universität
wien

MASTERARBEIT

Titel der Masterarbeit

"HTML5 audio and video annotations with resource detection under consideration of usability and user experience"

Verfasser

Paul Weichhart, Bakk.

angestrebter akademischer Grad

Diplom - Ingenieur (Dipl.-Ing.)

Wien, April 2012

Studienzahl lt. Studienblatt:	A 066 935
Studienrichtung lt. Studienblatt:	Medieninformatik
Betreuer:	Univ.-Prof. Dipl.-Ing. Dr. Wolfgang Klas

Acknowledgments

I would like to thank Univ.-Prof. Dipl.-Ing. Dr. Klas, Dr. Simon and Dr. Haslhofer for their guidance and encouragement. I also would like to use this opportunity to thank my parents for their support and patience, Cornelia for all the inspiration and reassurance, Georg for his guidance, Oliver for his assistance and to extend my thanks to my whole family and all of my friends.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Currently we can observe an ongoing trend of more and more multimedia content being published on the Web. A range of Web applications like Flickr, Youtube, Soundcloud, Vimeo, etc. exist, that allow end-users to organize content. Content of any type may be annotated through tagging and linking mechanisms. Until now, however, there exist no open standard-based solutions which allow users to link audio and video to contextually relevant information provided by other sources on the Web.

The rising importance of the semantic Web will soon necessitate an open standard-based approach which supports users in annotating and linking existing audio and video resources with relevant content.

Europeana[1], a digital library, provides access to a huge amount of paintings, music, films and books. Currently those resources are published on the Web. However, making use of individual expertise and experiences of different users, the documentation can be enhanced through annotating the individual media. This includes the process of adding information to specific content as well as linking those annotations to resources on the Web [1].

Europeana contains a virtual exhibition section where visitors can gather information on various topics. Each topic, like the current "Art Nouveau" exhibition, is organized in several themes, where users can access digital media like paintings, audio guides and videos. However, those are only sparsely documented and lack any detailed information. With the effort of annotations and the knowledge of individual experts, the documentation can be further improved for the benefit of later visitors.

A framework is therefore needed, which allows end-users to annotate video and audio files with semantic tags and furthermore links those sources contextually to relevant information on the Web. The framework should also build on open standards and should have a clean and structured user interface.

Another focus of this endeavor is to provide a usable interface. Therefore the sequence of annotating and linking those annotations needs to be efficient and simple.

[1] Europeana [Online]. Available at http://www.europeana.eu/portal

1.2 Research questions and goals

The overall research goal of this project is to conduct research in the field and design a usable, standard-based approach to support users in annotating and linking audio and video data on the Web.

This overall goal will be achieved by separating our research approach into multiple objectives:

- Annotating and Linking content on the Web: To enable access to a broad range of data sources, the approach will be based on state of the art research. The technical design will be based on Linked Data research, which provides technical access to many different data sets.
- Standard-based approach: Identify suitable standards that allow end-users to link and tag multimedia files. A standard is considered relevant if it will be supported by most browsers in the foreseeable future.
- Usability: The user interface has to provide sufficient usability for a wide range of proficiency levels while maintaining a streamlined user experience. Usability testing will be conducted according to established methods.

Considering the above mentioned parameters, this master thesis will focus on developing an open standard-based solution for annotating audio and video sources on the Web. Additionally, users have to be able to link annotations with contextually relevant sources on the Web. The interface of the application should be highly usable and exemplify a state of the art solution. The final application should also support the latest open standard Web technologies.

1.3 Overview

This master thesis is organized as follows:

Chapter 2 discusses the fundamentals of this work along with related Web technologies. At first, we shall discuss annotations and established frameworks, the concept of Linked Data, Web technologies like HTML5, CSS and Google's Web Tool Kit and take a look at audio and video fragment identification. Those concepts will be considered fundamental for this thesis and provide an approach for achieving the first two objectives.

After we looked at the fundamentals, chapter 3 introduces user interface design methodologies such as usability, user interface design and user experience along with basic design principles. We take a detailed look at the iterative design process and introduce approaches, tools and concepts for each stage. Those proven practices shall help achieving the third objective.

Chapter 4 will illustrate the design and implementation approach along with implementation details, design decisions and considerations based on the previous chapters.

A pilot study will be presented in chapter 5. We discuss the approach and the results of linking media automatically with contextual relevant RDF resources.

A usability study for evaluating the developed application is featured in chapter 6 along with the results.

Chapter 7 will provide a summary of the master thesis and will give an insight into the lessons learned during the study and list future research and design possibilities.

2 Related work

This section introduces and discusses the state of the art of relevant research topics. Relevant research includes annotations of multimedia on the Web, Linked Data and standards enabling the standard-based implementation. Additionally, any technology needed for the implementation will be discussed at the end.

2.1 Annotations

> "Annotating, the act of associating one piece of information with one (or more) other piece(s) of information, is a pervasive activity shared by all humanity across all walks of life. " [2]

In general, annotations are information added by users. Annotations allow experts, being familiar with a specific topic, to provide further information for other users. A simple example of an annotation is a comment added to a photo posted on Flickr[2]. Users who describe the content associate textual information with the picture. This information may be processed by search engines and provide easier access to content for other users.

When it comes to annotations, one has to distinguish between discrete and continuous annotations. Discrete annotations can link a medium with a certain reliability, the whole medium to another medium, or one medium with one or more concepts.

In contrast to discrete annotations, continuous annotations link a timeframe of a continuous medium with one or more concepts. Annotations can also have a temporal and spatial relationship to the media [3].

In the following we discuss approaches for annotating content on the Web. For each approach the theory and practical applications will be described.

[2]Flickr [Online]. Available: http://www.flickr.com/

2.1.1 Annotea

Annotea is a shared annotation system designed specifically for the Web and is based on RDF[3].

In general, RDF is a Web standard to structure information. It uses triplets of predicates and objects to describe unique resources, identified through URIs. Additionally, this framework can be used for data interchange on the Web.

One core design goal of Annotea is to feature as much existing W3C[4] standards as possible.

Basically, Annotea enhances Web standards and technologies with semantic Web technologies in order to provide advanced collaboration possibilities. Tools aim at supporting collaborations between multiple users and enhancing the results of those collaborations. Users create metadata for the underlying content in a broader sense. This metadata ranges from annotations to replies, bookmarks and other topics.

The annotation itself is a description, most likely some kind of note. URIs in turn identify individual annotations. Each annotation is composed by and associated to an author. The note, which can be seen as metadata, is related to the document [4].

The requirements and design goals of Annotea are, among others [5]:

- Use of open technologies
- Use of well structured documents
- Each annotation is associated with a URI
- Make use of RDF schema
- Annotations are stored into databases
- Distinction between private and public annotations
- Use of multiple annotation servers

End-users are able to create Annotea objects like annotations, comments, bookmarks, replies, etc. In combination with Web browsers, users are able to share, organize and combine this metadata among other users.

While this solution enhances collaborations for Web documents, it is not possible to annotate multimedia with Annotea.

[3]Resource Description Framework (RDF) [Online]. Available: http://www.w3.org/RDF/

[4]World Wide Web Consortium (W3C) [Online]. Available: http://www.w3.org

2.1.2 LEMO Framework

The basic idea of the LEMO annotation framework [6] is to enhance existing Web annotation solutions. In general, the framework reuses existing standards like RDF, SPARQL, RESTful Web service, Annotea as well as further W3C standards and the MPEG 21 standard. The framework features the following properties:

- Identification of objects by HTTP URLs
- Support of various annotation frameworks
- Addressing of several multimedia items and types
- Annotations are open to other applications

While Annotea is designed for annotating Web documents, the LEMO framework supports multimedia resources and the opportunity of region-based annotations. This concept requires a uniform annotation model and a uniform fragment identification. The basic idea is to support interoperability not only for different annotation solutions but also for different systems. The LEMO framework reuses established vocabularies like the Annotea annotation scheme to gain interoperability.

The fragment identification of this framework builds on URIs. Especially MPEG resources use the MPEG-21[5] standard. We discuss fragment identification in more detail in chapter 2.5.

Those annotations also become part of the Web as they can be shared and referenced through links.

The framework uses a back-end storage for archiving annotations. The basic architecture of the LEMO framework relies on RDF and features a SPARQL endpoint for querying stored annotations. This technology also supports a REST [7] interface for basic CRUD (create, read, update, delete) instructions.

[5] J. Bormans, K. Hill, MPEG-21 Overview v.5, (2002) [Online]. Available: http://mpeg.chiariglione.org/standards/mpeg-21/mpeg-21.htm

2.1.3 Open Annotation Collaboration (OAC)

The basic idea of Open Annotation relies on sharing annotations between different platforms and to support easy interaction for simple tasks. The OAC tries to keep the structure of annotations very simple. Each annotation contains a body and one or more targets to the resources.

Additionally, Open Annotation is based on the four principles within the research domain of Linked Data, which is presented in the chapter 2.3.

The Open Annotation Collaboration revolves around these core principles [2]:

- Focus on scientific purposes
- Annotations contain a body and one or more targets
- The body and the targets can have different authors
- All core entities are Web resources
- Provide help in conflict solving in cases of annotation misinterpretation
- Maximization of interoperability
- Use of established Web technologies for publishing and discovering

The OAC identifies annotations by a HTTP URI and features a body and a target resource. The body and the target are also identified by a HTTP URI. This structure is quite similar to the structure of Annotea [2].

2.2 Annotation solutions

We discussed different annotation frameworks in the last chapter and take now a closer look at annotation solutions as well as their field of application.

2.2.1 Annozilla

Annozilla[6] represents an annotation system especially designed for the Web and the Mozilla Web browser Firefox[7]. The system enables users to create annotations and to associate those to a specific Web page. Annozilla is based on the W3C standard Annotea and follows a similar approach.

The underlying format of the annotations rests upon RDF. The solution additionally features region-based annotations through the established W3C standard XPointer. Annozilla stores annotations on a Web server to ensure availability.

However, this approach features no multimedia annotations and is only available in combination with the Mozilla Firefox browser.

2.2.2 Vannotea

In contrast to many other annotation solutions this system enables video and audio-based annotations. Vannotea is developed by the Distributed System Technology Center at the University of Queensland and is, at the time of writing, in prototype stage. Vannotea aims to enable collaborative annotations in real time.

Additionally, this solution stores annotations on a server, so information can be shared in real time, for example during a collaborative video conference.

The design goals of this solution are collaborative indexing, collaborative browsing, collaborative annotating during discussions of video content at remote locations.

The Distributed System Technology Center developed Vannotea as a desktop standalone system, therefore Web integration proves to be quite difficult to realize [8].

[6] Annozilla (Annotea on Mozilla), [Online]. Available: http://annozilla.mozdev.org/
[7] Mozilla Firefox, [Online]. Available: http://www.firefox.com

2.3 Linked data

The main idea behind Linked Data is to make information more accessible and to explore the Web and related data. Four rules have been established, with the concept of Linked Data [9]:

1. Apply URIs as names for objects
2. Apply HTTP URLs to discover those names
3. Provide reasonable results and make use of established standards
4. More links should be delivered to discover related topics

DBpedia[8] is one example application of Linked Data on the Web. In general, DBpedia hosts the information of Wikipedia in structured RDF format, making it possible to process the available information by algorithms. The solution additionally supports SPARQL queries via an endpoint.

The fourth rule of Linked Data suggests to use Web links to establish relationships to relevant information. This is either achieved by linking content manually or by linking content automatically to relevant topics.

By linking content manually, users first look up resources and attach them to the content. However, this is only feasible for a small number of links. When there is need to process a large set of links or huge amount of information, a manual method would be very time consuming.

By providing information, which can be processed in an automated manner, it is possible to link annotations automatically to relevant resources on the Web. This requires highly sophisticated methods and algorithms. Otherwise the fourth rule of Linked Data is violated by linking the content to wrong resources.

The following chapter introduces two approaches for automatic linking information resources on the Web.

2.3.1 Interlinking resources

The first approach follows the principle of automatically interlinking resources. An interlinking algorithm, which is provided with information sources, links the resources to contextually relevant sources through string comparison.

The solution SILK, for example, follows the approach of interlinking resources. Its main purpose is to find relationships between different sources, to set the links and to publish the data. SILK allows the automated generation of owl:sameAs and other RDF links between resources. Additionally,

[8]DBpedia, [Online]. Available: http://dbpedia.org/About

SILK provides a language for specifying link conditions and enables deploying in distributed environments. It features different methods to operate in different conditions and to enhance performance and lower network load.

SILK and its link specification and selector language opens a variety of options when discovering relationships between sources. This framework uses the technologies RDF and SPARQL.

To be able to make use of the features of this framework, the underlying data has to be formatted in RDF. Otherwise this language will be unable to set relationships [10].

2.3.2 Named entity recognition

The second approach aims at discovering new information. By applying statistical models to unstructured text, entities like people, places, names, etc. can be extracted. In general, this sequence contains several steps like extracting named entities from unstructured data, discovering URIs for the entities and finally structuring the data.

Compared to the interlinking approach, this technology requires no pre-processed data. The initial format of the information does not have to be structured to extract named entites. Extracting named entities is possible with plain text. This leads to a variety of possibilities in linking data. This approach is called "named entity recognition". There are numerous solutions available; Automatic Content Extraction (ACE)[9], Stanford Named Entity Recognizer (NER)[10] and many more. Depending on solution, further processes like structuring the content and discovering Web resources are necessary.

Several solutions exist which enable named entity recognition. OpenCalais[11], a solution for extracting such entities, also tags those. These results can be accessed via a special interface. The underlying format of the original text does not have to be structured. The technology behind this approach makes use of Natural Language Processing (NLP) and Machine Learning algorithms [11].

This solution delivers named entities with no references and therefore URIs need to be established. DBpedia, a project with the goal of structuring the information of Wikipedia in the RDF format, delivers such a huge amount of references.

In chapter 5 we take a detailed look at those approaches and discuss and analyze their field of application, benefits as well as their shortcomings.

[9] Automatic Content Extraction (ACE), [Online]. Available: http://www.itl.nist.gov/iad/mig/tests/ace/

[10] Stanford Named Entity Recognizer (NER), [Online]. Available: http://nlp.stanford.edu/software/CRF-NER.shtml

[11] OpenCalais, [Online]. Available: http://www.opencalais.com/

2.4 Related Web Technologies

We discussed established standards and concepts for annotating and linking data on the Web in the previous sections. Now we take a closer look at related Web technologies and solutions for building such an open standard Web-based application.

2.4.1 HTML5

The new HTML5 standard builds upon and replaces the XHTML 1.0 and HTML 4 standard and introduces new capabilities like audio and video playback [12]. In general, the WHATWG (Web Hypertext Application Technology Working Group)[12] developed the current version in cooperation with the W3C. Both introduced HTML5 in March 2010. However, this standard is still under development and the final recommendation status will be presented in 2022. Currently most Web browsers like Mozilla Firefox, Google Chrome, Opera, Safari and Internet Explorer 9 support the new standard. HTML5 adds further functionalities like support of videos and audio files, drawing within a canvas element, basic offline functionalities, browser integrated SQLite database, JavaScript API, more HTML tags like article, header and footer, Web forms for email addresses and several other features.

2.4.1.1 HTML5 audio and video

Each of the above mentioned Web browsers support different video and audio formats. Table 1 and Table 2 show a compatibility list of the corresponding formats as well as codecs and operating systems. Due to constant updates of the analyzed browsers especially with respect to functionality related to the HTML5 standard, only the most recent versions of the Firefox, Opera and Chrome Web browsers are considered.

	Version	MP3	WAV	OGG
Chrome	17	yes	yes	yes
Firefox	10		yes	yes
Internet Explorer	6, 7, 8			
Internet Explorer	9	yes		
Opera	11.6		yes	yes
Safari	5	yes	yes	

Table 1: Audio codecs tested on Microsoft Windows and Mac OS X

[12]WHATWG, [Online]. Available: http://www.whatwg.org/

	Version	MP4 h.264	Theora	WebM
Chrome	17	yes	yes	yes
Firefox	10		yes	yes
Internet Explorer	6, 7, 8			
Internet Explorer	9	yes		
Opera	11.6		yes	yes
Safari	5	yes		yes

Table 2: Video codecs tested on Microsoft Windows and Mac OS X

By providing a MP3 file along with an OGG audio file transcoded with Theora, all major Web browsers have audio playback enabled. Video files should be provided in the format MP4 with a H.264 codec and in the WebM format.

