



University of Manouba  
National School of Computer Sciences



## Report of The Design and Development Project

---

**Subject: Think'n'Predict**

---

*Authors :*

Ms. Yosra BOUSSAID

Ms. Chayma MOUSSA

*Company:* Think-iT

*Supervisor company:* M. Wajdi KHATTEL

M. Rami HCINI

*Supervisor ENSI:* Dr. Faryel BEJI


*Adress:* 56 Boulevard de la Corniche Lac II Tunis, Tunisia



---

Academic Year:2019 /2020

## Appreciations and signature of supervisor

A large, empty rectangular box with a thin black border, intended for the supervisor's appreciations and signature.

## Abstract

---

**Abstract**—This report was realized as a second year design and development project in partnership with the company "Think.IT". As the industry is flooding with innumerable programming languages, our project aims to make insightful analytics and visualizations of the programming languages evolutions by track, by location and over posted dates. It also makes pertinent prediction of its evolution over the next year.

**Key words :**Analyse, programming language, Time series, predict.

---

## Résumé

---

Ce rapport a été réalisé en tant que projet de conception et de développement pour la deuxième année. C'était en partenariat avec l'entreprise Think IT. Étant donné que l'industrie est embourbée dans d'innombrables langages de programmation, notre projet vise à analyser et à visualiser les langages de programmation par voie, par emplacement et leur évolution au cours des dates publiés. Il fournit également des prévisions sur leur évolution au cours de la prochaine année.

**Mots clés:** Analyser, langage de programmation, séries temporelles..

---

## ملخص

---

اقترح هذا العمل من طرف شركة «ثنك أي تي» في اطار مشروع للسنة الثانية في التصميم و التطوير وهو محدد في البرنامج الرسمي للمدرسة الوطنية لعلوم الاعلامية كأحد المتطلبات من أجل الحصول على شهادة الهندسة. غارقة في عدد لا يحصى من لغات البرمجة، فإن مشروعنا يهدف إلى تحليل لغات البرمجة وتصورها عن طريق وموقع هذه اللغات وتطورها على مدى التواريخ المنشورة. كما أنها توفر توقعات بتطورها على مدى العام المقبل. كلمات مفتاح : سلسلة زمنية، لغة البرمجة، تحليل ا

# Acknowledgement

Throughout this project, we were fortunate to have the kind help as well as supervision of many. For that, we would like to take this opportunity to express our deep sense of gratitude to them.

First and foremost, we thank our supervisors **Wajdi KHATTEL** and **feryel LBEJI** for their endless patience, exemplary guidance, and careful monitoring. We value everything we have learned from them on both academic and personal levels.

Second, we appreciate the trust and the opportunity giving to us by **Think-it**. It allowed us to work within a professional environment on one of the top trending technologies in computer science.

Also, we would like to pay our respects to all of our teachers at the **National School of Computer Science**, not only for the quality education and the measurable training they provided us with during the last two years but also for the thoughtful measures during hard times.

We extend our thanks and appreciation to the **jury members** for the time given to examine and evaluate our work.

Last but not least, we are forever grateful to our families for their continued support and constant encouragement during our study years.

# Contents

<b>Introduction</b>	<b>1</b>
<b>1 Project overview</b>	<b>2</b>
1.1 Host company . . . . .	2
1.1.1 Presentation of the host company . . . . .	2
1.1.2 Activities and areas of expertise . . . . .	2
1.2 Presentation of the project . . . . .	3
1.2.1 Problem setting . . . . .	3
1.2.2 Project steps . . . . .	3
1.3 Expected result . . . . .	3
1.4 Conclusion . . . . .	4
<b>2 Preliminary Study</b>	<b>5</b>
2.1 Study of the existing . . . . .	5
2.1.1 TIOBE . . . . .	5
2.1.2 Stackoverflow . . . . .	6
2.1.3 critics of existing applications . . . . .	6
2.2 Study of art technologies . . . . .	6
2.2.1 Natural Language Processing . . . . .	6
2.2.2 Big data analytics . . . . .	8
2.2.3 Deep learning . . . . .	8
2.2.4 Dashboarding . . . . .	8
2.3 Conclusion . . . . .	9
<b>3 Analysis and Specification of requirements</b>	<b>10</b>
3.1 Specifications . . . . .	10
3.1.1 Actor's identification . . . . .	10
3.1.2 Functional requirements Specification . . . . .	10
3.1.3 Non-Functional Requirements Specification . . . . .	11
3.2 Requirement modeling . . . . .	11
3.2.1 Cases Diagram . . . . .	12

---

3.3	Sequence diagrams . . . . .	12
<b>4</b>	<b>Design</b>	<b>19</b>
4.1	Global Design . . . . .	19
4.1.1	logical architecture . . . . .	19
4.1.2	Physical Architecture . . . . .	21
4.2	Detailed Design . . . . .	23
4.2.1	Activity diagram . . . . .	23
4.2.2	The Data Cleaning or Data Preprocessing . . . . .	24
4.2.3	The Data Extraction . . . . .	25
4.2.4	The forecasting model training . . . . .	25
<b>5</b>	<b>Achievement</b>	<b>26</b>
5.1	Work environment . . . . .	26
5.1.1	Hardware environment . . . . .	26
5.1.2	Software environment . . . . .	26
5.2	Achieved Work . . . . .	28
5.2.1	Data set . . . . .	28
5.2.2	Data Preprocessing . . . . .	29
5.2.3	Data Extraction . . . . .	32
5.2.4	Data Classification . . . . .	33
5.2.5	Forecasting . . . . .	34
5.2.6	Data Visualisation . . . . .	37
	<b>Conclusion</b>	<b>40</b>
	<b>Bibliography</b>	<b>42</b>
	<b>Netography</b>	<b>43</b>

# List of Figures

2.1	Sample of TIOBE dashboard [1]. . . . .	5
2.2	StackOverflow developer survey [2]. . . . .	6
2.3	Big data analytics lifecycle [3]. . . . .	8
3.1	Use case diagram . . . . .	12
3.2	Sequence diagram-Scenario 1 . . . . .	13
3.3	Sequence diagram-Scenario 2 . . . . .	14
3.4	Sequence diagram-Scenario 3 . . . . .	15
3.5	Sequence diagram-Scenario 4 . . . . .	16
3.6	Sequence diagram-Scenario 5 . . . . .	17
3.7	Sequence diagram-Scenario 6 . . . . .	18
4.1	MVC architecture. [4]. . . . .	20
4.2	Package diagram. . . . .	21
4.3	The 3-tiers architecture.[ <b>3-tier architecture</b> ] . . . . .	22
4.4	Deployment diagram. . . . .	23
4.5	Activity diagram. . . . .	24
4.6	Activity diagram of Data Preprocessing . . . . .	24
4.7	Data extraction pipeline. . . . .	25
4.8	Activity diagram for forecasting model. . . . .	25
5.1	Dash architecture[ <b>dash architecture</b> ]. . . . .	27
5.2	Our data set . . . . .	28
5.3	Word2Vec neural network model.[6] . . . . .	29
5.4	Continuous Bag of Word Architecture.[7] . . . . .	30
5.5	Skip-gram Architecture .[ <b>Skip-gram</b> ] . . . . .	30
5.6	Data preprocessing code . . . . .	31
5.7	Building a model . . . . .	32
5.8	Data extraction . . . . .	32
5.9	Our renewed data set . . . . .	33
5.10	Data classification according to posted dates . . . . .	33
5.11	Data classification according to track . . . . .	34

---

5.12 Data classification according to location . . . . .	34
5.13 LSTM algorithm . . . . .	35
5.14 LSTM results . . . . .	35
5.15 SARIMA model parameters . . . . .	36
5.16 SARIMA results . . . . .	36
5.17 Fbprophet model . . . . .	37
5.18 Dash setup . . . . .	37
5.19 Dash packages . . . . .	38
5.20 Bar chart layout . . . . .	38
5.21 Dashboard layout . . . . .	39
5.22 Dashboard layout . . . . .	39
5.23 Bar chart callback decorator . . . . .	40



# List of Tables

2.1	Critics of existing applications . . . . .	7
5.1	PCs characteristics . . . . .	26

# Glossary of Acronyms

<b>NLP</b>	<i>Natural Language Processing</i>
<b>NLTK</b>	<i>Natural Language Toolkit</i>
<b>AI</b>	<i>Artificial Intelligence</i>
<b>US</b>	<i>United States</i>
<b>ENSI</b>	<i>Ecole Nationale des Sciences de l'Informatique</i>
<b>IT</b>	<i>Information Technology</i>
<b>CSV</b>	<i>(Comma Separated Value)</i>
<b>LSTM</b>	<i>(Long Short-Term Memory)</i>
<b>IT</b>	<i>(Information Technology)</i>
<b>API</b>	<i>(Application Programming Interface)</i>
<b>MVC</b>	<i>(Model-View-Controller)</i>
<b>SARIMA</b>	<i>(Seasonal Autoregressive Integrated Moving Average)</i>
<b>Json</b>	<i>JavaScript Object Notation</i>
<b>HTTP</b>	<i>Hypertext Transfer Protocol</i>

# General introduction

With immeasurable amounts of new data produced and analyzed every day, companies rush to mine as much useful information as possible in order to precisely gain a brief outline of what that data is telling them. Since dashboards summarize all these pieces of information in graphs, charts, metrics, and tables and show them on a single page, they seem to be the perfect solution for enterprises.

In this context, our project consists of building an insightful and interactive dashboard, that analyzes and visualizes the current trends in IT job offers in United States and give a big picture of the most in demand programming languages in order to help developers and job seekers to focus on their learning. Also, our dashboard visualizes the most in demand programming languages in the future. This dashboard is designed for Think-It employees.

In this report, we are going to introduce our work process. In the first chapter, we will start by presenting the host company, Think-IT, then we are going to introduce the problem, and lastly, we will give the expected result. In the second chapter, we are going to present existing similar solutions and define the technologies needed. Moving to the third chapter, we will analyze the specifications and requirements. In the fourth chapter, we will represent the design of our solution. In the fifth chapter, we will make clear how we implemented our dashboard. Finally, we conclude our work by giving an evaluation and suggesting a future amelioration of our project.

# Chapter 1

## Project overview

### Introduction

This chapter presents the general framework of the project. Initially, we will introduce the host company. Then, we will throw some light on the problem we intend to solve as well as our proposed solution.

### 1.1 Host company

This part covers the presentation of the host company in addition to its main activities and areas of expertise.

#### 1.1.1 Presentation of the host company

Think It, is a group of ambitious engineers who stands for the mission of unlocking human potential through technology. It was founded in 2017, in Tunisia, thanks to its bright, young people who promote a better future for the country besides including North Africa on the map as a digital innovation hub with highly-qualified and affordable technical talent. [Think-It].

