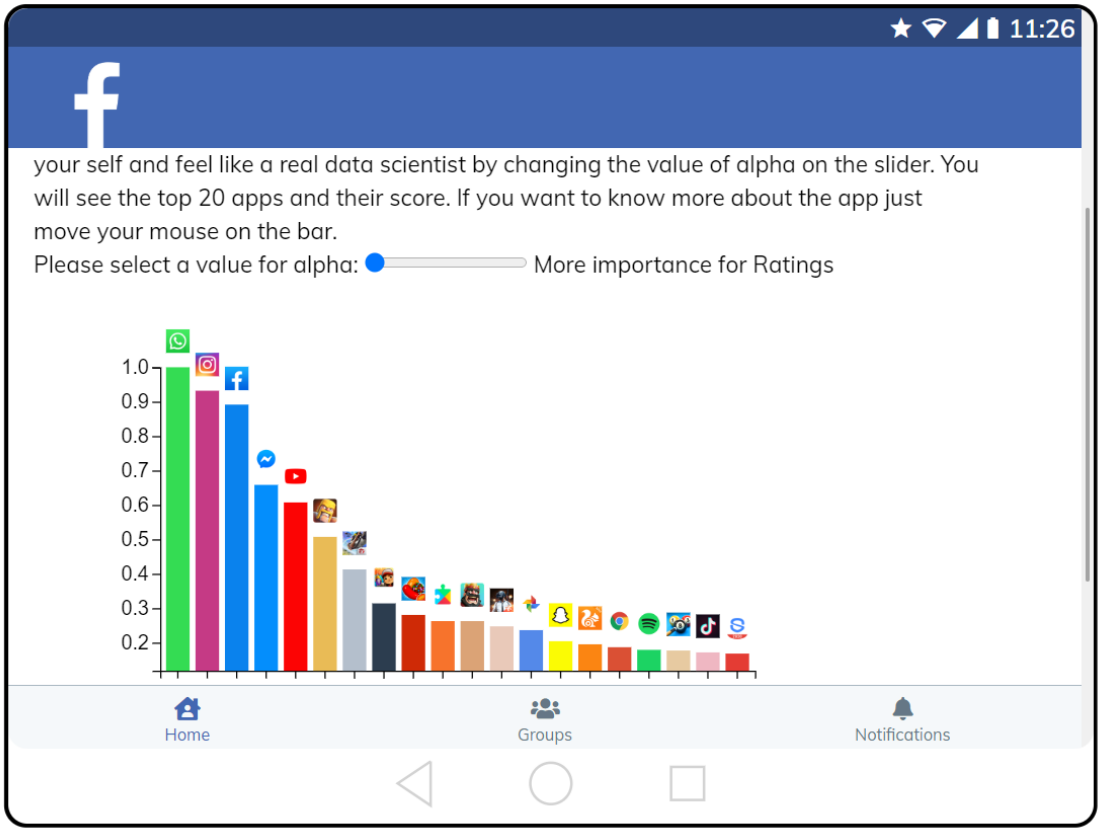
App of Fame

Data Visualization



Process Book

Ahmed Kooli, *Antoine* Schmider *&* *Maël* Wildi

May 2020

1 Overview and motivation

Mobile applications, or simply apps, are part of the daily life of mobile phone users and are thus most likely part of your life as well. They have different purposes and each one is unique. However some of them can be found on almost every smartphone, whereas others seem to forever be dusty apps that no one will download.

We thus asked ourselves the reasons that led to this disparity, and wanted to determine what actually makes an app successful. There are a lot of companies that sell statistics about apps on the market to help developers determine how they could improve theirs. The problem is that most of them target large companies and not a new developer that is about to launch his app. The idea of this project is simply to show similar statistics explaining the features that make an app successful, but in a more attractive and interactive way, and of course for free.

2 Path to the final result

2.1 Data acquisition

At first, we thought we could work with this Google Play Store Apps dataset from Kaggle in which we had information like the number of downloads, the ratings, the price and so on for 10'000 apps. But how did they selected those apps? It wasn't clear. Moreover at the beginning, we also wanted a dataset for the Apple Store in order to compare the two main stores but the only one we found was from 2019 and was also not complete. Thus, we decided that making our data by parsing it out ourselves was safer than trying to compare two datasets that are not from the same period and not well defined. As this would mean more work, we decided in the meantime to focus only on the Google Play Store. We used an API called Android Rank which allowed use to take the 500 most downloaded apps for each of the 48 categories (Communications, Tools...). This API gave us most of the information we had in the Kaggle dataset and even more with the growth of the number of ratings over the last 15/30 days. The only thing that was problematic with this API is that the number of requests was limited, therefore it took a few steps to acquire all the data. However the nice thing about using an API is that the data came out pretty clean. We only had to deal with some type conversion and some relatively small number of NaN values.

