<Travel Agency Management System >

Analysis and Design Document

Student:Deac Denisa Bianaca

**Group:30431**

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

The application for managing the activity in a tourism agency will be designed to allow employees to add, modify, and delete vacations, add and modify client information, and reserve vacations for clients. The application will store information about clients and vacations in a database that will be updated periodically based on the information provided in XML files by operators who collaborate with the agency.

# Elaboration – Iteration 1.1

# Domain Model

The domain model represents the conceptual classes and their relationships within the system. The classes are organized into packages based on their responsibilities within the system.

1. The domain model for the client-server application will consist of the following entities:

* Vacation: contains information about the vacation such as the destination, type of vacation, price, and availability
* Client: contains information about the client such as name, contact details, and reservation history
* Operator: contains information about the operator such as name and contact details
* XML file: contains information about hotels and trips availability provided by the operator

# Architectural Design

## Conceptual Architecture

A layered architecture is a software architecture pattern in which the system is divided into logical layers, each with a specific responsibility and interacting with the layers above and below it through well-defined interfaces. The layers are organized in a hierarchical manner, and each layer provides a set of services to the layer above it while consuming services from the layer below it.

The layered architecture for the tourism agency management system consists of the following layers:

1. Presentation Layer: This layer provides the user interface for the system and interacts with the user. It is responsible for handling user input and presenting data to the user. The presentation layer communicates with the application layer to request data and perform actions on behalf of the user.
2. Business Layer: This layer contains the business logic of the system. It is responsible for processing user requests, managing the state of the system, and coordinating the activities of the lower layers. The application layer communicates with the domain layer to retrieve and manipulate data.
3. Data Access Layer: This layer is responsible for interacting with the database or any other external data source. It provides an abstraction layer between the domain objects and the data source, which allows for easy switching of the data source without affecting the rest of the system.
4. Model : responsible for the entities of the project

**Diagram

Description automatically generated**

## Package Design

The package diagram shows how the system is organized into logical packages or modules, each containing a set of related classes. In this case, the packages include Presentation, BusinessLogic, DataAccess, Entities, and Infrastructure. The diagram shows the dependencies between the packages, which represent the flow of data and control between them.

*Diagram

Description automatically generated*

## Component and Deployment Diagrams

*[Create the component and deployment diagrams.]*

The component diagram shows the different components that make up the system and their relationships, while the deployment diagram shows how these components are deployed on hardware or software infrastructure. In this case, the components include the Presentation layer, Business Logic layer, Data Access layer, and Database. The deployment diagram shows how these components are deployed on servers and clients, with the Presentation layer deployed on the client side and the other layers deployed on the server side.

# Diagram Description automatically generated

# A picture containing diagram Description automatically generated

# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior

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

***SEQUENCE DIAGRAM*** *–* scenario for adding a new vacation of type Cruise by illustrating the Strategy design pattern used.

*Diagram

Description automatically generated*

Scenario for creating a new client with encrypted password:

*Diagram

Description automatically generated*

***COMMUNICATION DIAGRAM:***

Scenario for deleting a vacation and deleting all the related reservations for the given vacation.

Graphical user interface, diagram, application

Description automatically generated

Scenario for creating e reservation and notifying the two observers for creating a receipt and sending an email:

