VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

JunJie Chen

# PORTABLE SALES WEB APPLICATION

Technology and Communication

2014

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

## ABSTRACT

| | |
|---|---|
| Author | JunJie Chen |
| Title | Portable Sales Web Application |
| Year | 2014 |
| Language | English |
| Pages | 70 |
| Name of Supervisor | Ghodrat Moghadampour |

The purpose of the thesis was to implement a portable sales web application, which is capable of running on a tablet, laptop or desktop computer, working as a financial sales tool.

The application request was from FA Solutions Oy, a Finnish software company located in Helsinki, which helps their customers by providing financial IT solution, as they planned to expand their current product set with this application. The name of the application was later defined as FA Sables Mobile.

The application was designed to be used by salesmen from financial firms, who promote financial related products. With the help of the application, they will be able to follow the sales activities and to work with customer oriented sales. In specific, the core functionalities of the application were customer resource management (CRM) features and the ability to start and close a financial proposal or contract on a portable device. Other functionalities include report of salesmen activity, managing documents and configuring new sales processes.

All features of the application mentioned above were achieved; the application has been demonstrated and reviewed by different customers. The result indicated that all planned objective were completed and all requirements met. So far there are several customer deployments and the continuous development is still ongoing as more requirements are raised up by the customers.

Keywords Software engineering, web application, software development methodology

## ACKNOWLEDGEMENTS

The FA Mobile Sales project was started in October 2013, and by March 2014, it has been mostly finished.

Firstly, I would like to give my greatest appreciation to my supervisor Dr. Ghodrat Moghadampour for his support, assistance and suggestions I have been given. He has always been a mentor through my studies in VAMK, teaching us with valuable knowledge and massive passion.

Besides my supervisor, I would like to thank Juha Lehtonen and all my colleges in FA Solutions Oy, for the 2 years I worked with them, I had learnt a lot skills and gained experience that I would not have a chance to get from school.

My sincerely thanks also goes to all the people helped me with my thesis and my school life, I would not be who I am today without all the support.

# CONTENTS

## LIST OF FIGURES AND TABLES

## LIST OF APPENDICES

# 1. INTRODUCTION

The rapid development of the internet in recent decades has brought a huge impact on the world. Like nothing else, the internet as a media speeds up the information flow to almost immediate, and that results in a major improvement to the efficiency of almost all aspect of peopleís lives.

Nowadays the internet becomes more easily accessible, especially people are free to connect to the internet on the go. Wi-Fi connections can be found from many public places such as cafeterias, convenient shops, airports and railway stations. Mobile networks such as 3G and more advanced LTE also cover the most populated areas. This phenomenon contributes to the bloomy growth of active mobile devices connected to the internet. People are getting used to use mobile devices surfing the internet, checking maps and reading news while they are not at home, and with technologies like mobile payment, the way of doing business has been changed.

To take the advantage of the internet and utilize it to help the financial sales process, FA Solutions decided to implement browser-based software that is going to be used by salesmen from financial firms to work with their sales workflow. The software was designed to compound with CRM utilities, business process handling and data analyzing features, and it was to be capable of running on a tablet, laptop or desktop computer. The software was named as FA Sales Mobile.

The main objective of this project was to develop a functional, scalable and user-friendly mobile sales tool which is comfortable to use on a portable device, in order to replace the legacy way of doing the sales where paper document is on a voluntary basis.

The customizability and scalability of the software was considered as the key feature since the targeted users of the software are recognized as small to medium size firms in the private banking and asset management field, and each of them would run different processes for their own interest. Deep customization to the software for every customer could become a problem when customer number grows up, so the configuration for each customer's business process must reside outside the box.

To meet the requirement including making variety financial proposals and contracts, setting reminder for an event and tracking the activities with customers, advanced techniques like Business Process Model Notion and Apache Camel were used to externalize detailed business flow settings out of the standardized software.

# 2. RELEVANT TECHNOLOGIES

FA Mobile Sales is fundamentally a web application following JSR-286 specification (a Java portlet standard specification) and deploys to an enterprise portal as a portlet. Basically it was developed with Java programming language, involving technologies including Vaadin, Spring Framework, Hibernate, MySQL, Liferay Portal, Activiti, etc.The technologies involved to the software will be discussed briefly in the following subheadings.

## 2.1 MySQL Database

MySQL is the most popular Open Source SQL database management system; it is developed, distributed, and supported by Oracle Corporation. /1/

MySQL is a relational database management system (RDMS), written in C and C++, it is accessible from multiple programming languages with language-specific APIs and libraries, including the driver for Java called JDBC. Different features are available from query caching, complex SQL query (sub-selects), cursors and triggers to SSL support and Unicode support makes it a common choice of RDMS for small to medium sized singer-server deployments.

MySQL does not comply with the full SQL standard currently for some of the implemented functionalities, including foreign key references when using some storage engines other than the InnoDB (or third-party engines which supports foreign keys). Triggers are currently limited to one per action / timing. /1/

Just like most other transactional relational databases, MySQL is also strongly limited by hard disk performance. This is especially true in terms of write latency. /2/

MySQL was used as the database management system of FA Sales Mobile, for the reason it is open source and cost free, with good compatibility to different platforms and the stability when operating, and its capability fits to the scale of our usage.

## 2.2 Spring Framework

The Spring Framework is a Java platform providing a comprehensive infrastructure support for developing Java applications. It enables developers to build applications from ìplain old Java objectsî (POJOs) and to apply enterprise services non-invasively to POJOs. This capability applies to the Java SE programming model and to full and partial Java EE. /3/

The Spring Framework is an open source application framework and inversion of control container, the Spring container provides a consistent mechanism to configure the application and integrates with most of Java environments, from small to large. It is composed of several modules that provide a range of services.

1. Inversion of Control

The Inversion of Control (IoC) container manages Java beans from instantiation to destruction through its BeanFactory. The IoC container enforces the dependency injection, leaving the components loosely coupled and allows abstractions of code. So when a Spring managed bean is instantiated, it is injected with the beans defined in the configuration file, hence decoupling the relationship between the bean and its dependency beans to improve the reusability.

2. Aspect-oriented framework

The Spring Framework implements its own AOP framework. AOP terminology is a programming paradigm aiming to increase modularity by separating cross-cutting concerns. For example, an spect is used in FA Sales Mobile to separate the code of logging in the transactional management.

3. Data access

Spring supports all popular data access frameworks in Java, including Hibernate and JPA, and provides a list of features includes: Resource management, Exception handling, Transaction participation, Resource unwrapping and Abstraction. The Spring Framework does not offer a common data access API, instead, full power keeping the supported API intact, which makes it flexible and configurable.

4. Transaction Management

Springís transaction management framework works on the Java platform with an abstractive mechanism. Together with Springís data access framework, one can set up a transactional system without relying on JTA or EJB.

5. Model-view-controller

The Spring Framework features its own MVC framework. Much like Struts and Struts2, Springís MVC is designed around DispatcherServlet, which is responsible for dispatching request to an identified handler, and through configurable handler mapping, view resolution, locale and theme resolution, the request is handled and redirected to the result view.

Other Springís features include Remote access framework, Conversion over configuration, Authentication and authorization, Remote management and Configurable Messaging. Figure 1 shows how Spring is composed of modueles and the layer structure.

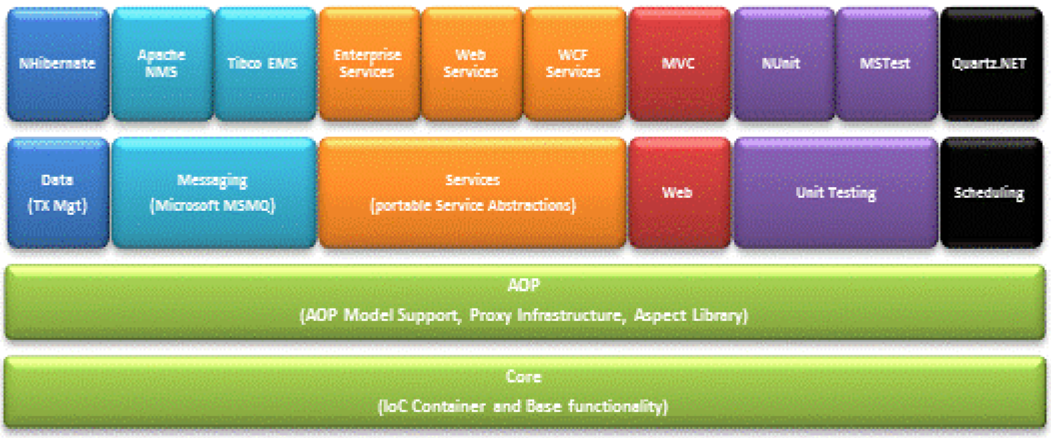

NHibernate
Apache NMS
Tibco EMS
Enterprise Services
Web Services
WCF Services
MVC
NUnit
MSTest
Quartz.NET
Data (TX Mgt)
Messaging (Microsoft MSMQ)
Services (portable Service Abstractions)
Web
Unit Testing
Scheduling
AOP (AOP Model Support, Proxy Infrastructure, Aspect Library)
Core (IoC Container and Base functionality)

**Figure 1**. Spring Framework modules. /4/

Spring was used as the framework and bean container in FAMobile Sales, integrated with JPA, AspectJ and c3p0.

## 2.3 Vaadin

Vaadin Framework is a Java web application development framework that is designed for making the creation and maintenance of high quality web-based user interfaces easy.

It supports two different programming models: server-side and client-side. The powerful server-driven programming model allows developer programs user interfaces much like programming a desktop application with conventional Java toolkits such as AWT, Swing, or SWT. It is also allowed to develop application solely on the server-side, by utilizing an AJAX-based Vaadin Client-Side Engine that renders the user interface in the browser. The client-side model allows developing widgets and applications in Java, which are compiled to JavaScript and executed in the browser. The two models can share their UI widgets, themes, and back-end code and services, and can be mixed together easily. /5/

The list of features Vaadin has are:

1. Comprehensive component framework

Vaadin has a large set of user interface components, controls and widgets, build in with lazy loading. It supports Data binding using MVC (model-view-controller) or MVP (model-view-presenter).

2. Customizable look and feel

Vaadin features a set of good looking predefined themes and styles, it also supports CSS and SASS based component styling, in a configurable fashion.

3. Secure Web Application Architecture

Vaadin runs application codes, validations and business logic runs in the server, with the server-side UI state management and secure parameter and request validation, putting together a strong and secure architecture.

4. Various IDE Tools

Vaadin features a list of IDE plugins and runs perfectly with maven and JUnit.

5. Web Compatibility

This is most important feature to Vaadin, the client-side is based on Google Web Toolkit (GWT) and no browser plugins is needed. Vaadin also supports all major web browsers, and comes with a browser window and tab support, back button support, deep-linking support and URL parameter and fragment handling. It also supports HTML5 Audio support.

6. Java Web Development

Vaadin uses Java as the programming language, so it features all the advantages come along with Java: type-safe, object-oriented web development. It is also compatible with any other JVM language like Groovy, which is useful for different purposes.

7. Extensible widgets based on Google Web Toolkit (GWT)

Vaadin enables the user to create their own implementation of component other than standard ones, the created extensible widgets are based on Google Web Toolkit (GWT). The Vaadin market is the place where user uploads their components and downloadable for other users which can be used as a plugin or ìVaadin Add-onsî.

8. Deployment

Vaadin supports deployment ways as a servlet and a JSR-286 portlet. In our application, we run Vaadin as a portlet in Liferay Portal.

Figure 2 shows the Vaadin architecture and its main components, it also describes how the requests and responses are forwarded and received.

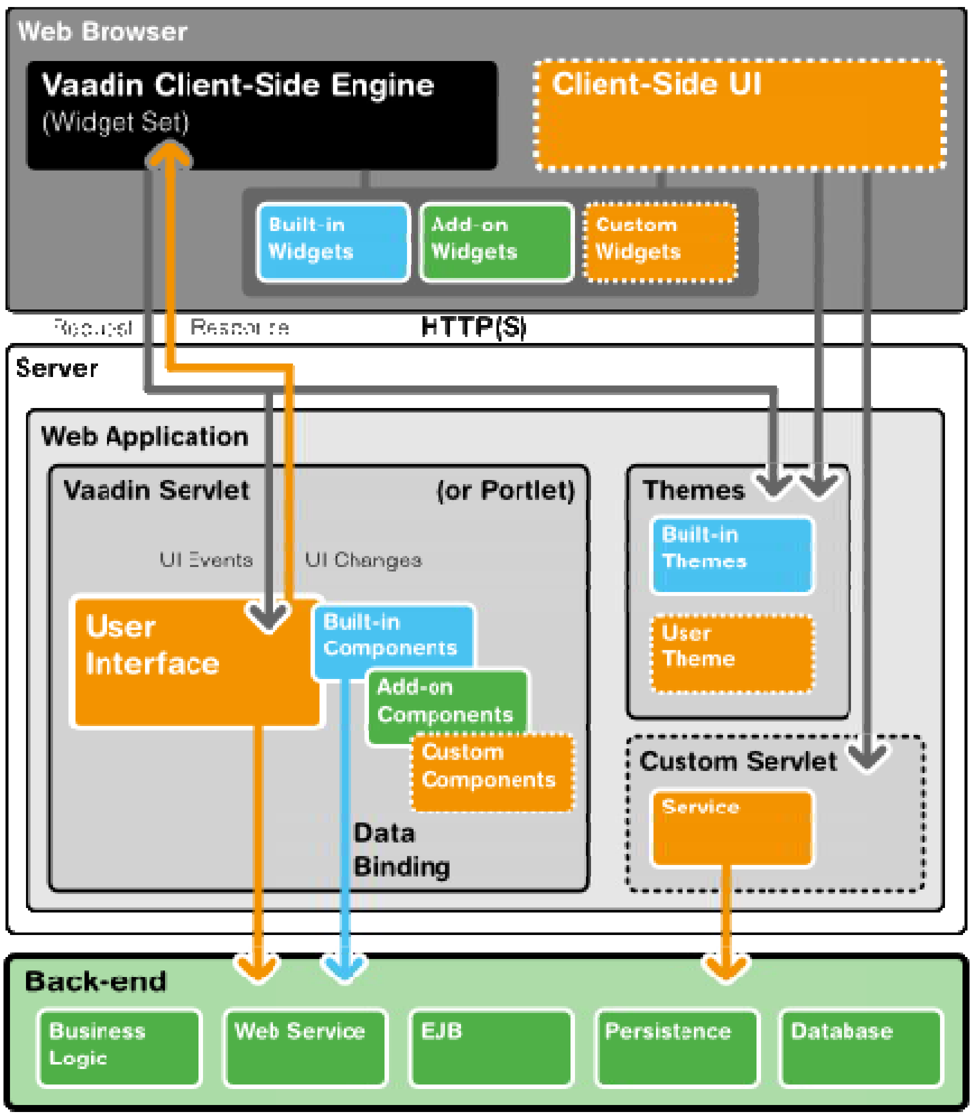


**Figure 2**. Vaadin architecture hierarchy. /5/

A conventional way to develop a rich web application involves the JavaScript programming, communication with Ajax, server-client RPC communication as well as HTML and CSS programming.

