Master Thesis

Generic Frontend for Exploring Sensor and Information Services

Author:
M.Sc. Uliana Andriieshyna
Matrikel-Nr: 3828303

Supervisors:
Dr.-Ing. Josef Spillner
Prof. Dr. rer. nat. habil.
Dr. h. c. Alexander Schill

April 10, 2014

Declaration of Authorship

I, Uliana Andriiehyna, declare that this thesis, titled "Generic Frontend for Exploring Sensor and Information Services", and the work presented in it have been done on my own without assistance. All information directly or indirectly taken from external sources is acknowledged and referenced in the bibliography.

Dresden, XX.XX.2013 _____

Contents

Chapter 1

Introduction

The increasing numbers of sensor devices has increased the number of sensor-specific protocols, platforms and software. As a result various approaches have been proposed to interconnect, maintain and monitor various type of sensors[1, 2, 3]. That specificly focused on a platform development, protocol definition and software architecture for the concreate user-oriented requirements and for a narrowly focused areas of usage instead of defining a common system approach. And mainly in proposed approaches was discovered such type of questions as security and privacy, easy of development and monitoring, social dimensions [4], that completely don't consider requirements and criteria of an end-user. Therefore the area of research of this master thesis is dedicated to define generic frontend for exploring sensor and information services. The folowing sections ground the motivation for the chosen research field, define the central research questions and goals of this master's thesis, and describe the overall structure of the work.

1.1 Motivation

In the recent years with the technological progress in the computer science, information systems, and in particular sensor data systems, have become an essential part in daily life of the modern society. People have started to use them more often not only for manufactory, business, education but also for private reasons. Currently, most of the research is concerned with the protocol and middleware levels, whereas the potential of a generic interactive access to sensor and information services needs to be explored. This involves their selection, mash-ups, and usage within a client-controlled interface. In this master thesis, a first web-based prototype (portal) for such services is to be created. Users should be able to explore not just services, but also the information provided by them, and eventually be led to advanced usage patterns such as the development of third-party applications to access the information data and real-time streams.

1.2 Research Questions and Goals

Creating composite third-party services and applications from reusable components is an important technique in software engineering and data management. Although a large body of research and development covers integration at the data and application levels, weak work has been done to facilitate it at generic level. This master thesis discusses the existing user interface frameworks and component technologies used in presentation integration, illustrates their strengths and weaknesses, and presents some opportunities for future work.

As mentioned in the previous section, there are already exist many solutions for creating sensor-aware applications. But these platforms focuses on a single area of usage and they are not commonly suitable to support the dynamic and adaptable composition and usage of different type of sensors in one portal.

- Concept for a generic information and sensor service portal
- Development of the portal and associated dependency tools
- Demonstration using a convincing scenario

Therefore, this thesis is aimed at the development of a concept that provides users a possibility to personalize their current environment indepently from any type and kind of devices.

1.3 Structure

The thesis is structured in the following way:

Chapter 2 defines the background of the master's thesis describing the basic used terminology and the foundation platforms. A reference scenario and the requirements to a concept that has to be developed are also introduced in this chapter.

Chapter 3 is devoted to the state of the art analysis. The related research works in the areas of the sensor-driven platforms, the component based groupware systems, the browser based and non-browser based systems are investigated and evaluated against the defined requirements.

Chapter 4 focuses on the concept of the generic Frontend for exploring sensor and Information sevices, considering possible approaches, strategies, frameworks and necessary criteries, defined in *Chapter 3*.

Chapter 5 provides the implemented functionalities of the concept and describes evaluaion of results.

Chapter 6 concludes the master's thesis underlining the achieved goals and providing prospects for the future work.

Chapter 2

Foundations and Requirements Analysis

The fundamental terms used in this thesis are described below for better understanding of the presented research work.

2.1 Frontends

In computer science, the front end is responsible for collecting input in various forms from the user and processing it to conform to a specification the backend can use. The frontend is an interface between the user and the backend[5] and the separation of software systems into front and back ends simplifies development and separates maintenance. Therefore need to be distinguished what are the main requirements to a generic frontend for exploring sensed data.