Users can enhance older browser versions that do not feature this new technology by providing a fallback to Adobe's Flash[13]. This way older Web browsers like Internet Explorer 8 or earlier are not excluded.

2.4.1.2 HTML5 canvas

The W3C and the WHATWG also introduced a feature called canvas with the release of HTML5. It enables the possibility of drawing simple shapes like lines, circles, rectangles and more. Furthermore a JavaScript API enables animation possibilities. In general, the canvas element supports 2D models. However, some browsers like Google's Chrome or Apple's Safari experimentally support 3D models.

Most of those functionalities were available before, but only in connection with a plugin like Adobe's Flash. Since this solution is proprietary, not all users welcome this technology, HTML5 therefore posing a valid alternative.

Adding further functionality is possible by using the provided JavaScript API. The working draft is available on the WHATWG specification site[14].

2.4.2 Cascading Stylesheets level 3

The objective of Cascading Stylesheets is to separate the content of a HTML site from its presentation. This is possible by adding a CSS file to a HTML Web page. Several rules, defined by the developer or designer, structure the content.

The first version of CSS (level 1) was introduced in 1996 and supported basic functionality. However in 1998 the W3C released CSS level 2 and

[13] Adobe, [Online], Available: http://www.adobe.com/

[14] I. Hickson, "HTML Living Standard", (2011), [Online]. Available: http://www.whatwg.org/specs/web-apps/current-work/multipage/

replaced this version by another revision level 2.1. Currently, this version constitutes a recommendation.

The latest version of CSS is level 3 released by the CSS Working Group[15]. The specifications of this standard are available on the W3C site[16].

While CSS in the first version introduced fundamental techniques for structuring the content, CSS2 and CSS2.1 improved the overall functionality. With CSS3, features like animations, text shadow, border radius, etc. were introduced. However, with increasing functionality, the overall complexity also increased. While level 1 and 2 rules are Web browser independent, the latest features require vendor specific prefixes like -moz- for Mozilla, -webkit- for WebKit browsers, -o- for Opera and -ms- for Microsoft browsers.

A huge benefit of CSS is its backward compatibility and that browsers ignore unknown rules. In general, all modern Web browsers support the latest CSS version [13, 14].

2.4.3 Google Web Toolkit (GWT)

Google provides a framework for developing Web applications called Google Web Toolkit (GWT)[17]. It enables developers to create complex browser-based applications using the object oriented programming language Java[18]. With this toolkit, developers are able to debug and test their applications within an integrated Java development environment like Eclipse[19]. Additionally, it enables developers to build dynamic Web applications using AJAX.

In general, GWT converts the Java code into HTML and JavaScript. Developers are able to integrate external JavaScript via a JavaScript Native Interface called JSNI. Additionally, they can define stylesheet rules directly in the Java code and benefit from cross browser support. However, developers are also able to embed a separated CSS file.

Some of the core features of the toolkit are [15]:

- Provide a Java to JavaScript compiler
- Provide widgets and panels for buttons, menu bars, etc.
- Support of remote procedure calls (RPC)
- Cross-browser support

[15] CSS Working Group, [Online]. Available: http://www.w3.org/Style/CSS/members
[16] CSS3 Specifications, [Online]. Available: http://www.w3.org/TR/CSS/
[17] Google Web Toolkit, [Online]. Available: http://code.google.com/webtoolkit/
[18] Java, [Online]. Available: http://java.com/de/
[19] Eclipse IDE, [Online]. Available: http://www.eclipse.org/

A huge advantage of the Google Web toolkit is the support of HTML5. Google integrated functionality for audio, video and the canvas tag. Furthermore, GWT provides an API for local Web browser storage.

However, some features of HTML5, like sliders or specific input fields, are still missing in the latest release.

2.4.4 JavaScript

In general JavaScript is a scripting language for Web applications. Netscape Communications Corporation and Sun Microsystem developed this technology and introduced it in December 1995. In contrast to other scripting languages like shell scripting languages, they developed JavaScript specifically for the Internet.

According to Tiobe[20] JavaScript is currently a very popular programming language.

The design requirements of this language matches aspects like open standard-based, enable cross-platform support and building network-centric applications.

Developers can create dynamic HTML Web sites and applications in combination with CSS and JavaScript.

The performance of JavaScript relies on the engine of the specific Web browser. In addition to basic instructions, AJAX enhances the user experience. It allows asynchronous data exchange between the server and the client. Javascript also allows the manipulation of the HTML tag tree using DOM [16].

[20]TIOBE Programming Community Index, [Online]. Available: http://www.tiobe.com/index.php/paperinfo/tpci/index.html

2.5 Audio and video fragment identification

Annotations are commonly used in combination with different media types like images, audios and video files. However, they regularly refer only to a segment of that media.

While URIs on the Web use a standardized approach to identify Web pages and specific sections, no unified approach exists for continuous multimedia content.

Annotation fragment identifiers address sections of a resource or attach metadata to resources. Several solutions exist that enable fragment identification, but they do not support a uniform approach. This requirement is essential when attempting to provide an adaptable solution. Fragment identification should also support interoperability as well as reusability.

To support several types of metadata and content and in order to deliver a uniform solution, both the metadata and the context should be separated.

Several solutions exist to fulfill these requirements. A possible solution presents part 17 of the ISO/IEC 21000 standard with the title: Fragment identification for MPEG resources[21].

This approach features a URI uniform fragment identifier for MPEG resources like video and audio. Additionally, it is possible to append identifiers through URIs and, for example, to select a timespan through parameters [6, 17].

[21] Introducing MPEG-21 Part 17 - an Overview, [Online]. Available: http://mpeg.chiariglione.org/technologies/mpeg-21/mp21-fid/index.htm

3 Framework design

We discussed the fundamentals for this thesis in the last chapter. Essential approaches like Linked Data as well as related Web technologies and fragment identification outline the foundation of this master thesis.

However, for building usable applications, we have to discuss user interface design guidelines and concepts first. While the last chapter introduced fundamental technologies for building an application, this chapter will introduce principles for designing applications like usability and user experience as well as the iterative sequence of designing user interfaces.

3.1 Usability

In general, usability is all about the interaction quality between human and computers. This includes installation of an application and maintenance of a system. Usability is defined by the following aspects [18]:

- Learnability: The application should be simple to learn and tasks should be accomplished without instructions
- Efficiency: The application should be efficient to use after the users have made themselves familiar with the fundamentals of the application
- Memorability: Even if the users did not use the application for quite some time they should be able to interact with the application without problems
- Errors: The application should not be prone to errors
- Satisfaction: Interaction with the application should be enjoyable for the users

Each aspect contributes to a great user interface that is simple and usable.

By letting the users interact with the system, proficiency can be achieved. The time it takes for the users to accomplish a certain task has a direct impact on an application's usability [18].

3.2 User interface design

> "The way that you accomplish tasks with a product - what you do and how it responds - that's the interface." [19]

The user interface is all about the interaction between the users and the application. It not only describes how this interface is designed but also how it works.

In general, user interfaces consist of several idioms like forms, text, graphic editors, as well as media players and many others. User interface designers or engineers should arrange those idioms in order to present an efficient to use interface [20].

A user interface is efficient and effective to use if the following aspects are featured [21]:

- Simplicity: The interface design of an application should be simple and include only the needed functions
- Familiarity: With previous knowledge and real world experience, users should be able to interact with an application
- Availability: User interfaces should include visual hints and other aids. Identifying these is easier for the users
- Flexibility: The user interface should be adaptable to the current situation and knowledge of the users
- Feedback: It is easier for the users to interact with a system if feedback is provided for each action
- Safety: The application should provide useful error messages and users should be able to undo actions
- Affordances: By applying realistic looks to objects, users know how to interact with the interface

User interface designers should consider each of those aspects when designing interfaces. At first however, they have to design the interface. This process includes several steps, that have to be iterative. In general, the approach of designing interfaces includes user research, task analysis, the procedure of designing models and prototypes and finally testing those models.

3.2.1 User research

At first, the user interface designer analysis the behavior of the users as well as their needs. In general users are not equal. They have different backgrounds, education and experience. Most users have the following characteristics [22]:

- Users are social
- Users make unconscious decisions
- Users are motivated by making progress and reaching a goal
- Users are easily distracted
- Users do not think or work further than requested
- Users create mental models
- Users make mistakes
- Users scan content

This information is based on concepts like "The Magical Number Seven, Plus or Minus Two" [23], the mental model or affordances. The "The Magical Number Seven, Plus or Minus Two" is a misinterpreted concept, which claimed that users can remember seven plus minus 2 things with their short term memory. However, this statement has been disproved [24] and the actual number of things most people remember in a given timeframe is three to four.

It is important to understand what users think, what they want to achieve and how they interact with an application.

Furthermore, the user interface designer should examine the target audience of the application. They have the opportunity to perform direct observations, job shadowing, contextual interviews, case studies, surveys or adopting the concept of personas.

Each of those methods helps to better allocate and furthermore understand the target audience of an application.

3.2.1.1 Mental and conceptual model

The mental model describes an object that a random user has in mind. In case of user interface design the mental model describes the user interface of an application. Users create such a model before they even interacted with the interface. The model is formed-based on information, narratives or previous experiences.

There are also conceptual models which represent the physical model. In case of user interface design, the conceptual model is the user interface. Part of those models are buttons, actions, the screen of the device and several other objects.

In the ideal case the conceptual model matches the mental model. Otherwise the users find the application difficult to use. A possible solution to change the mental model of users is to provide training videos, for example [21].

3.2.1.2 Personas

In general, personas are imaginary people who are using the final software product.

Personas help to better focus on the application. This is usually achieved by improved internal communication and by design solutions that are human-centered.

User interface designers collect information concerning these different characteristics during user interviews. They simplify and combine all these characteristics and create personas in a broader sense. In addition these personas receive names, knowledge, skills, goals, a job, a daily routine and if possible a picture to help the design team identify them more easily.

In the end personas represent groups of target users [25].

3.2.2 Task analysis

User interface experts specify the objectives of an application during the task analysis. This includes user goals as well as unusual events and emergencies. User observations or interviews can deliver more specific input on individual topics.

The focus of the task analysis lies on usability aspects like learnability or efficiency. Furthermore, experts define the minimum amount of features in this stage [18].

3.2.3 Designing models and prototypes

After a detailed user research, designers begin to create models and prototypes based on their collected information. In general, several models exist to design models and prototypes. From start to finish possible methods include activity diagrams, storyboards and sketches, wireframes and mock-ups.

User interface designers usually create activity models first. Individual procedures define the objectives of the users in a broader sense. The designers create the most important and common tasks only. Those activity diagrams have to be easy to read and procedures do not have to be complicated. However, if a sequence is quite complex, the user interface engineer has to review, further improve and simplify it.

In the next step user interface designers create storyboards. Since the design is still in an early state, those are usually drawn on paper so they can be easily changed and improved. Storyboards contain information about the content which will be presented to the users. The focus lies on more complicated tasks. Additionally, the user interface designer can attach notes with detailed information to individual images.

Designers create wireframes after they reviewed these storyboards. These wireframes represent the exact structure of the application. Those contain

information about the content as well as the layout and can be created digitally.

However, mock-ups represent the application in the final design. Mock-ups focus on design details like textures, shadows, colors, fonts and several other aspects. Experts create those by coding, like CSS, or by using graphic editors.

The process of designing interfaces should be iterative.

To test the user interface after the design phase user interface designers create early prototypes.

A method for testing user interfaces is called paper prototype. Designers create simple user interface sketches on paper, and test the sequences of an application with users later on. However, experts can also perform those tests on tablets with mock-ups for example.

Users receive several tasks to accomplish. The tests generally involve a user, a supervisor or observer and a person that simulates the functionality of the computer. With paper prototypes, tests can be accomplished, which illustrate the desired functionality. The tests reveal if the previous design decisions made by the designer are clear to the user. However, a third person replaces individual paper sheets one after the other to simulate a fully functional application.

Using this method, application processes as well as the application design can be tested and user interface designers can easily apply changes [25].

The setup of the paper prototype test is quite similar to the usability test, which will be discussed in chapter 3.2.4.

One essential approach in designing interfaces illustrates Fitts's law [26]. The law is based on one simple rule: The closer or bigger a target is, the quicker it can be reached.

Therefore all objects placed at the edge of a screen have an infinite size, simply because the mouse cursor can not leave the display.

Another concept is the radial menu, also known as context menu. Users usually access it via the right mouse button. However, by aligning the menu entries around the mouse pointer, the distance gets even shorter.

Additionally, user interface designers should separate small objects from each other. It enables users to select the correct button much easier.

By following these simple rules, user interface designers can further improve the usability of applications [25].

3.2.4 Usability testing

Since we discussed several usability concepts and best practices earlier, a quick overview over testing procedures should follow now. In general, testing is a key point in the design - develop - test circle. However, the usability test focuses especially on usability errors of user interfaces and delivers a deep insight into the behavior and interaction of users. Those tests are quite similar to paper prototype tests. However, designers and engineers use compiled code instead of paper prototypes.

Usability tests evaluate whether specific design decisions are reasonable for the users. Performing such tests enable engineers to improve on existing interface designs.

If the final software application does not aim for a specific niche of users, then those tests can be conducted with almost any user. Experts however should consider cultural differences in advance.

In general usability tests can be divided into three methods: moderated, unmoderated and free form test. During a moderated test, a facilitator introduces the users to the test and supervises the whole test. With an unmoderated test the supervisor introduces the users to the test in the beginning, while during a free form test users can explore the application on their own entirely [25].

3.2.4.1 Moderated test

The moderated test offers the opportunity to focus on usability problems of the interface, delivers accurate and precise results and highlights poor design decisions. Additionally, this method reuses some important tasks from the thinking aloud [27] method.

In general the users are asked to complete a list of tasks without any outside help. Supervisors or moderators record the whole test via a screen capturing software and a microphone for voice recording. They oversee the whole test and take notes but must not interact with the testers. After the users completed the tasks, the test is finished and the user interface experts can analyze the results.

Supervisors should prepare the computer, which hosts the application, before starting with the tests. Installed microphones record the statements of the users so supervisors have the opportunity to review the results afterwards. A screen capturing software helps tracking the inputs of the users. Supervisors also have to take care that the developed system is in a default state at the beginning. No inputs or states from previous testers should be visible.

It is important that users test the application on the target device. An application designed for mobile devices such as phones or tablets should be tested under similar conditions.

Supervisors can conduct the tests by either meeting the tester in person

or by conducting a remote test. In both cases the actual test is quite similar.

In the beginning, the users will be briefed on the topic and receive an introduction to the test itself. The supervisor presents the project with a short overview as well as the field of application and the benefits. The testers should get a basic overview and should clearly understand the value of the application.

The introduction to the test is more complex. User interface designers get the best results if the users obey the following guidelines [25]:

1. At first, the users receive a list of tasks to complete. Those tasks represent the most common procedures. Supervisors should not arrange time limits for each task, however if the users get stuck, the supervisor should cancel the task.

2. Testers should get a sheet with several tasks to complete. The wording of the tasks must not contain phrases of the user interface. This way, designers make sure that the users are led by the problem and not by the wording of the tasks or interface.

3. The supervisor or moderator of the test must not respond to questions; especially when the users get stuck. The supervisor should not interact or influence the tester or deliver hints to solve a problem. However, the moderator is allowed to tell the tester when a task is completed.

4. Supervisors should encourage the users to speak out loud what comes to their minds. Especially when they get stuck. This way, it is easier for the designers to retrace the actions of the users. Example: "I would like to close this dialogue window, but I can't see any close button!"

5. During the test, the supervisor should observe the whole test and take notes. This way they document problems and are able to revisit them with designers or experts afterwards.

6. Additionally, supervisors should record the test with a screen capturing software and a microphone. This makes it possible to evaluate the tests even better. The microphone allows to record the vocalized thoughts of the users.

7. The number of testers should be at last five[22]. It is possible to find about 85% of the usability errors with only five testers. To reveal 100% of the usability problems 15 testers are needed, however, the learning curve becomes flatter with each new tester.