Programming based on Vaadin reduces the work on the client (browser) side and enables the developer to focus on the server side implementations (e.g. business logic), and it results a higher output/cost ratio in development. During the practice, we found Vaadin to be flawless and stable.

Because of the features Vaadin has and its good performance, Vaadin was recognized as a suitable tool to be utilized as the rich internet application (RIA) framework for FA Sales Mobile.

Vaadin add-on ìTouchKitî was also used to build a mobile like web application. Vaadin TouchKit powers up Vaadin for creating mobile user interfaces that complement the regular web user interfaces of the applications. Just like the purpose of the Vaadin Framework is to make desktop-like web applications, the purpose of TouchKit is to allow creation of web applications that give the look and feel of native mobile applications. /5/

## 2.4 Activiti

Activiti is a light-weight workflow and Business Process Management (BPM) Platform targeted at business people, developers and system admins. Its core is a super-fast and rock-solid BPMN 2 process engine for Java. It is open-source and distributed under the Apache license. Activiti runs on any Java application, on a server, on a cluster or in the cloud. It integrates perfectly with Spring, it is extremely lightweight and based on simple concepts. /6/

Business process management (BPM) is a concept of aligning an organizationís business process with the needs of the clients; it uses a systematic approach in an attempt to continuously improve business efficiency and striving for innovation, flexibility and technology involvement. Activiti is one of the most open source implementation of BPMN2.0, providing an easy API managing the workflow.

Activiti was used in FA Mobile Sales for multiple purposes. For most, it was used to manage sales activities. We used it to track a business process and its underlying tasks, monitoring different attributes on the task such as assignee of the task, task due date and the state of the task, and since Activiti kept the history of tasks and processes in different states, e.g. finished tasks and processes, this was used to generate reports on sales menís sales activities.

## 2.5 Hibernate

Hibernate is an open-source ORM solution for Java applications. Hibernate provides data query and retrieval facilities that significantly reduce development time. Hibernate lets you develop persistent classes following an object-oriented idiom - including association, inheritance, polymorphism, composition, and collections. Hibernate allows you to express queries in its own portable SQL extension (HQL), as well as in native SQL, or with an object-oriented Criteria and Example API. /7/

Hibernateís primary feature is mapping from Java classes to database tables, and it also provides data query and retrieval utilities. It provides support for all major relational database systems and can be integrated with Spring Framework. It it can also be included as a feature in other programming languages than Java.

Features of Hibernate icludes:

1. Connections Management

Hibernate Connection management service provides an efficient management of the database connections. The database connection is the most expensive part of interacting with the database as it requires a lot of resources of opening and closing the database connection.

2. Transaction management

Transaction management service provides the ability to the user to execute more than one database statements at a time.

3. Object relational mapping

Object relational mapping is a technique of mapping the data representation from an object model to a relational data model. This part of the Hibernate is used to select, insert, update and delete the records form the underlying table. When we pass an object to a Session.save() method, Hibernate reads the state of the variables of that object and executes the necessary query.

Figure 3 describes the architecture of Hibernate in a simple form.

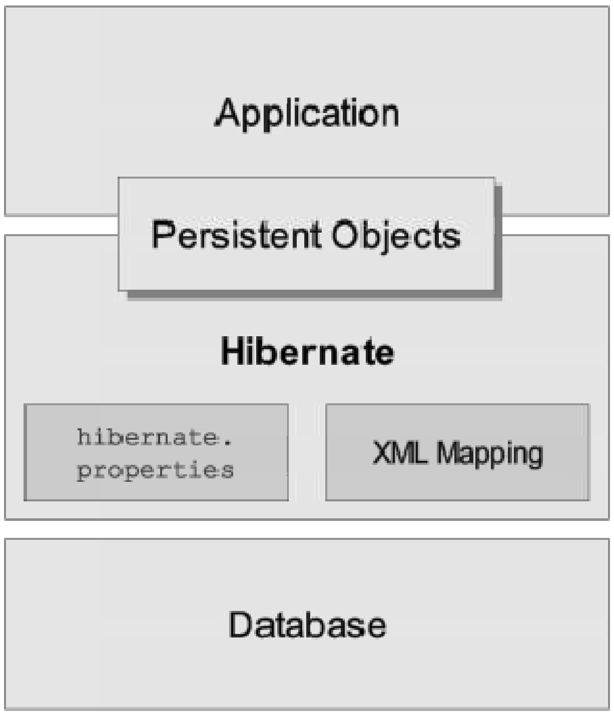


**Figure 3**. Hibernate basic architecture. /8/

To make data access more abstract and portable, we used Hibernate as the ORM framework. It was capable of writing vendor-specific SQL, and maintaining the mapping relationship between Java class and database tables. It was also a credit of its capability of managing the connection pool and good performance in operation.

## 2.6 Tomcat

Apache Tomcat is an open source web server and servlet container, it implements the Java Servlet and Java Server Pages specifications, and provide a HTTP web server for Java code to run in.

Apache Tomcat is a widely used implementation of the Java Servlet Specification, which has been developed as an open-source project by the Apache Software Foundation since 1999, when the project source was donated to the ASF by Sun Microsystems.

Tomcat is actually composed of a number of components, including a Tomcat JSP engine and a variety of different connectors, but its core component is called Catalina. Catalina provides Tomcat's actual implementation of the servlet specification. /8/

As a widely-used free solution for hosting HTML and Java based web applications, Tomcat fits the FA mobile Sales in terms of performance, stability and security. The production servers were running on Apache Tomcat 7.

## 2.7 Liferay

The Liferay Portal is an open source enterprise portal project with features commonly required for the development of websites and portals. It builds in with the CMS system that enables the user to put together a website or portal by assembling themes, pages, portlets / gadgets and a common navigation. It also provides support for plugins that extend into multiple programming languages, including support for PHP and Ruby portlets.

Liferay supports different operating system with JRE installed to host the JVM and an application server is required to contain the Liferay instance. Officially, a range of web servers are supported including Apache Tomcat, Glassfish and JBoss.

The server provides connectivity and interoperability using an Enterprise Service Bus (ESB), and there are multiple services offered by the servers which are leveraged by Liferay. /9/

Applications and extension can be deployed on the server. Liferay uses a number of technologies including EJB, Spring and Hibernate at its core to offer the various services.

Liferay implements Lucene Search Engine by default and can be configured to extend the SOLR Search Engine which is built on Lucene to extend capabilities to provide clustering, faceted search, filtering with additional enhancements and scalability. /9/

A Portlet Bridge is provided to deploy JSR 168/286 portlets and supports RIA applications.  Liferay contains Language adaptors such as for Python, Ruby and PHP which allows easy integration. /9/

The logical architecture of Liferay Portal framework is described in Figure 4.

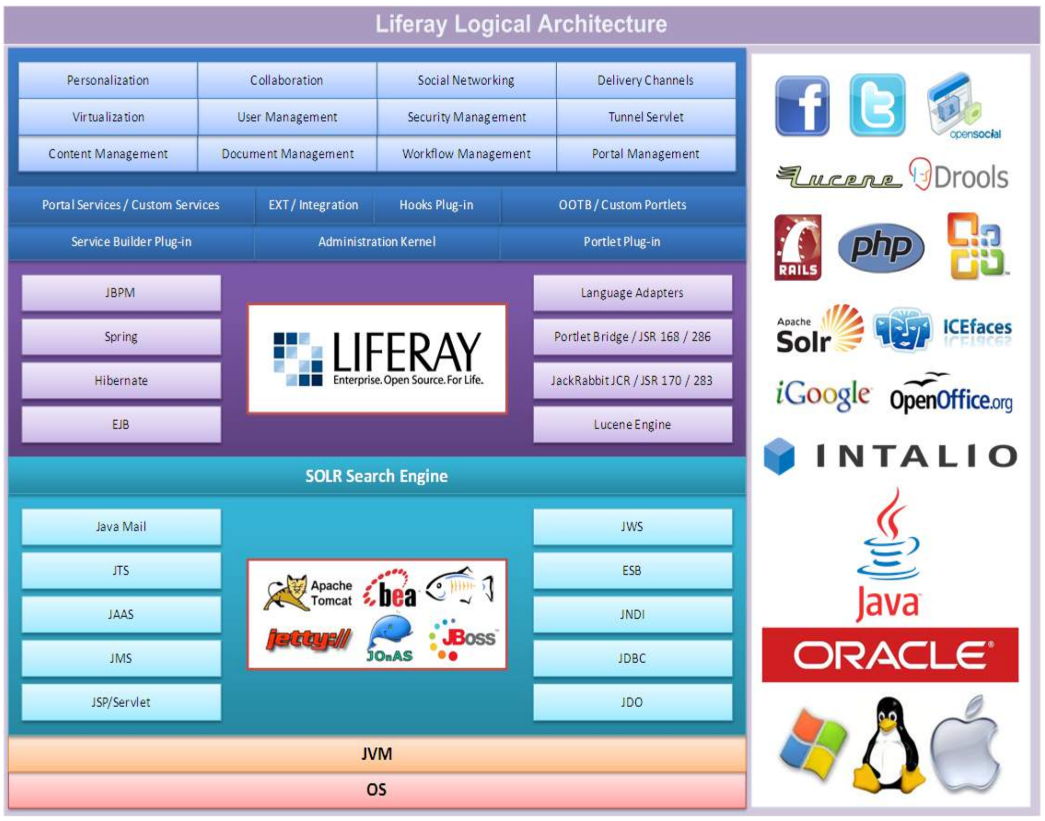


**Figure 4**. Liferay logical architecture. /9/

As a mature platform for websites and portals, Liferay was used for managing the web content and containing the portlets of the FA Mobile Sales. Though during the practice we recognized the merging process between different Liferay versions can be trick to tackle, but it was still a fully capable solution of portal for our application.

# 3. APPLICATION DESCRIPTION

A detailed description of the application is given below covering requirement analyzing, function recognizing, architecture designing and function implementations.

## 3.1 Functional Definition

The requirement of the application from the customer and technical point of view and its designed functions will be explained in this section.

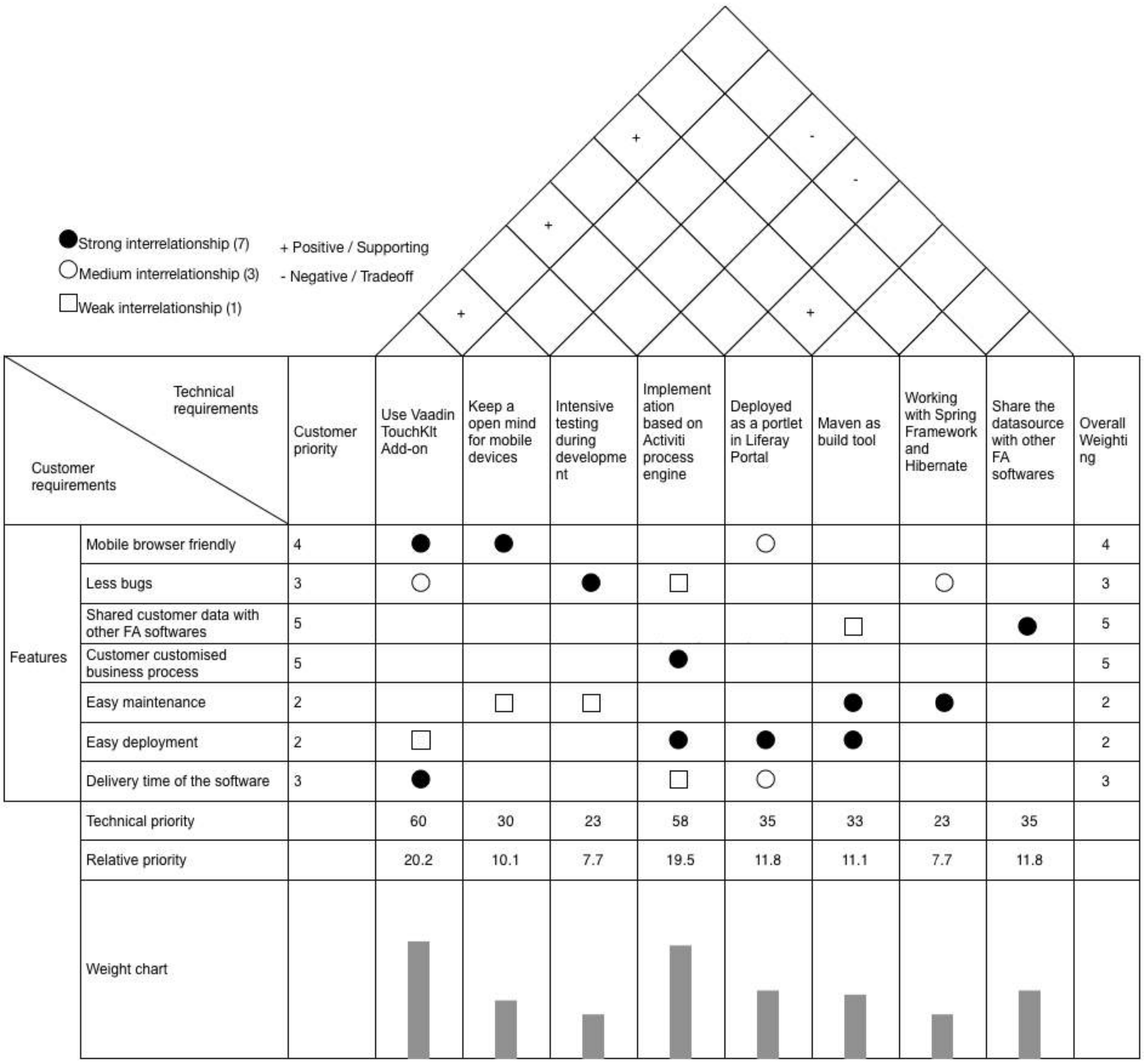
The required features from the customers are recognized as:

1. Mobile browser friendly: The appearance and usability of the software when running on a tablet is important. It needs to be properly designed to offer a good user experience.
2. Fewer bugs: The software needs to be a stable running application even considering different data scale, it should not have fatal flaw when it is released.
3. Shared customer data with other FA software: Since FA Solutions Oy provides other financial software on their service; this software needs to work together with the others, e.g. sharing the same customer data source.
4. Customer customized business process: Customers need to have the ability to create and configure their own business processes against the software. This feature is important for the software to work on different needs from the customer.
5. Easy maintenance and easy deployment: The releasing process and deployment process need to be fast and convenient. The bug fix deployment should be straight forward.
6. Delivery time of the software: Keeping a tight schedule with the customers and be punctual on each step during the development.