- Loosely-coupling
- Fine-grained structure
- Multy-user capability
- Cross-platforming
- Adaptivity

Web Portal

Such a system, that generates requested information dynamically, displays that information in a useful manner in Web, maintains control over the created/saved files, automatically updates the information, and supports a distributed environment[6]. A web portal is most often specially-designed Web page at a website which brings information together from diverse sources in a uniform way. Usually, each information source gets its dedicated area on the page for displaying information (a portlet); often, the user can configure which ones to display.Portals provide a way for enterprises and organizations to provide a consistent

look and feel with access control and procedures for multiple applications and databases, which otherwise would have been different web entities at various URLs. The features available may be restricted by whether access is by an authorized and authenticated user (employee,member) or an anonymous site visitor[7].

2.2 Data Sources

-types of coommon data sources in Web
-interface for interconnection

2.2.1 Sensors

2.2.2 Web

2.2.3 Generic Considerations

2.3 Concept Requirements

Like most modern applications, each of these is structured into three layers: presentation, application (also called the business-logic layer), and data.

1. the granularity of the functions that the component applications provide is generally well suited for high-level integration (for example, we can tell an application to begin monitoring machine xyz without considering how this activity will affect data in the integrated application's database)

2. it's more stable because the component application is aware of the integration (it exposes the API) and will attempt to stabilize the interface across versions

2.4 Summary

Chapter 3

State of the Art

The following chapter covers an overview and analysis of the existent solutions in the related research areas of web-based third-party applications, which were designed specially for retrieving differet type of sensed data to the user. At the beginning the prominent examples of dashboards platforms are studied and evaluated against the requirements described in the previous chapter with a purpose to clarify their individual capabilities.

3.1 ?Web of Sensors?

Since the thesis is targeted at the creation of a generic user-friendly data stream interaction system and currently every project focused on a specific area of realization and concrete sensor types, such as urban environment[3]. The real-time environmental monitoring portal or geospatial infrastructure for effectively or efficiently collecting and serving vast field data over the web. Internet based urban environment observation system that can real-time monitor environmental changes of temperature, humidity, illumination or air components in urban area. It provides web-based platform, where end-user can simply monitor urban area in his/her city. In environmental monitoring, the field server for constructing outdoor sensor network is a Web-based field observation device, which detects field environmental parameters and publishes them on Internet in real-time.
Smart city[1]tool using information technology and communication (ICT) to help local government to monitoring what currently happened in the city. pplication for monitoring city in single dashboard to help summarize the current condition of city. The architecture system use network sensor consisting of sensor nodes that has function to capture city condition like temperature, air pollution, water pollution, traffic situation. Also we can add another information socio-economic situation like public health service, economic indicator, energy supplies, etc. We have successfully developed the prototype of the smart city dashboard the give more accurate information of Bandung City, one of big cities in Indonesia. This study has implemented prototype system of smart city dashboard at Bandung. It consists of network sensor, server, application and also communication protocol which is used for city monitoring. The summary information of city can be displayed in single view to help people watching, analyzing and action to what being happen at the real-time in the city.

*Microsoft SensorMap[8]*has proposed a system to monitor and present physical sensors in the real world. SensorMap allows owners of sensor networks to register their physical sensors and publish their data on SensorMap. They use GeoDB to store the sensor network information, DataHub to retrieve the new sensor data to enable real time services, and the aggregator to summarize sensor data in a specific area to clients.
LiveWeb Portal[9] presents the architecture, design, and application of a sensorweb service portal, where sensorweb is a global observation system for varied sensory phenomena from the physical world and the cyber world. This system has been used to represent and monitor real-time physical sensor data and cyber activities from ubiquitous sources. LiveWeb meets its goal of providing an efficient and robust sensor information oriented web service, enabled with real-time data representation, monitoringand notification.LiveWeb has the following properties: the system enables sensorweb service accessible from anywhere, makes sensor network data readable by anyone, sensor network sharing, the system make sensor data format transparent to data users, real-time data display suits sensor data properties, an offline alert system strengthens real-time features.
Internet of Things[2] where presented a service platform based on the Extensible Messaging and Presence Protocol (XMPP) for the development and provision of services for Internet of Things(IoT) mainly focusing on the integration of things based on service technologies, scenarios in domains like smart cities, automotive or crisis management require service platforms involving real world objects, backend-systems and mobile devices. And argued necessary usage of XMPP client as protocol for unified, real-time communication and introduce the major concepts of our platform. Based on two case studies we demonstrate real-time capabilities of XMPP for remote robot control and service development in the e-mobility domain.
Dynvoker Portal[10] a generic human-driven ad-hoc usage approach, by including rapid service testing and dynamic inclusion of services as plugins into applications. Dynvoker consists of a relatively small application core which can be run as a servlet, a web service or a command-line application. explore method-centric and resource-centric services alike, output forms in various formats or integrate GUI services to provide a richer user experience. The generic design of many parts of Dynvoker has yielded a lightweight architecture which is freely available to any interested person as an open source project

