Programming Project June 2015:
NPO Geo Finder

June 26, 2015

Jonathan Klaiber 10004077

Contents

1 Introduction

The basic idea behind the NPO Geo Finder application was to give the general public a tool to their disposal whereby the can find an "NPO Journalistiek" (NPO-J) item based on a location. Thus, people that search for an NPO-J item can insert a location and will be provided with items associated with this location. Further, information about the discovered items is supplied; the date of production, the broadcaster, a short description of the content and as a matter of fact the link to the NPO-J item. "NPO Journalistiek" is an Internet platform on which the Dutch public broadcaster "NPO" provides journalistic items of its local subbranches[1]. The local sub broadcasters that deliver the content for "NPO Journalistiek" are "AVRO", "BNN", "EO", "KRO", "KRO-NCRV", "MAX", "NCRV", "NOS", "NPS", "NTR", "TROS", "VARA" and "VPRO". Items on the NPO-J platform are quite diverse, an item can be for example a documentary, a piece of news, a written offline or online article or a data visualization story (a kind of extended and data based article).

2 Application Purpose

The NPO Geo Finder application can be structurally divided into three main components as depicted in Figure 1. First, the most important component is the map with search box in the center of the application. Users can insert a location in the search box by which the NPO-J items are searched for. Besides, the map component displays the searched location and flags it with the commonly known Google Maps marker (it looks roughly like a red colored upside down waterdrop). The items that were found are also flagged with a marker that shows the item's broadcaster. Second, the episode list component is a white panel at the right side of the map. Items that do not have a location are displayed in the episode list. There are maximum of 10 items in the list and they are sorted by date. The third component is the filter bar. The filter bar is opened by a click and allows the user to apply filters. The program filter filters the local broadcasters that should be in- or excluded in the search. The second filter allows the user to choose a certain period of time to filter NPO-J items based on time.

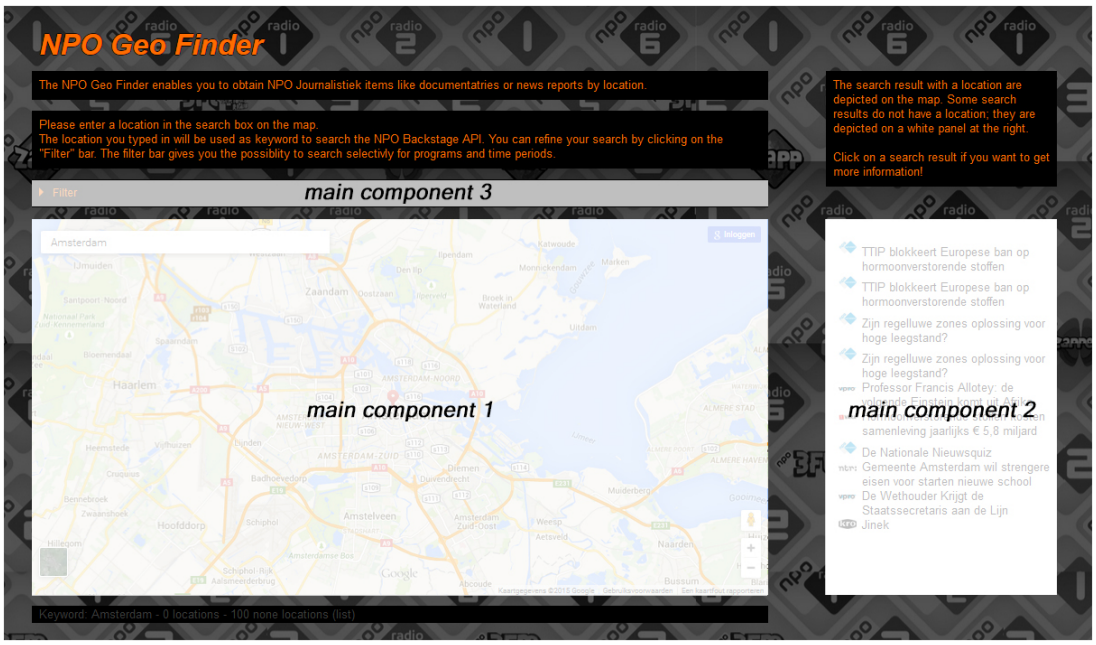

Figure 1: Structural division of NPO Geo Finder application.