To address these issues and to meet the requirements, there are special technical requirements:

1. Use Vaadin TouchKit Add-on: By using the powerful features Vaadin TouchKit Add-on provides, it would be easier and less work to implement a mobile user interface web application with good functionality.
2. Keep an open mind set for mobile devices: An important thing to keep in mind is how the application would be running on a tablet.
3. Intensive testing during development: Testing of the application should reduce the chance of the existence of bugs after the software has been delivered to customers. There are different test phases to go through, including unit test, black box test, white box test and customer acceptance test.
4. Implementation based on Activiti engine: To implement the software based on Activiti engine, we can take advantage of the services that Activiti provides which allows business processes to be defined in .xml files externally.
5. Deploy as a portlet to Liferay Portal: Liferay Portal is a mature enterprise portal which contains and manages portlets very well, to deploy the software as a portlet to Liferay Portal is a big plus from the maintenance and deployment point of view.
6. Maven as a build tool: Maven helps in many ways during the development and deployment of the software, different features of Maven can be used to improve the implementation process.
7. Working with Spring Framework and Hibernate: Both Spring Framework and Hibernate are mature and stable open source framework, while Spring manages the application context and beans, Hibernate controls the data source connections and queries.The two technologies are used for easing the development.
8. Share the data source with other FA software: A key factor that affects the functionality of the software and how it should be working with other FA software.

The relationship between the customer requirements and the technical requirements are further described in the Figure 5 below.


●Strong interrelationship (7)
○Medium interrelationship (3)
□Weak interrelationship (1)
+ Positive / Supporting
- Negative / Tradeoff
+
+
+
-
-
+
Weight chart

| | | Technical requirements / Customer requirements | Customer priority | Use Vaadin TouchKlt Add-on | Keep a open mind for mobile devices | Intensive testing during development | Implementation based on Activiti process engine | Deployed as a portlet in Liferay Portal | Maven as build tool | Working with Spring Framework and Hibernate | Share the datasource with other FA softwares | Overall Weighting |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Features | Mobile browser friendly | 4 | ● | ● | | | ○ | | | | 4 |
| | Less bugs | 3 | ○ | | ● | □ | | | ○ | | 3 |
| | Shared customer data with other FA softwares | 5 | | | | | | □ | | ● | 5 |
| | Customer customised business process | 5 | | | | ● | | | | | 5 |
| | Easy maintenance | 2 | | □ | □ | | | ● | ● | | 2 |
| | Easy deployment | 2 | □ | | | ● | ● | ● | | | 2 |
| | Delivery time of the software | 3 | ● | | | □ | ○ | | | | 3 |
| | Technical priority | | 60 | 30 | 23 | 58 | 35 | 33 | 23 | 35 | |
| | Relative priority | | 20.2 | 10.1 | 7.7 | 19.5 | 11.8 | 11.1 | 7.7 | 11.8 | |

**Figure 5**. QFD diagram.

As seen in Figure 5, using Vaadin Touchkit Add-on, applying Activiti engine and sharing the database with other FA software are the most important technical requirements contributing to satisfying customersí request, they are in the priority to be achieved.

Figure 6 shows the use case definition of FA Sales Mobile, it includes all the actions may act by a user, user management features like update/delete user accounts and user group, privileges settings and permission settings are provided by Liferay Portal which contains the application mounted in the deployment server.

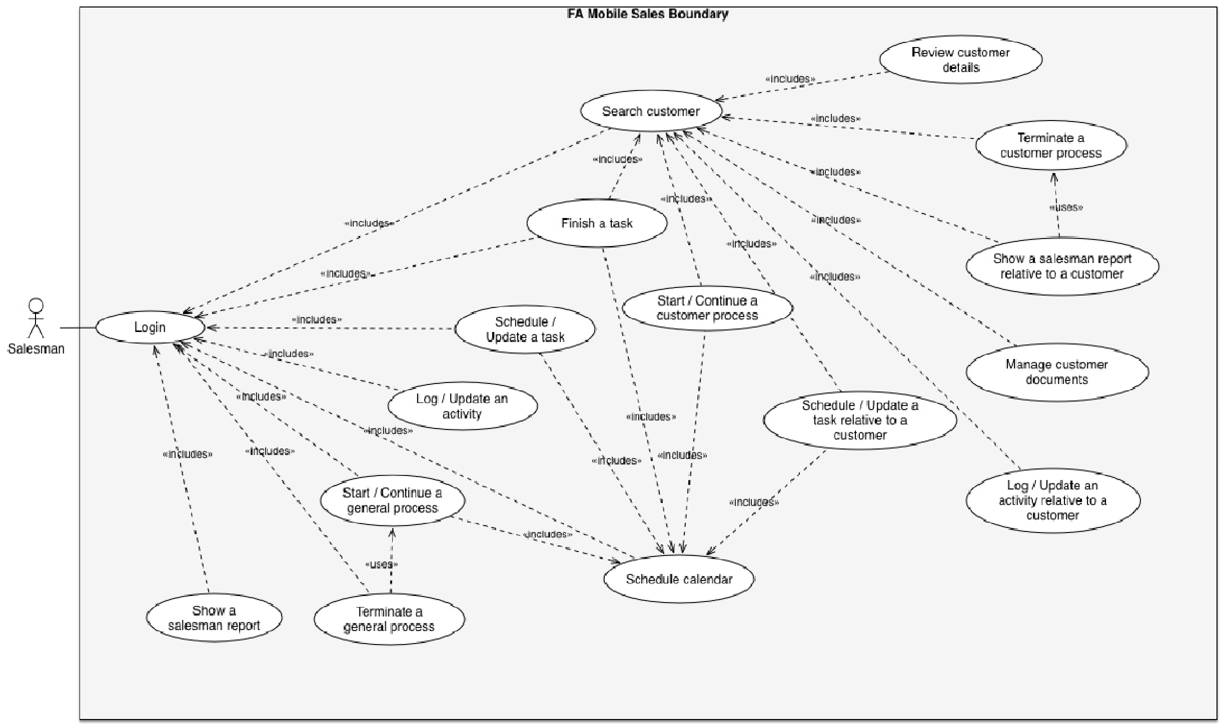

FA Mobile Sales Boundary
Salesman
Login
Search customer
Review customer details
Terminate a customer process
Show a salesman report relative to a customer
Manage customer documents
Log / Update an activity relative to a customer
Finish a task
Start / Continue a customer process
Schedule / Update a task
Log / Update an activity
Schedule / Update a task relative to a customer
Start / Continue a general process
Schedule calendar
Show a salesman report
Terminate a general process
«includes»
«uses»

**Figure 6**. Use case diagram.

1. Log, update an activity: Logs an activity that happened in the past, e.g. a seminar.Name and date of the activity are mandatory fields. The logged activity will be appeared in the activity list in the overview tab, descending ordered by date.

2. Log, update an activity relative to a customer: In addition to Log / update an activity, this function asks for a selected customer, a customer activity can be something that happened with a particular customer in the past, e.g. emailed John2 days ago about reassessing the asset.

3. Schedule, update a task: Similar to log an activity, but a task is a future thing that is to be done, e.g. attend a lunch meeting tomorrow.Name and due date of the task are mandatory fields, the scheduled task will be appeared in the task list in the overview tab, descending order by date, an overdue task is marked as red and appeared at the top.

4. Schedule, update a task relative to a customer: In addition to Schedule / update a task, this function asks for a selected customer, a customer task is a future thing to be done with a particular customer, e.g. text message Mary tomorrow afternoon about the result of the assessment.

5. Finish a task: A task can be marked finished once finished; it will transfer the finished task to be an activity and got recorded.

6. Start, continue a general process: A general process can be a sales process that does not require a customer to be started, e.g. a process that starts a seminar and monitoring the agenda of the seminar. The user can start a general type process from the overview tab.

7. Start, continue a customer process: A customer process differs from a general process because it needs a selected customer to boost, e.g. start a financial proposal for a customer. The customer process can be started from the customer tab with a custom selected.

8. Terminate a general process: A general process can be terminated in the task view, once a process is terminated; all relevant tasks are to be deleted.

9. Terminate a customer process: A customer process can be terminated similar to a general process.

10. Schedule calendar: Schedule calendar is a calendar containing all scheduled tasks assigned to the current user; the tasks showing in the calendar can be accessed to continue or modify.

11. Show a salesman report: This function enables the user to request for a salesmanís report, typically shows activities of a salesman or multiple salesmen by a specified time period.

12. Show a salesman report relative to a customer: Like showing a salesman report, this function shows a report of activities related to a particular customer.

13. Search customer: The user is able to search and filter from a list of customers, the customer list comes from the shared database which is accessible by other FA applications.

14. Review customer details: Customer details are shown once the customer has been selected, details like name, address, representatives, etc., are listed in the page.

15. Manage customer documents: Documents are files uploaded to the server per customer, for example introduction documents and signed proposals. The entry to upload documents is available from the customer tab, on tablets it will bring up the camera to take a photo. Other maneuvers that can be performed are delete and rename the uploaded files.

## 3.2 Application Class View

In the following section the class hierarchy structure and the responsibility of classes are discussed.

Figure 7 is the class diagram of FA Mobile Sales. There are 10 entity classes in the diagram.

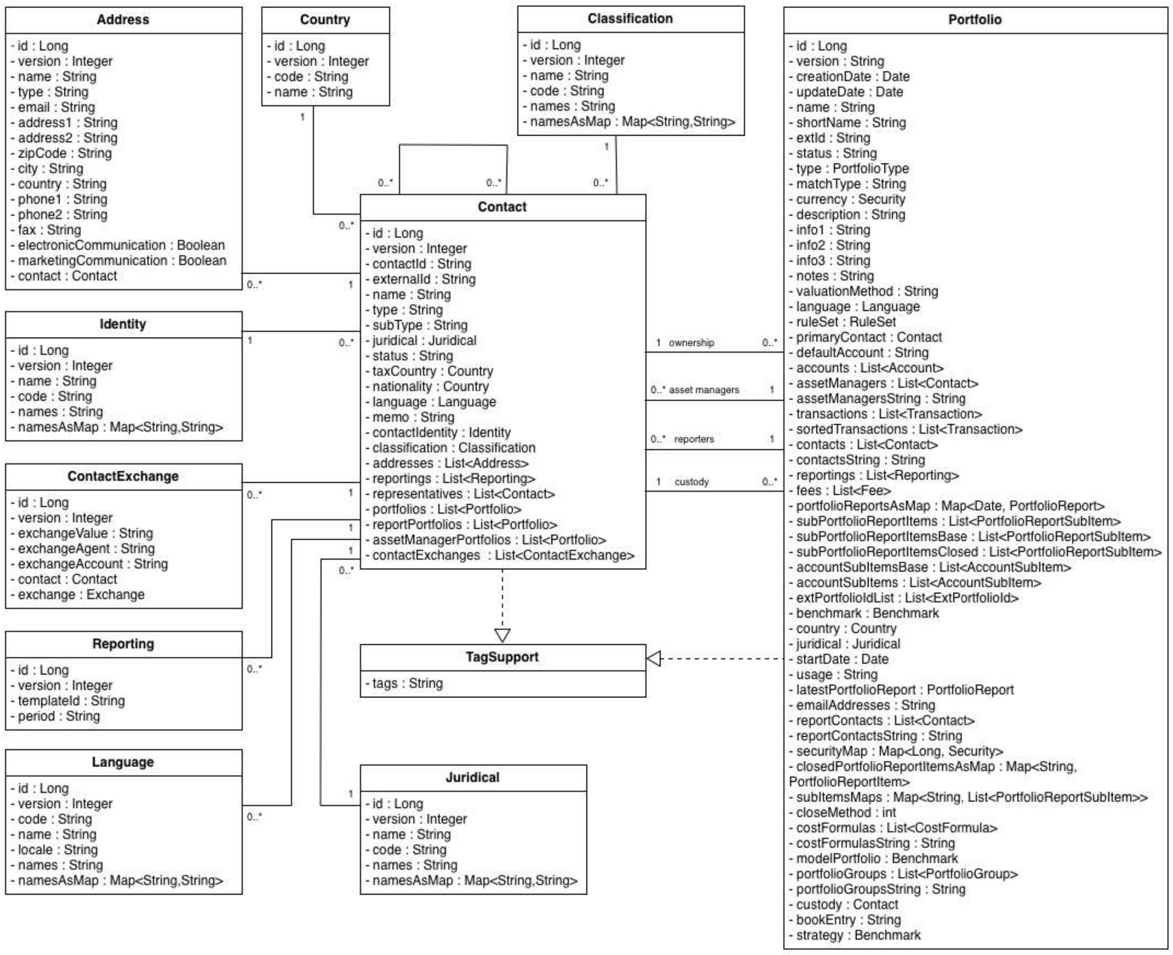


**Figure 7**. Class diagram.

1. Contact class: Contact class is the entity class that represents a contact person. There are multiple types a contact could be, most common one is a customer, but it can be a different type for example customer representative or custody. The attributes of Contact class includes name, id, country, etc.
2. Portfolio class: Portfolio class is the entity class that represents a portfolio. The portfolio object contains the information, such as its composition and current market values etc. A portfolio is owned by a customer, that relationship is represented as a 1-n link between the contact and the portfolio class. But the portfolio could also link to other types of contact, such as custody and asset manager; these relationships are all marked in the figure.
3. Classification class: Classification class represents the classification the contact may belong to, linked to Contact class in a 1-n relationship.

4. Country class:Linked to Contact class in a 1-n relationship, represents the country.
5. Address class: It is the entity class of address, contains mainly contact information of the contact. It is linked to the Contact class in an n-1 relationship, since multiple addresses can be added to a contact person.
6. Identity class: Represents the identity attribute of the contact person. Linked to Contact in a 1-n relationship.
7. ContactExchange class: Represents the exchange attribute the contact person may have. Linked to Contact in an n-1 relationship.
8. Reporting class: The reporting templates to be used when generating report to the contact. Linked to Contact in an n-1 relationship.
9. Language class: Entity class for a language, linked to Contact in a n-1 relationship, contact person may be eligible of more than 1 languages.
10. Juridical class: Represents the juridical information that is linked to contact, with a 1-n relationship.
11. TagSupport class: The class that is extended by Contact and Portfolio class. Containing only one field ì tagsî, it extends the child class entity with tagging features.

## 3.3 Event Sequence

The sequence diagrams describe the operation of the process and how the message has been passed in a time sequence.

### 3.1.1 Start a process

Figure 8 describes the event sequence for staring a process.

