

Evgenii Korepanov

DEVELOPMENT OF WEB INTERFACE FOR WORKING WITH TABULAR DATA

Bachelor's thesis
Degree Programme in Information Technology

2017

Author (authors)	Degree	Time
Evgenii Korepanov	Degree Programme in Information Technology	April 2017
Title Development of web interface for working with tabular data		54 pages 5 pages of appendices
Commissioned by Roman Denisov, Tpoint OY		
Supervisor Paula Posio		
<p data-bbox="142 716 280 751">Abstract</p> <p data-bbox="142 793 1451 940">The aim of this thesis is to investigate the process of migration of client-server applications into WEB basis. The example described in the thesis is intended to provide such a migration of client-server application, which is being developing as a service by Tpoint Oy company.</p> <p data-bbox="142 978 1451 1089">During this work, exact realization of client application was designed by the iterative method of developing, including prototyping and different techniques review. A justifying of finally selected method is presented in the text of this thesis.</p> <p data-bbox="142 1127 1451 1377">During the thesis work client-side specialized web application was designed, developed and implemented. This application is aimed to replace the solution used before, which is a native Microsoft Windows application based on MS Access. Functionality of old application was aimed to handle data of enterprise accounting system and all functions were planned to be realized in a new web application. This realization was partially fulfilled in this thesis as a working prototype of application that demonstrates concept of development, proves the viability of such a decision and possibility to develop the product up into full-featured.</p> <p data-bbox="142 1415 1451 1562">It is shown in the thesis that independent development of web components for a graphical user interface can have significant advantages over the use of ready solutions. This work serves as a start for development of universal user interface components for working with relative tabular data.</p>		
<p data-bbox="142 1598 302 1633">Keywords</p> <p data-bbox="142 1671 1451 1740">Client-server, Web development, Web interface, JavaScript, Web standards, Web Components, Coffeescript, Web application</p>		

CONTENTS

1	INTRODUCTION	5
2	GOALS	5
3	TRENDS OF WEB DEVELOPMENT	6
3.1	Operating systems and hardware diversity growth	6
3.2	Web application as universal solution	7
3.3	Adaptive design	8
3.4	Rich Internet Application as a Single Page Application - one for the price of three	9
3.5	Component approach as solution for web development hell	11
4	USED TECHNOLOGIES	13
4.1	Web components	13
4.2	Standards of Web Components	15
4.2.1	Custom Elements	15
4.2.2	Shadow DOM	16
4.2.3	HTML Templates	16
4.2.4	HTML Imports	17
4.3	BEM instead of shadow DOM	17
4.4	Event model	19
4.5	Data binding with D3js	19
4.6	Utilization of CoffeeScript as a cultivated version of JavaScript [20]	21
4.7	JQuery	22
4.8	Bootstrap	22
4.9	What was dropped as non-suitable	23
5	DESCRIPTION OF THE DEVELOPED APPLICATION	24
5.1	Hello-world example tag	24



South-Eastern Finland
University of Applied Sciences

5.2	System architecture	27
5.3	How it actually works	29
5.3.1	Main functionality	29
5.3.2	Configuration and customization.....	38
5.3.3	Mobile interface	43
5.3.4	Application work summary	48
6	SUMMARY	49
	REFERENCES	51
	APPENDIX	

1 INTRODUCTION

This thesis work is based on existing commercial software "Tpoint Logistics" [1] used for the warehouse, financial and logistics accounting. The existing solution is a native desktop application made using Microsoft Access as a client to Microsoft SQL Server.

Despite of the widespread presence of this approach to implementation of various accounting systems, the use of Microsoft Access has a number of significant disadvantages: platform dependence [2], inability to use Access in client-server architecture for connecting to SQL Server since version 2013 [3]. The scaling overall system functionality and implementation of new features to the product is also limited by the solution.

Impossibility to resolve these disadvantages required finding a replacement for the client part, which led to formulation of the goals of this thesis.

2 GOALS

The goals of this work were based on requirements for the application to have certain properties. Major property was to create cross-platform client application designed for both desktop and mobile devices. The application needs to be customizable for both program interface and functionality. The users were required to be able to work with software under their favourite operating system as well as to work outside of the office. The same trends can be observed among users of most accounting systems [4].

A new solution also needs to be easily modifiable and scalable to cover the needs of users that cannot already be satisfied with existing solution. Scaling overall system functionality and implementation of new features to the product is also limited by the current solution.

The list of application properties was combined based on set of goals. According to generalized requirements list, the application must be:

- web-based
- cross-platform
- desktop and mobile
- scalable
- customizable
- extendable
- lightweight.

Achieving mentioned goals can be divided into three phases:

1. development of the prototype that is ready to be extended: working demonstration of core functionality
2. development of a minimum viable product (MVP [5]): Navigation and CRUD (Create-Read-Update-Delete) to any data table
3. development of a full-featured product: ready replacement of existing solution

It is necessary to follow the trends in the software market for a delivery of a successful product. Next follows a review of discovered trends to business software.

3 TRENDS OF WEB DEVELOPMENT

3.1 Operating systems and hardware diversity growth

Extreme popularity of Microsoft Windows drops slowly but surely. Windows market share has decreased by 10 percent during the last eight years and it has a slow descending tendency [6], also see figure 51.

Diversity of used hardware platforms is growing, and the desktop platform is not the leader any more. The number of desktop platform users almost have not changed during the period of 2013-2015 but a 40% growth of digital media usage came from mobile and tablet devices users [7], also see figure 52

All of this causes demand for software to provide support for multiple operating systems as well as for multiple devices.

Due to huge growth of mobile devices popularity, mobile applications nowadays become more popular than desktop ones [8], also see figure 53. Particularly, time spent in mobile applications for business is about 40% of time spent for business on all application platforms and it is continuously growing [4], also see figure 54. Nowadays employees become more mobile. They can perform their work tasks outside of the office and use several devices including phones, desktop computers and tablets during a day [9]. According to Citrix mobile analytics report [9], the number of devices managed in the enterprise increased 72% from 2014 to 2015". That all causes the importance of mobile marketing in business application development.

3.2 Web application as universal solution

Web applications are a step forward compared with native applications when support for several runtime environments is required. The factors of these solutions are described next.

1) native application

Development is more expensive since every platform requires development and support of individual software versions. The cost of maintenance and updating becomes higher. Users may be using different application versions and there is a need to support many previously released versions and keep up with updates for all platforms. Native solutions for a chosen program, platform or equipment do not cover all devices or platforms that can be used to work with information.

2) web application

Web applications are much easier to maintain as it has common code base across multiple platforms. There is no need to think about backward application compatibility since there are no different application versions. This lowers maintenance costs. Users do not need to download, install and update the

application. The application can be released any time and updated dynamically or partly, as per the developer's preferences.

Web solution is the best option for ERP system user interface development as it works on all platforms since almost every device has a modern web browser. Another thing about web application is that it provides a convenient way for promoting and advertising. Potential users visiting promotional web site can get directly to the demo-version of a program.

3) Multi-platform development environments are not popular because they have problems of both native solution and web application. They do not cover the entire diversity of platforms and do not have all specific platform features implemented.

3.3 Adaptive design

More than half of Internet users visit websites with mobile devices and small screens [10], also see figure 35. This makes a web developer thinking about the functioning of the website not only on desktops but on mobile and touchscreen devices too.

Creation of two versions of web interface might be a decision but it doubles work of application support and development, which is not always cost-effective. The concept of adoptive design solves this problem. The site equally displays information in the most complete way, regardless of the type of screen and the size of the device. The content and colour scale do not change, only the form and the way to group information and navigation blocks of the site in the most convenient way for the user changes.

3.4 Rich Internet Application as a Single Page Application - one for the price of three

Today's web is not used as it was designed to be used. Originally, World Wide Web was designed as a network of web sites, where the web site is a resource consisting of one or more web pages with hypertext, multimedia (images, video, audio) and other types of static content. Today's web is more like a network of web applications. Web applications differ from web sites by having dynamic behaviour, meaning that the content of a web page changes without loading it from a web server, but by the program that brings interactivity to static content.

Earlier web applications worked by the principle of moving between pages by hyperlinks and sending forms to the server using a web browser. Such applications are built by client-server architecture with a thin client. HTML, being the markup language of documents and displayable by browsers, is ideal for this. The sequence of user actions is the constant sending of requests to the server. Server generates static web pages and returns them to the client. This approach is called Server-side HTML.

Method of HTML generation effectively works for simple applications. But performance falls down for complex application with rich functionality and many graphical elements. Method of HTML generation creates excess complexity for large structure of elements and makes interaction between elements difficult to implement. Server-side generation of HTML makes state of application stored at server that inevitably leads to cache invalidation problems:

"There are only two hard problems in Computer Science: cache invalidation and naming things"

— Phil Karlton [26]

Modern approach in development of web-applications is moving away from standard approach to an approach where it is standard to run such applications in web browser environment that has appearance and functionality similar to

desktop-applications. That class of programs is called Rich Internet Application (RIA) [27].

To provide RIA functionality software logic has to move from server to client environment. This means that static web pages are replaced by dynamic pages. When a page is static, it looks always the same, regardless of the user's actions. For example, the menu is organized by links to individual pages, not a drop-down list. By definition, dynamic web pages react to user actions and other events. For example, moving the mouse cursor over text leads to display of translation of the word.

Dynamics on web pages is implemented by using scripts written in JavaScript language that are executed by the browser. HTML elements support definition of event handlers. For example, event handler might be set to process "mouse click" event in the picture. Then if the user clicks on this picture, the handler defined for this event will be called. This allows for countless applications in dynamic web.

As the functionality of the application grows, so does the number of requests to the server. Each response from the server must be rendered by the client. This leads to an increase in the load for both the client and the server and an increase in the amount of information transmitted.

Developers can avoid this excessive exchange by dynamically loading only needed data at an appropriate moment without reloading entire page. For this, AJAX technology is used. AJAX stands for Asynchronous JavaScript and XML. It is a technology of requesting server without page reloading. Due to this, the response time is reduced, and interactivity of the web application becomes indistinguishable from desktop analogues.

AJAX technology is widely used for making Single Page Applications (SPA) [28]. SPA is a web application that uses single HTML document to display all content. Typically, SPA implements user interactivity by dynamically loading HTML, CSS

and JavaScript with AJAX queries. Applying AJAX at SPA improves development quality due to data being loaded only when needed. Compared to loading all page data at once, AJAX increases speed and decreases response time.

3.5 Component approach as solution for web development hell

In today's web, a number of tweaks that make possible the development of complex Web applications make the development process so difficult that Web frameworks had to appear to address such needs. Technologies, which have been created to help Web be developer-friendly, follow next. Third-party browser plug-ins such as Microsoft Silverlight, Java Applets, Adobe Acrobat Reader, Adobe Flash Player. JavaScript frameworks include Backbone.js, AngularJS, Ember.js, KnockoutJS, Dojo, Knockback.js, CanJS, Polymer, React, Mithril, Ampersand, Flight, Vue.js, Marionette.js, TroopJ, RequireJS, SocketStream, Firebase.

Plugins developed with Netscape Plugin API are not going to work in most browsers any more [11], [12]. In addition, Adobe Flash Player is replaced by HTML5 since all necessary capabilities of Flash Player are now implemented in HTML5 [13]. It means that correct operating of an application made using these techniques can not be guaranteed.

JavaScript frameworks are function libraries having their own rules of code organization. Their main idea is to create a set of high-level abstractions that hide details of web technologies and let developers work suitable level of abstraction. For example, a framework might let you to describe a web page using new inexistent tags and attributes that will be converted by framework to JavaScript objects manipulated by framework's functions.

The main problem of frameworks is that abstractions are not perfect. During overview of frameworks that initially looked like proper solution, it was found that with increasing complexity of the application, the number of nuances of the

framework is growing rapidly. Such nuances can not be explained without understanding source code of the framework.

Another point discovered is that frameworks provide a restricted way to make applications since they have their own rules. With any framework there comes a moment when requirements appear, that force changing framework behaviour. Inability to control third-party components (adjust them to specific needs) becomes then a real problem. For example, no suitable solutions were found that fully meet the requirements of simplicity, lightweight, customizable, and adaptive interface design.

