## Program Description:

Our program uses the twitter API to collect tweets and corresponding data on 5 different cryptocurrencies (Bitcoin, Ripple, Ethereum, Litecoin and Cardano). The tweets we collect will then be analysed by the Google Cloud Natural Language API. This API will return a sentiment value between -1 and 1 to determine if the post is negative, positive or neutral. We will use the sentiment, the number of followers and retweets of each post about a cryptocurrency to give advice whether the currency should be bought or sold. The user will also be able to see the trend of the currency and a market prediction.

## Functionality:

Here are the three services offered on our platform:

1. Users can collect tweets and prices of cryptocurrencies at this exact moment.
2. Collect tweets and prices of cryptocurrencies automatically at predetermined intervals.
3. Analyse a currency of the users choice.

## Approach:

1. The first thing we did after we came up with our program idea was to do a rough draft of our program structure. The goal was to use the Model View Controller structure. The structure is explained more in-depth in the chapter Application Overview Diagram.
2. We created a Database, at first with freemysqlhosting.net, and then with Google Storage. We created six tables: twitterData where we insert and then select all the data we require about tweets, including the sentiment analysis, and five tables for each currency, where we insert the market price and the date.
3. The most important part of the program are the tweets. This is why we started coding on the function to call the Twitter API. The API uses OAuth2 for authentication. This means that we had to create a twitter developers account to get the required access keys. After installing the OAuth2 Python package and setting up a config file we were able to make our first API calls.
4. The next step was to parse the response we got from the API. The response is in .json format. There is a list with 100 items which contain up to two tweets per item. A sample of a single item can be found in the appendix. For our use case we needed to get the username, tweet-Id, post date, tweet content, followers and retweets. This means that the script has to loop through each item in the list and store the corresponding values in a variable.
5. To analyse the sentiment of the tweets we used the Google Cloud Natural Language API. The authorization was done using an environment variable. The explanation for this can be found in the appendix. Each tweet text is sent through the sentiment analysis right after we have collected it from Twitter, and then inserted to the database.
6. The collection of the prices of each cryptocurrency is relatively straight forward. We used the Coinmarketcap Api and call for the price of each currency with a for loop through a list.
7. We created different Data Access Objects to pass the data we get from the APIs to the Database.
8. We then implemented methods to get the prices and values of tweets per day. We use Pandas Library to create tables after selecting the data from MySQL. We use the International Organisation for Standardisation [ISO 8601] as date format and only leave the years, months and days. Thus, we group our Pandas Table by day, and sum the value for the same day on each row. The calculation is described more in depth in the algorithm explanation below.
9. We finished by the View Interface, using TKinter and matplotlib.

## User Interaction:

When our user is interacting with our platform, he first sees a message from our Controller asking him to make a choice between 1) collecting recent tweets and current market values, 2) collecting tweets and market value automatically, 3) Analyze a currency. Here is an in depth description of each functionality:

1. Collecting recent tweets and current market value connects with both the tweetCollection.py and priceCollection.py files. On the one hand, the class tweetCollection is calling the Twitter and Google API to retrieve recent posts about all the currencies, and insert all the required data to the respective row in the MySQL database. On the other hand, the collectPrice function inside priceCollection class is called and connect with the coinmarketcap.com API to retrieve the current price of each currency and insert it to the respective table in the MySQL Database
2. Collecting tweets and market value automatically is similar to the first functionality, except it does not stop after one call, but loop until asked to stop. Prices get collected each hour. Tweets get collected at 8 am and pm. The process runs in the background using the threading package.
3. Analyzing the currency first opens the view interface, where the user is able to choose between the five currencies. Picking a currency shows a regression graph of the trend between the market value and twitter posts. In addition, it gives a prediction of the next day’s market value compared to the current one. Behind the interface, the viewInterface.py calls a TKinter window in appView class which creates the five buttons. Each buttons’ command calls the same regression function, which is formatted according to the specific currency. This function calls several functions: compute\_coef() function to determine the slope and intercept of the regression, getActualPrice() to collect the current market value, getPrice() from priceCollection to get a list of the market value from the past 6 days, getAnalysis() and get\_last\_tweets from tweetCollection to collect a list of the tweets from the past 6 days and the previous day (See Algorithm explanation below). Matplot library is then used to create the regression graph and show it to the user.

