

Bucharest University of Economic Studies

Faculty of Cybernetics, Statistics and Economic Informatics

Economic Informatics Specialization

Online Feedback Classification using Sentiment Analysis

Bachelor thesis

Scientific Coordinator

Conf. univ. dr. BOJA Cătălin-Emilian

Graduate

Gabriel DRUȚĂ

Bucharest 2021

Statement regarding the originality of the content

I hereby declare that the results presented in this paper are entirely the result of my own creation unless reference is made to the results of other authors. I confirm that any material used from other sources (magazines, books, articles, and Internet sites) is clearly referenced in the paper and is indicated in the bibliographic reference list.

**Table of Contents**

[1. Introduction 1](#_Toc80903204)

[2. Theoretical Background – Sentiment analysis 3](#_Toc80903205)

[2.1. Importance and actuality 3](#_Toc80903206)

[2.2. Literature Review 5](#_Toc80903207)

[2.3.1. Machine Learning Approaches 8](#_Toc80903208)

[2.3.2 Lexicon-based approaches 10](#_Toc80903209)

[3. Technology Stack 12](#_Toc80903210)

[3.1. Back-end Technologies 12](#_Toc80903211)

[3.1.1. Java 12](#_Toc80903212)

[3.1.2 Spring Framework 13](#_Toc80903213)

[3.1.3 PostgreSQL 14](#_Toc80903214)

[3.1.4 Facebook Graph API 14](#_Toc80903215)

[3.2. Front-end Technologies 15](#_Toc80903216)

[3.2.1. HTML, CSS and JavaScript 15](#_Toc80903217)

[3.2.2 ReactJS 16](#_Toc80903218)

[3.3. Editors and Versioning 16](#_Toc80903219)

[3.3.1. Visual Studio Code 16](#_Toc80903220)

[3.3.2. Eclipse IDE for Developers 17](#_Toc80903221)

[3.3.3. GitHub 17](#_Toc80903222)

[4. Architecture and Design 18](#_Toc80903223)

[5. Implementation details 20](#_Toc80903224)

[6. Conclusions 27](#_Toc80903225)

[Bibliography 29](#_Toc80903226)

# 1. Introduction

Feedback is considered to be the lifeblood of modern business. Whether it is a brand trying to engage with possible customers, a politician trying to reach and influence voters’ decisions, or entertainment studios gauging their audiences’ interest and dedication, being aware of the people’s voices and reactions and responding appropriately is a sure-fire way to succeed in these endeavors. The opinions of users are invaluable, they are the most potent sources of market research, as they literally tell what consumers want or need from a product or a service.

The sky-rocketing progress of modern web technologies and platforms such as forums, micro-blogging, review sites, peer-to-peer networks, and social media apps of all varieties, has given rise to an ever-expanding quantity of information in the form of user-generated content: opinions about product, content, personalities, thoughts about the current state of society, views on numerous religious and political issues.

These voices of the consumer are greatly influential towards shaping the behaviour and the views of other potential consumers; word-of-mouth and organic sharing, either positive or negative, hold a lot of weight when deciding which organization to do business with. The upside is that companies’ marketing departments can pay close attention and be aware of this customer feedback, they can discover which aspects of the business are doing well and where there could possibly be room for improvement. They can use these insights in order to better understand customer’s needs and expectations, and make data-driven decisions about the direction their product or service is going in.

The challenge for companies is tapping into this massive corpus of unstructured text data. No single person, even groups of persons, could manually browse through and navigate the incalculable amounts of comments, reviews, and mentions, in order to make sense of them and generate an actionable plan on the basis of the data. This is where the need arises for automated tools for doing this exact thing: gathering and examining customer feedback.

Social media, and for the purposes of this work, Facebook, is a great exposure tool that organizations, be it multinational companies, entrepreneurs, or even small, local businesses, like mom-and-pop shops can use to connect with their audience, make themselves, their product and their values known to the end consumer, by means such as organic posts, community engagement, and, why not, targeted advertisement.

This thesis paper aims to present a solution which could help the above-mentioned brands, with a relatively significant social media presence, to understand and summarize their audiences’ feedback, by applying sentiment analysis on the comments found on the company’s Facebook Page posts, in order to find out general mood towards them, their product, or their advertising campaigns, and, hopefully, in some cases, to a higher or lower degree, act on it and even influence it.

The end result would be an application which connects to a certain entity’s Facebook business account, navigates through all of its social publications, parses the comments for each post, and, through the use of natural language processing techiques and machine learning algorithms, assigns for each of them a sentiment rating, so that social media managers, even influencers and content creators who want to be on top of their audience engagement can figure out reactions, response rates, and expressed opinions about the subject of their posts, or more generally, about their brand.

The paper is structured in the following way: Chapter 1, the Introduction, has highlighted the motivations for the topic, the importance and actuality of the problem, the need of combining technologies in order to provide such a solution, and its benefits.

The second chapter will dive into the corpus of literature and relevant studies which surround the domain of sentiment analysis, as well as further defining the problem, its advantages, and a comparative analysis between existing approaches and the one chosen for this paper.

The third chapter will describe the technology stack. This includes detailed explanations of the algorithms utilised, the methodologies followed, and an overview of the frameworks and technologies employed.

Chapter four will be dedicated to the presentation of the system architecture and design, outlining the functionalities of the aplication and the main modules, while the last chapter before the conclusion will be detailing the specific elements of the project, the implementation of the classes and the most interesting functions, with accent on the proper use cases. The conclusion consists of an overview of the presented topics, the current limitations, and possible improvements in the near future, as well as for the long term.

# 2. Theoretical Background – Sentiment analysis

## 2.1. Importance and actuality

Social media networks have become an inseparable part of modern human life. Posts and comments play an essential role in the transmission and expression of information, there we find people’s opinions about the latest subjects, technologies, music, politics etc. These short texts posted on these platforms have gained importance through their simplicity and efficiency in influencing the masses. Sentiment analysis proves to be of great utility in monitoring social media networks, because they allow a birds’ eye view over public opinion behind various matters. Social media monitoring instruments which use algorithms and models as they will be described in this paper make the real time process of summarizing these texts and extracting a sentiment out of them faster and easier than ever before. Organizations have become aware of the applications and services which provide sentiment analysis in areas they operate in, and, by fueling them with their own data, their decision making process and business strategies can improve greatly.

Sentiment analysis, or opinion mining, is the domain of study that analyses people’s evaluations, sentiments, emotions and attitudes toward entities, for instance organizations, products, services, individuals, events or topics. It uses natural language processing (NLP) and classification techniques and mainly focuses on detecting the polarity of texts, differentiating between positive and negative opinions, sometimes including the neutral case, although at higher levels, the implementations can extend to classification of specific emotional states, subjectivity analysis, sarcasm etc., and thus it has a wide array of applications in almost every domain [1].

In the context of the sharp recent technological advancements of social platforms such as social networks, and with Facebook having well over 2.8 billion users active each month[13], it is no wonder that companies willing to influence and sell their product have seized the vast amount of opportunity and attention available to them. Because of the billions of pieces of content available to the consumer throughout the internet, they start to rely on word-of-mouth, or the opinions of others, in order to decide where their focus, attention, and ultimately money, go. In this way, online opinion and goodwill have more and more impact, and thus marketing companies start to appear, offering services such as social media monitoring applications, which promise to help brands earn and keep customers’ loyalty and engagement through careful monitoring of mentions, understanding online chatter, and identifying relevant content for taking appropriate action when needed. For these reasons, businesses are turning to the vast domain of opinion mining and sentiment analysis, and slowly the whole industry will adopt these invaluable techniques.

One great example of a succesful application for social media monitoring is Mediatoolkit. It offers real time brand tracking, monitoring channels like Twitter, Instagram, Facebook Pages, and even blogs and forums. It can find all tweets, posts and articles mentioning a respective brand, irrespective of language or region. It simplifies public relations and activities, and some users have gone even as far to say as it’s like having an extra person at the office[18]. An interesting feature it offers, however, is automatic sentiment analysis for various social networks, creating deep reports and analytics.

MonkeyLearn is a general text analysis platform, designed as SaaS, which offers the possibility to users to train their own custom machine learning algorithms in order to narrow down on what is required, for example keywords, intent, and also sentiment. They help support teams with automation of ticket tagging, offer easy API integration for developers, and offer product management services from product review analysis to survers and insights from converstions with customers.

BrandWatch is another online software product which claims to find meaning in the millions of conversations taking place on the Internet. Among brand management, it even ofers services like competition analysis, market research, and trend analysis. It features tutorial on how te get its clients started with their sentiment analysis tool, in order to figure out the relative significance and importance of a sppecific event or brand. It claims that in the future, sentiment analysis must move beyond the one-dimensional scale, if only because there are some many factors influencing opinion that a simple positvie or negative seems short-sighted, similarly to how a political orientation system would work, rather than only left or only right, we add dimensions that will offset unreliability at a granular level.

While these above-mention applications and services are great at their intended purpose, my proposed solution is to monitor single brands at a time, based on the feedback that their pages get on the social media, in order for their public relations master or their advertising team to get involved directly with their clients, at the personal level, because all of us want to feel important and listened to, and there is no other way of achieving this than having a brand respond directly to your concerns. Along large scale analytics such as numbers of shares and likes, having the possibility to detect early signs of controversy, to „put out the fires” on your own pages could show a level of care and attention that consumers may not be used to yet, increasing the likelihood of a positive brand experience and a favourable word-of-mouth.