#### 1.1.2 Activities and areas of expertise

Our host firm activities are:

- Web Development.
- Data Security.
- Micro-services.
- Technology Staffing.

- Blockchain.
- Data Science and Machine Learning.
- Cloud Services.
- Social Impact and Talent Development.

## 1.2 Presentation of the project

In this section we expose the general context of “Think’n’Predict” and its targets and then, we look into the process of achieving the final purpose.

### 1.2.1 Problem setting

Over the years, the demand for skilled developers remains a challenge for employers as a result of the rapid evolution in computer science technologies as well as the popularity of several programming languages. Also, developers have to know precisely where to target their learning in order to gain a good job prospect which triggered an urge for discovering a solution to help employers and employees.

### 1.2.2 Project steps

The essential milestones of Think’n’Predict project are :

1. Extracting programming languages from job descriptions.
2. Analyzing the programming languages.
3. Visualizing the programming languages on a dashboard.
4. Forecasting the evolution of the programming languages demands.

## 1.3 Expected result

The expected result of our project is a dashboard that visualizes the most demanded programming languages classified by different tracks and locations in the US. Also, this dashboard will present the anticipation of upcoming demands in job offers.

## 1.4 Conclusion

In this chapter, we have presented our host firm Think-It as well as our problem statement. Also, we have talked about our project goal. In the coming chapter, we will introduce the state of the art of data extraction, data analysis, data visualization, and forecasting.

# Chapter 2

## Preliminary Study

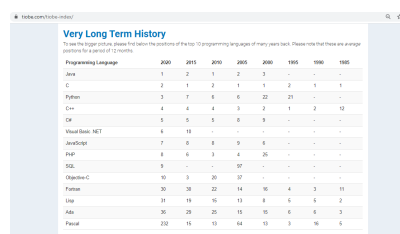
In order to implement the best model of “Think’n’Predict” project, we have to follow an organized process. In the current section, we will survey the widely used methodologies, compare them and select the most appropriate one to pursue in the accomplishment of our project.

### 2.1 Study of the existing

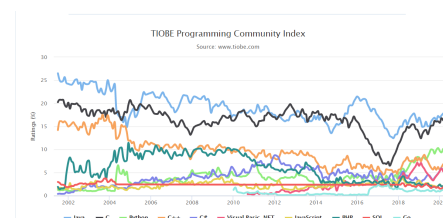
We devote this part to present projects offering a similar solution to ours, then we will study their deficiency and justify the need for the hallmark of our solution.

#### 2.1.1 TIOBE

TIOBE software is a company, found in 2000, expert in evaluating and tracking the quality of software. Tiobe has developed its own dashboard in order to visualize the most demanded skills for its employee which is clearly illustrated in the figure 2.1-a and 2.1-b .



a. TIOBE index for May 2020



b. Tiobe community index

Figure 2.1 – Sample of TIOBE dashboard [1].

### 2.1.2 Stackoverflow

StackOverflow is a website dedicated to programmers in order to interact and exchange information in the field of IT. Since 2011, Stackoverflow community elaborated “developer survey“, a study that visualizes the most used skill sets in a dashboard.

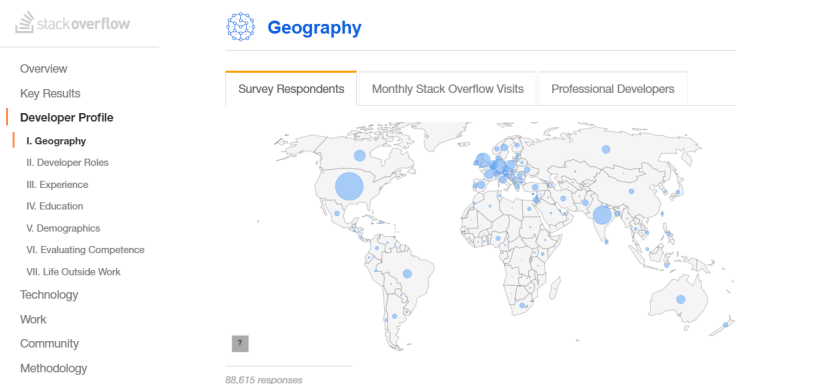


Figure 2.2 – StackOverflow developer survey [2].

### 2.1.3 critics of existing applications

The table 2.1 presents the advantages and disadvantages of the existing solutions. We perceive that both existing applications have notable strength whereas their characteristics did not suit the demand of think.IT community. Therefore, it was suggested to develop a specific dashboard in order to visualize from IT job descriptions, the most demanded programming languages.

## 2.2 Study of art technologies

In this section, we will define the different technologies and theoretical tools we need for our project.

### 2.2.1 Natural Language Processing

NLP is Natural Language Processing. This is the set of techniques that allow a machine interface to automatically analyze and process the written and oral statements of an individual and to express the necessary answers.



	TIOBE	Stackoverflow
Advantages	<ul style="list-style-type: none"> <li>• Interactive dashboard.</li> <li>• Multiple charts.</li> <li>• Precise and brief.</li> </ul>	<ul style="list-style-type: none"> <li>• Interactive dashboard.</li> <li>• Multiple charts with a big variety of information.</li> </ul>
Disadvantages	<ul style="list-style-type: none"> <li>• Analyzed data belongs to TIOBE's clients.</li> <li>• The dashboard presents the analysis of the previous year's data.</li> </ul>	<ul style="list-style-type: none"> <li>• All users of StackOverflow participate in the survey without control.</li> <li>• The dashboard is delivered by the end of every year without forecasting.</li> </ul>

Table 2.1 – Critics of existing applications

### 2.2.2 Big data analytics

Regarding the volume, velocity, and variety characteristics of big data, the process of analyzing it differs from ordinary data analysis. The nine steps of big data analytics are illustrated in figure 2.3. In our project, we have to start from stage four which is data extraction from job descriptions.

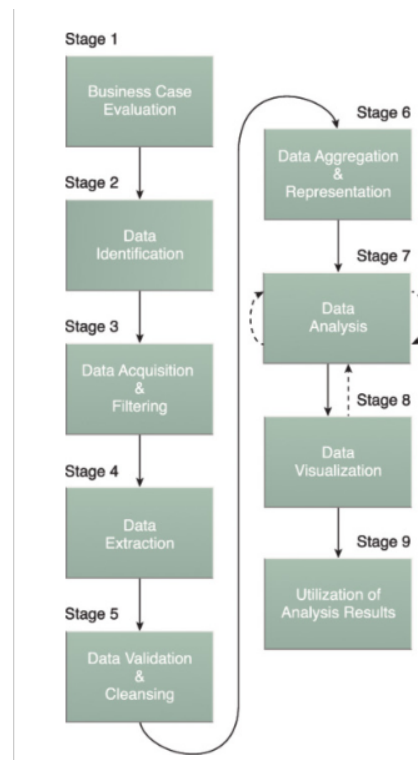


Figure 2.3 – Big data analytics lifecycle [3].

### 2.2.3 Deep learning

Deep Learning is a new subfield of Machine Learning research, which has been introduced to move Machine Learning closer to one of its original goals: Artificial Intelligence. Deep Learning consists of learning multiple levels of representation and abstraction to make sense of different types of data. (Li Deng and Dong Yu. Deep Learning; Methods and Applications)

### 2.2.4 Dashboarding

In order to visualize a large set of data, we can use dashboards. “A dashboard is a visual display of the most important information needed to achieve one or more objectives that

has been consolidated on a single computer screen so it can be monitored at a glance.”  
[Dashboarding]

## **2.3 Conclusion**

In this chapter, we have highlighted some existing solutions in order to build an innovative solution. And we have defined the most important terminologies that match the implementation of our project.

## Chapter 3

# Analysis and Specification of requirements

Through this chapter, we will incept the technical specification phase. Starting from the functional requirements until getting to non-functional ones, we will explain the use case throughout the sequence diagram.

### 3.1 Specifications

In the section, we will list the several needs of our solution.

#### 3.1.1 Actor's identification

An actor is an external element that interacts with the system. Our solution is destined to be used by 3 types of clients :

- The administrator: He is the one in charge of managing the database.
- The website's visitor: he can visualize the charts of programming languages ' evolutions by track and by field.
- The customer: Once registered as a customer, he can visualize the charts of programming languages ' evolutions by track and by field as well as the programming languages forecasting.

#### 3.1.2 Functional requirements Specification

Functional requirements consist of primary functions that the application being developed must exhibit. Our application to be developed offers the following use cases for the actors:

- **For the Administration.**
  - Add further data to the data set.

- Create new types of charts in the website.
  - Delete charts from the website.
  - Add the website's visitor who has filled the membership registration to the database as a customer.
- **For the Website's visitor:**
    - Visualize the existing programming languages.
    - Visualize charts of programming languages' evolutions by track and field.
- **For the customer:**
    - Get authenticated.
    - Visualize the existing programming languages in order to know the job's technical requirement.
    - View the charts of programming languages' evolutions by track and field.
    - View the charts of programming language forecasting.

### 3.1.3 Non-Functional Requirements Specification

Our website has to satisfy a list of non-functional requirements such as:

- **Usability:** The system must be easy to use.
- **Availability** Our system must be always available for a user.
- **Security:** the customer can't access his account unless he has been authenticated.

## 3.2 Requirement modeling

In the following, the requirements will be documented in the format of use cases and sequence diagrams.

### 3.2.1 Cases Diagram

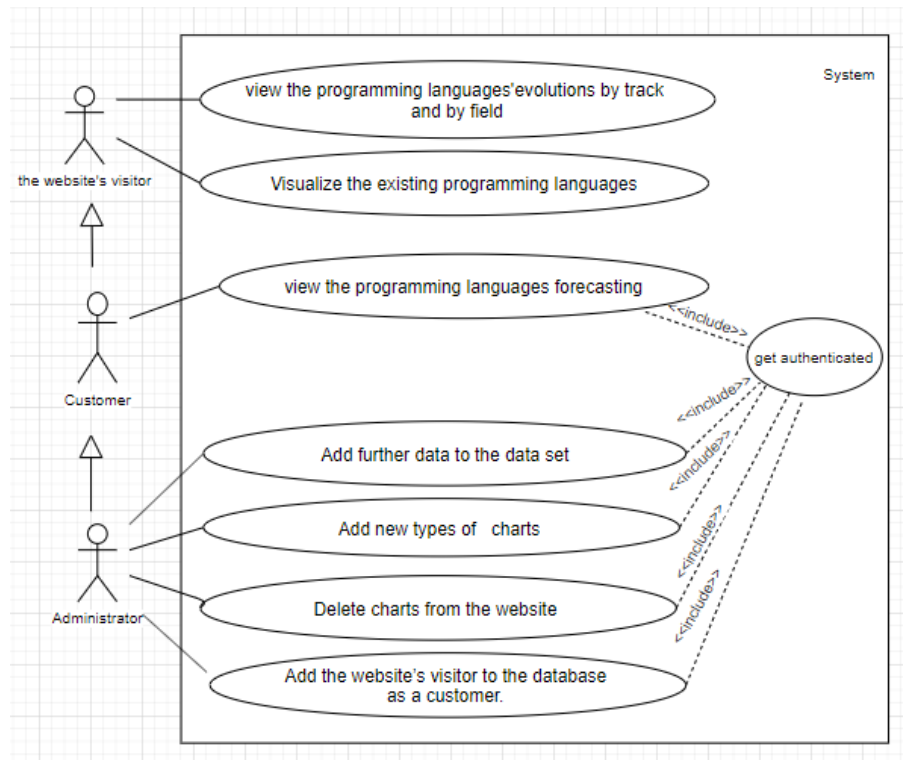


Figure 3.1 – Use case diagram

A use case diagram represents the interaction between the user and the system that shows the different uses cases in which the user is involved.