3.2 Frontend Development Approaches

In computer science, the frontend is responsible for collecting input from user and processing it to a backend system and another direction - collecting data from backend, namely sensor data steam, and processing it to the user-friendly interface. Therefore, on the one side, generic frontend has to satisfy architecture requirements from backend, such as: fine-grained distributred structure, cross-platforming, multy-user capabilities; and on the other side, define a dynamic user-friendly interface to a end-user. And to satisfy aforementioned requirements from backend server it is necessary to compare all available web-based applications.
To retrieve sensor data from different resources in one web-based interface existent next approaches :

- portal with portlets,
- mashup[1],
- HTML5 technology

Portal technology brings information together from diverse sources in a uniform way. Usually, each information source gets its dedicated area on the page for displaying information (a portlet); often, the user can configure which ones to display. The extent to which content is displayed in a 'uniform way' may depend on the intended user and the intended purpose, as well as the diversity of the content. Very often design emphasis is on a certain 'metaphor' for configuring and customizing the presentation of the content and the chosen implementation framework and/or code libraries[11, 12]. In portal technologies end-user can customize number of retrieved data sources, but for that he has to be aware what is it and how to integrate it in portal. User interface in portals have fixed layout, style and location on the web page. To make changes in it, end-user needs to have a deep knowlendge of the system architeture and of whole portal entirely.
Mashup is a web page, or web application, that uses content from more than one source to create a single new service displayed in a single graphical interface. The term implies easy, fast integration, frequently using open application programming interfaces (API) and data sources to produce enriched results that were not necessarily the original reason for producing the raw source data. The term mashup originally comes from pop music, where people seamlessly combine music from one song with the vocal track from another—thereby mashing them together to create something new. The main characteristics of a mashup are combination, visualization, and aggregation. It is important to make existing data more useful, for personal and professional use. To be able to permanently access the data of other services, mashups are generally client applications or hosted online[**?**]. Both commercial products and research prototypes have a broad range of features that simplify a mashups design process, and provide mashups storage and publication. But to customize retrived resources end-user have no option, as use only predefined type and numbers of applications, that was created by application or platform developer. Also Mashup approach is strictly platform- and customer-oriented. It is simply provides stack of tools, by using which user through an user-friendly interface The architecture of a mashup is divided into three layers:

- *Presentation / user interaction:* this is the user interface of mashups. The technologies used are HTML/XHTML, CSS, Javascript, Asynchronous Javascript and Xml (Ajax).
- *Web Services:* the product's functionality can be accessed using API services. The technologies used are XMLHTTPRequest, XML-RPC, JSON-RPC, SOAP, REST.

[1] `http://www.programmableweb.com/applications`

- *Data:* handling the data like sending, storing and receiving. The technologies used are XML, JSON, KML.

Architecturally, there are two styles of mashups: Web-based and server-based. Whereas Web-based mashups typically use the user's Web browser to combine and reformat the data, server-based mashups analyze and reformat the data on a remote server and transmit the data to the user's browser in its final form[13].
Mashups and portals are both content aggregation technologies. Portals are an older technology designed as an extension to traditional dynamic Web applications, in which the process of converting data content into marked-up Web pages is split into two phases: generation of markup "fragments" and aggregation of the fragments into pages. Each markup fragment is generated by a "portlet", and the portal combines them into a single Web page. Portlets may be hosted locally on the portal server or remotely on a separate server.

3.3 HTML5 Technology

To satisfy one of the main requirement about dynamic user-friendly interface, adaptable to any kinf of device, mashup rchitecture should be enhanced with a HTML5 Technology. It is based on various design principles, that truly embody a new vision of possibility and practicality[14].