## 2.2. Literature Review

One of the first researches in the field of sentiment analysis at the document level bring forward Turney[2], which attempted to tackle the problem by using an unsupervised classification algorithm, which did not require any training, in order to classify various reviews as either *recommended (thumbs up)* / *not recommended (thumbs down)* and worked by predicting the median semantic orientation of the pairs of adjectives and adverbs contained in the review. A text was to be classified as positive, recommended, if the average semantic orientation of the phrases, the difference between the information contained in the two-words phrases and a reference negative word like “poor” was higher than the difference between the information from the phrase and a reference positive word such as “excellent”, and if not, the text was to be classified as negative. This method, called Pointwise Mutual Information and Information Retrieval, which measures the similarity of certain pairs of phrases or words, ended up achieving moderately satisfactory results, ranging from 66 to 84 percent, but which, as we will find out, would be improved due to progress in hardware and processing power, or by combining the method of semantic orientation with different features, in a supervised algorithm of classification.

Pang and Lee[3] are another couple of prolific researchers in this domain. They have also endeavoured to classify documents not by topic, but by the overall sentiment transmitted, e.g determining if a review has a positive or negative meaning intended. They have used movie reviews from IMdB as data, and while finding that machine learning techniques definitely perform better that human-grade baselines, both in time and complexity, they set out to compare different machine learning methods such as Naïve Bayes, Support Vector Machines, and Maximum Entropy (not detailed on in the present work, but nonetheless prove to be very impactful in the field), and while these methods did not produce results similar in accuracy to their intended purpose, traditional topic-based categorization, they have examined a couple of factors that could be the reason why sentiment classification problems are much more challenging.

In another one of their works, Pang and Lee[12] have set out to determine the polarity of a document on a multiple scale, and thus to improve the fidelity of the results; rather than a review being simply assigned a “thumbs up” or “thumbs down”, they attempted to infer the implied rating of the author of a review, by generalizing the answers on a finer grained scale: possibly three or even four stars. This allows for a more precise understanding of the initial review, by adding classes between the extremes.

Another research direction, an emerging subdomain of opinion mining through semantic and syntactic features is identifying the degree of subjectivity/objectivity of a text, or in better words, identifying a document or a sentence as fact or subjective opinion. This problem can often be much trickier than the aforementioned polarity classification: subjectivity analysis focuses on finding out if a language unit (phrase, document, word) implies a personal state, attitude or opinion. This subjectivity may always vary from text to text, be it because of the context in which words appear, the statements before and after them; Therefore, as Su[6] has skillfully mentioned, it is trying to automatically detect the subjectivity of the senses of the word, rather than if the word itself is subjective, because different words in different context may have a varying degree of subjectivity (e.g positive charge of a proton vs positive first impression). It also can encounter the same problem as in sentiment analysis, which is the dependence on initial human annotators, and their definition of objectivity at the moment of annotation.

Relating back to Pang and Lee[14] on the topic of subjectivity summarization, they have shown that by applying text-categorization techniques on just the the portions of the document that are subjective, essentially removing the objective sentences from the document, before trying to apply the machine learning techniques of polarity classification, can significantly increase performance, or at the very least maintain the same level of accuracy while only retaining about two thirds of the reviews’ actual word count.

A more in detail model of opinion mining is aspect-based sentiment analysis. When determining opinions or views expressed about different characteristics/aspects of a product, service or an entity, it would be useful to improve the granularity of the opinions extracted: In most cases, a review for a hotel, or a piece of tech, a gadget, contains multiple points that the reader wants potential buyers to consider. We call these points aspects. The aspect means an attribute of the thing in question (e.g. a hotel’s staff, the proximity to high interest points, the price etc.). In order to achieve this, the deep learning algorithms described by Do[15] imply a few sub-tasks: identification of the features described, and only then determining if the opinion about them is positive, negative or neutral, all of it without the requirement of high-level feature engineering .

Apart from software methods used in the domain of opinion mining, it is also necessary the intervention of the human factor, in the condition that automated systems are not up to the challenge of analyzing the tendencies of an individual commentator, sometimes classifying them incorrectly based on the expressed sentiment. Generally, various factors such as country of origin, domain context, previous knowledge of slang, immersion in the specific culture, nuances of linguistics, can contribute to disagreement between levels of emotional or sentimental perceptions. The accuracy of such a system of analysis refers, mainly, on how well the classifications of the system match with the classifications of the human factor. According to the research though, human evaluators only agree in 80% of the situations, so the accuracy for an automated system cannot realistically get any higher than this value[22]. The fact that most of the times humans don’t even agree about the sentiment expressed concerning a given text, illustrates clearly how difficult it is for a computer this task of correctly classifying opinions. The shorter the text fragment is, the more difficult the task is; this is why it is imperative that a great corpus of text is analyzed.

**2.3 Methodologies**

The multitude of structured and unstructured data stored online represents has caught the attention of researchers in areas such as psychology, sociology, marketing, and various other domains. Sentiment analysis has become a key instrument in social media applications, including political press monitoring during an electoral campaign, gathering user opinions about products and services, and has it has even been demonstrated that changes of sentiment in mass-media correlate with fluctuations on the stock market[9].

**There are different** tools for sentiment analysis, exploring a variety of techniques, and generally, algorithms used for traditional text classification can also be used for this purpose, and are divided into two main groups: Machine Learning based approaches and Lexicon-based approaches.

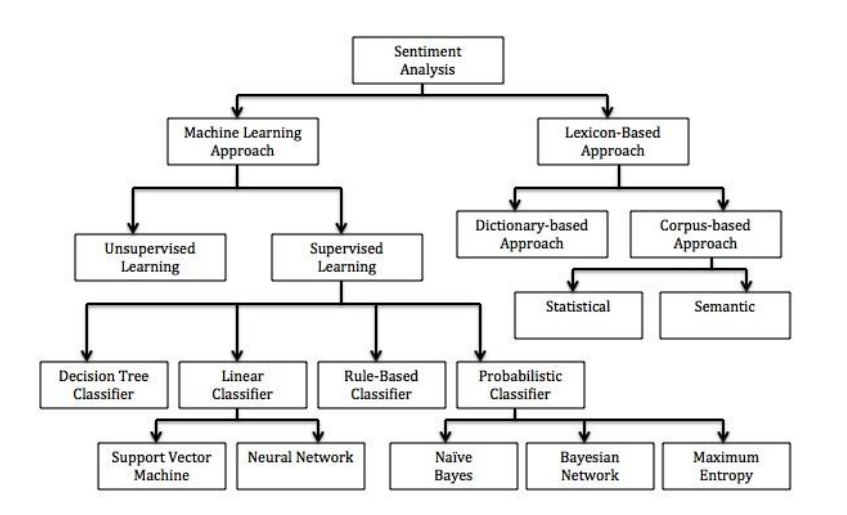


Fig. 2.1 Sentiment Classification Techniques[5]

## 2.3.1. Machine Learning Approaches

Machine learning approaches employ standard ML algorithms in order to resolve sentiment analysis as a text classification problem, where the documents are tagged with one of the three classes: positive, negative or neutral. The ”neutral” class we could do away with in the majority of the methods, as it could be excluded, however, many researchers agree that in the case of some classifiers such as Support Vector Machines or Max Entropy Method could actually gain an increase in the accuracy of the analysis, just by introducing the neutral class[11]. Machine learning techniques can further be divided in two categories: Supervised and Unsupervised Learning.

Unsupervised techniques do not require training data. Consequently, unsupervised methods are useful in cases where large amounts of training data sets are not available or are very hard to find. These are used for the purpose of automatically grouping similar types of data objects into a collection of objects.

Supervised methods are the most frequently used for sentiment analysis. In these, a model is created using a set of training data, with each separate record being manually tagged as belonging to the positive, negative or neutral class. The records are classified depending on syntactic characteristics, linguistics characteristics, or a combination of the two. A classifier is then trained using a one of the standard ML algorithms. After the training process is finished, and the model has been optimised using test data reserved from the data set, using techniques such as 10-fold cross-validation[10], the classifier will be employed to predict the emotional orientation of an unknown text, based on the chosen characteristics.

The key to a higher precision in sentiment analysis or any other text classification problem is selecting a set of useful features. Possible features for sentiment analysis include:

**Opinion rulesets**: There are many examples of expressions and compounded sentences which could indicate polarity, based on certain rules of composition or knowledges in a particular field. Excepting sentiment-bearing phrases and words, these rules can be used to increase the accuracy of our sentiment classification models.

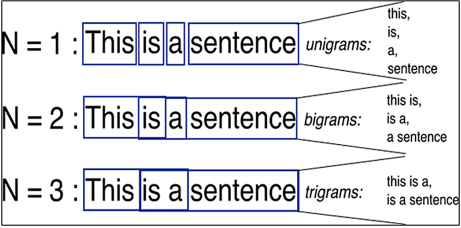
**Terms presence and frequence:** The number of occurences of certain specified terms is one of the commonest characteristics for traditonal text classification problems. These terms may be individual words (unigrams), or multiple words, n-grams, (bigrams and trigrams). The latter have been successfully used by researchers in order to generate results of a higher accuracy. However, in some domains, unigrams can actually perform better than n-grams, for example, in movie review sentiment classification[3]. The frequence of terms may not always prove to be effective in every sentiment analysis research paper, such as in the case of classic text classification, indentification of the subject of a document. In cases like this, term presence could actually be a better feature for sentiment classification, rather than term frequency[3]. Term presence is a binary value attributed to a term an indicates wether or not it is present in the text.