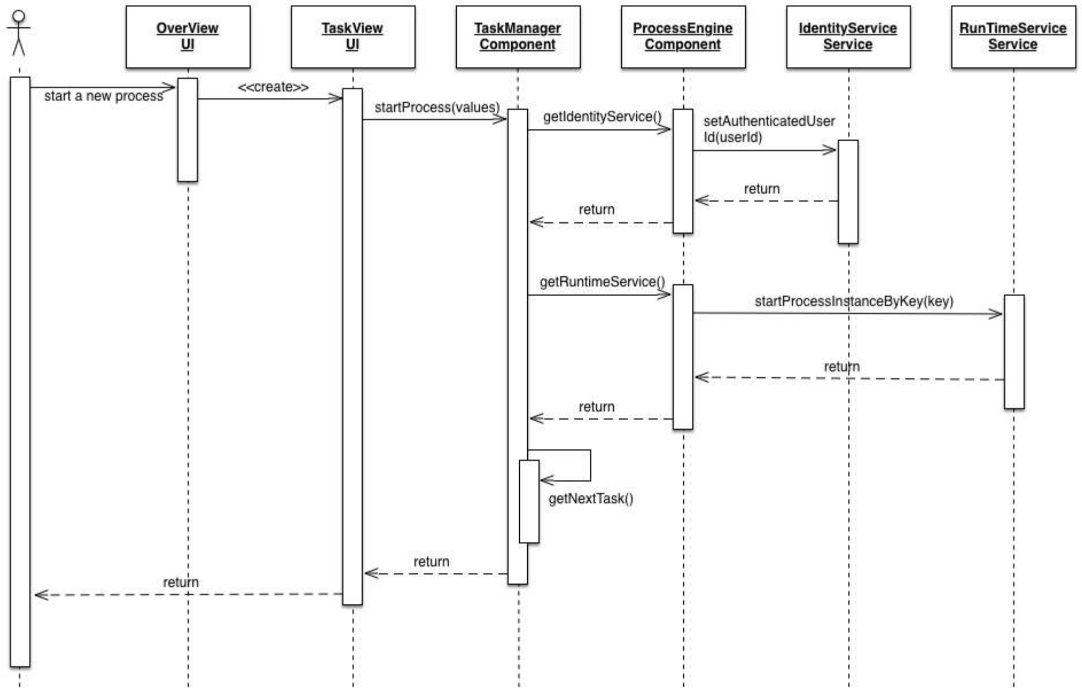


**Figure 8**. Sequence diagram of starting a process.

1. The user interrupts with the software by clicking the new process button in the user interface.
2. When the request to start a process is established, a TaskView object is created.
3. The request message is send to the TaskManager with the starting parameters, requesting to start the given process with the parameters.
4. The Task Manager assigns the initiator of the process to be  current user by calling method  IdentityService.setAuthenticatedUserId().
5. The TaskManager starts the process by calling method RunTimeService.startProcessByKey().
6. A ProcessInstance object is returned after the process is started, representing the started process instance.The TaskManager hands over the object and looks for the next task from the process by calling a self-contained method getNextTask().
7. Once the next task is received, it is passed to TaskView, to construct the task form view.
8. The task form is presented on the user interface.

### 3.1.2 Schedule a task

Figure 9 shows the event sequence to schedule a task.

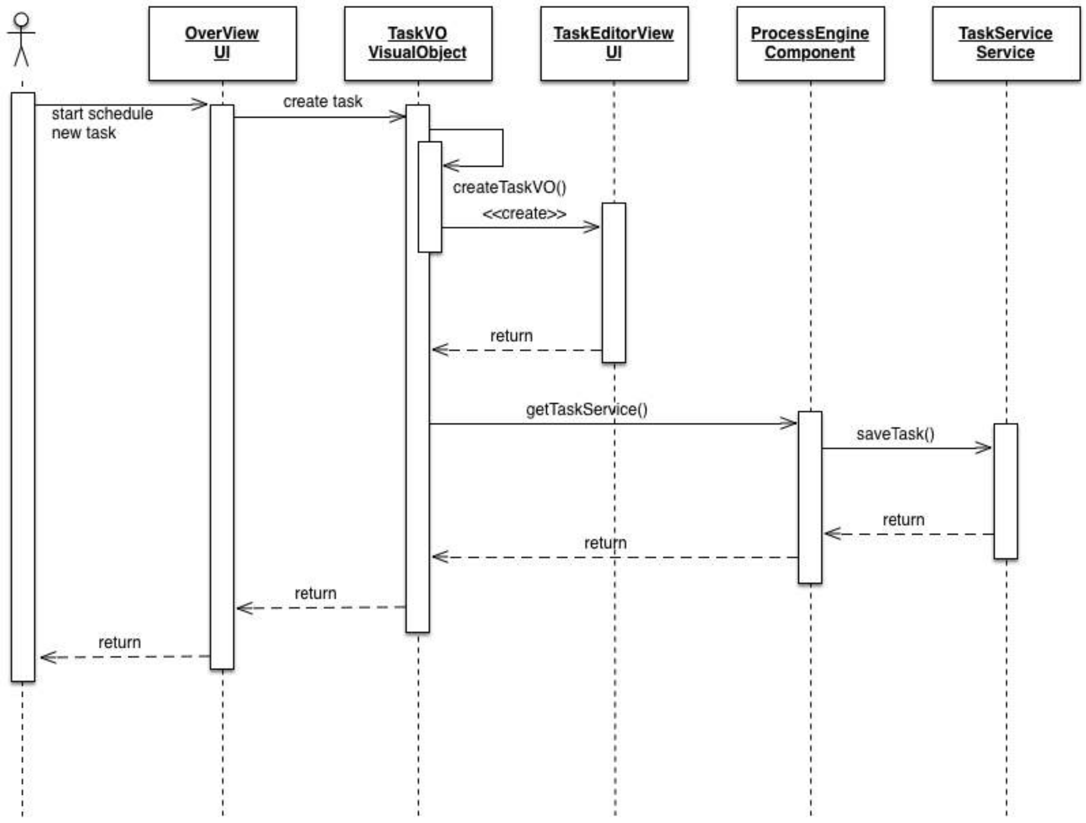


**Figure 9**. Sequence diagram of scheduling a task.

Figure 9 describes the event sequence of scheduling a task, the same sequence pattern is applied to logging an activity as well.

1. The user starts to schedule a new task from use interface.
2. A TaskVO object in instantiated by calling method TaskVO.createTaskVO()
3. The TaskVO object is passed to TaskEditorView for editing, attributes of the task can be modified from the user interface.
4. When editing the task is finished, TaskService.saveTask() method is called to save the task.
5. After saving successfully, a notification is returned and displayed.

### 3.1.3 Search customer

Figure 10 describes the event sequence of fetching the list of customers that makes up the customer search component.

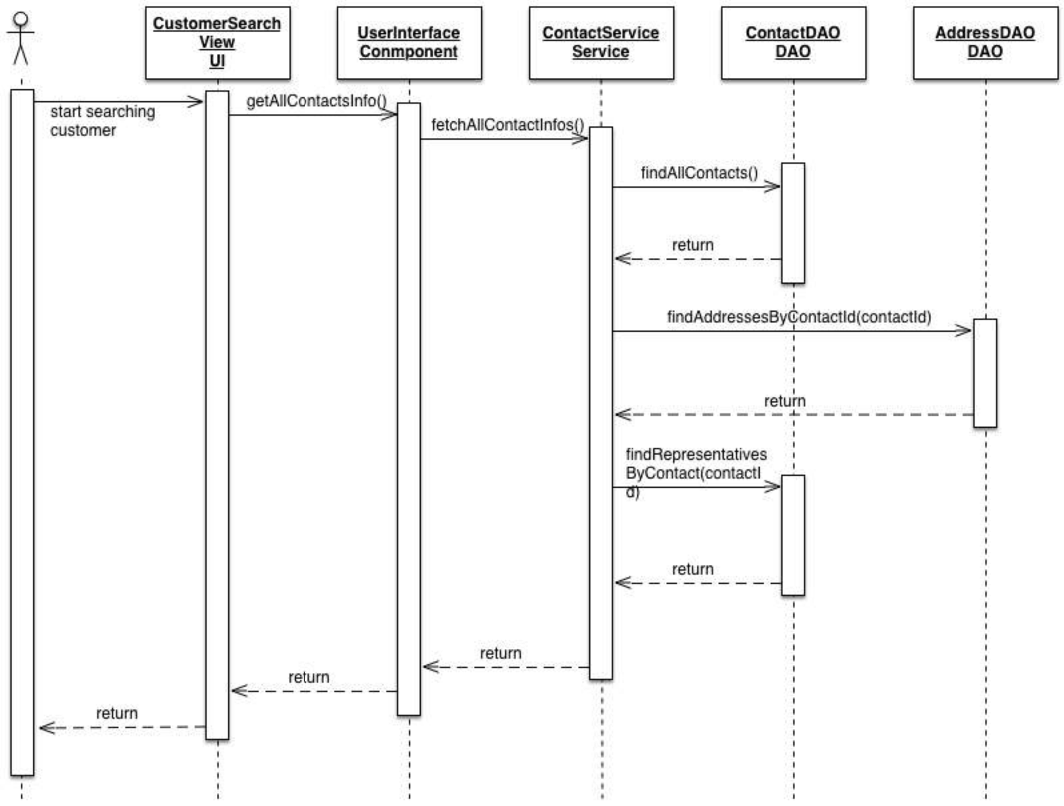


**Figure 10**. Sequence diagram of showing customer search.

1. When CustomerSearchView is instantiated, it asks UserInterface component for the customer list.
2. To load customers from database, UserInterface component calls ContactService.fetchAllContactInfos().
3. ContactService calls ContactDAO.findAllContacts() to get the list of all the customers.
4. ContactService calls AddressDAO.findAddressesByContactId() to get addresses for each contact and assign it to the contactís attribute.
5. ContactService calls ContactDAO.findRepresentatives() to get the contactís representatives and assign back the value.
6. The customer list is returned to CustomerSearchView, wherea table containing all customers is constructed.

### 3.1.4 Schedule calendar

Figure 11describes the event sequence of showing the schedule calendar.

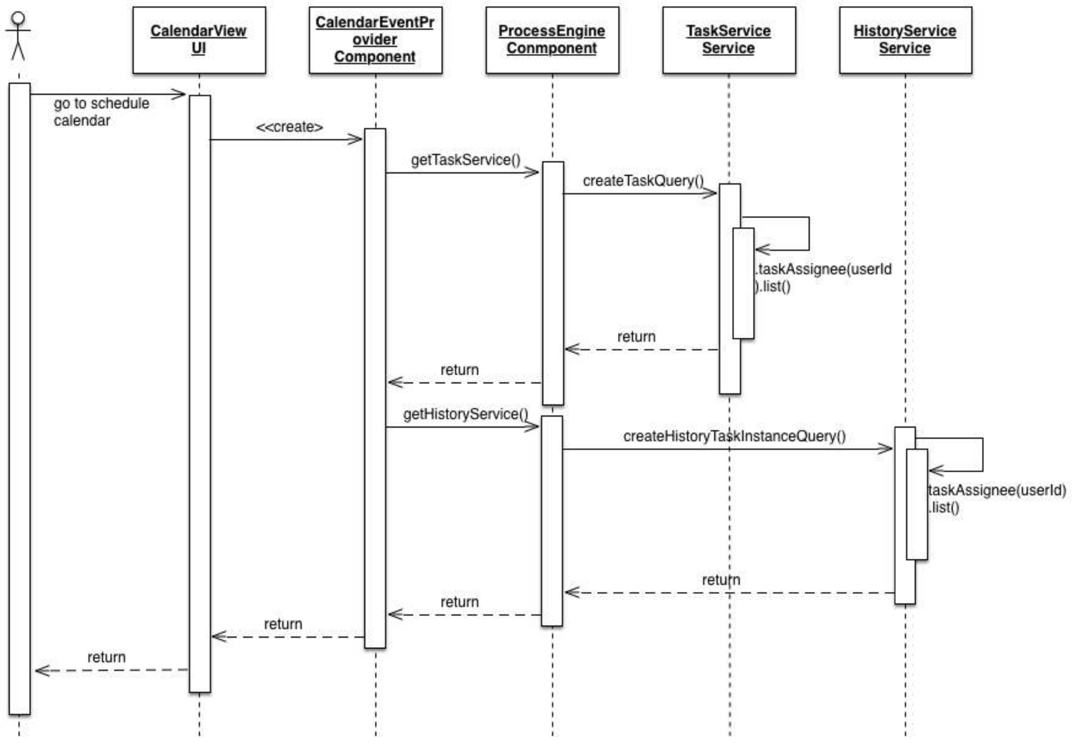


**Figure 11**. Sequence diagram of showing schedule calendar.

1. When EventCalendar is instantiated, it creates a CalendarEventProvider to fill the calendar with content.
2. TaskService.createTaskQuery().taskAssignee(userId).list() is called to fetch all running tasks corresponding to the current user.
3. HistoryService.createHistoryQuery().taskAssignee(userId).finished().list() is called to fetch all finished tasks corresponding to the current user.
4. The lists of tasks are merged and returned to CalendarEventProvider, tobe filled into the calendar in the Calendar View.

## 3.2 Application Components

The applicationís components structure will be discussed in this section.

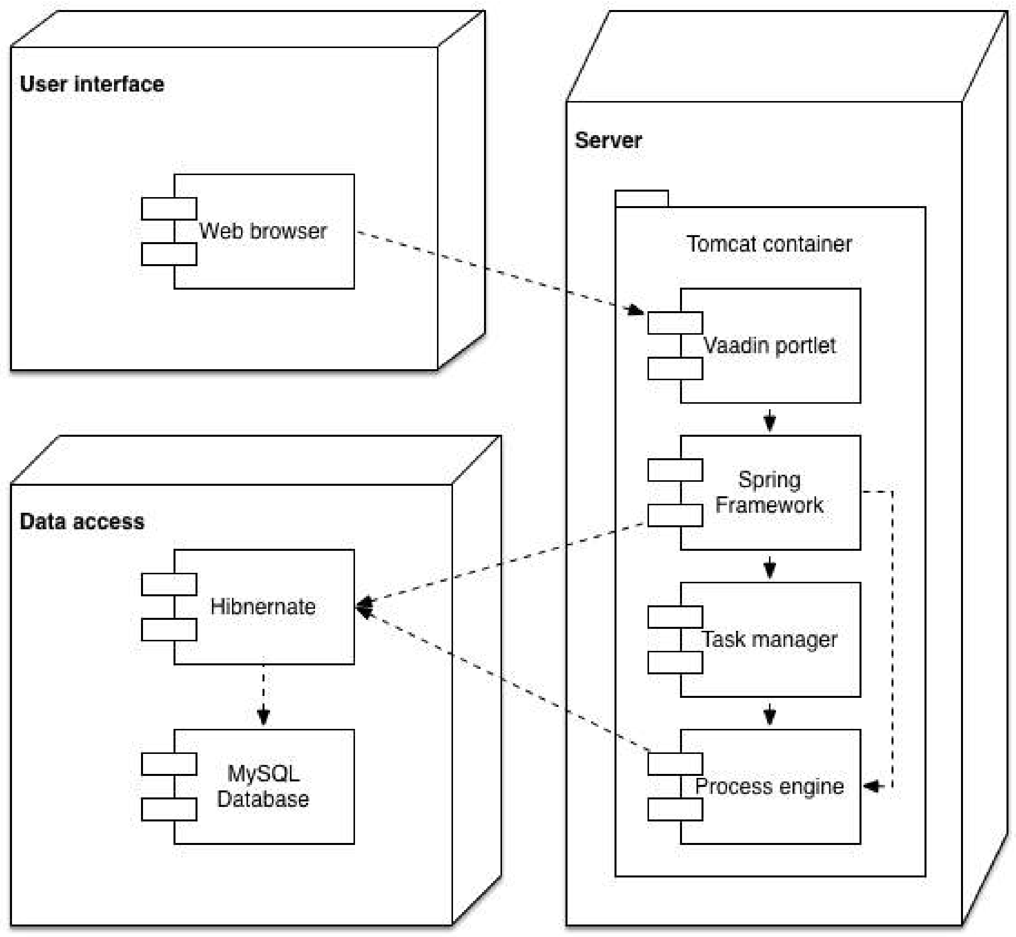


**Figure 12**. Component diagram.

Figure 12 is the component diagram of FA Sales Mobile, As it is stated in the diagram, the application is based on 3 major components: UI module, server module and Data access module. Each module is in charge of a specified duty in the application.