[22]J. Nielsen, Why you need only to test with 5 Users, (2000), [Online]. Available: http://www.useit.com/alertbox/20000319.html

3.2.4.2 Unmoderated test

The setup of the unmoderated test is quite similar to the previously introduced approach. Like before the supervisor should install a microphone and a screen capturing software and hand a task sheet out to the users. The application has to be in a default state at the beginning.

However, after the test and the guidelines were introduced to the users, the supervisor leaves the room. Before starting the test, the supervisor should instruct the users to skip tasks if they get stuck.

This way the supervisor cannot influence the users. Additionally, this kind of test is less stressful for the users because they do not feel monitored during the test [25].

3.2.4.3 Free form test

When taking a freeform test, the supervisor can either monitor the test, or similar to the unmoderated test, leave the room. However, the requirements and the setup is identical to the previous methods but no task sheet is needed.

No matter how the test takes place, the supervisor should motivate the testers to speak out loud what comes to their mind. By recording the voices, the results can further be evaluated [25].

3.3 User experience

In contrast to creating a usable interface under consideration of usability aspects, user experience is all about the appearance of an interface. It is responsible for enjoying an application during usage. User interface designers can achieve this by including small details like color schemes, different fonts, images of realistic objects or by applying textures.

The following sections aim to introduce an approach often called "affordances", as well as shed some light on topics such as Web typography.

3.3.1 Affordances

By adding realistic designs based on real world objects, to the user interface, it is easier for users to understand possible functionalities. User interface designers call this concept "affordances".

Features like shadows and textures can improve the usability of an application. By adding a shadow to a button, designers can guide the attention or highlight specific details.

If those features are used correctly, users can identify the functionality of a graphic element by simply looking at it.

Using icons in an application can improve the usability even more. However, too complex, too specific, or too simple symbols can confuse the users. Designers have to find a balance between those attributes.

Textures like metal, plastic, wood or leather can be applied to the design as well. This appearance achieves even more realistic designs and indicates that areas are moveable or that users can press buttons [25].

CSS3 added some features like shadows, gradients or, font-face to design more realistic interfaces. Those concepts are only available for supported browsers. However, with the help of graphic editors, user interface designers can create further design features like highlights or a semi-realistic 3D apperance. By exporting those images, the developers are able to attach an individual design to the user interface.

Companies like Microsoft, Apple and others released suggestions how to design interfaces for their platforms for a better user interaction. The Mac OS X Human Design Interface Design Guideline[23] released by Apple, suggests for example to simulate a light source on the top of the screen. Considering this information, designers should line up all the shadows and inner shadows the same way. This results in a homogeneous and well designed user interface.

Example: By adding a shadow, a border radius and a gradient to a button, users can much easier identify the functionality than would be the case with simple text surrounded by borders.

[23] Mac OS X Human Interface Guidelines, [Online]. Available: https://developer.apple.com/library/mac/

Figure 1: An example of a semi-realistic button

The button in Figure 1 was designed with Adobe Photoshop[24]. By adding a gradient to the surface the button simulates a convex shape. By further applying a drop shadow and an inner shadow, the button gains a realistic 3D effect. An inner shadow of the "Button" text makes it look like it was cut into the button. In the end texture was added to the button by applying noise to the surface.

User interface designers can easily apply similar methods to their interface and develop a much more realistic and better user interface.

3.3.2 Web typography

> "Web typography in particular entails the selection, arrangement and setting of type on the Web to enrich the meaning of text and to provide a framework upon which text can come to life." [28]

The objective of Web typography or typography in general, is to make content legible and readable. By arranging text to a layout or grid, content becomes part of the design. In general, typography and Web typography are defined by the following aspects [28]:

- Legibility: By choosing the typeface design, the rendering of characters and by taking care of micro typography, the content becomes legible. Details like contrast, line height and line width adjustments contribute to this attribute.
- Readability: The layout, the arrangement of text blocks and the rendering of single objects lead to a better readability.
- Measure: The measure or line length is responsible for the reading flow. A line width between 45 and 75 characters leads to a comfortable reading experience.
- Leading: Leading describes a measurement for the line height or the space between two lines. The best results can be achieved with a line height of 1.5 times of the font size.

[24] Adobe Photoshop, [Online]. Available: http://www.adobe.com/products/photoshopextended.html

- Tracking: By adjusting the space between separated characters typographers influence the tracking. The negative space between two letters is called kerning.
- Word spacing: Typographers define word spacing through the distance between two individual words.
- Typographic color: By choosing a color for a text, typographers apply a typographic color to the content. By taking care of consistent color and contrast, legibility can be assured.

Additionally, designers can align content on the left, in the center or on the right hand side of a page. By adjusting content, they can furthermore align text on the left and the right hand side. Typography also consists of different typefaces and fonts. A typeface like Helvetica is a collection of different font styles like Regular, Bold or Thin. In contrast to those, fonts are defined through single styles or weights like Adobe Garamond Pro Regular, for example.

Figure 2 shows the most common styles: Sans-serif and serif fonts. In general, serif fonts were designed for print media and sans-serif fonts for displays[25].

Figure 2: Sans - serif and serif fonts

The right choice of a font can increase the usability. Especially for small screens and small font sizes. A comparison[26] between several user interface fonts - like Segoe UI used on Windows, Lucida Grande used on Mac OS X, Ubuntu used on Ubuntu, Helvetica Neue used on iOS and Droid Sans used on Android, clearly shows that the best reading results can be achieved with the Droid Sans font. However, typographers can choose from several user interface fonts which lead to comfortable reading experience. Font faces especially designed for interfaces are for example: Aller Sans, Azro and PT Sans.

With a CSS level 2 feature called font-face, it is possible to include custom fonts on Web pages. Before the W3C introduced this approach, users had to install specific fonts on their computer to display them. This limited

[25]M. Bowley, A 20 minute Intro to Typography Basics, (2009), [Online]. Available: http://psd.tutsplus.com/articles/techniques/a-20-minute-intro-to-typography-basics/

[26]I. Cylkowsk, What should I look for in a UI Typeface, (2011), [Online]. Available: http://www.design-by-izo.com/2011/10/18/what-should-i-look-for-in-a-ui-typeface/

the selection of available fonts immensely. However, with this new technology, user interface or Web designers can provide single fonts to the users. By embedding TTF (True Type Font), EOT (Embedded Open Type), SVG (Scalable Vector Graphics) and WOFF (Web Open Font Format) formats, Web browsers can display any font. Even older browsers like the Internet Explorer 6, 7 or 8 as well as older software releases of mobile devices are supported.

4 Design and implementation

After we discussed the theoretical fundamentals and essential user interface design concepts, this thesis will now move on to combining aforementioned paradigms, introducing implementation details and documenting the user interface design progress of the thesis itself.

In the following chapter we take a look at insights and documented deliberations made during the implementation of the application. Those are based on the previously introduced concepts and technologies.

Since one of the main objectives of this case study was to design a usable solution, to find problematic processes and to further enhance the user interface, the overall usability and the experience of the clients, we now present the implementation of the previously introduced principles in the context of the prototypical application development in the course of this thesis. For an in depth understanding of the user interface, the reader may view the demo available after installing the prototype according to the instructions given in the Appendix C.

4.1 Audio client

The setup of the audio client is quite similar to the Soundcloud[27] Web application. However, the main focus of the client lies on the annotations attached to it. The design of the audio client pays attention to usability aspects introduced in chapter 3.1 and therefore the design process took several attempts.

4.1.1 Client design

At first the requirements for the audio client were defined. Fundamental actions like adding a title, a description and selecting a fragment needed to be integrated. Additionally, the users should have the opportunity to add several tags. The process of adding an annotation has to be straight forward and simple, to design a usable interface. In the next step, activity diagrams for the audio client were created. Those illustrate the process of the previous introduced requirements. Figure 3 shows the sequence of adding an annotation with the different options.

[27] Soundcloud, [Online]. Available: http://www.soundcloud.com

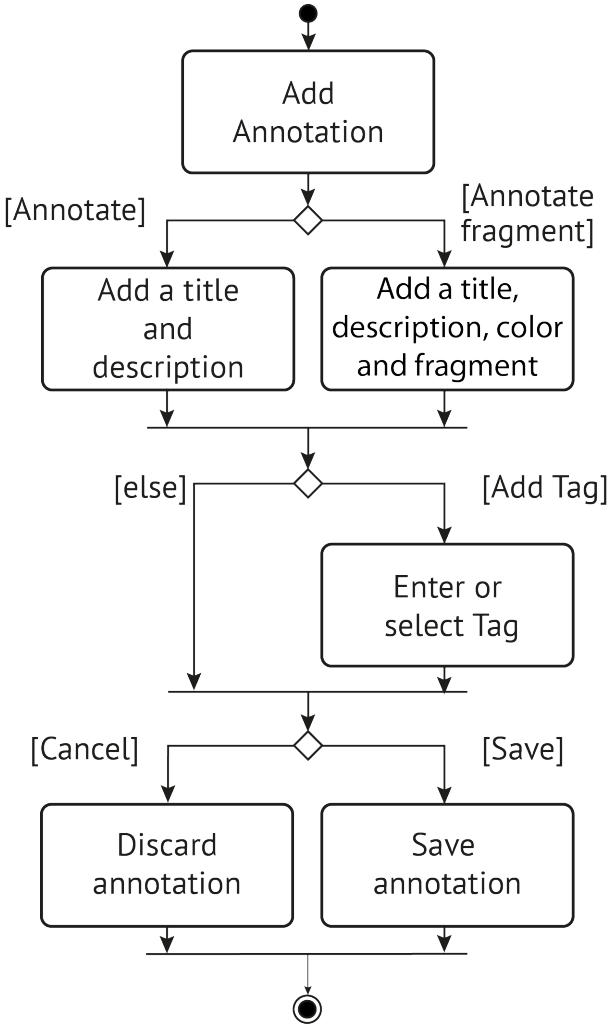

Figure 3: Activity diagram of adding an audio annotation

Those actions were transformed into a storyboard (Figure 4), that illustrates the action of adding an annotation. While this sketch does not pay attention to details, it contains all necessary elements to add an annotation. At first the storyboard shows the audio player (1) in its default state with one annotation (3). The settings for creating an annotation are hidden. However, by clicking on the "Annotate" button (2) the settings (4) are revealed, shown in the second drawing. By entering a text, choosing a timeframe and clicking on the save button (5), the annotation will be created and stored. The last drawing shows the audio client in the default state with the new annotation (6).

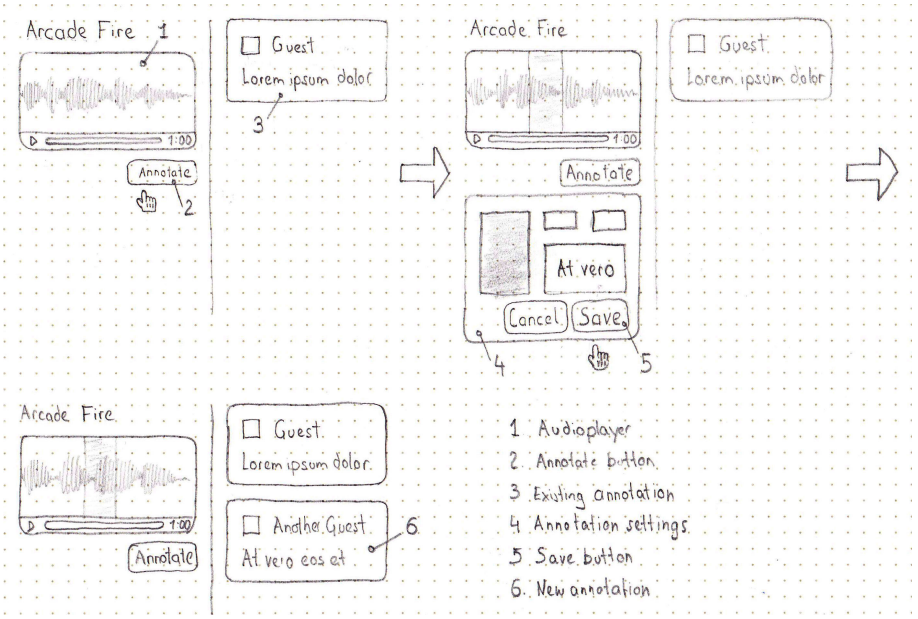

Figure 4: Storyboard of the audio client

In the next step the elements for adding an annotation were combined into a detailed sketch of the audio client. The drawing contains necessary buttons, textfields and labels. The structure of the player pays attention to usability concepts and aspects (Figure 5).

The drawing shows the audio client in editing mode. With the presented elements users are able to add an annotation. Just like Soundcloud, the player (1) also features a waveform. Additionally the player owns a play / pause button, a progress bar and displays the current playback time. The "Annotate" button (2), which reveals the settings is located underneath the audio player. Users define annotations through a beginning (5), an end (6), a description (7), several tags (8) as well as a random color from the colormap (4) on the left side. Users are able to save (10) an annotation or discard the settings (9). Available annotations are located on the right side (11). However this early design was reviewed and slightly changed and enhanced.

Figure 5: Sketch of the audio client

In general, the annotations were moved underneath the player. This is a logical step considering a hierarchic structure of a client. Now the audio player is located on the top. Underneath the player, the settings for adding or editing an annotation can be accessed. By default the options are hidden and can be activated via two buttons, in contrast to the sketch with only one button: One for adding a fragment and one for annotating the whole media. Available annotations by other users are placed underneath the settings on the bottom of the page.

The overall design process took several attempts and in the end the drawings were replaced by wireframes. Figure 6 represents the structure of the application with the player on the top, the setting underneath it and a list of annotations on the bottom.

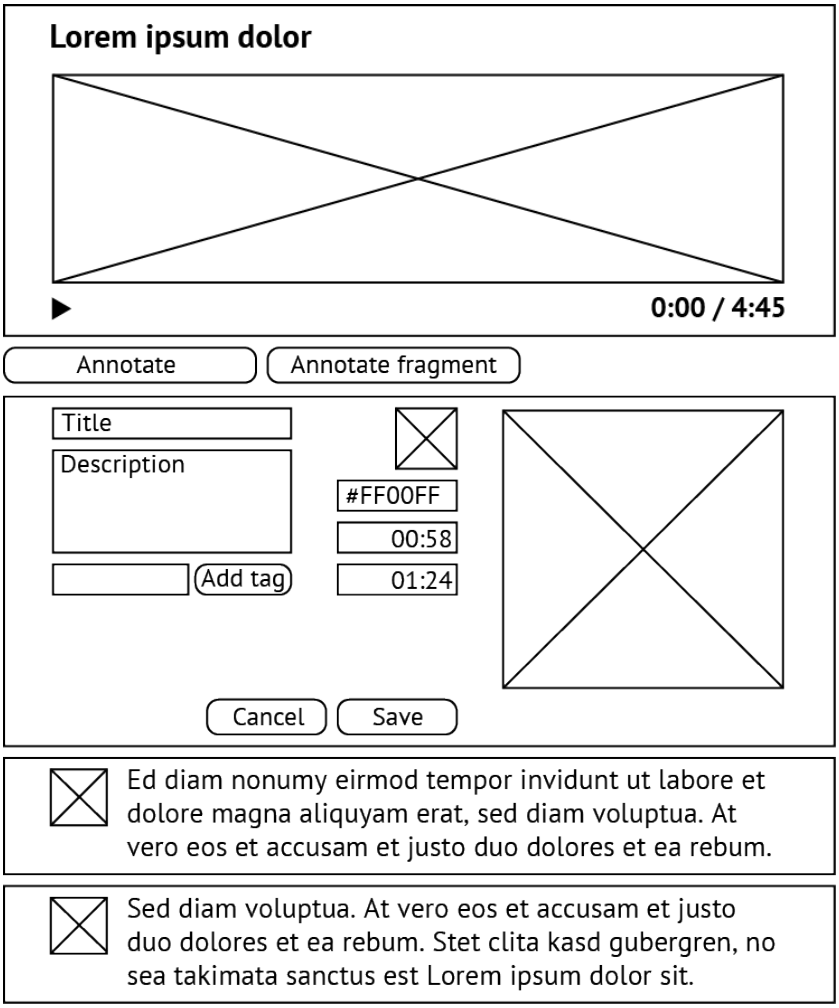

Figure 6: Wireframe of the audio client

In contrast to the sketch, this wireframe contains two buttons for annotating the whole media and fragments. Additionally, a textfield for the title, the tags as well as the color were added and the colormap was moved to the right side.

