Distributed Systems and Middleware Technologies

PisaEat

Alessandro Madonna, Andrea Tubak, Francesco Ronchieri

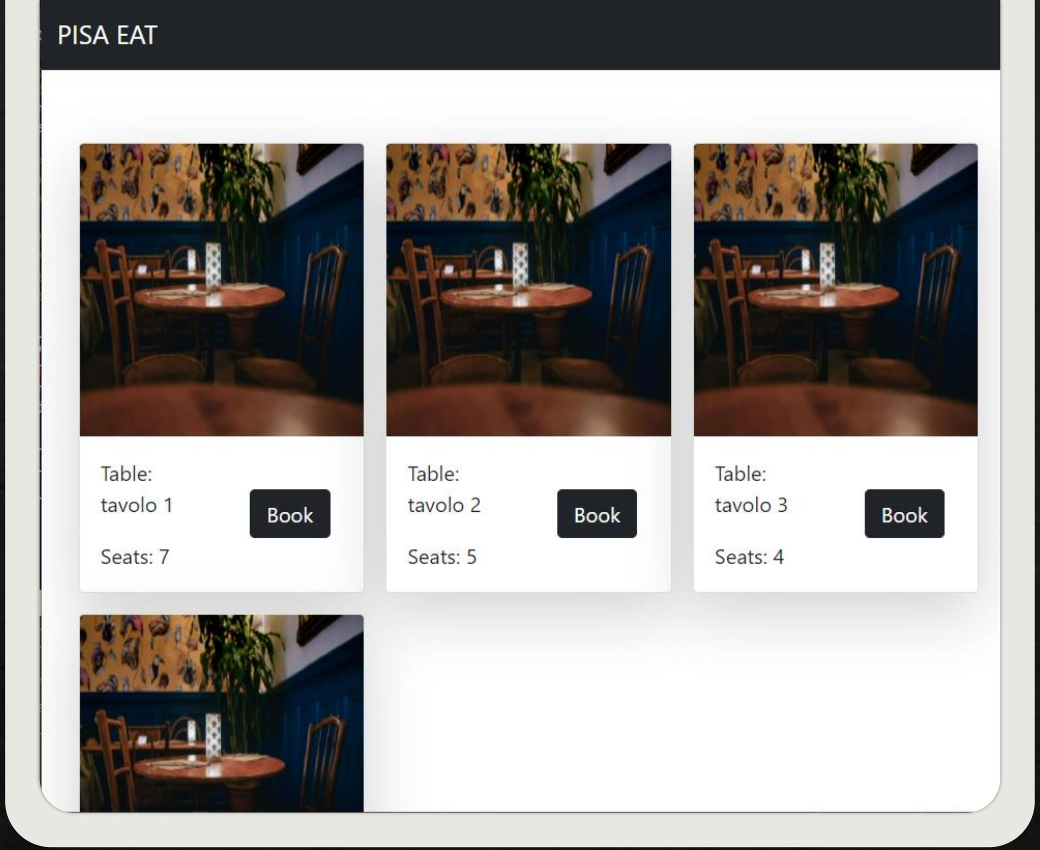
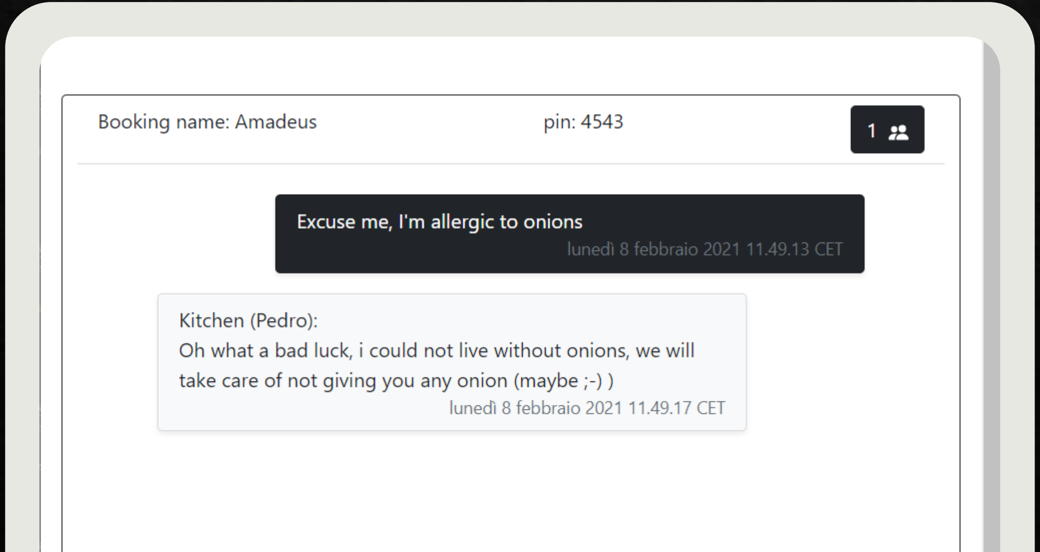