As practice shows, usage of a framework brings the requirement to know how it works and, therefore, requires studying of its source codes. That is, the complexity of many tasks is not decreased, but increased by appearance of new technology stack that must be studied and maintained. Efforts for mastering the framework's universal components are comparable to efforts of writing of self-made particular solution. Self-made solution meets all requirements with no limitations brought by the external framework. Thus, it was decided to develop self-made solution without using frameworks at all.

When a web site or web page is described as complying with web standards, it usually means that the site or page has valid HTML, CSS and JavaScript. Development tools were chosen in a way not to destroy web standards (add an extra technology stack), but to take advantages from utilization of web standards.

Complex interfaces have long been unresolved problem in the world of web applications.

One of the techniques that are typically used in implementation of complex projects is the fragmentation of the project into possibly small parts that are loosely dependent on each other, which are easily developed separately. Fragmenting of the development process reduces its complexity. Software development then becomes easier and therefore better testable and reliable. The

main outcome of component approach is an ability to reuse components everywhere, so they become building blocks of complex application.

Previously browsers did not have native support for tools describing the solution of a problem with the help of abstractions. The task of representing a web application as set of self-made tags was typically solved by various JavaScript-frameworks. This created difficulties, as various program layers were added. Each of added layers was not part of standard. Accumulation of the experience and knowledge of the developer community led to the creation of a standard of Web Components in 2011 by Alex Russell with the following considerations:

"We need more observable ways to share loose meaning to feed the process of progress"

— Alex Russell, 2011 [14]

4 USED TECHNOLOGIES

4.1 Web components

"Web components" is the set of a standards enabling to create new DOM elements with its own properties, methods, encapsulated DOM and styles. Nowadays those standards continue to be developed and have different level of implementation in browsers [15], also see figure 56.

Currently Chromium and its derivatives implement latest version of W3C specification for Web Components.

For browsers that do not support Web Components standard, Google team has developed polyfills – JavaScript library that implements missing parts. To explain specifics of production usage of Web Components, the following ways to use Web Components can be considered:

1. Chromium-based browsers. Most popular browser in the world, Google Chrome, has mostly production ready realization of Web Components. Chrome is a web browser based on Chromium, developed by Google. First stable version was released in December 2008. Today it is the mostly used browser. According to statistics below it has more than half of web browsers market share [16] (figure 57), [17] (figure 58).
2. Polyfills. Polyfill is a JavaScript library that implements capabilities of modern browsers in older ones. For example if you are ingrained Internet Explorer 8 user and web developer using standard property `textContent` of DOM element at his site wants you to be able to visit the site he has to add there polyfill which adds required functionality into old browser using available functions of the browser.

Example source code for such polyfill is shown in figure 1.

```
(function() {  
  // check for support  
  if (document.documentElement.textContent === undefined) {  
    // define the property  
    Object.defineProperty(HTMLElement.prototype, "textContent", {  
      get: function() {  
        return this.innerText;  
      },  
      set: function(value) {  
        this.innerText = value;  
      }  
    });  
  }  
})();
```

Figure 1. Polyfill example

Example code shows the principle of polyfills' work. First it checks the browser for property support. If the property has `undefined` value then it is not supported. Second step then is to define it. In example it is done by `Object.defineProperty` function. `textContent` property of DOM element is now defined for IE8 as an alias for `innerText` which works properly in most cases.

Polyfills of some web components' features today are needed for Firefox, Safari, Edge and other web browsers. Library can be found on the official site of WebComponents [18].

This thesis work was done using Chromium without polyfills to provide faster, more comprehensive and simpler results.

4.2 Standards of Web Components

Web Components platform include four standards which are Custom Elements, Shadow DOM, HTML Templates and HTML Imports. Next follows the description of each standard.

4.2.1 Custom Elements

Custom Elements standard provides a way to create new HTML tags and describe their own properties, methods, constructor, DOM content, and CSS styles. The name of custom tag must consist of two words divided by dash.

Syntax: `<my-tag></my-tag>`

Custom elements have their lifecycle reactions, allowing to attach certain behaviour to different parts of the element's lifecycle such as "created", "attached to the document", "detached out of the document", "attribute added, changed, or removed".

To create new element the following call is used:

`document.registerElement(name, prototype:proto)`

- name: two words divided by dash;
- proto: prototype object for new element.

To have standard properties and methods, new tag needs to have object, inheriting from `HTMLElement` as a prototype:

`Object.create(HTMLElement.prototype)`. It is also possible to extend standard html tag by using as a prototype of new element object, similar to following:

`Object.create(HTMLButtonElement.prototype)`. In this case tag `<button>` will be extended.

4.2.2 Shadow DOM

Shadow DOM is a standard that lets to create inside of an element DOM tree that is separated from external document and is not accessible from outside without special methods. It has its own naming and styling scope.

Content of that type is created for the element by calling:

`elem.createShadowRoot()`

which is then accessible as a property:

`elem.shadowRoot`

The initial content of element is then hided and shadow DOM is displayed instead. Specialized tag `<content>` can be added inside shadow DOM to specify the place to display the initial content.

4.2.3 HTML Templates

HTML tag `<template id="t"></template>` is intended to store a markup template. It is not displayed by default but is used for deferred rendering. Its content can be inserted into document many times with JavaScript and then manipulate as an essence belonging to instance of a component.

The following code assigns to `c` variable a `DocumentFragment` containing copy of template's content.

`c = document.querySelector('#t').content.cloneNode(true)`

4.2.4 HTML Imports

HTML Imports are a way to include and reuse HTML documents in other HTML documents. Imported documents called Imports are inserted as a part of main document with shared script and style spaces. It is needed to provide the full modularity of the application by splitting the whole code into files including HTML that was not able to be divided before since it has not been considered from a component approach perspective. Web Components standard supposes that main document can import files-definitions with all needed HTML, JS and CSS of components that can be then used.

Syntax:

```
<link rel="import" href="http://site.com/document.html">
```

Document loaded by `<link rel="import">` is handled, and its DOM is built but is not shown in the beginning. That DOM can be inserted later when needed.

4.3 BEM instead of shadow DOM

Shadow DOM has two major disadvantages. Firstly, it does not let to include external CSS files which decrease quality of code organisation. Secondly, shadow DOM has CSS scope that prevents using of CSS libraries such as Bootstrap. Therefore, this thesis uses BEM methodology for CSS to overcome shadow DOM obstacles.

BEM (Block, Element, and Modifier) is a component approach to web development. Its main idea is to divide interface into independent reusable parts. The methodology is based on 3 main notions that are represented by class attributes in HTML. Independence of blocks is achieved by naming scope. Names are words divided by dashes: "`name-of-block`", "`name-of-element`", "`name-of-modifier`". Using of id and tag name in selectors is not recommended. 3 main notions explained:

- Block. Block is a functionally-independent component that can be reused. It is named by noun like "button" or "menu". Blocks can be nested to each other for any level. Example is shown in figure 2.

```
<div class="menu">
  <div class="button"></div>
</div>
```

Figure 2. Blocks in BEM methodology

- Element. Element is a part of block and it cannot be used separately. Its name is also a noun (ex. "item", "text"). Elements can be nested but they must be part of element and not another block. Name of block sets the naming scope that guarantees that element is dependent of block: "**block-name__element-name**". Example of blocks having elements is shown in figure 3.

```
<div class="menu">
  <div class="menu__item">
    <div class="menu__content-of-input">
      <div class="button">
        <div class="button__text"></div>
      </div>
    </div>
  </div>
</div>
```

Figure 3. Elements in BEM methodology.

The HTML above has BEM-structure in CSS shown in figure 4.

```
.menu {}
.menu__item {}
.menu__content-of-input {}
.button {}
.button__text {}
```

Figure 4. BEM structure in CSS

- Modifier. Is an entity that describes the state, outlook or behaviour of block or element and it has an appropriate name. Name of modifier also includes full hierarchy coming from block: "**block-name__modifier-name**" or "**block-name__element-name__modifier-name**". In HTML class modifier name is

used together with block or modifier name and must not be used separately:

```
<div class="button button_disabled"></div>
```

```
<div class="menu__item menu__item_selected"></div>
```

Modifier can be of boolean type. When being presented in class attribute it means true-value.

Also it can be of type key-value. Then value is specified in name: "block-name_modifier-name_modifier-value" or "block-name__element-name_modifier-name_modifier-value"

4.4 Event model

Browser event model was selected to organize interaction of application parts.

Event model is based on two components, event generators and handlers.

Handler is a function or method that can be called. Generator fires an event and handlers subscribed on it are called.

It is easier to instruct modules to generate certain events than to link module handlers to each other, because in first case modules are loosely related that greatly simplifies making communication API. So, components do not know anything about each other. They have methods that fire events. To provide components interconnection, "router" in main script catches event of one component and then calls proper methods of other components. Components do not call each other's' methods directly.

4.5 Data binding with D3js

Data binding is a technique that binds business logic data source with user interface elements. It means that when data value changes, elements bound to the data automatically reflect changes. Also, when outer representation of data in interface control element is affected it is reflected in underlying data.

D3js standing for Data Driven Documents is a JavaScript library for data handling and visualisation in a web. It provides convenient methods for working with data arrays and creating DOM elements. D3js in the project is used for one-way data

binding - interface reflects data changes. Data input values are bound to elements in the DOM so that values are referenced to elements with applying mapping rules.

Data input for D3js is represented by array of any type including multilevel or JSON. DOM elements are represented by Selection. It is an object returned by `d3.select(css-selector)` or `d3.selectAll(css-selector)` function that contains the set of nodes corresponding to CSS-selector. The library realizes fluent interface approach which enables to create call-chains. Each method in the chain is called on the object returned by previous method.

The following call,

```
d3.select('div').selectAll('p').data(["first", "second", third])
```

first creates Selection of paragraphs inside the first div-element in the document. Then, last method `.data()` called on Selection returns Data-Join. It is a draft for future DOM, which reflects the structure of given data. Describing the structures D3js employs declarative approach. The draft is used to bring data structure to the selected part of DOM by modifying, deleting and creating new elements with the help of mapping functions. Instance of data is stored inside DOM-structure and helps D3js to analyse changes in data and modify only few parts of DOM thus improving rendering performance.

JavaScript code, shown in figure 5 will generate HTML, shown in figure 6.

```
data_join = d3.select('li#my').selectAll('li').data([1, 2, 3])
data_join.enter().append('li') # adding li-elements if there is not enough for all data items
data_join.exit().remove() # removing extra elements
data_join.text(function(d){return d*d}) # changing existing elements
```

Figure 5. D3js rendering

will produce such HTML:

```

<li id='my'>
  <il>1</il>
  <il>4</il>
  <il>9</il>
</li>

```

Figure 6. Result of D3js rendering

No matter how many `<il>`-elements were inside ``, we will get as many of `<il>` as there were elements in data array. The only requirement is that `"li#my"` element must exist.

This technique helps to render all parts of interface throughout the application.

4.6 Utilization of CoffeeScript as a cultivated version of JavaScript [20]

All script files in the project were written with CoffeeScript language.

CoffeeScript is a language compileble into JavaScript. It lets **you** to write cleaner, understandable and consequently more supportable code. It changes general syntax constructions of JavaScript that are not always easily-writable and easily-readable into simpler syntax and adds new useful features. Also CoffeeScript uses indentation instead of tons of brackets. As common practice shows, this way is simpler. Therefore, it is used actively in the project. CoffeeScript files are included directly in the main HTML page and are run-time compiled during application loading. Basic examples are shown below.

Example of variables' declaration in CoffeeScript is shown in figure 7 and its equivalent in JavaScript is shown in figure 8.

```

age = 2
male = true
name = "Sergei"

```

Figure 7. Variables in CoffeeScript

```
var age = 2,
    male = true,
    name = "Sergei";
```

Figure 8. Variables in JavaScript

Example of function's declaration in CoffeeScript is shown in figure 9 and its equivalent in JavaScript is shown in figure 10.

```
say = (speech) ->
  alert speech
say "Hell oo world!"
```

Figure 9. Functions in CoffeeScript

```
var say = function(speech) {
  alert(speech);
};
say("Hell oo world!");
```

Figure 10. Function in JavaScript

4.7 JQuery

JQuery is a JavaScript library that focuses on the interaction of JavaScript and HTML. JQuery is used in the project as it has the set of ready convenient functions especially for events, selection arrays and html-elements handling. Utilization of JQuery is a well-established practice in web development and it provides useful tools that have cross-browser support. It is also a dependency for bootstrap and coffeescript libraries, used in the project.

