Adding dynamic statistics to Avinode Marketplace

Bachelor thesis in Computer Engineering

NORA IDBRATT LUNDGREN & GUSTAV DAHL
Adding dynamic statistics to Avinode Marketplace
Abstract

The online air charter company Avinode desires dynamic data visualization in their web application Avinode Marketplace that presents statistics related to their customer’s aircrafts. The purpose of this bachelor thesis is to develop a module that displays statistics with the use of a JavaScript charting library, which will be implemented in a proof of concept. The main objectives of the project include an analysis of different JavaScript charting libraries as well as an understanding of the Avinode Marketplace architectural structure in order to create a solution that fulfils their requirements while maintaining the standardized structure. The project also requires a solution that calculates and stores the data which will be presented in the chart. This bachelor thesis is completely software oriented and is accomplished with the use of the programming languages Java, HTML, JavaScript and JPQL, and the libraries Hibernate ORM and amCharts. The calculated data is stored using TreeMaps and the result is tested with the frameworks JUnit and Mockito. The result of this project is a chart displaying various statistics by utilizing the charting library amCharts. The chart content is dynamically updated as the user switches between the different views that are available with drop down selectors.

Keywords: Web application, Statistics, JavaScript, Charting library, amCharts, Java.
Preface

This report covers a Bachelor thesis for the department of Computer Science and Engineering at Chalmers University of Technology. The project was performed at the company Avinode AB in Gothenburg.

Firstly, we would like to thank Avinode AB for providing us with the equipment that was needed in order to pursue this project. Many thanks go to our supervisor at Avinode AB, Fredrik Vakk, who provided us with the assignment and ensured that we always had everything that was needed in order to proceed with our work. We would also like to thank Robin Söderström, Mårten Bohlin, Henrik Rostam and Jens Rydholm for showing support and providing us with technical help and feedback when it was most needed. We also thank our supervisor from Chalmers University of Technology, Joachim von Hacht, for helping us refining the task into an appropriate subject for a Bachelor thesis, as well as improving the quality of this report.

Lastly, we send our gratitude to all employees at Avinode AB for motivating us to work hard by showing us great support and hospitality.

Göteborg, 2014-06-02
Nora Idbratt Lundgren & Gustav Dahl
# Table of Content

1. Introduction .......................................................................................................................... 1
   1.1 Background ..................................................................................................................... 1
   1.2 Purpose ............................................................................................................................ 1
   1.3 Limitations ....................................................................................................................... 2

2. Method .................................................................................................................................. 3

3. Technical background ......................................................................................................... 4
   3.1 HyperText Markup Language ...................................................................................... 4
   3.2 Scalable Vector Graphics ............................................................................................. 4
   3.3 Vector Markup Language ............................................................................................. 4
   3.4 HTML5 Canvas .............................................................................................................. 4
   3.5 Cascading Style Sheets ................................................................................................. 5
   3.6 JavaScript ..................................................................................................................... 5
   3.7 JavaScript Object Notation ........................................................................................... 5
   3.8 Enterprise JavaBean ...................................................................................................... 5
   3.9 Java Persistence API ..................................................................................................... 6
   3.10 Hibernate ORM ......................................................................................................... 6
   3.11 Java Servlet ............................................................................................................... 6
   3.12 JavaServer Pages ........................................................................................................ 6
   3.13 Data Transfer Object ................................................................................................. 7
   3.14 JUnit ............................................................................................................................ 7
   3.15 Mockito ........................................................................................................................ 7
   3.16 Microsoft SQL Server ............................................................................................... 7
   3.17 GlassFish .................................................................................................................... 7

4. Rendering technologies ....................................................................................................... 8
7.1.1 General discussion ........................................................................................................... 29
7.1.2 Environmental and ethical aspects ................................................................................... 30
7.1.3 Gathering of information ............................................................................................... 30
7.2 Further development ........................................................................................................... 30

References

Appendixes
Abbreviations and Terminology

Abbreviations

CSS – Cascading Style Sheet
DOM – Document Object Model
DTO – Data Transfer Object
EJB – Enterprise JavaBeans
HTML – HyperText Markup Language
HQL – Hibernate Query Language
JPQL – Java Persistence Query Language
JSON – JavaScript Object Notation
JSP – JavaServer Pages
JSTL – JavaServer Pages Standard Tag Library
MVC – Model-View-Controller
ORM – Object-Relational Mapping
SVG – Scalable Vector Graphics
SQL – Structured Query Language
UML – Unified Modeling Language
VML – Vector Markup Language
XML – Extensible Markup Language

Terminology

Hibernate ORM - A library used to map Java objects with entities of a relational database.
HTML5 Canvas - Rendering technique used to draw graphics on a web page.
Java Platform, Enterprise Edition (Java EE) - A Java platform that provides an API for web and network applications.
JUnit - Testing framework for Java code.
Mockito - Testing framework that allows the creation of fake (mock) objects.
Scriptlet - Java code embedded in JSP code.
1. Introduction

1.1 Background

Avinode AB is the world's leading platform for air charter, functioning as a rendezvous for private jet rental. The company originated from Chalmers School of Entrepreneurship with an idea of simplifying the process of chartering private jets. Before the entrance of Avinode, attempts of chartering a jet were done through a catalogue listing the different operators and their rentable jets. Since the information was printed it was not possible to see neither the price nor availability of a jet for a specific trip. As a result, the customer had to call the operator to receive the required information. Avinode simplified this process by introducing an online marketplace.

In 2002, Avinode launched a web application called Avinode Marketplace where operators and brokers were able to sell and buy air charter online. Today, over 6000 flight requests are sent a day, with more than 7000 operators and brokers using the Avinode Marketplace. In order to increase business awareness for a user, the Marketplace homepage contains statistic charts displaying availability, demand and pricing of aircrafts. These indexes are static images generated by an old version of Google Charts. Avinode is not satisfied with the current solution of displaying statistics through static images. Instead, Avinode wishes to find an alternative which supports interactivity and dynamic updating of the data visualization.

1.2 Purpose

The purpose of this bachelor thesis is to create a module in Avinode’s web application for data visualization. The module will complement their static implementation with charts that are interactive and dynamically updated. Users should be able to adjust the data presentation by specifying aircraft statistics that should be displayed.
1.3 Limitations

This is a strictly software oriented bachelor thesis for visualizing data in bar, line and pie charts with the use of a JavaScript charting library. The result will be a proof of concept and will therefore not be deployed in the running Avinode system. Instead, it will be implemented in a temporary environment with a smaller database.
2. Method

This bachelor thesis will be performed in a working methodology similar to Scrum, where the problem is divided into smaller pieces. Initially, for each subtask the time consumption, matter of importance and level of difficulty of implementation will be estimated and presented in a Gantt chart (see appendix 1).

Each task will be performed one at a time, starting with an understanding of what criteria a charting library should fulfil according to Avinode. Twenty libraries will be evaluated and four of these will later on be tested in practice with Spring MVC, which is a web framework using the design pattern Model-View-Controller. The practical tests will include the creation of interactive pie, line and column charts in web browsers used by Avinode’s customers. Thereafter, the library most suited for Avinode will be used to implement a proof of concept displaying statistics in their system.

An examination of Avinode’s system and the tools that they use is necessary before doing the actual implementation. The tools include Hibernate, Spring MVC, GlassFish and Git. Unit tests with JUnit and Mockito will be used to verify that the solution behaves as intended.
3. Technical background

3.1 HyperText Markup Language

The most common language used when creating the structure of a web page is HyperText Markup Language (HTML). It is a markup language that consists of a set of markup tags. The tags form elements that define the content of a document, which is interpreted by web browsers and displayed as a web page [1].

3.2 Scalable Vector Graphics

Scalable Vector Graphics (SVG) is an XML-based vector graphic file format used for graphic rendering. Each SVG object is a Document Object Model (DOM) object, which facilitates manipulation of element properties. Event listeners, such as mouse-over and on-click, can be assigned to any SVG object. Furthermore, SVG is resolution independent and scaling does not affect the image quality [2]. Supported browsers include Chrome 4.0+, Firefox 3.0+, Opera 9.0+ Internet Explorer 9.0+ and Safari 3.2+ [3].

3.3 Vector Markup Language

Another XML-based file format used for web vector graphics is Vector Markup Language (VML). It is compatible with Internet Explorer version 5+ and several charting libraries therefore use VML in combination with SVG or HTML5 Canvas in order to compensate for otherwise unsupported browser versions [4].