The only thing that was missing to this dataset was a real rank that was not simply based on the number of downloads. In fact, this is a good indicator but we thought that the average rating needed to be included as well as it is also deterministic of whether an app is good or not. So, we decided to make our own score to rank the apps. We decided that the information that would determine our score would be the number of installs, the number of reviews and the average rating. However, we had a problem with the number of downloads: it was given by the latest steps the app achieved and was not on a continuous scale so the gaps can sometimes be large. Thus, if we simply set the score to $(Nb\_Installs + Nb\_reviews) * Average\_rating$, the differences between the apps scores would be too large. This is the reason why we decided to take the log of the $Nb\_Installs$. Moreover, we normalized each term to have a score between 0 and 1. We ended up with:

$$score = (\alpha * \frac{log(Nb\_Installs) - min(log(Nb\_Installs))}{max(log(Nb\_Installs)) - min(log(Nb\_Installs))}$$
$$+(1-\alpha) * \frac{Nb\_reviews - min(Nb\_reviews)}{log(Nb\_reviews) - min(Nb\_reviews)}) * Average\_rating$$

This α is a parameter to tune. We selected $\alpha = 0.5$ as it gave accurate results with a smooth graph and a good balance between parameters. Note that this parameter can still be tuned by the user of the website. We then assigned each app to a global rank (between all apps) and a category rank (between apps of the same category).

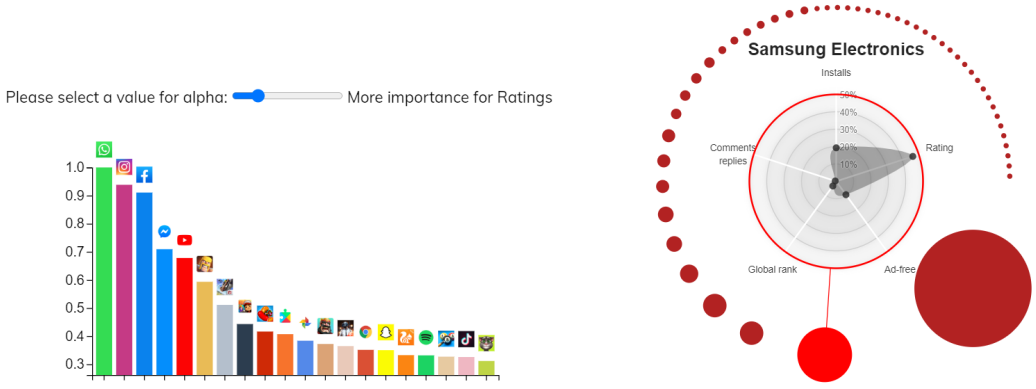
After setting these rankings, we started some basic visualizations on Python to see which information would be relevant. This step of exploring the data was important since it helped us structure

our analysis and know which features to keep. For example, the growth of the number of ratings over the last 15/30 days turned out to be an information that we couldn't really make use of. However at this point, we didn't have enough interesting graphs especially given the fact that we wanted to make at least one graph per app. Our supervisor helped us with that after Milestone 2 by suggesting to work on the comments as well. Hence, we found a new API called google-play-scraper. It completed our previous dataset with information on the apps developers, and on whether an app contained ads or not. But most importantly, we managed to parse the 500 most relevant comments for the 500 best ranked app (according to our score). Thus, we were able to have enough data to make interesting graphs in the app and that were coherent to the app itself.

2.2 Visualizing data

One of the challenges we faced when creating our plan for the visualization was how to display the characteristics of all these Android applications while keeping at the same time a readable graph. The first approach we had was to use interactivity to solve this: for instance in the ranking graph, we let the user change a coefficient that would give more importance to a characteristic than another. This enabled to highlight different kind of apps whilst displaying only a few apps at a time in the visualization.

We also used an entirely different approach, that consisted in displaying more items in the graph, but making the most successful ones bigger so that they are easily recognizable by the user. This was done, among other graphs, in the one dedicated to developers. The developers were represented by a red circle that was proportional to the number of apps in the top-500 that they created. Hence, we did not put them all on an equal footing and reproduced the dominance they face in the Google Play Store. All of this, while letting the user access the information of more modest developers. This method as well as the one detailed in the previous paragraph are illustrated on the Fig. 1.



