Estonian Information Technology College

Deploying IT Infrastructure Solutions

Üllar Seerme, Olga Trikk, Georgios Alexakis, Alexandros Kantas, Mindaugas Gedaminskas, Pijus Akelis

Visualization of Municipality's Budget

Documentation

Tallinn 2014

Table of Contents

1. Introduction

This is the final documentation/overview for the "Visualization of Municipality's Budget" project, which was a part of the "Deploying IT Infrastructure Solutions" intensive programme during 31.03.2014 - 11.04.2014 at the Estonian Information Technology College. The team was comprised of international students with participants being from Estonia, Greece and Lithuania.

The goal of the project was to create a prototype tool for the City of Tartu, which enables ordinary citizens to have a visual overview of their municipality's budget. This would also allow for these citizens to manipulate the data to come up with their own ideas for a suitable budget. An example of http://meieraha.eu was given as a possible solution. In terms of usability for the back-end, the administrator or municipality worker should have a simple interface, which allows him or her to easily insert data.

2. Requirements for us and the reader

The necessary requirements were given by the client in a document distributed before the beginning of the project. It stated that the prototype should be easy to implement in other municipalities, be compatible with HTML5 standards, support older versions of modern browsers like Firefox, Chrome, Internet Explorer, Safari and Opera. The site should endure 3000 pages an hour as its load. In terms of software the test platform needs to be running FreeBSD version 10 as its operating system, Apache web server version 2.4, PHP version 5.5, MySQL version 5.6 and Drupal version 8 as the preferred Content Management System. During the course of the project version 8 of Drupal was actually still in alpha.

To follow along with the documentation the reader should have a rudimentary grasp on programming as this is the key part of this project. There is no need to know the nitty-gritty about mainstream programming or scripting languages, but concepts such as functions, loops and if-statements are considered necessary. Anything else that is thought of as important will be explained in the documentation, but if something isn't then there aren't any terms that are too nebulous which can't be found with a quick search online.

It should be mentioned that this documentation/overview cannot be used to fully re-create our results. Reason being that due to our own negligence we did not document everything from the beginning and what we did document wasn't in enough detail to provide an accurate guide. However, if the reader wishes to learn more about our project, then he or she can continue.

3. Things that should be done

Problems we wanted to solve were that the whole solution should be lightweight in terms of code, so it wouldn't matter what the hardware running it would be. It should also be easier to use than the example site that was mentioned by the client. In terms of the design, it should be better looking, more space efficient in terms of resizing and comparing multiple revenues or expenditures. Data insertion should be easy for the average computer user in the municipality and for the administrator. There should be an option to save your version of the budget as either an image or a .csv file, which can then be easily imported to other data manipulation software.

4. Initial ideas

Even though the project itself started on Monday, 31st of March, we started with the actual project on the next day by brainstorming ideas based on the predistributed client info. A consensus was reached that it would be more appropriate to come up with a new design and the example solution of http://meieraha.eu was not the most user friendly experience because the site is not responsive towards different resolutions, the whole layout isn't effective as it wastes valuable space, it's hard to see precise comparison between different categories of revenue or expenditure whilst still having the ability to resize and move around.

During the brainstorming session we concluded that our version should either be a physics-based scale system where the user can drag & drop certain objects that affect a scale, which shows the balance between where the money comes from and where the money goes. The second idea was a simpler bar graph that shows bigger main categories, which can be manipulated in terms of width and ordered on a vertical axis, and can expand into smaller sub-categories to see finer detail. A fair amount of time was spent by looking at different visual styles for the final solution, and the main source for inspiration was a site called Dribbble, which is a community for graphic designers who can display their works publicly there. Searching for the word "graph" showed us a lot of good looking designs, but given the amount of time we had, and our lack of previous experience, it was clear to everyone that designs such as those were hard to do. A simple mock design was agreed upon: http://imgur.com/kD8XFM9

5. Laying out the necessary jobs

After agreeing on the design we paired the team members and assigned specific jobs. Due to the size of the team overall and the small windows of time we had to complete the assignment, the jobs weren't fixed throughout the project, but rather switched and moved around according to what needed to be done at the moment. The jobs we came up with at the beginning were that Alexandros and Olga were in charge of coming up with the database schema, Georgios and Mindaugas were in charge of finding a suitable JavaScript framework for the sliders because writing anything purely from scratch would be a huge time sink, finally Pijus and Üllar were in charge of both design and front- or backend.