3.4 HTML5 Canvas

A third technique that is used to draw graphics is HTML5 Canvas. Unlike SVG, HTML5 Canvas is based on raster graphics, which means that the image has a specific resolution and that scaling will affect the image quality. HTML5 Canvas does however have high performance [5]. Supported browsers include Chrome 4.0+, Firefox 2.0+, Opera 9.0+, Internet Explorer 9.0+ and Safari 3.2+ [6].
3.5 Cascading Style Sheets
In order to customize the appearance of markup languages, the style sheet language Cascading Style Sheet (CSS) is used. By writing the formatting of a document in CSS, the document content is separated from the document presentation, thus increasing flexibility and content accessibility. The specified properties for a given style sheet can be reused on several elements, hence it reduces code redundancy [7].

3.6 JavaScript
JavaScript is an object-oriented scripting language that implements prototyped-based inheritance [8]. It is commonly used for web development as a client-side script, in other words, a script that is embedded in or referenced from an HTML or Extensible HTML (XHTML) document. JavaScript can be used to create, delete, change and clone DOM elements [9].

JavaScript is often used together with Asynchronous JavaScript and XML (AJAX), which is a technique for updating parts of a web page without reloading the entire page with a new HTTP request [10].

3.7 JavaScript Object Notation
The text format JavaScript Object Notation (JSON) is commonly used for storing and transporting data. Even though JSON is based on JavaScript syntax, it is language and platform independent and may therefore be used in several programming languages. JSON has the benefits of being easy to read and write for humans, as well as being easy to parse and generate for machines [11] [12].

3.8 Enterprise JavaBean
The Enterprise JavaBeans (EJB) technology is part of Java Platform, Enterprise Edition (Java EE) and is a collection of server-side components with the purpose of facilitating business logic development. Business logic refers to a set of rules on how data should be created,
stored and modified. The responsibilities of EJB’s include management of transactions [13] [14].

3.9 Java Persistence API

Java Persistence API (JPA) is a specification that describes how to handle Object-Relational Mapping (ORM), which means mapping of Java objects to relational databases. The JPA specifications reduce the need to create queries that directly access database tables. Instead, the queries accesses the data through JPA entity objects that are mapped to the corresponding table. These queries are created with the object-oriented Java Persistence Query Language (JPQL) [15] [16].

3.10 Hibernate ORM

An example of a Java library that implements the JPA specifications is Hibernate ORM. Hibernate ORM maps data from the relational database to Hibernate objects that can be accessed with the use of queries written in Hibernate Query Language (HQL). HQL is similar to Structured Query Language (SQL), although HQL is object-oriented [17] [18].

3.11 Java Servlet

A Java Servlet is a server-side component often used for handling Hypertext Transfer Protocol (HTTP) requests and responses [19].

3.12 JavaServer Pages

The JavaServer Pages (JSP) technology is an extension of the servlet technology used to generate dynamic web content. JSP Standard Tag Library (JSTL) is a collection of JSP tags that allows the creation of customized tags with inbuilt logic. Even though it is possible to write Java code inside a JSP with the usage of a scriptlet, it should be avoided since it is against the code convention of Oracle [20] [21].


3.13 Data Transfer Object

In order to reduce latency in remote communication Data Transfer Objects (DTO) can be used to handle data transfer between processes. Instead of having several remote calls (from client to server) that acquires data, only one call is needed with the use of a DTO. The DTO will encapsulate all required data as it is initialized and the caller application would only require calling the DTO in order to access the data [22].

3.14 JUnit

The testing framework JUnit is commonly used as a tool to test that Java programs function correctly. JUnit has the purpose of unit testing small parts of Java programs with the use of boolean expressions [24].

3.15 Mockito

Another framework that facilitates testing is Mockito, which allows the creation of fake (mocked) objects. Fields and methods that are a part of the test should be specified. The outcome of methods is defined with the use of \textit{when} and \textit{thenReturn} methods [25].

3.16 Microsoft SQL Server

Microsoft SQL Server is a system for managing relational databases with the use of the query language Transact-SQL (T-SQL) [26]. The system became platform independent in 2014 and provides support for both .NET and Java [27].

3.17 GlassFish

GlassFish is a Java Platform, Enterprise Edition (Java EE) application server. A Java EE application server connects the client to the database and uses EJB’s for the business logic. GlassFish initiates EJB containers that execute the beans that are used by the application [28]. At present, there exist two versions of Glassfish: Oracle GlassFish Server (commercial) and GlassFish Server Open Source Edition (non-commercial) both developed by Oracle [29].
4. Rendering technologies

HTML5 Canvas and SVG are two techniques for rendering graphics. SVG elements are appended to the DOM, and can easily be manipulated with JavaScript and CSS. Its properties can be adjusted and event handlers can be added to each element. SVG objects are therefore more customizable and interactive compared to Canvas objects, which simply draws pixels that cannot be altered after creation.

Another benefit of using SVG is that when the object is scaled the quality remains unchanged, while the quality of a Canvas object is affected. However, Canvas has the advantage of having a higher performance compared to SVG, due to the fact that the document complexity increases with SVG being integrated into the DOM [4] [30].

Canvas objects can also easily be exported as PNG or JPG images, which SVG does not natively support. This can however be solved with the use of a plugin, which for instance the library Highcharts JS utilizes. By including a library such as Canvg.js into the HTML document, the possibility of converting the Highcharts JS objects from SVG to Canvas is enabled [31].
5. Implementation

This chapter will describe the entire process of implementation, from the initial analysis and evaluation of several charting libraries, to the creation of the dynamic statistic charts in a proof of concept. The analysis will be performed with respect to Avinode’s criteria and will include personal preferences regarding layout and level of difficulty of code comprehension. Some parts of this chapter should therefore be treated with caution.

5.1 Analysis and evaluation of JavaScript charting libraries

The following statements are important aspects and criteria of what a library should fulfil according to Avinode:

- License agreements and eventual pricings should be examined.
- The library should be reliable, comprehensible and compatible with all major browsers including Internet Explorer 8+.
- Column, line and pie charts should be available.
- The charts should be customizable, interactive and preferably have an aesthetic appearance as well as good performance.
- Available help and support is highly beneficial.

With respect to the criteria mentioned above, twenty libraries were briefly analysed. For each library advantages and disadvantages were evaluated, whereupon the library was graded on a scale from 1 to 10. The results of this evaluation are presented in figure 1. For a more detailed view about each of the libraries (see appendix 2).
<table>
<thead>
<tr>
<th>License</th>
<th>Non-commercial</th>
<th>Commercial (single dev)</th>
<th>Rendering technique</th>
<th>Browser support*</th>
<th>Charts**</th>
<th>Interactivity***</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>amCharts</td>
<td>Proprietary</td>
<td>Free</td>
<td>$140</td>
<td>SVG, VML</td>
<td>All</td>
<td>All</td>
<td>9</td>
</tr>
<tr>
<td>CanvasJS</td>
<td>CC BY-NC 3.0 or commercial</td>
<td>Free</td>
<td>$299</td>
<td>Canvas</td>
<td>Not IE8</td>
<td>All</td>
<td>4</td>
</tr>
<tr>
<td>ChartJS</td>
<td>Non-commercial or commercial</td>
<td>Free</td>
<td>$299</td>
<td>SVG</td>
<td>All</td>
<td>All</td>
<td>6</td>
</tr>
<tr>
<td>D3.js</td>
<td>BSD-3</td>
<td>Free</td>
<td>Free</td>
<td>SVG, VML</td>
<td>Not IE8</td>
<td>All</td>
<td>8</td>
</tr>
<tr>
<td>dc.js</td>
<td>Apache 2.0</td>
<td>Free</td>
<td>Free</td>
<td>SVG, VML</td>
<td>Not IE8</td>
<td>All</td>
<td>7</td>
</tr>
<tr>
<td>Dygraphs</td>
<td>MIT</td>
<td>Free</td>
<td>Free</td>
<td>Canvas</td>
<td>Only pie</td>
<td>All</td>
<td>1</td>
</tr>
<tr>
<td>Elycharts</td>
<td>MIT</td>
<td>Free</td>
<td>Free</td>
<td>SVG, VML</td>
<td>All</td>
<td>Only mouse-over</td>
<td>1</td>
</tr>
<tr>
<td>Plot</td>
<td>MIT</td>
<td>Free</td>
<td>Free</td>
<td>Canvas</td>
<td>Not IE8</td>
<td>All</td>
<td>6</td>
</tr>
<tr>
<td>Floxtr2</td>
<td>Proprietary</td>
<td>Free</td>
<td>Free</td>
<td>Canvas</td>
<td>Not IE8</td>
<td>All</td>
<td>2</td>
</tr>
<tr>
<td>FusionCharts XT</td>
<td>Proprietary</td>
<td>$399</td>
<td>$399</td>
<td>SVG, VML</td>
<td>All</td>
<td>All</td>
<td>5</td>
</tr>
<tr>
<td>Google Charts</td>
<td>Google API ToS</td>
<td>Free</td>
<td>Free</td>
<td>SVG, VML</td>
<td>All</td>
<td>All</td>
<td>9</td>
</tr>
<tr>
<td>gRaphael</td>
<td>MIT</td>
<td>Free</td>
<td>Free</td>
<td>SVG, VML</td>
<td>All</td>
<td>Neither zoom nor range-select</td>
<td>4</td>
</tr>
<tr>
<td>Highcharts JS</td>
<td>CC BY-NC 3.0 or commercial</td>
<td>Free</td>
<td>$90</td>
<td>SVG, VML</td>
<td>All</td>
<td>Not range-select</td>
<td>8</td>
</tr>
<tr>
<td>jqPlot</td>
<td>MIT or GPL 2.0</td>
<td>Free</td>
<td>Free</td>
<td>Canvas</td>
<td>Not IE8</td>
<td>All</td>
<td>6</td>
</tr>
<tr>
<td>JS Charts</td>
<td>Non-commercial or commercial</td>
<td>Free</td>
<td>$79</td>
<td>Canvas</td>
<td>Not IE8</td>
<td>All</td>
<td>3</td>
</tr>
<tr>
<td>ProtoChart</td>
<td>MIT</td>
<td>Free</td>
<td>Free</td>
<td>Canvas</td>
<td>All</td>
<td>Not range-select</td>
<td>1</td>
</tr>
<tr>
<td>Protovis</td>
<td>BSD-2</td>
<td>Free</td>
<td>Free</td>
<td>SVG</td>
<td>Not IE8</td>
<td>All</td>
<td>1</td>
</tr>
<tr>
<td>RGraph</td>
<td>MIT</td>
<td>Free</td>
<td>Free</td>
<td>Canvas</td>
<td>Not IE8</td>
<td>All</td>
<td>2</td>
</tr>
<tr>
<td>Shield UI Chart</td>
<td>Non-commercial or commercial</td>
<td>Free</td>
<td>$299</td>
<td>SVG, VML</td>
<td>All</td>
<td>Not range-select</td>
<td>7</td>
</tr>
<tr>
<td>xCharts</td>
<td>Proprietary</td>
<td>Free</td>
<td>Free</td>
<td>SVG</td>
<td>Not IE8</td>
<td>Not pie</td>
<td>1</td>
</tr>
</tbody>
</table>