PISA EAT

Table:
tavolo 1
Book
Seats: 7

Table:
tavolo 2
Book
Seats: 5

Table:
tavolo 3
Book
Seats: 4


Booking name: Amadeus
pin: 4543
1

Excuse me, I'm allergic to onions
lunedì 8 febbraio 2021 11.49.13 CET

Kitchen (Pedro):
Oh what a bad luck, i could not live without onions, we will take care of not giving you any onion (maybe ;-))
lunedì 8 febbraio 2021 11.49.17 CET

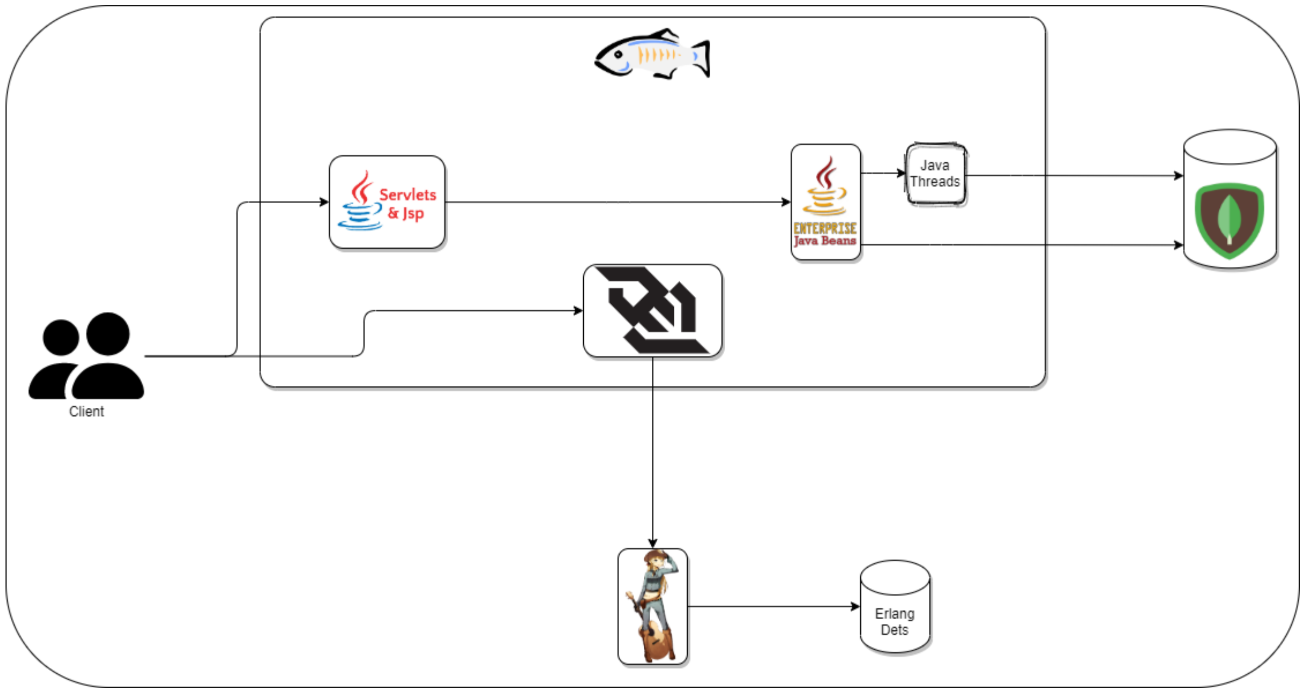
Introduction

PisaEat is a web platform where users can book a table on their favorite restaurant.

