Hospitality platform

Technical IT Specifications

Architecture and components

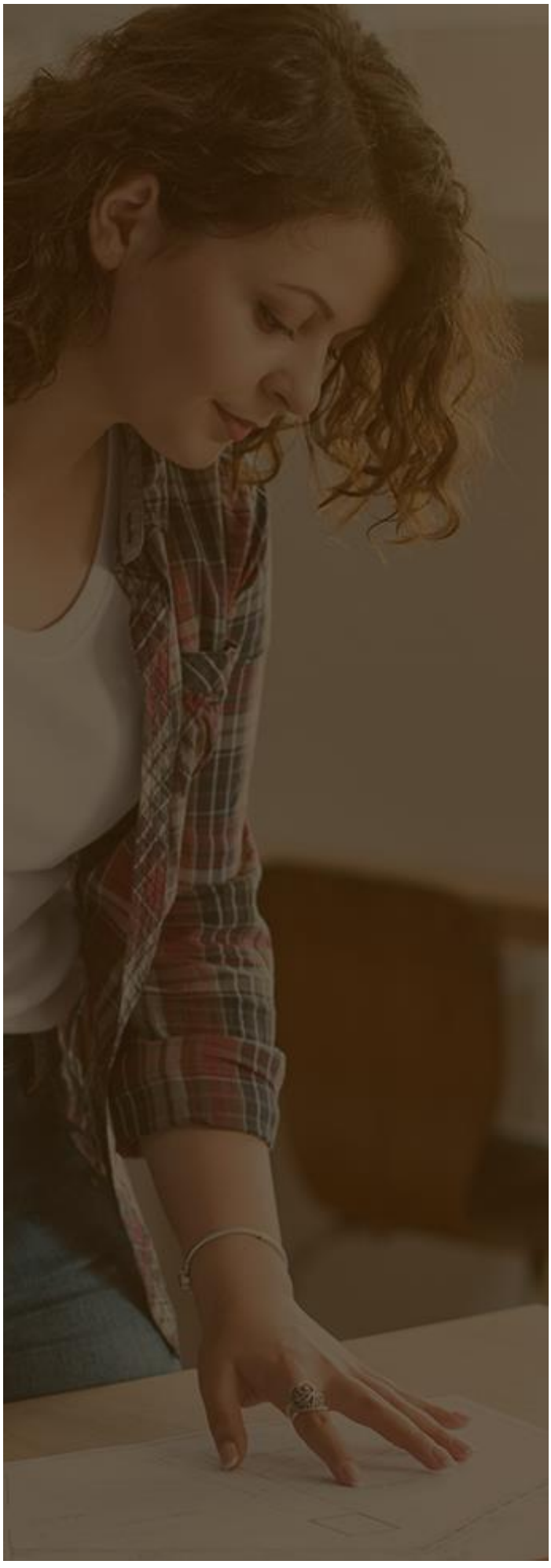

TABLE OF CONTENTS

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.

1. 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.

2. 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

Apart from that, in order to deal with scraping and crawling as a recommended solution is to use NOSQL databases or document-based NOSQL such as MongoDB. Lately Postgres has introduced Postgres JSONB, saving all the data as json, but parsed as binary. Anyways if Postgres is our choice for relational, using JSONB along with MongoDB wouldn't be a pain.

Why PostgreSQL

PostgreSQL comes with many features aimed to help developers build applications, administrators to protect data integrity and build fault-tolerant environments, and help you manage your data no matter how big or small the dataset. In addition to being free and open source, PostgreSQL is highly extensible. For example, you can define your own data types, build out custom functions, even write code from different programming languages without recompiling your database.

Advantages:

- Data types
- Data integrity
- Concurrency and performance
- Reliability
- Security

3. CONTAINERS

In order to provide a solution that does not have problems of the kind deployment issues, runtime errors and so on, implementing containerization with Docker has resulted a great approach to solving these kind of problems. Working with a clean architecture approach helps a lot in deploying each part of application separately in a docker container. In order to make each piece of application work smoothly with each-other Kubernetes might come handy later on. Docker Hub really helps the system in the way how it is packaged, deployed and managed.

Containers work a little like VMs, but in a far more specific and granular way. They isolate a single application and its dependencies—all of the external software libraries the app requires to run—both from the underlying operating system and from other containers. All of the containerized apps share a single, common operating system (either Linux or Windows), but they are compartmentalized from one another and from the system at large.