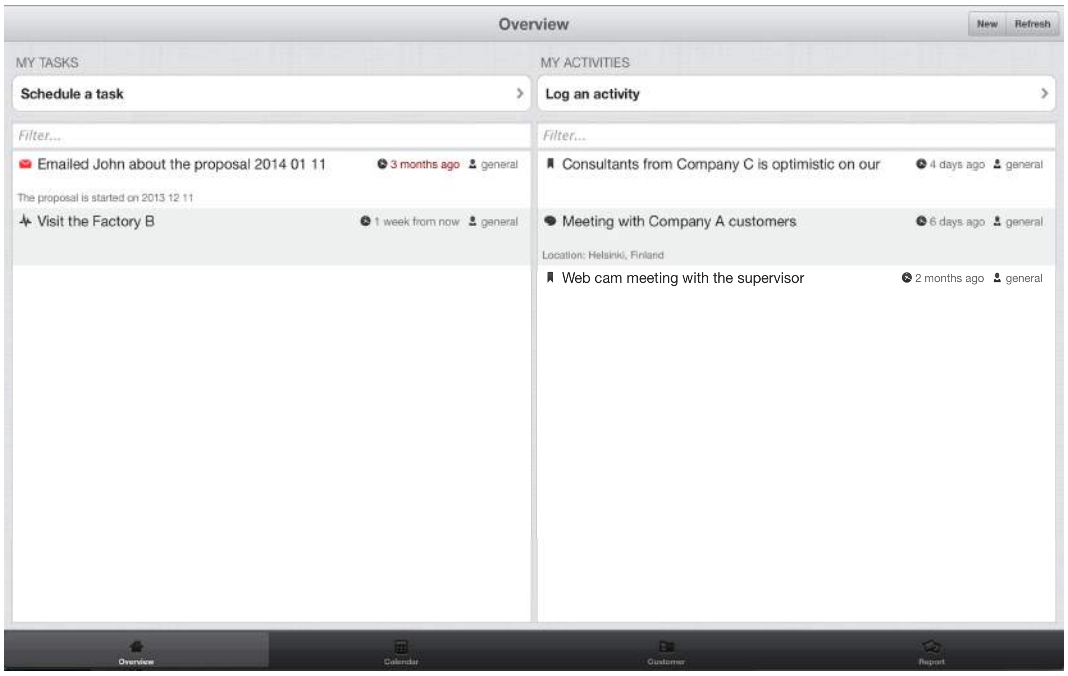
The user interacts with the  application directly through a web browser, and according to the type of the command given by the user, the request is sent to the server end for further processing. The Vaadin portlet is at the first place receiving information from the user interface end, after some simple manipulation, the request is sent by Spring Framework to different parts of the application for different purposes. Hibernate is used for managing database I/O.

# 4. GRAHPIC USER INTERFACE DESIGN

The user interface design had been one of the biggest challenges in the project. To design a user interface that is not only clean and neat but also directly shows to users the entry of each function with consideration on portable devices, we turned to use Vaadin with TouchKit addon. It was not difficult to make the choice, firstly, Vaadin had been proved to be a functional and efficient framework in building dynamic web applications; secondly, as FA Solutions Oy had been developing its software using Vaadin, there is a common experience and reusable code snippet that might help the development; and most importantly, the TouchKit provides a decent mobile app feeling user interface.

## 4.1 Overview Page

The over view page is where the user landed initially when logging in, it has a 2 column layout with a top bar. Both columns contain a button and a list.



Emailed John about the proposal 2014 01 11 — 3 months ago — general
The proposal is started on 2013 12 11
Visit the Factory B — 1 week from now — general

Consultants from Company C is optimistic on our — 4 days ago — general
Meeting with Company A customers — 6 days ago — general
Location: Helsinki, Finland
Web cam meeting with the supervisor — 2 months ago — general

**Figure 13**. Overall page.

On the left column, clicking the ìSchedule a taskî button will navigate the current page to a new task editor page, which allows the user to modify the task details and to save the task. The list under the schedule task button shows all tasks that are assigned to the current user, ordering by the due date in a descending order; overdue tasks are marked in red. Every cell in the list represents a task; the name of the task is shown as the header, with the description shows below. The taskís due date and assignee are shown on the right corner of the cell. The text box between the button and list is used for filtering the list by keywords.

The left column has a similar layout as the left, and basically does the same thing as the components in the left as well, except for it is showing the list for activities and the button ìLog an activityî will allow the user to create and to save an activity.

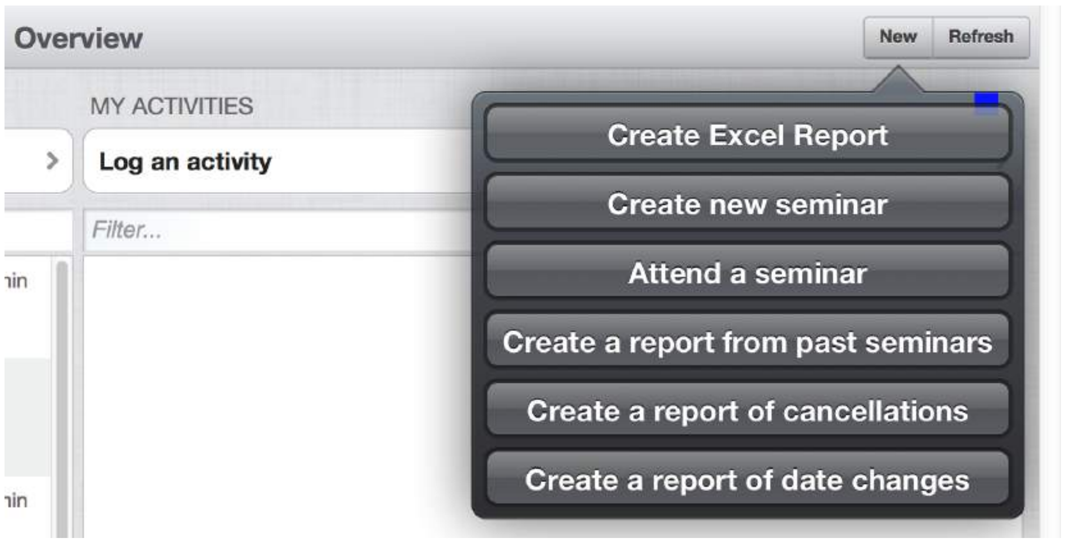


**Figure 14**. Start process menu.

On the right corner of the top bar there are two buttons. ìRefreshî button is used to reload the content of task list and activity list. ìNewî button is the entry for starting new general processes, clicking the ìNewî button will bring up a pop up showing all available processes to be started.

## 4.2 Calendar Page

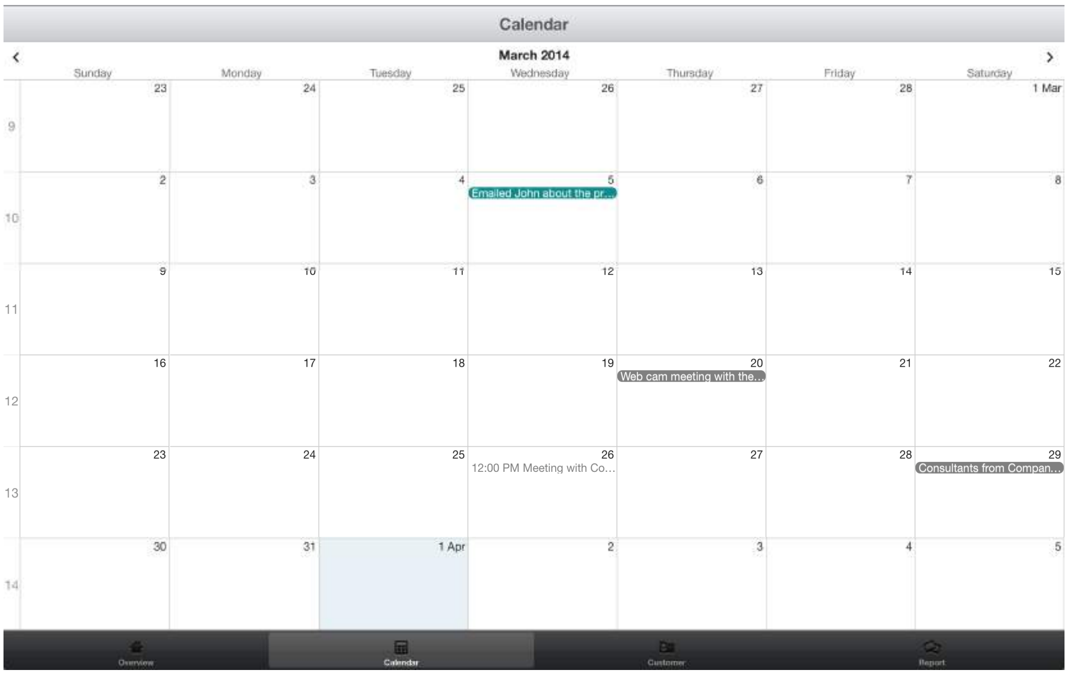
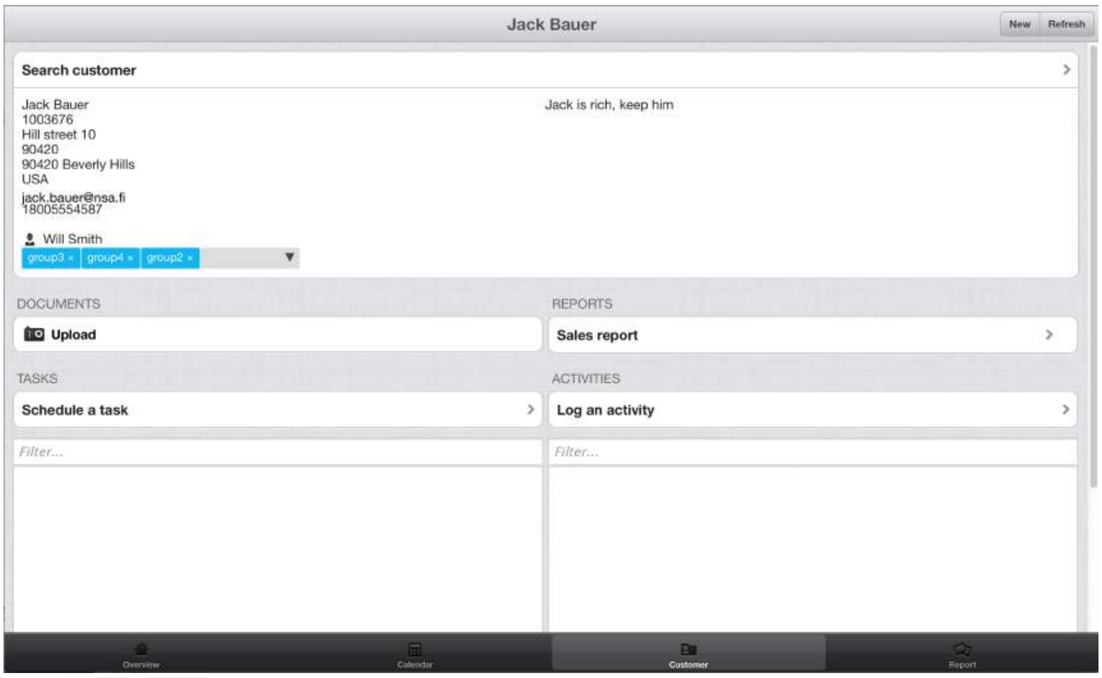

Calendar
March 2014
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Emailed John about the pr...
Web cam meeting with the...
12:00 PM Meeting with Co...
Consultants from Compan...
Overview
Calendar
Customer
Report

**Figure 15**. Calendar page.

The calendar page is the second tab from the tabs on the bottom, opening the calendar tab will navigate the current page to a calendar filled with tasks and activities according to the due date. Tasks which do not have due date specified are not shown in the calendar.

The two arrows aligned with the month name navigates between different months. Clicking on the tasks or activities in the calendar will navigate the current page to the task page or task editor page.

## 4.3 Customer Page



Jack Bauer
1003676
Hill street 10
90420
90420 Beverly Hills
USA
jack.bauer@nsa.fi
18005554587
Will Smith
group3 × group4 × group2 ×

Jack is rich, keep him

**Figure 16**. Customer page.

The customer page is the third tab in the application. The first box in the content shows the customer details e.g. names, contact information, representatives, etc. Clicking the ìSearch customerî button on top will clear the current customer selection and open a customer search page.