Fig. 2.2 N-Gram example[19]

**Part-of-Speech Tagging**: or POS Tagging of individual words is another characteristic frequently used in sentiment analysis research. Some parts of speech, such as adjectives, are considered to be strong indicators of opinion or subjectivity in a sentence. One of the earliest research works in sentiment analysis has been the identification of the semantic orientation of different adjectives[16] and subsequently finding a substation correspondence between the frequence of adjectives and sentence subjectivity. Nevertheless, adjectives are not the only indicators of subjectivity in a text. Researchers have shown that other parts of speech, such as verbs for example (to love) and nouns (gold) may also demonstrate subjectivity.

**Sentiment Lexicon**: Words that express sentiment or opinion have often been used so as to efficiently identify the polarity of sentences. For example, „amazing”, „brilliant”, „happy” show positive sentiment, while words like „poor”, „bad”, „terrible” indicate negative sentiment. Even if the majority of words that express feelings are adjectives and adverbs, they may also be nouns and verbs, as mentioned above. Outside of simple words, however, there exist expressions and phrases that express opinion on certain things, for instance „cloud nine”. Deconstructed, these words taken separately would not bear any meaning on the document, but as we can see, words, phrases and expressions together form a sentiment lexicon, or an opinion lexicon which can be used to identify polarity. More often than not, just a lexicon like this is not enough to determine the subjectivity of a text, because there may exist sentences without sentiment-expressing words, or some negative phrases in a domain may actually mean the opposite in another context, and vice-versa.

**Sentiment modifiers**: Sentiment modifers may change the polarity of certain words from negative into positive and, of course, the other way around. These modifiers are very important features which should be considered during the training phase of the model. Negation words, for example, (no, non-, un-) are the most common type of these kind of modifiers. In the following instance, „Hannah does not love hang-gliding”, if the negation word would not be attached as a training feature, then it could be wronly identified and labeled as positive because of the presence of the positive sentiment word „love”.

**Emoticons**: Emoticons are symbols used in digital communication, such as instant messaging, forum discussions, emails and micro-blogging, in order to convey human emotion. These emoticons could certainly be used in aiding the model to discover the intention behind the sent message. For example, a smiley face and a frown or an angry emoticon convey two very different emotions, even without using words to express oneself. Emoticons could prove to be a highly effective feature in the identification of sentiment of data in social media sites, reviews and emails.

## 2.3.2 Lexicon-based approaches

Lexicon based approaches are utilised on a great scale in tasks of sentiment analysis. In this approach, expressions of words, phrases, idioms with positive and negative connotations, are used in order to classify expressed views, forming together a lexicon of opinion. It is possible that this lexicon is created manually, although this method is quite time-consuming, not scalable, inefficient and last but not least, prone to errors. That is why there are automated techniques of creating an opinion lexicon, for opinion mining.

These techniques can be split into two categories: Dictionary-based approach and corpus-based approach.

**Dictionary based approach**: This approach creates the lexicon through an iterative process. In the first iteration, it is manually extracted a small set of opinion words with known positive or negative polarity. Then, synonyms and antonyms for the seleted words are searched and collected. This iteration continues until there are no more new words. The selected and searched words are added into the seed list. Eventually, the seed list is manually checked for possible errors. The disadvantage of the approach based on dictionary is that it is not apropriate for finding the polarity of a certain word in a specific domain or context.

**Corpus based approach**: As opposed to the dictionary based approach, the corpus based approach does not face the issue of finding the orientation of the word depending on the context or domain. In this approach, First it is selected a list of opinion words. Then, emerging sintactic models which appear in association with the seed word list are employed in order to determine the orientation of other words. For example, adjectives which appear after the conjuction „and” with a word from the seed list are considered to have the same semantic orientation. Similarly, the linking words „but” and „thought” convey a radical change of sentiment. A great corpus is used to find out if two combined words have the same sentiment orientation. The links between words form a graph. Clustering is applied on the graph in order to create a list of positive meaning words and also a list of negative meaning words. One disadvantage of the corpus based approach is that it requires an extremely large corpus for training, which is difficult to colect, set up and mantain. Consequently, it is possible that it may not be as efficient as the dictionary based approach, if applied exclusively.

# 3. Technology Stack

For designing this web application, I have found it necessary to employ a various and different technologies, to ensure that the development process goes smoothly, without hinders, and that I can build my idea based on the works of others before me, through using their contributions to this information technology field. Finding and choosing the most appropriate tools for the task at hand proved to be no easy feat, due to the availability of options; the development ecosystem is rich with choice, each technology being useful in different contexts, with adantages and disadvantages based on the requirements, the purpose of the application, and the experience of the developer.

With this being said, I will present the technologies that have made possible for me to create this Sentiment Analysis on Facebook Feedback web application: first of all, the programming language which I have chosen for the back end functionalities, both for the server and the data acquisition part, was Java, with the Spring Framework, in the Eclipse IDE. To aid with the sentiment analysis component I have referenced a couple of open source libraries such as SentiStrength and Stanford CoreNLP, and to interface with the Facebook servers, Facebook Graph API and restFB library.

The database I have chosen is PostgreSQL, Git for version control, and for the user interface, the classic front end development technologies: HTML, CSS, JavaScript, and ReactJS. Visual Studio Code allowed me to edit the code in a familiar, user-friendly environment.

## 3.1. Back-end Technologies

### 3.1.1. Java

Java has undoubtedly got to be one of the most ubiquitous high-level programming languages ever conceived. It is object-oriented, class-based, general-purpose, and specially designed with the end goal to have as small a number of implementation dependencies as possible. One very attractive feature is that it allows developers to use the concept WORE (write once, run everywhere); once compiled on a machine, Java code is able to run on all devices and virtual machines which support JVM, without the need for recompiling the code, and regardless of the computer architecture.[25]

Through its innovative character, Java brings together features found in other programming languages. Being object oriented, as opposed to the early procedural languages like C and Pascal, presents certain advantages, notably manipulating object which contain both data (member variables) and code(methods). Java forbids direct memory access, which makes it much more robust than the hybrid, partially object-oriented and partially procedural language C++.

Being higher-level, it is automatically easier to use, eliminating access to memory pointers in favour of references towards objects. It has also been designed from the start for Internet programming, having extended support for networks, sockets, IP addresses, all the way to URLs( Uniform Resource Locators) and HTTP(Hyper Text Transfer Protocol). Java also possesses built-in Multi-Threading support, using a smaller amount of memory and soliciting the microprocessor even less while performing parallel computing.

### 3.1.2 Spring Framework

Java Spring Framework is an open source, enterprise-level framework that is popular for helping developers create stand-alone, production-ready applications running on the JVM(Java Virtual Machine). Spring Framework offers its users a feature called Dependency Injection, objects define dependencies that are later injected into them by the Spring container, allowing the creation of modular applications, which have separately coupled components, being appropriate for microservices, and distibuted applications on the network. Spring also offers support for regular tasks expected from an application, such as type conversion, exception handling, i18n, management of resources and events, and many more. To sum up, developers are equipped with the tools necessary to create cross-platform, loosely-coupled Java applications able to run in every environment.[23]

Spring Boot is an add on that makes it easier to setup and configure Spring applications using capabilities like auto-configuration: applications are initialised with certain pre-set dependencies, automatically configuring the spring framework underneath; opinionated approach means you can can add and configure dependencies based on your needs and the scope of the project. Spring boot initializr is a simple webform where you choose starters without coding.The significant advantage of using Spring Boot over Spring Framework is that for simple projects, it is easier and faster to deploy stand-alone applications that just run independently.[23]

### 3.1.3 PostgreSQL

Databases are a vital persistence storage solution for any application, but epecially a web one like the one presented in this paper, predisposed to crashes, server errors, and even refreshing pages. A database management system, then is a must, but we have to consider the differences between relational and non-relational databases. This being said, my choice of database solution was none other than PostgreSQL. It is a database that is open-source, meaning the source code is available for all interested developers, it is object-relational, it uses a table model and organizes structured data fields into defined columns. It uses and extends almost every SQL language features, it is free and there is abundant documentation available online for it. A priority of this program is extensibility and the meeting of current standards.[24] It also has a very nice and friendly GUI (Graphical User Interface), which allows for intuitive CRUD operations management, as well as Command Line integration, which developers often find to be faster and more convenient.

### 3.1.4 Facebook Graph API

The Facebook Graph Application Program Interface is the main way for applications to access data to and from the Facebook social platform. All of the Facebook’s products and SDKs interact with this API is some type of way. It is named after the „social graph” – the idea of information representation on Facebook, containing edges, nodes and fields. It is based on HTTP, and apps can use is to query informations relating to posts, comments, user profiles, and even post requests for stories, uploading photos, and countless other tasks. It can be utilised like any other HTTP-based language, like cURL, meaning it can be used directly in the browser.[25]

Access Tokens are a way for our applications to access the graph, and everytime an http request is made, one is required, for reasons of security, authorization, accessing a User or Page’s information without the login credentials (password), and they also manage permissions, what type of data the app can access, and on behalf on whose User it executes the request.