Diagram

Description automatically generated

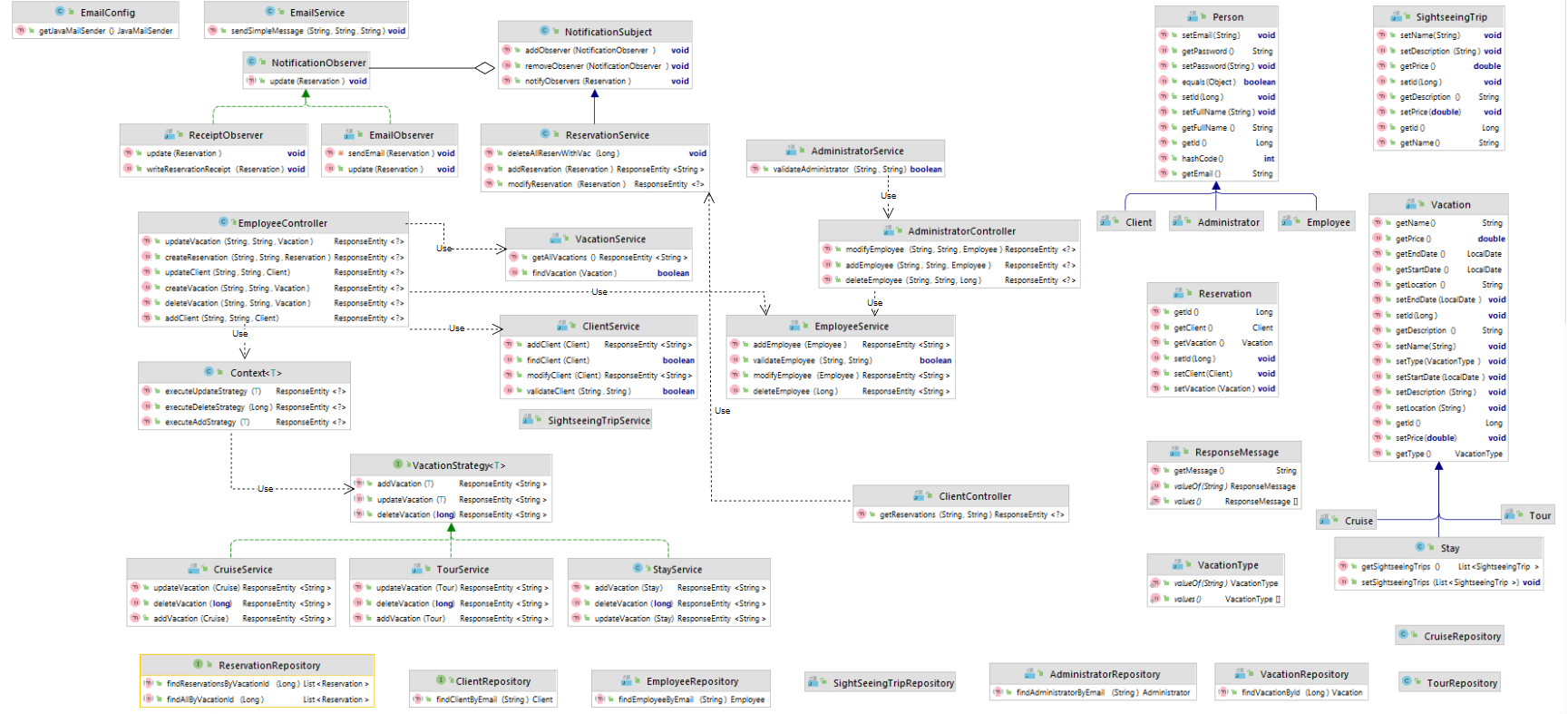
## Class Design

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

The following class diagram represents the main dependencies between classes, along with the design patterns used. First, there is the observer pattern which is connected with the action of creating a reservation existent in ReservationService. This class is the subject of the notification, and it notifies the 2 observers, EmailObserver and ReceiptObserver. The first one sends an email to the client with the details of the reservation and the second generates a receipt in a txt format. The email is sent using JavaMailSender and all the needed configurations are set in the EmailConfig class.

The second design pattern used is the Strategy Design pattern. The VacationStrategy interface defines methods for adding, updating and deleting a vacation. The class has a generic type such that, when implemented, is easier to handle the vacation types.

The concrete strategies are represented by the service classes for the 3 types of vacations where the implementation of the methods are made. The context class handles the type and the calls for each strategy.

**

# Data Model

The data model for this project consists of the following entities:

* Person: which can be of three types: Administrator, Employee and Client, all being users in the application and having stored their name and credentials.
* Vacation: which can be of three types: Cruise, Tour and Stay.
* Sightseeing\_trip: each Stay vacation has a list of sightseeing tips. This entity represents one of this trips and stores significant details.
* Vacation\_sightseeing\_trip: since a stay can have multiple sightseeing trips, there exists a one to many relationship between them. This relationship is illustrated in this table.
* Reservation: each client can book a vacation, representing the reservation entity having as foreign keys the id of the client and the id of a vacation.

Graphical user interface, application

Description automatically generated

# Unit Testing

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

For testing the system, I used validation through postman application, but also 2 unit testing for 2 of the most complex cases. The code makes good use of response messages, all sored in an Enum class, used along with the correspondent Http response.

*Graphical user interface, text, application

Description automatically generated*

The test units are done for deleting a vacation and modifying a reservation.

Graphical user interface, application

Description automatically generated

# 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