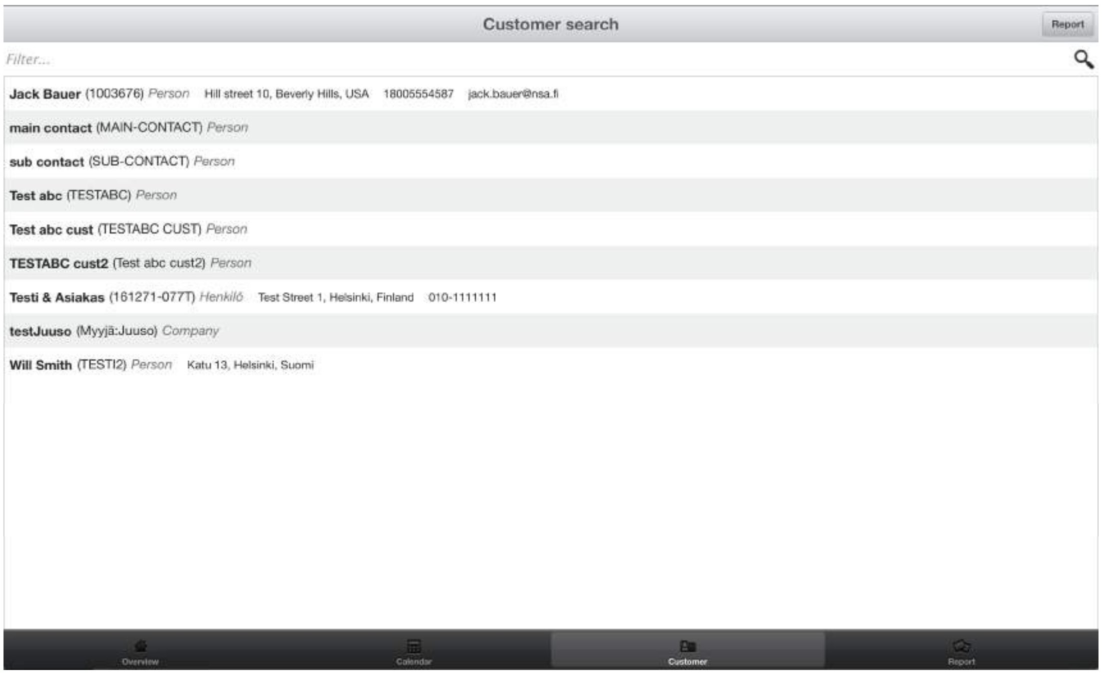
The document box in the left middle of the page is the entry for documents management. Clicking the upload button allows the user to upload a file from the local file system (or on mobile device, taking a picture), the uploaded documents are listed in the list below, and clicking the name will download the document. Clicking the wrench icon on the right of each document allows the user to delete or rename the document.

The reports box in the right middle of the page shows a list of available reports to be generated to the current customer.

The functionality of the customer task lists is similar to that of overview page, but all tasks and activity listed and created on the customer page are linked to the customer.

On the right top corner there are two buttons ìRefreshî and ìNewî, for reloading the current customer and starting a new customer process.

## 4.4 Customer Search Page



**Jack Bauer** (1003676) *Person* Hill street 10, Beverly Hills, USA 18005554587 jack.bauer@nsa.fi
**main contact** (MAIN-CONTACT) *Person*
**sub contact** (SUB-CONTACT) *Person*
**Test abc** (TESTABC) *Person*
**Test abc cust** (TESTABC CUST) *Person*
**TESTABC cust2** (Test abc cust2) *Person*
**Testi & Asiakas** (161271-077T) *Henkilö* Test Street 1, Helsinki, Finland 010-1111111
**testJuuso** (Myyjä:Juuso) *Company*
**Will Smith** (TESTI2) *Person* Katu 13, Helsinki, Suomi

**Figure 17**. Customer search page.

The customer search page is shown when the user visits the customer tab with no customer pre-selected or when the user is about to re-search a customer. The page consists of a text box and a list, where the list displays all the customersí name with details such as id, addresses, mobile number and email address. Typing text into the text box will activate the filter, resulting in the list to be filtered according to the given keyword.

Clicking the reading glass icon on the right will display / hide a box for advance searches, currently only searching by tags is supported in the advanced search box.

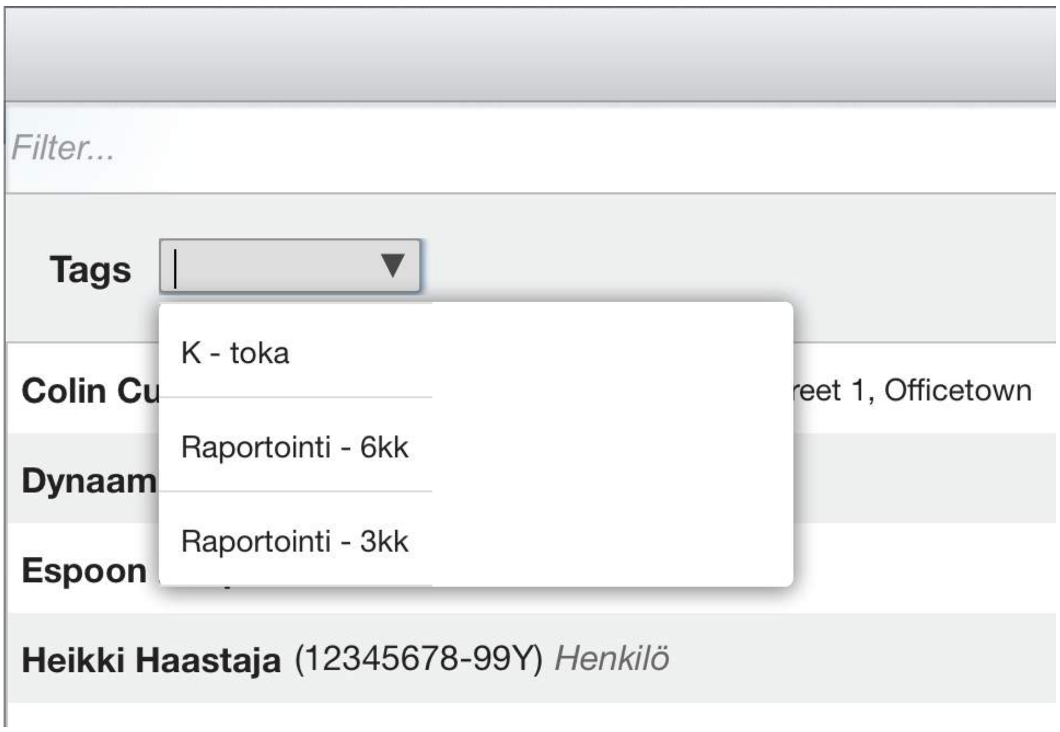


**Figure 18**. Customer advanced search.

There is a ìReportî button on the right top corner, clicking the button will show a pop up containing available reports to be started. Clicking the report button will generate the report based on the current customer search result.

## 4.5 Report Page

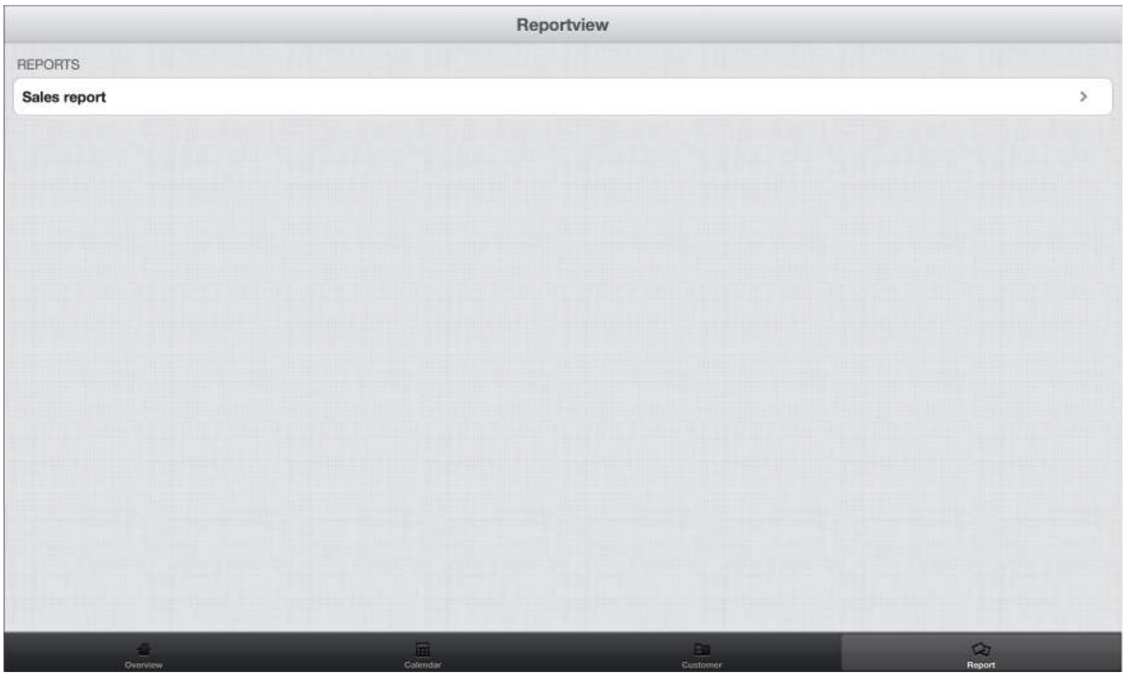
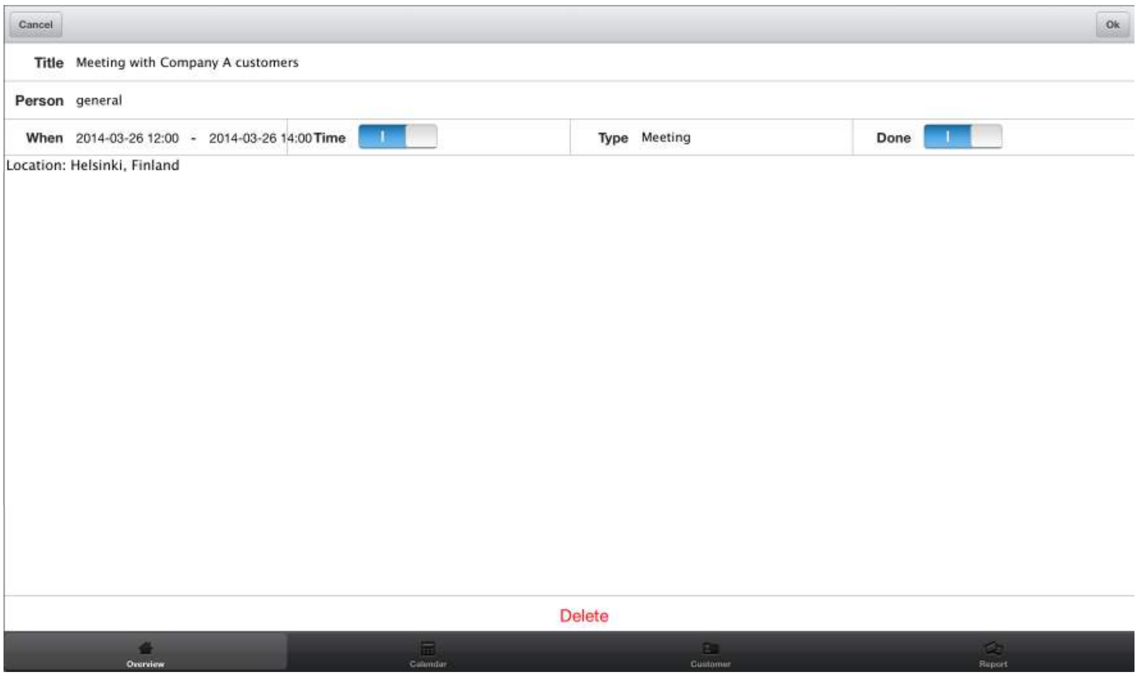


**Figure 19**. Report page.

The report page is the last tab on the bottom; it has a simple flow layout showing all available reports as a list. Clicking on the list item will navigate the current view to the generated graphical report.

## 4.6 Task Editor Page



Title Meeting with Company A customers

Person general

When 2014-03-26 12:00 - 2014-03-26 14:00 Time | Type Meeting | Done

Location: Helsinki, Finland

**Figure 20**. Task editor page.

The task editor page is shown when the user is trying to create / modify a task / activity. It basically shows a form for the attributes of task / activity. Changing the value of the ìDoneî switch will change the task to be done / change the activity to be undone. The save button is at the right top corner and the delete button is at the bottom.

# 5. IMPLEMENTATION AND DEPLOYMENT

## 5.1 Implementation of Different Parts

The implementation on different parts of the software will be discussed in the following sub headings.

### 5.1.1 Task list

The task list component is a user interface component that has been used on the overview page and customer page, showing the list of actives or tasks. Clicking the items allows the user to modify the task / activity or continue the process.

The code below is a snippet from TaskList class; it initializes the layout of the task list component with a text box and a table. The text field named ìfilterFieldî filters items in the table based on the text user input. The table named ìtaskTableî is the main component where the list of tasks and activities resides. The class argument BeanItemContainer<TaskVO> container is used as the tableís data source, each TaskVO object in container will be passed to a ColumnGenerator, where a table cell contains the task name and description, assignee and due date is generated. A click listener has been added to every table cell monitoring the click event,the implementation of clicked event fired is in layoutClick().

```
public class TaskList extends VerticalLayout implements
LayoutEvents.LayoutClickListener {

// Constructuer of TaskList class
Public TaskList(BeanItemContainer<TaskVO> container,
booleansortDateDesc) {
    // A new text fied is initilized
    filterField = new TextField();
    // Givin style name, that is used by css
    filterField.addStyleName(STYLENAME + "-filter");
...
```

```
    // A new table is initilized
    taskTable = new Table();
    taskTable.setSizeFull();
    // Set the table's data
    taskTable.setContainerDataSource(beanItemContainer);
    taskTable.setImmediate(true);
    taskTable.setMultiSelect(false);


…


// define a click listener, listening click events on table items.
@Override
public void layoutClick(LayoutEvents.LayoutClickEventlayoutClickEvent) {
    TaskButtonbtn = (TaskButton) layoutClickEvent.getSource();
  }
// define a text filter, applied on the text filed, used for reacting to the text
user type in to the filter text box.
private class TaskTextFilter implements Container.Filter{
    String filterPattern;
    @Override
    public boolean passesFilter(Object itemId, Item item) throws
UnsupportedOperationException {
      TaskVOTaskVO = (TaskVO)itemId;
  }
```

**Snippet 1. Task list instantiation.**

### 5.1.2 Loading all customers from database

All customer information is needed when the customer search page is visited, to fill the customer list. The code snippet below shows the implementation of loading all customers from the database with address and representative information attached.

```
@Transactional(propagation = Propagation.SUPPORTS, readOnly = true)
  // method is defined as fetchAllContactInfos
  public List<Contact>fetchAllContactInfos() {
    List<Contact>retVal = new ArrayList<Contact>();
    try {
      // Here a DAO method is called to get all contacts object from
database.
      for(Contact contact : contactDAO.findAllContacts()) {
        // Since all contacts object loaded by contactDAO.findAllContacts
are lazy loaded, we initialize address and representatives information
manually.
        contact.setAddresses(addressDAO.findAddressesByContactId(con
tact.getContactId()));
        contact.setRepresentatives(contactDAO.findRepresentativesByCo
ntact(contact.getId()));
        retVal.add(contact);
      }
    } catch (Exception e) {
      // Exception control
      LOG.error("Error in eagerFetchAllContacts", e);
    }

    returnretVal;
  }
```

**Snippet 2**. Service method to request for all customer with address and representative information.

The service method fetchAllContactInfos() in ContactService class in called initially by the request to load all customers, the transactional features of the method is defined in the annotation: the propagation attribute defines the transaction propagation behaviors, Propagation.SUPPORTS means this method call supports a current transaction, if there is no transaction exists, execute non-transactional; setting readOnly=true avoids data been manipulated by the method unintentionally.