**RestFB** is a Java-written, open source under the MIT Licence, flexibile and simple library client for the Facebook Graph API. The main purposes for which it was designed are having no runtime dependencies, meaning there is no need for additional libraries to be included in the project, no dependency clashes, it is portable, working both in Java applications and Android projects. It provides really simple methods to fetch and publish information to and from Facebook, you have the possibility to use the default implementations or customize the components, and it wraps the information retrieved from Facebook servers into Plain Old Java Objects, designed with ease of use in mind.[27] It has definitely aided with the speed of the development of my project, making the management of posts and comments as nodes in the graph quite straight-forward, without implementing many container classes.

## 3.2. Front-end Technologies

### 3.2.1. HTML, CSS and JavaScript

HTML (HyperText Markup Language) is a form presentation level language oriented towards the presentation of text documents on a single page, while employing a specialized rendering software called an User-Agent, the best example of such software being the web browser. HTML provides the means by which the content of a document can be annotated with various types of metadata and render indications. Render directions can range from minor decorations of text, such as specifying that a particular word should be in italics, or that an image must be introduced, down to sophisticated scripts, forms and image maps. Metadata can include information about the title and author of the document, structural information about how the document is divided into different segments, paragraphs, lists, titles, etc. and crucial information which allow the document to be linked to other documents to form hyperlinks (or the web).

CSS stands for Cascading Style Sheets and provides the descriptions and indications of how to style HTML pages, how the elements will be rendered and displayed. We can think of HTML as being the paper and CSS as the paint. The cascading in the name means that a style can be modified even after it was interpreted, in the same class, the priority scheme wll be known and predicted. The design of CSS was made with the purpose in mind to separate the content form the presentation, in order to reduce complexity when multiple web pages share the same styling characteristics and formatting.

We cannot say for either HTML or CSS that they are programming languages, because they are really more like means of styling and structuring web pags. JS (JavaScript), on the other hand, is an object-oriented programming language based around the whole concept of prototypes. It is especially used for introducing functionalities in web pages, Javascript code from these pages being run directly by the browser. The language is well known for its usage in building websites, but it is also used for access to embedded objects in other applications. The most common use of JavaScript is in the scripting of web pages. Web programmers can embed scripts in HTML pages for various activities such as data verification entered by users or the creation of menus and even animated effects. Some of this use cases can be done asynchronously with callbacks, if for example the method is expected to take a lot of time to finish, so it is skipped until the response is complete and ready.

### 3.2.2 ReactJS

React, developed by Facebook, is the most downloaded JavaScript framwork, and according to studies[28], the most likely one to land you a job in 2021. It is a library on top of JavaScript, intended for making it easy to build interactive, faster user interfaces in the frontend. Its building blocks are called components; a component can manage its own internal state, can be encapsulated, and they will render only when needed, when the data changes, they are reactive. Additionally, they can be put together in order to compose the final web page, independent of each other, instead of rewriting the DOM structure for every addition, we can simply import the component we want, or erase, modify them, until the final product takes shape. In this regard, React components are like building bricks, or a puzzle set, each fitting into each other as intended. The logic is written using JavaScript, writing a function which returns an HTML element or creating a class is usually enough to generate a component. Functions accept props (the information that is passed through, accessed through this.props) and are stateless

, while classes can maintain data in an internal state. If the state data is modified, the component will be updated by re-calling the method render().

## 3.3. Editors and Versioning

### 3.3.1. Visual Studio Code

Visual Studio Code is a light-weight, free, open source code editing software developed by Microsoft, available for Windows, Linux, and macOS. It is best used for non-compiling languages, namely the frontend web development trio of HTML, CSS and JavaScript, although it provides support for many other programming languages, like Typescript and Node.js.[28] Apart from writing code, it offers other very sought-after features for development such as IntelliSense technology, giving you suggestions and autocompleting code, which smoothens the process of writing, it can highlight code in different themes to make it more readable, code formatting through a simple shortcut, and last but not least, it interfaces with Git for version control, it has its own terminal and command line. It is an all rounder, and a complete text editor which includes most everything needed for fast-paced development.

### 3.3.2. Eclipse IDE for Developers

The IDE I have found I am the most familiar with and I have the most exprience with was Eclipse, an open-source, multiplatform Integrated Development Environment written in Java and is mostly suitable for Java projects, although through plug-ins it could also help develop applications in other languages.

### 3.3.3. GitHub

GitHub is a free web service which allows software developers to upload their projects and host, manage, and mantain their shared code. It implements the most used version control software Git, actively managed by the one of the biggest development community, which means users can save previous iterations of a project, the user can commit or revert changes to the code, it can go back to previous versions in case something goes wrong or it is not supported anymore, people can work in parallel on the same using code forks and branches, and at the end pull all the code together, allowing for even the smallest improvements to be integrated with ease.

It is perfect for big teams working on big projects, but I have also found it useful to save my progress, see the improvements over time, and not worry about making a small mistake that might ruin the whole setup of the project, because I can always download the working project from my repository.

# 4. Architecture and Design

The conceptual model is a client-server architecture, a distributed application structure, splitting the task between two integral, independent components, with the back end taking care of functionalities like connecting to the online servers for data acquisition, extracting comment text for sentiment analysis and metadata analysis, providing the processed data via a REST API to the front end, and also interfaces with the database for user registration and management, log in and authorization. The back end can be called the server, which the client can access through endpoints using the API, through requests and responses, most of the time using the TCP/IP protocol. The client makes a request asking the server for specific resources, the server receives the request, processes it, and then send a response back to the client.

REST is the acronym for „Representational State Transfer” and it represents an architectural model for creating services on the web. The services which apply this architectural style are called RESTful, and they use all the components which have turned the World Wide Web into a great success. Although REST is not a standard, it uses standards and protocols (i.e. HTTP, URI- Uniform Resource Identifier, XML, HTML, JPG). In practice, REST building a web service using HTTP, XML and URI the way the web was created.

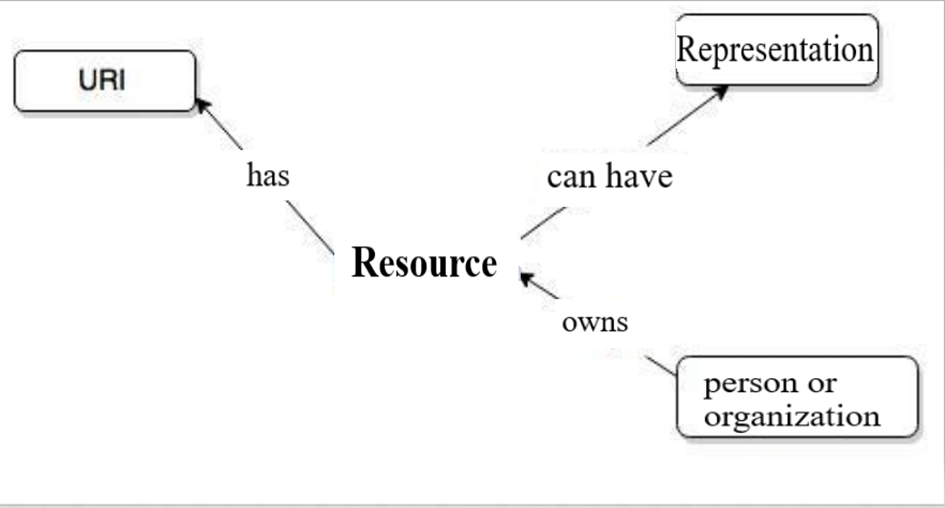


Fig. 4.1 The rest conceptual model diagram

In REST, it is about architecture, not design. REST is a set of rules a real architecture should conform to. Because it is not sufficiently detailed to define a real architecture, we will say that REST is an architectural style. The Web is an real example of an architecture which loosely follows REST regulations.

REST is nothing but another approach to web services. Technically speaking, the basic notions are: resource, URI, representation, and uniform interface. In a REST architecture, anything that can be referred to as an object is a resource, and generally speaking, anything that can be stored in a computer and represented as a stream of bytes is a resource.

The linking of these components in a manner as efficient as possible leads to the creation of REST architecture, and we distinguish two important characteristics: The data upon which the client tells the server to operate is situated in the URI, and the operation which the server executes on the data is directly described in the HTTP method.

In our application structure, the client is represented by the front-end section, built with React.js, and it has two main purposes:

The first task is to send the HTTP requests to the server. At the moment the client initiates an HTTP request, the system is opening up a TCP port to our server that matches the URL. When the connection is created, an HTTP method is sent to our server. The most common methods are: GET method; which asks for data from the server, HEAD; which is the same as GET but is does not receive the response body, POST; it is used to send information to the server; PUT; updates the current state with the one received in the body of the method, DELETE method for deleting the data at the specific URL, and finally, PATCH, which is used to partially update the resource, and, as a confession, in my opinion is the most problematic, I have had customers call at my workplace asking about the PATCH method implementation and why it is not taking every parameter they can imagine. After the request is processed by the server and a response is sent, the connection closes.

