**YeOrada**

System Design

12.04.2020

Ahmet Can Terlikçi

Ali Haydar Konuk

Yaren Çoşkun

Fatih Mehmet İdgü

Prepared for

SOFT3102 Software Development Practice



Table of Contents

[1. Introduction 1](#_Toc37549136)

[1.1. Purpose of the System 1](#_Toc37549137)

[1.2. Design Goals 1](#_Toc37549138)

[1.3. Definitions, Acronyms, and Abbreviations 2](#_Toc37549139)

[1.4. References 2](#_Toc37549140)

[2. Current Software Architecture 2](#_Toc37549141)

[3. Proposed Software Architecture 3](#_Toc37549142)

[3.1. Overview 3](#_Toc37549143)

[3.2. System Decomposition 3](#_Toc37549144)

[3.3. Hardware Software Mapping 4](#_Toc37549145)

[3.4. Persistent Data Management 4](#_Toc37549146)

[3.5. Access Control and Security 5](#_Toc37549147)

[3.6. Global Software Control 6](#_Toc37549148)

[3.7. Boundary Conditions 6](#_Toc37549149)

[4. Subsystem Services 6](#_Toc37549150)

[5. References 7](#_Toc37549151)

SYSTEM DESIGN DOCUMENT

# Introduction

In the YeOrada, we followed the 3-Tier Architecture. In our implementation, 3-Tier architecture is consists of Interface, Application Logic and Storage layers. The interface layer is consists of the web browser that users interact. The Application Logic layer is consists of the main web server of the YeOrada which realizes all the user operations. Finally, the Storage layer is consists of the database server of the YeOrada.

The following set of design goals will be help us to ensure the quality of the system and also give us the first ideas about system design of the project. In the section 1.2, they will be explanied in detail.

* YeOrada must be easy to use.
* YeOrada must complete searching and filtering operations about restaurants fast enough.
* YeOrada must be ready to handle invalid information given from the users.
* YeOrada’s website must be responsive.
* YeOrada must show reliable and up-to-date information to the users.

## Purpose of the System

The main purpose of YeOrada is, providing an easy, understandable and user-friendliness service for the people who are looking for restaurants, cafés, bars, pubs and so on. YeOrada offers free register for the newcomers regardless if they are restaurant owners or customers. Restaurants owners can customize their restaurant profiles in the YeOrada. Also, customers can also customize their profiles. In addition to that, as the main function of the system, customers can write reviews and post photos about restaurants.

## Design Goals

* **Ease of Use:**

A customer who has never used YeOrada before must able to find a specific restaurant which is he or she looking for and add a comment to it in maximum 10 minutes.

* **Quick Search:**

The restaurant search function in the YeOrada and filters in the restaurant search page must be display a result at most 30 seconds to the users.

* **Robustness:**

Users may enter invalid information to the forms or blank fields in the YeOrada. In this cases, YeOrada must not be crash and show proper error messages to the Users related to their invalid information.

* **Well-defined interfaces:**

The web interface of the YeOrada must be implemented as a responsive website that is the website should be compatible with different screen sizes and resolutions. Compatible means that all the interface elements of the pages of the website must be shown clearly to the users and must be useable properly under different screen sizes and resolutions.

* **Reliable Information:**

Some restaurants in the YeOrada may be removed from the system in any time by restaurant owners. YeOrada must not be show any of them to the customers in no way.

## Definitions, Acronyms, and Abbreviations

**Crash:** The web server of the YeOrada stops functioning properly and don’t give any appropriate feedback.

**Restaurant Owner:** The person which will be manage a restaurant’s, café’s, bar’s or pub’s account in the YeOrada. In the YeOrada we call all this different types of places as restaurant or client.

**Customer:** The end-user account in the YeOrada which belongs to a person.

## References

We took the [www.zomato.com](http://www.zomato.com) as example in our UI designs and system operations.