Once a table is booked, other users can join in (provided they know the right pin code) and exchange messages with the kitchen in order to communicate any kind of information (es. allergies or intolerances)

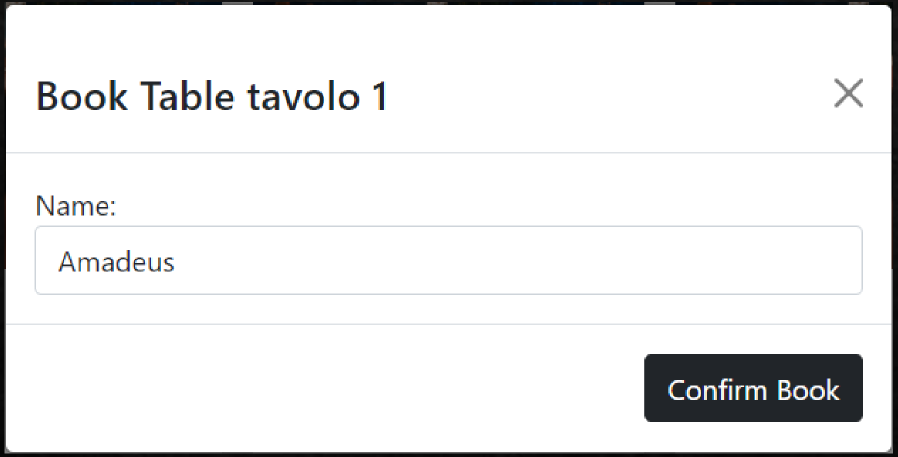
General Schema

- Glassfish (port 8080):
  - Servlet: http://<host>:8080/PisaEat/
  - WebSocket: ws://<host>:8080/chat-WebSocket/chat/
- Cowboy: http://<erl-host>:8081/api/
- MongoDB: http://<mongo-host>:27017



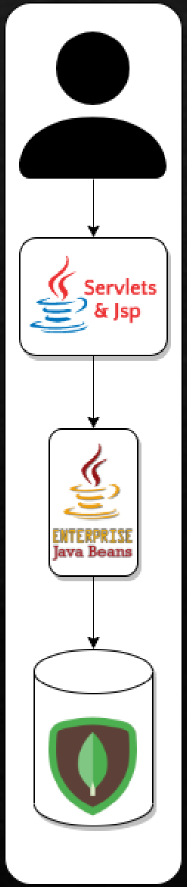
Synchronization Problems

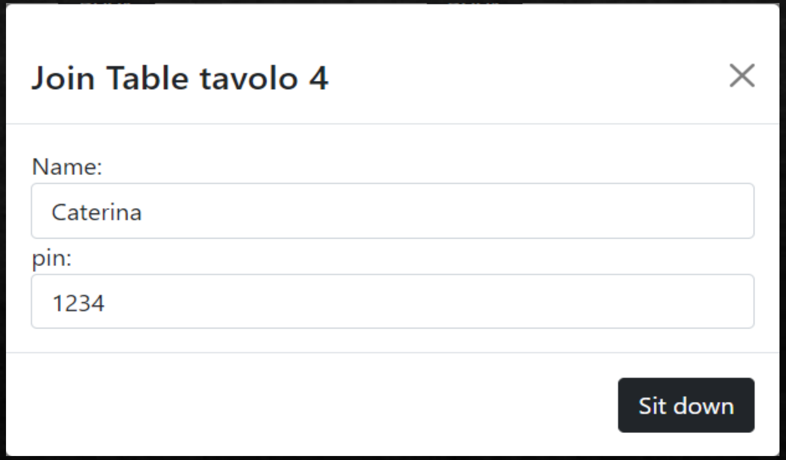
1. Two or more people can try to book the same table, at the same time. So, a race begins.
2. The seats of a table are limited, if there's only one seat left, a race can begin.


Book Table tavolo 1

Name:

Amadeus




Join Table tavolo 4

Name:

Caterina

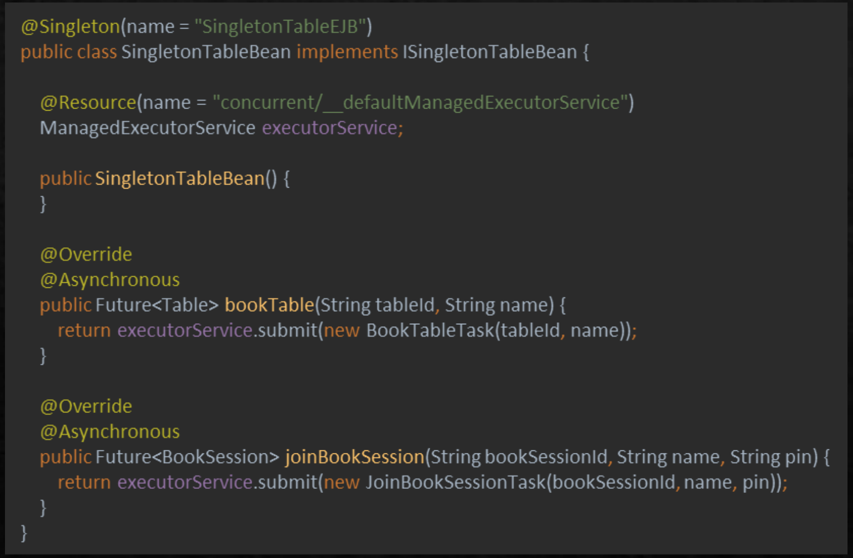
pin:

1234

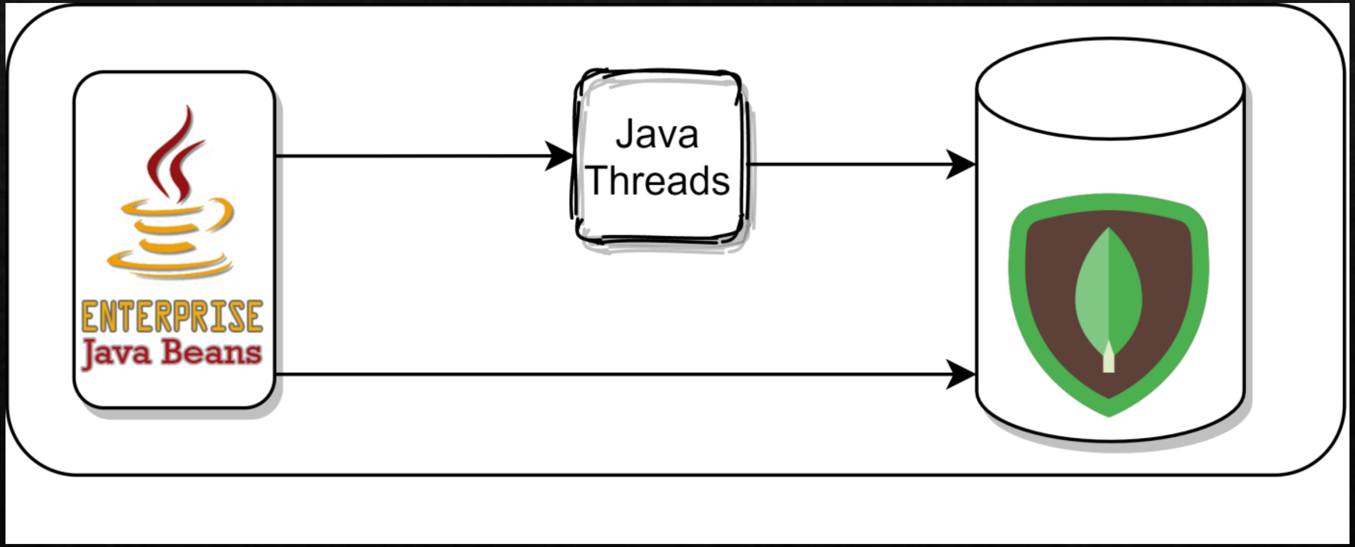
Solution to Syncronization Problems

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

- 1) take charge of the requests
- 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



```java
@Singleton(name = "SingletonTableEJB")
public class SingletonTableBean implements ISingletonTableBean {

  @Resource(name = "concurrent/__defaultManagedExecutorService")
  ManagedExecutorService executorService;

  public SingletonTableBean() {
  }

  @Override
  @Asynchronous
  public Future<Table> bookTable(String tableId, String name) {
    return executorService.submit(new BookTableTask(tableId, name));
  }

  @Override
  @Asynchronous
  public Future<BookSession> joinBookSession(String bookSessionId, String name, String pin) {
    return executorService.submit(new JoinBookSessionTask(bookSessionId, name, pin));
  }
}
```