* Later versions of Mozilla Firefox, Google Chrome, Opera, Safari and Internet Explorer 8+
** Line, column and pie charts
*** Mouse-over, on-click, zoom and range-select

5.2 Testing of the four most profitable charting libraries

The brief analysis resulted in a sifting to four prominent libraries including amCharts, D3.js, Google Charts and Highcharts JS. In order to gain more knowledge on how each library functions in practice, tests were performed by creating simple line, column and pie charts. Appendix 3 contains the code as well as the resulting chart of these tests. Worth noting from the comparison is that D3.js requires 90 lines of code to produce a line chart that contains less
information than what Google Charts does in 27 lines of code. Below is a conclusion of the practical testing.

5.2.1 amCharts

amCharts is favourable in many ways. The syntax is easy to comprehend and several features are provided automatically. For instance, by comparing the pie chart code with the result (see appendix 3) one can see that the percentage values are not handled manually, but are calculated and presented automatically. Another feature, which cannot be demonstrated in this report, is that each chart has mouse-over and on-click events activated in advance. If the provided features are not desired they can simply be deactivated.

The only problem that occurred with amCharts was that the CSS code could not be placed in a separate script tag without raising an error. Instead, it would have to be written directly in the corresponding div tag. The reason for this is unclear, since the examples provided on the official website uses the CSS in a separate script tag. However, this was not considered a major problem, since the only responsibility of the concerning style properties is defining the size of the chart.

5.2.2 D3.js

D3.js should preferably be used by highly experienced programmers, since it requires a lot of work to produce charts. The advantage of using D3.js is that the developer has control over every object and therefore has a great extent of customization abilities. The downside of this is that basically nothing is included beforehand, and a study on how to use the library is necessary before actually using it. Furthermore, maintenance could potentially be a problem if all successor developers would first need to study the D3.js syntax before the effective coding takes place.

5.2.3 Google Charts

Google Charts is a library that is easy to use and generally requires few lines of code in order to produce an attractive visualization of data. Large parts of the code remains unchanged
when switching between pie, line and column charts, which contributes to the simplicity of the library. Similarly to amCharts, Google Charts has some basic interactive features - such as mouse-over and on-click - enabled in advance. These features can easily be deactivated if desired.

5.2.4 Highcharts JS

Highcharts JS is easy to use, although it might require a bit more programming experience than amCharts and Google Charts when creating simple charts. This is partially because interactive features are not pre-set and would require JavaScript functions to handle action events. Although the library is relatively easy to use it is still very powerful and produces charts with professional appearances.

5.2.5 Outcome of testing

By carefully evaluating the general analysis and the practical testing, amCharts was considered to be most suitable for this assignment. While being one of the most easily manageable libraries, it still provides several features of high quality.

5.3 Avinode Marketplace system architecture

Avinode’s application is based on Java Platform, Enterprise Edition version 7 (Java EE 7) and uses the commercial Oracle GlassFish Server and a Microsoft SQL Server relational database.

5.3.1 System structure

The system uses the architectural pattern MVC consisting of three layers:

- A model, which is the core component of the application, responsible of handling data and business logic.
- A view, which is a representation of the model. This is what the user sees and it is modified by the model state via the controller.
- A controller, which connects the view and model. The controller receives requests from the user and manipulates the state of the model, whereupon the view is updated.
By using MVC, the user interface (view) is separated from the business logic (model) and a change in either layer will not directly affect the other layer. Communication between the layers is obtained through a controller. The next subchapter will describe how Avinode uses the MVC pattern in their system. Since the name of Java classes and JSP files differ depending on their assignment, Xxx will be used to generalize the names.

### 5.3.2 System flow

![MVC Diagram](image)

*Figure 2. Overview of the Avinode Marketplace system architecture*

Figure 2 displays a brief overview of how classes are divided in the MVC layers. The arrows represent the flow of how data is transferred through the system. It is beneficial to refer to the figure when reading the detailed description below.
Initially, a request is sent from the user interface (view) to the controller, whereupon the controller (XxxController) sends a command to the model, which in this case consists of several classes starting with a service bean. An instance of the bean must be injected into the controller in order to declare a reference to it. The injection is handled with the following annotation:

```
@EJB(mappedName = XxxService.PATH_TO_BEAN)
```

Thereafter, the communication with the model is initiated, and the service bean will commence a Hibernate session. Depending on whether the user request involves a reading or writing of data, an XxxHelper or XxxPersister object is created by the bean. If data is to be read, the user is validated in the XxxHelper and an interface, XxxStore, is called of which its declared methods are implemented in an XxxEntityQueryHelper. The implemented methods handles the communication with the Microsoft SQL Server database by using Java Persistence Query Language (JPQL) and will most often return an entity object (XxxEntityImpl), which contains data corresponding to attributes in the tables of the database.

In order for the controller to allow modification of the view (ViewXxx.jsp), the controller waits for the requested data, often in the form of a Data Transfer Object (DTO). Since the XxxEntityQueryHelper returns an entity, the XxxHelper must map the entity to a DTO by creating a DTO containing the relevant data that is accessed from the entity object. Thereafter, the DTO may be transferred all the way back to the controller, whereupon the data will be accessible by the view.

In summary, the controller is a single class known as XxxController, which modifies the view. The view may include one or several ViewXxx.jsp files. The model handles the business logic and consists of several classes and interfaces including XxxService, XxxServiceBean, XxxHelper, XxxPersister, XxxStore and XxxEntityQueryHelper.
5.3.3 Database structure

The image below (see figure 3) displays entities, attributes and relations of the relational database which Avinode uses that are of interest for this assignment. The use cases which resulted in this conclusion will be discussed in the subchapter 5.5.

![Diagram of database entities and relations]

Figure 3. Relevant entities, relations and attributes in the database

5.4 Requirements

This section will describe all iterations that were performed in order to produce the final result. Listed below are all use cases that will be included in this assignment. The resulting chart should display the number of times a day an aircraft belonging to the user company was:

- listed in a search result
- positioned at its home base as the search occurred
- positioned on another base than its home base as the search occurred
- clicked on as it appeared in a search result
- requested to be chartered (request for rental)
- displayed among first five results
5.5 Data format of proposed system

The data which will be displayed in the chart changes format as it passes through the system structure. Initially, all `FlightSearchHit` entities that are retrieved from the database in the `EntityQueryHelper` are stored in a list that is transferred to the `Helper`. Since the `FlightSearchHit` entities does not originally contain all necessary data, the entities must be mapped to DTO’s where all relevant data is stored by accessed attributes through relations. However, amCharts requires strings in JSON format to be accessed from the controller, and the controller itself should not contain any logic. Therefore, the DTO’s must be converted to JSON strings while remaining in the model. The `Helper` takes responsibility of the conversion and will pass the data in a processable format to the controller via the `ServiceBean`.