## Algorithm explanation:

How do we give advice on selling /buying?

In our calculation, we take into account the market value as well as the tweets from the past week. To assign a value of twitter posts, we sum the number of retweets, followers and sentiments per day, and divide by the number of posts. We assigned a value of 0 to posts which does not have a sentiment value.

Once we have two lists with seven values each, we use the linear regression formula:

Y = b\*1X+ b0

Where:

* Y = market value (dependant variable)
* X = tweets (independant
* b0 = intercept
* b1 = slope

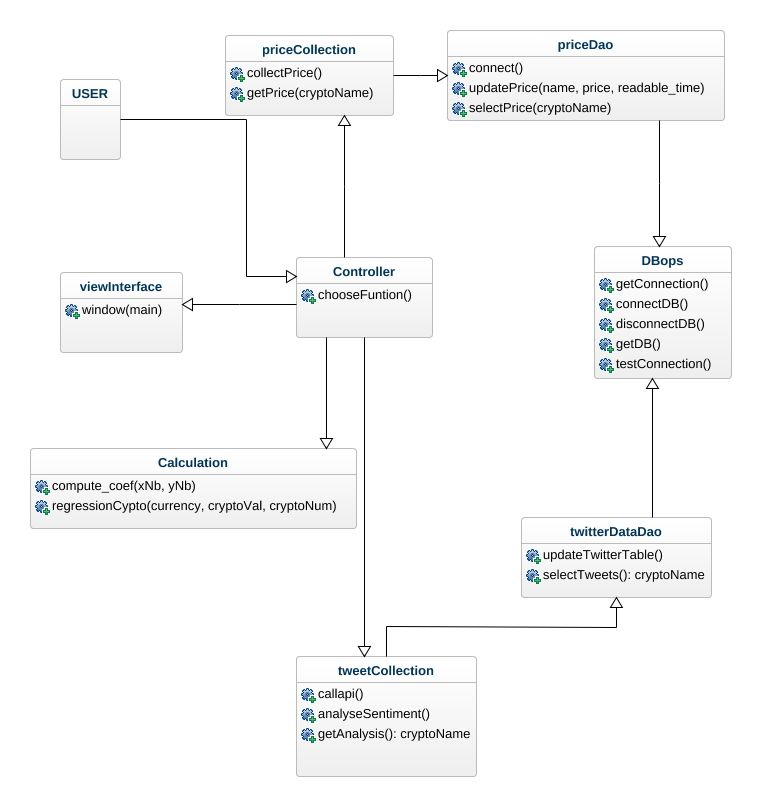
We use the least square method to find the constant b1 and b0:

b1 = (mean(x) \* mean(y) – mean(x \* y)) /( mean (x)^2 – mean( x^2))

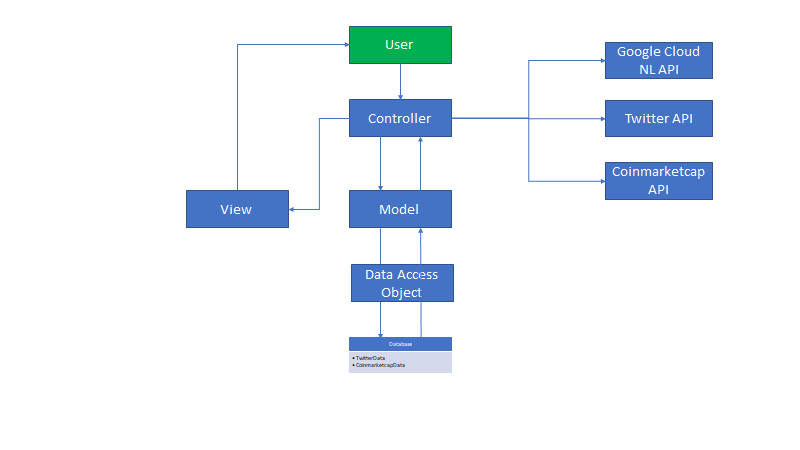
b0 = mean(y) – mean(x) \* b1

With these values, we are able to calculate the regression function, which shows the correlation or trend between the market value and the tweets. Then, by replacing X by the tweets in the past 24 hours, the regression shows us a prediction of the market value for the next day.

