Your Tickets

Analysis and Design Document

Student: Bianca-Elena Dondas

**Group: 30238**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 04/apr/2019 | 1.0 | Project Specification and Elaboration I | Bianca Elena Dondas |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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 5

III. Elaboration – Iteration 1.2 5

1. Design Model 5

1.1 Dynamic Behavior 5

1.2 Class Design 5

2. Data Model 5

3. Unit Testing 5

IV. Elaboration – Iteration 2 6

1. Architectural Design Refinement 6

2. Design Model Refinement 6

V. Construction and Transition 6

1. System Testing 6

2. Future improvements 6

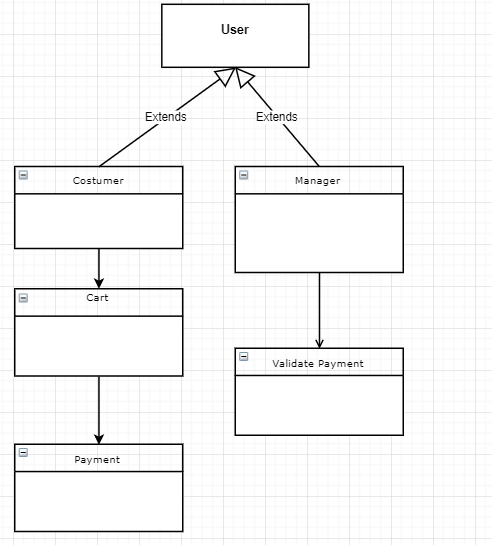
VI. Bibliography 6

# Project Specification

# Music is an art form, a cultural activity and an entertainment form. Music is found in every known culture, past and present, varying widely between times and places, and it exists since prehistory. Over time music has evolved and has become increasingly polar, musical genres have become more and more, as well as concerts. Before the internet exists to going to a concert, required a big effort. For example, you should find out where the concert is, what date of the concert is and where you can get a ticket. Now you simply google a band and find the tour dates and tickets to their concerts which you can buy online. “Your Tickets” is an app where you can do this but not only with a band. You can choose more bands, add it to your list and for each one of them you can get their tour dates, buy tickets and even access their store to buy t-shirts, posters, CDs, etc.

# Elaboration – Iteration 1.1

# Domain Model



# Architectural Design

## Conceptual Architecture

Client–server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests. Examples of computer applications that use the client–server model are Email, network printing, and the World Wide Web.

The client-server characteristic describes the relationship of cooperating programs in an application. The server component provides a function or service to one or many clients, which initiate requests for such services. Servers are classified by the services they provide. For example, a web server serves web pages and a file server serves computer files. A shared resource may be any of the server computer's software and electronic components, from programs and data to processors and storage devices. The sharing of resources of a server constitutes a service.

Whether a computer is a client, a server, or both, is determined by the nature of the application that requires the service functions. For example, a single computer can run web server and file server software at the same time to serve different data to clients making different kinds of requests. Client software can also communicate with server software within the same computer. Communication between servers, such as to synchronize data, is sometimes called or server-to-server communication.

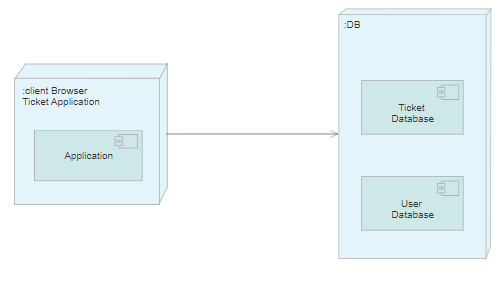
In general, a service is an abstraction of computer resources and a client does not have to be concerned with how the server performs while fulfilling the request and delivering the response. The client only has to understand the response based on the well-known application protocol, i.e. the content and the formatting of the data for the requested service.

Clients and servers exchange messages in a request–response messaging pattern. The client sends a request, and the server returns a response. This exchange of messages is an example of inter-process communication. To communicate, the computers must have a common language, and they must follow rules so that both the client and the server know what to expect. The language and rules of communication are defined in a communications protocol. All client-server protocols operate in the application layer. The application layer protocol defines the basic patterns of the dialogue. To formalize the data exchange even further, the server may implement an application programming interface (API). The API is an abstraction layer for accessing a service. By restricting communication to a specific content format, it facilitates parsing. By abstracting access, it facilitates cross-platform data exchange.