5.1 Database design and implementation

Based on the Excel workbook given to us by the client, Alexandros and Olga started to sketch out the design of the database. The Excel workbook was nothing more than two columns with one having the category of revenue or expenditure and the other having the appropriate value in euros. Some time was spent on deciding whether to normalize the database because then further additions to it would be greatly simplified, but since it was not an actual requirement and it would have demanded more time, we decided not to go ahead with it.

Instead of manually inserting the data from the Excel workbook to the MySQL database, Olga modified the workbook by grouping smaller sub-categories together. This saves us both physical space on the screen by having less categories displayed and the database itself is more compact. Once the workbook was modified, it could be saved as a .csv file and then imported from phpMyAdmin to populate the database and its tables with data. Once a file has been chosen for importing, it's possible to choose which columns to use.

In a working scenario the user would query the database every time he or she loads the site, which would be too intensive for the database engine and wouldn't make much sense because the data citizens can manipulate doesn't change that often. Maybe once or twice a year.

Because of that it was suggested that we query the database with PHP only when changes have been made, after which we load them to static JSON objects. Those objects, which are basically arrays for categories and their respective values, can then be looped through to represent them in the site.

5.2 JavaScript frameworks

As mentioned earlier, we needed to find some sort of framework which would do the heavy lifting for us, so we can focus on other things. During the inital stages we considered many JavaScript frameworks, such as: ChartJS, Rickshaw, Highcharts, CanvasJS, YUI Library, MorrisJS and Raphaël. In the end we decided to not choose a framework for charts, but rather use something called noUiSlider, which is a minimal range slider. Reason being that it's lightweight, supports multiple devices and is tested on several browsers, and it was easier to build around our design.

Based on the mock-up that was created for the overall site, Mindaugas and Georgios moved forward with fleshing out the implementation of the sliders. The idea was still same for the most part, but instead of having a single slider down at the bottom, each individual bar would be expandable by width and also by height if the user double-clicks the original slider, and there are more sub-categories to display. Those sub-categories themselves would be expandable as well.

5.3 Back-end

Once the database design was finished, it was possible to start creating the back-end in terms of a login panel for both the administrator and municipal worker who needs to insert data. This was done using PHP5 for functionality, HTML5 standards for representation and CSS for styling. All of those together describe a user interface, which behaves in a CRUD-like manner. Meaning that data can be Created, Read, Updated and Deleted. The underlying PHP code facilitates a connection between the MySQL database and a proprietary extension by Microsoft and Sybase called Transact-SQL was used to supplement SQL.

This webpage is not a final project. Some functions are not working properly. Webpage is split into two parts: front-end (HTML5) and back-end(PHP5, JSON). All data and values are stored in MySQL database.

In order to open a project all files from folder “visual” should be uploaded into the server. In “data” file there is a .sql file, which contains the code for the database. This query should be pasted into MySQL. Database will be created. Third step – open “visual/admin/includes” and find “connect.php” file. Open it with text editor. This file contains an address to database. It should be changed to one webhoster provides. To open a webpage it should be reached as index.html (example: visualization.com/index.html)

Admin panel: to update, delete or insert data an administration panel is created. It can be reached by adding /admin to url (example: visualization.com/admin). In order to log in user have to enter a username and password. Provided account username: visualadmin. Password: visualfxadmin. If administrator wants to change information about user logins, he can do that directly in database in the table “user_id”. Password is encrypted with MD5 algorithm (key1 is “salt” and key2 is “pepper”). If administrator wants to remove encryption, he have to remove a md5 function in login.php file line54.

Admin panel user guide:

- Login in with provided or newly created username and password
- Choose income or outcome taxes in navigation bar
- Choose year
- Insert, update, remove values from different taxes by entering a number and pressing insert
- All inserted data are preliminary and not visible in final website until an administrator confirms it.
- To confirm values users have to open “Add” page from navigation bar . Choose a year and press Confirm. Now values are inserted into json file and sent to the final website.
- In “add” page you can see a textbox which says “insert a year”. Here user can insert a new year, then choose it in income or outcome pages where he can manipuled with new year values.

What needs to be done:

- Data validation in admin panel
- Fixed reading from JSON. There are problems with inputing data from JSON file to final website. Not all sliders read correct values
- Added sliders for all different taxes (there are 5 income categories which contains about 4 subcategories each and there are 9 outcome categories which contains about 5 subcategories each)

- Database must be normalized
- Added a option to choose between years so a user can choose which year taxes values he wants to see on sliders (everytime an admin confirms a year, a JSON file is generated for each year confirmed, so a front end page should have a function which reads all those years and lets to choose between them).