(a) Letting user change parameters

(b) Making most successful ones eye-catching

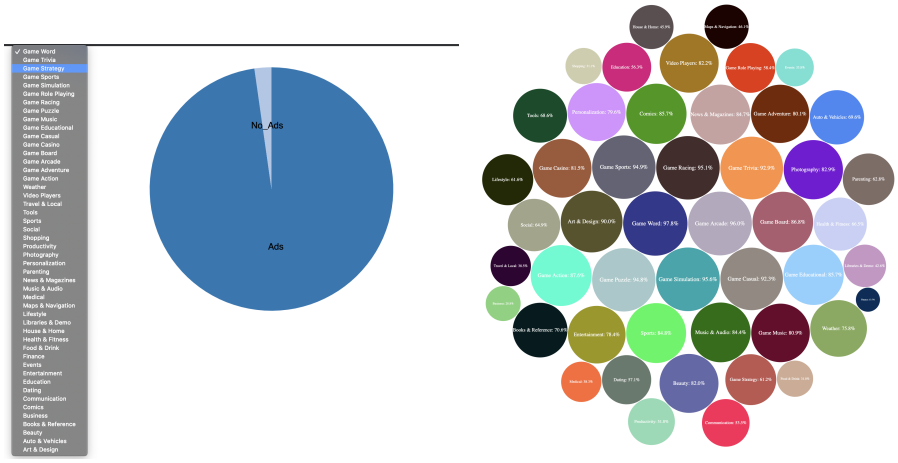
Figure 1: Some methods used to have readable graphs, in spite of the great amount of items to display.

Also, when we needed to display results by categories and then by apps, we used the Treemap approach which allows to keep the importance of each of the categories as well as to zoom in to see details about the apps within a particular category.

To gain space and make the graphs more interactive, we decided to make use of hover effects to add extra information on a specific selected item. We chose hover events instead of click events to ease the task of the user to jump from one app to another. Furthermore, in graphs where the item to hover could be small, like the red circles corresponding to developers in the chart Fig. 1(b), the information disappears only when the user has entered and left the extra information box, or when they select another item. In that way, the user doesn't have to worry about keeping the mouse at the exact right spot to read the extra information.

We also tried to find graphs that suited the most the insight about the data we wanted to display. For instance, rather than displaying a graph with the rates of each apps to show the evolution of the number of downloads, we decided to use a dynamic graph which actually evolves by displaying the number of installs from 2012 to 2020.

Another example of change we made, is for the percentage of apps that contain advertisements in a given category. We first decided to use piecharts to show this, with a drop-down list to select the category. But since this distribution is binary (either ads or no-ads), it was much easier to use a bubble chart with the percentage of ad-free apps: this way we could get rid of the annoying long list by displaying all the bubbles in one chart. Thus, enabling to directly compare the categories. The before-after result is illustrated in the Fig. 2.



(a) Before : piecharts and drop-down list (b) After: unique bubble chart

Figure 2: An example of rethinking of our graphs to make it more pleasant to use.

We also did our possible to implement innovative ways to present the data in order to make it memorable so that the visitor remembers the results displayed. To achieve that, we tried to take into consideration some details: for instance in the first graph shown in our website (Fig. 1(a)), we put the icon of the apps on top of the barplot, with the color of the bar matching the dominant color of the icon. Besides, we tried to keep uniformity between the graphs. For instance, all the categories of apps represented in our visualization have the same colors associated throughout the whole data story.

The designing process of each of our graph was first to discover existing implementations, notably on the D3 Graph Gallery and on Observable. Either we found an existing graph that was almost exactly like the one we were looking for, for example the race chart shown on Fig. 3. In this particular case, we mainly had to adapt our data to fit the format required by the script. Other times, for instance for the developers chart already shown on Fig. 1(b), we created ourselves the outer red structure as well as the kind of magnifying glass that appears on hover, and displayed in the center of it a spider chart showing some specific details that reuses an existing script.