The second role of the client is to access the functionalities of the system in order to build the user interface in an intuitive and easy-to-use manner. The main components in this project are: the Log In page, the Sign Up page, the Home page, and the Update profile component. The Login and Register page take care of processing the user information, taking it through forms and passing it along, as requested, to the rest of the application. The Home page is the main feature of the web application, as it contains the analytics for all the posts in the Facebook Page’s feed. In the Update profile component, users can manage their personal info, as well as changing the access token, essentially changing their facebook page profile on which they want the analysis to be made.

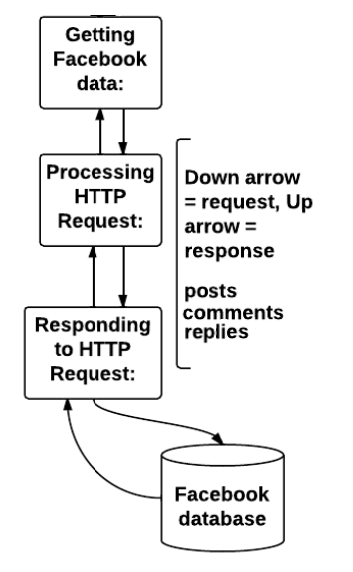
The server is represented by the back-end part, built with Java and Spingboot. The back end, too, has a few important roles. It communicates with the database, and, according to the user requests, it performs several CRUD operations (Create, Read, Update, Delete) on the database management system. It is also responsible for interfacing with the Facebook servers, in order to retrieve the post data and performing the sentiment analysis pipeline operatons on it. The back end contains endpoints, which process the incoming requests. In JavaScript, an endpoint is composed of a route, and two parameters: a request and a response. The request containing data inputted by the client. After processing a user’s HTTP request, a response is sent in return, and since it uses an HTTP protocol for the transfer, it must also send an HTTP status code. The HTTP status codes display the end state of the request. Grouped into 5 categories depending on their characteristics, the status codes can be:

* Informational responses 1XX: the request was received, no new input from the client is required, it is just informing about something, e.g. 100 Continue, 102 Processing
* Successful responses 2XX: the request has succeeded, e.g 200 OK, 201 Created
* Redirection messages 3XX: the resource may be found elsewhere, e.g 301 Moved permanently, 307 Termporary Redirect
* Client Error Responses: 4XX: the request generates an error, e.g 400 Bad Request, 403 Forbidden, 404 Not Found
* Server Error Responses: 5XX: indicate that the error occured at the server level, e.g. 500 Internal Server Error, 504 Gateway Timeout

In addition to status codes, the response from the server may have a message, comprised of a header and a body, usually in the format of JSON or xml. The json format is usually preferred, because it has an easier syntax, it is faster and easier to be parsed rather than xml, and it does not depend on a particular programming language, its data structure is written like an object, very easy to be understood by humans.

With this being said, we must keep in mind that in order for our system to be called RESTful, the cliend and the server must be kept separate, change in one must not affect change in the other, and they must be stateless, they cannot know each other’s state.

In the case of accounts of various entities ran on social media platforms such as facebook, we could consider every comment to a post, every reaction, every share, as feedback to that account; feedback on the quality of the post, the engagement attemps, the event, the new product advertisement campaign. With the amount of data and content generated everyday on Facebook platform, depending on the size of the page proposed to be analyzed, we could expect updates and new data almost in real time.

I have elected not to store the values retrieved from Facebook into the database, because of a very important reason in my opinion: storage. The data is already stored on the Facebook servers anyway at all times, we only need to access them and analyze the texts when needed; at the same time, with new comments being generated possibly every minute, validating all the information all the time, in order to not introduce duplicates seems infeasible from a relational database design perspective.

In summary, we do not need to burden our own database with complexities that could prove to be more trouble than they are worth. The drawback to this approach is that for every refresh of the analysis, the whole facebook server is queried using the Facebook graph API and passed through the sentiment analysis pipeline again, leading to a loss in time performance, it may take a while for the results to be processed again.

Fig. 4.2 Facebook requests

The databases management system is PostgreSQL, and because of what is mentioned above, the only resources the database will manage will be the Users. The table will contain the fields id – integer(10), first\_name – varchar(255), last\_name – varchar(255), email – varchar(255), password – varchar(255), and access\_token– varchar(255).

We will need the persistent storage to remember accounts, and preferences, but also checking the data in the database against the JSON Web Token which will be created at sign up, to implement security matters such as authentication and authorization.

# 5. Implementation details

In this chapter we will take a deeper dive into the functionalities and inner workings of the project, detailing the development steps, and explaining in detail the source code and how the aforementioned technology stack and libraries work together and are implemented.

## 5.1. Connecting to Facebook

Because of changes in the Facebook Privacy Policy, and possibly because of a few other security concerns, Facebook does not allow the parsing and accessing of public pages through its Graph API. At the moment of writing, the only way to access a Page’s data, be it public or private, is through the protocol called OAuth 2.0.

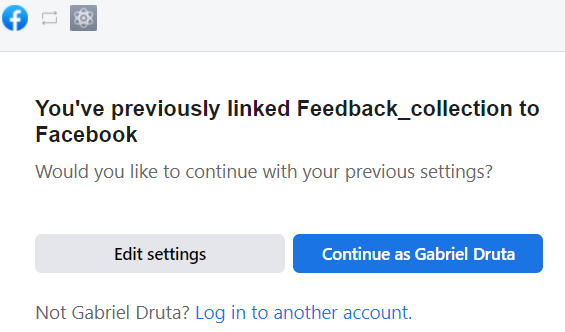
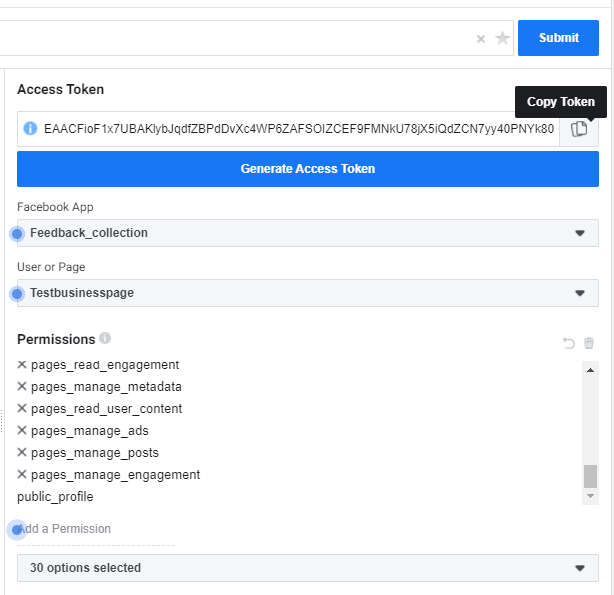
In order to do this, I had to register as a Facebook Developer on my account, create a Facebook App – for testing purposes, I did not need to involve my actual code when I development, although I still have had to go through the App Review process, which meant setting up platform settings according to regulations, I had to submit and upload a privacy policy for my App, and last but not least, I was supposed to explain what permissions my application might ask for, for example „manage\_posts”, offering explanations to possible users about what the app will do with their data and the level of access the offer to it. Finally, all I had to do was connect my account which manages the facebook page to my app, and I was ready to start the authorization process for the Graph API.

Fig. 5.1.1. Facebook app pop up

Next up, we open the Graph Explorer tool, which loads with my app already configured, and the /me node, which allows me to query info like my id and name.

Clicking the Generate Access Token button, will prompt us to give permissions for our name and profile picture, data which is already public, and then we can select the option for a Page Access Token to be created. The token contains information such as the application making the request, the person who makes the request, whether or not the token is still valid, and the scope of permissions, the scope of data which the app is authorized to request and receive. This access token is what will allow the end users of our sentiment analysis system to interface with the Facebook servers.

The connection to the Facebok servers is created, using the authorization token provided by following the steps above. After the connection is successfully established, the function begins to iterate through the feed, and because of the pagination aspect, connections only give a limited amount of post and comments at a time, for example ten, then „loading more posts” is required.

Fig. 5.1.2. Generating the access token

The model class that the requested facebook data from the feed will be stored in is the CommentNode class. It serves as a container for both the Posts and the Comments, being that in its structure it contains a List of itself, a list of CommentNode objects. In this way, comment replies are supported, up to an indefinite nesting level; as deep as we want to go in the replies to the replies, that deep the CommentNode object will be able to store the data, while preserving the hierarchy.

## 5.2. Sentiment Analysis Pipeline

The sentiment classification in this project is consisted of two separate mechanisms: one is a three-class system, namely "Positive", "Negative" and "Neutral", while the other one is rating the sentiment on a negative-positive scale, assigning each comment a score, which could vary from -4(very negative) to 0±1(neutral) all the way up to 4(very positive).

Even if this type of classification of sentiment on the Facebook feedback has a smaller number of classes compared to other text classification problems, it faces much bigger challenges. Some words may have a certain orientation in a context, and a completely different one in another context. Thus, identifying the context of the text would often be crucial in order for the comment to be precisely labeled with positive or negative emotion. This fact makes sentiment analysis quite challenging, because there is no real feasible definitive method of understanding words in context.