## 3.3 Sequence diagrams

A sequence diagram makes easy the understanding of each interaction with the system. In this section, we are going to show some scenarios.

**Scenario 1:** The authentication of the customer

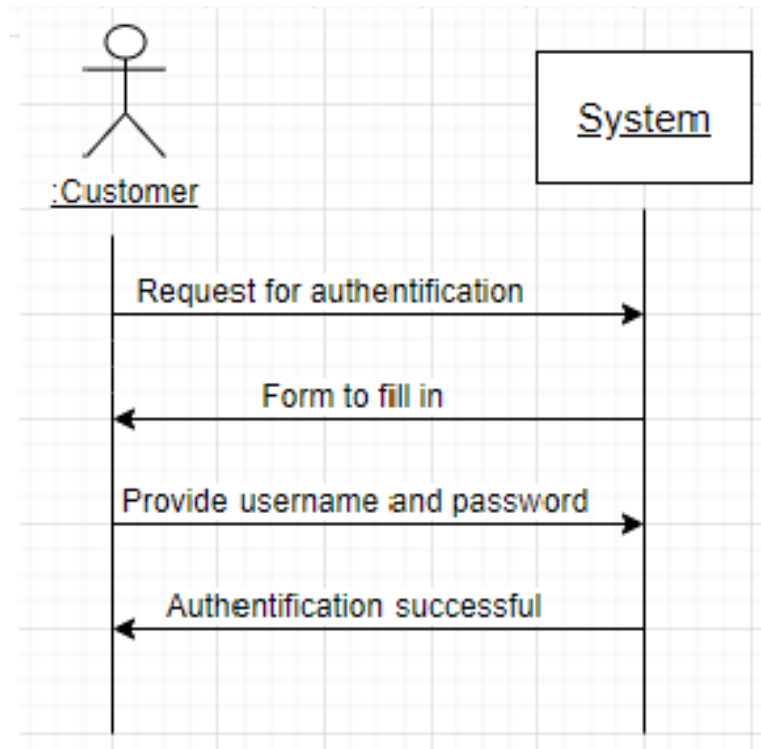


Figure 3.2 – Sequence diagram-Scenario 1

**Scenario 2:** When the user wants to see all the programming languages that exists in order to know the job 's technical requirement. His request will be transmitted to the system and then the programming languages list will be displayed as dedicated in the figure 3.3.

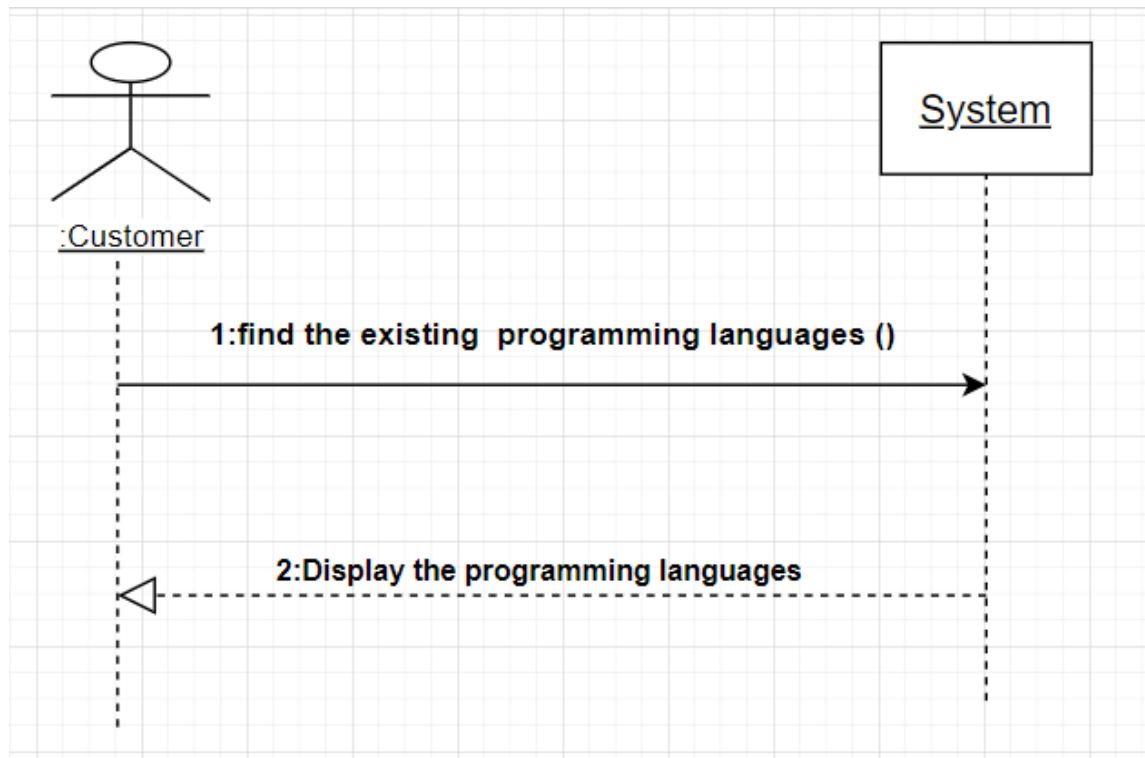


Figure 3.3 – Sequence diagram-Scenario 2

**Scenario 3:**When the user wants to see all the programming languages that appears in a region from the map in order to know the most demanded in that region. His request will be transmitted to the system and then the charts will be displayed as dedicated in the figure 3.4 .



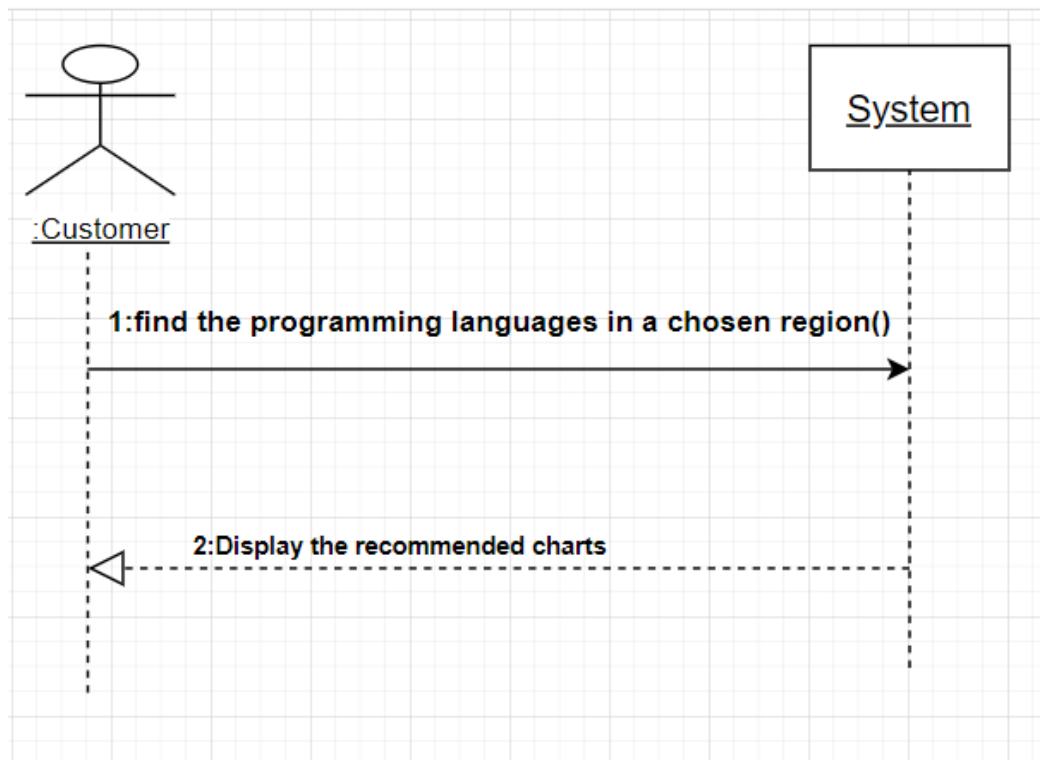


Figure 3.4 – Sequence diagram-Scenario 3

**Scenario 4:** When the user wants to see all the programming languages that appears in a chosen field in order to know the most demanded in that field. His request will be transmitted to the system and then the charts will be displayed as dedicated in the figure 3.5.