2.3 Link data to apps

One important step is assembling the graphs we have into one final product that has the form of a website. The organization of all the information can be tricky since the user can find difficulties to access the results if it is not done correctly. This is the reason why we wanted the final product to have a very clear display that would make everything accessible without the user having to spend time looking for what he needs. We thus chose to design it as a mobile phone with apps containing the results. The advantage of this type of display is that the analysis can be split into different

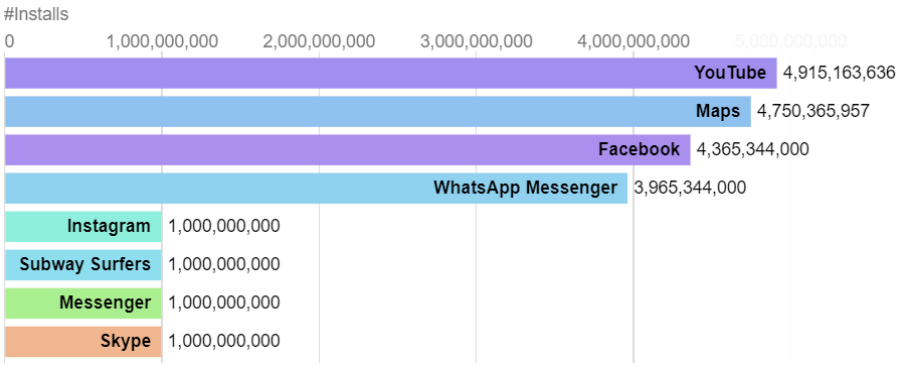


Figure 3: Race chart whose script was available on Observable

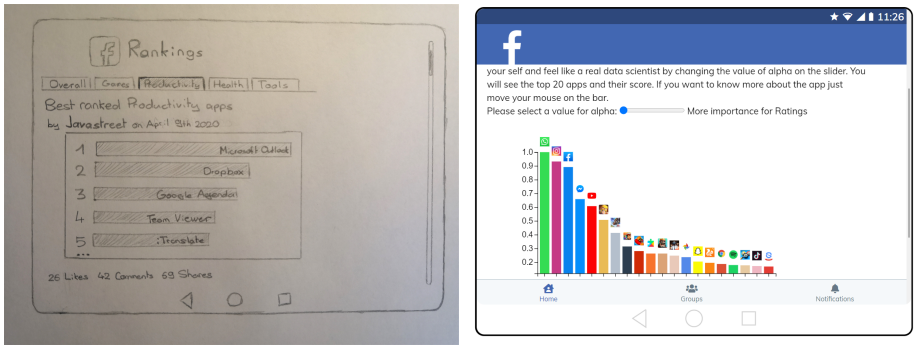
parts, and each part appears in a different app. Each child (app) is then easily accessed from the root (home screen). This will also make the product user-friendly since people who visit the website are most likely smartphone users as well, and will thus have no difficulty understanding the way it works. Moreover, the visualization would also be related to the main topic which is "what makes an Android app successful", which links the analysis and the visualization. On top of that, there are three buttons exactly like in a real Android smartphone. Even though they are part of the visualization aspect, we wanted to make use of them in order to navigate inside the website. As we can see on Fig. 4, the central button gets you back to the home screen where you can choose the app, and the left button gets back to the home page of a specific app (since each app can have a few different pages inside of it).

(a) Before: phone design (b) After: phone design

Figure 4: Design of the phone's home screen.

Let's now talk about the apps. As said before, each one of these apps corresponds to a different category of results (i.e. rankings, reviews...). Then, there's a second layer of depth inside the app where the results are split again into sub-categories (for example: global ranking, proportion of app categories in that ranking...). But the tricky part was to display all these results while giving the illusion to the user that it is a real smartphone app. We tried a few designs at first but the illusion wasn't there because even if everyone knows what an app looks like, it's not that easy to find its key elements. Indeed, when we get used to something, we sometimes can't notice what is obvious, so we had to pay close attention to what famous apps like Facebook, Twitter or YouTube had in common in order to reproduce it in our website. The most common thing is the fixed elements, meaning elements that don't move when scrolling inside the app. Even though it doesn't

seem that important, adding fixed elements really improved the visualization. There are two of them in mostly all our apps: the top banner, and the bottom navigation bar (Fig. 5). The banner contains information related to the phone (time, battery, WiFi) and to the real app we refer to, including for example the logo or the name. On the other hand, the navigation bar is composed of 3 clickable tabs, which also help splitting the analysis into different parts.



(a) Before: app design (b) After: app design

Figure 5: Example of an app design.