- Compatibility(inharit all previous techniques and standards)
- Utility
- Secure by Design(origin-based security model that is not only easy to use but is also used consistently by different APIs.)
- Separation of Presentation and Content(CSS3)
- Interoperability(Native browser ability instead of complex JavaScript code; a new, simplified DOCTYPE;simplified character set declaration; powerful yet simple HTML5 APIs)
- Universal Access(suport users with disabilities by using screen readers; media independence-HTML5 functionallity should work across all different devices and platforms; support for all world languages)

3.4 Sensor Data

Since the thesis is targeted at the creation of a generic user-friendly data stream interaction system and currently every project focused on a specific area of realization and concrete

sensor types, such as urban environment[3]. The real-time environmental monitoring portal or geospatial infrastructure for effectively or efficiently collecting and serving vast field data over the web. Internet based urban environment observation system that can real-time monitor environmental changes of temperature, humidity, illumination or air components in urban area. It provides web-based platform, where end-user can simply monitor urban area in his/her city. In environmental monitoring, the field server for constructing outdoor sensor network is a Web-based field observation device, which detects field environmental parameters and publishes them on Internet in real-time.

smart city[1]tool using information technology and communication (ICT) to help local government to monitoring what currently happened in the city. pplication for monitoring city in single dashboard to help summarize the current condition of city. The architecture system use network sensor consisting of sensor nodes that has function to capture city condition like temperature, air pollution, water pollution, traffic situation. Also we can add another information socio-economic situation like public health service, economic indicator, energy supplies, etc. We have successfully developed the prototype of the smart city dashboard the give more accurate information of Bandung City, one of big cities in Indonesia. This study has implemented prototype system of smart city dashboard at Bandung. It consists of network sensor, server, application and also communication protocol which is used for city monitoring. The summary information of city can be displayed in single view to help people watching, analyzing and action to what being happen to city.

LiveWeb[9]

robots[2].

We need to determine types of sensed data:
- data-stream
- data on demand

Many of the Web applications seen today incorporate different patterns to merge functionality directed toward the end user. This section focuses on the design approaches for portal solutions. It addresses the design stage of the solution development process, highlighting the key issues that are specific to portal and application designs.

3.5 Summary

This chapter briefly introduced main approaches for building web-based dashboards by retriving sensed data. Main focus was given to its multy-user usability, adaptive UI design, dynamic content composition. Where portal and mashup technology come into a picture.

	Portal	Mashup
Classification	Older technology, extension of traditional Web server model using well-defined approach	Uses newer, loosely defined "Web 2.0" techniques
Philosophy/approach	Approaches aggregation by splitting role of Web server into two phases: markup generation and aggregation of markup fragments	Uses APIs provided by different content sites to aggregate and reuse the content in another way
Content dependencies	Aggregates presentation-oriented markup fragments (HTML, WML, VoiceXML, etc.)	Can operate on pure XML content and also on presentation-oriented content (e.g., HTML)
Location dependencies	Traditionally, content aggregation takes place on the server	Content aggregation can take place either on the server or on the client
Aggregation style	"Salad bar" style: Aggregated content is presented 'side-by-side' without overlaps	"Melting Pot" style - Individual content may be combined in any manner, resulting in arbitrarily structured hybrid content
Event model	Read and update event models are defined through a specific portlet API	CRUD operations are based on REST architectural principles, but no formal API exists
Relevant standards	Portlet behavior is governed by standards JSR 168, JSR 286 and WSRP, although portal page layout and portal functionality are undefined and vendor-specific	Base standards are XML interchanged as REST or Web Services. RSS and Atom are commonly used. More specific mashup standards such as EMML are emerging.

Figure 3.1: Comparative Characteristic

Chapter 4

Concept

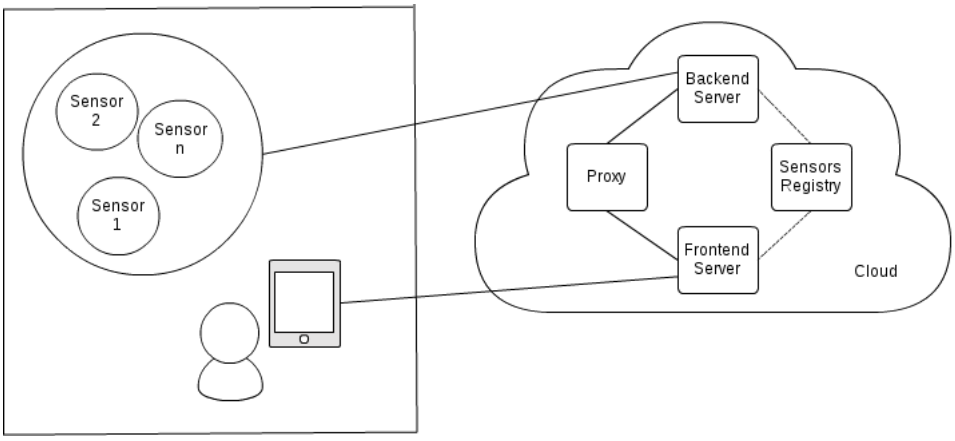
The chapter describes a concept of a generic frontend for exploring sensor data, that in the same time controlled and provisioned by end users request. The concept is developed based on the analysis of the current state of the platform, the defined requirements to the third-party services and applications (section X.X) and knowledges gained from the studied related works (chapter 3).