Thread Syncronization

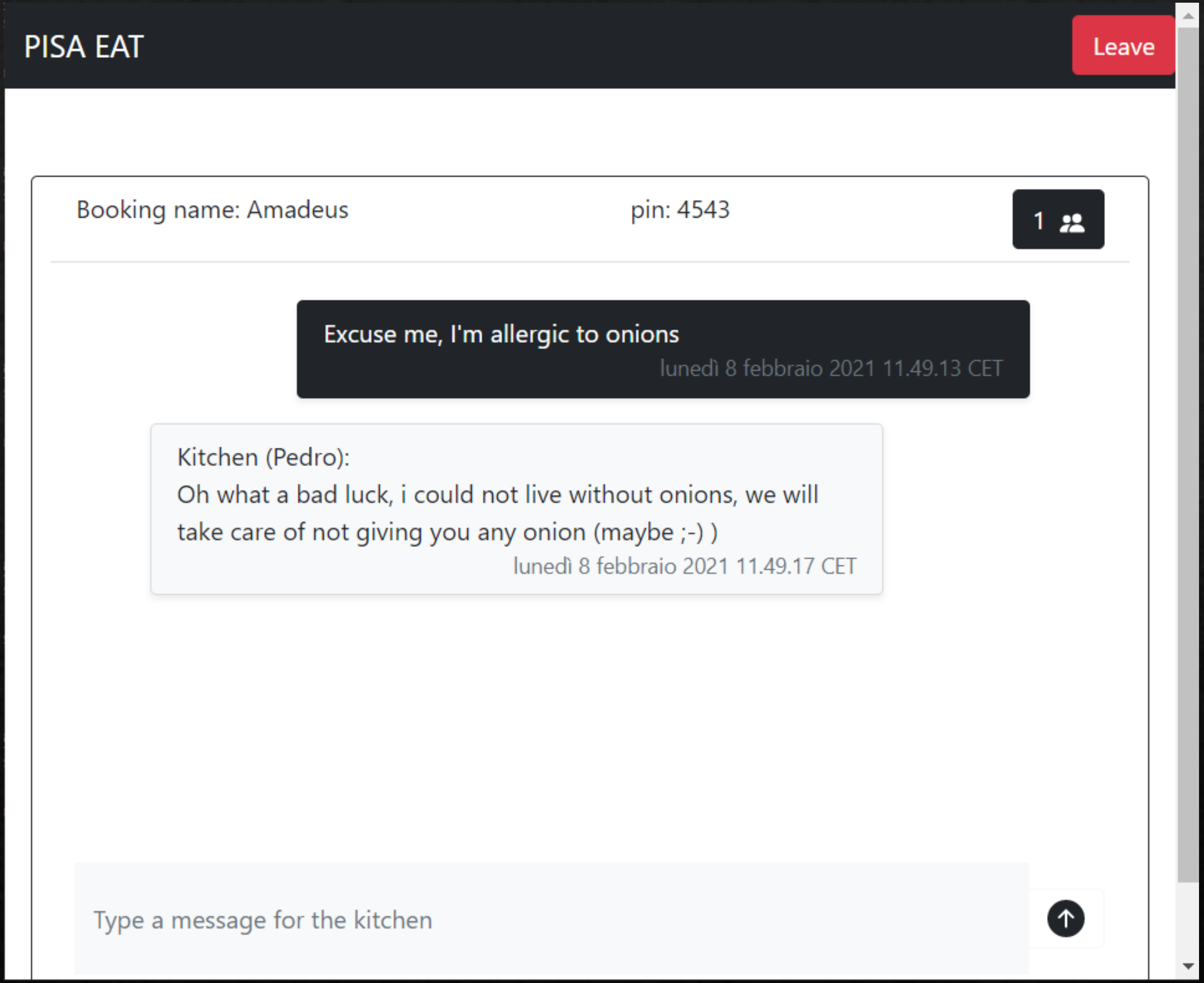
As said before, the synchronization among threads is handled by tasks submitted to GlassFish Executor Service.

To obtain a performance compromise, accesses in mutual exclusion are provided based on a hashing function applied to the identifier of the resource that we want to modify

```java
public class BookTableTask implements Callable<Table> {
   private static final Striped<Lock> tableLocks = Striped.lock(1024);
   ...

   @Override
   public Table call() throws Exception {
      //not synchronized code
      tableLocks.get(tableId).lock();

      try {
         //synchronized code

         tableLocks.get(tableId).unlock();
         return returnTable;
      }catch (Exception e){
         tableLocks.get(tableId).unlock();
         throw e;
      }

   }
}
```

Comunication Problem

After booking a table, there is a chat with the kitchen in order to communicate any kind of information (es. allergies or intolerances)