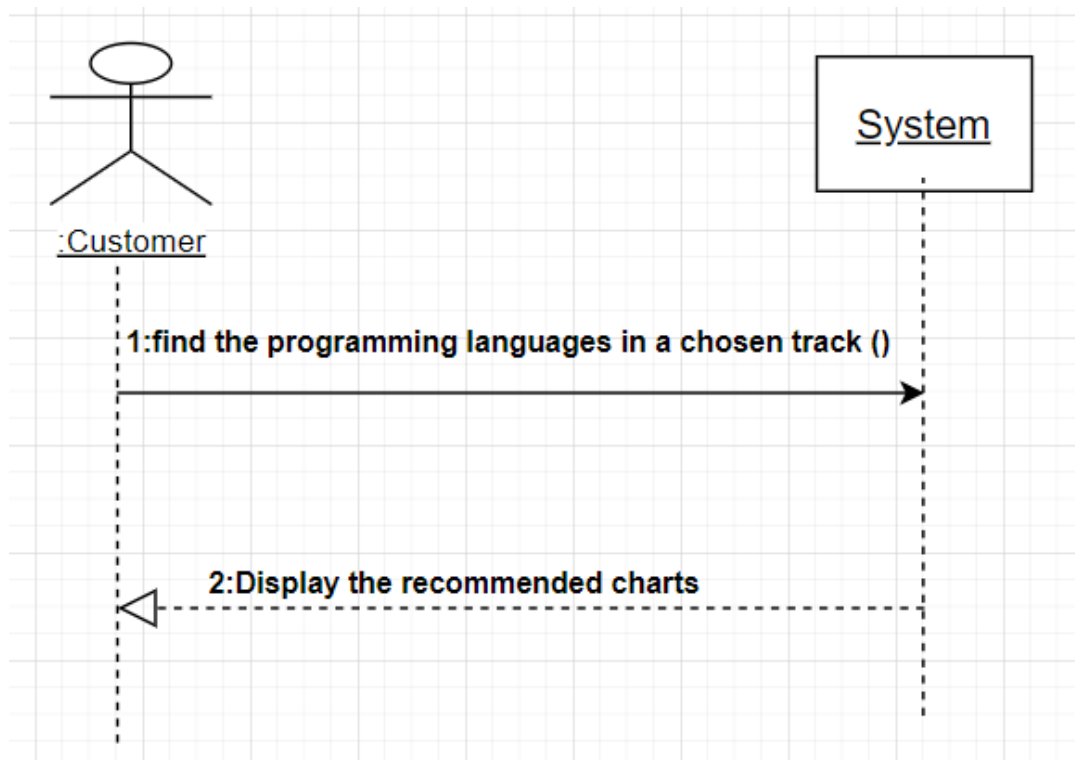


Figure 3.5 – Sequence diagram-Scenario 4

**Scenario 5:**The user wants to see the evolution of a specific programming language over the given dates. His request will be sent to the system and then the charts will be displayed as shown in the figure 3.6.

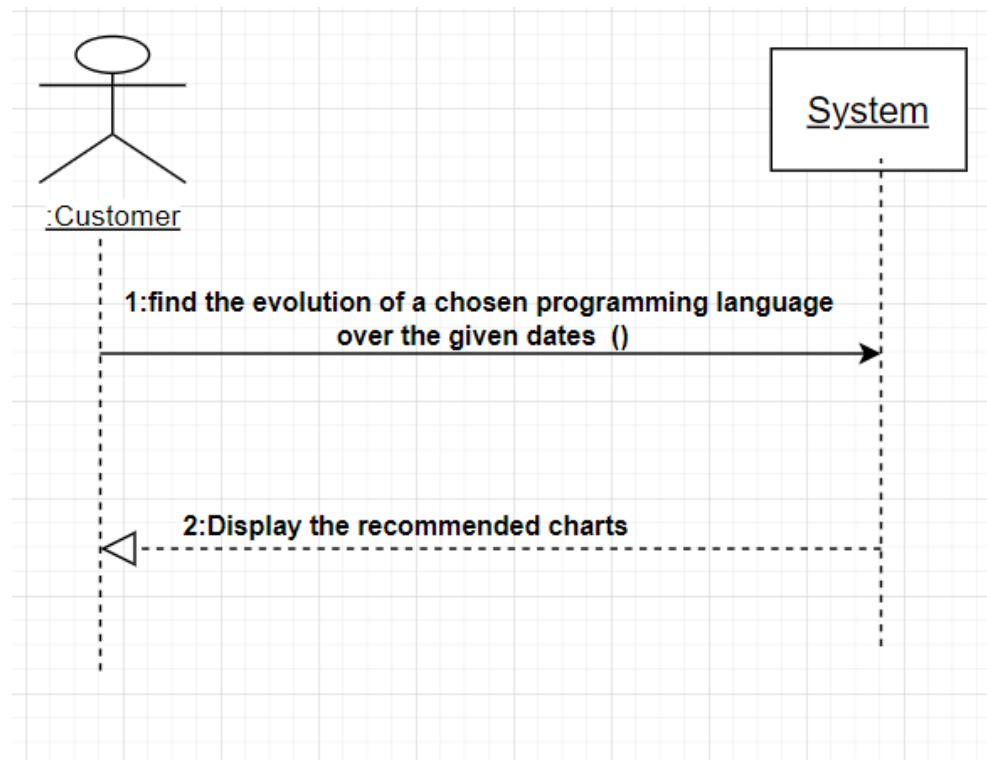


Figure 3.6 – Sequence diagram-Scenario 5

**Scenario 6:** The user wants to see the forecast of a specific programming language over the next year. His request will be sent to the system and then the charts will be displayed as shown in the figure 3.7 .

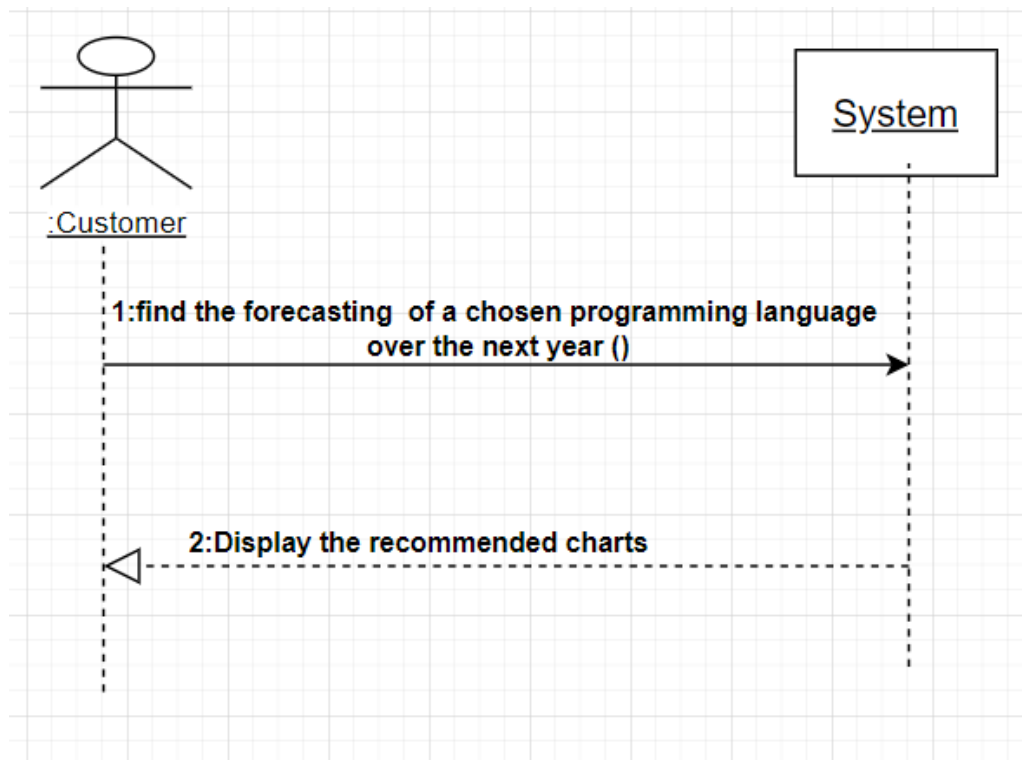


Figure 3.7 – Sequence diagram-Scenario 6

## Conclusion

In this chapter, we have detailed the functional and nonfunctional specifications of our application, as well as the description of the actors, use cases and its variation with the sequence diagrams. In the next chapter, we will focus on the description of the architectural aspect of our solution.

# Chapter 4

## Design

In the previous chapter, we have made a detailed study of the requirements needed for our project. Now we shall elaborate on the design and architecture of the software; starting by implementing the global design both physical and logical, and finishing with detailed architecture of every part of the system.

### 4.1 Global Design

An adequate and solid design is required for understanding and solving a problem. It should be easy to understand, and cheap to modify in order to ensure proper operation and good performance.

#### 4.1.1 logical architecture

The MVC ("Controller view Model") is a development architecture that consists of separating the source code into modules. In our case, this development architecture is the most appropriate that makes easy the implementation of our web application since it is commonly used in web application.

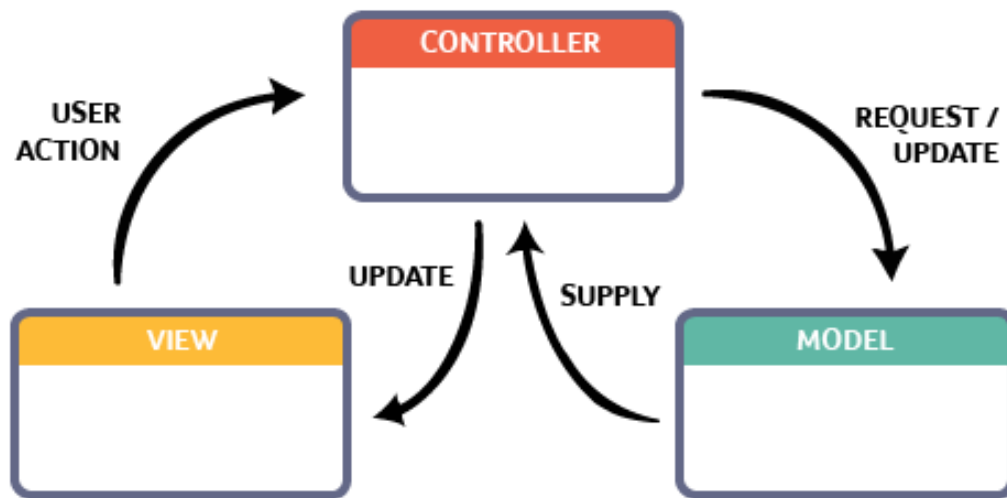


Figure 4.1 – MVC architecture. [4].

- The model: The model defines only the pure application data, it does not contain logic describing how to present the data to the user. Indeed, it allows access to information, modification, addition, updating...
- The view: The view is the user interface, which displays the data from the model on the screen but it does not know what this data means or what the user can do to manipulate it.
- The controller: The controller represents all the application execution that exists between the view and the model. It is responsible for controlling the flow of the application execution by returning the response requested. The controllers of our application are grouped in a file called `views.py`.

