Visualization of velo STAR stations and car-park in Rennes

Brage Anthony   De Crevoisier Tangui   Hassani Kassim

1 Abstract

Currently, the city of Rennes has no application in order to display in real time the filling rates of bike stations in free circulation. The idea here was to set up a visualization to locate in real time this information on the map of Rennes. In addition to this aspect, a representation of parking filling rates has been added. Thus, the two main added-values of this application are the following:

- Specific user : If a user hesitates between taking car or a velo star, he will be able to look at this visualization in order to know if there are places available near his final destination for the desired means of transport. This visualization can be used as a consulting tool.

- Municipality of Rennes : The observation of the activity in the different car-park or velo STAR station gives the possibility to understand, for the car-park , if there is a need to increase the number of places or to decrease it. About velo STAR stations, it can provide informations about which one are little or over used and determinate if it is worth to move, delete, or add either stations or new bikes.

2 Introduction

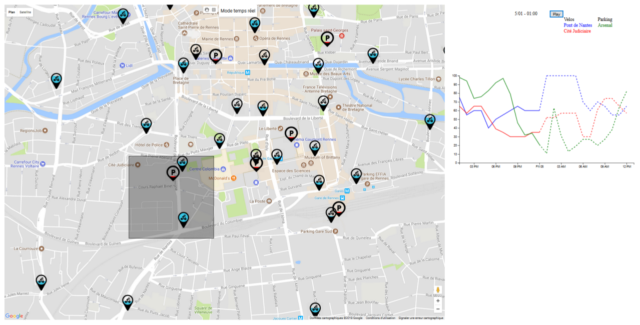


Figure 1: First show of the visualization

The representation of datas, usually called Data Visualization is a large topic that can be treated in different ways. Based on the mode of transport called *Velo'v* in Lyon, we wanted to create a specific visualization of theses bikes. The system of *velo'v* works like this, there are some stations in the city with bicycle and you can rent it for the price of 1 for 1 hour. When you finish to use the bicycle, you have to deposit it in an other station which is not full. That is why, we thought that users of *velo'v* need informations about stations, in order to see quickly the number of free spot. An application already exist in Lyon, that is why we decided to create it in Rennes. The current name of theses bikes in Rennes is *velo star*

The first approach we wanted to do was a representation in real time of all stations in Rennes. As we said before we will see in the next part that something similar already exist even if it is not exactly the same. That is why we asked ourselves the following questions: What does impact the use of *velo star* ? What can modify the disponibility of *velo star* in a station ? What kind of visualization can show informations about this ? Or what can we add to get some value-added ? At the beginning, we decided to study the impact of affluence in bars on *velo'v* station. Why was it interesting ? For example, bars close at 1 a.m and last public transport are generally at midnight, so if we can see that the situation of *velo'v* stations evolved at closing times of bars, we can suppose that people take *velo'v* after drinking and take measures in case there are no bikes left to avoid them taking the car for example.

However, after discussion, we understood that the correlation between *velo'v* and bars was too small. That's why we thought about it : Could we add something on the map in order to help people in their everyday life ? Indeed, when a person want to go in the center of the city, the first question is : Should I take my car ? Even if in most of the cases, peoples prefer to use public transport, we wanted to provide an alternative to this, and create a tool that could help them to choose because there is no application about that. Hence, we decided to propose a visualization which show in real time the situation of all *velo star* station and car-park in Rennes. So, if someone want to go in a specific area in Rennes, he will have the possibility to check on this visualization if there is a *velo star* station near the destination, if there is a car park next to it and if there are spots left in both of these park. Then this will help him to take a decision. After that, we thought about other useful tools we could add to this visualization. A prediction of park's situation of next hours should be interesting to add so as to help people to forecast a specific trip.

For this visualization, we needed a lot of data about *velo star* stations. To do that, we stocked five days of data in order to see the *velo star* station situation and park-car evolution over time. An API can provide that kind of data, we will explain more in details how we retrieved them in this article.

This article will first describe some related work like application already done, then we will describe in details the goal of that project. After that, we will talk about discussion, what we could add to that visualization, which kind of problem we faced and finish with a conclusion.

