Visualization of Information for: ”Concurso Nacional de Acesso ao Ensino Superior”

Micael Pedrosa[1]
Universidade de Aveiro, Aveiro, 3810-193, Portugal

I. Introduction

This document describes the development of a client-server software whose purpose is to present several visualizations from gathered data of the ”Concurso Nacional de Acesso ao Ensino Superior”, the students positions for Portuguese Universities. The source code is available at `https://github.com/shumy/vi-universities`.

II. Methods

The software is composed by a server part developed in the Xtend [1], and the client UI using the Angular [2] framework. The server has an embedded graph database [1] from the Neo4J [3] project that runs in read-only mode, providing flexible schema and other valuable features [2].

A. Data Model

The raw data is available as files in CSV format. From the analysis of those files, a CSV parser was developed in the class **CSVReader**. In the source code it's easily identifiable the order of the parsed fields in the ”fieldMapper” array, where there are recurrent fields that have a numeric postfix identifying the order of application in the CSV line. The CSV fields are then mapped and loaded to the model in Fig. 1 in the class **ViuCLI**, transforming a flat model in a normalized relational model using graph relations. The mappings from the CSV fields for each class is done as following:

Student
name → Nome
grade_10_11 → Nota10_11
grade_12 → Nota12

Institution
code → OpcaoInstituicaoCodigo
name → OpcaoInstituicaoNome

Course
code → OpcaoCursoCodigo
name → OpcaoCursoNome

Contingent
name → OpcaoContingente

Application
order → ”the numeric postfix”
year → Year
grade → OpcaoNotaCandidatura
applicant_order → OpcaoOrdemCandidato

Relations are inserted as following: the ”from” relation connects the Student to the Application year ”on” the Course ”of” the corresponding Institution. The ”Colocado” CSV field is mapped to the ”placed” relation when the value is true, used to indicate that the student was accepted in the course.

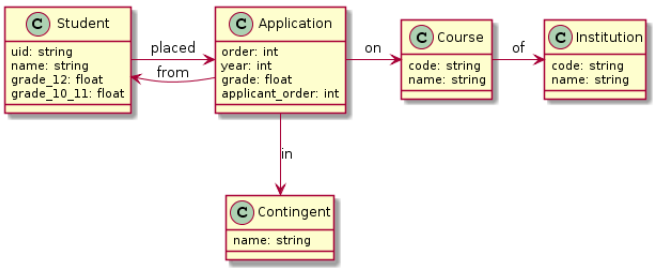

Fig. 1. Normalized model representing available relations and fields to query via Cypher.

The model can be queried with the Cypher [4] language specification through the query service. These queries are used by the UI to feed the visualizations. The Cypher syntax is very straight forward to understand, e.g. if we what to ask the DB the question ”Give me the maximum and minimum grades and the number of applications by year”, this can be done with: ”MATCH (a:Application) RETURN a.year, count(*), min(a.grade), max(a.grade) ORDER BY a.year”.

B. Query Service

The query service works with the graph database in read-only mode. Any Cypher syntax trying to change the data returns in error, no other security measures are necessary. The service is provided at the url: *http://host:port/query/:cypher* with the input ”cypher” parameter and returning the JSON corresponding to the query RETURN structure. The Spark-Java [5] lib is used to build this simple web service, located at the **HTTPServer** class.

[1]https://www.eclipse.org/xtend/
[2]https://angular.io/
[3]https://neo4j.com/
[4]https://neo4j.com/developer/cypher-query-language/
[5]http://sparkjava.com/

C. Visualizations

Visualizations where mainly constructed using the d3js [6] library, for the exception of the overview presented in the home page, that used Google Charts [7]. Interactivity is provided by Angular Material [8] components. Visualizations where designed following a pre-defined life-cycle: retrieving data from the server using Cypher queries, transforming the JSON result to fit the visualization model, initiate d3js scales and draw the visualization. Resize is applied on the *window.onresize* event, and refresh when input data parameters changes. This methodology is standardized to all d3js visualizations.