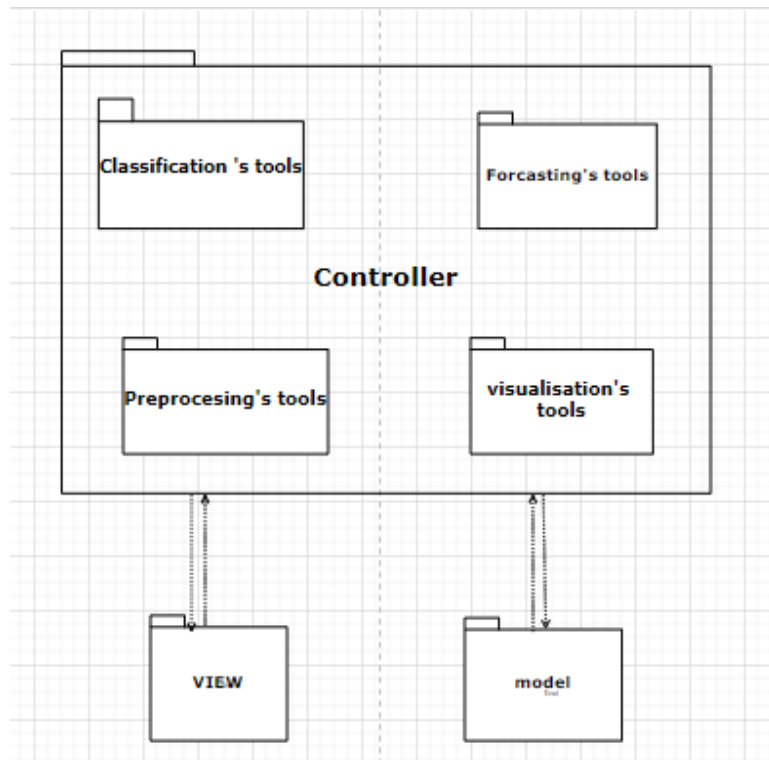


Figure 4.2 – Package diagram.

### 4.1.2 Physical Architecture

The adequate physical architecture for our web application is the 3-tier architecture providing speed of development, scalability, performance, and availability. Also called the three-layers architecture it is divided into the presentation layer, the application layer and the database layer as shown in the figure 4.3.

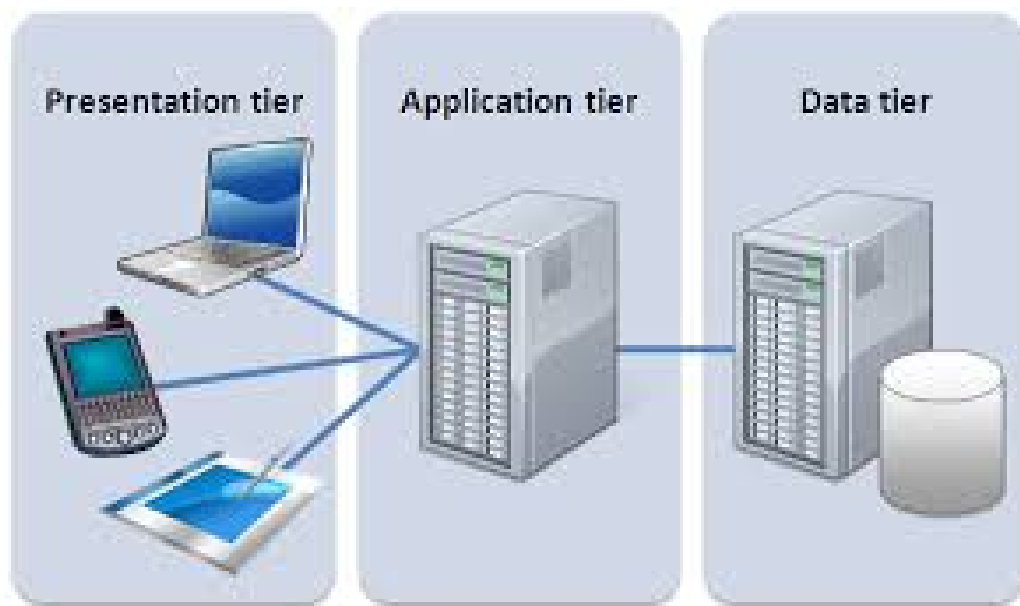


Figure 4.3 – The 3-tiers architecture.[3-tier architecture]

- **Presentation tier :** It is the user interface where the web page of the application is located. It is accessible through a web browser or web-based application and linked with the other layers through API calls.
- **Application tier:** This layer contains all the application execution. It also moves and processes the data between the two other layers. It's often written in Java,.NET, C#, Python, C++, etc.
- **Database tier:** This layer is the data access layer that stores or retrieves the information from data bases or file systems. Data in this layer is kept independent of the application servers or the application execution.



The deployment diagram shown in figure 4.4 represents the physical infrastructure that we used for our system and the way its components are distributed as well as their relationship to each others.

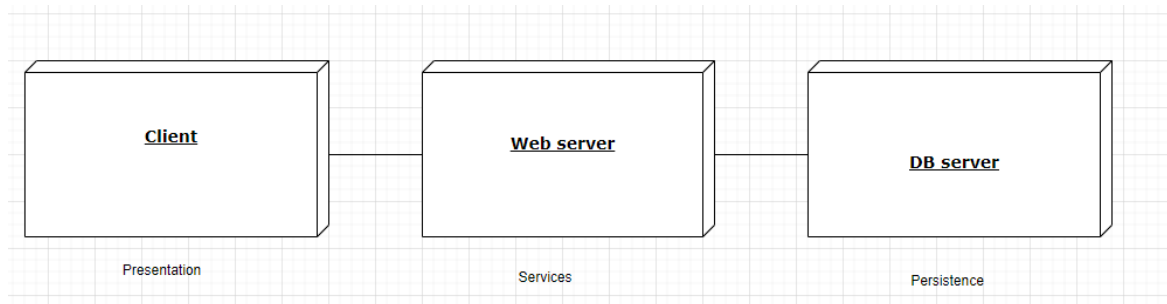


Figure 4.4 – Deployment diagram.

## 4.2 Detailed Design

After presenting the general design of our project, we are going to dive in its details. Our contribution with Think-IT firm consists in these four components: Data Cleaning, Data extraction, Data Classification, Data Visualisation and Forecasting. For a better understanding, we are going to look at each of these components in details.

### 4.2.1 Activity diagram

We are going to represent the dynamic aspects of the system using activity diagram. It is basically a flowchart to represent the flow from one activity to another as shown in the figure 4.5.

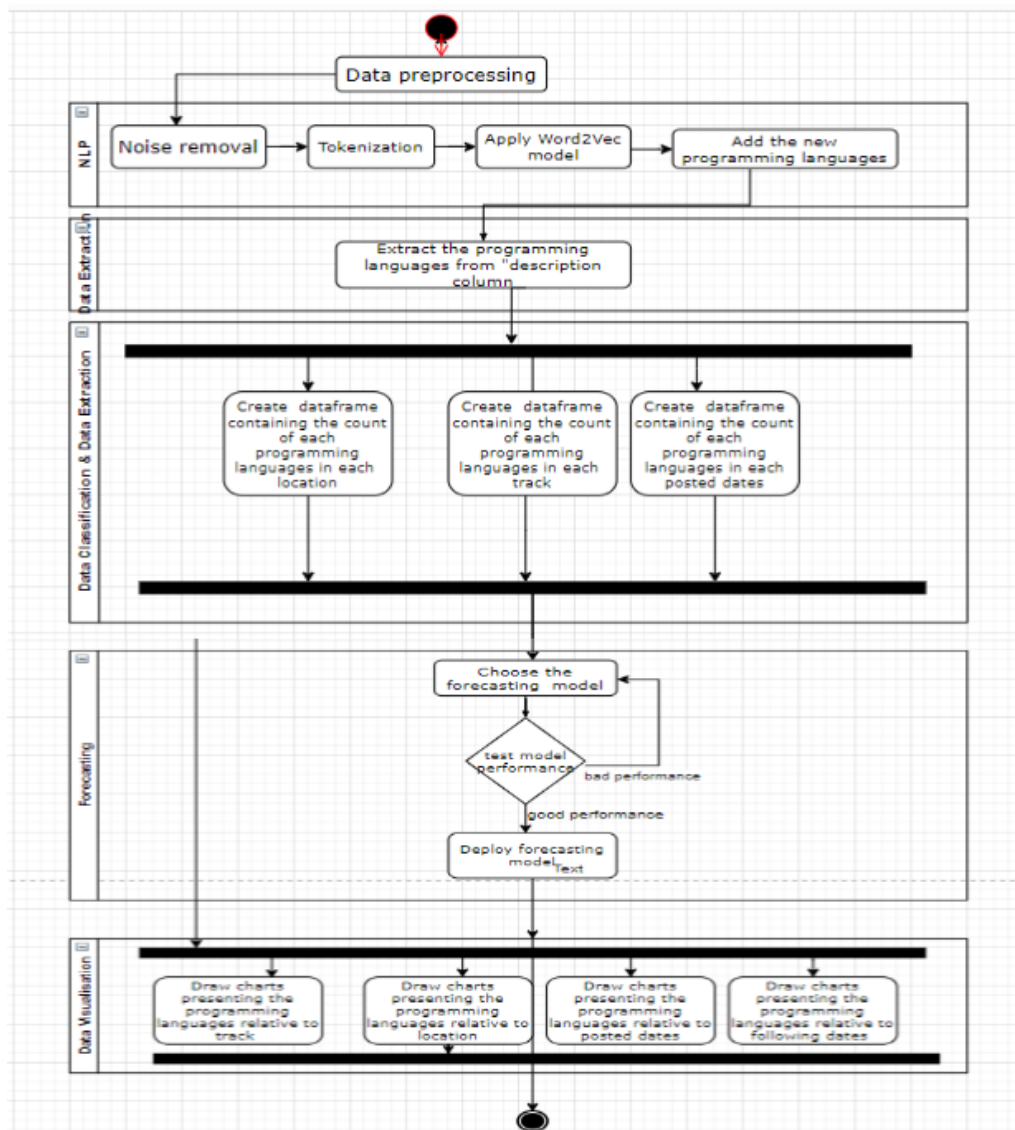


Figure 4.5 – Activity diagram.

### 4.2.2 The Data Cleaning or Data Preprocessing

This part consists in extracting new programming languages from the data set that do not exist in our list of programming languages.

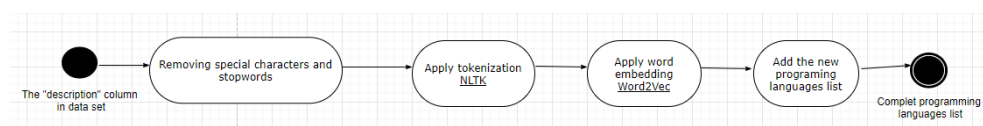


Figure 4.6 – Activity diagram of Data Preprocessing .

### 4.2.3 The Data Extraction

The second phase consists in extracting the programming languages words that exist in the column "description" in the data set and add it in a new column "extracted-words".

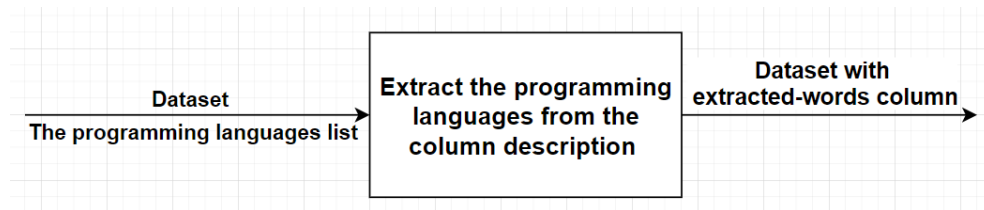


Figure 4.7 – Data extraction pipeline.