3 Related Work

The main problematic of this project is to be able to represent in the best possible way, some data about *velo star* and park-car. The panel of choices was very large and we made our choice based on available data.

3.1 Velo'v application in Lyon

The first one is a released project concerning the city of Lyon [1]. This website contains a tab called Les Stations and provides multiple kinds of visualization (figure 2). On the first visualization, we observed a map which uses zoom interactions to focus on *velo'v* stations in a certain area. When there are many stations to display it is represented by a gray circle with the number of stations inside. To see in details one of the station, the user have to click on it. Theses stations are represented with markers. When a marker is red, this means that a station is full of *velo'v* meanwhile when a marker is white, the station does not contain any *velo'v*. When you click on one of theses markers, further informations are given like the number of available *velo'v*, the number of available stands, and an availability forecast for example. The others visualization in this website are almost the same. One of them displays where to rent a *velo'v*, another where to return one. Its also possible to create an itinerary.

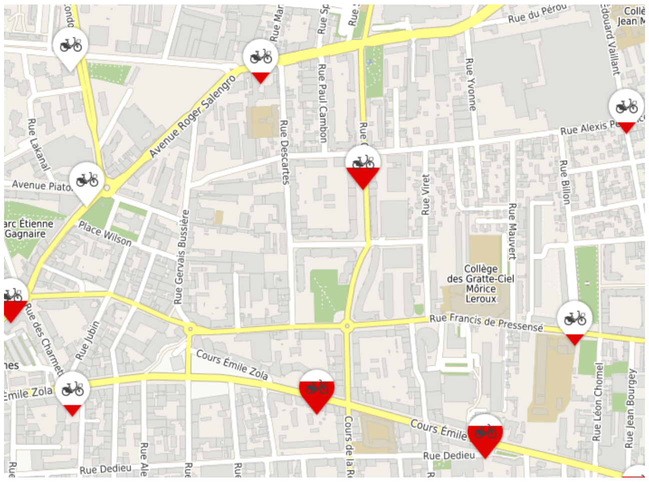


Figure 2: Sample of official Velo'v application

3.2 Velo STAR application in Rennes

The second application released is about the city of Rennes [2]. This application allow users to see where are the velo star station(figure 3). This can be useful for users if they first want to see where are stations. Then if you click on a station, you can see the number of *velo star* available and left spot. The problem here is that you can't understand just with the visualization where you have the most of left spot in a area, you have to check all of them.

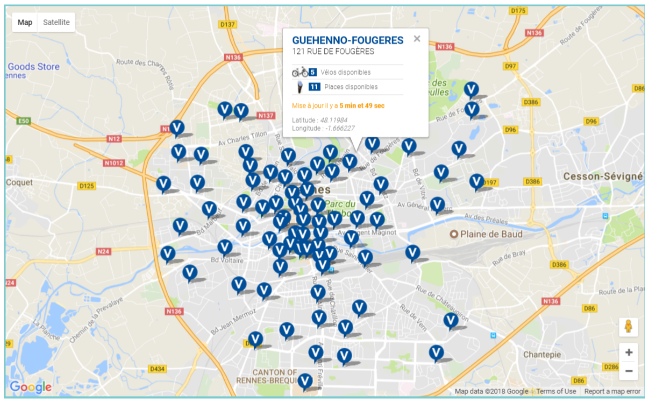


Figure 3: Velo STAR application

3.3 Parking application in Rennes

The third project is about the park-car in Rennes [3] . Indeed, this visualization allow users to get the price of each park-car and the number of spot left. This can be important for users who want to get the cheapest park-car or just to have some information about it. We didn't want to add the price to avoid an overload on the visualization. Furthermore, the goal of our visualization is not really focused on the price of the ride but more on the accessibility of a area.