# Current Software Architecture

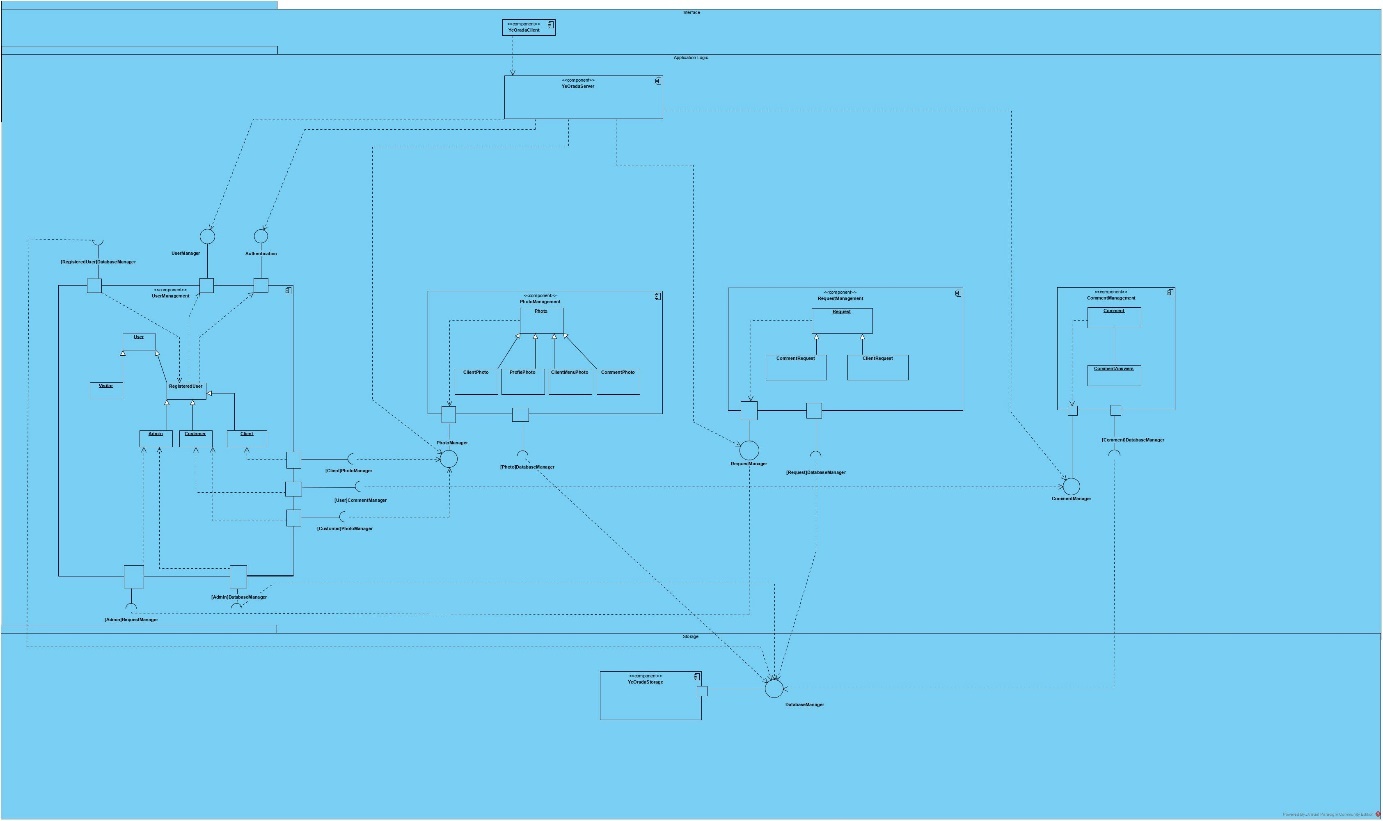
There was no previous system for YeOrada, so we use 3-tier architecture in our system. A 3-tier architecture is a type of software architecture which is composed of three “tiers” or “layers” of logical computing. 3-Tier is a layered architectural style and basically consists of the layers: Interface, Application Logic and Storage. Layers can be named different but their functions are pretty much same. Interface layer is consisting of the web browser which is commonly used today to browse in the web and display websites. Application Logic or Middleware is consisting of the system’s itself which is a web server. The web server may be change website to website, but its functions is the same: operating the functions of the users. Finally, the storage layer is consisting of the storage components of the system. It may be flat files or a relational database. They are often used in applications as a specific type of client-server system. 3-tier architectures provide many benefits for production and development environments by modularizing the user interface, business logic, and data storage layers. Doing so gives greater flexibility to development teams by allowing them to update a specific part of an application independently of the other parts. This added flexibility can improve overall time-to-market and decrease development cycle times by giving development teams the ability to replace or upgrade independent tiers without affecting the other parts of the system.