[5]

### 4.2.4 The forecasting model training

In order to predict the progression of the programming languages over the next months, we must have a relevant model to be trained as shown in the figure 4.8.

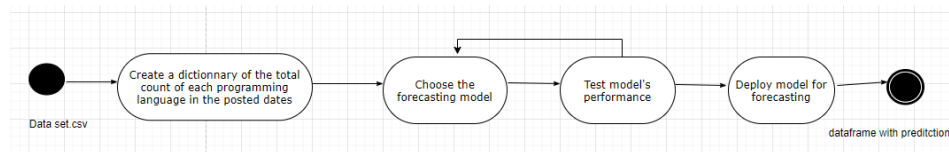


Figure 4.8 – Activity diagram for forecasting model.

## Conclusion

In this chapter we have described the design of our application. We elaborated the architectural solutions that cover an overview of its global and detailed aspects.

# Chapter 5

## Achievement

After detailing the design and architecture of our project, it's time to specify the development work that we have achieved. We will start by presenting the hardware and software environment used in the implementation of the application. Then, we will talk about the implementation of each phase and the reasons behind choosing its technologies.

### 5.1 Work environment

This section consist in presenting the hardware environment adding to the software environment to develop our project.

#### 5.1.1 Hardware environment

To achieve our project, we used our machines that are characterized by:

Memory	Operating system	Processor	Hard disk
Windows	Intel core i3 CPU 2.00GHZ	4GB	500GB
Windows	Intel(R) Core(TM) i5-6200 CPU @2.30GHZ 2.4GHZ	4.00GB	5.00GB

Table 5.1 – PCs characteristics

#### 5.1.2 Software environment

We will present the software tools needed to achieve our project.

- **IDE Jupyter Notebook** is a client-server application used for creating and sharing documents in the browser. These documents are able to contain live code, equations, paragraphs and visualizations. We used it for Data cleaning and Data classification.

- **Cloud Service: Google colab:** We used this free cloud service to develop our deep learning code.

### Technological choices

In this part, we will introduce the programming languages, and libraries that we have used to build our project.

- **Programming language**

**Python:** is a high-level programming language and an object-oriented approach characterized by providing a large standard library and supporting programming paradigms. Python is also easy to use as consequence we opt for it to facilitate our task in implementing machine learning and deep learning.

- **Frameworks:**

**Dash:** Dash is a python framework from plotly for building web applications. It is written on top of Flask, Plotly.js, React and React Js. Dash applications:

- consist of a Flask server that communicates with front-end React components using JSON packets over HTTP requests.
- are written purely in python, so NO HTML or JavaScript is necessary.

Dash is ideal for building data visualization apps with highly custom user interfaces. It's particularly suited for anyone who works with data in Python.

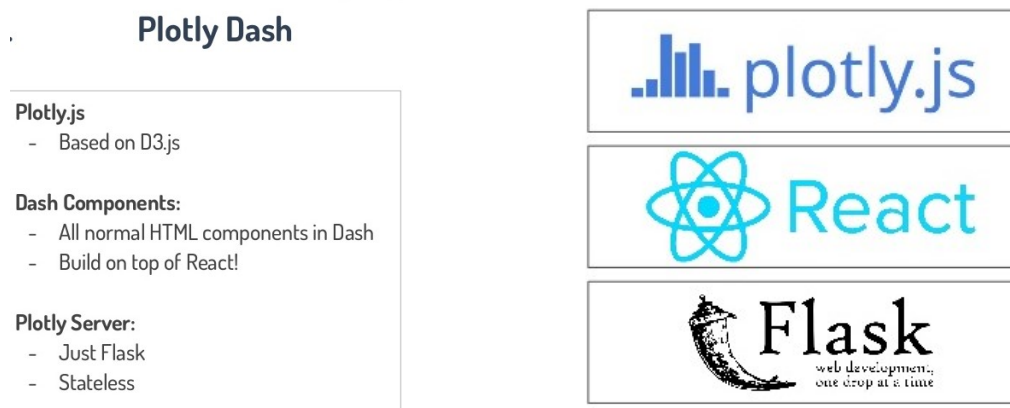


Figure 5.1 – Dash architecture[dash architecture].

- **Tools:**

**OverLeaf:**Online LaTeX editor Draw.io: Online diagram software

- **Libraries :**

Pandas,Spacy,numpy,NLTK,gensim,pickle,csv

**Pandas:** Used for data manipulation and analysis. Pandas library provides data structures and operations for manipulating time series.

**Spacy:**This library is used for processing and analysing data in Natural Language Processing. It produces word vectors and use more than fifty languages in tokenization.

**Numpy:**is used to support multi-dimensional arrays and matrices, and to operate mathematical functions on these arrays.

**NLTK:** is the main tool for natural processing and machine learning. It supports classification, stemming, tagging, semantic reasoning and tokenization.

**Gensim:** We used gensim to identify semantic similarity between two documents through vector space modeling.

**Pickle:**We used pickle to implement binary protocols for a Python object structure.

**CSV:**We implement csv module to read our data in CSV format.

## 5.2 Achieved Work

### 5.2.1 Data set

Think-IT company provide us with a data set composed of 410646 rows and 11 columns. It has 10 features: title, companyname, location, description, posteddates, ds, webdevelopment, mobile, cybersecurity.

Ds, webdevelopment, mobile and cybersecurity are the features that represent the fields that the company work in. If it has a value equal to one ,this means that the company works in this field, if not means that the company do not work in it.

	title	company_name	location	description	posted_date	ds	web_development	devops	mobile	cyber_security
0	\n Software Engineer	Twitter	San Francisco, CA 94103 (South Of Market area)	We are responsible for providing a secure, rel...	2019-04-27	0	0	0	0	0
1	\n Technical Customer Support Analyst	SPS Commerce	Minneapolis, MN 55402 (Downtown West area)	Description\nWe are currently looking for a Cu...	2019-04-13	1	0	0	0	0
2	\n Global Financial Crimes Transact...	MUFG	New York, NY 10020 (Midtown area)	Discover your opportunity with Mitsubishi UFJ ...	2019-04-27	1	0	0	0	0
3	Analyst 3, Billing Systems	Comcast	Atlanta, GA 30339	Comcast brings together the best in media and ...	2019-04-04	0	0	0	0	0
4	Computer Software Analyst II -	Newport-Mesa Unified School District	Costa Mesa, CA 92626	\$88,343 - \$108,229 a yearThe Vacancy, Basic Fu...	2019-03-23	1	0	0	0	0
5	\n Marine Electrician/Electrical Te...	The GBS Group	Virginia Beach, VA 23452 (North Central area)	Overview\nThe GBS Group, an engineering servic...	2019-04-18	0	0	0	0	1
6	\n UI Developer	Harbinger Partners	Omaha, NE 68197	Duties and Responsibilities\nDevelop workflow ...	2019-04-11	0	1	0	0	0

Figure 5.2 – Our data set

### 5.2.2 Data Preprocessing

Think-IT provide us with a file in a json format containing a list of programming languages. We aim to extract all the programming languages that exist in the column description in our data set. We opt for word embeddings that consist in mapping a word to a vector of real numbers. It captures the words with similar meaning thanks to having similar vectors. Word embedding have three algorithms:

- **Embedding layer:** is a word embedding that can be learned as part of a deep learning model. It requires a cleaned input data and integer encoded. Known as the first hidden layer of a network, it has three arguments: input-dim, output-dim and input-length. This approach is also as the slower approach due to needing a lot of training data.
- **Word2Vec:** Word2Vec is a technique to construct such an embedding by grouping related models. It is a shallow two-layered neural network with only a hidden layer between input and output.

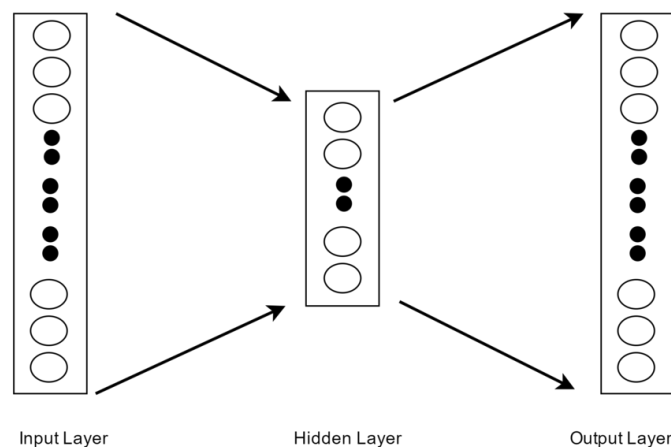


Figure 5.3 – Word2Vec neural network model.[6]

Word2Vec has two architectures :

- **Continuous Bag of words (CBOW)** In Continuous Bag of words, the current word is predicted using the window of surrounding context windows as the figure shows.

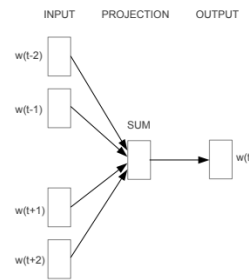


Figure 5.4 – Continuous Bag of Word Architecture.[7]

### – Skip-gram

Skip-Gram approach is used to predict a sentence given an input word.

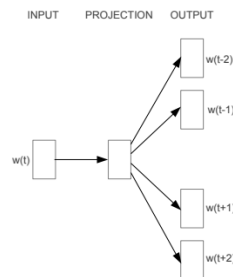


Figure 5.5 – Skip-gram Architecture .[Skip-gram]

This approach is characterized by being an efficient and high-quality word embeddings thanks to requiring low space and small time complexity. It allows larger embeddings from larger corpora of text.

- **GolVe**: is an approach coined from Global Vectors and an extension of word2vec method. It is an unsupervised model that constructs word co-occurrence matrix using statistics across the whole text. It results a good word embeddings.

We choose to work with Word2Vec thanks to its benefits.

We start by preparing the data. First, we load and open the programming languages 's file by importing the **pickle** module, then we convert the content into a list. The second step consists in loading the data set and opening it as a dataframe thanks to pandas module. Then, we convert the column description into a list. The next step involves importing **RegexpTokenizer** from **nlTK** toolkit to split each string into substrings using a regular expression. After converting all tokens in lower case, we remove english stop words using stop word module from **nlTK** toolkit.