4.8 Bootstrap

As a tool for making adaptive design, Bootstrap css library is used in the project. Bootstrap is a HTML+CSS open source framework, the set of tools and templates

for faster web applications development. It is dynamically updating and some its features might not work in old browsers.

Using Bootstrap in this thesis enabled a quick creation of first template, some functionality was added that let to create working prototype of application.

The main advantages of bootstrap are the following:

- saving time by using ready classes and design.
- allows to create adaptive design that is well displayed on devices of different types with no need to change markup.
- Ready design. Unified style of all elements and pages as a whole that displays well in different browsers (except old).
- Open source enables making changes of default behavior if needed.
- Popularity. Source code is examined by many developers. Factor of standardisation.
- Adaptivity. Design is ready to be displayed properly on different devices.

Main Bootstrap tools used in thesis:

- Typography. Ready design of paragraphs, headings, text alignment.
- Tables. Tools of tables designing.
- Navigation. Classes for designing of tabs, pages, menu and toolbars.
- Forms. Enables to create different forms: single or multi-line, with tips, validation, labels and drop-downs. Forms can be styled and highlighted by adding classes.
- Popups. Realizes dialog windows design.
- Buttons. Includes different type buttons design as well as button groups and drop-downs.

4.9 What was dropped as non-suitable

The following techniques were considered for using in the project but were not applied.

- Bootstrap Studio [21]. Was used to create the first HTML+CSS draft of the application but it did not find any use afterwards.
- JQuery UI [22]. The library of user interface components to make adaptive design. Bootstrap was preferred because it is based on HTML+CSS when JQuery UI has a lot of JavaScript realization. Due to this it is badly-

customizable and not so lightweight. Looks good only on mobile devices but the design is bad for desktop.

- W2UI [23]. One of the best ready solutions for assigned task. But it does not support mobile interface and has slow speed performance. It has lots of good features to make interface but is difficult to customize for specific needs which all the same inevitably arise.
- Model View Controller (MVC) [24]. The application was first developed with MVC pattern. With growth of application complexity but the pattern became insufficient as it divides development into 3 blocks only. Need of block component architecture appeared and instead of division into model view and controller application became to be consisting of blocks. Each block was done by as a component implementing MVC inside itself.
- Object Oriented Programming approach in JavaScript. As a way to split the application into component blocks object oriented programming was tried and small application have been implemented this way and it works now as a part of software package. But in the context of current application such architecture have been supplemented by WebComponents as they offer additional functionality and have reach built-in support of HTML features.
- Template processors. This technology was beforehand excluded from use-list as it adds additional level to the stack of technologies which needs to be developed and supported. After finding out that template processors can be replaced by D3JS, the last one was adopted.

5 DESCRIPTION OF THE DEVELOPED APPLICATION

5.1 Hello-world example tag

Every web component has its own HTML, CSS and CoffeeScript file that can be included in the main document. As a result of several trials, canonical example of web component was developed. It was named hello-world component. CSS files are optional for the component but normally always used. Hello-world component's files are listed below:

- hello-world.html
- hello-world.css
- hello-world.coffee

Minimum possible content of every file is presented in figures 11, 12, 13.

- hello-world.html is in figure 11.

```
<template>
  This text represents the content of hello-world element. All rights reserved.
</template>
<script>window.documentHelloWorld = document.currentScript.ownerDocument</script>
```

Figure 11. hello-world.html

HTML file contains initial markup for the component and should be included in the main document by HTML-import `<link rel="import" href="hello-world.html">`. The document then will have its own separated namespace and a small JavaScript code is used to define variable `documentHelloWorld` into global context (as a property of window object). Variable contains execution context of hello-world.html.

- hello-world.coffee is in figure 12.

```
HelloWorld =
  __proto__: HTMLElement.prototype
  createdCallback: ->
    @appendChild window.documentHelloWorld.querySelector('template').content.cloneNode(true)
  test: (text) ->
    test_elem = document.createElement('div')
    test_elem.innerHTML = text
    this.appendChild test_elem
  document.registerElement 'hello-world', prototype: HelloWorld
```

Figure 12. hello-world.coffee

The script in figure 12 does the following. First, it creates a (?) prototype object for the custom tag representing the component. Then the tag `<hello-world>` is registered into main document. Prototype object `HelloWorld` has the same prototype as `HTMLElement`. That brings standard functionality of html element to our new tag. Also, `HelloWorld` object sets own properties and methods for new tag. `createdCallback()` method is used as a constructor. It is a method realizing lifecycle reaction in Custom Elements standard and it is called when tag is added to the document or when it is registered. This constructor creates a copy of DOM from `<template>`-draft referred through the context of hello-world.html `window.documentHelloWorld`. Then constructor adds this DOM inside the `<hello-`

`world></hello-world>` tag. Element's DOM can be modified using initial data of the component before it is inserted into document. Example `test()` method of the tag modifies its internal DOM when called.

- `hello-world.css`

To follow BEM methodology CSS file should describe only classes and their names must start with 'hello-world' as shown in figure 13.

```
.hello-world__element-a{}
.hello-world__element-b{}
.hello-world__element-b_modifier{}
```

Figure 13. `hello-world.css`

Example of usage is following. Assume our main document contains code, listed in figure 14.

```
<body>
  <hello-world id='cool-tag'>
    <hello-world id='not-cool-tag'></hello-world>
  </hello-world>
</body>
```

Figure 14. Example of `hello-world` HTML tag usage

After new tag is registered, the DOM-tree will look as listed in figure 15.

```
<body>
  <hello-world id='cool-tag'>
    <hello-world id='not-cool-tag'>
      This text represents the content of hello-world element. All rights reserved.
    </hello-world>
    This text represents the content of hello-world element. All rights reserved.
  </hello-world>
</body>
```

Figure 15. Initial content of `hello-world` HTML tag

In the main script `test()` method of the component is called:

```
document.querySelector('#cool-tag').test('Hell oo world')
```

Then we will get the DOM-tree, listed in figure 16.

```

<body>
  <hello-world id='cool-tag'>
    <hello-world id='not-cool-tag'>
      This text represents the content of hello-world element. All rights reserved.
    </hello-world>
    This text represents the content of hello-world element. All rights reserved.
  <div>Hell oo world</div>
</hello-world>
</body>

```

Figure 16. Content of hello-world HTML tag modified.

Described basic web component is used as template for all project components.

5.2 System architecture

The daily routine of users of accounting systems consists of working with large number of records of the same type. The task of data handling often includes working with large set of records. Data sets can be represented in a spreadsheet view and are called 'tabular data'.

This work describes design decisions, development process and actual usage experience of generic tabular web interface.

To transfer structured data between frontend and backend JSON format is used. It is more easy-to-use and lightweight than XML and supported well in JavaScript. Frontend is a Single Page Application. All data transfer happens in the background without page reloading using AJAX technology.

Example of the methods provided by the server:

- `login(username, password)` - requests session id
- `get_filters_list(table)` - requests dictionary of type "filters_list" with unique name Table
- `select(table, offset, limit, sorting, filtering)` - requests content of table

Structure of 6 elements can cover various enterprise accounting systems due to only simple configuration is involved in customizing tables for different purposes.

Class diagram in Unified Modelling Language [25] is shown in figure 17. This diagram graphically describes the relationships of six components used in the system.

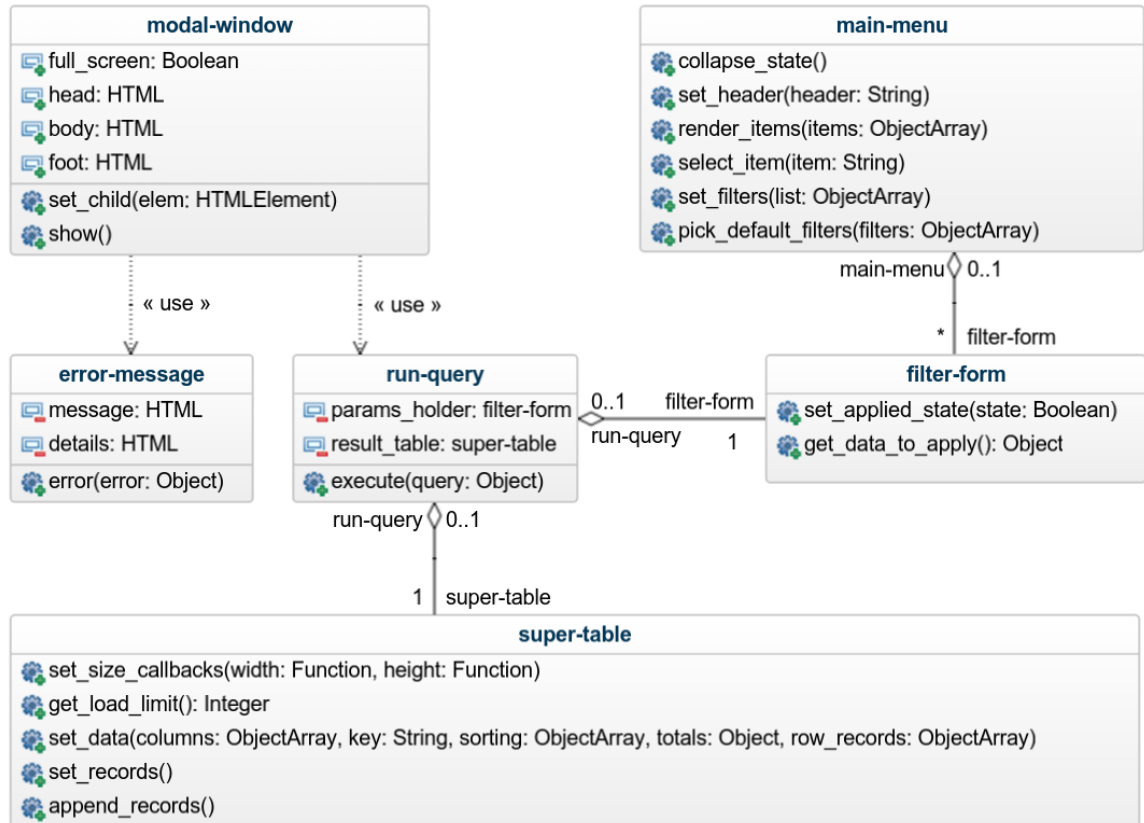


Figure 17. UML class diagram of system components

In application architecture, table is a unit providing scalability. Tabular component called 'super-table' can be reused many times for different data views and can be nested to bring ability to represent hierarchical data in different ways. Super-table component can be used as standalone or as a part of another component.

5.3 How it actually works

5.3.1 Main functionality

After logging in, a user sees tabular interface shown in figure 18. The interface consists of two basic parts: “main menu” located on the top and “data table” which is below the menu.

Row	Client	Job	Container	Unload
1	Nokia	038	YYYY8765432	ATR/0
2	BX Boshc	035	HLXU1234567	UL10
3	Samsung	042	MSCU9494941	UL15
4	Samsung	042	MSCU9494941	UL15
5	Samsung	044	MSCU7799779	JN8/0
6	Samsung	044	MSCU7799779	JN8/0
7	Samsung	048	MSCU8855220	UL22
Total: 178				

Figure 18. Tabular interface of program

In the example (Figure 18) there are 4 dropdown lists (menu groups) in the left part of main menu. Menu groups (such as “Workflow”, “Finances”, ...) are lists of data tables available to user. Application has access rights separation so that different user groups see different tables and menu groups. A menu with one group opened is shown in figure 19.

Row	Job
1	038
2	035
3	042
4	042
5	044

Figure 19. Main menu

The main action of menu items (such as “Charges”, “Bills, ...”) is switching between data tables. In addition to being able to switch between data tables, some menu items implement custom actions, such as "logout". These actions are set by JavaScript in configuration file. Action “logout” in the menu is shown in figure 20.

Row	Client
1	Nokia
2	BX Boshc

Navigation: Workflow ▾ Finances ▾ Handbooks ▾ Tools ▾ **Stock**

Tools Menu: Queries, Logout

Figure 20. Custom action "Logout" and name of table view "Stock"

The interface has UI component providing server-side filtering of data in table. It is represented by `<filter-form>` tag. Filters can have different number of parameters of different types. When a filter is picked, it can be applied by entering required parameters. An API-request is then sent and data for filters (name and list of parameters for each filter) comes from the server and it is used by UI component to render html-form. If the filter does not have parameters (bool-filter), it is applied directly when picked.

A configuration file can set default filter(s) for table.

