<Assignment Name>

Analysis and Design Document

Student: Bălan Oana-Alexandra

**Group: 30236**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

[Application description]

*Realizarea unei aplicații care să fie folosită pentru a vinde bilete la o conferință ce are loc pe perioada a trei zile.*

# Functional Requirements

*[Present the functional requirements]*

*Administratorul va putea realiza următoarele operații:*

* *CRUD pe bilete(modificarea prețului unui bilet, modificarea numărului de bilete disponibile)*
* *CRUD pe casieri*
* *Modificarea numărului MAX\_CAPACITY*
* *Raport cu numărul de bilete vândute de un casier*
* *Raport cu încasările pe o zi*
* *Raport încasări totale bazate pe biletele vandute*

*Casierul va putea:*

* *Vânzare de bilete - creat fisier .txt pe disk pentru fiecare bilet, cu detalile biletului*
* *Vizionarea istoricului biletelor vândute de către el ordonate după timp și dată (ultimele bilete vândute fiind primele)*

# Non-functional Requirements

*[Discuss the non-functional requirements for the system]*

* *Se va folosi patternul arhitectural:* ***Layers***
* *Aplicație desktop folosind JAVA/C#*
  + *Frameworkuri sugerate:*
    - *JAVA - Swing*
    - *C# - Windows Presentation Foundation (WPF)*
* *Datele vor fi stocate într-o bază de date*
  + *MySql/ Sql Server Express/ Oracle Express*
* *Accesul la baza de date se va face folosind* 
  + - *Un pattern de acces de date*
  + *Frameworks/Libraries*
    - *JDBC - Java*
    - *Ado.Net - C#*

2. Use-Case Model

*[Create the use-case diagrams and provide one use-case description (according to the format below).*

*Use-Case description format:*

*Use case: <use case goal>*

*Level: <one of: summary level, user-goal level, sub-function>*

*Primary actor: <a role name for the actor who initiates the use case>*

*Main success scenario: <the steps of the main success scenario from trigger to goal delivery>*

*Extensions: <alternate scenarios of success or failure>*

*]*

**

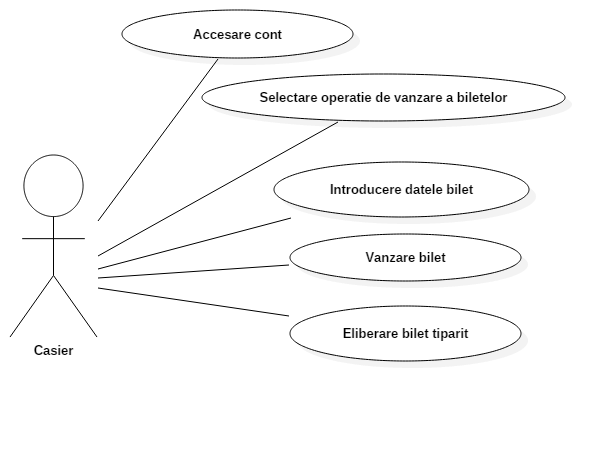
*Use case: Vânzare bilet*

*Level: Casierul trebuie să se logheze în contul său, să aleagă operația de vânzare a biletului, să introducă datele biletului, să aștepte răspuns de la baza de date și dacă s-a făcut vânzarea, să elibereze biletul*

*Primary actor: Casier*

*Main success scenario:Acces permis în cont, introducere date despre bilet corecte, vânzare bilet,eliberare bilet*

*Extensions: Casierul ar putea să vândă mai multe bilete deodată, să furnizeze mai multe servicii precum un raport al acțiunilor lui la sfârșitul zilei, să se dorească vânzarea unui anumit număr de bilete de către el*