A screenshot of a code editor window. The editor has a light gray background and a dark gray sidebar on the left. The code is written in Python and is color-coded. The code imports several libraries: csv, pandas as pd, pickle, nltk, and itertools. It then loads a pickle file named 'list\_prog.pickle'. A loop iterates over the loaded data, processing each item by lowercasing, splitting, and tokenizing it. It also downloads the NLTK stopwords and filters out any words that are in the stopwords list. The final result is a list of processed words.

```
+ Code + Texte
import csv
import pandas as pd
import pickle
import nltk
from nltk.tokenize import RegexpTokenizer
from nltk.corpus import stopwords
import itertools
list_programming_languages = pickle.load(open('list_prog.pickle','rb'))
list_programming=[]
for i in list_programming_languages:
    ch=i.lower().split('/')
    list_programming.append(ch)
list__programming=itertools.chain.from_iterable(list_programming)
listprogramming=list(list__programming)
nltk.download('stopwords')
desc=df['description'].values.tolist()
newlist=list()
for line in desc:
    rempunc=RegexpTokenizer("[\w']+")
    tokens=rempunc.tokenize(line)
    words=[w.lower() for w in tokens]
    stopwords=set(stopwords.words('english'))
    words=[w for w in words if not w in stopwords]
    newlist.append(words)
```

Figure 5.6 – Data preprocessing code

Now, it is time to built our model. We implement **gensim** module which is a topic modeling toolkit, providing an implementation of word2vec. We apply these parameter values :

- **Min-count = 1:** It will ignore all the words with a total frequency lower than than 1.
- **Size = 50:** The dimensionality of the word vectors is equal to fifty.
- **Workers = 3:** We apply three threads to train the model.
- **Window = 3:** Maximum distance between the current and predicted word within a sentence is equal to three.
- **Sg =1:** It is a training algorithm and 1 for skip-gram and 0 for a Continuous bag of words. We choose skip-gram because it works well with the length of our training data.

Then, we search for the top-10 most similar entities by applying the function `most-similar()` and we add it to a list.



```
import gensim
model=gensim.models.Word2Vec(newlist, min_count=1,size= 50,workers=3, window =3, sg = 1)

# In[16]:

p=[]
for i in listprogramming:
    try:
        m=model.most_similar(i,topn=10)
        for x in m:
            p.append(x[0])
    except KeyError:
        print ("not in vocabulary")
```

Activate Windows  
Go to Settings to activate Windows.

Figure 5.7 – Building a model

### 5.2.3 Data Extraction

We used data extraction in order to add a new column containing the extracted programming languages from "description " column. To achieve this task, first we load en-core-web-sm from spacy module to make tokenization. It is a small English model trained on written web text involving vocabulary, vectors, syntax and entities. Then, we create a function to extract all the programming languages as shown in the figure 5.8.



```
nlp = spacy.load("en_core_web_sm")
[ ] def extract_data(row):
    try :
        description = row['description']

        doc = nlp(description)

        words=[]
        for token in doc :
            ch=token.text.lower()
            if (ch in listprogramming):
                words.append(ch)
        return words
    except:
        return []

[ ] df['extracted_words'] = df.apply(extract_data, axis=1)
```

Activate Windows  
Go to Settings to activate Windows.

Figure 5.8 – Data extraction

Finally, we apply this function and we add its results in a new column as in indicated in the figure 5.9.

```
[ ] df.head()
```

title	company_name	location	description	posted_date	ds	web_development	devops	mobile	cyber_security	extracted_words
\n are eer	Twitter	San Francisco, CA 94103 (South Of Market area)	We are responsible for providing a secure, rel...	2019-04-27	0	0	0	0	0	[python, go, scala, java]
\n lcal ner port lyst	SPS Commerce	Minneapolis, MN 55402 (Downtown West area)	Description\nWe are currently looking for a Cu...	2019-04-13	1	0	0	0	0	[xml]
lobal cial nes ct...	MUFG	New York, NY 10020 (Midtown area)	Discover your opportunity with Mitsubishi UFJ ...	2019-04-27	1	0	0	0	0	[matlab, r, python, sql, vba]

Figure 5.9 – Our renewed data set

### 5.2.4 Data Classification

Data classification is the process of organising data in a specific categories. We use it to create a data frame containing the count of each programming language in the posted dates as demonstrated in the figure 5.10.

```

a_language_date={i:0 for i in r}
dates_programming_languages_counts = { j :a_language_date.copy() for j in listprogramming }
def count_prog_lang(row):
    for lang in row['extracted_words']:
        # Increase the value count of the prog.lang in that date
        dates_programming_languages_counts[lang][row.posted_date] += 1
df.apply(count_prog_lang,axis=1)
p=pd.DataFrame(r,columns=['posted_dates'])
dic={i:list(dates_programming_languages_counts.get(i).values()) for i in listprogramming }
ds = pd.DataFrame(dic)
h=pd.concat([p,ds],axis=1)
h.head()

```

	posted_dates	javascript	r	dart	perl	c#	delphi	object pascal	php	common lisp	other(s):	hack	groovy	vba	kotlin	c	sql
0	2019-03-22	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	2019-03-23	4	0	0	0	0	0	0	3	0	0	0	0	3	0	3	22
2	2019-03-24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	2019-03-25	2	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0
4	2019-03-26	3	0	0	0	0	0	0	0	0	0	0	0	0	0	5	5

Figure 5.10 – Data classification according to posted dates

We also use it to create a data frame containing the count of each programming language in the track as demonstrated in the figure 5.11.

```
[ ] TRACKS = ['ds', 'web_development', 'devops', 'mobile', 'cyber_security']

[ ] # Initialize a python dictionary that will have the count of each prog.lang as 0
init_values = {prog_lang : 0 for prog_lang in list_programming}
init_values

[ ] # Initialize a python dictionary that will have the count of each prog.lang as 0 in each track
programming_languages_counts = { track : init_values.copy() for track in TRACKS }
programming_languages_counts

[ ] # Define a function that will increase the count of the programming languages extracted
# in the `programming_languages_counts` dictionary
def count_prog_lang(row):
    for track in TRACKS:
        if row[track] == 1:
            for lang in row['extracted_words']:
                # Increase the value count of the prog.lang in that track
                programming_languages_counts[track][lang] += 1

# Use pandas.apply to loop through the dataframe
df.apply(count_prog_lang, axis=1)
```

Activate Windows  
Go to Settings to activate Windows.

Figure 5.11 – Data classification according to track

We created a data frame containing the count of each programming language in each location as demonstrated in the figure 5.12.

```
: init_values = {prog_lang : 0 for prog_lang in listprogramming}

programming_languages_counts = { location : init_values.copy() for location in LOCATION }

: def count_prog_lang(row):
    for lang in row['extracted_words']:
        # Increase the value count of the prog.lang in that track
        programming_languages_counts[row.newlocation][lang] += 1
df.apply(count_prog_lang,axis=1)
```

Figure 5.12 – Data classification according to location

### 5.2.5 Forecasting

After realising data preprocessing, data extraction and data classification, it's time to make prediction of programming languages over the next year. In order to make an efficient prediction, we have tested the performance of three different models: LSTM, SARIMA and fbprophet. Along with each model, we have searched for the best parameters and after comparing its results, we choose the best model.

We have implemented LSTM algorithm, starting by importing **tf.keras** which is TensorFlow's high-level API for building and training deep learning models, as shown in the figure 5.13.

```
[ ] # train the model
def build_model(X_train, y_train):
    # prepare data
    # X_train, y_train = train[:,0], train[:,1] # REMOVED
    # define parameters
    verbose, epochs, batch_size = 1, 150, 16
    n_timesteps, n_features, n_outputs = X_train.shape[1], X_train.shape[2], 1 # Output always 1 here and you can't get it from the
    # define model
    model = Sequential()
    model.add(LSTM(128, activation='relu', input_shape=(n_timesteps, n_features), return_sequences=True))
    model.add(LSTM(64, activation='relu', return_sequences=True))
    model.add(LSTM(32, activation='relu'))
    model.add(Dense(16, activation='relu'))
    model.add(Dense(n_outputs))
    model.compile(loss='mse', optimizer='adam')
    # fit network
    model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, verbose=verbose)
    return model

[ ] model = build_model(X_train, y_train)
```

Figure 5.13 – LSTM algorithm

After running the model, we have tested its results by implementing **root-mean-square error** that consists in measuring the differences between the predicted values and the values observed.

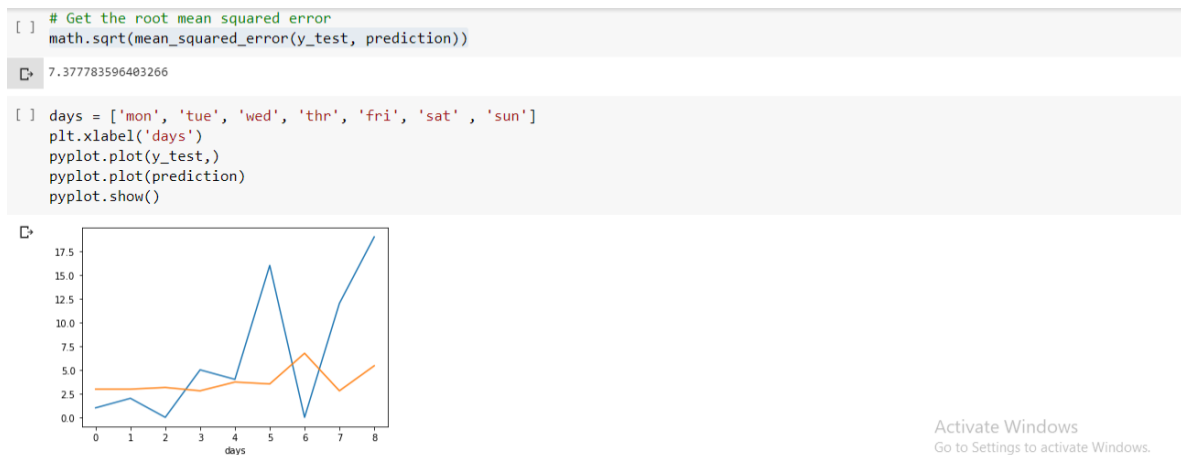


Figure 5.14 – LSTM results

As the result was not good enough because LSTM requires larger data size. We tried SARIMA model, We have started by create a function that returns the best parameters as shown in figure 5.15.

```
[ ] def find_best_sarima(train, eval_metric):

    p = d = q = range(0,2)
    pdq = list(itertools.product(p, d,q))
    seasonal_pdq = [(x[0], x[1],x[2], 12) for x in list(itertools.product(p, d, q))]
    counter = 0
    myDict = {}
    for param in pdq:
        for param_seasonal in seasonal_pdq:
            try:
                counter += 1
                mod = sm.tsa.statespace.SARIMAX(train,order=param,enforce_stationarity=False,enforce_invertibility=False)
                results = mod.fit()
                myDict[counter] = [results.aic,results.bic, param_seasonal]
            except:
                raise Exception
                continue
    print(myDict)

    dict_to_df = pd.DataFrame.from_dict(myDict,orient='index')

    if eval_metric == 'aic':
        best_run = dict_to_df[dict_to_df[0] == dict_to_df[0].min()].index.values
        best_run = best_run[0]
    elif eval_metric == 'bic':
        best_run = dict_to_df[dict_to_df[1] == dict_to_df[1].min()].index.values
        best_run = best_run[0]
    print(myDict[best_run])
```

Figure 5.15 – SARIMA model parameters

Then, we applied the function `predict()` and we visualize the results as shown in the figure 5.16.