- Concept for a generic information and sensor service portal
- Development of the portal and associated dependency tools

4.1 Requirements

4.1.1 Non-functional Requirements

We need to determine types of sensed data:
- data-stream
- data on demand



Case.png

Figure 4.1: Use Case

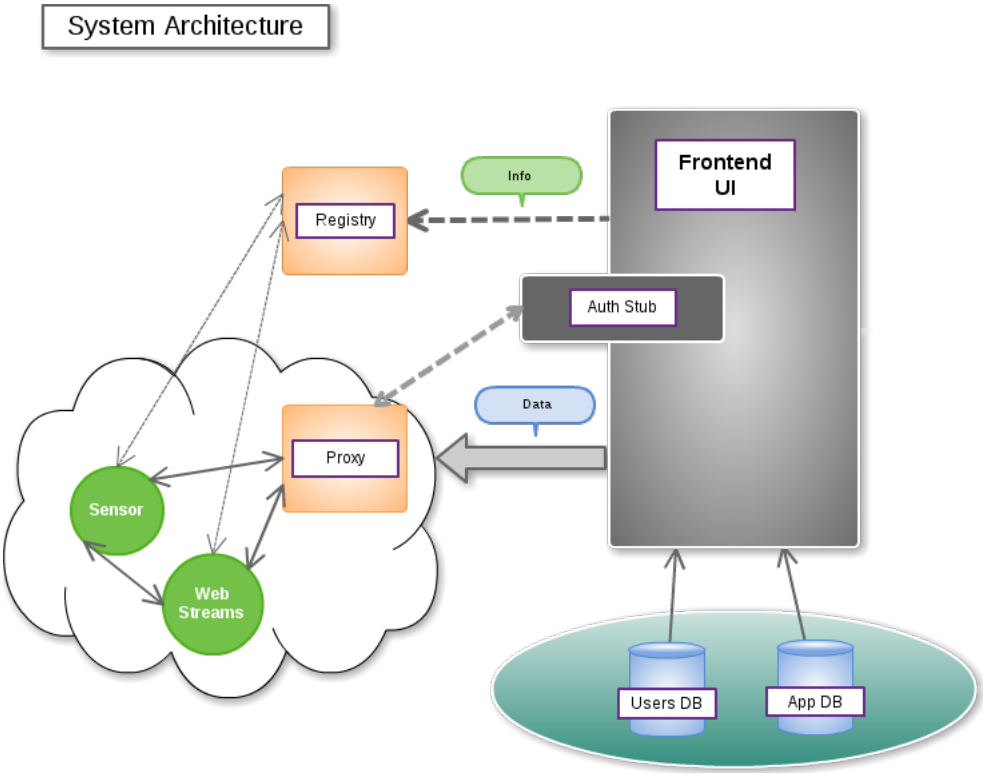


Figure 4.2: System Architecture

4.1.2 Functional Requirements

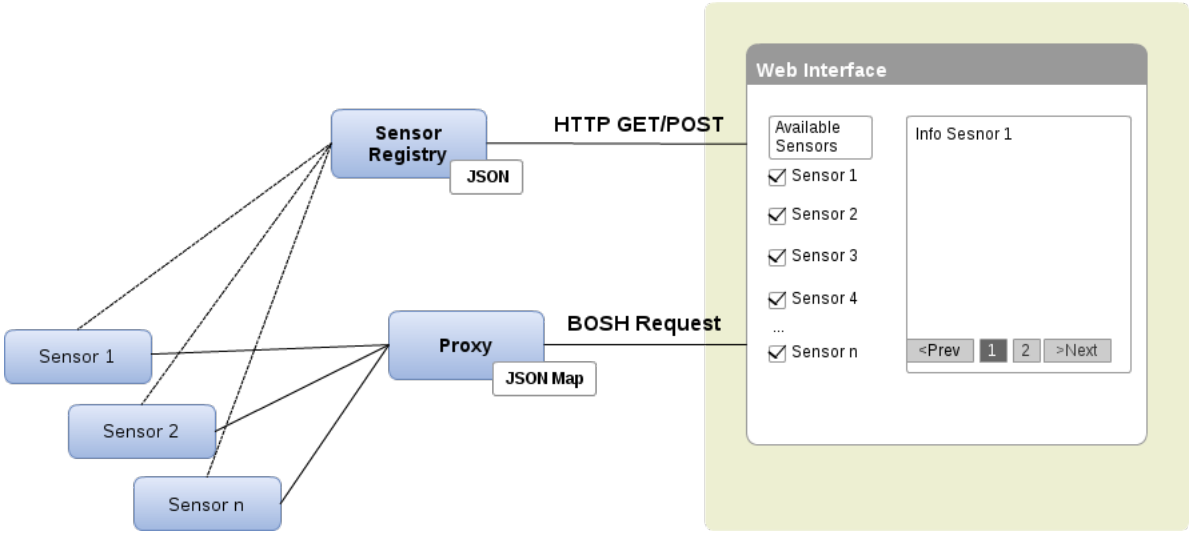


Figure 4.3: Interface

4.1.3 Sensor Registry

4.1.4 Proxy

4.1.5 Authentification Stub

4.1.6 Databases

4.1.7 XMPP Client

4.2 Summary

In this master thesis, a first web-based prototype (portal) for such services is to be created. Along with it, a light-weight scenario service registry will be needed. Users should be able to explore not just services, but also the information provided by them, and eventually be led to advanced usage patterns such as the development of third-party applications to access the information data and real-time streams.

Chapter 5

Implementation and Evaluation

The chapter presents a prototype of the reference scenario considered in the section 2.3. The prototype implements the major aspects proposed in the concept (chapter 4).

5.1 Overview of Framework

5.2 Web-based Framework Analysis

- **Bootstrap**
  Bootstrap is the most popular and widely used framework, nowadays. It's a beautiful, intuitive and powerful web design kit for creating cross browser, consistent and good looking interfaces. It offers many of the popular UI components with a plain-yet-elegant style, a grid system and JavaScript plugins for common scenarios.

  It is built with LESS and consists of four main parts: Scaffolding – global styles, responsive 12-column grids and layouts. Bear in mind that Bootstrap doesn't include responsive features by default. If design needs to be responsive this functionality have to be done manually. Base CSS – this includes fundamental HTML elements like tables, forms, buttons, and images, styled and enhanced with