

Università di Pisa  
  
  
  
DISTRIBUTED SYSTEMS AND MIDDLEWARE TECHNOLOGIES

**PISA-EAT  
Prroject specifications**  
  
  
  
  
Accademic year 2020-2021  
  
  
  
  
Andrea Tubak, Francesco Ronchieri, Alessandro Madonna

Summary

[Introduction 3](#_Toc63414153)

[Overall idea 3](#_Toc63414154)

[Assumptions 3](#_Toc63414155)

[Critial areas 4](#_Toc63414156)

[Synchronization/Coordination 4](#_Toc63414157)

[Proposed solution 4](#_Toc63414158)

[Communication 4](#_Toc63414159)

[Proposed solution 4](#_Toc63414160)

[Erlang part 4](#_Toc63414161)

# Introduction

## Overall idea

Pisa-Eat is an online platform that offers a new way of interacting between customer and restaurateur.

Restaurateurs can register their restaurant on the platform and make it available to all users.

A user can browse the site in search of his favorite restaurant and consult the available tables, once he has found the one that best meets his needs, in terms of number of seats and position, he can book it and share it with his friends.

At the table, users can consult the menu and start filling in the order autonomously through the platform, adding, modifying, or removing the dishes; at the same time, users can communicate directly with the kitchen, specifying, for example, allergies or special requests that cannot be expressed directly from the platform.

Once all users are ready, each of them will have to confirm the order so that they can be processed by the kitchen.

The restaurateur will have access to a reserved area, where he can see and interact with every aspect of the restaurant (e.g., consult the status of the tables, change their arrangement, check orders, etc.)

Always under the control of the restaurateur, access to the reserved area can also be given to staff with different privileges related to the job (e.g., waiter, cook, etc.) to speed up interaction between staff and improve service.

## Assumptions

To avoid spending time implementing superfluous functionalities and instead focusing on the topics discussed in class, the workload will be reduced by limiting us to creating a reduced version of the overall idea, in particular these assumptions will be considered:

* All the reserved area will not be considered
* Only one restaurant will be present, and the arrangement of the tables cannot be changed
* The order page for the user will be limited to only allow communications with the kitchen

# Critial areas

## Synchronization/Coordination

A critical area for synchronization/coordination is table reservations, since a user must have the mutual exclusion on the table that he wants to book (this also applies to joining a table)

### Proposed solution

the critical sections for competition will be managed through a Singleton EJB which will have the task of:

1) take charge of the request

2) create an ad-hoc task to manage the request (containing code for mutual exclusion)

3) let Glassfish default ManagedExecutorService execute the task by returning a Future <> of the return type to the caller

## Communication

An area where communication is involved is in the order section, where users can communicate with the kitchen

### Proposed solution

An ERLANG REST web service will be implemented to post messages with the kitchen

# Erlang part

In order to implement a part of the project in Erlang, the web service for communication with the kitchen will be implemented by using Erlang