The task of the visualization should be to make it easy for users to find NPO items associated with a location. The objective is to provide an application that is as simple as possible to enable

[1]Information retrieved from `http://journalistiek.npo.nl/over-npo-journalistiek`

user to getting started straight away without much explanatory text. When the user has obtained some search result, the task of the visualization is to provide him/her with extra information about the results. The main purpose of the NPO Geo Finder is to link users to the item that they want to "consume". The basis of this link is a location. The NPO Geo Finder links users and items by a location. In order to fulfill this purpose every search result provides information, among others, an item URL link whereby the user is redirected to the item. Finally, the task of the visualization also entails that a user can refine his/her search to obtain more precise search results. This is necessary to "reach" all possible data points because the NPO Backstage API[2] (NB-API) search is limited to 100 results per request and per default sorted by date. Without the filter function, the visualization could not provide a link between the user and all data points (all NPO items).

The raw data with regard the location is the name of the location and the geo location (latitude and longitude values). The name of the location is used to generate a request to the NB-API. Next to the location name as the keyword, the chosen filter values are attached to the request. The incoming data from the NB-API is firstly formatted with regard to the broadcaster. The default option is "npo-journalistiek", if no broadcaster is supplied by an item. It is crucial to note that the program (or broadcaster) filter is applied at this point. When a program is not in the request (because it is unchecked in the program filter bar), the items that are tagged with this program will be deleted here. In contrast, the date filter is directly applied on the NB-API request. This set-up has been chosen in this way because it is not yet[3] possible to provide an NB-API request with a program filter (and must therefore be filtered locally). Secondly, the incoming items are divided into location and non-location items. All items are transformed into NPO item objects (with or without location data) and based on this division either depicted as a marker on the map (with location) or as a list entry (without location). When incoming data is transformed into NPO item objects, only necessary information is stored like, id, title, program, date, location, URL link, description and distance to search marker (only for items with location).

3 Design

Figure 2: Terms to refer to objects of the NPO Geo Finder.

[2] http://backstage-docs.openstate.eu/user/api.html

[3] Through correspondence with the NPO Backstage API admins I came to know that extensive filtering on the NB-API request will be available from some point in July 2015 on.

Figure 2 depicts terms of important design features to be able to talk about the design choices that have been made. The NPO Geo Finder application's main component is a map. The purpose of the application is to link users via a location to NPO items and therefore the most straightforward and intuitive choice was to use a map. Items without locations have been chosen to be represented in a list. It would not have made sense to depict these items on the map as well because it would have lead to confusion and distraction with regard to items that actually have a location. The general design idea was to loosely base the design on the design of various Google products. Google's core purpose is to provide a link between users' search terms and websites. They create this link by means of a sorted list. Google Maps purpose is to provide a link between a location search term and the location, they do that by means of a map. This boils down to the general principle that when an application is an agent or mediator between user input and output, the linking should make use of the relation and nature of input and output (search term $\rightarrow$ **list** $\rightarrow$ website, or location search term $\rightarrow$ **map** $\rightarrow$ location on map). What comes in handy when a visualization's design is partly based on Google products is that is has a very high recognition rate. Almost everybody who uses a computer and the Internet is also using one or more Google products. Thus, almost everybody already knows how to use and interpret the various design features.

The NPO Geo Finder does not only make use of this principle with regard to the map and the item list but also to the result bar and the filter. The result bar at the bottom of the map displays the amount of search results obtained in a manner that is well know from the Google search and hence easy recognizable and interpretable. The font color (it's dark grey) of the result bar is chosen to be not very different from the background color in order to not highlight the result bar to much. The amount of search results obtained (information that the result bar depicts) should be an implicit feature that should only be of interest for a few users (for example to check how many locations there really are in total, thus also outside the current view box borders). The filter bar is a feature that is well known from for example Google image search ("advanced search" in the most recent version of Google image search) and hence is intuitive and straight forward to use. The drop down bar with a small arrow is a common icon that users associate with a clickable feature on an interface and will therefore facilitate usage without much extra explanation. Finally, the general element sequence from top to bottom, the introduction text, the filter bar, the map (and next to it the list) and the result bar at the bottom depict a logical (in terms of functionality) top to bottom flow. The top to bottom stream works as follows: The introduction is read, next the filters are applied to narrow down the search, the search term is inserted and the results are depicted on the map and in the list. The user can interact now with this (most) important information. Interested users may even move further down to have a look at the result bar, but it is not necessary to take this last step. All in all, the arrangement and kind of feature display a logical control flow that makes the use of the application simple and intuitive.