At this point, we only had the basic structure of an app, but we wanted to make it more challenging by having a unique design for each one. It was then a matter of looking at the details in the real apps like for instance the format of a Facebook post, a tweet or a WhatsApp conversation. This designing part was actually quite time-consuming since we had to almost start from the beginning for each app design. But at the end, this increased the diversity of the visualization aspect while still having a common thread for all apps like the fixed elements that makes everything consistent.

We also managed to use the features that are specific to each app to display a result. From there, we faced two situations. Sometimes we had a graph for which we tried to find the app with elements that correspond well to it, such as the usage of a YouTube video for an animated graph that changed over time. Other times it was the opposite: we tried to find ways to use an app feature that we found interesting visually, like how the grades were displayed in EPFL Pocket Campus. As shown on Fig. 6, we used this feature by replacing the courses names with the apps developers names, and the grades with the number of apps they developed that appear in the 500 most famous apps. This was also a way of displaying a ranking of a relatively large amount of items (62 developers) without making the visualization cluttered with information.

There are a few other app features that we used that helped improving the visualization by creating interactive elements. Indeed, when we think about a YouTube video for instance, one of the main associated features are the like and dislike buttons. The same goes for a Facebook post or a tweet that can be liked. We thus added these individual elements as much as we could for the apps, again without having too much of them. The interesting part is when the user will try to like a post and suddenly realize that it actually does something. Even if it has no benefit to the analysis, we wanted to have an interactive website, to represent the possible interactions one can have with real apps.

2.4 Data story

Even though making a few separate apps has visual and analytical advantages, it comes with a problem: making a data story that can easily be followed. In other words, when we had all of our 6 apps ready to use, we questioned ourselves whether the user would be able to follow the order of reading we had in mind. Indeed, the data story can lose all its narrative aspect if the reader gets lost while having to choose which app to start with. We also didn't really think of this part for the previous milestones which is why it was important not to rush it and have a clear idea in mind.

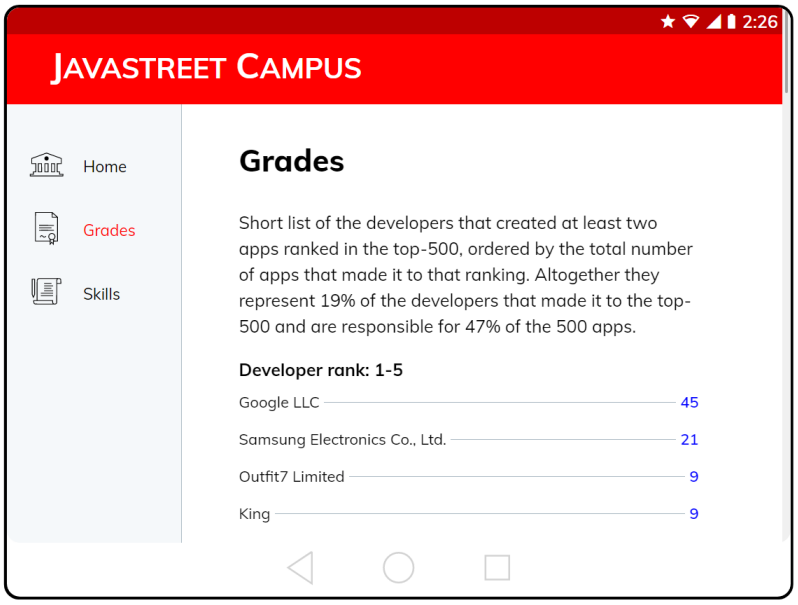


Figure 6: Developers app design, inspired by EPFL Pocket Campus app.

The solution we found was to force a specific order of reading by making the apps available one after the other. The first idea we had was to only display one app at first, and when the reader was done exploring that app and got back to the home screen, an animation would start showing a new app being downloaded. However we didn't want to spend too much time on this aspect in order to focus on more important visualizations, so we found another solution that was as efficient, yet much simpler. All the apps are displayed from the beginning of the story, but only the first one would be accessible. Then we use the same concept as before, meaning that when the reader is done with an app, the next one becomes available. As we can see on Fig. 7, the icons of the non-available apps are in black and white, which instinctively directs the user towards accessible apps with colored icons. This solution has also the advantage of acting as a table of contents by showing all the different results from the very beginning, making the reader aware of what the whole website contains.