Some sentences, such as the ones that are interrogative or affirmative, may contain words which could be intrepreted as containing sentiment value, but in fact do not express it. For example, in the phrase „Is this a good product?” despite the fact that it contains a positive connotation word, „good”, it does not actually say that the product is good, so a lot of care is necessary when dealing with these type of questions. However, not all questions are devoid of positive or negative informations about their object: „If you need a good lawnmower, look no further” this expresses positive sentiment about the specific lawnmower. Thus, it is difficult to identify one sentence as neutral when all of these cases could appear and need to be considered.

One of the most difficult challenges in NLP is identification of sarcastic statements, hot takes or sarcastic comments. There is a considerable chance that in the sentiment analysis of feedback from social media, such as Facebook comments, some challenges appear, which are not found usually during analysis of formal text. Social media often contains jargon, internet slang, abbreviations, word shortenings, imports from other languages, whose meaning can vary and quickly change over time, according to cultural trends. This media is also informal, therefore different words can have different semantic orientation depending of the group that expresses them. For instance „OMG, that attitude was BAD!” or „Jimmy Carr’s sense of humour is ridiculous” convey positive emotions towards both the scene and Jimmy Carr, even though „bad” and „ridiculous” are negative words in themselves, out of context.

With that being said, in the following pages I will be describing the steps required to apply the sentiment analysis on the acquired Facebook data. For redundancy purposes, and to make sure the results will not be biased, I will have used two analysis methods: Stanford Core NLP, an open source toolkit used for fundamental Natural Language Processing which uses deep learning and a computational model over a sentiment treebank, and SentiStrength, a lexicon based, free for academic research library for sentiment analysis in short, informal texts.

The first sentiment analysis procedure is implemented using **Stanford CoreNLP**. Stanford CoreNLP is an open source library for processing and analysing natural language in Java. It is widely used among research withing the NLP community and also within the governmental and commercial users of the open source NLP technology. This is due to the fact that it follows a simple and approachable design, straight-forward and intuitive interfaces, not requiring a lot of associated „baggage”, meaning it is portable and easy to implement in almost any kind of project, and last but not least, because of its inclusion of good quality and robust analytic instruments, aong which we can count part-of-speech tagging(POS), named entity recognition(NER), syntactic parsing, sentence splitting, and sentiment analysis toots, offering also template models for the english language.

Each comment text is inserted into an Annotation object, and after that, a sequence of Annotators put information into an analysis pipeline. The adnotations used in our program are shown in figure 5.1.

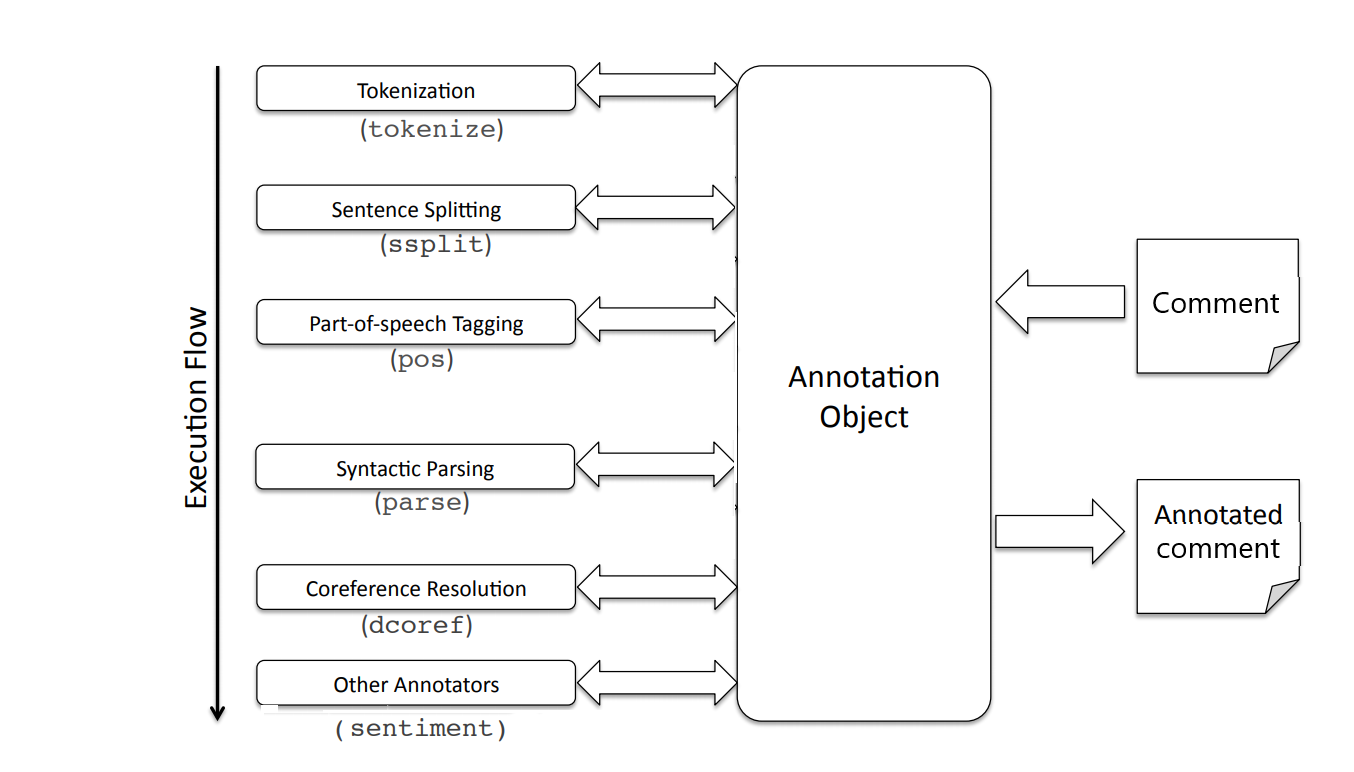


Fig. 5.2.1. General architecture of Stanford CoreNLP sentiment analysis system[20]

In the tokenization stage, the comment is divided into tokens, and can reasonably handle web and noisy text. Then, the sequence of tokens is split into sentences. The parts of speech are tagged, using a Max-Entropy POS tagger, and then lemmatized into base forms. A full syntactic analysis proceeds to take place, including dependency parsing, and finally the annotator object is ready for sentiment analysis, using a recursive deep model over a compositional sentiment treebank[20].

In the program, the Java class SentimentAnalyzer initiates the object with the line *props*.setProperty("annotators", "tokenize, ssplit, parse, sentiment"), and then runs all of the annotators on the Text parameter received as input, Annotation annotation = *pipeline*.process(text); until a result of the type Postive, Negative or Neutral is returned String sentimentType = sentence.get(SentimentCoreAnnotations.SentimentClass.class);

An example: for the comment „*I can appreciate efforts and I love the work that you did with the last campaign!*” the next steps will be done, according to the online tool *corenlp.run:*

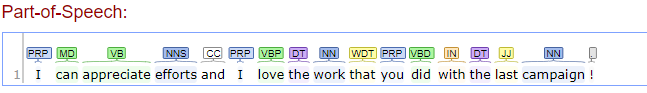


Fig. 5.2.2 Part-of-Speech Tagging on corenlp.run site

Most sentiment prediction algorithms work by taking each word in isolation, giving positive words positive points and negative points for negative words, at the end summing the total and ending up with a sentiment score. In this way, we ignore the order in which words appear, and it is likely that we lose important information.

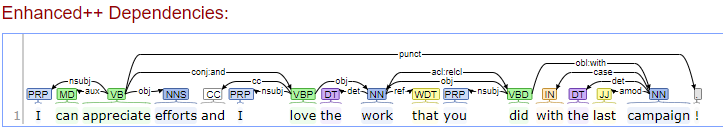


Fig. 5.2.3 Dependency Parsing

In contrast to other approaches, the Stanford CoreNLP deep learning model creates a whole representation based on the structure of a sentence. It calculates the sentiment by learning how the relationship and order between words compose the meaning in long phrases.

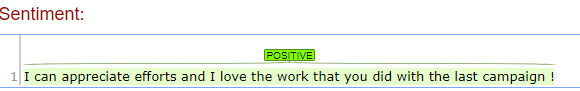


Fig. 5.2.4 Final comment classification

Of course no single model is perfectly accurate, but this one is not as easily fooled as previous implementations, and using sentiment trees along with Recursive Neural Tensor Networks, it manages to outperform baselines of fine-grained sentiment prediction labels.

The second sentiment analysis module I will be employing throughout this project is **SentiStrength.** It is a sentiment strength detection program, developed during the CyberEmotions project[21] to detect the strength of sentiments found and expressed in social media texts on the web. It uses a lexical approach, that makes use of a list of sentiment-related words, and also has sets of rules that deal with standard linguistic and online social methods of expressing emotions, among which we count emoticons, deliberate misspellings, and exagerrated punctuation. It also has the option to be refined for cartain topics and contexts, and can even be formatted for different languages, something which I am seriously considering as a long-term plan, creating a sentiment analysis program with a lexicon database for the Romanian language.