III. Usage

This section describes how to compile, run and use the available tools provided by the software, as also how to use the global and local visualization filters.

A. Compile and Run

Instructions on how to install the required tools are available in the respective sites. The server part of the project is compiled via Gradle Build Tool [9]. In the "server" folder execute the "gradle" command to compile. Execute the "gradle export" that will download all the required dependencies to run. The NPM [10] and Angular CLI [11] tools are used to compile and run the UI.In the "uu" folder, execute the "npm install" that will download all the required dependencies. Optionally, the "ng build" can be executed to build a deployable web version.

Once compiled, the server starts with the available Command Line Interface (CLI) via"./viu-cli –server" and provides the service on port 4567. Inside the "ui" folder the "ng serve" command (available in the Angular CLI) can be executed to start the UI service. The UI should be available at port 4200 by default.

B. CSV Load

Since the only web service available uses the DB in read-only mode, the way new data is inserted is via the Command Line Interface (CLI). Loading a CSV file in the correct format is executed with "./viu-cli –load <file-path>". Multiple files are selected using simple regex expressions like "./path/file-prefix.*". The output results will look like:

Students (total=*, loaded=*)
Institutions (total=*, loaded=*)
Courses (total=*, loaded=*)
Contingents (total=*, loaded=*)
Applications (total=*, loaded=*)
Results (total=*, loaded=*)

Results contains the total of processed items for the corresponding entities and the ones that where loaded. The operation is idempotent, meaning it can be applied multiple times on the same file. When entities already exist in the DB they are not accounted for the "loaded" counter. Warning logs can appear, signaling the lack of some data fields, e.g: "Non existent grade: (uid=137*531-BRUNO MIGUEL MORAIS CACEIRO, application=1)".

C. UI

In the home page, besides the overview visualization of the "accepted applications per year", it is also possible to apply a set of global filters to all d3js visualizations. The filter preferences are stored in the domain cookies, adding persistence across page reloads and limiting the loaded data from the query service, improving page load performance. Provided filters are: year range and course selection. All d3js visualizations have local filters that are applied after the data is collected from the server, these are located at the right section of the page.

IV. Visualizations

The following sections describe the available visualizations and the insights provided by them.

A. Placed (by Year)

Example provided in Fig. 2, it's located at the **Demand** menu, showing the number of accepted students in each option for the pre-selected courses and the selected year.

Insights: In here it's possible to select courses per year and see what was the most popular first choice compared to the total number of positions occupied. Option 1 is normally occupied by the best students, and a good volume for this shows the preference of those students.

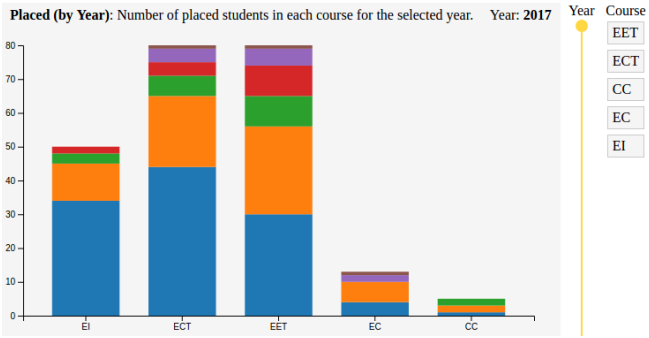

Fig. 2. Example of the Placed (by Year) visualization.

B. Demand vs Placed (by Course)

Example provided in Fig. 3, it can be reached from the **Demand** menu when selecting one course of the pre-selected list. It shows the relation between the demand (students

[6] https://d3js.org/
[7] https://developers.google.com/chart/
[8] https://material.angular.io/
[9] https://gradle.org/
[10] https://www.npmjs.com/
[11] https://cli.angular.io/

applications for each year and option) with the side bar, indicating the effective number of accepted students for that year.