5.6 Realization of use cases

5.6.1 Count and display the total number of aircrafts in a fleet

The first iterative phase involved creating the required structure in all three layers of the MVC design pattern. The goal was to display a single value on the screen by sending a request from clicking a menu. Since the aircrafts listed in the user company’s fleet is required before the search hits can be handled, the number of `MarketedAircraft` entities having a `company_id` matching the caller company is counted. The caller company is retrieved in the `Helper` by accessing the current process context. The counting is performed with an execution of a JPQL query. The result is thereafter transferred to the controller and added as an attribute with the following Spring MVC method call to allow the view to access the value:

```java
model.addAttribute("Aircrafts", countedAircrafts);
```

Thereafter, the user is redirected to the JavaServer Page (JSP) in which the data is accessed with the following line:

```java
var aircrafts = ${Aircrafts};
```
No mapping to a DTO is required since the query returned a count value of the data type \textit{Long}.

5.6.2 Display the identification numbers of the aircrafts

The goal of the second iteration was to display the identification number of each aircraft listed in the user company’s fleet. A query that returns a list of \textit{MarketedAircraft} entities registered to the caller company is used. Thereafter, the entity list is mapped to a list of DTO’s that only contained a field for a registration number. The DTO list is passed to the controller, allowing the view to access the data and display the registrations number on the screen.

5.6.3 Display flight search hit information

The purpose of the third iteration was to display data related to flight search hits. This is performed by taking advantage of the data which is retrieved from the previous iteration. The JPQL query is modified so that it returns a list of \textit{FlightSearchHit} entities that has an \textit{aircraft_id} that matches the aircrafts in the company fleet. See appendix 4 for the final JPQL result.

The relevant data of the returned entities is stored in a list of \textit{SearchStatisticsDTO}’s, including the aircraft registration numbers, the date on when the search occurred as well as booleans whether the aircraft was positioned on its home base and if the search hit was clicked. Thereafter, the DTO is passed to the controller, whereupon the values are accessed by the view and displayed on the web page.

5.6.4 Get search hit occurrences for unique dates

The initial thought on how to store and count relevant data was to use several nested TreeMaps. One map containing the date of when the search occurred, another map keeping track of all registration numbers of the marketed aircrafts belonging to the caller company, and lastly a map that contains three different counters. The three counters include the number of total search hits, also the number of search hits that occurred while the aircraft was
positioned on its home base as well as a counter in the opposite case. This resulted in the following nested maps:

```
TreeMap<String, TreeMap<String, TreeMap<String, Integer>>>
```

TreeMaps has the benefit of automatically sorting keys. A disadvantage is that accessing each and every entry set would require three nested for-each loops with a complexity of \(O(n1*n2)\), where \(n1\) and \(n2\) are the sizes of the outer and middle maps. The innermost map is ignored since its size is constant by only storing the three counter types. The performance would be inefficient with large datasets, although it was still considered to be an approach for providing a solution due to the limited amount of time of the bachelor thesis.

By taking advantage of the booleans stored in the `SearchStatisticsDTO`, counting object state occurrences is easily done with the use of for-each loops and the result is stored in the innermost map with relation to the corresponding date and aircraft registration number. An additional factor that is to be taken into account is the position of the aircraft as the search occurred. Therefore, three counters (Total, Home base and Other base) is created and the calculated values are stored in the correlative place. In order for amCharts to be able to process the data, a method is needed that converts the maps to a string with a processable format. Below is an example of what the data should look like:

```
{"date":"2014-05-01",
 "SE-ABC":1,
 "SE-DEF":3,
 "SE-GHI":5,
 "SE-MNO":4}
```

The format consists of a date followed by each aircraft registration number (e.g. SE-ABC) registered to the caller company that was listed in a search result on that particular day. Next to each registration number is the total number of times that the specific aircraft appeared in a search result list on that day.
5.6.5 Use amCharts to present calculated data

Due to the risk of having overlapping data when using line graphs, column graphs was chosen to present the search hit result. A drop down selector is used to choose between the different search hit counters (Total, Home base and Other base), whereupon the values are presented for each aircraft (see figure 4).

![Stacked column chart displaying search hit occurrences](image)

The following JavaScript code creates the chart containing as many stacked column graphs as there are aircrafts registered to the company:

```javascript
var chart;
var chartData = [[${Total}],
                 [${Home}],
                 [${Other}]];
chart.dataProvider = chartData;

var regNrs = ${RegNrs}.split(",");

//Continued on next page
```
In order to switch between the datasets (Total, Home base and Other base) with the use of a drop down selector, the following HTML and JavaScript code must be added:

```html
<form>
    <select id="dataType" onchange = "selectDataset()">
        <option value="0"> Total </option>
        <option value="1"> Home base </option>
        <option value="2"> Other base </option>
    </select>
</form>

function selectDataset() {
    var type = document.getElementById("dataType");
    var value = type.options[type.selectedIndex].value;
    chartdataProvider = chartData[value];
    chart.validateData();
    chart.animateAgain();
}
```

```
AmCharts.ready(function() {
    chart = new AmCharts.AmSerialChart();
    chart.dataProvider = chartData[0];
    chart.cateforyField = "date";
    chart.dataDateFormat = "YYYY-MM-DD";

    for(var i=0; i<regNrs.length; i++) {
        var graph = new AmCharts.AmGraph();
        graph.type = "column";
        graph.title = regNrs[i];
        graph.valueField = regNrs[i];
        chart.addGraph(graph);
    }

    chart.write("chartdiv");
});
```
A benefit of using amCharts is that the library can parse dates and will automatically create a full period from the first entry to the last even though data for a specific date might be missing. The only required code which must be included is to tell the category axis to parse dates, define the minimum period as well as define the chart data date format (e.g. YYYY-MM-DD).

```javascript
categoryAxis.parseDates = true;
categoryAxis.minPeriod = "DD";
chart.dataDateFormat = "YYYY-MM-DD";
```

### 5.6.6 Add line graph that displays additional data

Three additional counters is required to display the number of times a search hit was requested, clicked and if it was displayed among the first five results in the search list. The chart design had to be taken into account before deciding where and how the data should be stored. Since the charts are divided in three different views (*Total*, *Home base* and *Other base*), each of the additional counters would require a value for every case. The possibility of using an additional TreeMap was excluded, since it would have a distinctive reduction in performance. Consequently, the counters could not be stored with relation to each aircraft.

Instead, three counters (*Requested*, *Clicked* and *Top-Five*) are placed in the middle TreeMap together with the aircraft registration numbers, thus having three different cases (*Total*, *Home base* and *Other base*). The DTO is updated with methods returning boolean states whether the flight search hit entity had a relational request entity, if it was clicked and if its position in the search result was less or equal to five. Thereafter, the TreeMap counters are calculated and added to the data string in the following format:

```json
{"date":"2014-05-06",
 "SE-ABC":3,
 "SE-JKL":6,
 "Clicked":5,
 "Requested":0,
 "Top-Five":1}
```
Each counter is represented by a line chart that is viewed by selecting the desired counter type with the use of a second drop down selector. Only one line chart was deliberately chosen to be shown at a time to avoid overlapping line charts and to facilitate comprehension (see figure 5).

An information box (balloon) is used to displays detailed information. The balloon appears as the user hover the mouse over any chart and displays either the aircraft registration number or the line chart counter along with its corresponding value. It also shows the total number of search hits when hovered over a stacked column chart.

Figure 5. Line chart displaying the number of times a search hit was either requested, clicked or appeared among the first five results

The following code demonstrates how the line graphs are added to the chart and how only one of these is visible when the chart is initiated.
The structure is completed with classes that contain all required fields and methods. The final structure is shown as a Unified Modeling Language (UML) diagram in figure 6. The diagram only contains the Java classes and interfaces that were created for this assignment. Therefore, other previously created classes that are inherited or implemented from are not shown. Furthermore, the link to the view as well as the database is not shown. A comparison of figure 6 and 2 might facilitate the understanding of the classes in a more general demonstration.
5.6.7 Add a range selector to enable zoom and scroll functionalities

A range selector (scrollbar) is added to the chart that allows the user to adjust the visible range. The scrollbar provides the functionalities of zooming and scrolling, which is highly beneficial when displaying large amounts of data. Both axes changes values depending on the displayed data. The value axis (Y-axis) adjusts the range to the maximum value of the visible period, which is an automatic functionality of amCharts. The category axis (X-axis) changes the date format depending on the size of the period that is being displayed. However, these format conversions were specified manually in the code.