extensible classes. Components – collection of reusable components like dropdowns, button groups, navigation controls (tabs, pills, lists, breadcrumbs, pagination), thumbnails, progress bars, media objects, and more. JavaScript – jQuery plugins which bring the above components to life, plus transitions, modals, tool tips, popovers, scrollspy (for automatically updating nav targets based on scroll position), carousel, typeahead (a fast and fully-featured autocomplete library), affix navigation, and more.

- **Foundation**
  Foundation is a powerful, feature-rich, responsive front-end framework. With Foundation user can quickly prototype and build websites or apps that work on any kind of device, with tons of included layout constructs, elements and best practices. It's built with mobile first in mind, utilitizes semantic features, and uses Zepto instead of jQuery in order to brings better user experience and faster performance.

Foundation has a 12-column flexible, nestable grid powerful enough to create rapidly multi-device layouts. In terms of features it provides many. There are styles for typography, buttons, forms, and various navigation controls. Many useful CSS components are provided like panels, pricing tables, progress bars, tables, thumbnails, and flex video that can scale properly your video on any device. And, of course, JavaScript plugins including dropdowns, joyride (a simple and easy website tour), magellan (a sticky navigation that indicates where is the user on the page), orbit (a responsive image slider with touch support), reveal (for creating modal dialogs or pop-up windows), sections (a powerful replacement for traditional accordions and tabs), and tooltips.

- **GroundworkCSS**
  GroundworkCSS is a new, fresh addition to the front-end frameworks family. It's a fully responsive HTML5, CSS and JavaScript toolkit built with the power of Sass and Compass which gives the ability to rapidly prototype and build websites and apps that work on virtually any device.