PISA EAT

Booking name: Amadeus   pin: 4543   1

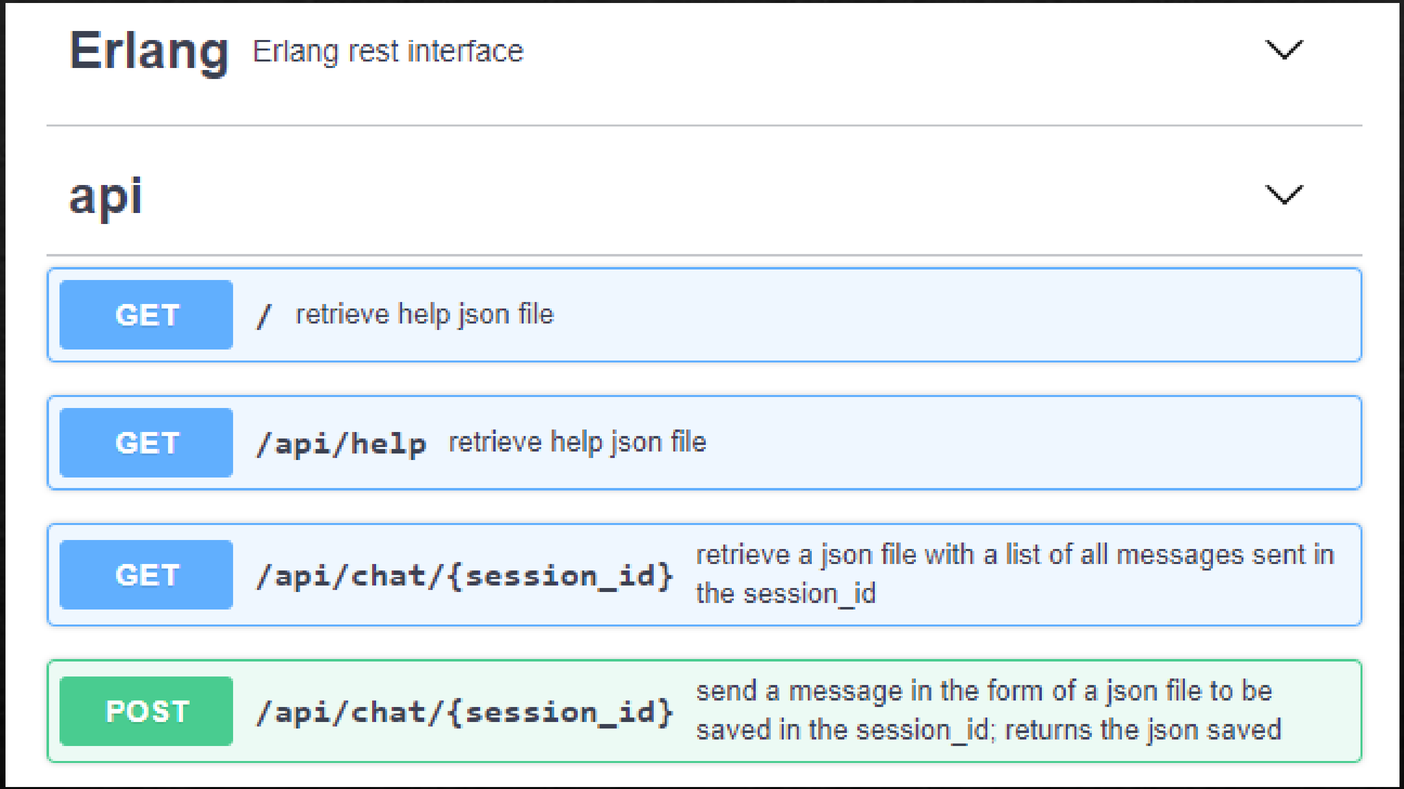
Excuse me, I'm allergic to onions
lunedì 8 febbraio 2021 11.49.13 CET

Kitchen (Pedro):
Oh what a bad luck, i could not live without onions, we will take care of not giving you any onion (maybe ;-))
lunedì 8 febbraio 2021 11.49.17 CET

Communication Solutions

In order to implements the chat we made use of 2 components:

1) An Erlang Rest Web Service (ERWS) with some base operations for reading and saving messages in a dedicated dets table
2) A WebSocket: it was necessary in order to coordinate the messages sent by other users of the same table:
   - ◇ Storing incoming messages using the ERWS
   - ◇ Forwarding the messages to all other clients currently in the session



Erlang Erlang rest interface

api

GET `/` retrieve help json file

GET `/api/help` retrieve help json file

GET `/api/chat/{session_id}` retrieve a json file with a list of all messages sent in the session_id

POST `/api/chat/{session_id}` send a message in the form of a json file to be saved in the session_id; returns the json saved

End