Figure 7 shows a relatively small period and therefore displays the dates and highlights new months. Figure 8 covers a larger period and therefore only views the months while highlighting new years. This feature was added to improve the readability.
The following code creates a scrollbar that zoom the chart to view the 15 latest dates as the chart is initiated. If there are less than 15 values in the dataset, amCharts will simply view all available entries without any problem.

```javascript
var chartScrollbar = new AmCharts.ChartScrollbar();
chart.addChartScrollbar(chartScrollbar);
chart.addListener("init", zoomChart);

function zoomChart() {
    chart.zoomToIndexes(chartData[0].length - 15, chartData[0].length - 1);
}
```

![Figure 7. The view of a zoomed in chart](image1)

![Figure 8. The view of a zoomed out chart](image2)
In order to ensure that the currently displayed interval is maintained as a change of dataset occurs, a help function is used that is called after the chart has been animated with new data.

```javascript
function selectDataset() {
    var type = document.getElementById("dataType");
    var value = type.options[type.selectedIndex].value;
    var startDate = chart.startDate;
    var endDate = chart.endDate;

    chart.dataProvider = chartData[value];
    chart.validateData();
    chart.animateAgain();
    remainZoomRange(startDate, endDate);
}

function remainZoomRange(startDate, endDate) {
    chart.zoomToDates(startDate, endDate);
}
```

### 5.7 Software testing

An important part of software development is to ensure and verify that the created software module behaves in an expected manner. Avinode uses the two testing frameworks JUnit and Mockito to test their code. Mockito is used to create fake (mock) objects, where only the parts of a class that will be tested by JUnit must be pre-defined.

Initially, the class that is supposed to be tested must be mocked and the required fields must be specified. Furthermore, the methods are mocked by defining their outcome with the use of Mockito’s `when()` and `thenReturn()` methods. Lastly, JUnit is used to assert that the mocked object behaves as intended. The classes that were tested include `SearchStatisticsDTO` and `SearchStatisticsHelper`. 
6. Result

A module was successfully implemented in the Avinode Marketplace web application that fulfils the previously stated requirements. The result is a chart created with the charting library amCharts that displays search statistics related to the user’s registered aircrafts, of which content can be interactively adjusted. The chart consists of two graphs. The first is a stacked column graph that displays the number of times per day any of the user’s aircrafts appeared in a search result. The second is a line graph that shows how many times the aircraft was clicked on, requested or if it appeared among the first five results.

The user has the ability of selecting what type of data should be displayed with the use of two drop down selectors. One selector controls the type of data that should be displayed in the line graph. The other selector controls if the displayed data should be based on calculations in total or when the aircraft was positioned on its home base or another base as the search occurred.

Other features include a chart cursor, which allows the user to adjust the visible range, as well as an information box that appears if the mouse hovers over a graph. These features were implemented in order to increase the readability of the chart. The final result is shown in figure 9, 10 and 11.

![Figure 9. Balloon of stacked column chart value in a large time interval](image)
Figure 10. Balloon of line graph in a small time interval

Figure 11. Zooming and scrolling of chart
7. Conclusion

7.1 Discussion

7.1.1 General discussion

An important part of this project was to gather information and decide upon which charting library was most profitable for this particular assignment. Due to the fact that the research was prioritised in the initial phase, a favourable and reliable library was found. The research also proved to be an advantage when creating the back-end logic since the required data format as well as the library’s possibilities was known. The coding was therefore performed with respect to the library. Another important research was the study of the Avinode Marketplace system structure to ensure that the module maintained the code standard.

Prior to all phases, a planning was made to increase the awareness of available time. Time limits on problem solving were stated in order not to lose valuable time. If a solution to a problem would require too much inefficient time consumption, another approach would be considered instead. The planning proved to be extremely profitable, since the estimated time plan (see appendix 1) was accurately followed.

There was however some issues regarding the storage of calculated data. Since the chart data had to be in a specific format, a solution that could easily parse the calculated data to the required format was essential. Several approaches were considered, but some appeared to be too complicated and would probably require more time than what was remaining. Therefore, the use of nested maps was considered to be a potential solution since it would most certainly provide a correct result. A disadvantage of this approach is the reduced performance that most likely will be noticeable when handling large amounts of data, since accessing data requires nested for-each loops. This was a known problem, but due to the fact that our module would only process a limited amount of data, the idea was not rejected. Furthermore, the solution seemed to be of suitable level of complexity considering the time that was left.

A decision that may be needed to clarify is that no pie chart was used in the result. Pie charts were included in the criteria of what a library should include. However, this was not a
requirement of the resulting chart itself. It is simply a benefit that the library provides the possibility of including a pie chart if such data visualization is needed.

7.1.2 Environmental and ethical aspects

Environmental aspects must be considered since it is vital that the environment is prioritized in order to sustain a habitable planet for the future generations. The greenhouse effect is one of the biggest threats and our own causes of pollution must be clarified and decreased. It is important that aviators are aware of the effects that some (often elder) aircraft models has on the environment. This might result in that Avinode users with these types of aircrafts are not being requested, which will be revealed to the operator through the search statistics chart. A result of this might be to resign these aircrafts from their fleet and replace them with more environmentally beneficial and profitable aircrafts.

When considering ethical aspects in aviation it is important that safety is of high priority. Avinode cooperates with their sister company Wyvern that is a leader company in aviation safety auditing, consulting and information services. The operators of Avinode have the possibility of purchasing a Wyvern verification and perform a safety inspection with respect to Wyvern’s safety criteria. This provides the Avinode Marketplace with operators that prioritize aviation safety.

7.1.3 Gathering of information

Information that was needed to produce the content of the charting library analysis as well as the technical background was gathered from reliable sources. The references that are used for the research are from the official websites of the charting libraries. The remaining references of the technical background are mainly from reliable sources such as W3Schools and Oracle.

7.2 Further development

In order to improve the solution, other approaches on storing data would be considered. The improvement would increase the performance when handling large amount of data. One alternative solution could involve replacing the nested maps with a custom data structure, in
which the data would be handled in a more efficient way. This would eliminate the need of nested for-each loops and thus improving the performance.

A functionality which was considered to be added but not implemented was that the line graph data should be dependent on the aircrafts. This was left out since it would require a fourth map in the data structure. Currently, the line graph shows the calculated data in total. Therefore, we had to restrict some other functionality, such as dynamically hiding data of aircrafts since the line graph would not be updated as well. Consequently, the line graph will not risk displaying values greater than the total of the column chart. If the line graph data were to be dependent on the aircrafts, it would allow the use of showing desired data only. The user would be able to see how many times each aircraft was requested, clicked and appeared among the first five results. This functionality could be added after a replacement of the current data structure.
References


[26] Microsoft (2014) *Transact-SQL References (Transact-SQL).*


Appendixes

Appendix 1: Gantt chart of estimated time plan

Appendix 2: A detailed analysis of JavaScript charting libraries

Appendix 3: Practical testing of four libraries

Appendix 4: Final version of the database query
Appendix 1:

Gantt chart of estimated time plan

<table>
<thead>
<tr>
<th>Goals</th>
<th>LV1</th>
<th>LV2</th>
<th>LV3</th>
<th>LV4</th>
<th>LV5</th>
<th>LV6</th>
<th>LV7</th>
<th>LV8</th>
<th>LV9</th>
<th>LV10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine chart criteria</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Examine 20 JavaScript-libs, select 4 for further examination</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Further testing of the 4 chosen libraries</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Determine one library suited for solving Avinodes problem</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overview of Avinodes system</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Further/deeper analysis of system</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prepare development environment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simple database fetch (press button -&gt; get data -&gt; view data)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implement a solution for the problem (view statistics)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unit tests with JUnit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Report writing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix 2:

A detailed analysis of JavaScript charting libraries

**amCharts**

amCharts offers several types of charts that are highly customizable and interactive. The interactive features include mouse-over, on-click, zoom and range selector. The charts are rendered in SVG and VML, thus supporting all required browsers. They also run on mobile devices using iOS or Android 4+. Documentation and a large amount of examples and guides can be found on amCharts’ website. They have many customers including NASA, Audi, Samsung and Cisco [32].

A link to amCharts’ support, which has a 100% response rate, is provided on their website and is accessed by signing in with a Google, Facebook or Twitter account. The library can be downloaded for free with the limitation of having an attribution link to the official amCharts website in the top left corner of the charts. A commercial license can be purchased and shared among several developers from $140 (single website) to $700 (multiple websites), which removes the amCharts link and also provides 6 months of priority support [33]. Grade: 9/10

**CanvasJS**

CanvasJS can render 100,000 data-points in 100 milliseconds using the HTML5 Canvas element. It is compatible with most browsers except Internet Explorer version 8 and older, where a plugin would be necessary in order to provide support. They offer several types of charts that are customizable and interactive (mouse-over, on-click and zoom), although the general layout is not as impressive as other libraries. Documentation and dynamically edited examples are provided on their official website [34].