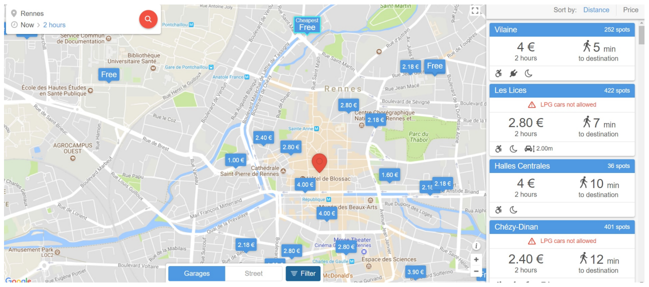


Figure 4: Parkopedia application : Rennes

3.4 Links between metro and velo'v

The fourth project deal with the influence of public transport over *velo'v*. This project was done last year in the context of the course Data Visualization at Lyon 1 [4]. The purpose was to observe the impact of *velo'v* stations near the metro one (figure 2). This project use a map representation to show the different metro lines and the different stations. This visualization is interesting for us because it shows the evolution of *velo'v* stations over time, which is something we will need to do for our visualization.

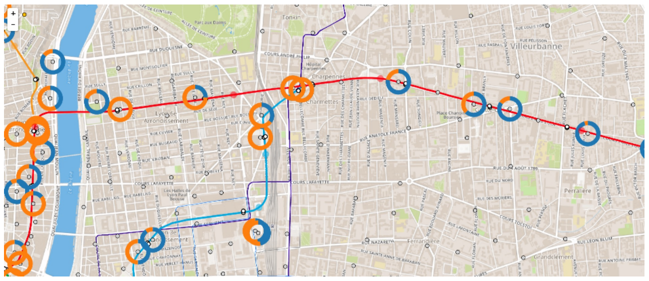


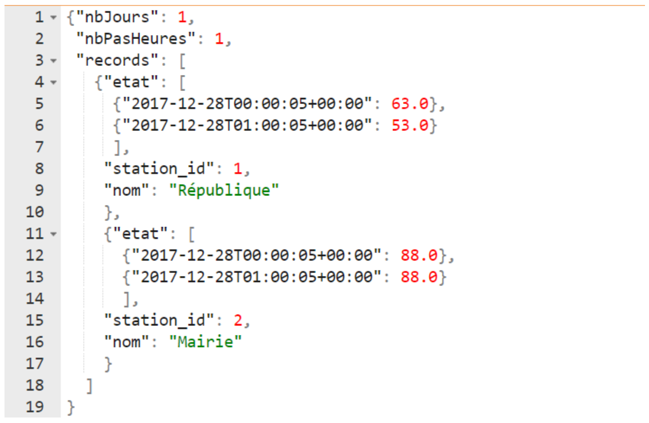
Figure 5: Simple of visualization between subways and velo'v

4 PROJECT DESCRIPTION

4.1 Data used

In order to fill our app, we decided to select two different datasets from the open data portal of Rennes. Theses datasets provides us the occupation and prices of Rennes park [5] and states of each Le velo STAR stations [6]. We retrieve theses data through their respective API.

To build our historical data, we decided to create a python script whose job is to request each hour the two API and store the fresh data in a JSON file. This script was launched between the 1ST January and the 8th January on a virtual machine provided by the University Lyon 1. However, it appears that the API on which we requested crashed during the data collection, leaving us with an incomplete dataset (cut the 5th January), but still enough to show something. We used the library python APSScheduler that lets schedule functions to be executed each hour. The car-parks API and VeloStar API are not structured in the same way, we decided to create a new structure and keep solely useful information.



```
{"nbJours": 1,
 "nbPasHeures": 1,
 "records": [
   {"etat": [
     {"2017-12-28T00:00:05+00:00": 63.0},
     {"2017-12-28T01:00:05+00:00": 53.0}
     ],
    "station_id": 1,
    "nom": "République"
    },
    {"etat": [
      {"2017-12-28T00:00:05+00:00": 88.0},
      {"2017-12-28T01:00:05+00:00": 88.0}
      ],
    "station_id": 2,
    "nom": "Mairie"
    }
  ]
}
```

Figure 6: Structure of our generated JSON file

- ”nbJours” represents the number of days collected
- ”nbPasHeures” the steps (in hour) between two dates
- ”station_id” the id of the car-parks or *velo star* ,
- ”nom” the name of the place
- ”Etat” contains all records of occupancy rate.

For example, records[i] contains all fill rates of the station or car_park with the id records[i].id_station. Elements in ”Etat” are dictionaries where the keys are timestamps and the values are a fill rate.