In figure 21 there is the "Stock" table view with applied filter "in stock" by default. It has "cross" button to remove filtering condition. "Apply" button applies picked filters. A list of available filters for current table is displayed when "Filters" button is clicked. The opened list is shown in figure 21.

Handbooks ▾ Tools ▾ **Stock** Filters ▾ in stock ✕ Apply

Job	Container
038	YYYY8765432
035	HLXU1234567
042	MSCU9494941
042	MSCU9494941
044	MSCU7799779
044	MSCU7799779
048	MSCU8855220
057	TTTT6655443
057	TTTT6655443
063	

- all
- client
- container
- document
- in stock by weight
- job
- model
- period unload
- picked cars
- stored days
- unload
- unload qty
- vin

Figure 21. Filters

The filter "stored days" can be picked by clicking 4-th item from the bottom in the list shown in figure 21. Being picked filter is removed from the list and being unpicked it is returned back. HTML form named "stored days" will appear on the panel. Changes made from figure 21 after described actions are displayed in figure 22. Browser window is resized in figure 22 so that everything fits to the Thesis page.

Interface components have resize control to follow the browser window size.

Workflow ▾ Finances ▾ Handbooks ▾ Tools ▾ **Stock** Filters ▾ in stock ✕

stored days Prefix Days ✕ Apply Remove ✕

Row	Client	Job	Container	Unload
1	Nokia	038	YYYY8765432	ATR/08
2	BX Boshc	035	HLXU1234567	UL10
3	Samsung	042	MSCU9494941	UL15

Total: 178

Figure 22. "stored days" filter form with parameters

The appeared filter form shown in figure 22 has two input fields for filter parameters having parameter names in their placeholders. These parameters are represented at server side in SQL-query corresponding to the filter. Such architecture enables to create various custom filters only limited by SQL capabilities. The task of the filter component is to provide a convenient way for passing filter conditions and parameters.

Button "Remove" also appeared when picking "stored days" filter, as shown in figure 22. This button removes all picked filters and requests non-filtered table from the server. It was not there before picking "stored days" filter because only one filter "in stock" was set. "X" button in the right of filter removes that filter.

Since the filter form is displayed, parameters can be typed in. While they are typed, background of input fields is white. This means that data which currently is in input fields is not applied yet (have not been sent in request). State of form during typing process is shown in figure 23.

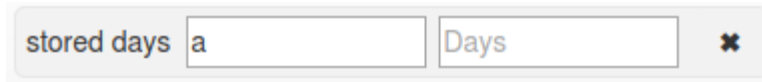


Figure 23. Filter form during typing of parameters

After user clicks "Apply" button or waits for 3 seconds filters are applied and changed table comes from server. It can be seen that total records amount has changed from 178 (in figure 22) to 7 (in figure 24). Background of filter parameters input fields is now grey. If input will be changed again, the background will become white until new values are applied. Table with parameterized filter applied is shown in figure 24.

Row	Client	Job	Container	Unload
1	Nokia	038	YYYY8765432	ATR/08/
2	New Client	0196		ARRIVA
3	New Client	0196		ARRIVA
Total: 7				

Figure 24. Table with parameterized filter applied

If "Remove" button (figure 24) is clicked, an unfiltered table comes from the server. The result is shown in figure 25. Now there are 359 records, more than there was in the beginning because default filter has gone too.

Row	Client	Job	Container	Unload	Date	Model
1	Nokia	038	YYYY8765432	ATR/08/1	14.03.2008	DVD PL
2	Nokia	038	YYYY8765432	ATR/08/1	14.03.2008	PRINTE
3	NEVAS	031	NNNN6565656	UL9	17.03.2008	CAR
4	NEVAS	031	NNNN6565656	UL9	17.03.2008	CAR
Total: 359						

Figure 25. Totally unfiltered table.

There is no "Apply" and "Remove" buttons in figure 25 because now they have no use. Declarative semantic of D3js can be clearly observed in the behaviour of "Apply" and "Remove" buttons. The following code sets the whole behaviour and outlook modifications:

```
controls = d3.select(elem).select('.main-menu_filters-controls')
buttons = controls.selectAll('button').data ->
  n = PickedFilters.length
  return [] if n is 0
  return ['Apply'] if n is 1
  return ['Apply', 'Remove &#10006;'] if n > 1
buttons.exit().classed('hidden', true)
buttons.classed('hidden', false)
buttons.html(d) -> d
controls.classed('btn-group', buttons[0].length > 1
```

Filter form supports parameters of different types. For example, if parameter is of type "date", user may enter one, two, or three valid numbers divided by anything. First number then represents day, second - month and third - year.

If there is only one number entered - it is interpreted to the day of current month, if there are two - to the date of current year. Three numbers is interpreted to the full date. Consider the filter "period", shown in figure 26.

The image shows a filter form for the parameter 'period'. It consists of two input fields: 'Date min' and 'Date max'. Both fields are currently empty. To the right of the 'Date max' field is a small 'x' icon used for clearing the field.

Figure 26. Filter "period" with parameters of type "date"

Date may be entered with the previously described ways. Examples of the input are shown in figure 27.

The image shows two examples of the 'period' filter form. In the first example, the 'Date min' field contains the number '7' and the 'Date max' field contains '8-9'. In the second example, the 'Date min' field contains '1-2-3' and the 'Date max' field contains '4-5-6'. In both examples, the 'Date max' field is highlighted with a blue border, and there is an 'x' icon to its right.

Figure 27. Examples of input for filter parameters of type "date"

When filter having parameters of type "date" is applied, the input is converted to proper date format. Examples of inputs from figure 27 converted to proper date format are shown in figure 28.

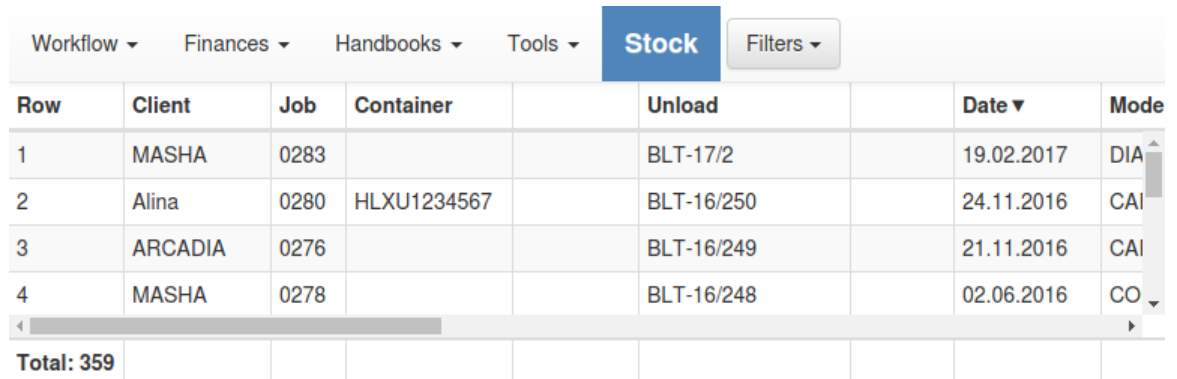
The image shows two examples of the 'period' filter form with the input converted to a proper date format. In the first example, the 'Date min' field contains '07.04.2017' and the 'Date max' field contains '08.09.2017'. In the second example, the 'Date min' field contains '01.02.2003' and the 'Date max' field contains '04.05.2006'. Both examples have an 'x' icon to the right of the 'Date max' field.

Figure 28. Input to filter parameters converted to proper format when applying

Background of the input fields is grey until the user starts input since current values are applied. There is a plan for future support of suggestion box, dropdown list and calendar to input parameters.

Interface also supports a server-side sorting. To sort the table by some column the header of the column must be clicked. Tables can be configured to have default sorting.

In figure 29, table is sorted by "date" column in descending order so that the latest items in the stock are displayed first. Down directed triangle indicates this fact.

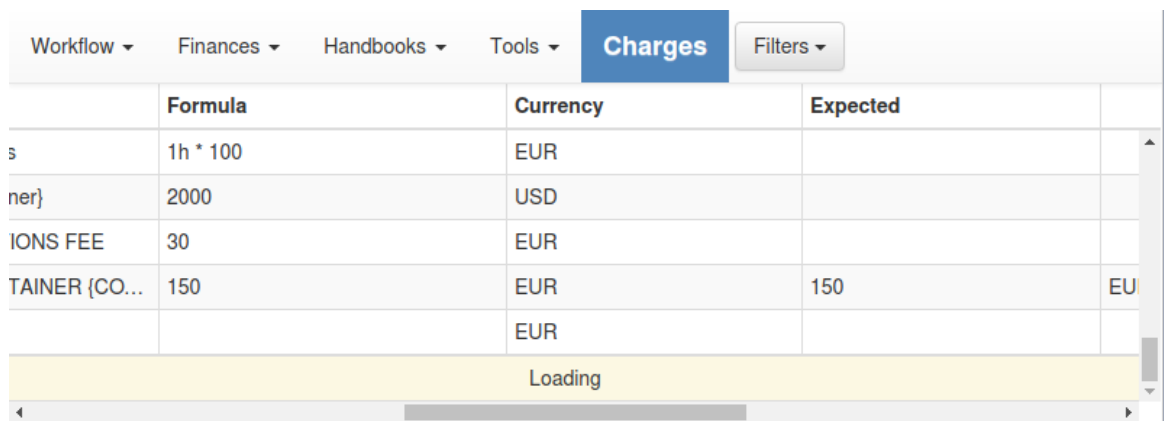


Row	Client	Job	Container	Unload	Date ▼	Mode
1	MASHA	0283		BLT-17/2	19.02.2017	DIA
2	Alina	0280	HLXU1234567	BLT-16/250	24.11.2016	CAI
3	ARCADIA	0276		BLT-16/249	21.11.2016	CAI
4	MASHA	0278		BLT-16/248	02.06.2016	CO
Total: 359						

Figure 29. Table "Stock" sorted by "Date" column in descending order

Client side sorting and filtering are not used in the program. It helps to keep data connectivity in server database and client interface synchronized, as well as to keep interface simpler by realizing data processing on the server with SQL. But client side sorting might be realized later.

Interface realizes lazy-loading, means that the table is loaded in parts during scrolling. Because sorting and filtering are server-side actions, table is loaded from the beginning when it is sorted or filtered. Last displayed row of the table indicates the process of loading as shown in figure 30.



Formula	Currency	Expected	Mode
1h * 100	EUR		
2000	USD		
IONS FEE	EUR		
CONTAINER (CO...	EUR	150	EU
	EUR		
Loading			

Figure 30. Loading indication

Last displayed row turns from “loading” (figure 30) to "end of table" (figure 31) when the whole table is loaded

Workflow ▾ Finances ▾ Handbooks ▾ Tools ▾ Trucks Filters ▾			
tr_plates	tr_vin	tr_country	tr_is_trailer
EMZ 434		FINLAND	<input type="checkbox"/>
AH 345 78	VNG09348TJDFIOJ0ER9	RUSSIA	<input type="checkbox"/>
HH 767	3948398438JI43	FINLAND	<input checked="" type="checkbox"/>
YVJ 980			<input type="checkbox"/>
EMB167		FINLAND	<input type="checkbox"/>
End of table			

Figure 31. "Whole table is loaded" indication

Table component of the interface realizes vertically frozen header and footer of the table. When the table is scrolled vertically, header and footer do not move and during horizontal scrolling they move synchronously with the table rows. HTML + CSS do not provide this functionality at all and even CSS “display:fixed” not does the job. So, it is done using several HTML + CSS + JavaScript tricks. Thus, five HTML table tags are used to display one table. Vertically and horizontally scrolled table is shown in figure 32.

	LD	ST Qty1	ST Qty2	ST Weight	ST Weight N	ST Volume
	1	1000	5	1500	2500	5
	2	5	250	3500	2650	11.25
	1	600	6	1500	1250	2.45
	1	500	6	1000	292	8.46
	2	500	3	1500	1125	12.5
		0		25		
	1	700		3500	3150	14
	1	100		2828	2000	5
		1		1900		
	65	16496	100407	382947.62		16193.2

Figure 32. Vertically and horizontally scrolled table

The columns of table have auto-size and they fit to width of content. Example is shown in figure 33.