The library is free for non-commercial use under the Creative Commons Attribution-NonCommercial 3.0 (CC BY-NC 3.0) license with the limitation of having an attribution link in the bottom right corner of the chart. The prices of acquiring a commercial license depends on the number of developers and also if priority support is desired, with a lowest possible price of $299 [35]. Grade: 4/10
ChartJS

The charting library ChartJS is developed by DevExpress, which utilizes SVG for graphical rendering. No information regarding other rendering techniques or plugins are mentioned on their website, they do however claim to natively support Internet Explorer 8. ChartJS is free to use for non-commercial projects and requires a 12 month licence fee of $299 for commercial use. They have a well written and structured API with many examples. They have a large variety of charts that are customizable and aesthetic with all desired interactive features (mouse-over, on-click, zoom, range selector). However, the demonstrations of interactivity appear to be rather slow compared to other libraries [36].

In conclusion, ChartJS has many advantages, such as attractive chart appearances and interactive features, although the reaction time is somewhat slow. Furthermore, the API is easy to follow and the code is easily comprehended. It is also supported by all required browsers. The main disadvantage of ChartJS is the pricing for commercial use. Grade: 6/10

Data-Driven Documents (D3.js)

D3.js is an open source charting library offering great extent of different charts rendered in SVG. The library is released under the Berkeley Software Distribution (BSD) 3-Clause license and is therefore free for both non-commercial and commercial use. There are plenty of examples, guides and even books on how to use D3.js. The charts are interactive (mouse-over, on-click, zoom and range selector) and highly customizable due to a data-driven approach of DOM manipulation. Furthermore, several other charting libraries are based on D3.js, such as DC.js, xCharts and Rickshaw [37].

One of the disadvantages of D3.js is that it is not natively supported by Internet Explorer 8 and older versions. Another disadvantage is that D3.js, compared to other libraries, requires a lot of work in order to create simple things. This could potentially be a problem of time consumption when creating or modifying charts. Grade: 8/10
**dc.js**

dc.js is an open source charting library based on D3.js under Apache License version 2.0. It has native Crossfilter support, which is a library that increases effectiveness when working with large datasets. dc.js does not offer as impressive variety of chart options as D3.js does, but the available charts are still customizable and interactive. Documentation and examples are easily accessible. They also have an active Google group for support [38]. Grade: 7/10

**Dygraphs**

Dygraphs is a completely free, open source charting library released under the Massachusetts Institute of Technology (MIT) licence. The charts are rendered with the use of the HTML5 Canvas tag and the ExplorerCanvas (ExCanvas) plugin in order to provide support for Internet Explorer 8+. The library is limited to line charts only and therefore does not fulfil an important requirement of required chart types, hence it is not suitable for Avinode [39]. Grade: 1/10

**Elycharts**

Elycharts is an open source charting library based on Raphaël.js and is released under the MIT license using JQuery. Elycharts complies with the browser compatibility requirements by using SVG and VML. Documentation is accessible, although it is not well informative. The charts look unintuitive and are not as appealing as other library charts. Furthermore, the only available interactive feature is mouse-over [40]. Lastly, the community of Elycharts appears to be limited with a small amount of users, which makes the library seem unreliable. Consequently, there are only ten threads addressing Elycharts on Stack Overflow [41]. Grade: 1/10

**Flot**

Flot is an open source charting library released under the MIT licence. Flot uses the HTML5 Canvas tag for rendering line, bar and pie charts. However, the charts are not natively implemented in Flot, instead it relies on plugins. Plugins are also used to enable interactive features such as mouse-over, on-click and zoom. Flot is compatible with all of the required
browsers by using more plugins. The documentation is written similar to a guide, but the content is divided by relevance, which makes it relatively easy to follow [42].

To Summarize, Flot is an attractive library when it comes to open source alternatives. The excessive use of plugins has both its advantages and disadvantages since the extensibility and customisability increases, although the complexity might increase. Grade: 6/10

**Flotr2**

Flotr2 is an open source charting library based on the library Flotr (not to be confused with Flot). Flotr2 uses the HTML5 Canvas tag and is compatible with all major browsers. FlashCanvas or ExCanvas may be used in order to provide support for older versions of Internet Explorer. The documentation is very limited and does not provide a complete overview of the available features and their capabilities. Moreover, the creators do not offer help through support. Flotr2 does however support all required chart types and some minor interactive features excluding range selectors [43]. Grade: 2/10

**FusionCharts XT**

The company FusionCharts has 22,000 customers including NASA, Google, Apple, Facebook and Microsoft. Their product, FusionCharts XT, provide customizable and interactive Flash and JavaScript charts rendered in SVG and VML. A fee is required for both commercial and non-commercial use with the lowest possible price at $399.00 offering a single non-commercial website for one developer. For each license, there is a month priority support (phone or email) and unlimited access to documentation, tips, articles and how-to-guides. They also have a forum.

FusionCharts XT includes several interactive features such as mouse-over, on-click, drill-down, zoom and the possibility to export the JavaScript charts as images (JPEG, PNG and SVG) or PDF. These features along with their reliability and comprehensive compatibility covering PC, Mac, iPhone and Android devices, including Internet Explorer 6.0+, are great benefits [44].
A disadvantage of FusionCharts XT is that it uses Flash, which is not desired by Avinode. However, it is possible to create pure JavaScript charts. Furthermore, the inevitable pricing for both commercial and non-commercial use is a drawback. They do however offer a free trial version [45]. Grade: 5/10

**Google Charts**

Google Charts is a charting library developed by Google Inc. using the Google Visualization API. It provides several customizable charts with interactive features like mouse-over, on-click, zoom as well as chart range control. The API is well written with several examples. SVG and VML are used as rendering technologies, thus making the library natively supported by all required browsers. Google Charts is free to use, although Google may decline usage if their policy is violated. They have a support forum with over 6000 threads [46]. Grade: 9/10

**gRaphaël**

gRaphaël is an open source charting library under the MIT licence based on the graphics library Raphaël.js. (gRaphaël, 2014) Raphaël.js is a library with the purpose of simplifying the management of vector graphics and uses SVG and VML as rendering techniques [47]. gRaphaël provide the ability to create bar, line and pie charts with some minor interactive features including mouse-over and on-click. gRaphaël supports all modern browsers and has a decent API. Furthermore gRaphaël is well supported with a forum containing over 3000 threads [48].

In conclusion, the large forum is an advantage when it comes to help discussions. However, gRaphaël lacks the interactive functionality of zooming. Grade: 4/10

**Highcharts JS**

Highcharts JS is a charting library developed by a Norwegian company, Highsoft AS, with customers such as NASA, Facebook, Twitter, Ericsson and BBC. Highcharts JS offers a large variety of interactive charts with plenty of accessible examples and a comprehensive API. The charts are customizable and offer mouse-over, on-click and zoom as interactive events. It
utilizes SVG and VML as rendering technologies depending on the browser, thus covering all of the necessary browsers for Avinode’s customers [49].

Highcharts JS is free for non-commercial use under the Creative Commons Attribution-NonCommercial 3.0 license. A purchased developer license is required for commercial use, and the prices are from $90 up to $3,600 depending on the number of webpages and developers, and also whether premium support is desired or not. They offer support through forums and limited free access time to first line support. Some of the benefits of premium support compared to ordinary support is longer free access time and guaranteed initial response time of 36 working day hours [50].

The library fulfils all of Avinode’s requirements with reliability, accessibility, interactivity and compatibility as main advantages. Disadvantages of Highcharts JS include pricing and that the range selector feature is not available, although it is provided in Highsoft’s more expensive product, Highstock JS. Grade: 8/10

jqPlot

jqPlot is an open source jQuery plugin under the MIT and GPL version 2.0 licenses, thus being entirely free of charge. It offers customizable charts rendered in HTML5 Canvas with on-click, mouse-over, zoom and range selector as interactive features. Documentation, examples and a link to a discussion group on Google Drive are found in the official website [51].

Negative aspects of jqPlot are that the layout is not impressive and that it does not natively support Internet Explorer 8. Since there are other free libraries with the same functionalities as jqPlot, but with a nicer layout, the grade of jqPlot is significantly reduced. Grade: 6/10

JS Charts

JS Charts is a charting library by Jumpeye Components, which uses the HTML5 Canvas tag for chart rendering. JS Charts is supported by all the modern browsers, but a plugin is required for Internet Explorer 8. The library is free for non-commercial use with the limitation
of the charts being watermarked. They require a $79 licence fee for commercial use per domain.