For ours visualizations, we request on real time the two datasets from the API of Rennes. We collect them but we did not saved theses data in raw. However, we perform some tasks like retrieve different characteristics of stations (Available bikes or car park, available stand, name of place). GPS coordinates are used to place *velo star* stations and car-parks.

4.2 Description of the visualization

Our application provides two different visualizations. The first one, aim to show at the current time, the state of each parks and *velo star* stations and tells to user what is the best mode transport to use. The second one, is to show the evolution over a passed period of time. Theses two visualizations contains also a graph, useful for the user if he wants to compare severals stations.

4.2.1 Library and API(s) used

The project has been developed with the D3 library. We made the choice to use Google Maps API [7] and jQuery library in addition for some components.Our visualization is built around two main area zone. The first area displays Rennes map and the second one displays multiple time series graph.

To modelize the city, we used Google Maps API. This API provide us many functionalities we have exploited. The first functionality is the one to display the map. We decided to use this API instead of D3 because we have the possibility to display the road names and some other things like monuments and important places of the city (university, hospital, town hall). With D3 library, it was also feasible, we might have been able to draw Rennes city with their boundaries, but we might not been able to place markers car-park and bikes as easily as Google’s API. We choose to use the Google Map API also because data reload faster than with D3. Furthermore for the visualization related to a defined time period, at each state modification, each bike station and park are redrawn. In addition, some modifications does not appears, and this anomaly has an impact on the visualization.

We also used a tool which provide the selection on the map too select the different stations and park. The API allows us to draw on the map, and draw every kind of polygon, but, we decided to allow the user to draw only rectangle on it. Stations and car-park which are displayed in the rectangle zone build multiple series on the line graph of the second area thanks to informations gathered through request on API.

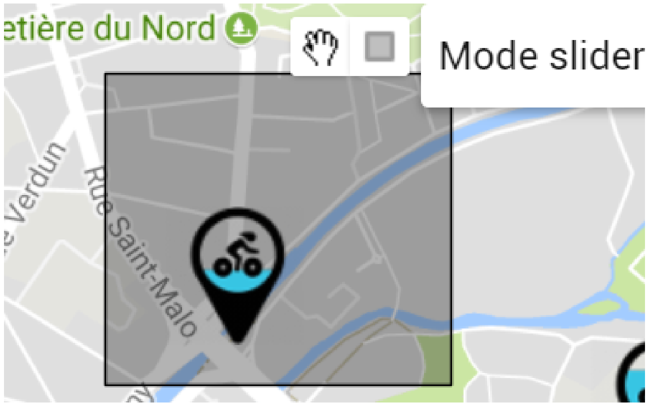


Figure 7: Example of draw rectangle on the map

We used D3 library to add Rennes boundaries. Theses boundaries are retrieved through the OpenStreetMap API. The Google Map API doesnt propose this functionality natively. The second area zone contains a line graph, a slider and a play button. The graph is entirely realized with D3 library. It uses our generated data for the visualization related to a defined time period, and use data from OpenData Rennes API’s.

4.2.2 Visualization in real time

The main goal of that first visualization is to show, in real time the situation of *velo star* station and car-park. As we said in the last part we firstly created the contour of Rennes.

Then, on one hand, we had to place the station of *velo star* on the map. For that we retrieved data from the API. Thus, we took only informations we needed from it. For each station, we compute the filling rate like this :

$$fillingRate = \frac{numberAvailableBikes}{numberSpotTotalStation} \qquad (1)$$

Moreover, we kept the name of each station and the coordinates. After that we placed the *velo star* stations on the map with the use of theses coordinates and we chose a icon with a bike inside. For this visualization, we fill each icon of station with the filling rate. It means that the part of icon which is filled represent bikes in station and the rest is the number of spot available to deposit *velo star* . About the visualization, when you click the cursor on a station, it displays the name off the station, the number of bikes and the number of available spots.

On the other hand, we did the same thing for the car-park. We just avoid to display the number of car in park-car because this is useless for users.