In contrast to the method described above, the neural network method, a lexical method like SentiStrength can have advantages, because it uses language information coupled together with grammatical structure knowledge, such as the role of negations, to approach the problem of classifying social web texts. SentiStrength is structured around the dual positive-negative scales, due to the fact that psychology research has reported that human often can experience both positive and negative emotions simultaneously, and to some extent, independently. It works faily well, without the need for training data, unless we want to fine-tune it for niche applications and domains. Where it does perform less well is in instances where sarcasm and irony is widespread, and in highly-specialized, narrowly-focused topics with rare terms, with unconventional meanings.

I have managed to get a hold of the Java version of SentiStrength by reaching out to the creator/researcher, through an email, and asked for the academic version of the .jar archive license.

The core SentiStrength algorithm is a lexicon of 2310 sentiment words and stems, and also others added during testing, particularly for the evolution of internet speech. The algorithms outputs, for each text, a positive sentiment score and a negative sentiment score. To match this, each word or stem in the dictionary is assigned a value within these ranges. The scores have been initially assigned by humans based on a corpus of comments from the social media site MySpace, and subsequently improved through additional testing and updates.

As opposed to the Stanford CoreNLP model, the lexicon works in a simple way. SentiStrength takes a text as input, it splits the words, separates the punctuation and the emoticons, and then each word is checked against the dictionary for matching sentiment terms. If a match is found, the respective sentiment score is saved. It does not offer features such as grammatical parsing (part of speech tagging) to differentiate between different word senses, which could prove to be a discadvantage down the line, but this is because it is designed to be used on short, informal text from the Internet, and does not rely on grammar being standard for optimal performance.

The algorithm is incorporated in the project through the SentiStrengthCom.jar file. A model class is called, SentiStrength sentiStrength = new SentiStrength();

an array of command line parameters to send is created

String ssthInitialisation[] = {"sentidata", "c:/SentStrength\_Data/", "scale"};

The object is initialized,

sentiStrength.initialise(ssthInitialisation);

and then the algorithm can be used: score=sentiStrength.computeSentimentScores(comment.getMessage());

SentiStrength example analysis of the comment: *“I absolutely hate the newest update, everything is so messy and confusing”*

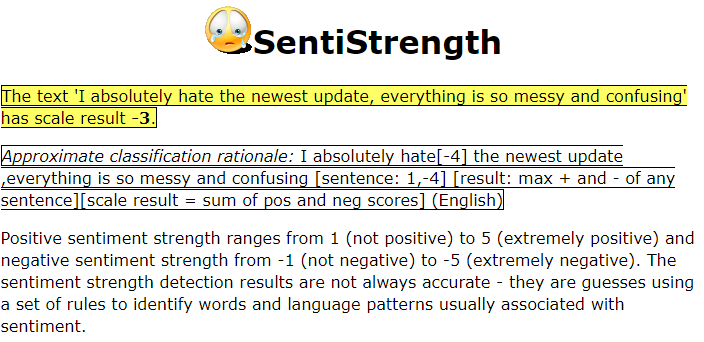


Fig. 5.2.5 SentiStrength sentiment classification example

## 5.3. The back end structure

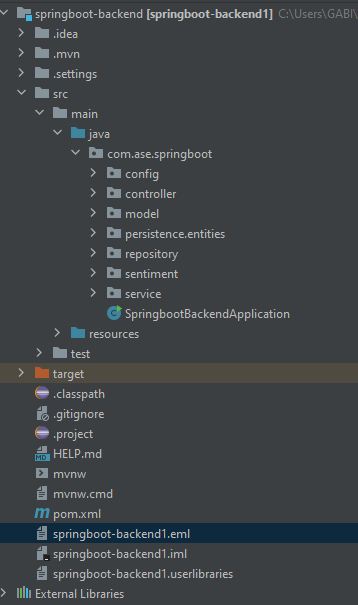


Fig. 5.3.1 Server Structure

Firstly, all the important configuration settings are stored in the pom.xmlfile, which means that they will always be separated from the rest of the project. This way of handling configurations is what sets apart Spring Boot from regular Java implementations, exactly because of the Dependency Injection aspect of the framework, which makes it easier for my application to be deployed in all sorts of environments and it is a much safer way of implementing a server. Here are stored the maven configurations, the PostgreSQL database credentials, and also the setups necessary for implementing the secret token that I will use with the jsonwebtoken dependency.

The next step for a functioning server was to create the PostgreSQL database, and then, connect the user entity, the persistence layer, thorugh to the Data Transfer Object in the dto folder inside the services folder, along with the mapper class, which converts user information to dto and again converts information from dto to entity format.

Now, that the comment system and the database are put in place, we are now able to make different operations on them, all the while giving a potential client, like the frontend, the opportunity to make requests in the backend. These requests are possible if data can be fetched. Controllers are the methods that allow the client to perform various operations on the classes inside the server, be it the database or, in our case, the CommentNode collection.

## 5.4. The Front End Structure

Here is what the React project structure looks like:

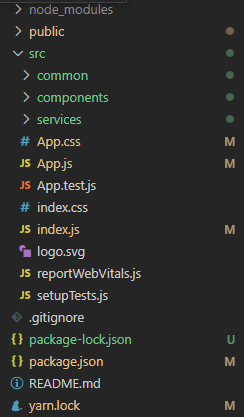


Fig.5.3.1. Front end structure

In the figure we can observe that there are three main folders: node\_modules, public and src. In the node\_modules folder is where we can find all the dependencies necessary. As any react project, being a single-page application, this frontend needs only one html file, meaning that the whole web page is basically rendered from the index.js file in the src folder. The main index.html file that was automatically created can be found in the public folder.

The index.js file is the core of this project. Here, by using React Router, I have implemented three routes: login, register, profile, and home. These four routes lead to the different components that can be found inside the components folder.

We can start with the first two routes, login and register. Upon opening the application, the user is prompted to a login screen. Here, he can opt to log in to the application, or register if he doesn’t yet have an account. He can access the register page even from the top bar menu. When entering the register page, he can actually view the Register component dysplayed on the screen in the shape of a form. Here, after some inputs have been completed correctly with the details about the user, if all the credentials are valid, when pressing the “Sign up” button, the client makes a POST request aimed at the server through the ”register” method inside the auth.service.js file inside the services folder.

After registration, the user is then redirected to the log in page, and to be more exact, the Login component is now rendered, where the user is prompted to enter his credentials, the email and his password, and if they are valid, another POST request is made to the server.

This time though, the response is returning a authentication token that was saved to the local storage of the web browser. With this authentication token, the user will from now on be able to perform operations on itself, for instance updateing the access token, or even deleting his account. At this point, a 403 Forbidden status code means that the identity of the client is known to the server, but the client does not have access rights to the content, meaning it is unauthorized, the server refusing to respond with the requested resource.

After loggin in, we are redirected to the Profile page of the user, where we can see the data like the name, the email, the current Access Token, but here is also where we have the choice of deleting the current account, or updating the information, by being able, through another form, to change details like the name and the email, but most importantly, to update the access token in case the current one has expired, became invalid, or we want to switch to a different Facebook Page altogether.

With the access token from Facebook valid, not expired and correctly inputted in the user data, the home route can now render the whole App component, which is actually, the whole application. The App component is only a functional one and its only objective is to render the Home component. Here, a Home component is shown in which there are rendered a full stack of available Post components with data from the back end, and for each Post, its stack of analyzed comments. When the user clicks on one of the posts from the feed, the specific Comment components are being rendered on the right of the screen.

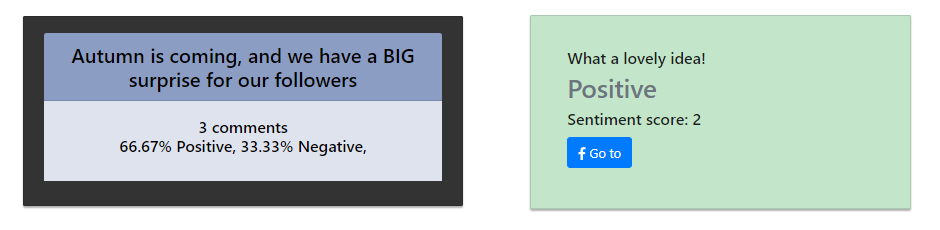


Fig. 5.4. Post and its respective comment

The comments will be displayed in green color if the overall sentiment result is positive, in red if it’s negative, and obviously grey for neutral. The Post component will display the text of the post, the number of comments, and the ratio between positive, negative and neutral comments.

We use the comment id to create a hyper link in order to go directly to the facebook website, and directly to the comment in question, then being able to react, reply, or even delete it if we are so inclined, or if it would help the image of the page.

The Registration Sequence Diagram details the interactions between the user, the server and the database. When the user click on the Sign Up button at the top of the page, the system will display a form, which must be completed with data such as First Name, Last Name, Email, Password, and optionally, a facebook Access Token. The access token is optional at the point of registration, as it can be updated later. If the form is not valid, it must mean that the email was entered in the wrong format, the password did not meet the criteria, or the names were blank. According to every issue, an error message will be shown. If the form is valid, the system will check on the database to see whether or not the user exists already on that email. If it already exists, the log in cannot take place, but if it doesn’t, a new account will be stored in the database with those credentials, and subsequently the user can log in.