The library offers all of the required chart types, all customizable and some charts are available in both 2D and 3D. The available interactive features are mouse-over and on-click. An important aspect when it comes to development is the quality of the documentation. Their API is somewhat confusing, since all of the methods are simply sorted in alphabetical order and not grouped after relevance [52]. Grade: 3/10

**ProtoChart**

Protochart is an open source project under the MIT licence using HTML5 Canvas and prototype.js for the chart visualisation. It is possible to display the data as bar, line and pie charts and Protochart claim to be natively compatible with all required web browsers. However, this project is no longer being under development and the documentation page is down. ProtoChart is therefore not of interest [53]. Grade: 1/10

**Protovis**

Protovis is an open source charting library released under the BSD 2-Caluse licence. However, it is not being further developed. Instead, the team created D3.js, which builds on many of Protovis’ concepts. They both have a rather advanced level of syntax, although their data-driven approach increases the customization possibilities. Protovis only uses SVG to render charts, thus making the library natively incompatible with Internet Explorer 8 [54]. Grade: 1/10

**RGraph**

RGraph is an open source charting library under the MIT license and is therefore free to use. The library provides several types of customizable and interactive charts rendered in HTML5 Canvas. However, the general layout as well as the zoom functionality is not impressive, for the zoom simply enlarges the image instead of actually zooming in.
An advantageous feature, which Canvas enables and which RGraph utilizes, is that graphs can easily be stored as PNG images. A disadvantage is that it is not supported by Internet Explorer 8, although this can quite easily be solved with the use of a plugin such as ExCanvas. Support is handled on an active forum where priority support can be purchased for £199 per year. Documentation and examples are provided on their website [55]. Grade: 2/10

**Shield UI Chart**

Shield UI is a company from Seattle that provides a selection of different products to simplify the design process of web applications. IBM, Boeing, Sony, Philips and Honda are a few of their customers. Their products are available in different programming languages including JavaScript/HTML5, ASP.NET, ASP.NET MVC and Java Apache Wicket. The product which is of interest for this research is the Shield UI Chart JavaScript library. The charts are rendered with SVG and VML and are therefore compatible with all required browsers. The charts are aesthetic, customizable and interactive (mouse-over, on-click and zoom).

Shield UI Chart is free to use for non-commercial use, but a license fee of $299 is required for commercial use. By retrieving a license, the developer is also guaranteed a maximum support response time of 24 hours [56]. Grade: 7/10

**xCharts**

xCharts is a D3.js-based open source charting library developed by tenXer. It provides line and bar charts rendered in SVG with mouse-over and on-click as interactive features. Documentation and a limited amount of examples are found on their website, while support is handled on GitHub. However, the response rate seems to be rather low giving an unreliable impression [57].

xCharts does not offer pie charts or zooming and is therefore not suitable for further testing. Furthermore, since it is not rendered in VML, it lacks the compatibility with Internet Explorer 8. The only advantage is that it is free to use. Grade: 1/10
Appendix 3:

Practical testing of four libraries

amCharts

*Pie chart*

```html
<html>
<body>

<script type="text/javascript" src="http://www.amcharts.com/lib/3/pie.js"></script>

<style>
    #chartdiv { width: 100%; height: 300px; font-size: 14px; }
</style>

<script type="text/javascript">
    var chart = AmCharts.makeChart("chartdiv", {
        "type": "pie",
        "legend": {
            "position": "right"
        },
        "dataProvider": [
            { "country": "Germany", "litres": 165.8 }
        ],
        "country": "Australia", "litres": 139.9
    }, { "country": "UK", "litres": 99 },
    { "valueField": "litres", "titleField": "country" });
</script>

</body>
</html>
```

JS chart by amCharts
Line chart

```html
<html>
<body>
<script type="text/javascript" src="http://www.amcharts.com/lib/3/serial.js"></script>
<div id="chartdiv" style="width: 100%; height: 300px"></div>
<script type="text/javascript">
var chart = AmCharts.makeChart("chartdiv", {
    "type": "serial",
    "dataDateFormat": "YYYY-MM-DD",
    "valueAxes": [{
        "axisAlpha": 0,
        "position": "left",
        "title": "Hours of sleep"
    }],
    "graphs": [{
        "valueField": "value"
    }],
    "categoryField": "date",
    "categoryAxis": {
        "gridPosition": "start"
    },
    "dataProvider": [{
        "date": "2014-04-27",
        "value": 6.2
    }, {
        "date": "2014-04-28",
        "value": 8.0
    }, {
        "date": "2014-04-29",
        "value": 3.5
    }]
});
</script>
</body>
</html>
```
```html
<html>
<body>

<script type="text/javascript" src="http://www.amcharts.com/lib/3/serial.js"></script>

<div id="chartdiv" style="width: 100%; height: 300px; font-size: 11px"></div>

<script type="text/javascript">
var chart = AmCharts.makeChart("chartdiv", {
    "type": "serial",
    "dataProvider": [{
        "country": "USA",
        "visits": 3825,
        "color": "#FF0000"
    }, {
        "country": "Japan",
        "visits": 1009,
        "color": "#FF9E91"
    }, {
        "country": "Germany",
        "visits": 1322,
        "color": "#FCD202"
    }],
    "valueAxes": [{
        "axisAlpha": 0,
        "position": "left",
        "title": "Visitors from country"
    }],
    "startDuration": 2,
    "graphs": [{
        "colorField": "color",
        "fillAlpha": 0.9,
        "lineAlpha": 0.2,
        "type": "column",
        "valueField": "visits"
    }],
    "categoryField": "country",
    "categoryAxis": {
        "gridPosition": "start"
    }
});
</script>
</body>
</html>

JS chart by amCharts

<table>
<thead>
<tr>
<th>Country</th>
<th>Visits</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>3825</td>
</tr>
<tr>
<td>Japan</td>
<td>1009</td>
</tr>
<tr>
<td>Germany</td>
<td>1322</td>
</tr>
</tbody>
</table>

- USA: 3825 visits
- Japan: 1009 visits
- Germany: 1322 visits
```html
<html>
<head>
  <style type="text/css">
    .slice text {
      font-size: 16pt;
      font-family: Arial;
    }
  </style>
</head>
<body>
  <script type="text/javascript" src="http://mbostock.github.com/d3/d3.js2.1.3"></script>
  <script type="text/javascript" src="http://mbostock.github.com/d3.geom.js2.1.3"></script>
  <script type="text/javascript" src="http://mbostock.github.com/d3.layout.js2.1.3"></script>
  
  var w = 300, //width
      h = 300, //height
      r = 100, //radius
      color = d3.scale.category20c(); //builtin range of colors
  
  data = [{"label":"one", "value":20},
          {"label":"two", "value":50},
          {"label":"three", "value":30}];

  var vis = d3.select('body').
    append('svg:svg')
      .data([data])
      .attr('width', w)
      .attr('height', h)
    .append('svg:g')
      .attr('transform', "translate(" + r + " , " + r + ")")
    .selectAll('g.slice')
      .data(pie)
      .enter()
      .append('g')
      .attr('class', "slice")
      .append('path');

  var arcs = vis.selectAll('g.slice')
    .data(function(d) { return d.value; });
  arcs.append('svg:path')
    .attr('fill', function(d, i) { return color(i); })
    .attr('d', arc);

  arcs.append('svg:text')
    .attr('transform', function(d) {
      //set the label's origin to the center of the arc
      d.innerRadius = 0;
      d.outerRadius = r;
      return 'translate(' + arc.centroid(d) + ');
    });
  arcs.append('svg:text')
    .attr('text-anchor', "middle")
    .text(function(d, i) { return data[i].label; });
</body>
</html>
```
Line chart

```html
<html>
<head>
  <script src="http://mbostock.github.com/d3/d3.min.js"></script>
  <style>
    /* tell the SVG path to be a thin blue line without any area fill */
    path {
      stroke: steelblue;
      stroke-width: 1;
      fill: none;
    }
    .axis {
      shape-rendering: crispedges;
    }
    .x.axis line {
      stroke: lightgrey;
    }
    .x.axis .minor {
      stroke-opacity: .5;
    }
    .x.axis path {
      display: none;
    }
    .y.axis line, .y.axis path {
      fill: none;
      stroke: #0000;
    }
  </style>
</head>
<body>
  <div id="graph" class="chart" style="position:absolute;top:0px;left:0; float:left;"></div>
  <!-- implementation heavily influenced by http://bl.ocks.org/1166403 *-->

  // define dimensions of graph
  var m = [80, 80, 80, 80]; // margins
  var w = 880 - m[1] - m[3]; // width
  var h = 480 - m[0] - m[2]; // height

  // create a simple data array that we'll plot with a line
  var data = [4, 7, 2];

  // X scale will fit all values from data[] within pixels 0-w
  var x = d3.scale.linear().domain([0, data.length]).range([0, w]);