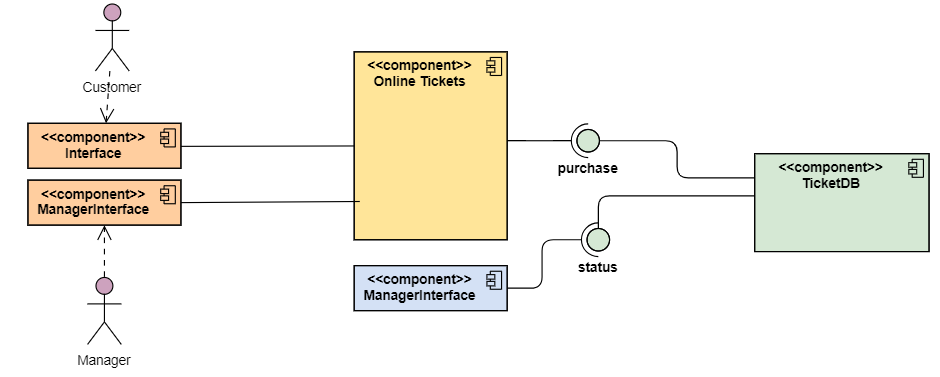
A server may receive requests from many distinct clients in a short period of time. A computer can only perform a limited number of tasks at any moment, and relies on a scheduling system to prioritize incoming requests from clients to accommodate them. To prevent abuse and maximize availability, server software may limit the availability to clients. Denial of service attacks are designed to exploit a server's obligation to process requests by overloading it with excessive request rates.

