

# Beach Booking

Service Oriented Software Engineering

Università degli Studi dell'Aquila

Academic Year: 2018/19

Fabio Di Silvestro – 260651

## Summary

SUMMARY	2
INTRODUCTION	3
SERVICES' DESCRIPTION	3
Authentication	3
Parking	3
Beach	3
AccuWeather	3
Prosumer	4
LOAD BALANCER	4
CLIENT	4
MAVEN ARCHETYPE	5
ARCHITECTURAL VIEW	6
Component Diagram	6
Sequence Diagrams	7
INSTALLATION INSTRUCTION	9
CONTENTS OF THE PROJECT FOLDER	9

## Introduction

The goal of the system is to offer the possibility to search and eventually book a beach umbrella in the selected location (city). The application provides also weather forecasts about the location and parking information. In order to access the functionalities of the system, the user must carry out the registration process and login.

## Services' Description

### Authentication

The service Authentication is in charge to manage the user-part of the system. It is a REST service and exposes the following functionalities:

- register(User user) -> String
  - o POST. Insert the user in the database.
- login(User user) -> String
  - o POST. Return a unique key that allows the user to access system's services.
- logout(String key) -> void
  - o GET. Delete the key.

### Parking

This simple SOAP Web Service provides information about the near parking. It is called by the prosumer for each available beach.

### Beach

The Beach REST service is the core component of the system. It manages the search and the booking of the beaches. The functionalities are:

- getBeaches(String city) -> List<Beach>
  - o GET. Return the list of the available beach within the city selected.
- bookBeach(Long beachId, Date date, String username) -> Booking
  - o GET. Book the selected beach for the selected date.
- deleteBooking(Long bookingId) -> void
  - o GET. Delete the selected booking. It is performed a logic delete.
- getListOfBooking(String username) -> List<Booking>
  - o GET. Return the list of the booking of the user.

### AccuWeather

The external service is invoked first of all to get the location key and later for taking the 5 days forecast.

## Prosumer

The prosumer is the link between the clients and the services. It acts like an orchestrator and maintains the logic about how and when calls the service providers. In some operation the calls to the providers are made in parallel (e.g. in getBeaches, the beach service and the parking service). In almost each service before invoking the required functionalities is performed the check on the user key.

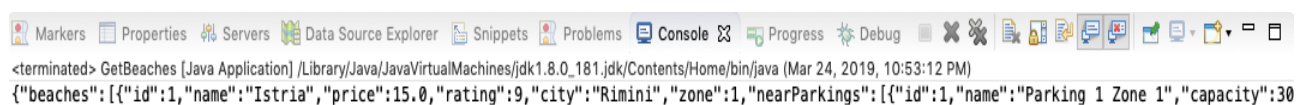
## Load Balancer

In order to increase the overall non-functional requirements of the system, in particular Performance and Availability, all the services are deployed on more than one application container. This system distribution is completely transparent to the client and to the prosumer because a proxy load-balancer is used. HAProxy is one of the best software load-balancer, used by big company. In this case has been configured for selecting the application container with the lowest active connections (thus load).

## Client

A java application allows to perform calls to each service provided by the prosumer. The output is printed in the console.

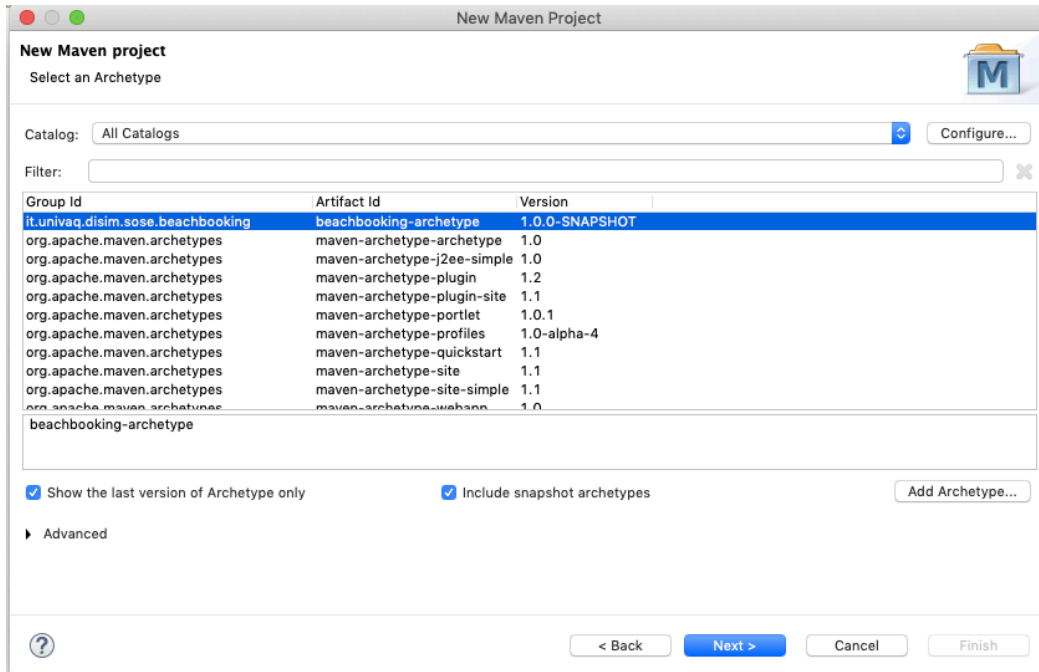
e.g.



```
<terminated> GetBeaches [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_181.jdk/Contents/Home/bin/java (Mar 24, 2019, 10:53:12 PM)
{"beaches":[{"id":1,"name":"Istria","price":15.0,"rating":9,"city":"Rimini","zone":1,"nearParkings":[{"id":1,"name":"Parking 1 Zone 1","capacity":30
```