Row	Client	Job	Job Date	Type	Marks	Consignor	Consignor ref
9	ALCANTARA	0270	22.03.2016				
10	LG	0272	10.03.2016	Container forwarding		CONSIGNOR	
11	LG PLUS	0273	10.03.2016	Container unloading			
12	LG PLUS	0274	10.03.2016	WH loading		CONSIGNOR	
13	Mexx	0271	02.03.2016	Container forwarding		CONSIGNOR	
14	LG	0270	10.03.2016				
Total: 290							

Figure 33. Columns with auto-width

By default, columns are configured to have maximum width of 60px. If content overflows the maximum width, it is wrapped and cut by dots as it happens in figure 34.

	Country	City	Postal Code	Address
	NEDERLAND	AB VENRAY	5804	Industrieterrein Smakterheide Maa...
	GB	LONDON	W1T 3NL	COLEGRAVE HOUSE 70 BERNE...
	RUSSIA	MOSCOW	129090	GILYAROVSKOGO STR.H.4.B.5
	RUSSIA	TVERSKAYA OBL. STARITSKYI R...	171361	D.LOUGOVAYA 67 A

Figure 34. Cell content overflow

To display the whole content of the cell it must be clicked. Then data is shown as multiline and cell content is highlighted and ready to be copied to clipboard (shown in figure 35).

	Country	City	Postal Code	Address
	NEDERLAND	AB VENRAY	5804	Industrieterrein Smakterheide Maashesesweg 89
	GB	LONDON	W1T 3NL	COLEGRAVE HOUSE 70 BERNE...
	RUSSIA	MOSCOW	129090	GILYAROVSKOGO STR.H.4.B.5

Figure 35. Display the whole content of the cell

The future plan assumes the opportunity to configure width, maximum width and other style parameter of table columns in the configuration file.

5.3.2 Configuration and customization

Configuration file revision follows next. There is a default config file that can be edited separately for different users. Config consists of two parts. First part defines menu items. For example, code in figure 36 describes menu groups: Workflow, Finances and so on.

```

w = window
F = 'finances'
w.config =
  menu: [
    {Workflow:[
      {table: 'q_jobs'}
      {table: 'q_containers'}
      ...
    ]}
    {Finances:[
      {table: 'q_charges'      , roles:[F]}
      {table: 'q_bills'      , roles:[F]}
      ...
    ]}
    ...
  ]

```

Figure 36. Configuration of menu

Each user group, shown in figure 36, has list of tables lying under the group. Tables have "role" property to differentiate access rights. Tables "q_charges", "q_bills" and so on are shown only to user of type "finances" which is defined by "F" variable. Others are shown for all users. Menu item can also have predefined action assigned, as listed in figure 37.

```

Tools:[
  {table: 'queries'}
  {action: 'logout', caption: 'Logout'}
]

```

Figure 37. Custom actions in menu

Click on "Logout" item in "Tools" group will end user's session.

Second part of the config describes table views. Code is then similar to listed in figure 38.

```

tables:
  q_jobs:
    caption: 'Jobs'
    key: 'job_id'
    sort:[
      {field:'job_date', direction:'desc'}
    ]
    columns:[
      {field:'rn'           , caption:'Row', sortable:false}
      {field:'client'      , caption:'Client'}
    ]
  ...

```

Figure 38. Tables configuration

Code in figure 38 sets name of table view displayed on the top of the page (`caption: 'Jobs'`). "job_id" is set as a key field of table. Also, column and direction is set for default sorting. Then columns of the table are described. "columns" property of table specifies for each column, which field of data coming from the server is shown in the column. For some columns, sorting can be disabled. Custom content with custom behaviour can be assigned to a table column by config. Have a look at configuration of "queries" data view, shown in figure 39.

```

queries:
  caption: 'Queries'
  filters: ['all']
  columns: [
    ...
    {field:'qry_name', caption:'Query'}
    {
      sortable:false
      render: '<button class="btn btn-default">run</button>'
      on_cell_click: (record) ->
        if event.target.closest('button')
          document.querySelector('#query').execute record
    }
  ]

```

Figure 39. Custom table content

In example, shown in figure 39, default filter "all" is set for table view. One of the columns is called "Query" and it uses "qry_name" field of data coming from server. Sorting is disabled for it.

"Render" property sets the custom content for every cell of the column. It can be a function of cell data returning html

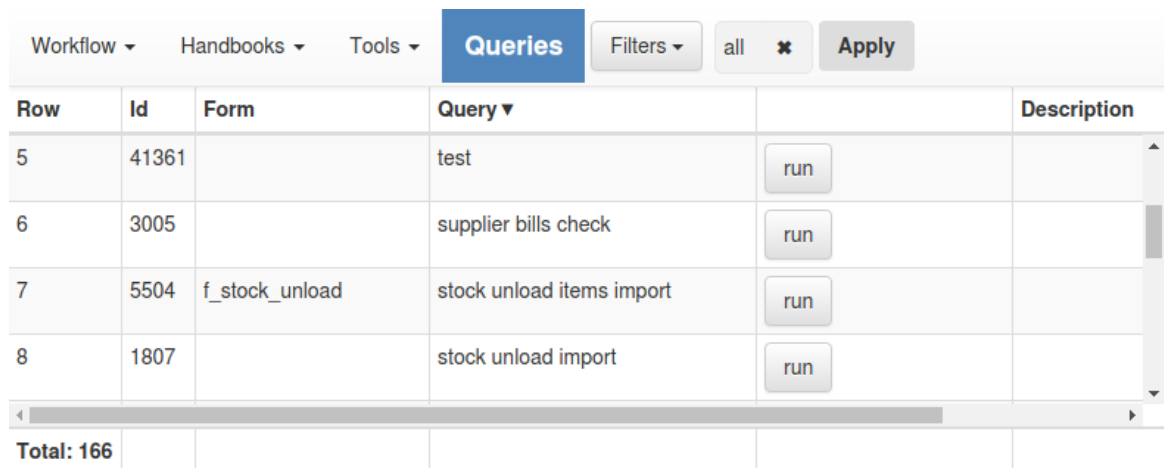
```
(render: (data) -> return '<p>Data is: ' + data + '</p>')
```

In our case data is not rendered and html is set as a string to put button named "run" into every cell.

"on_cell_click" property sets custom reaction on click for the cell. It is a function having data record as an argument. That is the record of data from server belonging to table row where the cell is. In the example the reaction is set so that when we click the button rendered inside the cell, query with the name specified in the record is called. Let us see what is the query.

Query is an SQL query stored in the server. When we click "run" button, we enter parameters and send API request. Parameters are used in SQL at the server and then result table is returned.

Next follows description of the way it works in the application. For example, "test" query shown in the table in figure 40.



Row	Id	Form	Query	Description
5	41361		test	
6	3005		supplier bills check	
7	5504	f_stock_unload	stock unload items import	
8	1807		stock unload import	

Total: 166

Figure 40. Stored SQL queries.

When user clicks "run" button, full-screen pop-up window is shown. This window provides to user a way of processing certain query. The same before described HTML-form component is used for entering the query processing parameters of table filters. Pop-up window for query processing in parameters entering phase is shown in figure 41.

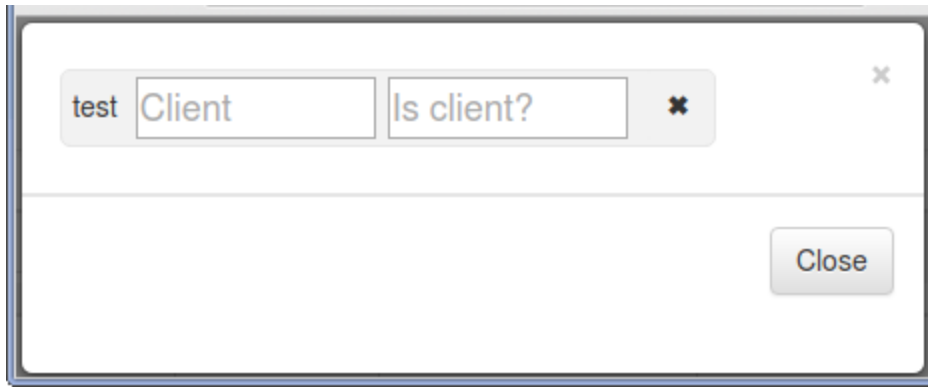


Figure 41. Pop-up window for query processing. Parameters entering phase.

When user enters parameters and presses enter or waits for 3 seconds, API request is sent and table result is returned. In the example, result contains two record-sets. They can be switched by clicking tabs "1" or "2". This result is shown in figure 42.

Row	cl_id	client	job_id	job	cnt_id	container	ul_id	unload	ul_ref_prefix
1	1	LG	151982	0140	74896	HHHU6546545	61707	JN8-10/557	JN8-10/
2	1	LG	376719	0237			142359	EU-15/1	EU-15/
3	1	LG	376719	0237			142359	EU-15/1	EU-15/
4	1	LG	376719	0237			142359	EU-15/1	EU-15/
5	1	LG	380414	0240			143835	BLT-15/2	BLT-15/
Total: 33									

Figure 42. Pop-up window with query result.

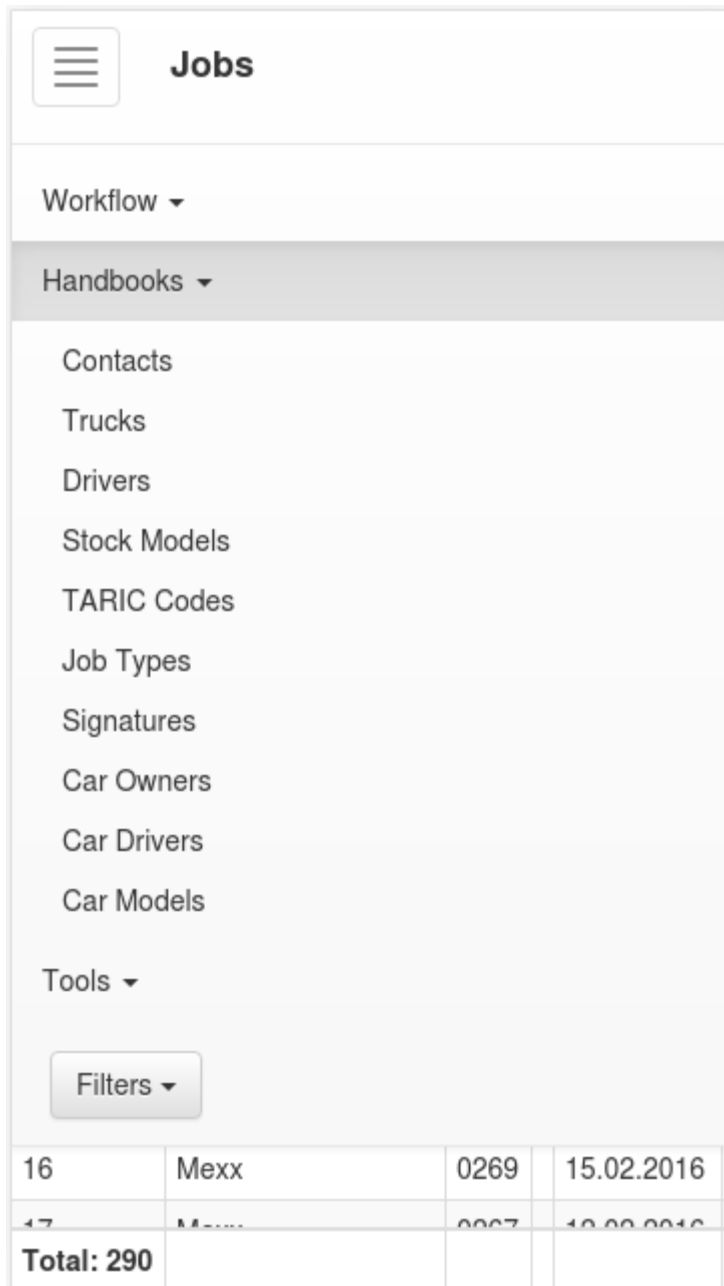
All functionality of application is also available for mobile devices. Example program view for smartphone is shown in figure 43.

5.3.3 Mobile interface

 Jobs			
Row	Client	Job	Job Date ▼
1	MASHA	0283	19.02.2017
2		0282	15.02.2017
3		0281	23.01.2017
4	Alina	0280	23.11.2016
5	New Client	0279	09.11.2016
6	MASHA	0278	02.06.2016
7	DANIIL	0277	10.05.2016
8	ARCADIA	0276	23.03.2016
9	ARCADIA	0275	22.03.2016
10	LG	0272	10.03.2016
11	LG PLUS	0273	10.03.2016
12	LG PLUS	0274	10.03.2016
13	Mexx	0271	02.03.2016
14	LG	0270	18.02.2016
15	Megant	0268	15.02.2016
16	Mexx	0269	15.02.2016
17	Mexx	0267	10.02.2016
Total: 290			

Figure 43. Example program view for smartphone.