**

3. System Architectural Design

**3.1 Architectural Pattern Description**

*[Describe briefly the used architectural patterns.]*

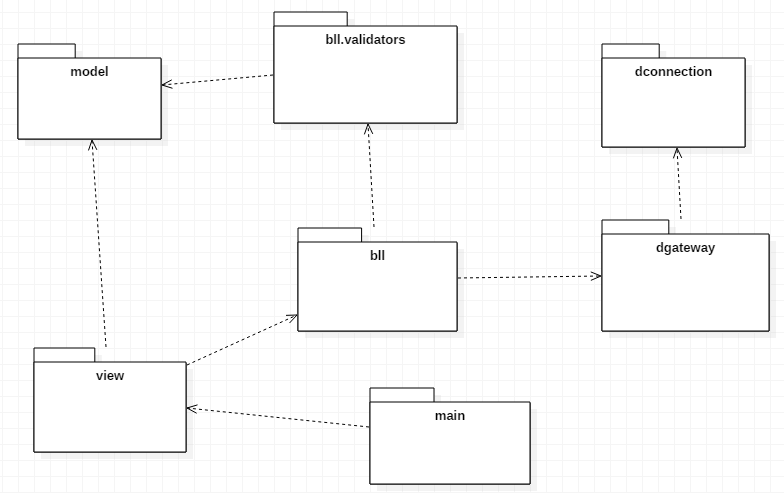
*Componentele din modelul de arhitectură în strat sunt organizate în straturi orizontale, fiecare strat realizând un anumit rol în cadrul aplicației (de ex. Logica de prezentare sau logica de afaceri). Deși modelul de arhitectură în strat nu specifică numărul și tipurile de straturi care trebuie să existe în model, cele mai multe arhitecturi stratificate constau în patru straturi standard: prezentare, afaceri, persistență și bază de date. Aplicațiile mai mici pot avea doar trei straturi, în timp ce aplicațiile de afaceri mai mari și mai complexe pot conține cinci sau mai multe straturi.*

*Fiecare strat al modelului arhitectural stratificat are un rol specific și o responsabilitate specifică în cadrul aplicației.*

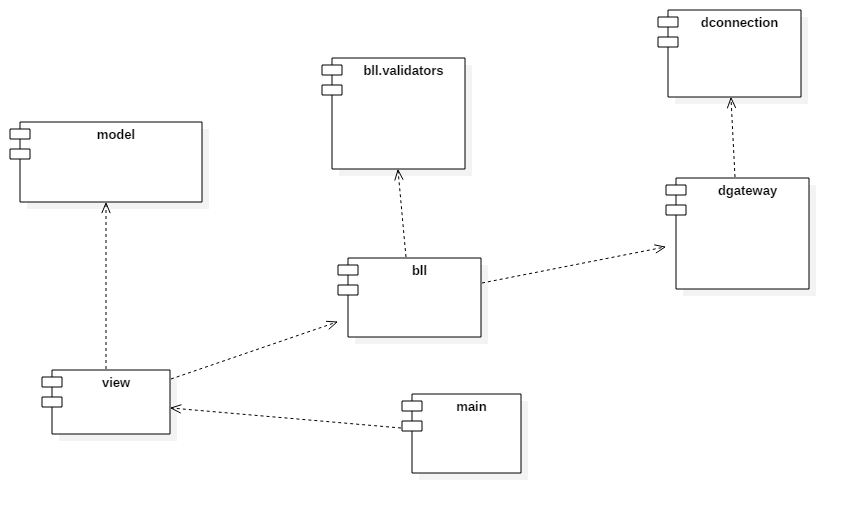
**3.2 Diagrams**

*[Create the system’s conceptual architecture; use architectural patterns and describe how they are applied. Create package, component and deployment diagrams]*