Figure 3 shows the three different types of markers used in the application. To make it easy for users to interact with the obtained search results, the markers and list entry bulletins are shown with images of the associated broadcasters. At a glimpse, users can categorise the search results because the broadcaster gives them an explicit hint on what kind of content can be expected without even reading the titles in the list entries (only non location items). This feature allows the user to make a quick preselection of which item is the most interesting for him/her.

Figure 4 shows the two types of info windows. The choice of the info windows, the info window of a marker and the info sublist in the list, were based on simplicity and clarity. The info windows use clear labels (like date, link or description) to show the available information in simple and plain black colored text. The goal of the info window design choices is to reduce distraction while providing the necessary information.

Figure 3: Three different types of markers.

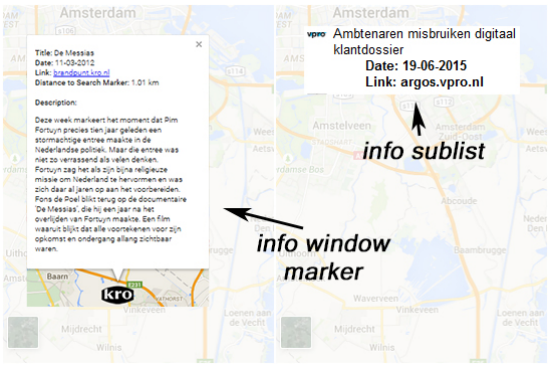

Figure 4: Info window of marker and info sublist in list.

4 Implementation

The NPO Geo Finder depends heavily on two external data sources that are vital for its functioning; the Google Maps API (GM-API) and the NPO Backstage API (NB-API). Major parts of the application's functionality with regard to locations uses the GM-API: The location search box and the map. The auto-complete while inserting a location into the search box. Information about the searched location, for example the name, latitudinal and longitudinal location information and borders or dimensions of a location (used to determine what parts of the maps have to be shown to the user). Search marker objects and object properties. Map interactivity like zooming or dragging the markers. While the GM-API mainly provides functionality that ensures that the application works properly, the NB-API delivers the content. The NB-API is used to send requests to the NPO-J platform that contain all information (keyword and filters) the user is looking for. Communication with the NB-API is triggered by inserting a location and searching for it. The answer of the NB-API contains extensive information about the retrieved items that is used in the visualization of the search results.

At the point where search results have to be transformed into item objects in order to be displayed in the application, the JSON library is used to parse the JSON formatted data into item objects. This choice seemed straightforward to me because we could already obtain experience with this library during the data visualization course. The transformation of data from or to JSON does not play a big role in the application, but I name it here because of completeness.

In all parts of the application that are related to functionality of the GM-API, the GM-API own methods and objects are used. An example is the marker object[4] of Google Maps. This object can be applied when a marker has to be set on a Google Maps map and is straightforward in its use and customizable to an satisfying extent. When there are several markers on the same location, an issue that was raised more than once be my fellow students during the presentations and that I knew to encounter at some point, the Google marker objects can be set to be draggable, a relatively easy solution for this problem. Moving the markers to some nearby location with a reference to the overlapping location would have been another easy to implement solution but would have had distorted the results in my opinion. The info window of the markers[5] is also a Google Maps method that I decided to implement because it is clear and works well together with the map.