  // Y scale will fit values from 0-10 within pixels h-0
  var y = d3.scale.linear().domain([0, 10]).range([h, 0]);

  // automatically determining max range can work something like this
  // var y = d3.scale.linear().domain([0, d3.max(data)]).range([h, 0]);

  // create a line function that can convert data[] into x and y points
  var line = d3.svg.line()
    .x(function(d,i) {
      // return the X coordinate where we want to plot this datapoint
      return x(i);
    })
    .y(function(d) {
      // return the Y coordinate where we want to plot this datapoint
      return y(d);
    })

  (Continued on the next page)
// Add an SVG element with the desired dimensions and margin.
var graph = d3.select("#graph").append("svg")
  .attr("width", w - m[1])
  .attr("height", h + m[0] + m[2])
  .append("g")
  .attr("transform", "translate(" + m[3] + "," + m[0] + ")");

// Create y-axis
var yAxis = d3.svg.axis().scale(x).tickSize(-h).tickSubdivide(true);
// Add the y-axis.
graph.append("g")
  .attr("class", "y axis")
  .attr("transform", "translate(0," + h + ")")
  .call(yAxis);

// Create left y-axis
var yAxisLeft = d3.svg.axis().scale(y).ticks(4).orient("left");
// Add the y-axis to the left
graph.append("g")
  .attr("class", "y axis")
  .attr("transform", "translate(-25,0)")
  .call(yAxisLeft);

// Add the line by appending an svg:path element with the data line we created above
// do this AFTER the axes above so that the line is above the tick-lines
graph.append("svg:path").attr("d", line(data));
```
<DOCTYPE html>
<html lang="en">
<head>
<script type="text/javascript" src="http://mbostock.github.com/d3/d3.js"></script>
</head>
<body>
<script type="text/javascript">
  //Width and height
  var w = 500;
  var h = 200;
  var barPadding = 1;

  var dataset = [ 5, 10, 13, 19, 21, 25, 22, 18, 15, 13 ];

  var svg = d3.select("body")
    .append("svg")
    .attr("width", w)
    .attr("height", h);

  svg.selectAll("rect")
    .data(dataset)
    .enter()
    .append("rect")
    .attr("x", function(d, i) {
      return i * (w / dataset.length);
    })
    .attr("y", function(d) {
      return h - (d * 4);
    })
    .attr("width", w / dataset.length - barPadding)
    .attr("height", function(d) {
      return d * 4;
    })
    .attr("fill", ":2d578b");

  svg.selectAll("text")
    .data(dataset)
    .enter()
    .append("text")
    .text(function(d) {
      return d;
    })
    .attr("text-anchor", "middle")
    .attr("x", function(d, i) {
      return i * (w / dataset.length) + (w / dataset.length - barPadding)
    })
    .attr("y", function(d) {
      return h - (d * 4) + 14;
    })
    .attr("font-family", "sans-serif")
    .attr("font-size", "11px")
    .attr("fill", "white");
</script>
</body>
</html>
```
Google Charts

Pie chart

```html
<html>
<body>
<div id="chart_div"></div>
<script type="text/javascript" src="https://www.google.com/jsapi"></script>
<script type="text/javascript">

  google.load('visualization', '1.0', {'packages':['corechart']});
  google.setOnLoadCallback(drawChart);

  function drawChart() {
    var data = new google.visualization.DataTable();
    data.addColumn('string', 'Topping');
    data.addColumn('number', 'Slices');
    data.addRows(
      [['Salami', 18],
       ['Curry', 28],
       ['Gravel', 30]]);

    var options = {'title':'My Favorite pizza topping',
                   'width':400,
                   'height':300,
                   };

    var chart = new google.visualization.PieChart(document.getElementById('chart_div'));
    chart.draw(data, options);
  }
</script>
</body>
</html>
```

Line chart

```html
<html>
<body>
    <div id="chart_div" style="width: 900px; height: 500px;"></div>
    <script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript">
        google.load("visualization", "1", {packages:['corechart']});
        google.setOnLoadCallback(drawChart);
        function drawChart() {
            var data = google.visualization.arrayToDataTable([
                ['Year', 'Sales'],
                ['2004', 1000],
                ['2005', 1170],
                ['2006', 660],
                ['2007', 1030]
            ]); 
            var options = { 
                title: 'Company Performance' 
            }; 
            var chart = new google.visualization.LineChart(document.getElementById('chart_div'));
            chart.draw(data, options);
        }
    </script>
</body>
</html>
```
Column chart

```html
<html>
<body>

  <div id="chart_div"></div>

  <script type="text/javascript" src="https://www.google.com/jsapi"></script>
  <script type="text/javascript">

    google.load('visualization', '1.0', {'packages':['corechart']});

    google.setOnLoadCallback(drawChart);

    function drawChart() {
      var data = new google.visualization.arrayToDataTable(
        [['Year', 'Sales'],
        ['2004', 10],
        ['2005', 20],
        ['2007', 5]]);

      var options = {'title':'How Much Pizza I Ate',
                      'width':800,
                      'height':400};

      var chart = new google.visualization.ColumnChart(document.getElementById('chart_div'));
      chart.draw(data, options);
    }

  </script>

</body>
</html>
```
Highcharts JS

Pie chart

```html
<html>
<head>
  <div id="container" style="min-width: 310px; height: 400px; margin: 0 auto"></div>
</head>
<body>
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
  <script type="text/javascript" src="http://code.highcharts.com/highcharts.js"></script>
  <script type="text/javascript" src="http://code.highcharts.com/modules/exporting.js"></script>
  <div id="container" style="min-width: 310px; height: 400px; margin: 0 auto"></div>
  <script type="text/javascript">
    $('#container').highcharts({
      title: {
        text: 'Browser market shares at a specific website, 2010'
      },
      series: [{
        type: 'pie',
        name: 'Browser share',
        data: [
          { name: 'firefox', y: 20, },
          { name: 'Chrome', y: 30, },
          { name: 'Internet Explorer', y: 7, }
        ]
      }]
    });
  </script>
</body>
</html>
```

Browser market shares at a specific website, 2010

- Firefox: 20%
- Chrome: 30%
- Internet Explorer: 7%
Line chart

```html
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
<script src="http://code.highcharts.com/highcharts.js"></script>
<script src="http://code.highcharts.com/modules/exporting.js"></script>
</head>
<body>
<div id="container" style="max-width: 700px; height: 400px; margin: 0 auto"></div>
<script type="text/javascript">
$(function() {
  $("#container").highcharts({
    title: {
      text: 'Monthly Average Temperature',
    },
    xAxis: {
      categories: ['Jan', 'Feb', 'Mar', 'Apr', 'May']
    },
    yAxis: {
      title: {
        text: 'Temperature (°C)'
      },
      plotLines: [{
        value: 0,
        width: 1,
        color: '#0088cc'
      }]
    },
    legend: {
      layout: 'vertical',
      align: 'right',
      verticalAlign: 'middle',
    },
    series: [{
      name: 'Tokyo',
      data: [7.9, 6.9, 9.5, 14.5, 18.2]
    }]
  });
});
</script>
</body>
</html>

Monthly Average Temperature

![Temperature Chart](https://highcharts.com)

Tokyo
Column chart

```html
<html>
<body>

<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
<script src="http://code.highcharts.com/highcharts.js"></script>

<div id="container" style="min-width: 160px; height: 360px; margin: 0 auto"></div>

<script type="text/javascript">
$(function () {

    $('#container').highcharts({
        chart: {
            type: 'column'
        },
        title: {
            text: 'Flight hours'
        },
        xAxis: {
            categories: ['Cessna C172R', 'Cessna C172RG', 'Piper PA28']
        },
        yAxis: {
            min: 0,
            title: {
                text: 'Hours'
            }
        },
        series: [{
            name: 'John Smith',
            data: [71.5, 49.9, 106.4]
        }]
    });

</script>

</body>
</html>

Flight hours

![Column chart graph]

- Cessna C172R
- Cessna C172RG
- Piper PA28

John Smith

Highcharts.com
Appendix 4:

Final version of the database query

```java
final JpaContext jpa = this.getJpaContext();
final StringBuilder query = new StringBuilder();
query.append("select s from ").append(EntityDefinition.FlightSearchHitEntity).append(" as s ");
query.append("where s.").append(FlightSearchHitEntityImpl_.aircraft.getName()).append(" in ";
query.append("select a from ").append(EntityDefinition.MarketedAircraftEntity).append(" as a ");
query.append("where a.").append(MarketedAircraftEntityImpl_.marketingCompany.getName()).append(" = :company ");

final List<FlightSearchHitEntityImpl> hits = jpa.createQuery(FlightSearchHitEntityImpl.class, query.toString(), QueryOptionImplImpl.readAll(), new QueryParameter("company", company));

return hits;
```