|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hospitality platform | | | | |
|  |  | | |  |
| Technical IT Specifications | | | | |
|  | | Architecture and components |  | |

TABLE OF CONTENTS

[Introduction 3](#_Toc18231627)

[1. Programming languages 4](#_Toc18231628)

[2. Database schemas 4](#_Toc18231629)

[3. Containers 4](#_Toc18231630)

[4. Hosting platforms and servers 4](#_Toc18231631)

[5. Cloud services 4](#_Toc18231632)

[6. Extras 4](#_Toc18231633)

# Introduction

Building such solutions mainly based in web-scraping and API consumption (Rest and SOAP web services), requires some principles to be pre-defined and some rules to be followed in order to create robust, scalable and “durable-utilized-beautiful” solutions. Based on that the design pattern of the project will follow the clean architecture principles, mainly consisting on:

* The backend architecture of the application will be separated into three layers, where the inner-layer should NOT know anything about outer-layer, and the outer-layer can implement only the it’s neighbor inner-layer.
* Each of the layers will be responsible for a piece of logic of the application, as follows:
  + Enterprise Business Logic – which includes the entities Http data access and implement a pattern such IRepository or Unit of Work.
  + Application business logic – Which includes the validations, data-models interfaces and implementations.
  + Gateways and controllers – Where the moving-parts of the machine will rest. Includes interface adapters and implementations.

As for the organizational and managing part I would suggest using JIRA and for the version control GitHub, GitLab and Bitbucket each of them would do the job.

Also part of developing clean software highly would recommend TDD(Test driven development), for higher quality and less buggier application.

# Programming languages

Python - Putting this as the first of the list for three main reasons. First of all great libraries and analyzing tools for web-scraping, fast development, large community support. Using frameworks like BeautifulSoup and/or Scrapy web-scraping wouldn’t take too much effort.

**C# (.NET)** -Except the scraping part of the application/system, from the description also we are tending to offer our own services such as rating, comments and more. To achieve that we need a web-service architecture constructed in order to operate. Since I have been dealing with C# for quite a while now, I am into the design and architecture patterns and best-practices.

**PHP** – Since it is massively used in the web it might be an alternative to perform certain tasks and deal with things that PHP provides easy solutions on that. Having some attention to Laravel framework.

**JavaScript/TypeScript** – As for the front-end we are mainly thinking of Angular since I have been dealing with it for a while now, and it offers a full package with all the tools provided to perform any kind of task.

# DataBAsE schemas

Based on the complexity of the system, we are proposing a hybrid database schema with both relational and non-relational databases included. As for the relational I have been dealing with a lot of relational databases, such as: Oracle, MSSQL-Server, PostgreSQL and MySQL, all in enterprise level. From my point of view for that kind of application PostgreSQL fulfills most of the requirements. Also it comes with some great benefits such as:

* Open-source
* Built mainly in Python
* CI/CD support
* Cloud service support

# CONtainers

# Hosting platforms and servers

# Cloud services

# Extras