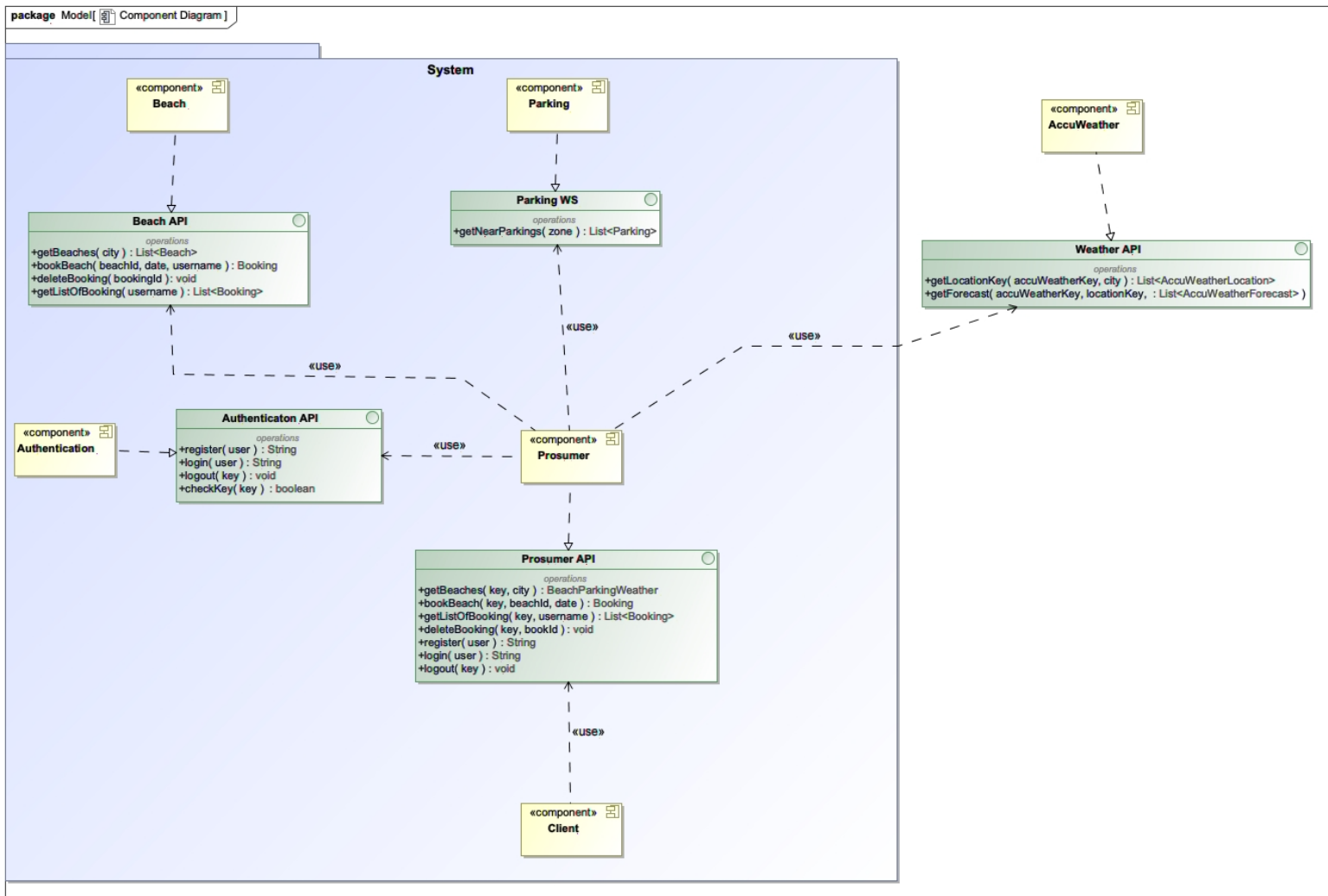
## Maven Archetype

The Maven Archetype for the prosumer has been created. The installation of the archetype is required “mvn install” in order to use it. The project from the archetype can be build either from the command line or from the Eclipse wizard.



