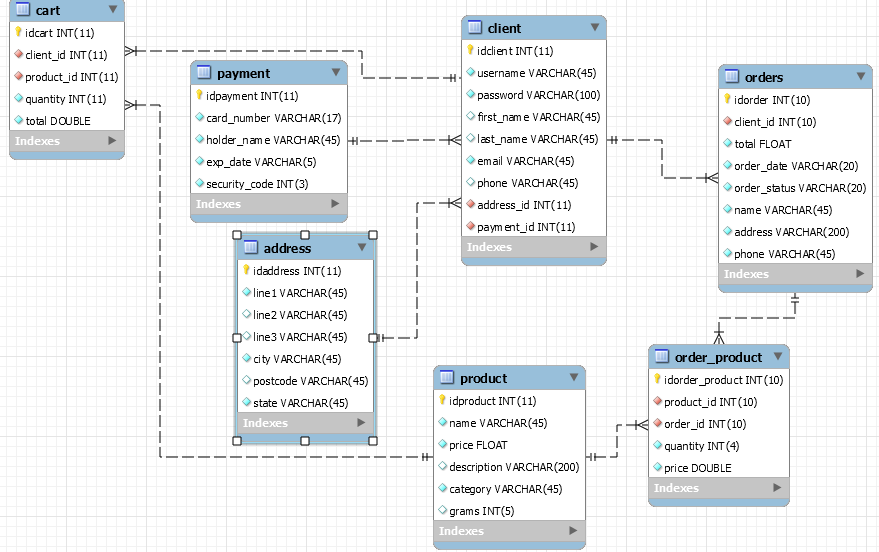


Communication diagram:

## Class Design



# Data Model

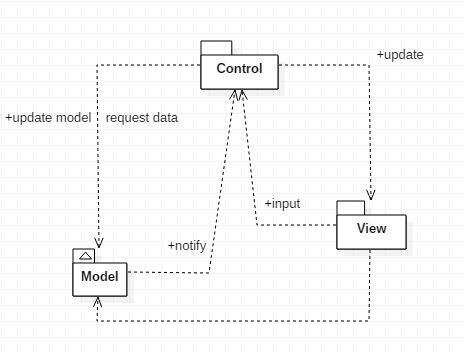


# Unit Testing

# Elaboration – Iteration 2

# Architectural Design Refinement

Sistemul foloseste modelul de arhitectura Model View Controller. Modelul manipulează operațiunile logice și de utilizare a informației. Componentei View îi corespunde reprezentarea grafică, sau mai bine zis, exprimarea ultimei forme a datelor: interfața grafică ce interacționează cu utilizatorul final. Rolul său este de a evidenția informația obținută până ce ea ajunge la controller. Cu ajutorul Controller-ului putem controla accesul la aplicația noastră.

******

# 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