During the implementation phase several smaller changes and minor enhancements were added to the client.

4.1.2 Player design

The user interface design course of the audio player is quite similar to the client, however it was designed after the implementation was finished. To test the client during the implementation, a simple player with CSS only was created. It featured a title, a play / pause button, the wave form as well as a label for the playback time. Figure 7 shows an early version of the audio client.

Figure 7: Early version of the audio client built with CSS only

This client featured all necessary components to test the audio client. While this early concept did not pay attention to small details or usability issues, it introduced an important concept for all subsequent clients. The player is kept in grey to draw the attention to the colored and more important annotations.

After the implementation, the audio player was designed too. Several sketches were designed on paper and were reviewed afterwards. One example is shown below in Figure 8.

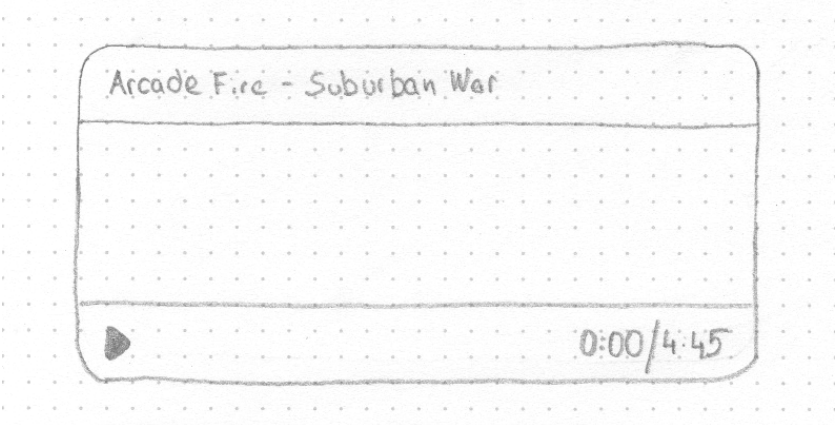

Figure 8: Sketch of the audio client

Those sketches were finally replaced by mock-ups designed with Adobe Photoshop. While most of the usability concerns were eliminated in the early design phase, some details turned out wrong after evaluating the mock-up.

The first version of the audio player had a dark appearance and a quite small play button (Figure 9). The design of this player was analyzed by integrating the player into the implementation. A short test run revealed that the dark colors of the player did decrease visibility with the different annotation colors and therefore the first version was replaced with a brighter design.

Figure 9: Audio player mock-up with a dark theme

Another player with a much brighter appearance (Figure 10) was designed in the next revision. Also a larger play / pause button for a easier interaction was designed. The design is based on the first model but features more smaller details like shadows and highlights. Additionally a semi-realistic 3D apperance was added to improve the user experience.

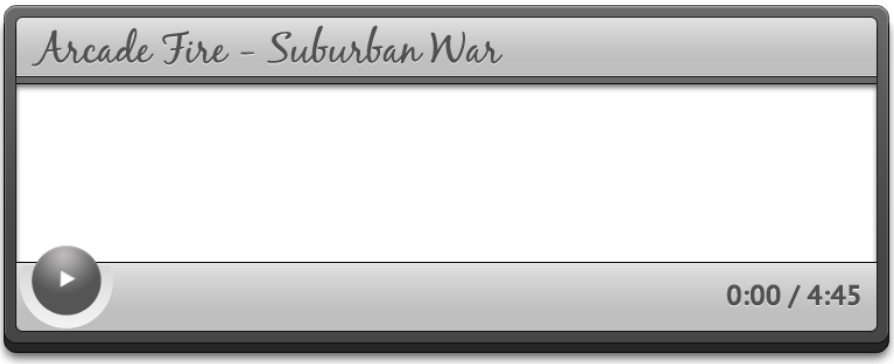

Figure 10: Brighter audio player with a semi-realistic 3D appearance

Another test run revealed the quite bold design of the player. Since this version came with increased horizontal size, less space was available for the annotations. However, the bigger play button improved the usability.

During another design iteration, a new, flatter player was created. Like before, design details like drop shadow, font-face and inner shadows were added, to increase the user experience.

Figure 11 shows a mock-up of the third interface design.

Figure 11: Mock-up of the audio client

In general, the title of the audio file was relocated and the border on the bottom was removed. Additionally, the play button has been designed slightly darker to make it more recognizable. Even with this version of the player the play / pause button did not change its size and the text is still readable.

A final test run revealed no further issues and therefore this iteration remained the final design choice.

4.1.3 Functionality

After the three design - test iterations, the mock-up and the play / pause button of the audio player were integrated into the implementation as separated images. The assembled player and client in its default state is shown in Figure 12 below.

Figure 12: Picture of the audio client

The waveform of the player represents the timeline of the audio. Users can change the playback time through dragging the cursor or by just clicking on the wave form. The playback time changes immediately to present the users the current time. Additionally, a cursor, realized through a gradient below the wave form, indicates the current position. Due to lack of supported events in the WebKit browsers, the wave form is currently only available in the Firefox Web browser. The color of the player and client itself is kept in grey in order to not distract the users from the individual annotations.

To add an annotation to the audio file, the users can choose between attaching it to the whole media or to a selected time fragment. Both opportunities can be accessed by two buttons underneath the player as shown in Figure 13.

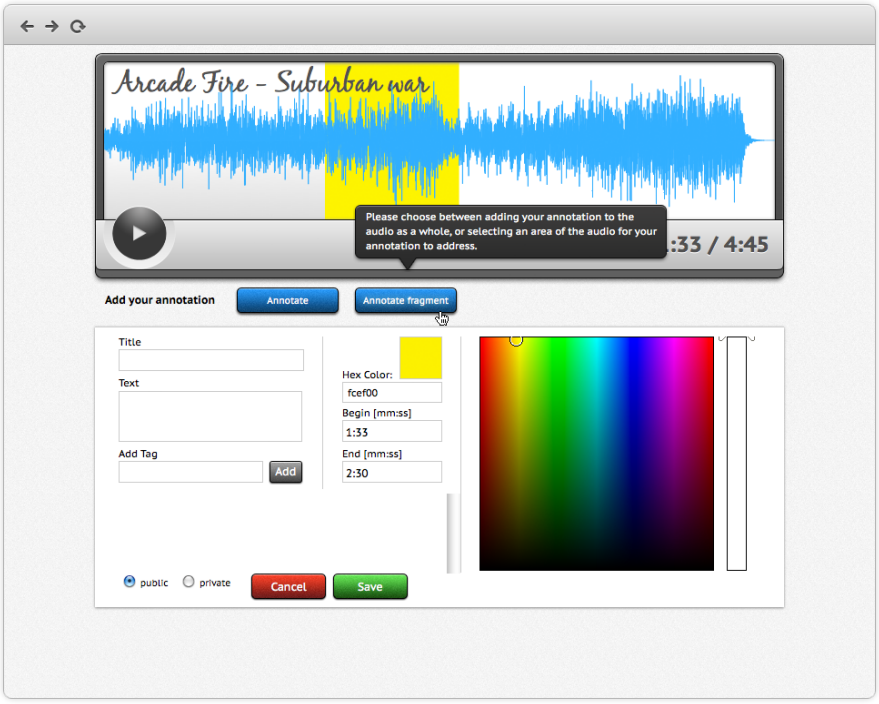

Figure 13: Screenshot of the audio client in editing mode

By hovering one of the two buttons with the mouse cursor, a tooltip appears and aims to explain the difference between annotating the whole media and a fragment. By choosing one option and clicking on the button, users are able to access the different settings. Those are revealed with an animation.

Users who choose to add an annotation to the whole audio can set a title and a description and append several tags to the annotation. If users want to annotate a fragment, they have to furthermore set the start as well as the end of the fragment for the annotation and choose a random color from the Colorpicker. Additionally, they are able to change the saturation of the color through a slider located next to the color map. A small preview next to the Colorpicker, displays the selected color. Alternatively, users can enter a hex code into the ”Hex color” textfield for a specific color .

Users can add time-based annotations by marking a region on the canvas or setting up the respective text boxes. The player highlights the selected region underneath the wave form in the corresponding color (Figure 14).

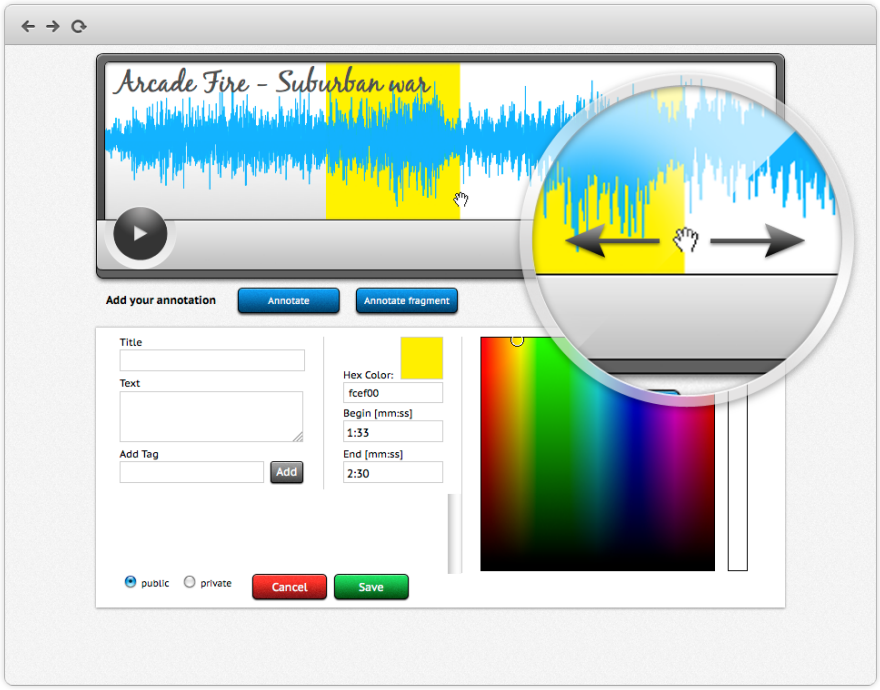

Figure 14: Changing the duration of an audio annotation

By hovering over the begin or the end of the highlighted annotation, the default cursors changes its appearance and indicates interaction possibilities. Users are able to change the duration of the annotations through clicking on one edge and dragging the cursor. By changing the duration the corresponding textfield will be updated. In contrast to applying changes through the progress bar, users can modify the time span in the textfields and the highlighted annotation will be updated.

By entering a description into the "Text" box, named entities will asynchronously be generated and retrieved via a lookup. Hits will be displayed above the waveform. Users have the opportunity, similar to the setup in the pilot study introduced in chapter 5, to either select matching tags through the Tag Cloud or by entering a keyword into the specific "Add Tag" text box. Additionally, descriptions are shown by hovering a tag with the mouse cursor. Each selected tag will be listed underneath the text boxes. Users can remove individual tags from the list by clicking the red button attached to each tag (Figure 15).

Figure 15: Retrieved contextual relevant tags

Although the Tag Cloud covers the timeline of the audio player, users are able to access the player by hitting the ”alt” key on the keyboard. A banner below the tags indicates that behavior.

Users can save an annotation by clicking on the save button or discard their inputs by choosing cancel. Either way, the settings fade out with a transition. In general, the application saves annotations into the database, displays them underneath the audio player and highlights the duration in the progress bar (Figure 16).

The selected tags are listed below the description. Users can read a short description of those tags by hovering one tag with the mouse cursor.

During playback, the player highlights individual annotations. If the current playback time reaches the beginning of an annotation, the shadow of the corresponding description changes it color to yellow. This indicates the relevance of the annotation. If the description is located at the end of the list, the scrollbar automatically moves to the end in order to display it. Additionally, the player changes the opacity of a timeframe if the mouse cursor hovers a corresponding description.

Figure 16: Audio player with an attached annotation

Each annotation features several options. Those are: Replying, replying with an annotation, deleting, editing and playing the annotation. Replying adds another description below the selected annotation and enables users to comment an annotation. Replies with an attached annotation work the same way. However, authors can select a timeframe. Deleting simply removes an annotation. Editing enables the users to change their annotation i.e. their description, timeframe and color. The play option plays the corresponding annotation from the beginning and automatically stops at the end.

Users can access the options on the bottom of each description. The annotation hides the symbols by default but users can display them by hovering a specific annotation with the mouse cursor. The appearance is achieved with a CSS3 feature called transition. Furthermore, each symbol displays a tooltip with a short description (Figure 17).

Figure 17: Options and tooltips of an annotation

4.2 Video client

For the design of the video client, existing applications like Vimeo[28] or YouTube[29] were studied.

As before, the goal was to design a video client that focuses on annotations with high usability as well as user experience aspects. Additionally, the design and usability has to refer to the audio client in order to present a similar experience. The design process of this client consisted of several stages as well.

4.2.1 Client design

At first, requirements for the video client were defined. Similar to the audio client, users should be able to annotate the whole video or just a fragment of it. In general, users need to define a description and be able to apply several tags to the annotation. If they choose to annotate a fragment, they have to be able to select the begin and the end of an annotation and pick a random color. Additionally they should be able to mark a region in the video with a specific shape.

In the next step, activity diagrams were created to define the process of the most important tasks. Figure 18 illustrates the action of adding an annotation to the video client and shows the different options.

With this simple and straight forward process, it is possible to design and develop a user interface that pleases the usability aspects.

[28] Vimeo, [Online]. Available: http://vimeo.com/

[29] Youtube, [Online]. Available: http://youtube.com

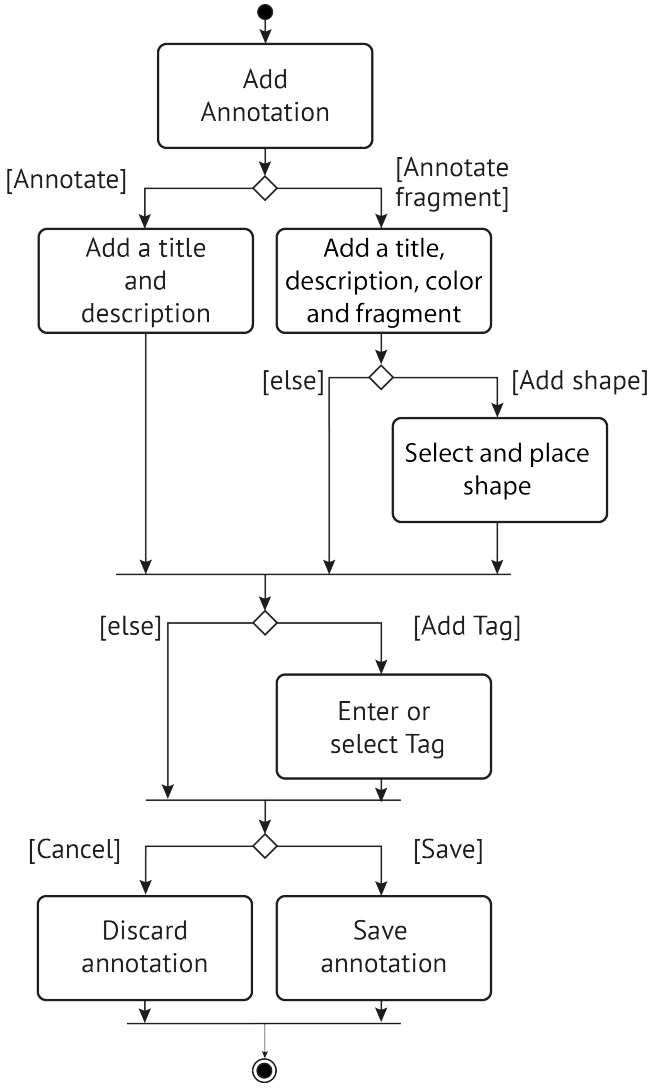

Figure 18: Activity diagram of adding a video annotation

In the next step, a storyboard (Figure 19) for illustrating the sequence of adding an annotation was created. The first drawing shows the setup of the video client which is quite similar to the audio client. However, the player (1) is limited to video playback only. Previous defined annotations are located next to the player. By clicking the ”Annotate” button (2), users are able to define annotations through the settings (4), which are hidden by default. To store inputs, users have to press the ”Save” button (5). Like before, the settings will be hidden and the new annotation (6) will be appended to the list.