On mobile devices menu is wrapped into sandwich. When menu is opened, it covers the table and there are hierarchical menu groups (figure 44).



16	Mexx	0269	15.02.2016
17	Mexx	0007	10.02.2016
Total: 290			

Figure 44. Mobile menu.

List of filters for the table is also observable on mobile devices by touching "Filters" button (figure 45).

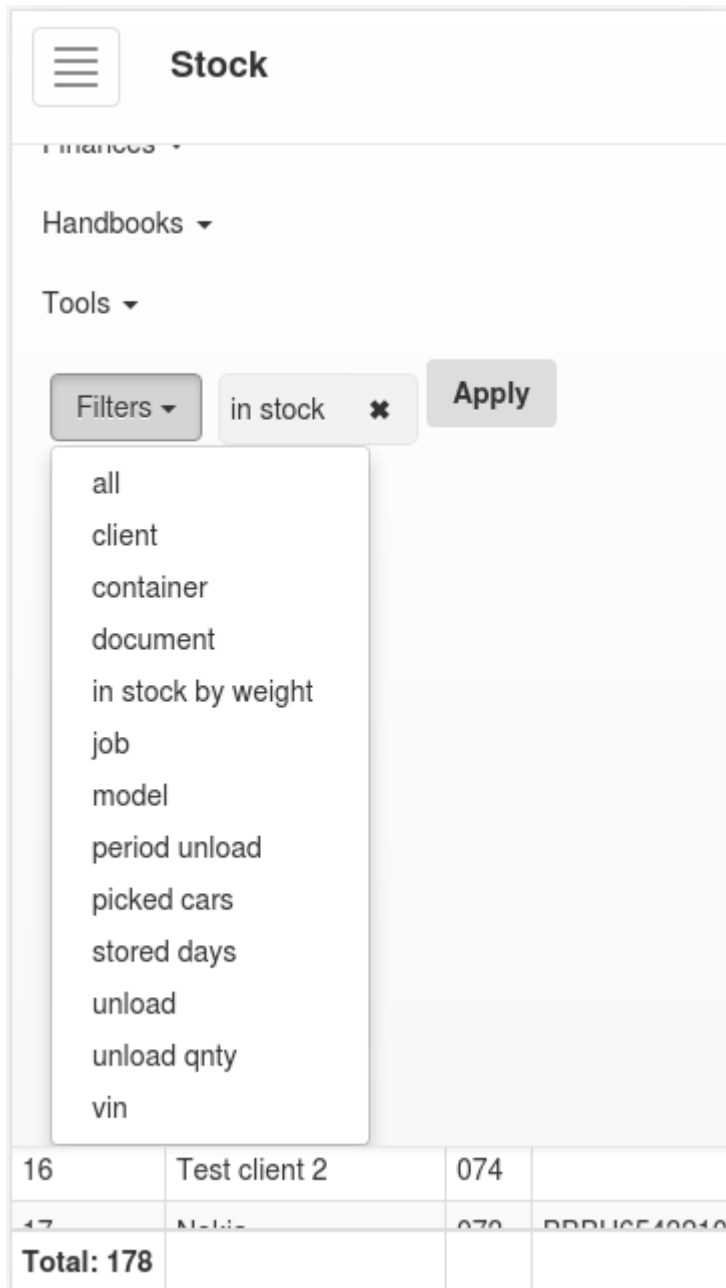



Figure 45. Filters on mobile devices.

Any number of filters can be picked and edited also on mobile devices (figure 46).



Stock

Workflow ▾

Finances ▾

Handbooks ▾

Tools ▾

Filters ▾

in stock ✖

document

document

✖

period unload
✖

Date min

Date max

container

Container

✖

Apply

Remove ✖

16	Test client 2	074	
17	M-11-	070	DDDUCE40010
Total: 178			

Figure 46. Mobile filters editing.

Work with queries is possible on mobile devices too. For example, processing of "test" query can be run by touching proper button (figure 47).


 Queries		
Query ▼		Description
test	<input type="button" value="run"/>	
supplier bills check	<input type="button" value="run"/>	
stock unload items import	<input type="button" value="run"/>	
stock unload import	<input type="button" value="run"/>	
stock storage	<input type="button" value="run"/>	
stock specific terms	<input type="button" value="run"/>	
stock report for client	<input type="button" value="run"/>	
stock report extended	<input type="button" value="run"/>	В отчет по:
stock on date	<input type="button" value="run"/>	
stock missing/duplicated numbers	<input type="button" value="run"/>	
stock import unload	<input type="button" value="run"/>	
	<input type="button" value="run"/>	

Figure 47. Queries on mobile devices.

When user runs query on mobile device, full screen pop-up window is opened (figure 48).

Row	cl_id	client	job_id	job	cnt_id	cont
1	1	LG	151982	0140	74896	HHH
2	1	LG	376719	0237		
3	1	LG	376719	0237		
4	1	LG	376719	0237		
5	1	LG	380414	0240		
6	1	LG	415613	0265	205744	TTNI
7	1	LG	415613	0265	205744	TTNI
8	1	LG	415613	0265	205744	TTNI
9	1	LG	415613	0265	205744	TTNI
10	1	LG	415613	0265	205744	TTNI
11	1	LG	415613	0265	205744	TTNI
12	1	LG	415613	0265	205744	TTNI
Total: 33						

Figure 48. Query processing on mobile device.

5.3.4 Application work summary

That was the description of current version application work. As said before the interface is built using 6 web components. Plan for future is to continue development of components and realize single form view, sub tables, records editing and so on up to full CRUD of all data views. The main principle is to solve as many accounting tasks as possible avoiding growth of complexity in program structure complexity - using one structure for everything.

6 SUMMARY

The result of this thesis is the solution that is easy to support and extend. Total amount of code lines written is about 1100, excluding configuration file. Code has been rewritten many times. Due to fast rendering speed there is no need for graphical displaying of waiting process. This result is considered important because modern users have high expectations for a site loading delay. According to the “Forbes” publication [19], users expect pages to load in two seconds, and after three seconds, up to 40% of users will abandon the web site. Time of the application loading in Chromium is shown in figure 49.

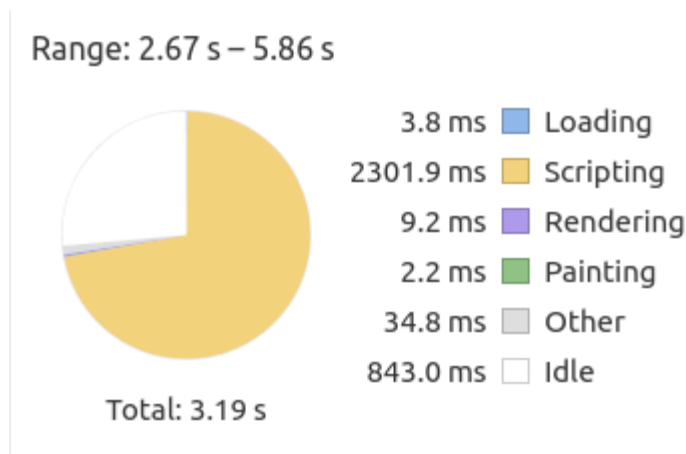


Figure 49. Time of application loading.

Time of displayed table view switching is shown in figure 50.

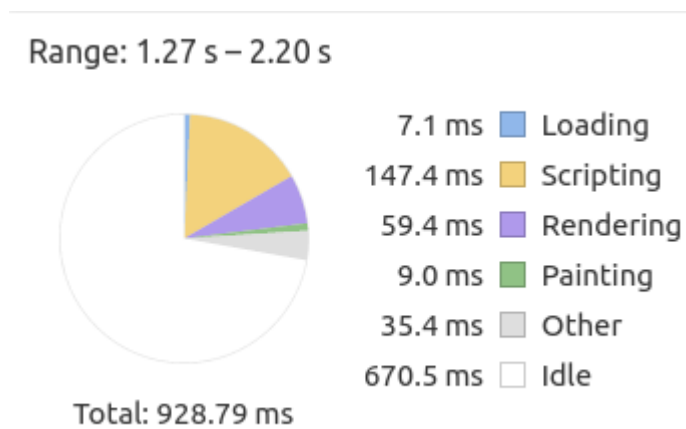


Figure 50. Time of table view switching.

Solving general problem allowed eliminating dependence of spaghetti code from number of tables. One code base processes any number of related tables. Solution does not depend on kind of tables needed to be displayed, their structure and their count. Application state is set by configuration file that allows for declarative style programming rather than imperative. Simplicity of supporting is achieved by minimal code base and prevalence of self-developed code. Developed application can be extended to implement plans for its future.

REFERENCES

1. Tpoint Oy. WWW page. Available at: <https://tpoint.fi> [Accessed 18 May 2017].
2. Microsoft TechNet. System requirements for Office 2010. WWW page. Available at: <https://social.technet.microsoft.com/wiki/contents/articles/13967.system-requirements-for-office-2010.aspx> [Accessed 18 May 2017]
3. Microsoft developers network. Changes in access. Office 2013 and later. WWW page. Updated 28 July 2015. Available at: <https://msdn.microsoft.com/en-us/library/office/ij618413.aspx> [Accessed 18 May 2017]
4. ComScore. All digital growth now coming from mobile usage. WWW page. Updated 3 April 2016. Available at: <http://marketingland.com/digital-growth-now-coming-mobile-usage-comscore-171505> [Accessed 18 May 2017]
5. Eric Ries, 2009. Minimum Viable Product: a guide. WWW page. Updated 3 August 2009. Available at: <http://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html> [Accessed 18 May 2017]
6. Stat Counter global stats. Desktop operating system market share worldwide. Feb 2009 to Feb 2017. WWW page. Available at: <http://gs.statcounter.com/os-market-share/desktop/worldwide/#monthly-200902-201702> [Accessed 18 May 2017]
7. Diana Goovaerts, 2016, Death of Desktop: 65% of Digital Media Consumed on Mobile. WWW page. Updated 31 March 2016. Available at: <https://www.wirelessweek.com/news/2016/03/death-desktop-65-digital-media-consumed-mobile> [Accessed 18 May 2017]
8. Dave Chaffey, 2017. Mobile marketing statistics compilation. WWW page. Updated 1 March 2017. Available at: <http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/> [Accessed 18 May 2017]
9. Citrix. 7 Enterprise Mobility Statistics You Should Know. WWW page. Updated February 2015. Available at : <https://www.citrix.ru/articles/7-enterprise-mobility-statistics-you-should-know.html> [Accessed 18 May 2017]
10. Stat Counter global stats, 2016. Mobile and tablet internet usage exceeds desktop for first time worldwide. WWW page. Available at: <http://gs.statcounter.com/press/mobile-and-tablet-internet-usage-exceeds-desktop-for-first-time-worldwide> [Accessed 18 May 2017]

11. Chromium Blog, 2013. Saying Goodbye to Our Old Friend NPAPI. WWW page. Updated 23 September 2013. Available at: <https://blog.chromium.org/2013/09/saying-goodbye-to-our-old-friend-npapi.html> [Accessed 18 May 2017]
12. Bugzilla@Mozilla, 2016. Remove support for all NPAPI plugins (except Flash). WWW page. Available at: https://bugzilla.mozilla.org/show_bug.cgi?id=1269807 [Accessed 18 May 2017]
13. Adobe corporate communications, 2015. Flash, HTML 5 and open web standards. WWW page. Updated 30 November 2015. Available at: <https://blogs.adobe.com/conversations/2015/11/flash-html5-and-open-web-standards.html> [Accessed 18 May 2017]
14. Alex Russell, 2011. Web Components and Model Driven Views. Video and transcript, WWW page. Available at: <https://fronteers.nl/congres/2011/sessions/web-components-and-model-driven-views-alex-russell> [Accessed 18 May 2017]
15. Web Components. WWW page. Updated 11 May 2017. Available at: <https://www.webcomponents.org/> [Accessed 18 May 2017]
16. Stat Counter global stats. Browser Market Share Worldwide from Jan 2009 to Feb 2017. WWW page. Available at: <http://gs.statcounter.com/browser-market-share#monthly-200901-201702> [Accessed 18 May 2017]
17. W3Counter. Web browser usage trends. WWW page. Available at: <https://www.w3counter.com/trends> [Accessed 18 May 2017]
18. Web Components. Polyfills. WWW page. Available at: <https://www.webcomponents.org/polyfills> [Accessed 18 May 2017]
19. Lara Swanson, 2014. Web Performance Is User Experience. WWW page. Updated 16 January 2014. Available at: <https://www.forbes.com/sites/oreillymedia/2014/01/16/web-performance-is-user-experience/#1854781e5a52> [Accessed 18 May 2017]
20. Coffeescript. WWW page. Available at: <http://coffeescript.org/> [Accessed 18 May 2017]
21. Bootstrap studio. WWW page. Available at: <https://bootstrapstudio.io/> [Accessed 18 May 2017]
22. JQuery UI. WWW page. Available at: <https://jqueryui.com/> [Accessed 18 May 2017]
23. W2UI. WWW page. Available at: <http://w2ui.com/web/> [Accessed 18 May 2017]