The method first calls contactDAO.findAllContacts(), getting the list of all contacts, then iterates through the list looking for the address information and representatives linked to the customer.

```
public List<Contact>findAllContacts() {
    LOG.debug("Find all contacts");
    List<Contact>retVal = new ArrayList<Contact>();
    // create a JPA query asking for rows from contact table
    retVal = entityManager.createQuery("from Contact").getResultList();
    returnretVal;
  }
```

**Snippet 3**. DAO method to load all contacts.

ContactDAO.findAllContacts() method executes a JPA query to select all contact objects from the database.

```
public List<Address>findAddressesByContactId(String contactId) {
    LOG.debug("find address by contact id {}", contactId);
    // create a JPA query asking for rows from address table with given
condition
    return entityManager.createQuery(
        "FROM Address a WHERE a.contact.contactId= :contactId ORDER
BY a.id ASC"
    ).setParameter("contactId", contactId).getResultList();
  }
```

**Snippet 4**. DAO method to load addresses for contact.

AddressDAO.findAddressesByContactId(contactId) method takes contactId as the argument and makes up a JPA query to get a list of addresses by contactId.

```
Public List<Contact>findRepresentativesByContact(Long id) {
    LOG.debug("find representatives by contact " + id);
    List<Contact>retVal = new ArrayList<Contact>();


    retVal = entityManager.createQuery("select c.representatives from
Contact c where c.id = :id").setParameter("id", id).getResultList();
    returnretVal;
  }
```

**Snippet 5**. DAO method to load representatives for contact.

ContactDAO.findRepresentativesByContact(id) method takes id of the contact as the argument and makes up a JPA query to get representatives by id.

### 5.1.3 Start a process

Starting a process will create a new row in the Activiti process instance table, below is the code snippet that starts a process.

```
public ProcessInstance startProcess(Map formValues) {
   // call process service method to set the starter user id
    processEngine.getIdentityService().setAuthenticatedUserId(userId);
    // check start parameters
    if(formValues != null ) {
      if( variables != null ) {
        variables.putAll(formValues);
      } else {
        variables = formValues;
      }
    }
    // start the process by key
    ProcessInstanceprocessInstance =  processEn-
gine.getRuntimeService().startProcessInstanceByKey(processDefinition.getKey
(),variables);
    // reset starter user id
    processEngine.getIdentityService().setAuthenticatedUserId(null);
    this.processInstance = processInstance;
    returnprocessInstance;
  }
```

**Snippet 6**. Start a process.

The argument formValues of type java.util.Map is passed as the starting parameter of the process. Setting the authenticated user id before starting the process is a necessary step, as it set the initiator of the process. Method processEngine.getRuntimeService().startProcessInstanceByKey(processDefinition .getKey(),variables) is called to start the process, using the definition key and staring parameters.

### 5.1.4 Process definitions

The Process definition is defined in .bpmn20.xml files and deployed to Activiti tables in the database.It describes a closed business flow using task as nodes. There are different types of tasks that are supported, the user task is the most common one that usually along with a form defined, waits for user to finish. Other possible tasks include service task, script task, asynchronous task, etc.

```
<definitions id="definitions" targetNamespace="customer">
<!—- start of process definition -->
      <process id="MOBILE_SAMPLE_SALES" name="Create new proposal" ac-
tiviti:candidateStarterGroups="User">
<!—- start event is the first event occur once process is started -->
    <startEvent id="start" activiti:initiator="initiator"/>
```

```
<!—- sequence flow is the element to connect tasks and events -->
      <sequenceFlowsourceRef='start' targetRef='product_select_generate' />
<!—- script task includes a piece of groovy script for different purposes -->
      <scriptTask id='product_select_generate' name='generate product selection
ui' scriptFormat='groovy'>
         <script>
            import groovy.xml.*;
            importfi.fasolutions.mod.portfoliomanagement.domain.*;
            importjava.util.*;


...


            execution.setVariable('product_selection',IOUtils.toByteArray(new
StringReader(xml.toString()),'UTF-8'))
            execution.setVariable('total_product_number',i)
         </script>
      </scriptTask>
      <sequenceFlow id='flow_13' sourceRef='product_select_generate' targe-
tRef='product_select_form' />
<!—- user task usually involves a form shown to user -->
      <userTask id="product_select_form" name="Product selection" activi-
ti:assignee="${initiator}" activiti:formKey="product_selection" />
      <sequenceFlow id='flow_14' sourceRef='product_select_form' targe-
tRef='theEnd' />
      <endEvent id="theEnd" />
    </process>
</definitions>
```

**Snippet 7**. Process definition on simple sales process

Above is an example of a sales process for demonstration purposes. The process starts with a start event, and has two tasks followed. The start event, end event and tasks are connected logically by ìsequenceFlowî elements. The first task with id ìproduct_select_generateî is a script task using Groovy scripting language, the script inside the task body creates a UI component, which is a list showing different financial products and the fee percentage corresponding to the product, the information of the product is fetched from the database. The second task, with the name ìproduct_select_formî, is a user task with a form key. It is interpreted by the software and renders a form that contains the component created before. The form is presented to the user and waiting for command to continue.

### 5.1.5 Sales report

A sales report is a process that produces a graphical component illustrating sales activities with different criteria. Below is the process definition of a simple sales report.

```
<!—- a report is a process generates a visual component (graphic) -->
<definitions id="definitions">
  <process id="Mobile_sales_report" name="Sales report" activi-
ti:candidateStarterGroups="User">
    <startEvent id="start" activiti:initiator="initiator" />
    <sequenceFlow id='flow' sourceRef='start' targetRef='plot' />
<!—- using groovy script to produce a visual component by querying data from
database -->
    <scriptTask id='plot' name='plot a graph' scriptFormat='groovy'>
      <script>
        importgroovy.sql.Sql;
        importcom.vaadin.ui.*;
                        // define visual component
        def root = new CssLayout()
        Chart chart = new Chart(ChartType.BAR)
        Configuration conf = chart.getConfiguration()
        conf.setTitle("Sales acitvity report")
```

```
            XAxis x = new XAxis()
            x.setCategories(users.toArray(new String[users.size()]))
            conf.addxAxis(x)
            YAxis y = new YAxis()
            y.setMin(0)
            y.setTitle("Activities")
            conf.addyAxis(y)
            Legend legend = new Legend()
            legend.setBackgroundColor("#FFFFFF")
            legend.setReversed(true)
            PlotOptionsSeries plot = new PlotOptionsSeries()
            plot.setStacking(Stacking.NORMAL)
            conf.setPlotOptions(plot)
// set up database connection and get the query result
            defsql = Sql.newInstance('jdbc:mysql://37.233.89.79:3306/test-
salk-
ku?rewriteBatchedStatements=true&amp;useUnicode=true&amp;characterEnc
oding=UTF-8&amp;autoReconnect=true&amp;jdbcCompliantTruncation=false',
'test-mysql', 'ddPALQK5NfVsPMb5', 'com.mysql.jdbc.Driver')
            def query = 'select count(*) as count_, ACT_HI_VARINST.TEXT_ as
type_, ACT_HI_TASKINST.ASSIGNEE_ as assignee_ from ACT_HI_TASKINST
left join ACT_HI_VARINST on ACT_HI_VARINST.TASK_ID_ =
ACT_HI_TASKINST.ID_ and ACT_HI_VARINST.NAME_ = "type" left join
ACT_HI_VARINST as ACT_HI_VARINST_2 on ACT_HI_VARINST_2.TASK_ID_ =
ACT_HI_TASKINST.ID_ and ACT_HI_VARINST_2.NAME_ = "contactId" where
PROC_DEF_ID_ is null and END_TIME_ is not null ' + searchCriteria + ' group by
type_, assignee_ order by type_, assignee_'

            ...
            // draw the chart and set as a variable to the process
            chart.drawChart(conf);
            root.addComponent(chart)
            execution.setVariable("vaadin",root)
            }
          </script>
        </scriptTask>
        <sequenceFlow id='flow2' sourceRef='plot' targetRef='theEnd' />
        <endEvent id="theEnd" />
      </process>
    </definitions>
```

**Snippet 8**. Sample sales report process

To distinguish the report process with other processes, the report process has a target name space ìreportî. The sample sales report process only has one script task, where a bar chart is created by collecting information from the database using SQL query directly. Before the process ends, it assigns the created component to process variable ìvaadinî, it triggers the application to open a page containing the component. The start form definition asks the user for the reporting time period before the report is generated.

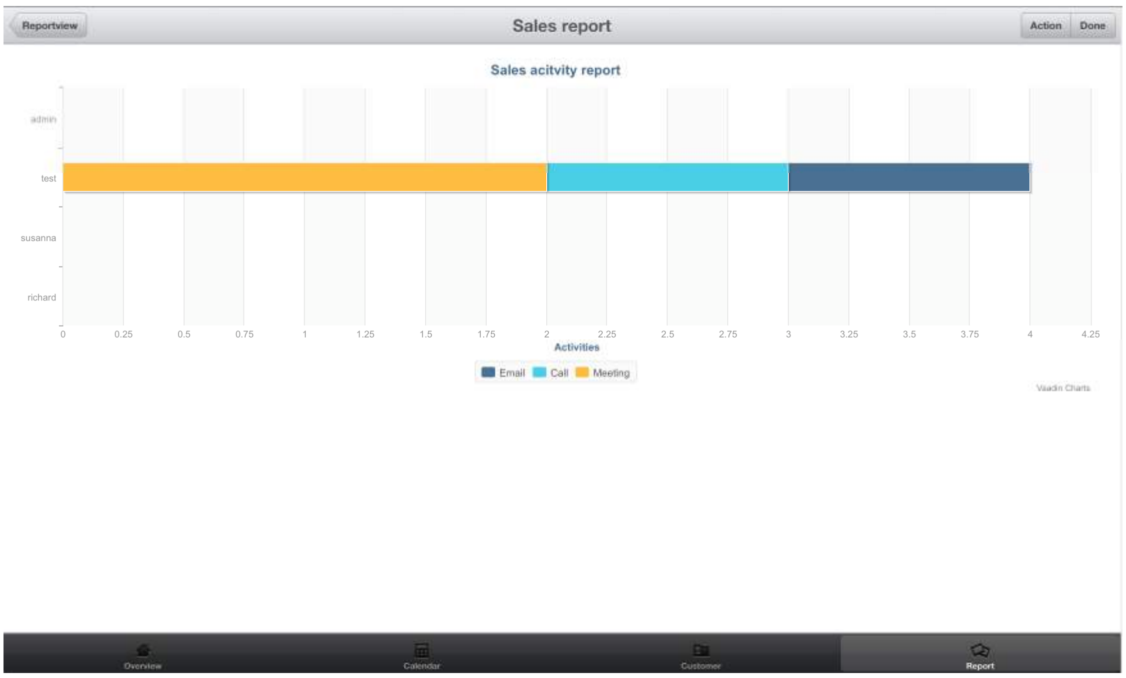


**Figure 21**. Sample sales activity report generated in the report view.

## 5.2 Deployment

The application is deployed as a portlet to the Liferay Portal, in the following section, the deployment configurations will be explained.

### 5.2.1 Deployment descriptor file

The deployment descriptor file describes how the web application is deployed in a container; it directs a deployment tool to deploy a module or an application with specific container options, security settings and describes specific configuration requirements. /10 /

```
<web-app id="WebApp_ID" version="2.4">
      <!—- define the web applications name and description -->
  <display-name>fa_app_gui</display-name>
  <context-param>
    <description>Vaadin production mode</description>
          <!—- start parameters -->
    <param-name>productionMode</param-name>
    <param-value>true</param-value>
  </context-param>
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/applicationContext.xml</param-value>
  </context-param>
```

**Snippet 9**. Context parameters of the deployment descriptor file

The ìcontext-paramî element contains the declaration of the applications initialization parameters. The Vaadin productionMode is enabled to disengage debug tools and improve the performance. The ìcontextConfigLocationî parameter is set as the path of applicationContext.xml, the file defines Spring beans.

```
<!—- filter definitions -->
<filter>
    <filter-name>Invoker Filter - ERROR</filter-name>
            <!—- filter class -->
    <filter-
class>com.liferay.portal.kernel.servlet.filters.invoker.InvokerFilter</filter-
class>
             <!—- start parameters of the filter -->
    <init-param>
      <param-name>dispatcher</param-name>
      <param-value>ERROR</param-value>
    </init-param>
  </filter>
  <filter>
    <filter-name>Invoker Filter - FORWARD</filter-name>
    <filter-
class>com.liferay.portal.kernel.servlet.filters.invoker.InvokerFilter</filter-
class>
```

```
    <init-param>
      <param-name>dispatcher</param-name>
      <param-value>FORWARD</param-value>
    </init-param>
  </filter>
  <filter>
    <filter-name>Invoker Filter - INCLUDE</filter-name>
    <filter-
class>com.liferay.portal.kernel.servlet.filters.invoker.InvokerFilter</filter-
class>
    <init-param>
      <param-name>dispatcher</param-name>
      <param-value>INCLUDE</param-value>
    </init-param>
  </filter>
  <filter>
    <filter-name>Invoker Filter - REQUEST</filter-name>
    <filter-
class>com.liferay.portal.kernel.servlet.filters.invoker.InvokerFilter</filter-
class>
    <init-param>
      <param-name>dispatcher</param-name>
      <param-value>REQUEST</param-value>
    </init-param>
  </filter>
  <filter-mapping>
    <filter-name>Invoker Filter - ERROR</filter-name>
<!—- the url pattern indicates how the filter works on special request-->
    <url-pattern>/*</url-pattern>
    <dispatcher>ERROR</dispatcher>
```

```
   </filter-mapping>
   <filter-mapping>
      <filter-name>Invoker Filter - FORWARD</filter-name>
      <url-pattern>/*</url-pattern>
      <dispatcher>FORWARD</dispatcher>
   </filter-mapping>
   <filter-mapping>
      <filter-name>Invoker Filter - INCLUDE</filter-name>
      <url-pattern>/*</url-pattern>
      <dispatcher>INCLUDE</dispatcher>
   </filter-mapping>
   <filter-mapping>
      <filter-name>Invoker Filter - REQUEST</filter-name>
      <url-pattern>/*</url-pattern>
      <dispatcher>REQUEST</dispatcher>
   </filter-mapping>
```

**Snippet 10**. Filters and filter mappings in the deployment descriptor file

The filter and fitter mapping elements defines the filter classes and its url mapping. The servlet filters are used for preprocessing requests and post processing responses. The four filters used by Liferay portal are defined in this section.

```
<listener>
      <listener-
class>com.liferay.portal.kernel.servlet.PluginContextListener</listener-
class>
   </listener>
   <listener>
```

```
    <listener-
class>com.liferay.portal.kernel.servlet.SerializableSessionAttributeListen
er</listener-class>
  </listener>
  <listener>
    <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-
class>
  </listener>
  <listener>
    <listener-
class>com.liferay.portal.kernel.servlet.PortletContextListener</listener-
class>
  </listener>
```

**Snippet 11**. Listeners in the deployment descriptor file

The listener element defines event listeners for the web application. The listeners are capable of tracking key events in the application. There are four listeners defined in the section: PluginContextListener, SerializableSessionAttributeListener and PortletContextListener are required for Liferay portal to be started, ContextLoaderListener is used to initialize Springapplication context.

```
<!—- servlet deployment descriptor file -->
<servlet>
    <servlet-name>Vaadin2Application</servlet-name>
    <display-name>Vaadin2Application</display-name>
    <servlet-
class>com.liferay.portal.kernel.servlet.PortletServlet</servlet-class>
             <!—- init param element defines different attributes e.g. port-
        let class and vaadin class to be started-->
    <init-param>
```

```
      <param-name>portlet-class</param-name>
      <param-value>com.vaadin.server.VaadinPortlet</param-value>
    </init-param>
    <init-param>
      <description>Vaadin application class to start</description>
      <param-name>application</param-name>
      <param-value>fi.fasolutions.fa.ui.GUIServlet</param-value>
    </init-param>
            <!—- widgetset need to be specified as google web tool kit use
  it to render the web page-->
    <init-param>
      <description>Application widgetset</description>
      <param-name>widgetset</param-name>
      <param-value>fi.fasolutions.fa.gwt.MobileAppWidgetSet</param-
value>
    </init-param>
  </servlet>
<servlet-mapping>
    <servlet-name>Vaadin2Application</servlet-name>
    <url-pattern>/Vaadin2Application/*</url-pattern>
  </servlet-mapping>
```

**Snippet 11**.Servlet and servlet mapping in the deployment descriptor file

The servlet element declares a servlet, including the servlet name, display name, the class to use and initial parameters. As we are running Vaadin as a portlet, the start class of the application is set to be PortletServlet, and we configured the application parameter as an initial parameter to clarify the actual servlet class to be started.

```
<session-config>
    <session-timeout>600</session-timeout>
</session-config>
```

**Snippet 12**. Session configurations in the deployment descriptor file

The ìsession-configî element defines the session attributes for the application. The session time-out is set to be 600 seconds, to avoid the session to be timeouted in case of long waiting calls.