5.4 Setting up the test platform

What follows is a rough guide to setting up a virtual machine with the required software, aside from Drupal version 8 Content Management System, which was skipped over due to time constraints. More focus was put on functionality.

5.4.1 Virtual Machine

As per our requirements we needed to set up an environment with FreeBSD version 10.0 with amd64(x86-64, x64) platform, Apache web server version 2.4.9, PHP version 5.5, MySQL version 5.6 and Drupal version 8. This was accomplished by creating a virtual machine on **Ubuntu 13.04** using **VirtualBox 4.2.10**. Once VirtualBox is installed, the user can create a new virtual machine by clicking **New** and inserting an appropriate name. From there a **Type** should be chosen, but if the user inserts a name that indicates the operating system, then the application chooses the type itself. Next, same goes for the **Version**, which should either be 32-bit (default) or **64-bit (our selection)** according to the image file being used. Choosing the amount of memory used is dependent on the applications the server will run. In our case we chose **2GB of memory**. When asked to either use an existing or create a new virtual hard disk drive, we created a new one using **VirtualBox Disk Image** file type. Storage on the physical hard drive was **dynamically allocated** to use physical hard drive space efficiently. For the size of the virtual hard disk we chose 2GB, which should be enough for the applications that we are running. Once the virtual machine is set up, the user needs to go into the Settings of the VM and go to Storage, and under the Storage Tree choose the empty drive. Once that's done, it's possible to **attach the .iso** of the FreeBSD operating system to the CD/DVD drive. If that has been done, it is now possible to boot up the machine.

Since this documentation will not be maintained in the long run, it is wise to follow along with the official documentation on installing FreeBSD, but we will give you our two cents as well: http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/bsdinstall-start.html

If the machine has been booted up, it will show a menu where pressing Enter is the default option and it's the one that should be chosen to follow along. After that, the FreeBSD installer opens up where Install should be chosen. For the Keymap, we chose the **default keymap**. Next, we have the option to choose the partitioning. We chose the **Guided Partioning Tool** and since any further partioning wasn't needed, it was possible to just continue with the installation. A password should be entered for the root user. A network interface should be chosen from the given list, after which it is possible to either Accept or Decline the use of IPv4, DHCP, IPv6 and SLAAC. We accepted all of them. In the **resolver configuration** we searched the itcollege.ee network. When given an option to choose Local or UTC, we chose "no" because we did not know whether the machine's CMOS clock was set to UTC. Region was chosen as Europe and country as Estonia. **System services** can be added by pressing Space on the desired service, we chose the sshd, ntpd and dumpdev. Adding any users to the install is optional. Now the user can apply the configuration and exit the installer. After rebooting the user can unmount the .iso file in the same location as it was earlier applied and then login using the credentials that were created (just "root" with the chosen password by default).

5.4.2 Apache 2.4 install

After finishing the installation the user should run "portsnap extract" and "portsnap fetch update", and after that "cd /usr/ports/ports-mgmt/pkg" and "make install clean".

1. cd /usr/ports/www/apache24
2. make install clean
3. Default options for install

A good guide to follow: http://www.cyberciti.biz/faq/freebsd-apache-web-server-tutorial/

5.4.3 MySQL 5.6 install

1. cd /usr/ports/databases/mysql55-server
2. make install clean
3. Default options for install

A good guide to follow:
http://www.cyberciti.biz/howto/cookbook/2006/03/install-mysql-server-using-binary.php

5.4.4 PHP 5.5 install

1. Cd /usr/ports/lang/php55
2. make install clean
3. Default options for install

A good guide to follow:
http://www.freebsdmadeeasy.com/tutorials/web-server/install-php-5-for-web-hosting.php

6. Conclusion

During the project we learned how to work together as a team which, given that it was a multicultural team, was even harder than it would be otherwise. Our work started out by brainstorming the predistributed client info and coming up with ideas for what we could do differently and possibly better. After agreeing on a design, work began as we started researching possible JavaScript frameworks and/or libraries to use. Meanwhile, we managed to flesh out the database design and implement it by simply re-saving the Excel workbook as a separate .csv file.

Once the database was set up, it was possible for the back-end work to begin by creating an admin panel for logging in and doing data insertion. While the back-end work was being done, front-end was going along swimmingly because we had a clear image of what we wanted to achieve.

All in all, we would consider our prototype as something that can be improved, but is generally moving in the right direction. Such a tool should be available in every government and municipality because it both gives a clearer overview of where the money is coming from and where it's going, and it incentivizes citizens to partake in the improvement of their local municipality or government.