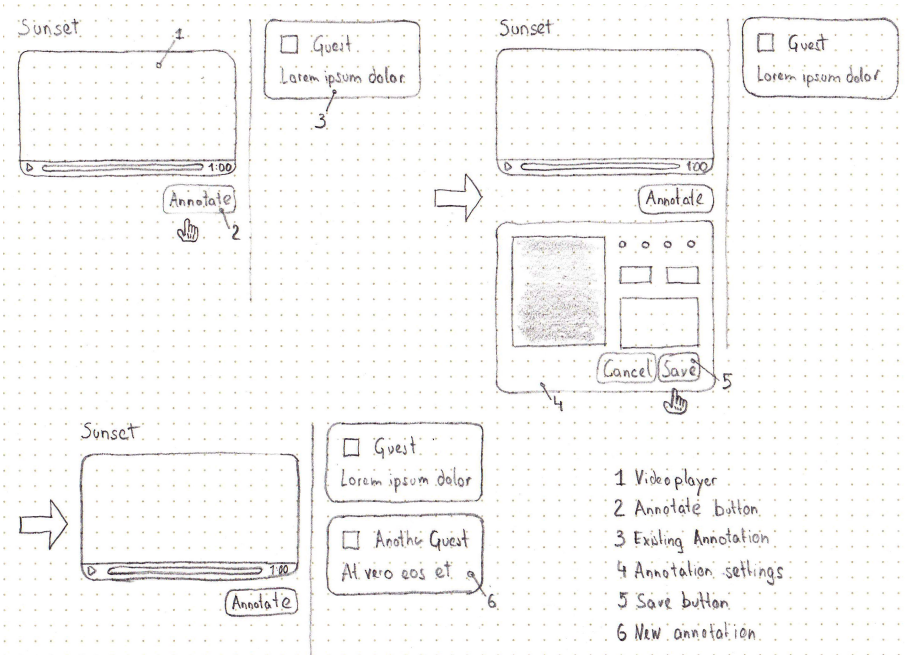

Figure 19: Storyboard of the video client

Furthermore, detailed sketches of the video client were created. The setup of the video client looks quite familiar, however, the settings differ to the ones from the audio client. In general, Figure 20 shows the setup in more detail. This drawing does not focus on details, but pays attention to usability aspects. In general the sketch contains the player (1), the annotate button (2), the settings for adding an annotation (3) as well as the list of already defined annotations (12). In contrast to the audio client, those settings feature different shapes (5) for spatial-based annotations. However, the text fields for the begin (6), the end (7) as well as the description (8) and the tags list match the audio client.

In the end, those sketches were reviewed and improved, to ensure a usable interface.

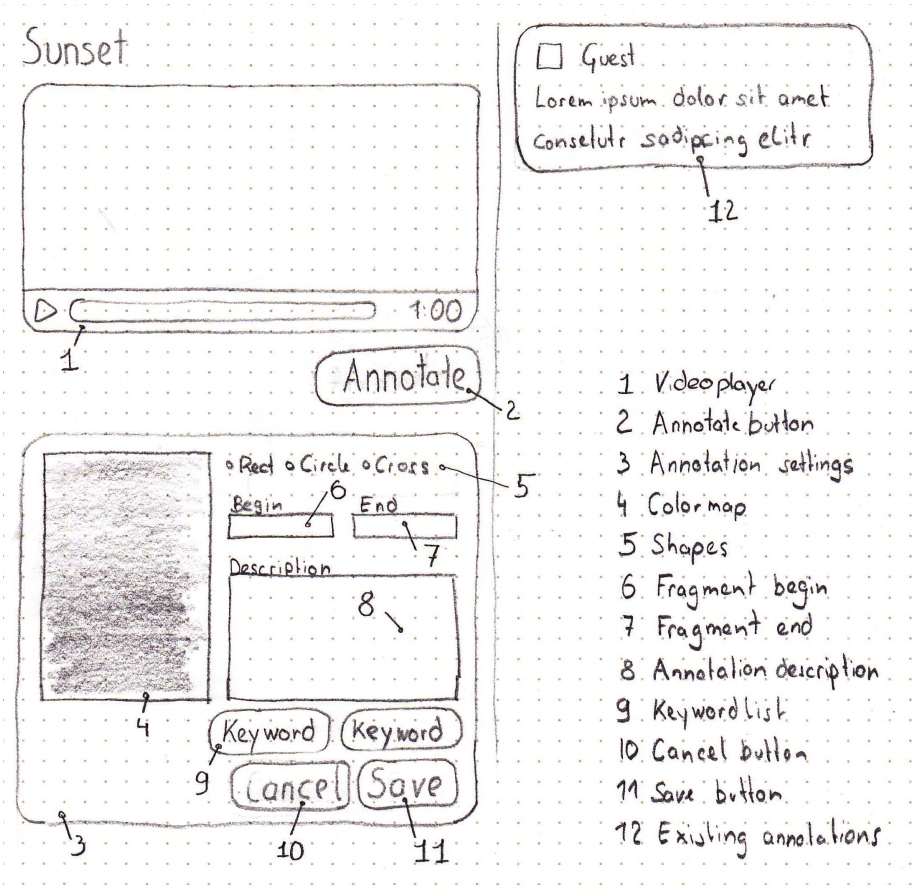

Figure 20: Sketch of the video client

In the next step, a more detailed wireframe (Figure 21) diagram was created, to allocate the exact positions of the elements. This wireframe presents an updated version of the video client sketch. It contains two buttons for adding an annotation to the whole video and annotating a fragment, as well as several radio buttons for different shapes. Additionally, the descriptions were relocated to the bottom of the page.

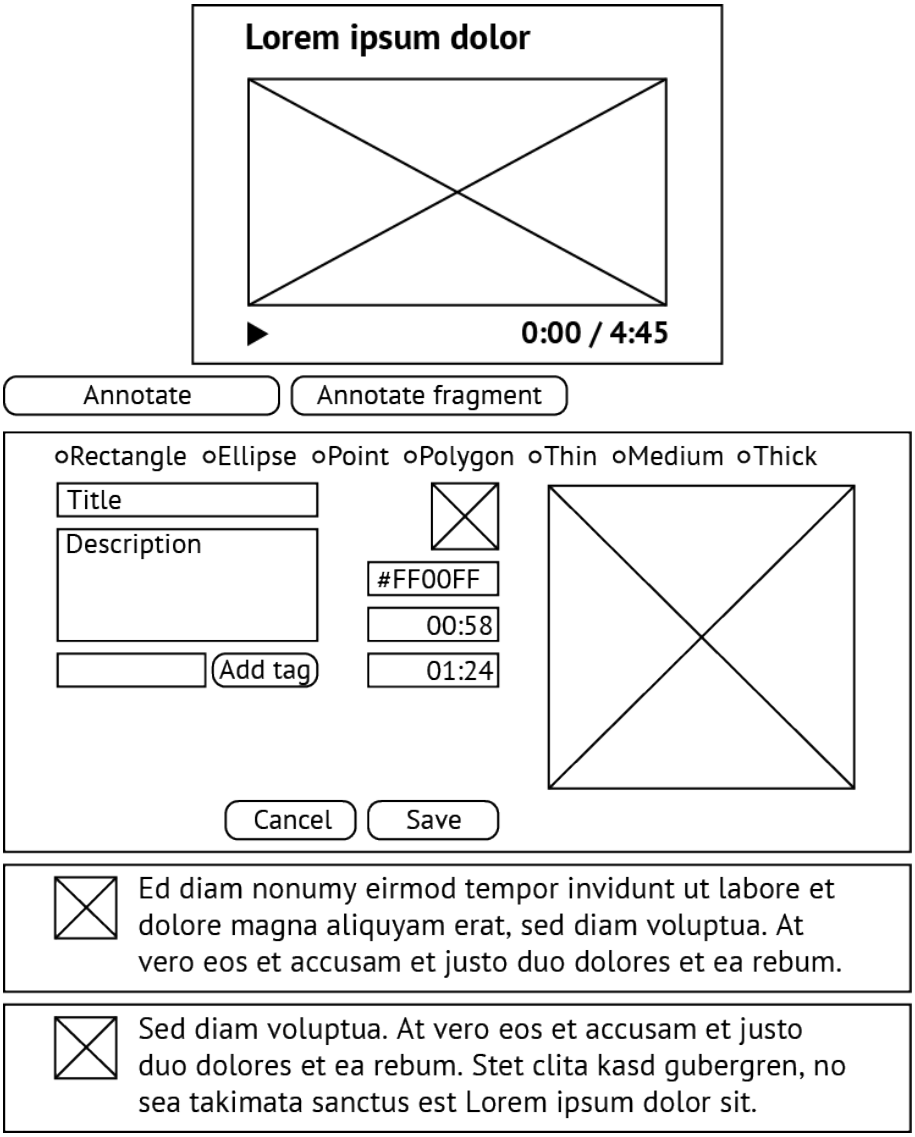

Figure 21: Wireframe of the video client

4.2.2 Player design

The goal of this study, was to design a clean and minimalistic interface for the video player. The requirements were to design a video player similar to the audio player and to please the usability aspects. The experience should also refer to the audio client.

At first, a player with CSS only was created to test the clients. The player contained all necessary elements like a playback time, a title and a play / pause button (Figure 22).

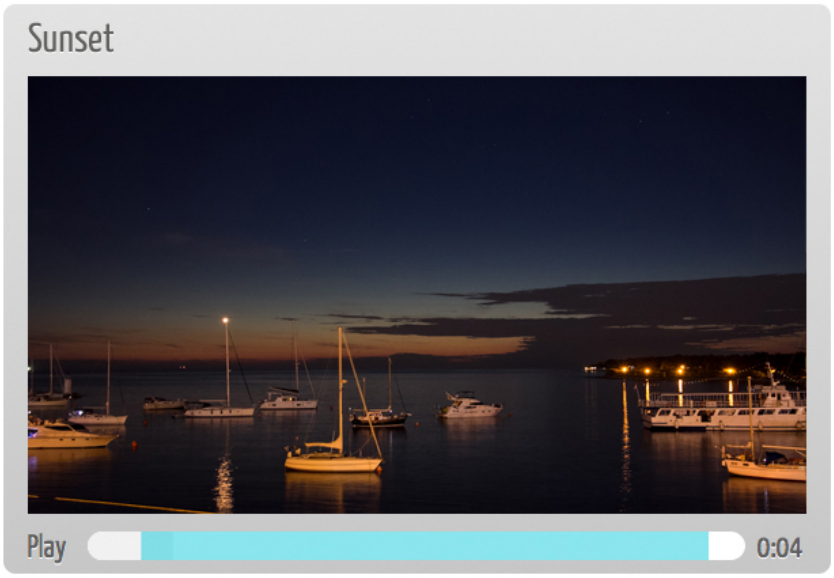

Figure 22: Video client designed with CSS

After the implementation phase, the design - test iteration similar to the design process of the audio client started. At first a simple sketch of the player (Figure 23) was drawn to focus on the usability aspects. In general the goal was to design a functional video player that highlights the different annotations and that is a pleasure to use.

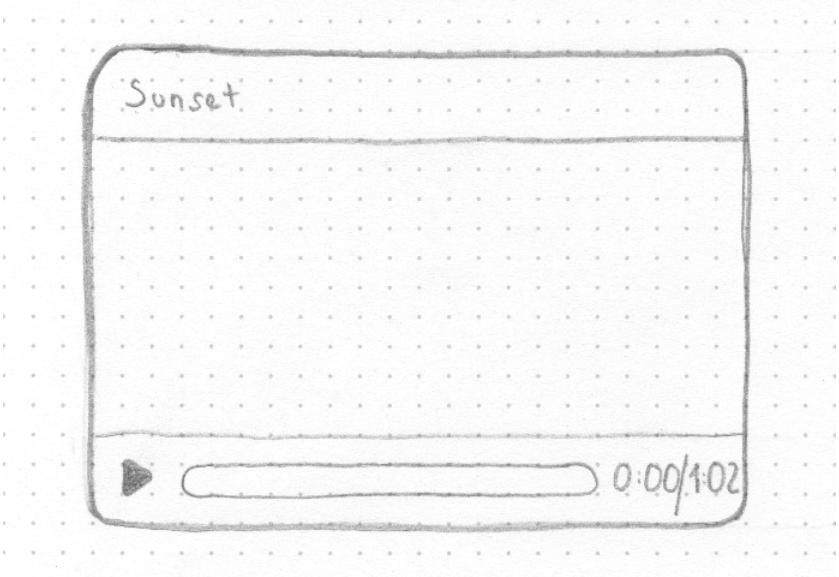

Figure 23: Sketch of the video client

In contrast to the audio player, this player features a separated progress bar. It displays the progress of the video playback and enables users to change the playback time. While the wave form of the audio client represents the progress bar of the audio client, this bar is located on the bottom of the player. In general the design incorporates the 16:9 format of the video.

After the drawings were reviewed, mock-ups were created. Since the design of the dark audio player did not work in combination with the annotations, a similar design was not created. However, a video player that matches the appearance of the second version was designed. The mock-up is shown in Figure 24.

A large play / pause button was added as well as some details like shadows and highlights. This version was designed for videos with a resolution of 640 x 360 pixels. However, the design of this player was too bold and the video was to large, thus less space was available for the annotations. Therefore, another redesign of the player took place and a second mock-up similar to the audio player was created.

Figure 24: Mock-up of the video player

In general, the title was relocated and the bottom border was removed. To highlight the play button, it has been designed slightly darker. Additionally, the original resolution of the video was changed to 512 x 288 pixels. Like before this version also features details like shadows, highlights and has a semi-realistic 3D apperance to improve the user experience. The updated and final version of the video player is shown below in Figure 25.

Figure 25: Mock-up of the final video player design

4.2.3 Functionality

After the design procedure, the player and the play / pause button were integrated into the project as images. A picture of the video client is shown below in Figure 26.

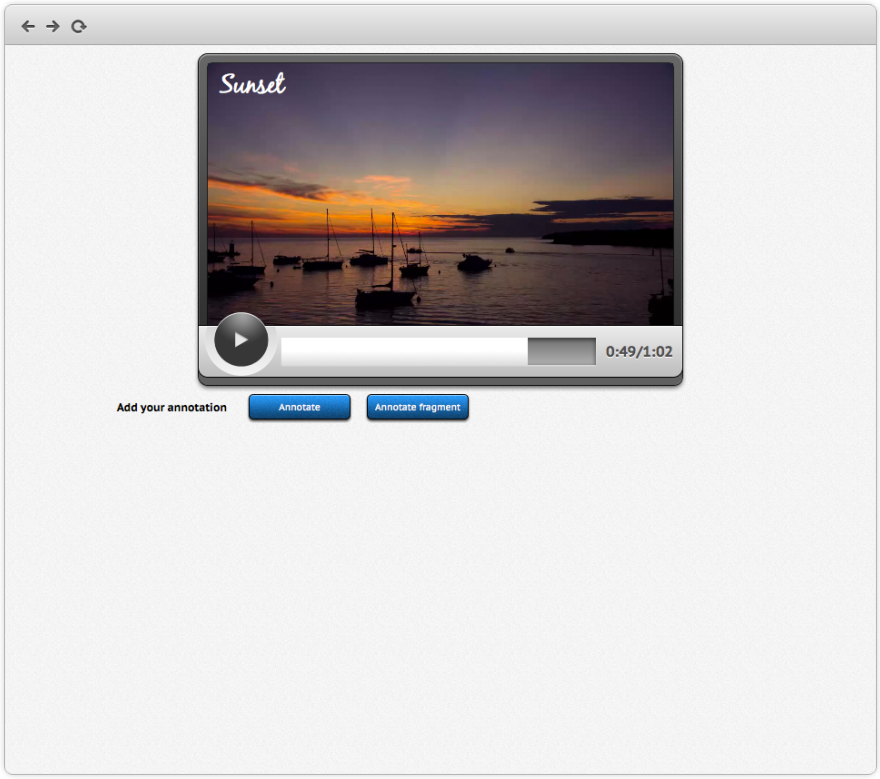

Figure 26: Screenshot of the video client

Users can access the annotation settings via two buttons below the player. One button enables the possibility of annotating the whole media, while the other supports continuous annotations.

In general, the video client features the same annotation settings. This includes the text boxes, labels as well as a color map for the annotations.