  It offers an extremely flexible, nestable, fraction-based, fluid grid system that makes creating any layout possible. GroundworkCSS has some really expressive features like tablets and mobile grids which maintain the grid column structure instead of collapsing the grid columns into individual rows when the viewport is below 768 or 480 pixels wide. Another cool feature is a jQuery ResponsiveText plugin which allows to have dynamically sized text that adapts to the width of the viewport: extremely useful for scalable headlines and building responsive tables. The framework includes a rich set of UI components like tabs, responsive data tables, buttons, forms, responsive navigation controls, tiles (a beautiful alternative to radio buttons and other boring standard form elements), tooltips, modals, Cycle2(a powerful, responsive content slider), and many more useful elements and helpers. It also offers a nice set of vector social icons and a full suite of pictographic icons included in FontAwesome. To see the framework in action user can use the resizer at the top center of the browser window. This way user can test the responsiveness of the components against different sizes and viewports while exploring the framework's features. GroundworkCSS is very well documented with many examples, and to get user started quickly the framework also provides several responsive templates. The only thing as a weakness is the missing of a way to customize download.

- **Gumby**
  Gumby is simple, flexible, and robust front-end framework built with Sass and Compass.

  Its fluid-fixed layout self-optimizes the content for desktop and mobile resolutions. It support multiple types of grids, including nested ones, with different column variations. Gumby has two PSD templates that get user started designing on 12 and 16 column grid systems. The framework offers feature-rich UI Kit which includes buttons, forms, mobile navigation, tabs, skip links, toggles and switches, drawers, responsive images, retina images, and more. Following the latest design trends the UI elements have Metro style flat design but can use Pretty style with gradient design too, or to mix up both styles. An awesome set of responsive, resolution independent Entypo icons,

is completely integrated into the Gumby Framework. Gumby has also a very good customizer with color pickers which helps to build your custom download with ease.

- **Kube**
  Lastly, if user need a solid, yet simple base without needless complexity and extras, for your new project, Kube can be the right choice. Kube is a minimal, responsive and adaptive framework with no imposed styling which gives to user the freedom to create. It offers basic styles for grids, forms, typography, tables, buttons, navigation, and other stuff like links or images.

  The framework contains one compact CSS file for building responsive layouts with ease and two JS files for implementing tabs and buttons in your designs. If user is looking for maximum flexibility and customization, user can download developer version which includes LESS files, with variables, mixins and modules.

Figure 5.1: Framework Comparison[1]

5.3 Data Flow Model

5.4 Model-View-Control Pattern

In the design shown in Figure 1 on page X Model represents the application object that implements the application data and business logic. The View is responsible for formatting the application results and dynamic page construction. The Controller is responsible for receiving the client request, invoking the appropriate business logic, and based on the results, selecting the appropriate view to be presented to the user. The Model represents enterprise data and the business rules that govern access to and updates to this data. Often the Model serves as a software approximation to a real-world process, so simple real-world modeling techniques apply when defining the Model. A View renders the contents of a Model. It accesses enterprise data through the Model and specifies how that data should be presented. It is the View's responsibility to maintain consistency in its presentation when the Model changes. This can be achieved by using a push Model, where the View registers itself with the Model for change notifications, or a pull Model, where the View is responsible for calling the Model when it needs to retrieve the most current data. A Controller translates interactions

with the View into actions to be performed by the Model. In a stand-alone GUI client, user interactions could be button clicks or menu selections, whereas in a Web application, they appear as GET and POST HTTP requests. The actions performed by the Model include activating business processes or changing the state of the Model. Based on the user interactions and the outcome of the Model actions, the Controller responds by selecting an appropriate View.