# Proposed Software Architecture

## Overview

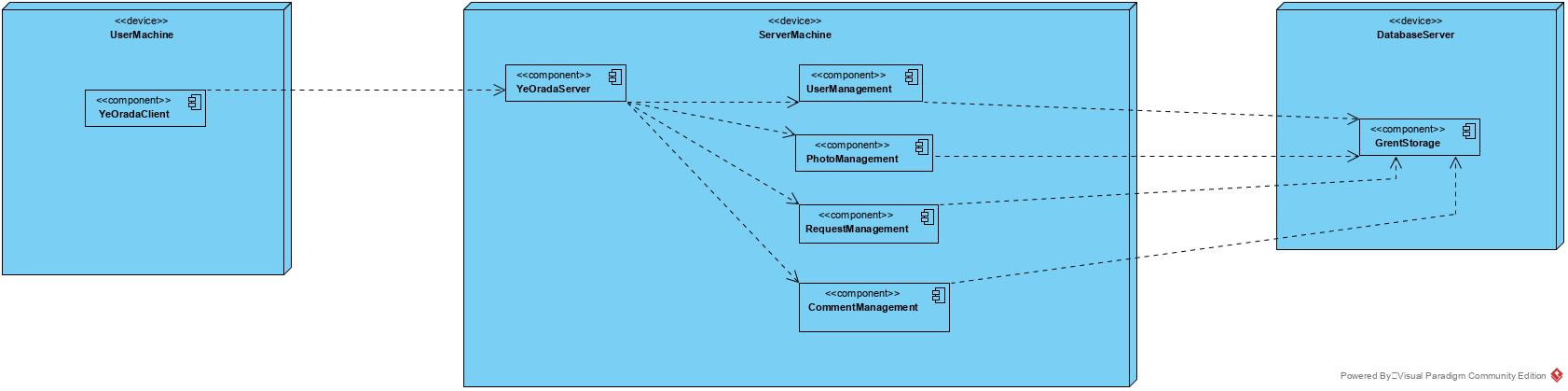
In the system decomposition of the GRENT, we basically followed the classic 3-Tier architectural style. 3-Tier archiectural style is composed of three different layers which each of them on a specific machine. In the GRENT, we applied the same style. There are three layers which are Interface, Application Logic and Storage. Interface layer consists of the YeOradaClient subsystem which is the Web Browser that implements the interface of the YeOrada. Second layer, the Application Logic is consists of the Web Server that we use in our implementation. It is the composition of UserManagement, PhotoManagement, RequestManagement and CommentManagement subsystems. UserManagement subsystem is responsible from the management of the Clients, Admins and Customers. PhotoManagement component is responsible from the management of Client Photos, Profile Photos, Client Menu Photos and Comment Photos. RequestManagement subsystem is responsible from the management of Comment Requests and Client Requests. Finally, ComponentManagement subsystem is responsible from the management of Comments and Comment Answers. Final layer in the system decomposition is Storage Layer which is the layer that consists of YeOradaStorage subsystem which is responsible from the database and storage management of the GRENT.

## System Decomposition



*Fig 1.1: The Subsystem Decomposition of the YeOrada (UML Component Diagram).*

## Hardware Software Mapping

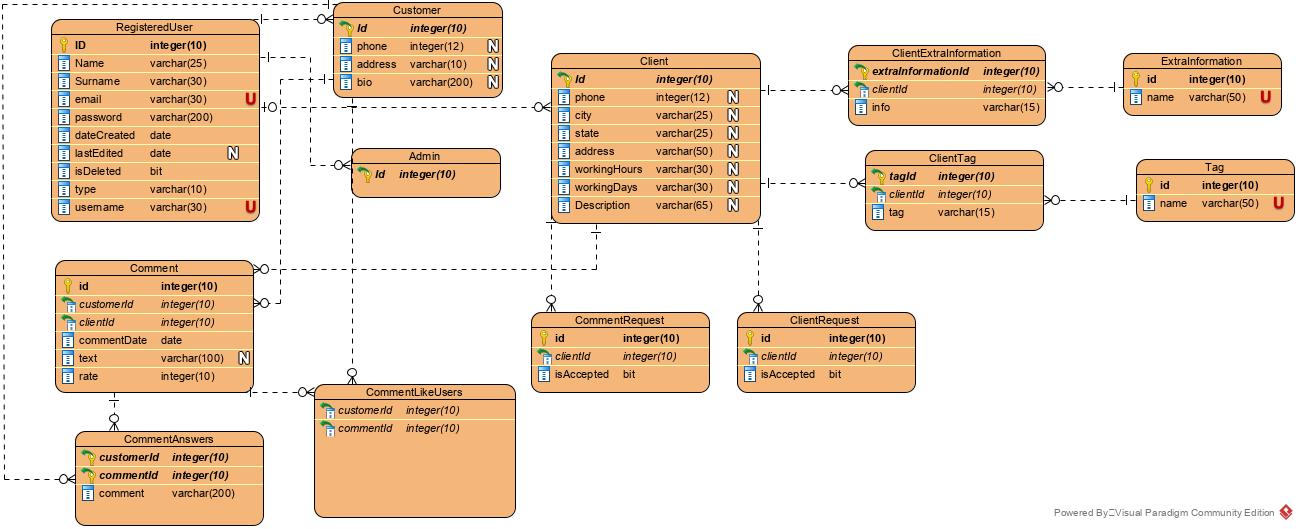


*Fig 1.2: Hardware-Software mapping of the YeOrada (UML Deployment Diagram).*

By the implementation of 3-Tier architectural style, there are three layers in three different devices(machines) in the YeOrada. First device is the UserMachine which consists of the Web Browser that users interact with. Second device is the ServerMachine which consists of the web server of the YeOrada and the third device is the DatabaseServer. It consists of the database server of the YeOrada.