Insights: In here it's possible to compare the demand in each selected option and compare it to the actual occupation. We can see the progress of these results through the years. On the years that the demand is lower relative to the occupation, it's an indicator to open more positions on the course.

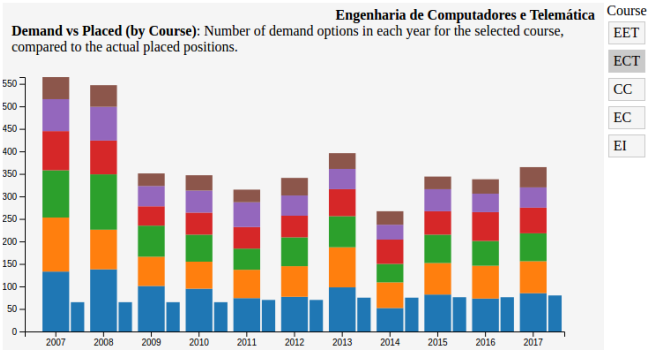

Fig. 3. Example of the Demand vs Placed (by Course) visualization.

C. Max, Min and Avg Grades

Example provided in Fig. 4, it's located at the **Grades** menu, showing the maximum, minimum and average grades per year of the accepted students on the courses. Local filters can be applied for the pre-selected courses.

Insights: The positions of the min, max and avg grades compared to each course can give an indicator on where the best students are placed. We can see this evolution throughout the years.

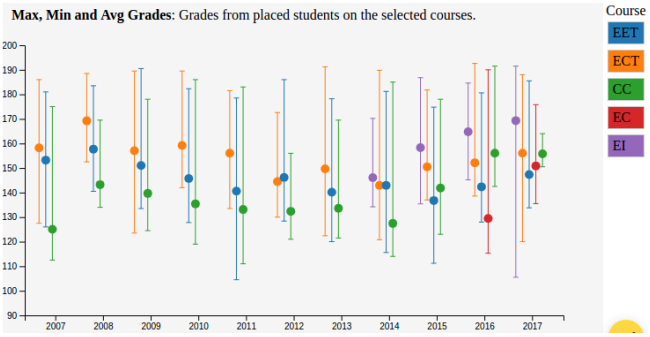

Fig. 4. Example of the Max, Min and Avg Grades visualization.

D. Grades Curves

Example provided in Fig. 5, it can be reached from the **Grades** menu when clicking the button "Show grade curves" on the right bottom side. It shows the ordered grades from all the accepted students on the selected courses and year.

Insights: In here the students grades are detailed for the respective year. We can see the grades spread throughout the students volume. It's interesting to see, sometimes sharp dips in the lines where the grades suddenly drop. The curve can give nice cutoff numbers for the open positions, where we can balance the students grades or select the best ones.

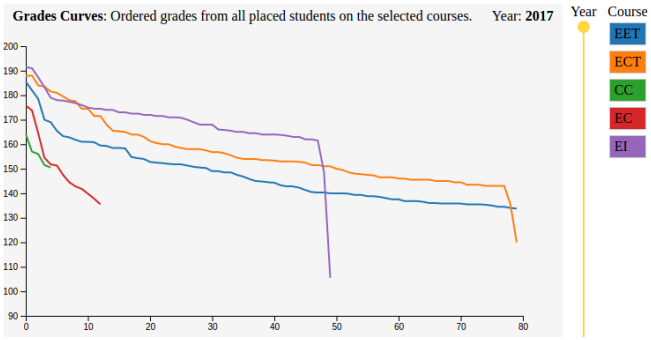

Fig. 5. Example of the Grades Curves visualization.