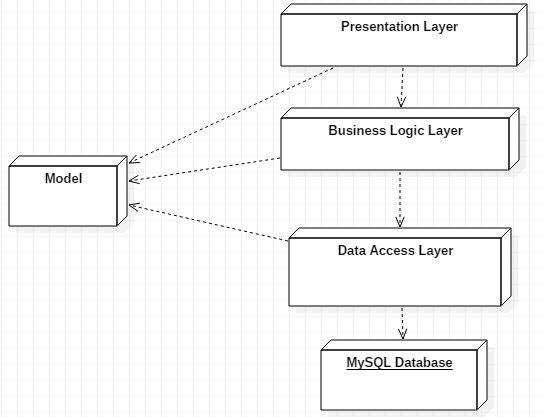
*Diagramă de pachete:*

**

*Diagramă de componente:*

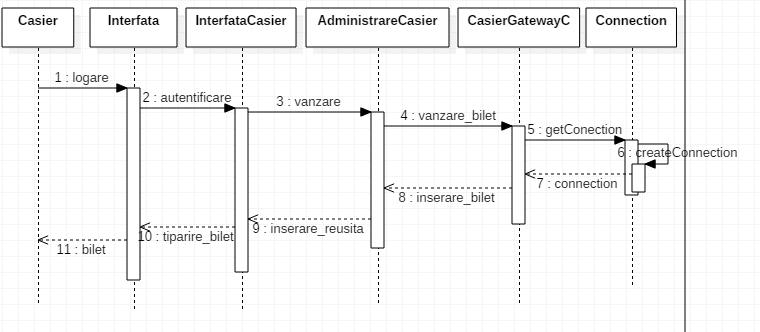
**

*Diagramă de implementare:*

**

4. UML Sequence Diagrams

*[Create a sequence diagram for a relevant scenario.]*



1. Class Design

**5.1 Design Patterns Description**

*[Describe briefly the used design patterns.]*

*Un design pattern folosit este Interpreter Pattern. Un Interpreter Pattern spune că "să definească o reprezentare a gramaticii unei limbi date, împreună cu un interpret care folosește această reprezentare pentru a interpreta propozițiile în limbaj".*

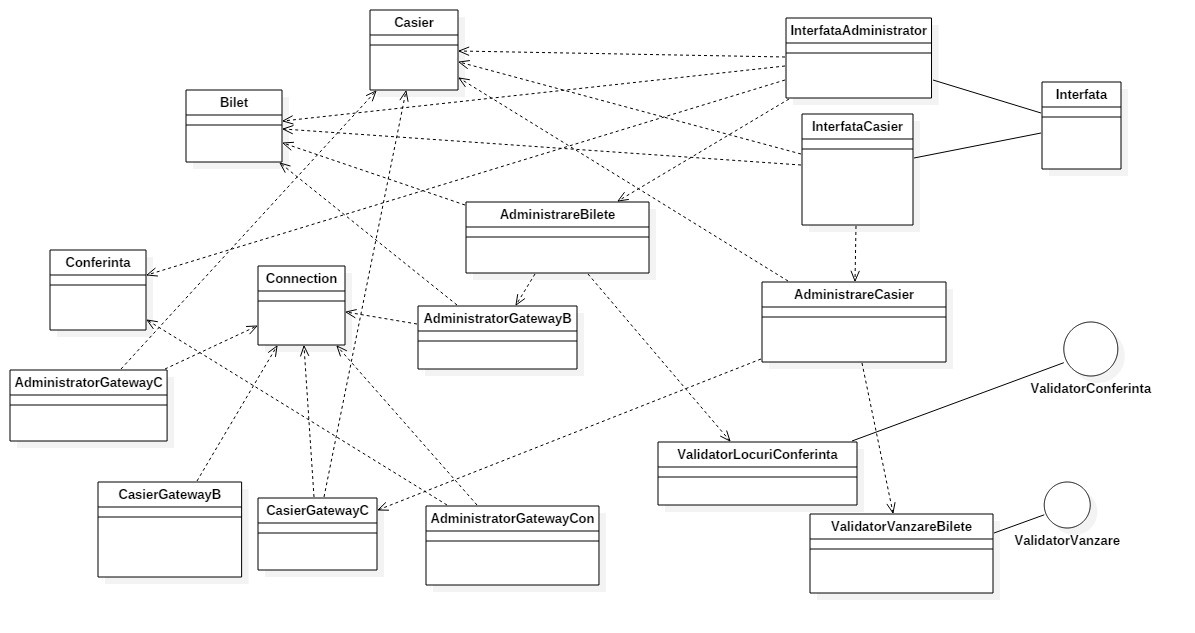
*Practic, modelul interpretului are o zonă limitată în care poate fi aplicată. Putem discuta modelul Interpreterului numai în termeni de gramatici formale, dar în acest domeniu există soluții mai bune și de aceea nu este folosit frecvent.*

*SQL Parsing folosește Interpreter Pattern, iar eu am folosit MySQL.*

*Un alt design pattern este Table Data Gateway, care păstrează toate interogările SQL pentru a accesa un singur tabel sau vizualizare: selectează, inserează, actualizează și șterge.*

**5.2 UML Class Diagram**

*[Create the UML Class Diagram and highlight and motivate how the design patterns are used.]*



6. Data Model

*[Present the data models used in the system’s implementation.]*

*Modele de date folosite sunt clasele: Bilet, Casier și Conferință, care sunt și tabele în baza de date.*

7. System Testing

*[Present the used testing strategies (unit testing, integration testing, validation testing) and testing methods (data-flow, partitioning, boundary analysis, etc.).]*

1. Bibliography

*<https://www.javatpoint.com/>*

*<https://martinfowler.com/eaaCatalog/dataMapper.html>*

*<https://martinfowler.com/eaaCatalog/tableDataGateway.html>*