The benefits of Docker containers show up in many places. Here are some of the major advantages of Docker and containers:

Docker enables more efficient use of system resources

Instances of containerized apps use far less memory than virtual machines, they start up and stop more quickly, and they can be packed far more densely on their host hardware. All of this amounts to less spending on IT.

The cost savings will vary depending on what apps are in play and how resource-intensive they may be, but containers invariably work out as more efficient than VMs. It’s also possible to save on costs of software licenses, because you need many fewer operating system instances to run the same workloads.

Docker enables faster software delivery cycles

Enterprise software must respond quickly to changing conditions. That means both easy scaling to meet demand and easy updating to add new features as the business requires.

Docker containers make it easy to put new versions of software, with new business features, into production quickly—and to quickly roll back to a previous version if you need to. They also make it easier to implement strategies like blue/green deployments.

Docker enables application portability

Where you run an enterprise application matters—behind the firewall, for the sake of keeping things close by and secure; or out in a public cloud, for easy public access and high elasticity of resources. Because Docker containers encapsulate everything an application needs to run (and only those things), they allow applications to be shuttled easily between environments. Any host with the Docker runtime installed—be it a developer's laptop or a public cloud instance—can run a Docker container.

Docker shines for microservices architecture

Lightweight, portable, and self-contained, Docker containers make it easier to build software along forward-thinking lines, so that you're not trying to solve tomorrow's problems with yesterday's development methods.

4. HOSTING PLATFORMS AND SERVERS – CLOUD SERVICES

For hosting platforms always a good solution about that are cloud hosting providers. Highly recommending Amazon Web Services, over other technologies such as Azure or Google Cloud. Major reasons why AWS include:

1- Better integration. It provides all the tools needed for a smooth interaction between parts of an application.
2- More stable. Database synchronization between clusters tends to be better configured and easier to maintain in AWS.
3- Servers does not have any significant change between platforms because they are highly customizable, but in general for storage purposes mainly AWS seems to be cheaper.
4- Consistency & Reliability. While AWS is an extremely useful platform for backups and disaster recovery, it is also extremely reliable. Despite a high-profile outage earlier this year, an independent review found that since 2015, AWS has been “far better at keeping its public cloud service running than either Microsoft or Google.” It also found that 40 percent of the platform’s total downtime during the same time period was tied to a single outage.
5- Flexibility & Scalability. In the early days of Amazon, company engineers developed a computing infrastructure that could be easily scaled up or down to quickly meet the needs of the growing business. This extremely flexible system is now the hallmark of AWS, and is one of the main reasons to choose AWS. Thanks to the company’s massive cloud-based platform, businesses no longer have to deal with the constraints of a physical computing infrastructure, and can rest assured that access to servers and storage is available on demand.

6- Pay-As-You-Go Pricing. We’ve already established that Amazon’s flexible cloud-computing platform allows users to automate routine tasks and quickly scale capacity up or down as needed. But we haven’t discussed the fact that this extreme flexibility lets Amazon offer a pay-as-you-go approach which can greatly improve your business’s bottom line (by as much as 70 percent, in some cases). And of all the reasons to choose AWS, the platform’s flexible pricing structure may be the most popular. Because customers are able to stop and start instances as needed, they only end up paying for

what they use. And the fact that users can easily adjust storage/server levels up or down means overspending on capacity and infrastructure is a thing of the past.

5. EXTRAS

Using Jira as a Project Management Tool. Why JIRA:

Reason 1. JIRA tool can be used for multiple purposes: bug, issue, feature, task tracking. Companies can also use it in some non-standard ways for various business needs such as warehouse automation tool, document flow, expenses optimization etc.(you can see examples of such usage in our recent presentation on JIRA). And of course, JIRA is a great tool for project management.

Reason 2. JIRA tool is easily integrated with external systems and applications. It is often integrated with such tools as Zendesk (customer service), Git (revision control), Salesforce, SugarCRM, Microsoft Dynamics CRM (client relations management).

Reason 3. There is a powerful Atlassian suite of products that can be easily integrated with JIRA and extend its initial functionality (Confluence, Bamboo, FishEye, Stash etc.). Of course, there are many other products that have the same functions, but it is definitely more convenient and effective to use products from the same manufacturer that have been specifically designed to compliment each other, and a lot of users prefer doing just tha

END