V. Heuristic Evaluation

As explained by other authors [3], different usability evaluation methods can be applied. **Nielsen** applies a general approach evaluation, where **Zuk and Carpendale** are suited for a visual and cognitive evaluation. **Forsell and Johanson** was constructed from Nilsen methods. Formal heuristic evaluations are performed by a group of experts; in this case it's performed an informal evaluation with only one evaluator. This enables rapid and iterative development while still being guided by well-founded principles, it can reveal the presence of problems, but does not prove their absence. The Forsell [4] method was applied for a overall evaluation of the system, including a grade rate for each heuristics:

0 → No problems detected
1 → Cosmetic problems only
2 → Minor usability problems
3 → Major usability problems
4 → Usability catastrophe

Information coding: *Perception of information is directly dependent on the mapping of data elements to visual objects. This should be enhanced by using realistic characteristics/techniques or the use of additional symbols.*

The Max, Min and Avg Grades graph accurately represents the data. The available bar graphs are adequate to represent the volume of students; absolute values are generally used, although some percentages would be useful. **Grade 1**.

Minimal actions: *Concerns workload with respect to the number of actions necessary to accomplish a goal or a task.*

In general, the actions are directly available by one-click or mouse over the component. Navigation between related visualizations should be improved. **Grade 2**.

Flexibility: *Flexibility is reflected in the number of possible ways of achieving a given goal. It refers to the means available to customization in order to take into account working strategies, habits and task requirements.*

The application is oriented to provide the data representations in a direct and fast way. Flexibility is not a requirement. **Grade 0**.

Orientation and help: *Functions like support to control levels of details, redo/undo of actions and representing additional information.*

Documentation is overall poor, but it's possible to control detail levels by filtering years and courses. Visualizations should have more information about the presented. **Grade 3**.

Spatial organization: *Concerns users orientation in the information space, the distribution of elements in the layout, precision and legibility, efficiency in space usage and distortion of visual elements.*

The layout of the visualizations and filters was carefully selected, providing efficiency and precision. Legibility may be affected in the way the resumed information is presented. **Grade 1**.

Consistency: *Refers to the way design choices are maintained in similar contexts, and are different when applied to different contexts.*

Consistency is maintained across visualizations. Descriptions and filters are localized in the same places and course selection works in similar way. **Grade 0**.

Recognition rather than recall: *The user should not have to memorize a lot of information to carry out tasks.*

The presentations are not overloaded with data, although sometimes the information may not be immediately recognized. E.g in the Demand vs Placed (by Course), the sidebar may not be recognized as the accepted positions. The global filters are recognized immediately on top of the home page. **Grade 1**.

Prompting: *Refers to all means that help to know all alternatives when several actions are possible depending on the contexts.*

Navigation and links are very direct, with just one alternative and without needing much prompt. However due to limitations on the components library, some inputs could be improved, e.g the global filter for the year range should be a single range component instead of two. **Grade 1**.

Remove the extraneous: *Concerns whether any extra information can be a distraction and take the eye away from seeing the data or making comparisons.*

The amount of information is small. The interface is not cluttered with spread information. **Grade 0**.

Data set reduction: *Concerns provided features for reducing a data set, their efficiency and ease of use.*

The dataset can be pre-selected in the global filters, focusing the search space. **Grade 0**.

VI. Conclusions

A small set of visualizations were provided, with a good foundation to extend if other ideas are suggested. Although the heuristic evaluation was made by the same person that developed the application, when focused on the evaluation task having very well defined metrics, some issues were detected. Those were not recognized in the development phase, proving the usefulness of the heuristics even when applied by a single evaluator.

References

[1] R. Angles and C. Gutierrez, "Survey of graph database models," *ACM Computing Surveys (CSUR)*, vol. 40, no. 1, p. 1, 2008.

[2] S. Krishnamurthy, "The value of schema-less databases," https://blog.couchbase.com/the-value-of-schema-less-databases/, 2016, accessed January 18, 2018.

[3] B. S. Santos, B. Q. Ferreira, and P. Dias, "Heuristic evaluation in information visualization using three sets of heuristics: an exploratory study," in *International Conference on Human-Computer Interaction*. Springer, 2015, pp. 259–270.

[4] C. Forsell and J. Johansson, "An heuristic set for evaluation in information visualization," in *Proceedings of the International Conference on Advanced Visual Interfaces*. ACM, 2010, pp. 199–206.