Figure 5.16 – SARIMA results

The results were not good because it needs larger data size. So we implemented Fbprophet model and visualize the results.

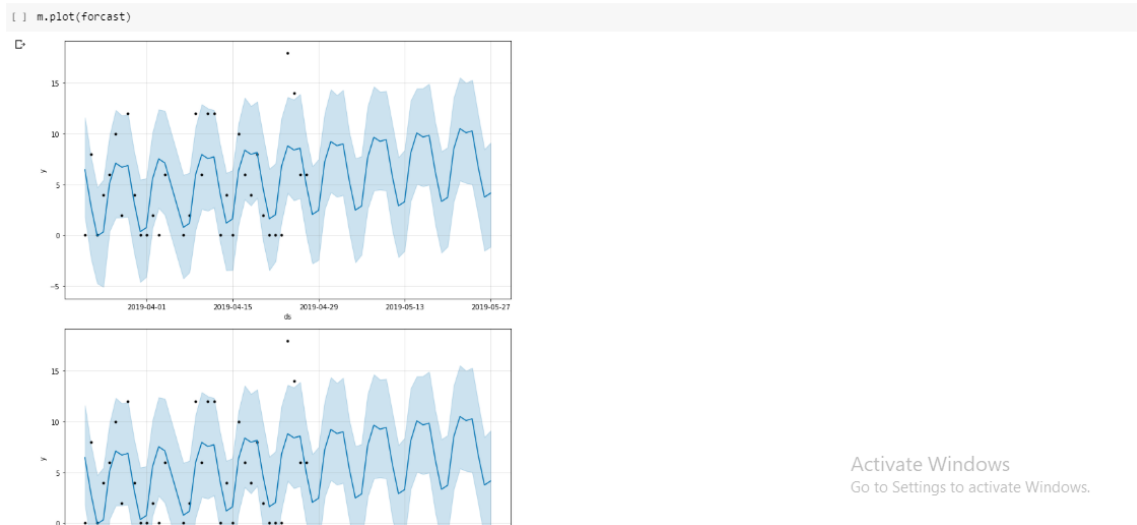


Figure 5.17 – Fbprophet model

Fbprophet model gives the best performance thus we opt for it.

### 5.2.6 Data Visualisation

In order to visualize the data previously prepared, we have built an interactive dashboard using Dash. We started by installing the packages in the figure 5.18.

```
C:\WINDOWS\system32>pip install dash0.23.1    # The core dash backend
C:\WINDOWS\system32>pip install dash-renderer0.13.0 # The dash front-end
C:\WINDOWS\system32>pip install dash-html-components # HTML components
C:\WINDOWS\system32>pip install dash-core-components # Supercharged components
C:\WINDOWS\system32>pip install plotly        # Plotly graphing library
```

Figure 5.18 – Dash setup

After setting up Dash, we can start creating our application which is composed of two parts. First, we have created the app layout which presents how the dashboard will look like. Then, we have defined the callbacks which describes the interactivity of the dashboard.

#### Dashboard layout:

In this step, we have created the dashboard layout with the dash-html-components and the dash-core-components.

```

import dash
import dash_table
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output
import plotly.graph_objs as go
from datetime import date, timedelta, datetime

```

Figure 5.19 – Dash packages

We have built a bar chart, pie chart, time series line chart and a map to visualize the data's evolution by different tracks and locations.

```

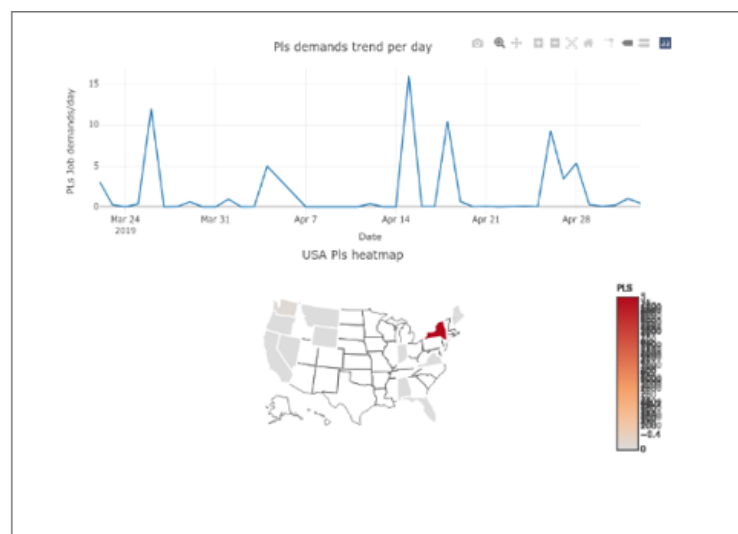
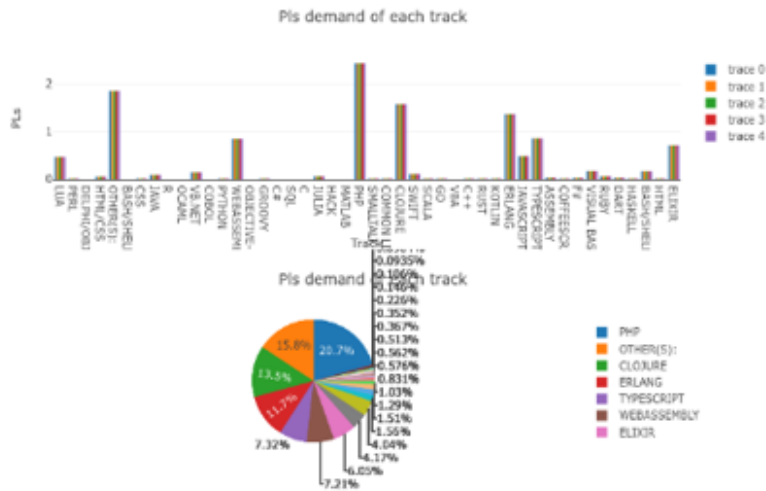
def plot_bar_3(state=None, lang=None):
    # df changes
    logging.debug('Updating bar, dataframe shape:', df.shape)
    return dcc.Graph(
        id='pl_count_bar_3',
        figure={
            'data': [
                {
                    'x': list(list_pls),
                    'y': df.groupby('tracks').mean().sum(),
                    'text': TRACKS[track],
                    'type': 'bar',
                } for track in TRACKS
            ],
            'layout': go.Layout(
                title='Pls demand of each track',
                xaxis={'title': 'Track'},
                yaxis={'title': 'PLs'},
                margin={'l': 100, 'b': 80, 't': 80, 'r': 100},
                clickmode='event+select',
                hovermode='closest',
                dragmode='select',
                height=300
            )
        }
    )

```

Figure 5.20 – Bar chart layout

The interface of the dashboard using only the layout is represented by the figures 5.21 and 5.22.





## The dashboard interactivity

In this step, we have created the callbacks using `app.callback` decorator which includes the inputs and outputs of every chart so that every time we change the inputs automatically the outputs will change and display on charts.

```
@app.callback(  
    Output('pl_map_div', 'children'),  
    [Input('pl_count_bar', 'clickData')])  
def update_map(clickData_count_bar):  
    trigger = get_triggered_input_from_context(callback_context)  
    return update_graph(trigger, None, clickData_count_bar, plot_map)
```

Figure 5.23 – Bar chart callback decorator

## Conclusion

In this final chapter, we presented our working environment. Then, we dived into the implementation of our solution.

# Conclusion

Along this project, we introduced the different steps to build an interactive dashboard for Think-It employees. This dashboard aims at highlighting the trends in programming languages demands, extracted from IT job offers coming from US.

This project began by presenting the host company, Think-It, and giving the project's main milestones. Then, we focused on giving a study of existing solutions and defining the new concepts encountered in our project. After that, we devoted a big interest in specifying the different use cases, as well as developing the design of "Think'n'Predict" since was the most important phase in the project, and we gave the adopted architecture for the dashboard. Finally, we built the dashboard and we demonstrated every step.

In the achievement of the design and development project, we encountered many issues and challenges mainly in relation with the used technologies and the limited performance of our hardware.

The developed dashboard in this project can be further improved. Actually, we can enrich it with more data coming from several continents and analyse them by different criterion so that all users from all the globe can use the dashboard and be up to date with the evolution of new offers in IT job demands.

# Bibliography

- [Lid01] Elizabeth D. Liddy. Natural Language Processing. Marcel Decker, 2001.
- [Pro ]LazyProgrammer.Deep Learning: Natural Language Processing in Python with Word2Vec.

# Netography

- [1] TIOBE group. June 2001. URL: <https://www.tiobe.com/tiobe-index/>.
- [2] Stackoverflow group. 2019. URL: <https://insights.stackoverflow.com/survey/2019>.
- [3] Wajdi Khattak Thomas Erl Paul Buhler. February 08, 2016. URL: <https://www.informit.com/articles/article.aspx?p=2473128&seqNum=11>.
- [4] . December 14 2019. URL: <https://hackernoon.com/beginners-guide-to-ruby-on-rails-mvc-model-view-controller-pattern-4z19196a>.
- [5] , *author* =.
- [6] Nisansa de Silva. [https://www.researchgate.net/figure/word2vec-neural-network-model-This-is-a-neural-network-model-with-a-single-hidden-layer\\_fig2\\_317399369](https://www.researchgate.net/figure/word2vec-neural-network-model-This-is-a-neural-network-model-with-a-single-hidden-layer_fig2_317399369). June 2017. URL: [https://www.researchgate.net/figure/word2vec-neural-network-model-This-is-a-neural-network-model-with-a-single-hidden-layer\\_fig2\\_317399369](https://www.researchgate.net/figure/word2vec-neural-network-model-This-is-a-neural-network-model-with-a-single-hidden-layer_fig2_317399369).
- [7] Dipanjan Sarkar. 2018. URL: <https://www.kdnuggets.com/2018/04/implementing-deep-learning-methods-feature-engineering-text-data-cbow.html>.