Users can add time-based annotations through the progress bar, similar to the audio client. Furthermore, spatial-based annotations are enabled. Users are able to draw different shapes onto the video canvas. This feature uses the HTML5 canvas layer capabilities. Users have the opportunity to choose between a rectangle, an ellipse or circle, a point, a polygon or not to add any shape (Figure 27).

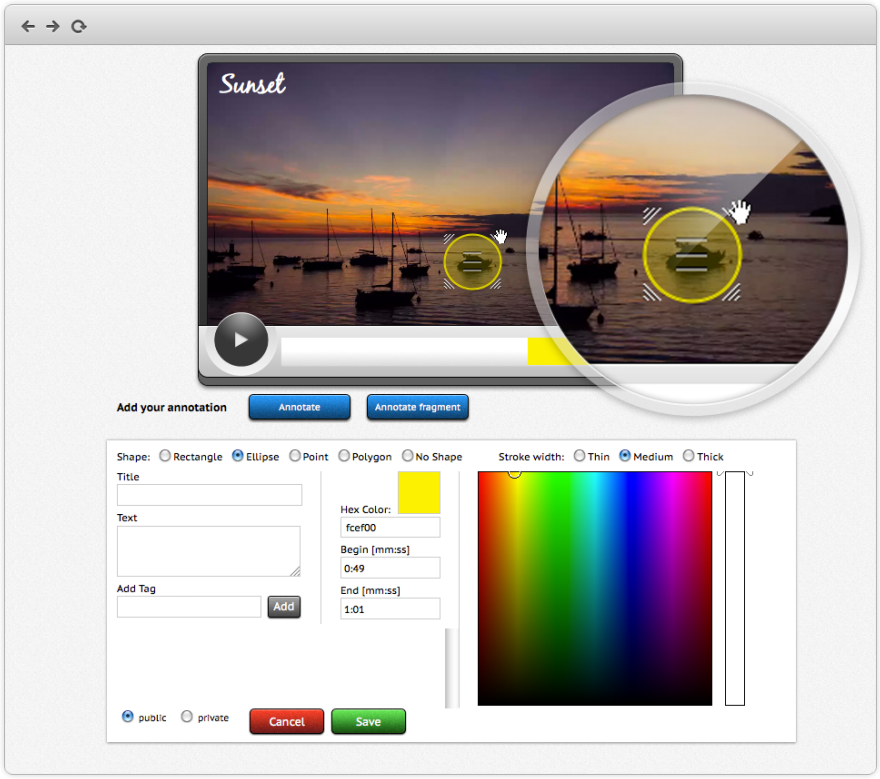

Figure 27: Adding an annotation to the video client

By default, the ellipse shape is selected. However, users are able to choose between the provided options. Additionally, they can change the height and width of the rectangle and the ellipse through clicking and dragging an edge of the shape with the mouse cursor. By hovering the edges, the default mouse cursor changes its appearance and indicates interaction possibilities. However, users can not change the diameter of the point. The polygon allows setting of individual points which will be connected in the order they were placed. Additionally, the users have the opportunity of not adding any shape. They can also choose between different stroke sizes for the shapes. The possible choices are: Thin (1 pixel), medium (2 pixels), thick (3 pixels).

The video client features the same color map, where users are able to pick a random color or enter a specific color into the ”Hex Color” text field. A small preview box displays the selected color. It will be applied to the timeframe in the progress bar and to the shape.

Similar to the audio client users can change the duration of an annotation in the ”Begin” and ”End” text field or through the progress bar below the video. By hovering over one edge of the timeframe, clicking and dragging the mouse cursor, users can shift the duration (Figure 28).

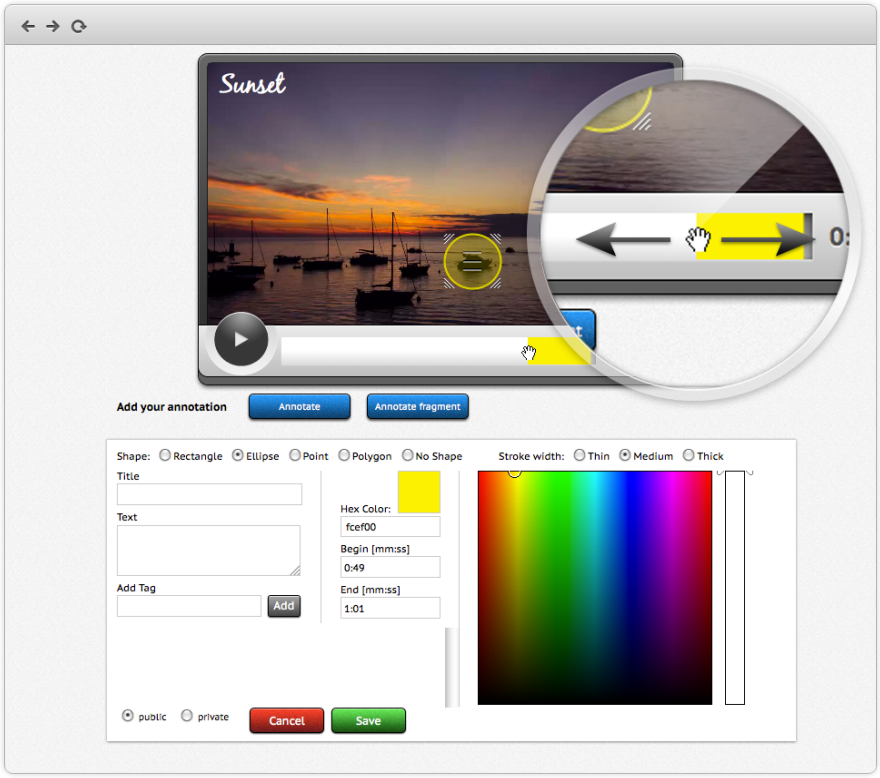

Figure 28: Editing of a video annotation

The video client also supports semantic tags. By entering a description into the ”Text” field, contextually relevant tags will asynchronously be retrieved and displayed. The users have the opportunity to select matching tags by selecting them in the Tag Cloud or by entering specific tags into the ”Add Tag” text box. Users are able to access a short description of the tags by hovering them with the mouse cursor.

The selected tags are listed below the ”Add Tag” text box. Similar to the audio client, users are able to remove previously added tags (Figure 29).

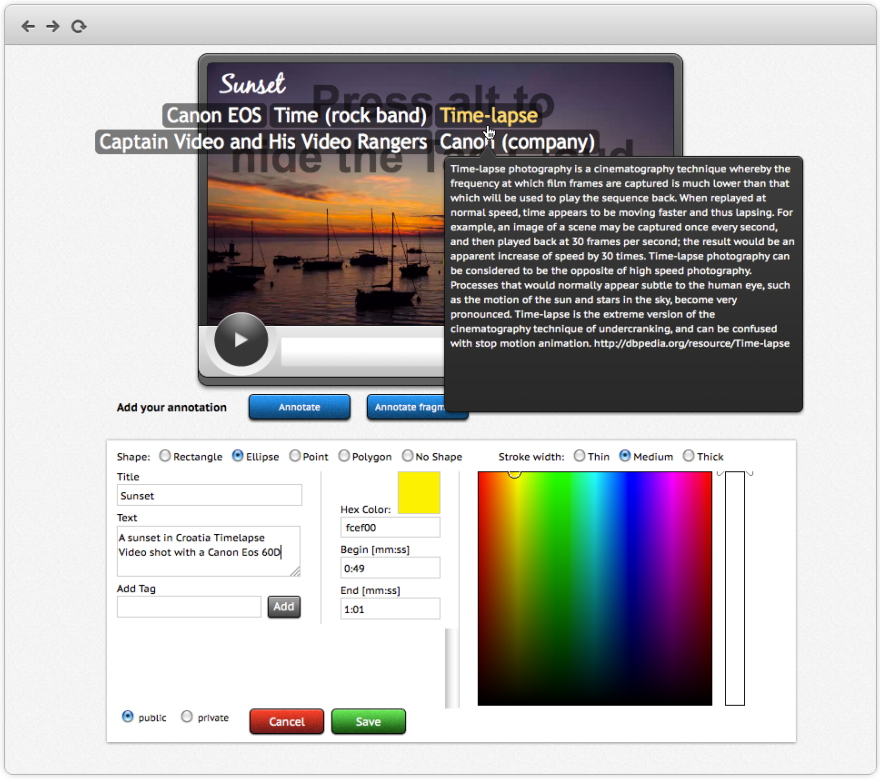

Figure 29: Highlighting of a video annotation

By hitting ”Save” the annotation will be stored into the database, while ”Cancel” discards the entered values.

After saving, the annotation will be listed below the video player. The newest one is always listed on top of previously defined annotations.

By hovering a specific description, the shadow of its box turns yellow and the attached shape will be displayed. At the same time, the options are revealed (Figure 30). A tooltip describes the functionality of the icons more precisely.

Similar to the audio client, users are able to comment an annotation via ”Reply to annotation”, reply with an annotation, or to edit, delete or play the posted annotation. Edit allows the users to change the color, title, duration, tags and description. The play option plays the annotation form its beginning to its end. During playback the shape will be highlighted.

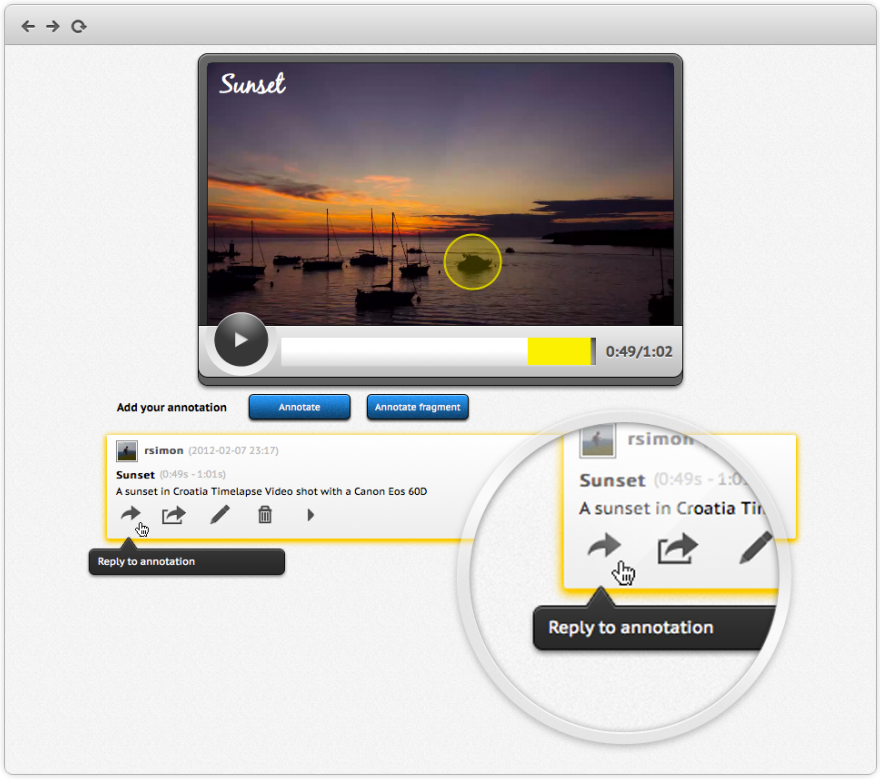

Figure 30: Highlighting of a video annotation with its options

4.3 Technical implementation background

The next sections will cover the documentation of the discussed implementation. This includes some implementation background to the annotation server as well as the audio and video client.

In general, the University of Vienna developed the YUMA framework[30] in combination with the AIT[31]. This project features an image and a map client and has been developed with the Google Web Toolkit (GWT).

4.3.1 Annotation server

The Austrian Institute of Technology (AIT) developed the YUMA annotation server which has kindly been made available for this master thesis. Because the video and the audio client are part of the YUMA annotation project, the server has been reused for this master thesis. The source code[32] is available through GitHub[33].

Since this server was not part of the implementation process in the course of this thesis, it shall only briefly be mentioned here.

The storage of the annotation server is realized with a Hibernate[34] framework as well as a MySQL[35] database. Both are responsible for the storage of the annotations, fragments and tags. The database features three tables: An annotations table, a plain literals table and a tags table.

The annotations table stores annotation related information like date created, created by, fragment, modification date, object URI, the scope of the annotation as well as the description, the title and the type of the annotation i.e. audio or video. The tags table features tag related information like a description, a label, a type as well as a URI. The plain literals table holds information for language code and further related information.

The server stores fragments of an annotation through a query string similar to URLs. The string holds information about the color in the RGB format and about the duration of an individual annotation. The following example illustrates a fragment from second 0 (begin = 0) to second 57 (end = 57.0274) in the color red (r = 255, g = 0, b = 0). Individual information is separated by & signs:

begin=0.0&end=57.0274&r=255&g=0&b=0

[30] EuropeanaConnect Media Annotation Prototype YUMA, [Online]. Available: http://dme.ait.ac.at/annotation/

[31] Austrian Institute of Technology AIT, [Online]. Available: http://www.ait.ac.at/

[32] YUMA Annotation server, [Online]. Available: https://github.com/yuma-annotation/server

[33] GitHub, [Online]. Available: https://github.com/

[34] Hibernate framework, [Online]. Available: http://www.hibernate.org/

[35] MySQL Database, [Online]. Available: http://dev.mysql.com/

4.3.2 Annotation client

AIT already developed an image and map client which users can download from GitHub[36]. The video and audio client were seamlessly integrated into this project. Those clients use the same database and interfaces.

Unless specifically mentioned, the audio as well as the video client were built with the GWT. The setup of the audio and video client is quite similar to the image and map client.

A simplified hierarchy of the audio and video client is listed below:

- Viewer panel
- Annotation panel
  - Button panel
  - Scroll panel
    - Annotation edit form panel
      - Control panel
    - Annotation tree nodes

The panels are organized as follows:

- Viewer panel: The viewer panel contains the corresponding player, the progress bar and the Tag Cloud.
- Button panel: The button panel includes the two annotation buttons as well as the instruction labels for the buttons.
- Annotation edit form: The annotation edit form panel contains the title, text and tag boxes as well as the control panel.
- Control panel: The control panel includes the Colorpicker and the timeframe text boxes.
- Colorpicker: The Colorpicker is responsible for the display color of individual annotations. Users can select a specific color with the color map or with a hex code dialogue field. Additionally, the Colorpicker features a saturation slider. In general, auroris.net[37] developed the Colorpicker, which was released under the Artistic License. However, this project has been simplified and reused for the audio and video client.

[36] YUMA Annotation client, [Online]. Available: https://github.com/yuma-annotation/client-suite

[37] auroris Colorpicker, [Online]. Available: http://code.google.com/p/auroris/

- Annotation tree nodes: The annotation tree nodes represent the individual annotations. Those are defined through an avatar, a title, a timestamp, a description as well as several buttons for deleting, editing, playing and replying to that corresponding annotation.

4.3.2.1 Technical information of the audio player

The HTML5 audio tag enables the audio playback. The audio file is provided in the formats OGG and MP3 to support common Web browsers like Google's Chrome, Mozilla Firefox, Microsoft's Internet Explorer 9 and Apple's Safari.

The featured track is retrieved from the Free Music Archive[38] which is released under the Free Music Archive license[39].

The progress bar, which contains the waveform, the cursor of the audio playback as well as the time-based annotations has been developed with the new HTML5 canvas technology. Since the canvas element holds graphics until they are removed, methods for erasing and redrawing are needed each time the appearance changes. For a better overall performance, those layers were separated to minimize the method calls.

Additionally, several CSS3 features have been added to the audio client. The progress bar features an inner shadow, the annotations make use of gradients and the tooltips support a drop shadow to simulate a 3D effect. This client also features several fonts which were embedded with the CSS2 approach font-face. For the title of the player, the font "BlackJack Regular" was used. Font-squirrel[40] hosts this font for free but did not release it under a specific license. For the user interface texts, the font PT Sans has been used. This font is available through ParaType[41] and is released under the ParaType Free Font license[42]. All those features contribute to a more realistic interface and directly affect the user experience.