24. MVC Architecture. WWW page Available at:
https://developer.chrome.com/apps/app_frameworks [Accessed 18 May 2017]
25. Class diagram. WWW page. Available at:
https://en.wikipedia.org/wiki/Class_diagram [Accessed 18 May 2017]
26. Two hard things. WWW page. Updated 14 August 2015. Available at:
<https://martinfowler.com/bliki/TwoHardThings.html> [Accessed 18 May 2017]
27. Rich Internet Application. WWW page Available at:
<https://www.techopedia.com/definition/2531/rich-internet-application-ria>
[Accessed 18 May 2017]
28. Single Page Application. WWW page. Available at:
<https://www.codeschool.com/beginners-guide-to-web-development/single-page-applications> [Accessed 18 May 2017]

LIST OF FIGURES

Figure 1. Polyfill example	14
Figure 2. Blocks in BEM methodology	18
Figure 3. Elements in BEM methodology.	18
Figure 4. BEM structure in CSS	18
Figure 5. D3js rendering	20
Figure 6. Result of D3js rendering	21
Figure 7. Variables in CoffeeScript	21
Figure 8. Variables in JavaScript	22
Figure 9. Functions in CoffeeScript	22
Figure 10. Function in JavaScript	22
Figure 11. hello-world.html	25
Figure 12. hello-world.coffee	25
Figure 13. hello-world.css	26
Figure 14. Example of hello-world HTML tag usage	26
Figure 15. Initial content of hello-world HTML tag	26
Figure 16. Content of hello-world HTML tag modified.	27
Figure 17. UML class diagram of system components	28
Figure 18. Tabular interface of program	29
Figure 19. Main menu	29
Figure 20. Custom action "Logout" and name of table view "Stock"	30
Figure 21. Filters	31
Figure 22. "stored days" filter form with parameters	31
Figure 23. Filter form during typing of parameters	32
Figure 24. Table with parameterized filter applied	32
Figure 25. Totally unfiltered table.	33
Figure 26. Filter "period" with parameters of type "date"	34
Figure 27. Examples of input for filter parameters of type "date"	34
Figure 28. Input to filter parameters converted to proper format when applying ..	34
Figure 29. Table "Stock" sorted by "Date" column in descending order	35
Figure 30. Loading indication	35
Figure 31. "Whole table is loaded" indication	36
Figure 32. Vertically and horizontally scrolled table	36

Figure 33. Columns with auto-width	37
Figure 34. Cell content overflow	37
Figure 35. Display the whole content of the cell	38
Figure 36. Configuration of menu	38
Figure 37. Custom actions in menu	39
Figure 38. Tables configuration	39
Figure 39. Custom table content	40
Figure 40. Stored SQL queries.....	41
Figure 41. Pop-up window for query processing. Parameters entering phase. ...	41
Figure 42. Pop-up window with query result.....	42
Figure 43. Example program view for smartphone.....	43
Figure 44. Mobile menu.....	44
Figure 45. Filters on mobile devices.....	45
Figure 46. Mobile filters editing.....	46
Figure 47. Queries on mobile devices.....	47
Figure 48. Query processing on mobile device.....	48
Figure 49. Time of application loading.....	49
Figure 50. Time of table view switching.....	49

Delete the references parts from the list of figures

APPENDIX

Figure 51. Desktop operating system market share - worldwide.

Figure 52. Growth in digital media time.

Figure 53. Mobile marketing statistics compilation

Figure 54. Share of content category time spent by platform

Figure 55. Mobile and tablet internet usage exceeds desktop

Figure 56. Web Components status of realization in web browsers

Figure 57. Browser market share worldwide

Figure 58. Web browser usage trends]

Desktop Operating System Market Share Worldwide Jan 2009 to Feb 2017

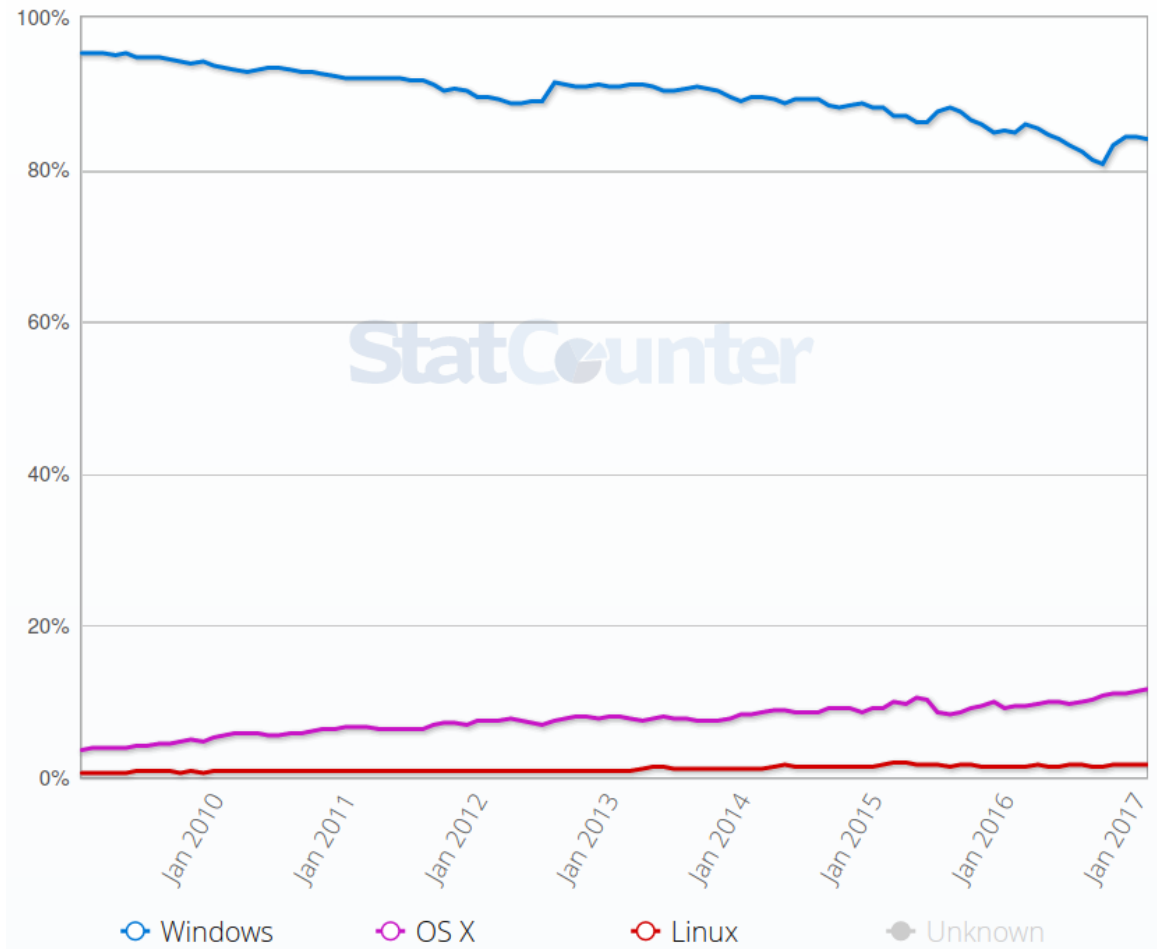


Figure 51. Desktop operating system market share - worldwide.

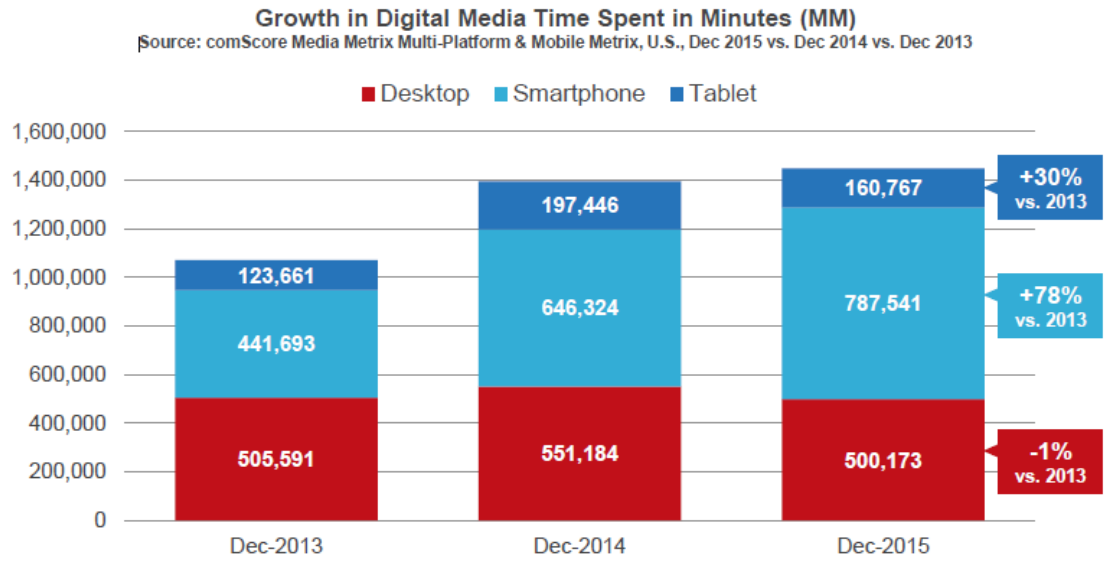


Figure 52. Growth in digital media time.