## Class Diagram:



## Application Overview Diagram:

Controller: The controller call the three APIs to get the Data which is needed. This Data is then implemented into a Database via the Model. The User who interacts with the controller from the view will be able to choose between the aforementioned functionalities.

View: The View is responsible to show the results. It is able to display a buy or sell recommendation based on the Twitter Data collected. It shows the buttons which calls the controller to get data from the model and passes it again to the view to show the regression graph.

Model: The Model interacts with the Database through a Data Access Object (DAO).

The Database is split up in two main parts: The twitterTable where the information about the tweets is saved as well as the sentiment and a table for each of the currencies is storing the market prices.

## 

## 

## 

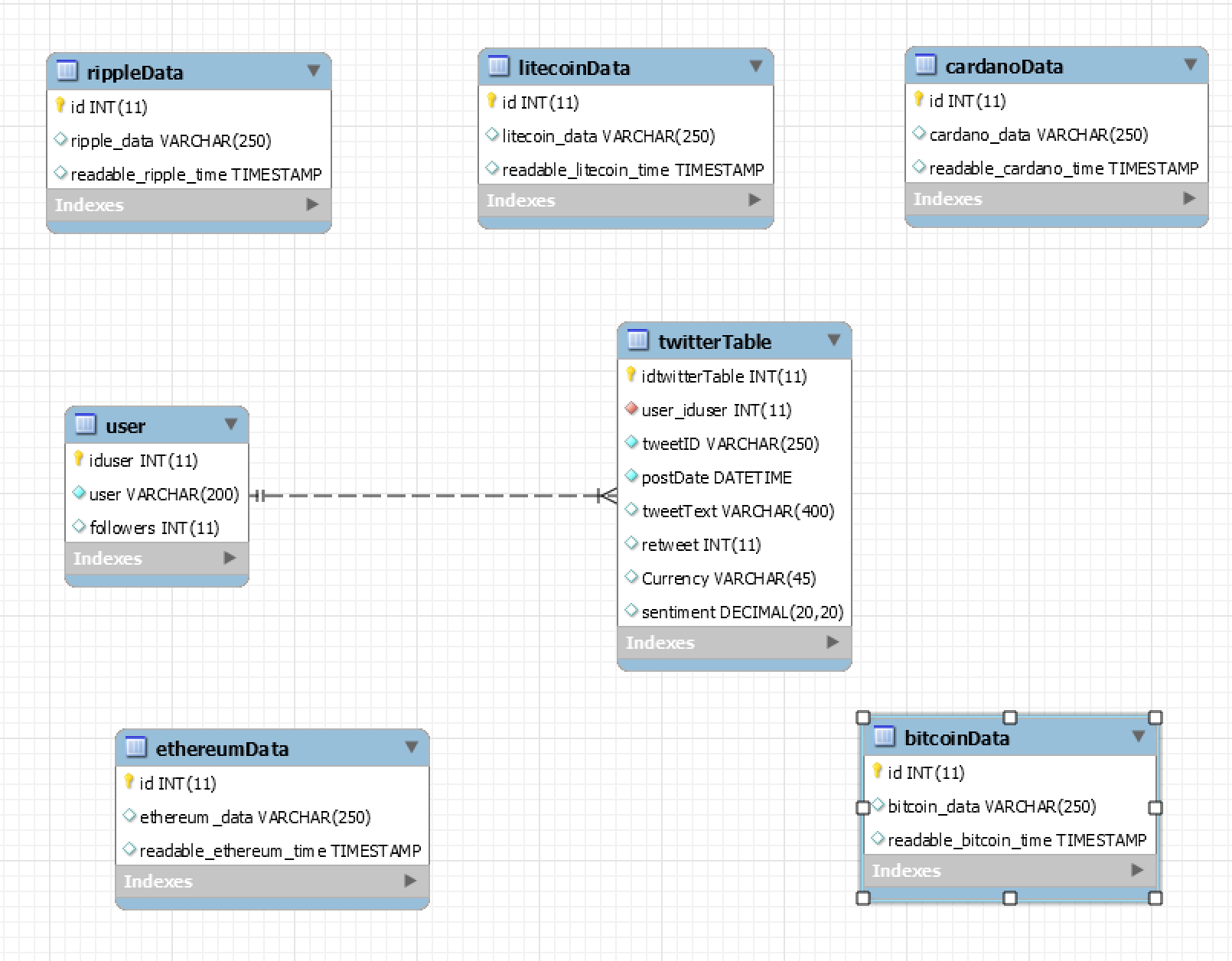
## 

## DB1 Diagram:

This is the diagram of the database we used for our program.

## DB2 Diagram:

We later tried to implement the DB2 which takes the users and followers out of the twitterTable since those are not dependent on the tweet itself. We sadly did not manage to implement this database in our code.



## Problems encountered:

While we were developing our program, we encountered several problems.

1. MySQL database: We first started to use freemysqlhosting.net for our purposes. However, it was limited to 5MB and once we collected about 20’000 tweets, the 5MB were reached, we were blocked from the database and the data was lost. We had to create a new database with Google Storage, which gives us 10GB.
2. Parsing the Twitter json response:
3. The Twitter API as well as the Google Natural Language API limitations restricted us from collecting and analysing each and every posts about the cryptocurrencies.
4. The fact that we were not able to reach as many data as required, our regression analysis shows a very small, if not null correlation between tweets and market value.
5. We were not able to insert Data into our db2 which is a normalized database. The reason for that is, that we did not manage to automatically update the foreign key value with the input statement in python.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Appendix:

Authorization with the Twitter API and our MySQL Database:

Connection to the DB is only possible if the IP is whitelisted. We sadly did not find a different way.

We have created a config.py file with all the information needed to connect to the Database and authorize with the Twitter API. We did this for the following reasons:

* It makes it easier to change to another Database Host, because one doesn’t need to change anything except the config file.
* Multiple users can run personal config files with their respective credentials to connect to the Twitter API
* Sensitive data is not passed to Github, due to a gitignore file which prevents the upload of the config.py file.

Authorization to the Google Cloud Language API:

The Authorization to this API works with an environment variable. The path to the GKey.json file needs to be set as an environment variable.

How to set an environment variable:

1. Search for “Advanced” in Windows.
2. Open the Advanced System settings.
3. Press on “environment variables” in the lower right corner.
4. Set a new system variable with the following specifications: GOOGLE\_APPLICATION\_CREDENTIALS = “Path to GKey.json file”
5. The Google Cloud Language API automatically checks if an environment variable is set. If this is the case and the key is valid the authorization is done.
6. Twitter API keys and Token

myKey = "OoeDEgkX47RFyKcgKGplhg23Q"

mySecret = "4o0jJ8tNmFPlSbkyy8lMDaYazXpIc1f1GF21phvlpy28aPXVgm"

myToken = "989848008694083584-6ZReWbJgEIdqheN8LbO3d9FIlitUNhp"

myTokenSecret = "vLb5aLG4VOoZQBD2A4BCsb2DCZ8QhogR6A8Ly7OjUXxbL"

1. Google Storage and MySQL database access

User='yoop'

password='yoopisthebest'

host='35.205.26.168'

database='db1'

charset='utf8mb4'

In addition to the passwords, some libraries must be installed:

* schedule → to allow a call scheduled
* Pandas → to manage tables and row after selecting from the Database
* request → to request API from URL
* bs4/beautifulSoup → to scrap data from bitinfocharts
* mysql.connector → to connect, select and insert into the database
* numpy → for calculations
* matplotlib → for regression graph
* TKinter → for the window and view interface
* datetime, dateuil, time → to arrange and manage dates
* pytz → to deal with timezone
* json → to format request API
* oauth/ oauth2 → to authenticate to API