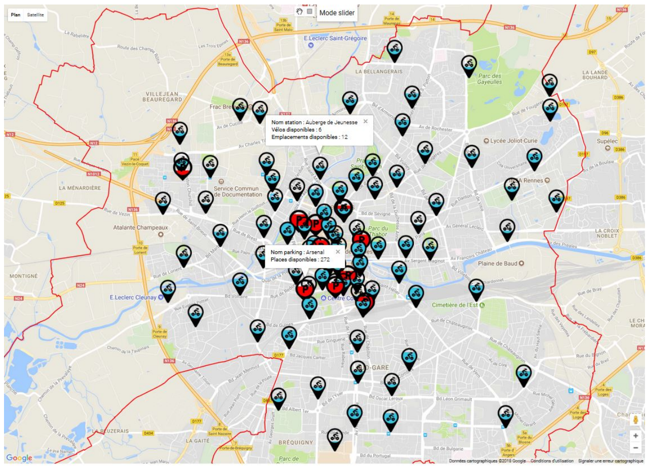


Figure 8: Example of visualization in real time

4.2.3 Visualization related to a defined time period

To access to this visualization, the user needs to click on the button Mode Slider. *Velo star* stations and parks are already placed because of the visualization based in real time. This data visualization requests on our generated JSON file. A slide bar, and a "play" button appears on the second area.

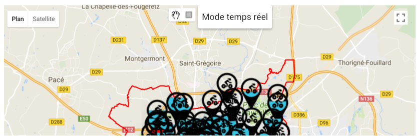


Figure 9: Example of visualization with a slidebar related to a defined time period

We decided to implement a second visualization to add value to our project. The main value is to see the development over time of the different stations and parks. A user can slide the cursor of the slide bar and according to the position of the cursor, the map displays the state of the selected items at a specific date and hour. When the play button is pressed, maps and graphical visualization are updated, and we can see the evolution of every car-parks and stations of the city and the graph is redrawn depending on the position of the cursor from the slide bar. This update provide a dynamic visualization To use it, we need to iterate through our generated JSON file.

4.2.4 Graphical visualization

The graph displays multiple lines charts if a rectangle is drawn by user. The x-axis represents the time, the y-axis represents the fill rate of the car-parks and *velo star*. We use the same scale for each station, to standardize, and because it is easier for the user to understand the meaning of each line chart. The graph is also time-centered on a date, that means that the x-axis display the values on an interval of [date - 12 hours; date + 12 hours] based on that date, and when the date is changed, the axis will change automatically.

Concerning the data visualization itself, it is possible to display a large number of stations, but to keep a certain visibility, we defined 7 colors for selected stations, hence, the graph is readable as long as the number of selected stations is 7 or below. The graph contains also a legend that indicates the name of station or car-parks by referring to his color. It also display the corresponding filling rate of the last hour of all stations selected by passing the mouse on a specific line and at the desired hour (If 2 stations are selected, passing the mouse on 14h30 will show the values of both stations at 14h).

The slide bar mentioned before permit to also change accordingly the data displayed in the graph. The lines of the graph are stroked and dotted and each one represent an aspect of the data :

- Stroked part is all about real values, they are displayed between [date - 12 hours ; date] and based of the JSON file that contains data of the APIs

- Dotted part is about the prediction and they are displayed between [date ; date + 12 hours]. At first, the prediction was supposed to take the average filling rate of a specific station during the 7 last days and consider it as "what is going to happen". But due to some issues with retrieving data (that we will talk about later on), it was safer to take only the last day. In fact, to display something we need enough data to predict what is going to happen, that is why even thought we have data from 1st to 5th January, it is starting from the 2nd in the visualization, because prediction of that day is based on what happened the 1st.

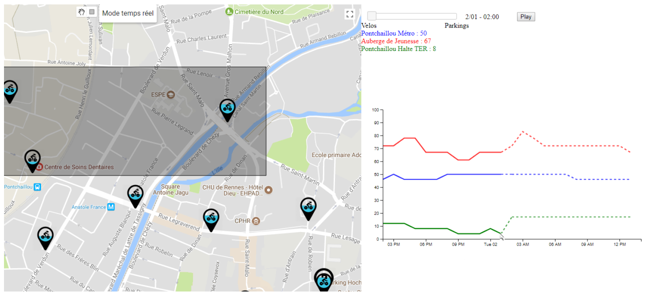


Figure 10: Graph visualization with legend

5 Discussion

5.1 Options to add

5.1.1 Clustering of markers

The first thing that jump out when you open the application is the mass of indicators on the map. Actually there is no automatic option to agglomerate them in order make the visualization less dense, adding this should be the highest priority because it is a visualization criteria. An easy way to achieve this to calculate the distance between two markers pondered with the z axis and if the result is over a specific limit, to regroup them as a cluster, this way the user will understand that more than one marker is here.