## Component and Deployment Diagrams

Deployment Diagram



Component Diagram

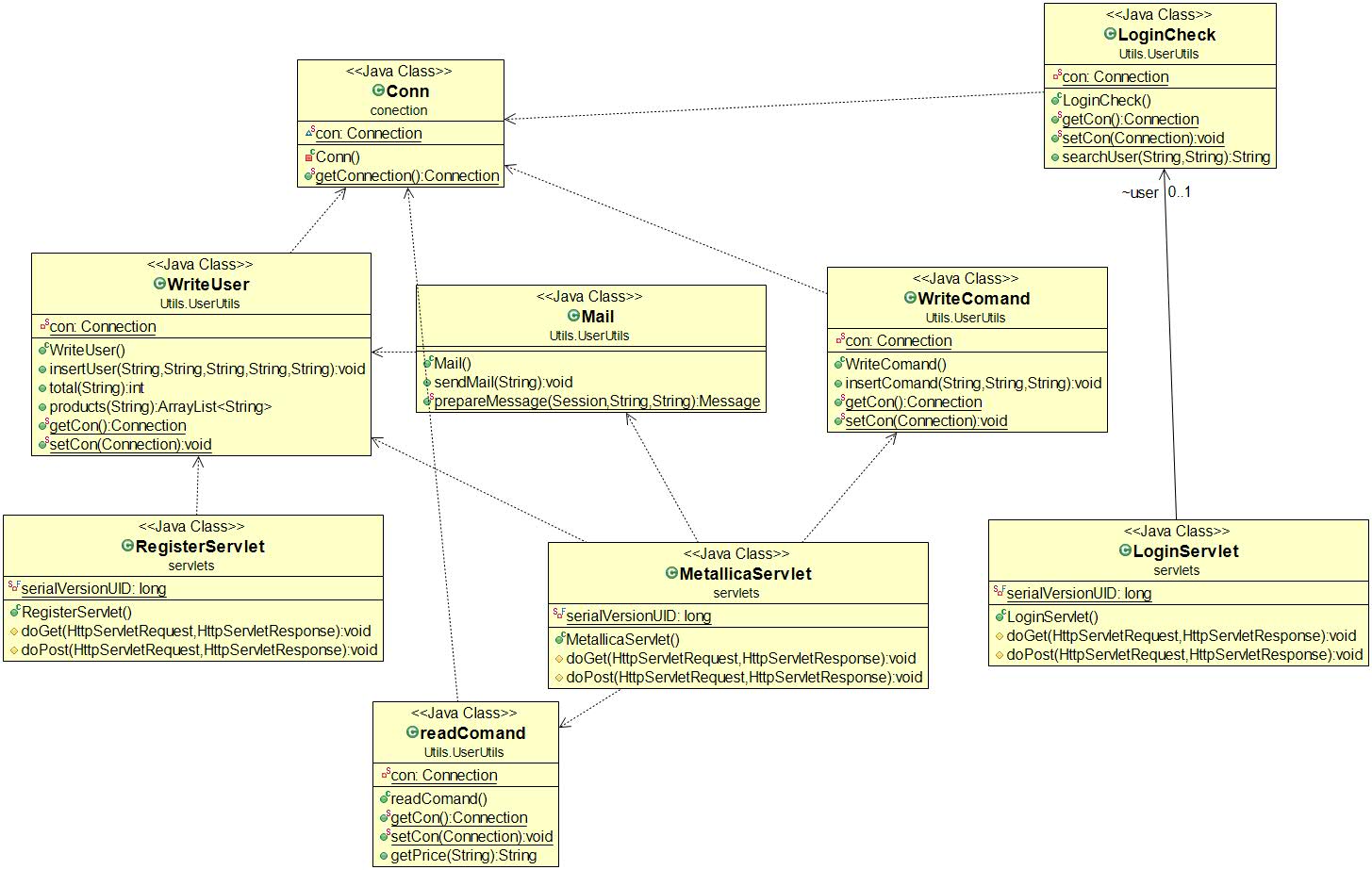


# Elaboration – Iteration 1.2

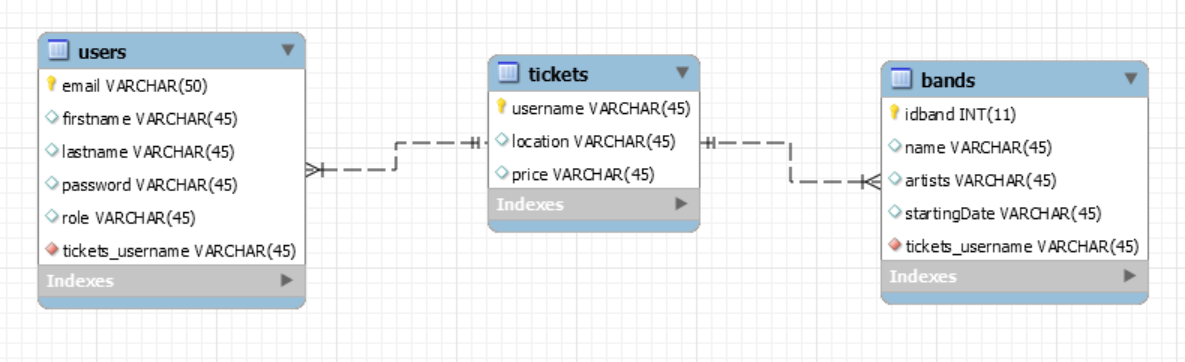
# Design Model

## Dynamic Behavior

## Class Design



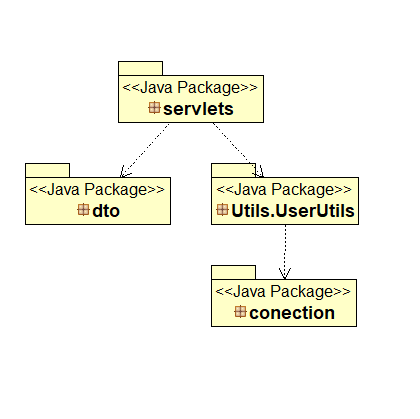
# Data Model



# Unit Testing

# Elaboration – Iteration 2

# Architectural Design Refinement



# Design Model Refinement

# Construction and Transition

# System Testing

*[Describe how you applied integration testing and present the associated test case scenarios.]*

# Future improvements

* Paying with credit card
* Chat for users
* Notify a user when a band he likes is in his town

# Bibliography