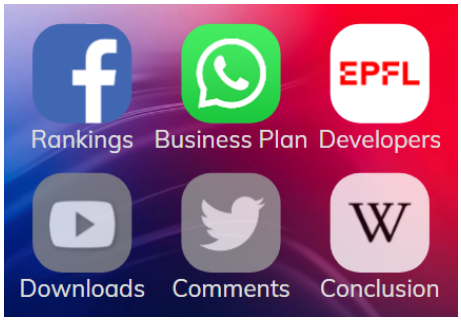


Figure 7: Status of the apps after accessing the Business Plan app and returning to the home screen.

As the saying goes, misfortunes never come singly. This problem (that we just managed to solve) came with another: how to explain all this data story concept made of apps unlocked one after the other to a reader that just started using the website and had no idea of what to expect. Moreover, we didn't want to spoil the phone visualization idea by adding for example an introductory page that isn't related to that theme. We thus decided to make use of a common feature to all smartphones: the lock screen (Fig. 8). It is the first thing that appears when loading the website

and also when opening a real smartphone, and it contains two main elements. The first one is a clickable "notification" that displays on click an introductory message explaining the website content and how it works. The second element is a password that a unique pattern unlocks, the latter being stated in the notification. Basically, in order to access the website, the user has no other choice but to read the whole notification and thus understand beforehand how the website works. This also shows from the beginning that the website has interactive elements, which can encourage the user to find them when exploring the apps. We also thought that the password was a cool visual feature and making use of it as part of the data story was an improvement for our website.

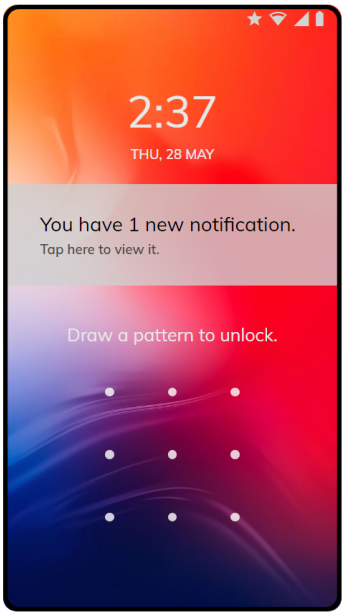
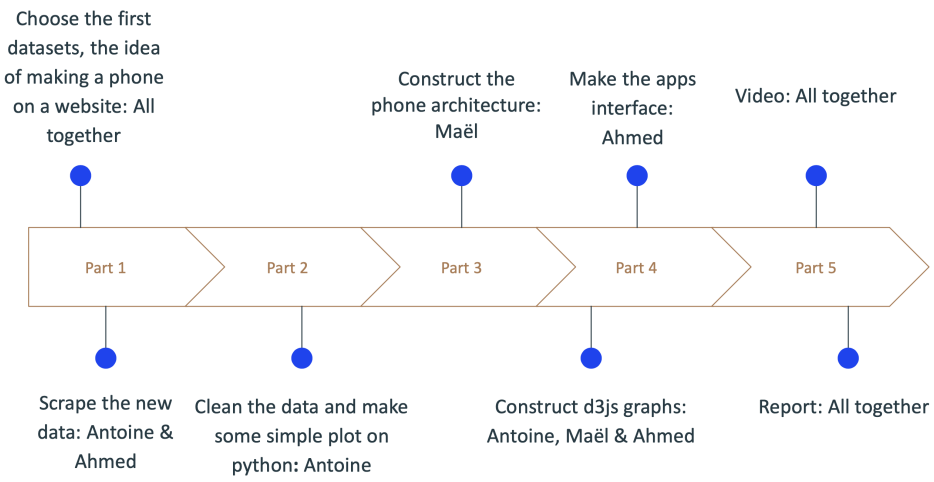


Figure 8: Design of the phone's lock screen.

At the moment where everything was structured, the data story could finally be followed. As explained in the very beginning of this process book, the goal is to gradually have info about the data to finally understand at the end the specificity of each feature that could make an app successful. We thus wanted the very first app to be introductory regarding the data by displaying a global ranking we obtained. The following app stated some basic analysis, and the following ones went deeper by describing more specific characteristics. Finally the last app recaps all the results and also acts as a conclusion.

3 Peer assessment



4 Conclusion

The success of an app is something we can evaluate through formulas but it is hard to determine how to achieve this success. We tried to make a website to give people that are new to this domain a small insight of the keys to success.
At the end of this project, we can all agree that we have learned a lot, not just programming skills but also how to "design" a project from the beginning, how to express our ideas to be understandable and how to work in parallel. This has been a bit challenging at some points but we made it.