Interactive parts of the application; filter, episode list and result bar are primarily made with the D3.js[6] library. The D3.js library is the first choice for interactivity in Java Script because it is flexible, relatively easy to implement and it saves a lot of extra work. The filter drop down bar and the episode list are created at the moment that they are called, by clicking on the filter bar or by inserting a location, and removed when they are called again instead of making them hidden and not hidden. I could implement the filter toggle and episode list toggle in this way because the application does not eat up a lot of resources. With creating these elements every time they are called I could avoid getting bugs because of overlapping interactive elements.[7].

5 Project Reflection

The original idea[8] for the NPO Geo Finder application was to search for an location and to see all NPO-J items that took place on or near by this location. On the first functionality graph the setup was to get the map borders of the location that was searched for and to feed these border location as a filter into the NB-API. Unfortunately, this plan did not work out entirely. After some troubles in the beginning with the NB-API because it was temporarily down for maintenance work before the weekend, I realized once I set up a test API connection in python (to find out whether my Java Script was not working properly or the API; it turned out to be the Java Script) that with every search request a keyword has to be provided. Moreover, after I exchanged several emails with a NB-API admin it became apparent that filtering on locations was not possible. Figure 5 shows an email from a NB-API admin regarding sending requests without a keyword and about filtering. I chose to use the inserted location name as the keyword for the request to make the visualization working. This was not really in line with the original idea but the best possible solution with regard to the current functionality of the NB-API.

Another issue I encountered during working with the NB-API was the mixed quality of the NB-API data. Unfortunately only few NPO-J items have location information available (my rough estimation is that 10% of all NPO-J items have a location tag), certainly less than I had expected. Another drawback of the NB-API is that the item links are broken in almost all cases (I guess about 95% of all links). The low amount of location information and the vast amount of broken links definitely limit the usefulness of the NPO Geo Finder. I think that these NB-API data issues are due to the recent start of the API. In the course of time the quality of API data will probably improve as the applications of more and more people rely on the NB-API data. To conclude, in contrast to the design document the functionality of the NPO Geo Finder changed to some extent mainly because of the data and functionality of the NB-API. In my opinion the current version of NPO Geo Finder is the

[4] `https://developers.google.com/maps/documentation/javascript/markers`

[5] `https://developers.google.com/maps/documentation/javascript/examples/infowindow-simple`

[6] `http://d3js.org`

[7] At some point many elements of the application worked with making them hidden and not hidden which caused a lot of problems, although, when implemented correctly, this should not cause any problems.

[8] see design.md in repository

closest that I could have come to the origininal idea given the circumstances. Once the functionality of the NB-API's filter has been extended (see email in Figure 5), the Geo Finder application can be easily transformed into the original idea.

An improvement of the NPO Geo Finder application would be the implementation of a sort function to give the user the possibility to sort the data that is coming from the NB-API because the maximum of data points per request is limited to 100. Users could further refine their search by providing them with multiple sort options, for example to sort the incoming data based on relevance. A sort function would make it easier for users to reach all possible data points. In the current implementation the incoming data is sorted by date. This does theoretically not hinder the user to reach all available data points because of the filter by time period function but it could be made easier for the user to do that.

In terms of design, I'm satisfied with the decisions I have made in the design document and that are explained in the design section. Some minor design issues as for example the style of the episode list and of the info sublist, could be addressed and improved but the general basic design ideas work well.

Hi Jonathan,

Leuk idee voor een app! Momenteel kan je nog niet zoeken zonder query, maar ik ben er mee bezig om dat wel te ondersteunen. Over max. een week werkt dit.
Wat betreft het filteren. Ik ben er even achteraan gegaan en dat kan ook niet. Ik heb de documentatie aangepast die nu duidelijk weergeeft op welke velden je kan facetteren en filteren (rights, source_id, collection, author, date, date_granularity, media_content_type). Dit ga ik uitbreiden zodat dit wel op alle andere velden kan! Als het goed is kan je dan je app maken :). Ik laat het weten wanneer deze veranderingen zijn doorgevoerd.

Wat betreft de meet-up. Die hebben we verplaatst van 17 juni naar donderdag 9 juli. Na de meet-up hebben we dan namelijk onze office warming, en dat leek ons een leuke combinatie :)

Groeten,
Sicco

Figure 5: Email answer NB-API admin 11.06.2015.