**Purpose**

The system design is documented in the System Design Document (SDD). It describes additional design goals set by the software architect, the subsystem decomposition (with UML class diagrams), hardware/software mapping (with UML deployment diagrams), data management, access control, control flow mechanisms, and boundary conditions. The SDD serves as the binding reference document when architecture-level decisions need to be revisited.

**Audience**

The audience for the SDD includes the system architect and the object designers as well as the project manager.

**Table of Contents**

1. Introduction 2

1.1 Overview 2

1.2 Definitions, acronyms, and abbreviations 2

1.3 References 2

2. Design Goals 2

3. Subsystem decomposition 2

4. Hardware/software mapping 2

5. Persistent data management 2

6. Access control and security 2

7. Global software control 3

8. Boundary conditions 3

**Document History**

|  |  |  |  |
| --- | --- | --- | --- |
| Rev. | Author | Date | Changes |
| 1 | Vadym Khyzhniak | 01.08.2022 | Added the “Design Goals” part. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

*The purpose of the Introduction is to provide a brief overview of the software architecture. It also provides references to other documents.*

## Overview

The program represents a client-server architecture. Spring Boot Framework

was used to implement it. The interactions work the following way:

The buttons on client graphic interface (provided by JavaFX) clicked by client trigger an event that sends a corresponding request to the server. After the request was processed it is being sent back to the client side and corresponding events happen on the screen.

## Definitions, acronyms, and abbreviations

## References

# Design Goals

*This section describes the design goals and their prioritization (e.g. usability over extensibility). These are additional nonfunctional requirements that are of interest to the developers. Any trade-offs between design goals (e.g., usability vs. functionality, build vs. buy, memory space vs. response time), and the rationale behind the specific solution should be described in this section. Also the rationale of all other decisions must be consistent with described design goals.*

## *2.1* ***Functionality vs minimum number of errors***

The functionality for the client was the prioritized design goal. Thus, the trade-off appeared: functionality vs minimum number of errors, which is developers’ design goal. With more functionality the amount of edge cases and hence potential number of errors increases. The functional variety was brought to the golden mean, so that all possible flows of events could be taken into account.

## *2.2* ***Increased productivity (smoothness) vs functionality***

The increased productivity was the prioritized design goal, so that the client could use the application smoothly. The amount of restaurants displayed on the map in each district has to be decent, so that the client has a wide range of the restaurants to choose from. The decision was to limit the number of displayed restaurants’ markers on the map to only 280, so that the program does not freeze during the use and still has lots of alternatives to offer.

# Subsystem decomposition

*This section describes the decomposition of the system into subsystems and the services provided by each subsystem. The services are the seed for the APIs detailed in the Object Design Document.*

# Hardware/software mapping

*This section describes how the subsystems are mapped onto existing hardware and software components. A UML deployment diagram accompanies the description. The existing components are often off-the-shelf components. If the components are distributed on different nodes, the network infrastructure and the protocols are also described.*

# Persistent data management

*This section describes how the entity objects are mapped to persistent storage.*

*It contains a rationale of the selected storage scheme, file system or database, a description of the selected database and database administration issues.*

System files with unique ids were used for efficient parsing, own efficient saving and parsing classes were created to both save and get the data from files.

# Access control and security

*This section describes the access control and security issues based on the nonfunctional requirements in the requirements analysis document. It also describes the implementation of the access matrix based on capabilities or access control lists, the selection of authentication mechanisms and the use of encryption algorithms.*

When a new user is created he is saved persistently on the server. When a user wants to login a request is made to the server that compares the username with the hashed password.

# Global software control

*This section describes the control flow of the system, in particular, whether a monolithic, event-driven control flow or concurrent processes have been selected, how requests are initiated and specific synchronization issues.*

The system is divided into client and server. These two major components communicate via an event-driven control flow. When the user presses a button on the client interface it triggers an event that sends a request to the server. This request is then processed and sent back to the client. This happens asynchronously, so that the program is still responsive while the user waits for the response.

# Boundary conditions

*This section describes the use cases how to start up the separate components of the system, how to shut them down, and what to do if a component or the system fails.*

To make a reservation, select a restaurant from the list under search bar, click on “go to Reservation”, enter a date, choose a time slot, select the number of people, select the table on the table schema, press “Make Reservation”. To cancel the reservation (in case it is not confirmed), press “Cancel Reservation”. If some data entered was invalid, corresponding messages would appear on the screen. If you want to stop the process of reservation, press “Home”. In case when the program does not respond to your actions, just use force quit out of the app, no other alternatives are provided for such scenario.