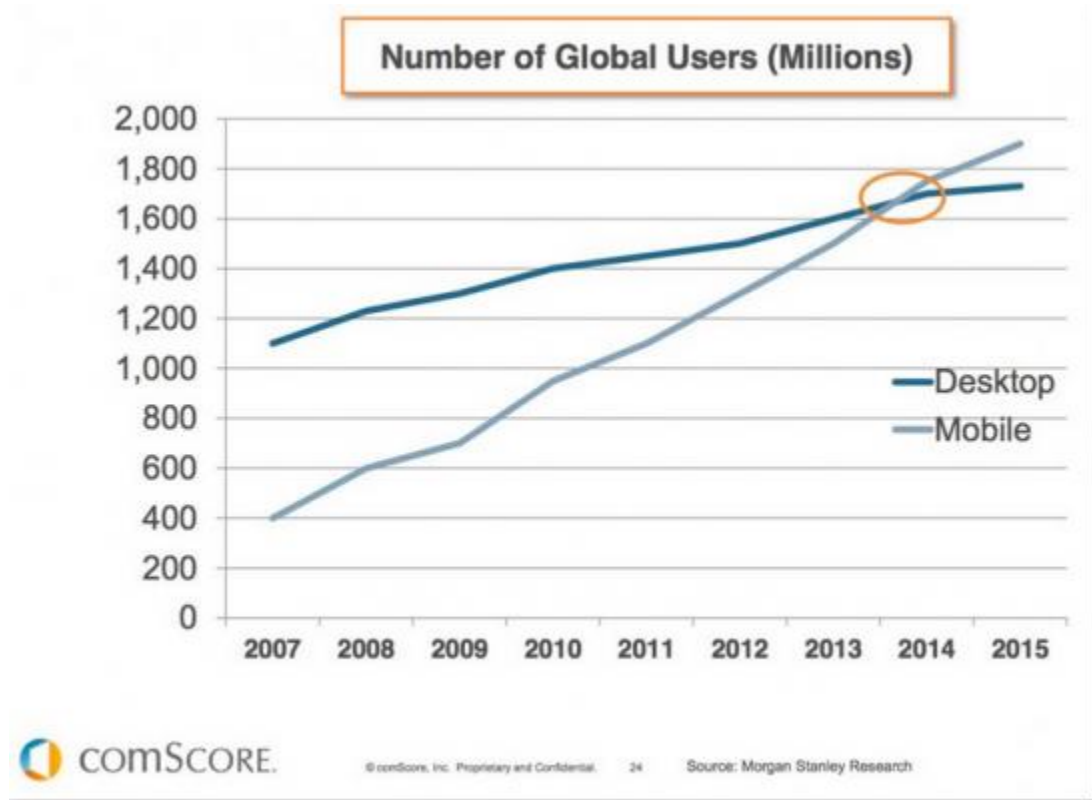


Figure 53. Mobile marketing statistics compilation

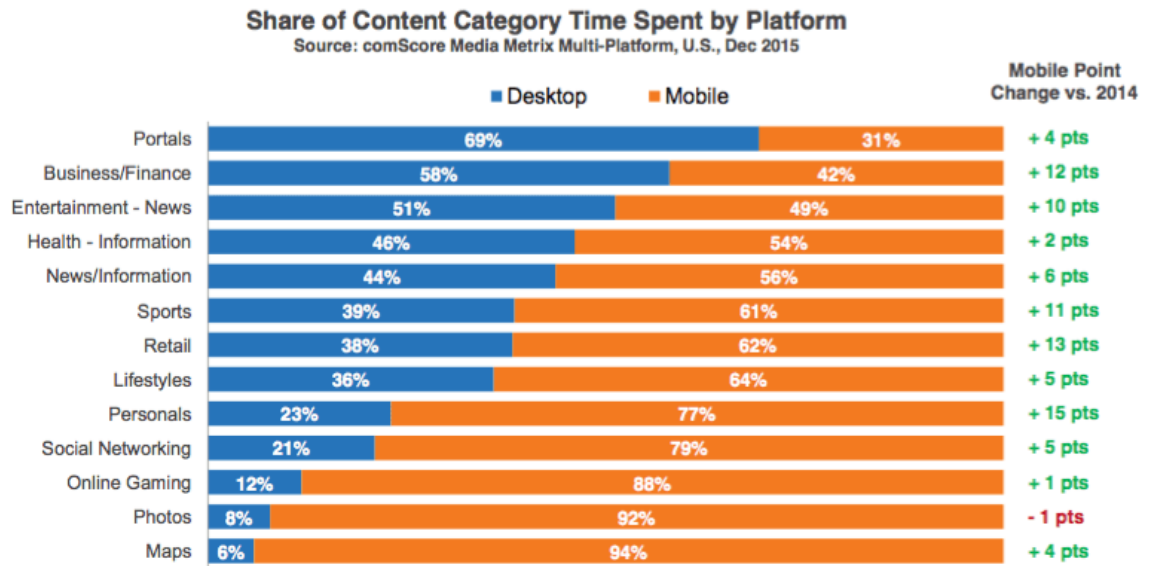


Figure 54. Share of content category time spent by platform [Reference 4]

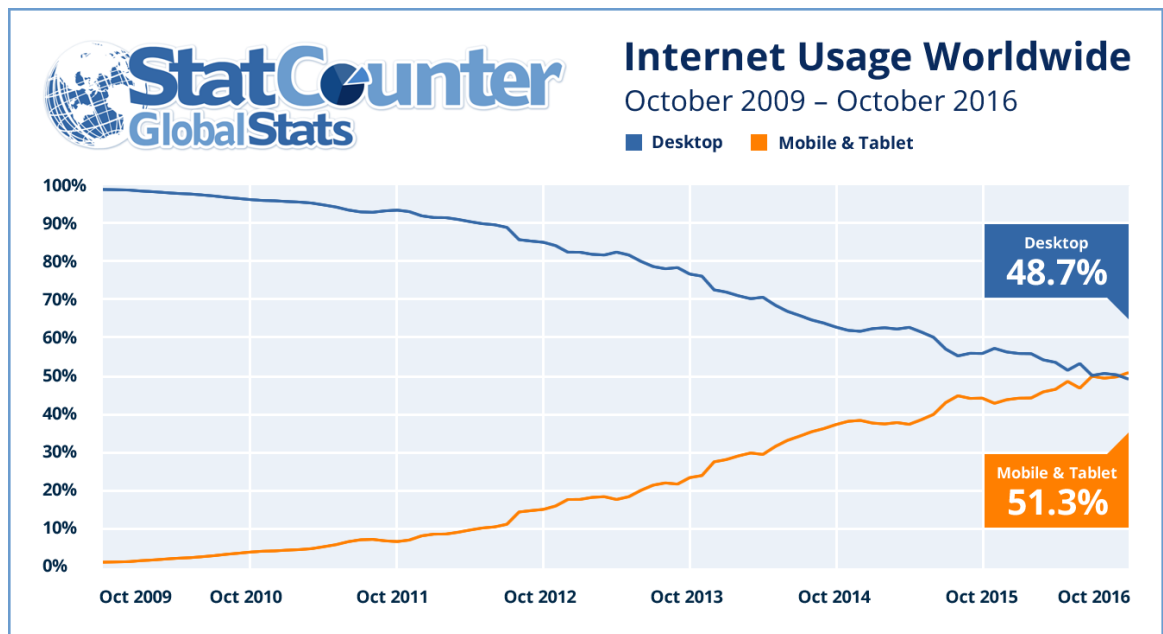


Figure 55. Mobile and tablet internet usage exceeds desktop [Reference 10]

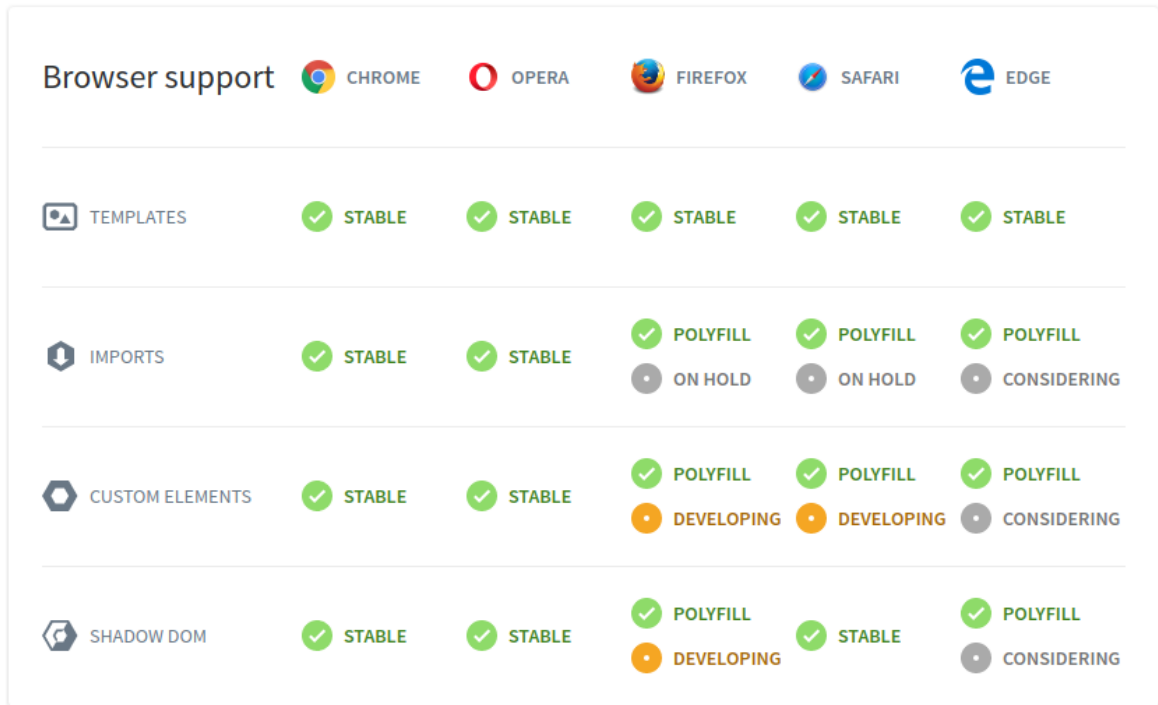


Figure 56. Web Components status of realization in web browsers [Reference 15]

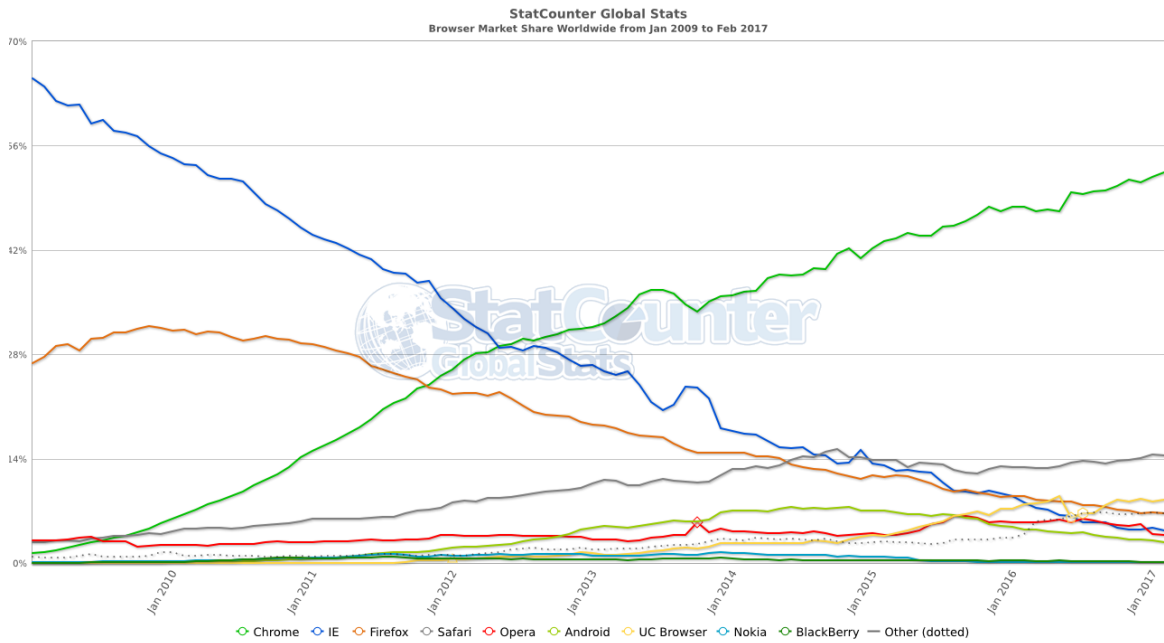


Figure 57. Browser market share worldwide [Reference 16]

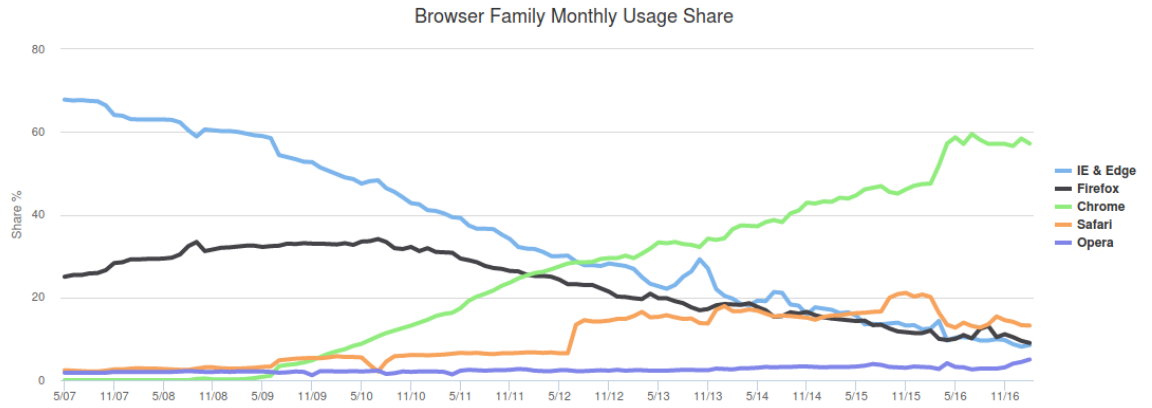


Figure 58. Web browser usage trends [Reference 17]