[38] Free Music Archive, [Online]. Available: http://freemusicarchive.org/

[39] Free Music Archive License, [Online]. Available: http://freemusicarchive.org/FMA_License

[40] Font Squirrel, [Online]. Available: http://www.fontsquirrel.com/

[41] ParaType, [Online]. Available: http://www.paratype.com/public/

[42] ParaType Free Font License, [Online]. Available: http://www.paratype.com/public/pt_openlicense_eng.asp

4.3.2.2 Technical information of the video client

The video player is being maintained with the HTML5 video tag. The timeline of the player is quite similar to the audio player. It features two layers for a better performance. One canvas layer is designed for the progress cursor and another one for the annotations.

The video client features the same audio track as the audio player. The track is available through the Free Music Archive, which has been released under the same Free Music Archive license. The video is provided in the formats MP4, which features a H.264 codec, and WebM.

The fonts of this video client, are subject to the same licenses as noted in the audio client.

The geometric shapes were incorporated via the Raphaël[43] JavaScript library and are placed above the HTML5 video layer of the player. The application stores the annotation also into a MySQL database to make it available to other users.

4.3.2.3 Named entity and tag retrieval

The retrieval of the named entities as well as the tagging occurs asynchronously in the background. The application identifies and presents named entities while the users enter a description into the "Text" box. An approach, similar to the strategy in chapter 5 has been acquired for this sequence.

However, instead of using OpenCalais for the named entity recognition, DBpedia Spotlight[44] has been used. This service not only enables named entity recognition but furthermore annotates text with DBpedia URIs. In general, this solution uses four steps [29]: Spotting of phrases, selection of candidates, disambiguation and taking care of configuration settings.

By sending text passages using a RESTful or SOAP Web service, applications can retrieve and further process results. The solution delivers named entities in JSON or XML format. However, those results only contain URIs for the identified phrases. To further present a description to the users, the application retrieves a short abstract of the extracted named entity via DBpedia.

By combining those features, the application generates semantic tags. Users are able to attach those to individual annotations.

[43] Raphael JavaScript library, [Online]. Available: http://raphaeljs.com/

[44] DBpedia Spotlight, [Online]. Available: http://dbpedia.org/spotlight

5 Study 1: Linking media with RDF resources

A pilot study took place before the actual editing process of the thesis itself had began. A detailed study has been performed to investigate the automatic linking process between media and RDF resources based on the rules of Linked Data. An application has been developed to research the results of semantic and augmented tagging of pictures posted on Flickr. Since the goal was to investigate an automatic process, manual linking of resources is not included in this study.

5.1 Goals

One of the main objectives of this thesis was to investigate an automatic linking solution based on the fourth rule of Linked Data as introduced in chapter 2.3. This study has been performed to analyze the benefits and the quality of an automatic linking approach.

5.2 Dataset and study design

In general the idea was to analyze the quality of an automated process. One of the application's task, was to retrieve comments posted to an individual picture and to furthermore link relevant information to this picture by adding contextual relevant tags.

In order to evaluate the quality of the linking process the developed solution should present the generated results to the users for a judgement and should store the information into a database afterwards. The application should also perform a precision analysis and compare the number of relevant results to the number of all the returned results.

Flickr and its Commons section delivered a great starting point. In the first step, the developed solution retrieved unstructured comments of pictures and further pre-processed them. A variety of people comment on pictures hosted on this site and therefore attach contextually relevant information. The application analyzed those sources in a further step.

One approach follows the idea of structuring information in general. OpenCalais, a semantic enriching content solution, extracts relevant named entities from an unstructured text. This approach is called named entity recognition.

The application published the plain text comments of individual pictures via a RESTful Web service and retrieved the named entities afterwards. The underlying format of those named entities was also plain text and therefore required further processes. To link the text and furthermore the picture to resources on the Web, the application discovered a URI. DBpedia, a platform that is hosting all the information from Wikipedia in RDF

format, provides such URIs. Those can be found via a lookup which delivers ranked results.

In the end, the application presented all those features to the users with a simple user interface shown in Figure 31[45]. The solution displayed each picture along with the extracted named entities and a retrieved URL from DBpedia. Simple checkboxes were attached to each extracted named entity for the relevance analysis. After marking all relevant suggestions, the application stored the whole information into a database.

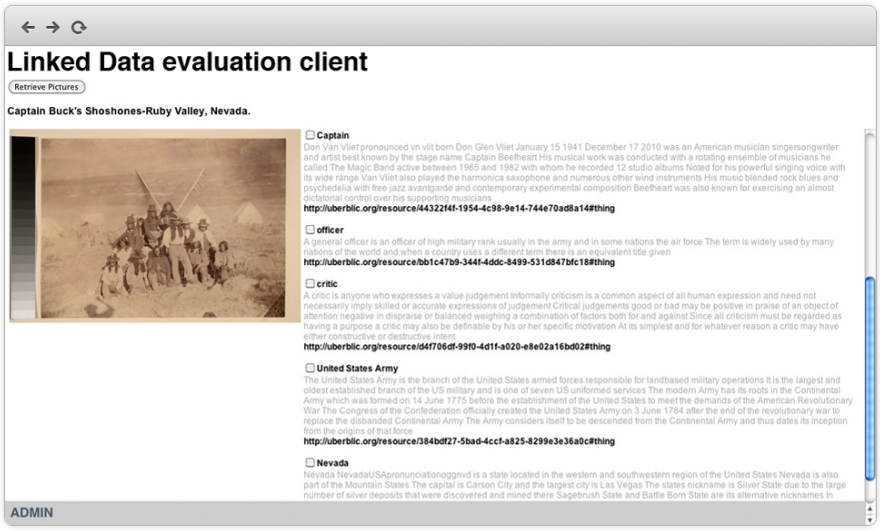

Figure 31: Screenshot of the pilot client

The stored information was accessible via an administration client. Before users were able to access the results, the application pre-processed the stored information and calculated the precision of each picture.

In general the precision is the number of relevant documents returned by the application. Based on the objective of the application, the precision represents a performance measure of the automatic generated suggestions. The performance is calculated with the following equation [30]:

$$Precision = \frac{\#(relevant\ items\ retrieved)}{\#(retrieved\ items)} \tag{1}$$

The initial idea of the application was based on a comparison between OpenCalais and SILK [10], an interlinking solution for structuring information. Because the underlying format for the comments is plain text, this

[45]Captain Buck's Shoeshones-Ruby, George Eastman House, [Online]. No known copyright restriction. Available: http://www.flickr.com/photos/george_eastman_house/4494140424/

solution does not deliver any results. Therefore the evaluation of the project refers only to the results from OpenCalais.

In general, two different Flickr sets with 500 pictures each, has been processed. However, pictures with no comment or with no lookup result were ignored. During the study, the DBpedia lookup service had been discontinued, and uberblic[46] has been used for the description and URI discovery of the second set.

5.3 Results

The results of this pilot study are:

Flickr picture set	Look up service	Precision
The Library of Congress	DBpedia lookup	0.57
George Eastman House	uberblic	0.48
Precision of all pictures		0.53

Table 3: Precision results of the pilot study

Since two different picture sets and two different look up services have been used, no statements regarding the quality of the look up services can be made. However, the average precision result of both services is quite low and therefore the final conclusion of this study is:

> **About 50% of the automatic extracted and detected results are relevant.**

Since this result is quite low, human judgement is essential and an automatic interlinking process does not deliver accurate results.

5.4 Conclusions and lessons learned

Some conclusions of this study are:

1. The most important point is the quality of the annotation added to a media. The more precise the annotation, the higher the quality of the semantic tags.

2. The relevance of the extracted named entities is not always clear either. A short description displayed next to the named entities is helpful. It offers a good reference for more detailed information. Most of the time a specific named entity and the DBpedia and uberblic lookup referred to the same topic. However, the returned result did not match the subject in every case.

[46] uberblic, [Online]. Available: http://platform.uberblic.org/

3. Of importance is the ranking algorithm used by DBpedia and uberblic. For simplicity reasons the application presents only the first hit for each query to the users. Therefore the quality of the descriptions and the URIs are strongly influenced by the ranking algorithm used within the DBpedia and uberblic lookups.

4. Another influential factor is the length of the annotations. In general the length of the annotations added to specific media varies strongly. The more words are used to describe a media or are part of it, the more tags the users get to choose from.

6 Study 2: Annotation tool usability

While chapter 3 presented the foundation and theoretical approach of usability tests, this chapter documents the actual accomplishment and shows the results of the usability tests. The results and conclusion section will document the lessons learned from this study.

6.1 Goals

The usability designer's objective is to design an application that matches the mental model (chapter 3.2.1.1) of the users. This is possible by designing an application that is easy to learn and effective. The usability engineers test the reflections and decisions through a usability test with volunteers. The test itself should be iterative. The designers should consider the design of the application after each usability test.

In the end, the design of the application meets the user interface requirements introduced in chapter 3.2.

6.2 Users and study design

A qualitative research method has been chosen for the usability test. This approach delivers most accurate results and the opportunity to focus on the main problems of the interface. The "moderated test" [25] reuses some important tasks from the thinking aloud [27] method. Readers of this master thesis can find a detailed introduction in chapter 3.2.4.1. For evaluation purposes a group of five people had been selected, containing both people with basic knowledge of computer systems and people with very little to no prior knowledge.

This section describes the approach of introducing the testers to the subject as well as the usability test itself.

At first, the users were welcomed and thanked for their time. Afterwards the testers were instructed concerning the topic of this test; including information about the ongoing master thesis as well as the desired output, annotations in general, the benefits of this master thesis and the current status.

The concept of annotations was presented with help of an image posted on Flickr. The image[47], which is shown below (Figure: 32) features an annotation. By showing and explaining the benefits of annotations the concept could be explained more clearly.

The picture in question shows a night photography of stars. If the mouse cursor hovers the picture, an annotation will be displayed. In this case an annotation added by an expert. The annotation marks the pole star

[47]Schuurmans J., Star Trails, copyright by Jelle Schuurmans, all rights reserved, [Online]. Available: http://www.flickr.com/photos/jelle-s/6096764061/

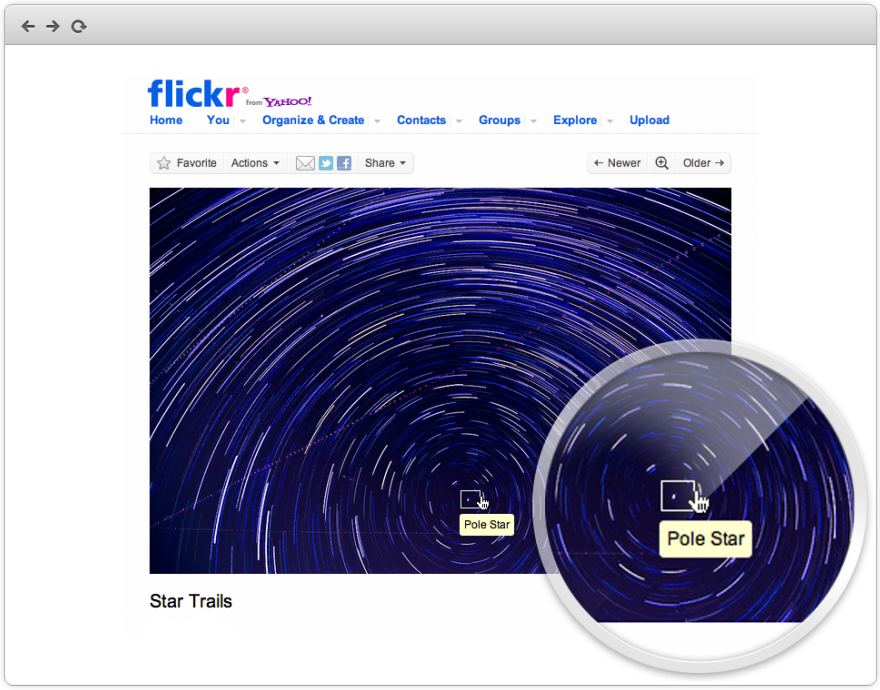

Figure 32: Simple image annotation example

in the picture. For subsequent users or visitors of this site, the annotation delivers useful information.

In an additional step, the benefit of resource linkage was presented to the users. Adding an annotation with a link to the Wikipedia entry for the pole star enabled users to look up additional information.

In the next step the testers were introduced to the usability test and the key points. The probably most important point is to instruct the tester about one simple rule. "Errors are not user errors but program errors, discovering these errors is the very reason this test is being conducted." It is important to inform the testers about this guideline, so they do not feel uncomfortable if they can not solve a task. This statement was repeated several times.

It is of great importance to not lead the users by delivering hints or even the solution to occurring problems. If the users get stuck, the instructor should not tell or show clues or hints, especially if the users ask for help. The instructor should motivate the testers: "Try to solve the problem by yourself. Imagine you are all alone and have nobody to ask how to solve this problem."

If a tester cannot solve a problem within a suitable timespan, then the instructor has to interrupt the ongoing task and lead over to the next in-

struction. If the testers request an instruction to the solution, the instructor can show the solution after the users finished their tasks.

If possible, the instructor should motivate the testers to speak out loudly what comes to their minds. This information is especially useful when reviewing the results. The instructor should encouraged the users during the test, to vocalize their thoughts. A simple example could be: ”I see you have a problem. Can you express what you want to accomplish?”. However, the instructor should not push the users.

In the next step, the users were handed a piece of paper with 16 instructions - 8 for the audio client and 8 for the video client. The tasks for the audio client were:

1. Listen to the whole audio.
2. Add a basic description like ”This is a song called ”Suburban war” by arcade fire from Canada. It has been released by Universal Records.” to the audio.
3. Add another description and set the duration to 12 seconds. Append a random keyword from your description.
4. Add another description where drums, a guitar or vocals can be heard. Append the corresponding keyword.
5. Listen to your second description.
6. Remove your first description.
7. Append another random description to your last description.
8. Edit your last description and add another keyword from your edited text.

The tasks of the video client were:

1. Watch the whole video.
2. Add a basic description like ”This is a time-lapse video that has been shot in Croatia.” to the video.
3. Add another description and set the duration to 4 seconds. Append a random keyword from your text and mark a random boat in the video.
4. Add another description and append the keywords ”Novigrad”, ”Croatia” and ”Mediterranean Sea”. Mark a random section.
5. Listen to your second description.

6. Remove your first description.

7. Append another random description to your last description.

8. Edit your last description and append another keyword from your edited text.

If the testers had no further questions, the test started afterwards.

Some background information about the test itself:

The application has been tested on an Apple MacBook Pro with a Firefox 5.0.1 Web browser. Users with no Mac OS X experience received a short introduction to the most important functions of the operating system. The application has been tested on the development notebook because at that time no public service was available and the setup of the whole architecture would have taken too long.

The surroundings during the test were familiar to the users. Testing location was always quiet and comfortable to avoid stressing the testers.

6.3 Results

This section summarizes the conclusions after the usability tests. This includes the results of the usability tests as well as lessons learned from the tests. At first, some examples from the tests are featured. Afterwards, some reflections, how the user interface and the usability could be further enhanced are presented.

In general, the overall results were quite good. The testers nearly finished all of the tasks, only some tasks seemed problematic to some users. Only one task has been aborted because of an application error and one user finished a task but the database threw an exception (the data could not be stored into the database).

However, some tasks or actions were not clear to every user. For about 50% of the users the difference between "Annotate" and "Annotate fragment" was not clear. Even though all users experienced the tooltip, they were not certain which button to pick.

The first tester had problems to define the begin and the end of an annotation. In the first version of the client, the entity of the time points was set to seconds but this information was hidden to the users. This problem had been changed after the first test.

By hovering over on end of an annotation in the progress bar with the mouse cursor the users can change the duration of the annotation. Only one tester recognized this possibility.

Most of the time, the Tag Cloud did not show up. This can be caused by too long response times or by not identifying any named entities. Therefore the users added tags from their descriptions through the "Add Tag" text box. Just because the vocabulary is limited to countries and cities, the testers added not all tags. However, if the users entered a keyword and clicked the "Add" button, the task was checked.

Sometimes the highlighting of the annotations automatically caused the Web page to scroll down and the canvas and the testers were not able to access the player any longer. By using the arrow keys on their keyboards the testers were able to rectify this situation.

One of the tasks asked "Remove your first description". Some testers had problems to identify the order of their annotations even though the header of an annotation displayed the time.

The article "The Psychologist's View of UX Design"[48] discusses some key points that could be followed during the test. For example:

Testers will not think further or work more than asked

This behavior has been noticed during the tests. For example: The title of an annotation does not necessarily need to be declared. The testers who noticed this behavior did not enter a title during the whole test. Another example would be the video client. The users can choose between four different shapes, but just because the default shape is a circle, no user changed or even noticed the radio buttons for the other shapes.