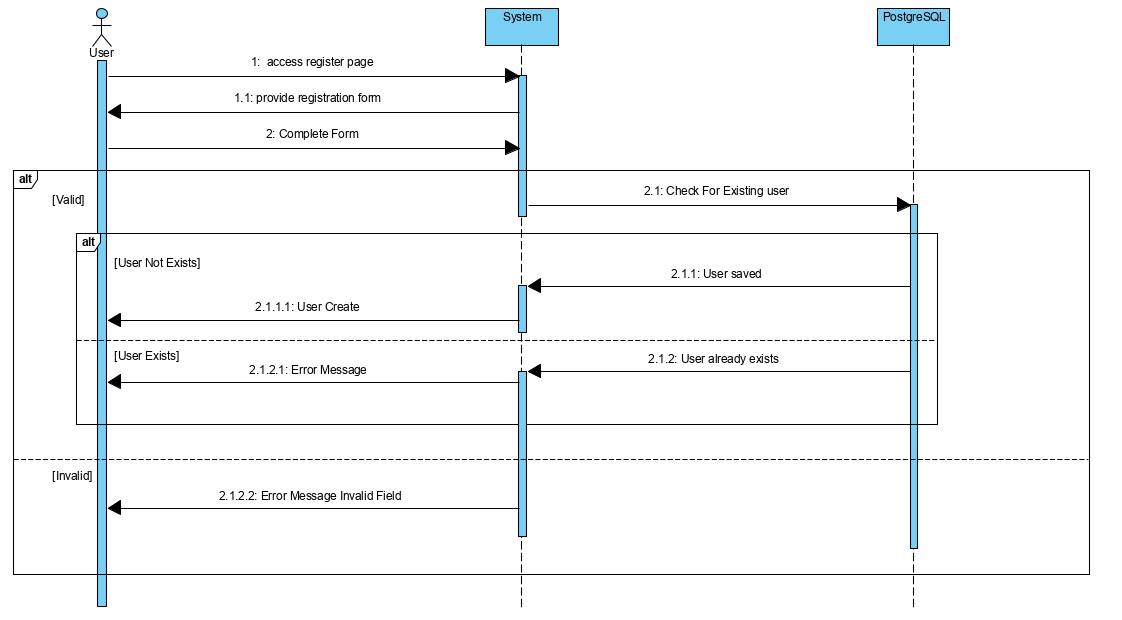


Fig. 5.3 Register Sequence Diagram

# 6. Conclusions

By designing and implementing this project, I feel like a simple, efficient and easy to use solution was obtained for reporting on Facebook comments in real time, using sentiment analysis as a means of feedback classification. I feel that while working on this thesis, my overall knowledge, skills, and confidence in the computer science field have significantly improved. As such, a much better understanding of the machine learning was gained by implementing the sentiment classification algorithms. In addition, my fundamental knowledge base in the domain of natural language processing was acquired.

The final result of this bachelor’s thesis is a web application for online sentiment analysis on users’s comments towards a particular page’s Facebook posts, advertisements, or community engagements. This was done so that people’s emotion and reactions towards the issue can be determined, if the feedback of the people overall is positive, negative or lacking significant subjectivity. The results could be deeply beneficial in several cases, especially in the business and PR domains, where quality of service provided is a significant factor on customer attraction and retention, so if the clients have a positive experience interacting with the page, they are much more likely to keep the relationship and even act as an advocate for the products or services of the respective enterprise.

The application follows the configuration model of client-server. The application was built by using the latest technologies and the most popular ones available on the market at the moment to a student like me, Java Springboot with for the server and React.JS for the client. Through this, I also feel like I have checked JavaScript into my list of fairly-well mastered programming languages. I am now confident that from now on, I will be able to enhance the interactions between the end users and the systems developed, relying on the versatility of JavaScript, the intrinsic values of the objects produced, and the various native visual frameworks available. In the same vein, my previously acquired Java skills were sharpened, by learning more advanced concepts such as web development and security.

Were I to start again the development process from scratch, there are definitely some improvements, some design changes that can be added in order to streamline and empower the application. For starters, I would pick better classification algorithms, more focused on the medium of social networks, or I would even consider building my own sentiment analysis method. This would iron out most of the inaccuracies and inconsistencies that arise from employing two separate algorithms, and may as well increase the precision rates, because, as we have noticed throughout this paper, no single text classification algorithm is truly perfect, or could even have a hope of having a match of more than eighty percent-ish with the classifications made personally by humans.

A second refinement I would consider is the enhancement of the User Experience, through better user interface design and better front-end capabilities, since, admittedly, at the moment, in its current state, the application can’t hold a candle to the experiences provided by the social platforms that the intended end users – entrepreneurs, marketing specialists, PR managers – are probably accustomed to.

Plans for future upgrades to the functionalities of the web service also include multiple social media pages support, meaning that on a single account, a user could browse through and analyze the sentiment on the feeds of several Facebook pages, through separate authentication tokens, and why not, in the long term, the broadening of the network scope, on to platforms like Instagram, Twitter, and even LinkedIn.

All in all, I consider this project, Sentimentalysis, to be a great stepping stone for my future career in software engineering and development, the end result being a sentiment analysis system with data from social media as input, two different approaches for classification, all enclosed in a web application with user management and security capabilities, which has already piqued the interest of acquaintances in the social media content creation field.

# Bibliography

[1] B. Liu, Sentiment Analysis and Opinion Mining, Morgan & Claypool Publishers, 2012, p7-9

[2] P. Turney, "Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised

Classification of Reviews". Proceedings of the Association for Computational Linguistics, 2002 pp. 417–424.

[3] B. Pang; L. Lee, S. Vaithyanathan, "Thumbs up? Sentiment Classification using Machine Learning Techniques". Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). 2002 pp. 79–86.

[4] B. Liu, Sentiment Analysis: mining sentiments, opinions, and emotions, Cambridge University Press, 2nd edition, 2020 (1st edition, 2015)

[5] H. Yousef, Ahmed & Medhat, Walaa & Mohamed, Sentiment Analysis Algorithms and Applications: A Survey. Ain Shams Engineering Journal, 2004

[6] F Su, K. Markert, "From Words to Senses: a Case Study in Subjectivity

Recognition". Proceedings of Coling 2008, Manchester, UK.

[7] Zabin, J. & Jefferies, A., 'Social Media Monitoring and Analysis: Generating Consumer Insights from Online Conversation' , Aberdeen Group Benchmark Report, 2008

[8] Pang, B., & Lee, L., Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval, 1(1–2), 2008, pp. 1-23

[9] Vikram Singh Bisen, “How Sentiment Analysis in Stock Market Used for Right Prediction?” Medium, https://medium.com/vsinghbisen/how-sentiment-analysis-in-stock-market-used-for-right-prediction-5c1bfe64c233

[10] A Gentle Introduction to k-fold Cross-Validation, https://machinelearningmastery.com/k-fold-cross-validation/

[11] M. Koppel; J. Schler, "The Importance of Neutral Examples for Learning Sentiment". Computational Intelligence 22., 2002, pp. 100–109

[12] B. Pang, Bo; L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales". Proceedings of the Association for Computational Linguistics (ACL). 2005, pp. 115– 124.

[13] https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

[14] B. Pang, Bo; L. Lee . "A Sentimental Education: Sentiment Analysis Using Subjectivity

Summarization Based on Minimum Cuts". Proceedings of the Association for Computational Linguistics (ACL). 2004 pp. 271–278.

[15] Hai Ha Do, PWC Prasad, A Maag, A Alsadoon, Deep Learning for Aspect-Based Sentiment Analysis: A Comparative Review, Expert Systems with Applications, Volume 118, 2019, p272-299

[16] V. Hatzivassiloglou, K. McKeown, Predicting the Semantic Orientation of Adjectives, Universidad Nacional de Educación a Distancia, 1997

[17] M. Darwich, Shahrul Azman Mohd Noah, N. Omar, Corpus-Based Techniques for Sentiment Lexicon Generation: A Review, Journal of Digital Information Management, 2019, pp 296-304.

[18] Media Tool Kit [Online]. Available: https://www.mediatoolkit.com/

[19] http://recognize-speech.com/language-model/n-gram-model/comparison

[20] Stanford.edu, “The Stanford CoreNLP Natural Language Processing Toolkit”, [Online]. Available: https://nlp.stanford.edu/pubs/StanfordCoreNlp2014.pdf

[21] CyberEmotions, Collective Emotions in Cyberspace, [Online]. Available: http://www.cyberemotions.eu/

[22] J. Serrano-Guerrero, J. A. Olivas, F. P. Romero, & Herrera-Viedma, Sentiment analysis: A review and comparative analysis of web services. Information Sciences, 2015 pp 18–38.

[23] IBM Cloud Education, Java Spring Boot, [Online]. Available: https://www.ibm.com/cloud/learn/java-spring-boot

[24] PostgreSQL, What is PostgreSQL, [Online]. Available: https://www.postgresql.org/about/

[25] Wikipedia, Java (programming language), [Online]. Available: https://en.wikipedia.org/wiki/Java\_(programming\_language)

[26] Facebook for Developers, Facebook Graph API, [Online]. Available: https://developers.facebook.com/docs/graph-api

[27] RestFB, [Online]. Available: restfb.com

[28] Microsoft, "Docs," Microsoft, [Online]. Available: https://code.visualstudio.com/docs