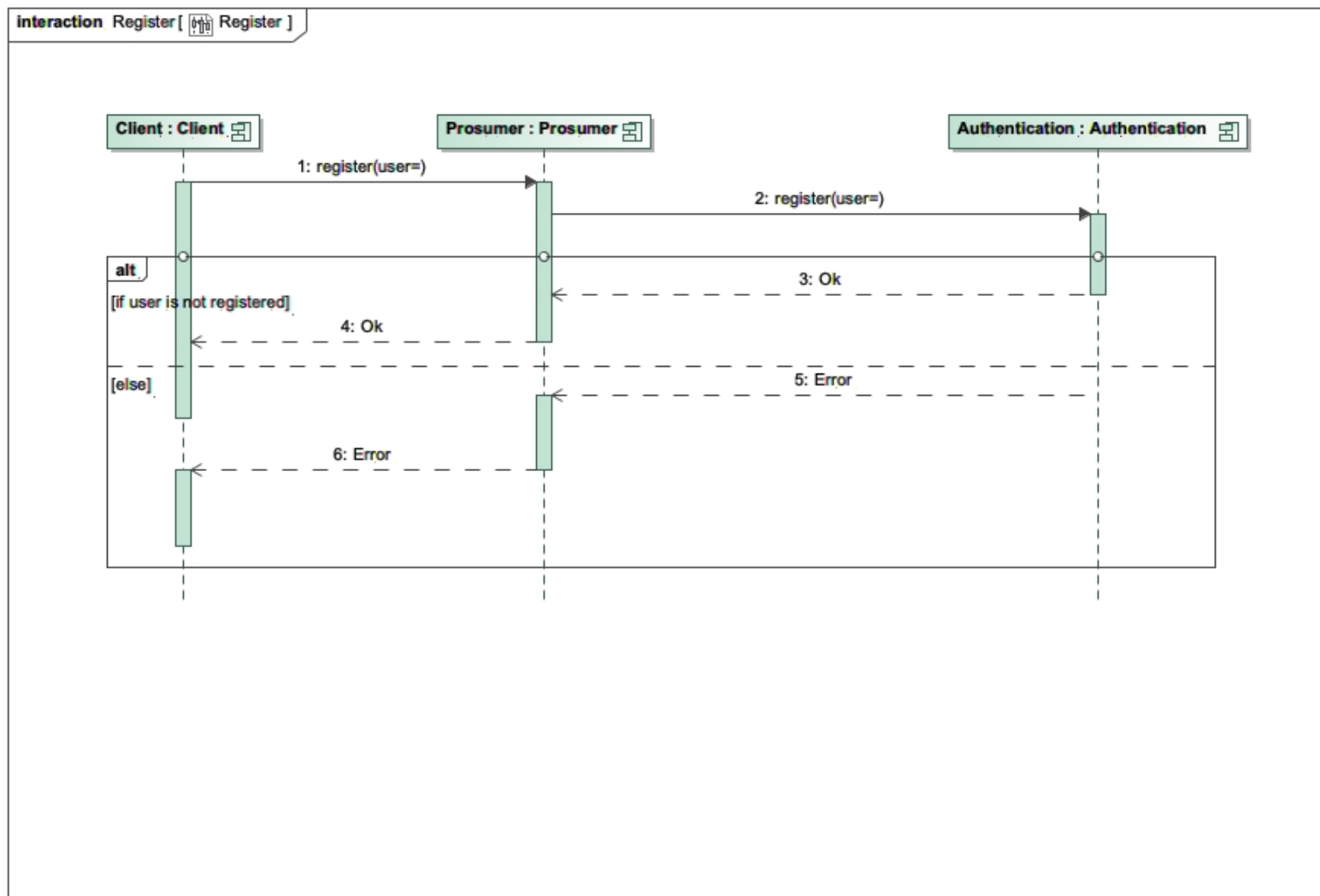
## Architectural view

### Component Diagram

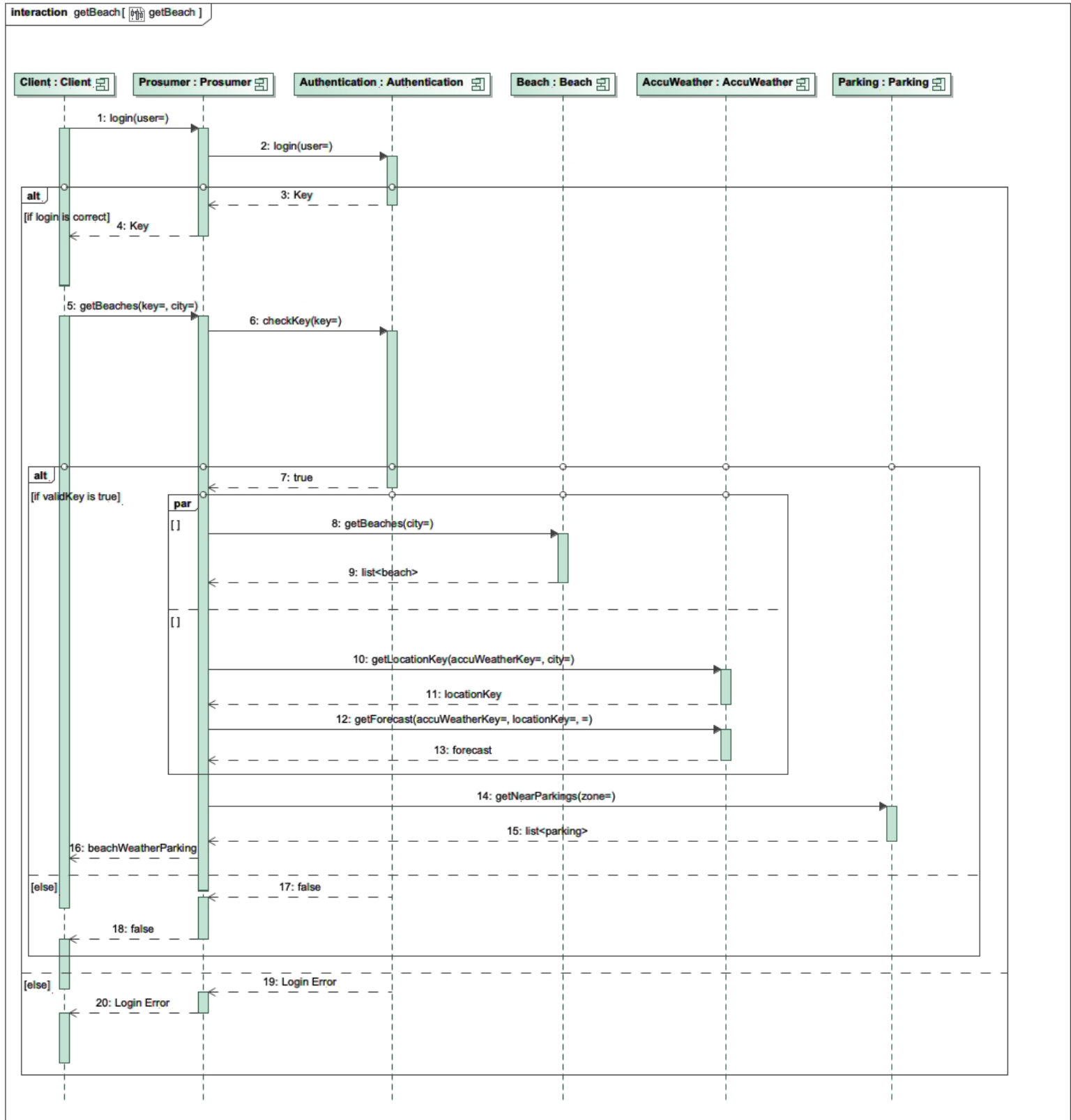


## Sequence Diagrams

### Register



## GetBeaches





## Installation Instruction

In order to execute the application, the following steps are required:

- Download Apache Tomcat (the application has been tested with the 8.5.38 version)
- Configure at least two separate instances of Tomcat (I used two instances on the same machine with ports 8080 and 6080)
- Deploy the services' wars on the two instances
- Install HAProxy and edit the "haproxy.cfg" file with the right IP addresses and ports (in the project folder there is the config file I have developed). A service reload is needed to let the service update
- Now the application is running correctly. It is possible to view information about the servers from the HAProxy dashboard.
- Since the prosumer offers REST services, it is possible to test the system through a web browser.
- Alternatively, it is possible to use the java client I have developed to interact with the system.

## Contents of the project folder

1. Documentation PDF file
2. #4 WAR files (services)
3. #4 Maven Projects (services)
4. #1 Maven Project (client)
5. Maven Archetype for the prosumer
6. HAProxy configuration file
7. UML Diagrams