6.4 User interface improvements

The following list introduces solution approaches to usability problems of the application. Since those are featured in this section most of the improvements were not implemented.

At first, the unit of the time fields was changed. After the first test, the unit was changed from "seconds" to "minutes and seconds". Additionally, a hint was placed next to the title of the field.

The problem of picking the wrong annotation button can be eliminated through improving the explanation of the annotation buttons or by deactivating the time fields and the color map. This improvement has been added to both clients after the usability tests. Figure 33 shows the disabled user interface elements. Users can not access those settings while annotating the whole media. Additionally, the text boxes for the begin and the end of an annotation show the whole duration of the media. This way errors can be prevented.

[48] S. Weinschenk, The Psychologist's View of UX Design, (2010), [Online]. Available: http://uxmag.com/articles/the-psychologists-view-of-ux-design

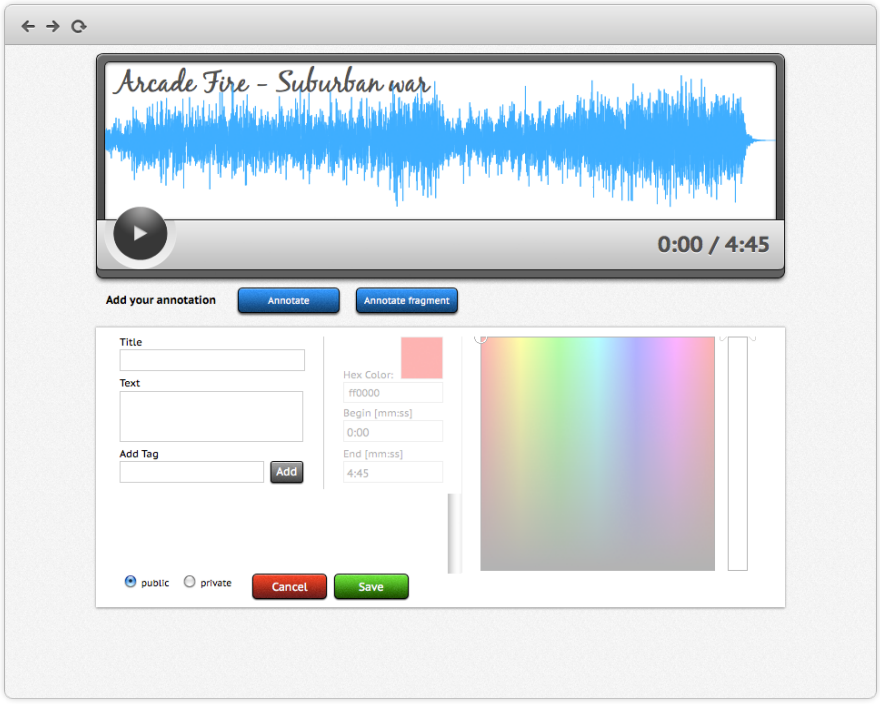

Figure 33: Disabled audio settings

To indicate a background process for generating and retrieving tags, a wait cursor can be applied. Additionally, a label can inform the users about the ongoing process.

Some users had problems to identify the order of their annotations. This problem can be eliminated by improving the appearance of the time stamp next to the annotation title.

7 Conclusions

7.1 Conclusions

This master thesis not only contains a theoretical part, but also features a practical implementation. The main objective of the implementation is to prove the possibility of building such a Web and open standard-based application that pleases usability aspects.

While it is possible to develop such an application with the capabilities of the LEMO framework and the new HTML5 technology in combination with the GWT, a public service for demonstrating the functionality is not available. However, this master thesis features an appendix with the compilable and deployable projects and a detailed installation guide. Readers find a DVD attached to this master thesis, that contains the source code, documentation of the source files, a detailed installation guide and two WAR files for deploying the audio and video client as well as the server.

Due to the required use of a MySQL database it was not possible to provide a public access to the working project. However, with the detailed instruction guide included on the DVD, it is possible to install the requested environment, the server and the application on a computer and to take a closer look at the presented solution.

7.2 Lessons learned

The lessons learned section focuses on experiences, made during the master thesis. In general, it is possible to develop an open standard-based solution with resource linkage. With help of the LEMO framework, new features introduced in HTML5 and established tools and languages like GWT and JavaScript, developers can create powerful Web applications.

The study of resource linkage revealed that the automatic generated and retrieved results are not accurate. However, with a semi-automatic approach, users can judge the suggested results and therefore the rules of Linked Data are not violated.

The usability study revealed the benefits of a corresponding test. During five tests, with technical adapted and technical unversed users, the design of the developed clients could be examined. The results highlighted some problematic design decisions as well as areas for potential improvement.

7.3 Future work possibilities

The following tasks and approaches were researched, but due to lack of available features in Web browsers and GWT, they were not included in this thesis and should therefore be implemented in future studies:

At first, a Flash fallback for older Web browsers should be added. This way, excluded browsers like the Internet Explorer 7 or 8 will be supported. Currently a Flash fallback is not provided in Google's Web Toolkit. However, this feature can be added in the future. Additionally, the HTML5 canvas element should be replaced to support obsolete Web browsers. A possible approach follows the idea of creating HTML div elements in combination with CSS.

At the end of the master thesis, the WebKit browsers still did not support the necessary events to feature a wave form in the audio client. A detailed research has been undertaken but no solution has been found. This can be changed as soon as the WebKit engine supports the corresponding events.

Since the Google Web Toolkit does not feature slider bars in its latest version, both clients have no volume slider. This feature can be added as soon as Google improves its development tool.

In addition to those tasks, the following list presents proposals for further research:

Currently the audio and video client has been designed with Adobe's Photoshop. However, all Web browsers that are capable of HTML5 also support CSS3. With help of this technology, the clients can be designed with several HTML div tags and CSS only. This way smaller changes do not need to be realized through Photoshop but can be applied through CSS file changes. Developers without Photoshop maybe would like to add or change the appearance of the clients. This feature may also be researched and added.

Most of the modern smart phones support HTML5 and CSS3, which opens the possibility of supporting those. Just because those devices have different screen sizes a method called responsive Web design [31] can be applied to the clients. This approach works with CSS media queries and allows the adaptation of all kinds of screen sizes.

A Summary

Currently there is an ongoing trend to publish more and more multimedia on the Web. A range of solutions exists that allow end-users to organize content and to add additional information by means of annotations. Until now, however, there exists no open standard-based solution which allows users to annotate and link audio and video to contextually relevant information from other sources on the Web.

Foundation for this thesis constitutes the LEMO framework deployed by Europeana. This solution enables fragment identification and therefore introduces annotating and linking of continuous media. Due to the linkage of contextually relevant resources, the overall documentation can significantly be improved.

New developments in the research area of Linked Data enable automatic access to a huge amount of well defined public data.

The goal of this thesis is to determine whether a combination of new standards enables developing a Web-based audio and video application, that is usable, considers the four rules of Linked Data and that is standard-based.

The concluding statement of this master thesis is that such an application can be built. The result is based on two case studies; a qualitative research study which revealed usability issues - subsequently eliminated - and a case study, which in turn showed that a semi-automatic approach for generating and retrieving tags would prove to be efficient when annotating and linking resources.

B Zusammenfassung

Gegenwärtig herrscht ein Trend immer mehr Multimedia im Web zu veröffentlichen. Es existiert bereits eine grosse Anzahl an Applikationen, die es EndbenutzerInnen ermöglicht Inhalte zu organisieren und darüber hinaus noch zusätzliche Anmerkungen hinzuzufügen. Bis jetzt existiert aber keine EndbenutzerInnen freundliche Applikation, die es BenutzerInnen ermöglicht Audio und Video Dateien mit relevanten Informationen aus offenen Web-Quellen zu annotieren, zu verknüpfen und die auch auf offenen Standards basiert.

Grundlage für diese Arbeit bietet das von Europeana eingesetzte LEMO Framework. Dieses Framework macht Fragment Identifikation nutzbar und ermöglicht damit Annotierungen und Verlinkungen von kontinuierlichen Medien. Durch die Verlinkung von kontextuell relevanten Ressourcen, kann die Dokumentation der Medien erheblich verbessert werden.

Neue Entwicklungen im Forschungsgebiet von Linked Data ermöglichen es, automatisiert auf eine Menge öffentlich zugänglicher und wohl definierter Daten zuzugreifen.

Ziel der vorliegenden Arbeit ist es fest zu stellen, ob durch eine Kombination von neuen Standards eine Implementierung einer Web-basierten Audio und Video Anwendung möglich ist, die usable ist, den Zielen von Linked Data entspricht und auf offenen Standards basiert.

Abschliessend lässt sich festhalten, dass eine solche Anwendung entwickelt werden kann. Das Ergebnis basiert auf zwei durchgeführten Studien; eine qualitativ durchgeführte Studie, welche usability Probleme aufzeigte - die im Anschluss eliminiert wurden - und eine weitere durchgeführte Studie, welche wiederum aufzeigte, dass ein semi-automatischer Ansatz zur Generierung von Tags die effizienteste und effektivste Möglichkeit zur Annotierung und Verlinkung darstellt.

C DVD attachment

Attached to this master thesis, readers can find a DVD for installing and running this project. The attachment contains all the necessary files to debug and deploy the server and both clients. In general, the DVD contains a detailed guide for the installation, two WAR files for deploying the project, the source code files of the audio and video client, the source code of the annotation server as well as the documentation of the compiling code.

With the provided installation guide, it is possible to install the annotation server and the audio and video client of this master thesis. The DVD contains two WAR files for the server and the clients. Both contain all the necessary classes, binaries, pictures and media to get the whole project running. The guide lists necessary applications and links for the download. Additionally, it gives detailed instructions in order to get the demo running on Windows and Mac OS X computers. The setup and installation sequence is straight forward and does not require certain skills. However, it is advantageous to have fundamental programming and database knowledge.

Additionally, it is possible to change, extend and improve this project, since this DVD contains two projects, with compile able files. Those two Eclipse projects allow users to modify the audio and video client as well as the server. Users find a detailed documentation attached to the essential Java classes. By importing those projects into Eclipse, debug and deploy executions are possible. However, users need to install the GWT plugin for eclipse.

D Scientific curriculum vitae

Elementary school
Grillparzer Volksschule, St. Pölten

Secondary school
BRG St. Pölten, Josefstrasse

Upper school
3rd September 1997 - 24th June 2002
HTL und VA St. Pölten - Automation Engineering

Civilian Service
1st October 2002 - 30th September 2003
Seniorenwohnheim Stadtwald, St. Pölten

Studies
1st October 2003 - 28th February 2009
Bachelor studies at the University of Vienna and the Technical University of Vienna in business informatics

25th January 2006
Submission of the 1st bachelor thesis with the title ”Prozessorientierte Zeit- und Personenkoordination”

23rd August 2006
Submission of the 2nd bachelor thesis with the title ”Yellow Pages Search Server”

1st March 2009 - 11th April 2012
Master studies at the University of Vienna and the Technical University of Vienna in media and informatics

11th April 2012
Submission of the master thesis with the title ”HTML5 audio and video annotations with resource detection under consideration of usability and user experience”

References

[1] B. Haslhofer, E. Momeni, M. Gay, R. Simon. Augmenting Europeana Content with Linked Data Resources, presented at I-SEMANTICS, Graz, Austria, (2010)

[2] R. Sanderson, H. v.d. Sompel, Open Annotation: Alpha3 Data Model Guide, (2010), [Online]. Available: http://www.openannotation.org/spec/alpha3/

[3] W. Klas, Multimediale Systeme 1, University of Vienna, Vienna, (2010), pp. 4 - 18

[4] J. Kahan, M.R. Koivunen, E. Prud'Hommeaux, R. R. Swick, Annotea: An Open RDF Infrastructure for Shared Web Annotations, presented at WWW10, Hong Kong, (2001), [Online]. Available: http://www.w3.org/2001/Annotea/Papers/www10/annotea-www10.html

[5] M.R. Koivunen, Annotea and Semantic Web Supported Collaboration, presented at UserSWeb 2005, [Online]. Available: http://annotea.org/eswc2005/01_koivunen_final.pdf

[6] B. Haslhofer, W. Jochum, R. King, C. Sadilek, K. Scheller, The LEMO Annotation Framework, (2009), pp. 5 - 13

[7] R. T. Fielding, Representational State Transfer (REST), (2000), [Online]. Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[8] R. Schroeter, J. Hunter, D. Kosovic, Vannotea, (2003), [Online]. Available: http://itee.uq.edu.au/ eresearch/papers/2003/schroeter-kcap03.pdf

[9] T. Berners-Lee, Linked Data - Design Issues, (2006), [Online]. Available: http://www.w3.org/DesignIssues/LinkedData.html

[10] J. Volz, C. Bizer, M. Gaedke, G. Kobilarov, Silk - A Link Discovery Framework for the Web of Data, presented at LDOW, Madrid Spain, (2009)

[11] M.G. Butuc, Semantically Enriching Content Using OpenCalais, (2009), [Online]. Available: http://www.eed.usv.ro/SistemeDistribuite/2009/Butuc1.pdf

[12] J. Keith, HTML5 for Web Designers, New York: A Book Apart, (2010)

[13] A. Budd, CSS Mastery - Advances Web Standards Solutions, Springer Verlag New York, (2006)

[14] D. Cederholm, CSS3 for Web designers, New York: A Book Apart, (2010)

[15] R. Hanson, Introduction to GWT, (2009). [Online]. Available: http://gwtsandbox.com/ete2009/intro-to-gwt.pdf

[16] D. Spinellis, It's About Time to Take JavaScript (More) Seriously, IEEE Computer Society, (2010), pp. 60 - 62

[17] W. Jochum, Requirements of Fragment Identification, University of Vienna, Vienna, (2007), [Online]. Available: http://eprints.cs.univie.ac.at/396/1/Identification.pdf

[18] J. Nielsen, Usability Engineering, London: Academic Press, (1993)

[19] The Humane Interface: New Directions for Designing Interactive Systems, J. Raskin, Addison-Wesley Professional, (2000), pp. 2

[20] J. Tidwell, Designing Interfaces, O'Reilly Media, (2010)

[21] M.J. Davidson, L. Dove, J. Weltz, Mental Models and Usability, (1999), [Online]. Available: http://www.lauradove.info/reports/mental%20models.htm

[22] M. Ward, A. Charchar, F. Inchauste, M. Rundle, J. Jovanovic, C. Heilmann, V. Anayian, C. Kolb, S. Weinschenk, S. Bradley, The Smashing Book #2, Freiburg: Smashing Media GmbH, (2011), pp. 254 - 272

[23] G. A. Miller, The magical Number seven, plus or minus two, American Psychological Association, (1995)

[24] D.M. Jones, The 7+-2 Urban Legend, presented at MISRA C Conference, (2002), [Online]. Available: http://www.knosof.co.uk/cbook/misart.pdf

[25] L. Mathis, Design for use, The Pragmatic Bookshelf LLC, (2011)

[26] Fitts' Law Group, Fitts' Law, (1996), [Online]. Available: http://ei.cs.vt.edu/ cs5724/g1/

[27] M. W. van Someren, Y. F. Barnard, J. A.C. Sandberg, The thinking aloud method, Academic Press, London, (1994)

[28] A. Cattaneo, A. Maier, C. Spooner, D. A. Monsef, D. Leggett, D. Fadeyev, J. Gube, K. Knight, R. Schmidt, S. Snell, J. Tan, The Smashing Book, Smashing Media GmbH, Luebeck, (2009), pp. 64 - 73,

[29] P. N. Mendes, M. Jakob, A. Garcia-Silva, C. Bizer, DBpedia Spotlight: Shedding Light on the Web of Documents, presented at I-SEMANTICS, Graz, Austria, Sept. (2011), [Online]. Available: http://www.wiwiss.fu-berlin.de/en/institute/pwo/bizer/research/publications/Mendes-Jakob-GarciaSilva-Bizer-DBpediaSpotlight-ISEM2011.pdf

[30] W. Klas, B. Haslhofer, Multimedia Retrieval, University of Vienna, Vienna, (2010), pp. 277 - 280

[31] E. Marcotte, Responsive Web Design, New York: A Book Apart, (2011)