In the implementation of the project, we used the Django web framework. Corresponding framework is implemented with a Tomcat web server in the project. Also, for the storage, we used a MySQL database server.

## Persistent Data Management

****

*Fig 1.3: Entity Relationship Diagram of the YeOrada database server.*

In the implementation of data storage, we used a relational database rather than flat files due to high amount of data and the high number of views which uses the same data source. Also, we chose the MySQL database server because of the familiarity of team members. The detected objects in the class diagram that must be persisted in the system are RegisteredUser, Customer, Admin, Client, Comment, CommentRequest and ClientRequest which can be seen in the Fig 1.3, as a ER diagram.

Some tables in ER diagram are created to support and provide functionality to the objects in the class diagram. CommentAnswers table is created to store answers of the comments in the system. CommentLikeAnswers table is created to store which Customer liked which Comment in the system. ClientTag table stores the predefined tags of the Clients. These predefined tags are stored in the Tag table. ClientExtraInformation table stores predefined informations about Clients (Is there a wifi? Is there a alcoholic drinks? And so on.). These predefined informations are stored in the ExtraInformation table.

In the system, we realized the encapsulation of the database by using Django’s own database management interface. This type of management provides security and prevents code recurrences.

## Access Control and Security

The user authorization in the YeOrada is specified below as an access control matrix.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| OBJECTS  ACTORS | Comment | Photo | Request | Customer | Client | Visitor |
| Client |  | updateMyClientPhotos()  updateMyMenuPhotos() |  |  | updateMyClientAccount()  updateMyGeneralInfo()  deleteMyAccount() |  |
| Customer | addComment()  likeComment()  answerComment() | changeProfilePhoto() | makeCommentRequest() | updateMyAccountSettings()  changeMyPassword() |  |  |
| Admin |  |  | verifyComment()  verifyClient() |  | addClient()  updateClient() |  |
| Visitor |  |  |  |  |  | register()  login() |

## Global Software Control

In the YeOrada we used the Django web framework with Python programming language to realize the system. Python is a event-driven programming language but also supports Threads. Since it is a website, our system based on threads. In the system, Threads are accumulated by the Django framework and Apache Tomcat server so it is a automatic process that we don’t implement it in any way.

## Boundary Conditions

In the RAD document, we pretty much completed the exception handling related to software faults and specified their use cases but we did not mention start-up and shutdown behaviours of the system.

**Start-up:** System Admin logs in to the YeOrada Tomcat Server environment, checks the server logs to see if there is a crash or planned shutdown in the server. If there is unusual crash, the System Admin starts the server with the last saved data by the server to avoid data loss. Else, System Admin starts the server as usual from beginning.

**Shutdown:** System Admin logs in to the YeOrada Tomcat Server environment and activates the shutdown function of the system. Since it is a planned shutdown, server shutdown itself directly, without persist any data to the database.

# Subsystem Services

There are 4 main subsytems in the Application Logic layer of the system. These are UserManagement, PhotoManagement, RequestManagement and CommentManagement subsystems. In this section, we will examine services provided by these subsystems. Note that these services may be change or updated in the ODD document later.

UserManagement:

This subsystem has the UserManager and Authentication services. UserManager provides the functions related to user operations. It provides add/remove/update and change password function of the RegisteredUser object. Authentication service provides register and login functions of the RegisteredUser object.

PhotoManagement:

This subsystem has the PhotoManager service. PhotoManager provides the add/update/remove functions of the ClientPhoto, ProfilePhoto, ClientMenuPhoto and CommentPhoto objects.

RequestManagement:

This subsytem has the RequestManager service. RequestManager provides the add and verify functions of the ClientRequest and CommentRequest objects.

CommentManagement:

This subsystem has the CommentManager service. CommentManager prodives add and like functions of the Comment object and add function of the CommentAnswer object.

# References

1. Bruegge B. & Dutoit A.H.. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 3rd ed.
2. https://online.visual-paradigm.com/diagrams/tutorials