### 5.2.2 Portlet descriptor file

The portlet descriptor file is used to describe the attribute of the application when it is deployed as a portlet.

```
<!—- portlet deployment descriptor file -->
<portlet-name>Vaadin2Application</portlet-name>
      <display-name>Vaadin2Application</display-name>
      <portlet-class>com.vaadin.server.VaadinPortlet</portlet-class>
      <init-param>
         <name>UI</name>
         <value>fi.fasolutions.fa.apps.MobileUserInterface</value>
      </init-param>
      <init-param>
         <name>widgetset</name>
         <value>fi.fasolutions.fa.gwt.MobileAppWidgetSet</value>
      </init-param>
     <init-param>
       <name>style</name>
       <value>height:100%</value>
     </init-param>
      <supports>
         <mime-type>text/html</mime-type>
```

```
    <portlet-mode>view</portlet-mode>
    <portlet-mode>edit</portlet-mode>
    <portlet-mode>help</portlet-mode>
</supports>
<portlet-info>
    <title>FA-app-gui Portfolio</title>
    <short-title>FA-app-gui Portfolio</short-title>
</portlet-info>
```

**Snippet 13**. Portlet deployment descriptor file

Similarly to the deployment descriptor file,the display name, portlet class and initial  parameters are defined in the portlet.xml, other attributes defined are style, supported portlet modes and content type and information of the portlet.

### 5.2.3 Data source definition

The datasource used by the application is defined as the ìdateSourceî bean in the application context.

```
<!—- spring config file defines data source -->
     <!—- c3po connection pool-->
     <bean id="dataSource"
            class="com.mchange.v2.c3p0.ComboPooledDataSource"
            destroy-method="close">
     <!—- driver class -->
          <property
     name="driverClass"><value>${jdbc.driverClassName}</value></property>
     <!—- jdbc url -->
          <property name="jdbcUrl"><value>${jdbc.url}</value></property>
     <!—- login confidents-->
          <property name="user"><value>${jdbc.username}</value></property>
```

```
        <property
    name="password"><value>${jdbc.password}</value></property>
    <!—- connection pool settings -->
        <property name="initialPoolSize"><value>3</value></property>
        <property name="minPoolSize"><value>3</value></property>
        <property name="maxPoolSize"><value>100</value></property>
        <property
    name="autoCommitOnClose"><value>true</value></property>
        <property
    name="unreturnedConnectionTimeout"><value>30000000</value></prop
    erty>
        <property
    name="idleConnectionTestPeriod"><value>120</value></property>
        <property name="acquireIncrement"><value>1</value></property>
        <property name="maxStatements"><value>0</value></property>  <!--
    0 means: statement caching is turned off.  -->
        <property
    name="numHelperThreads"><value>3</value></property>  <!-- 3 is default
    -->
        <property
    name="automaticTestTable"><value>test_salkku</value></property>
      </bean>
```

**Snippet 14**. Data source declarations in applicationContext.xml

The "dataSource" bean is required by Spring to establish the database connection, we used c3p0 library to manage the datasource.

In particular, c3p0 provides several useful services:

1. Classes which adapt traditional DriverManager-based JDBC drivers to the newer javax.sql.DataSource scheme for acquiring database Connections.

2. Transparent pooling of Connection and PreparedStatements behind DataSources which can "wrap" around traditional drivers or arbitrary unpooledDataSources. /11/

The property list of "dataSource" bean describes the parameters used to establish to connection to the data source. We used "initialPoolSize", "minPoolSize" and "maxPoolSize" parameters to define a connection pool no less than 3 connections and no more than 100 connections.The connection pool helps the application with the performance and eliminates situations such as overflow on connections.

### 5.2.4 Spring JPA and Hibernate configuration

Spring JPA offers a comprehensive support for JAVA Persistence API, in FA Sales Mobile we used Spring JPA for abstraction and crud operations while having Hibernate as the object mapper layered on the database.

```
<bean
class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean
" id="entityManagerFactory">
          <!—- reference to datasource bean -->
   <property name="dataSource" ref="dataSource"/>
        <!—- parameter definitions -->
   <property name="persistenceUnitName" value="persistenceUnit" />
   <property name="persistenceUnitManager" ref="pum"></property>
   <property name="jpaVendorAdapter">
     <bean
class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">
     </bean>
   </property>
   <property name="jpaProperties">
     <props>
       <prop key="hibernate.dialect">${jdbc.dialect}</prop>
       <prop key="hibernate.hbm2ddl.auto">update</prop>
```

```
            <prop
key="hibernate.ejb.naming_strategy">org.hibernate.cfg.ImprovedNamingStrat
egy</prop>
            <prop key="hibernate.jdbc.batch_size">0</prop>
            <prop key="hibernate.show_sql">false</prop>
            <prop key="hibernate.jdbc.batch_versioned_data">true</prop>
            <prop key="hibernate.cache.use_second_level_cache">true</prop>
            <prop key="hibernate.cache.use_query_cache">true</prop>
            <prop key="hibernate.order_updates">true</prop>
            <prop key="hibernate.order_inserts">true</prop>
            <prop key="hibernate.connection.autocommit">false</prop>
            <prop key="hibernate.connection.isolation">3</prop>
            <prop
key="hibernate.transaction.factory_class">org.hibernate.transaction.JDBCTran
sactionFactory</prop>
            <prop
key="hibernate.cache.provider_class">org.hibernate.cache.EhCacheProvider</
prop>
            <prop
key="hibernate.cache.region.factory_class" >org.hibernate.cache.ehcache.Singl
etonEhCacheRegionFactory</prop>
            <prop key="hibernate.cache.use_second_level_cache">true</prop>
            <prop key="hibernate.enable_lazy_load_no_trans">true</prop>
         </props>
      </property>
   </bean>
```

**Snippet 15**. JPA EntityManagerFactory declarations in applicationContext.xml

The ìentityManagerFactoryî bean defines an entity manager factory bean of type org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean, The

EntityManagerFactory interface is used by the application to obtain an application-managed entity manager.

The LocalEntityManagerFactory Bean creates an instance of class type EntityManagerFactory suitable for environments which solely use JPA for data access. The factory bean will use the JPA PersistenceProvider autodetection mechanism (according to JPA's Java SE bootstrapping) and, in most cases, requires only the persistence unit name to be specified.The persistence unit name is defined in the ìpersistenceUnitNameî property. To link the entity manager factory bean to a JDBC datasource, the datasource bean is passed in as a property. Setting the property ìjpaVendorAdapterî to HibernateJpaVendorAdapter exposes Hibernateís persistence provider and EntityManager extension interface. The Hibernate parameters and their attributes are set in the ìjpaPropertiesî element.

# 6. TESTING

A comprehensive testing procedure was never stopped during the lifecycle of FA Mobile Sales as bugs were raised up and new improvements were suggested. We used two testing approaches to ensure the quality of the software, as JUnit test case and general test session.

## 6.1 Unit Test Cases

A unit test is a piece of code that executes a specified functionality in the code to be tested; each unit test case usually tests one exact function, leaving little margin for other things interrupting the test unit. By using the unit test, the health of the code is protected and it is especially helpful when coming to some complex logical code pieces.If every bit is correct, the integrity is established.

Unit tests were applied largely to FA Mobile Sales, to check the business logic implementations. Below is a JUnit test case testing the functionality to save a contact.

```
@Test
  @Transactional
  public void testSavingAndLoadingContact() {
    Contact contact = new Contact();
    contact.setName("NIMI");
    contact.setContactId("C1");
    Assert.assertNull(contact.getVersion());
    contactDAO.saveContact(contact);
    Assert.assertNotNull(contact.getVersion());

    Contact contact2 = contactDAO.loadContactByContactId("C1");
    Assert.assertNotNull(contact2);
  }
```

**Snippet 16**. Unit test case to save a contact

## 6.2 General Test Session

A test session was held on the important time point e.g. before a new release.During the test session we checked the new features implemented between versions to make sure they are not conflicting with or breaking existing functionalities and the completion of the software.

On a daily basis, we received bug reports from the customers; these bugs were recorded and waiting to be handled, with a priority definition.

The table below lists all the bugs that have been reported and fixed.

**Table 1**. List of Fixed bugs.

| Fixed date | Abstract of the bug |
|---|---|
| 12/Nov/13 4:02 PM | Fa mobile sales: search a customer and start a process |
| 14/Nov/2014 7:30 PM | Issues of Fa mobile sales working with iPad |
| 5/Nov/2014 8:37 AM | Deploy latest version of Fa mobile salesto server. |
| 06/Feb/14 7:38 PM | Fa mobile sales:startable process according to user group |
| 24/Feb/14 12:57 PM | In calendar, click a task should navigate to the task view.Implement filter the task list. Customer search style fix. Customer view show customer tagging info. Put uploader and uploading time in to document entry. |
| 10/Feb/14 8:46 PM | Fa mobile sales: Refresh document list in customer view after add a document to document library in fasalkku |
| 25/Feb/14 10:35 AM | Fa mobile sales: Historical tasks in calendar |
| 11/Feb/14 9:57 PM | Support for adding attachments manually in process task windowFa mobile sales |
| 11/Feb/14 9:57 PM | Support for adding attachments manually in process task window. |
| 23/Feb/14 2:46 PM | Do not serialize entities into Activit tasks or processes |
| 20/Feb/14 8:16 PM | Fa mobile sales:Customer sales process can not be started after jpa manager is applied to activiti engine |
| 08/Oct/12 10:50 AM | can you check / test the IE, Fa mobile sales |

# 7. CONCLUSION

By the time the report is finished, the application is still involved in an extensive and continuously developing iteration. The initial goal though, which is to help our customers to manage customer resources and do sales processes has been completed. It has been reviewed by multiple customers of FA Solutions Oy for acceptance tests, and has several customer deployments. Positive altitudes are given in general to our work.

Through the development of the software, it opened up a smooth learning curve of the software engineering subject for me, each phase of the has been carefully considered and implemented. Invaluable experiences are gained on customer requirement analyzing, software planning, developing and software testing.

The biggest challenge comes from the planning phase of the development, when designing the architecture of the software according to the requirements from the customer. I worked with my experienced collegues for the goal. Every aspect of the needs and potential possibility extensions that we came up with was reviewed while there are some certain features we decided not to support for the reason that it broke the principle of implementing a "product" overî customized software". With other help from my friends and teachers, I was proud of myself building such an architecture strong and agile enough, and the implementation later had proved that.

Our customers have risen up their wishes on the contemporary implementation during the acceptance test session, credited to our agile implementation.Most of the requirements can be achieved by flexible utilizing Activiti processes, other are in discussion and being analyzed. There have been good suggestion on the application e.g. that it should receive a copy of the email sent to a customer in the backend and automatically records the email activity to the database. It is believed the scenario will become reality in the near future.

To conclude, an opportunity to work with FA Mobile Sales project helped me not only to practise on my skillsets, but also taught me on the way dealing with people and doing things properly. It opened up a new window that I could look further on my career options and choices. The unforgettable experience will last.

# 8. REFERENCES

/1/ What is MySQL. 2011. Accessed 12.3.2014. http://dev.mysql.com/doc/refman/5.1/en/what-is-mysql.html.

/2/ MySQL Innodb ZFS Best Practices. 2009. Accessed 20.3.2014. https://blogs.oracle.com/realneel/entry/mysql_innodb_zfs_best_practices.

/3/ Introduction to Spring Framework. 2014. Accessed 20.3.2014. http://docs.spring.io/spring/docs/3.0.x/reference/overview.html.

/4/ Spring Framework: Chapter 2. Introduction. 2011. Accessed 22.3.2014. http://www.springframework.net/doc-latest/reference/html/introduction.html.

/5/ Book of Vaadin. 2014. [WWW]. Accessed 20.3.2014. https://vaadin.com/book/.

/6/ Activiti 2014. Accessed 12.3.2014. http://www.activiti.org.

/7/ Tutorial: Introduction: Hibernate and Workshop. 2013. Accessed 12.3.2014. http://docs.oracle.com/cd/E15051_01/wlw/docs103/guide/ormworkbench/hibernate-tutorial/tutHibernate1.html.

/8/ HibernateArchitecture. 2012. Accessed 20.3.2014. http://docs.jboss.org/hibernate/core/3.2/reference/en/html/architecture.html.

/9/ Liferay Logical Architecture. 2011. Accessed 22.3.2014. http://www.liferay.com/community/wiki/-/wiki/Main/Logical+Architecture.

/10/ Java EE: XML Schemas for Java EE Deployment Descriptors. 2013. Accessed 13.3.2014. http://java.sun.com/xml/ns/javaee/index.html.

/11/ c3p0 - JDBC3 Connection and Statement Pooling. 2013. Accessed 12.3.2014. http://www.mchange.com/projects/c3p0/>