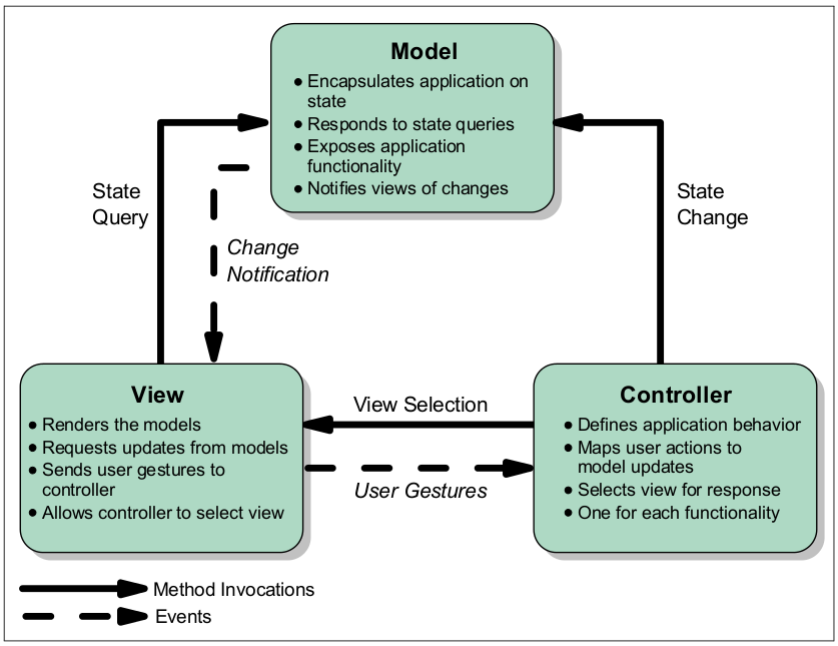


Figure 1 The Mode-View-Controller design pattern

Figure 5.2: MVC Pattern

5.5 Database Model

5.6 Use Cases

5.6.1 Frontend

5.6.2 Choosen Environment

5.6.3 Choosen Tools

5.6.4 Use Cases Realization

5.6.5 Summary

Chapter 6

Conclusion and Outlook

6.1 Conclusion

The chapter summarizes the presented thesis providing an overview of each chapter and describing the achievement of the thesis's goals defined in the section 1.2. At the end of the chapter suggestions for the future work are made.

6.1.1 Addressed research questions

Achieved Goals

- Some;
- Statements;
- Here;

6.1.2 Practical results

6.2 Future work

To conclude the thesis, suggestions for the future work are made. These suggestions are devoted to improve either the developed concept or the current implementation.
Visualization and interaction metaphor for the introduced access control
Enhanced user interface for application part
User interface to support the introduced dynamic composition

List of Figures

List of Tables

Bibliography

[1] S. Suakanto, S.H. Supangkat, Suhardi, and R. Saragih. Smart city dashboard for integrating various data of sensor networks. In *ICT for Smart Society (ICISS), 2013 International Conference on*, pages 1–5, 2013.

[2] Schuster Schill Ackermann Ameling Bendel, Springer. A service infrastructure for the internet of things based on xmpp. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on*, pages 385–388. IEEE, 2013.

[3] Xianfeng Song, Chaoliang Wang, Masakazu Kagawa, and Venkatesh Raghavan. Real-time monitoring portal for urban environment using sensor web technology. In *Geoinformatics, 2010 18th International Conference on*, pages 1–5. IEEE, 2010.

[4] Michael Eggert, Roger Häußling, Martin Henze, Lars Hermerschmidt, René Hummen, Daniel Kerpen, Antonio Navarro Pérez, Bernhard Rumpe, Dirk Thißen, and Klaus Wehrle. Sensorcloud: Towards the interdisciplinary development of a trustworthy platform for globally interconnected sensors and actuators. *arXiv preprint arXiv:1310.6542*, 2013.

[5] Wikipedia. Front and back ends — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Front_and_back_ends&oldid=572495173`, 2013. [Online; accessed 15-November-2013].

[6] Mohsen Rezayat. The enterprise-web portal for life-cycle support. *Computer-Aided Design*, 32(2):85–96, 2000.

[7] Wikipedia. Web portal — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Web_portal&oldid=578361576`, 2013. [Online; accessed 17-November-2013].

[8] Suman Nath, Jie Liu, and Feng Zhao. Sensormap for wide-area sensor webs. *Computer*, 40(7):90–93, 2007.

[9] Xiaogang Yang, Wenzhan Song, and Debraj De. Liveweb: A sensorweb portal for sensing the world in real-time. *Tsinghua Science & Technology*, 16(5):491–504, 2011.

[10] Josef Spillner, Marius Feldmann, Iris Braun, Thomas Springer, and Alexander Schill. Ad-hoc usage of web services with dynvoker. In *Towards a Service-Based Internet*, pages 208–219. Springer, 2008.

[11] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. Restful web services vs. big'web services: making the right architectural decision. In *Proceedings of the 17th international conference on World Wide Web*, pages 805–814. ACM, 2008.

[12] Daniel Su Kuen Seong. Usability guidelines for designing mobile learning portals. In *Proceedings of the 3rd international conference on Mobile technology, applications & systems*, page 25. ACM, 2006.

[13] Michael Michael Thomas Bolin. *End-user programming for the web.* PhD thesis, Massachusetts Institute of Technology, 2005.

[14] Ian Hickson and David Hyatt. Html5: A vocabulary and associated apis for html and xhtml. *W3C Working Draft edition*, 2011.