5.1.2 Zoom on a specific area for users

A tool we could have had on this visualization would be a search box. Indeed, if a user want to enter a specific destination (street address, restaurant name, company name), he could type it in the search field and it would automatically zoom on the area he want to see.

5.1.3 Add the possibility to show only *velo star* stations or park-car

An other tool we could integrate to this visualization would be a possibility for the user to see only *velo star* station or car-park. In a situation where someone know which means of transport he want to use this could improve the visibility. A way to realize this could be two check boxes, the first one to select only *velo star* station and the second for car-park.

5.1.4 Consulting tool for users

If the user want to go to a specific area with *velo star* , he need to take one in a station and then to deposit it in an other station near from his arrival point. Thus, we could integrate an option on the visualization which could take in a first search box the starting point and in an other one the final destination. Then the tool could propose some *velo star* stations near the point of start which are not empty and some near the point of arrival which are not full. Finally the user could choose what he prefer between the proposition.

5.2 Difficulties encountered

During the realization of this project, we encountered several issues. The one that caused us the more troubles is the real-time visualization. In fact, the real-time is working for the map, but to achieve the prediction correctly, we first need enough data to be able to predict something, and then we need them in an accessible file. The main problem with the first point is that the availability of the API is not dependent of us, during the development, we could not a test a lot a times because the API had an ”internal error”. Even through, to be able to display these datas in the case we manage to get them, we need to store them on the server and that can be a bit tricky. That is why we cannot use the graphical visualization in real-time mode.

The other major issue we had, and sadly, is still present, is the markers. Sometimes, for a reason we could not find and resolve in time, bike or car-park markers are switching, not showing correct data and even disappearing. We already know why (or at least we have an idea), but we could not find the time needed to repair this.

6 Conclusion

To conclude on this project, we managed to create a visualization that can show data either in real-time or based on older days of the city of Rennes. In one hand we have a spatial visualization on a map with markers on it, and in the other a graphic that can help to easily compare the filling rate of each station with also a small prediction.

Thus, on one hand, the first visualization in real time will be very useful for an user in Rennes who hesitates between taking the car of a velo star . It provides an easy way to understand the exact situation just with a look on the map, which is a great upgrade compared to what is actually existing. Plus, we made a prediction of the next twelve hours to allow the user to know how will be each station in the future. That is why he will be able to anticipate a trip during his day.

On the other hand, the use of a slide bar on a defined time period is a real added-value for the municipality of Rennes. Indeed, when you use the slide bar on some days, you can easily understand which stations or car-park are really used. Then, they could deduce that some stations are useless or too often full, and it could allow them to delete, add or move some either stations or bikes. If they think about car-park, they can reflect about which one does not have the optimized number of parking place and modify it in consequence.

References

[1] V. G. Lyon, “Location de vlos en libre service dans lyon et villeurbanne.” https://velov.grandlyon.com/fr/les-stations.html.

[2] “Le velo star.” https://www.levelostar.fr/fr/stations/plan-dimplantation.html.

[3] “Parkopedia.” https://www.parkopedia.fr/parking/rennes/.

[4] L. S. et PICARD Colas, “Visualisation de l’impact des arrives de rames de mtro sur les stations vlo’v.” https://stanislasleroy.github.io/DataVizProject/.

[5] “Api vlos rennes.” https://data.rennesmetropole.fr/explore/dataset/topologie-des-stations-le-velo-star/.

[6] “Api parking rennes.” http://www.data.rennes-metropole.fr/espace-developpeurs/api-parking/.

[7] Google, “Google maps api.” https://developers.google.com/maps/?hl=fr.