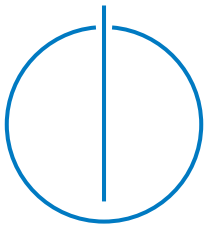
Fakultät für Informatik
Der Technischen Universität München

Master's Thesis in Information Systems

Implementing a Web Client for Social Content and Task Management

Björn Michelsen



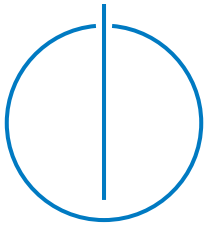
Fakultät für Informatik
Der Technischen Universität München

Master's Thesis in Information Systems

Implementing a Web Client for Social Content and Task Management

Implementierung eines Web Clients für soziales Content- und Aufgabenmanagement

Author: Björn Michelsen
Supervisor: Prof. Dr. Florian Matthes
Advisor: Felix Michel
Date: September 15, 2016



I assure the single handed composition of this master's thesis only supported by declared resources.

Munich, September 15, 2016 Björn Michelsen

Acknowledgments

I would like thank my advisor Felix Michel for his support. Furthermore, I would like to thank Thomas Reschenhofer and the SocioCortex team for the collaboration regarding the SocioCortex back-end and API.

Abstract

Complex, knowledge-intensive processes are becoming increasingly important for modern enterprises. The support of such processes through information systems yields a huge potential for their increased efficiency and effectiveness. SocioCortex, the social information hub, is one approach for such a collaborative information system. It builds upon the concept of hybrid wikis, an end-user-oriented way to create, store, and distribute knowledge in both unstructured and structured ways. It also aims to include task centered collaboration aspects. The idea behind SocioCortex is to provide a back-end as a basis for dedicated clients, for example for generic content and task management, modeling, or visualization. The objective of this thesis is the implementation of a web client for one of those use cases, the generic content and task management. The approach of this thesis is based on a review of existing work regarding hybrid wikis, task centered collaboration approaches and initial design and implementation examples of the content manager. Furthermore, the implemented result are evaluated through tests with enterprise stakeholders familiar with Enterprise Architecture Management (EAM).

Contents

Outline of the Thesis

Part I: Introduction

Chapter 1: Introduction

Gives a short introduction about the context and motivation behind this work. It introduces the idea behind SocioCortex and its relation to the web client developed in the course of this thesis. Furthermore, the chapter will outline the approach.

Part II: Related Work

Chapter 2: Related Work

The chapter provides insights in the related work of this thesis. It includes and overview of the hybrid wiki approach and work that forms the foundation of this thesis.

Part III: Implementation

Chapter 3: Use Cases

Lists and details the use cases that the implemented web client supports. The described use cases are grouped together by major components or system aspects they relate to.

Chapter 4: Design Challenges

Discusses design challenges that were identified during the development of the content manager. The challenges are discussed on a use case level and a general application level.

Part I.

Introduction

1. Introduction

1.1. Introduction

With the rapid growth of information available in enterprises, it becomes increasingly difficult for them to manage the use of this information effectively and efficiently [GBD09]. In the past, various tools from the area of information and communication technology (ICT) have been used in order to aid enterprise stakeholders in handling this information, especially in support of knowledge-intensive processes. One of such approaches is the wiki, a web-based tool to capture, share, and organize information collaboratively [LC01].

However, a downside of a wiki is that is does not support certain use cases and query operations that require more structure. Thus, an iteration on the basic wiki concept is the semantic wiki, which adds structured data to otherwise unstructured wiki pages [MNS11]. Core aspects of semantic wikis are that they allow users to add properties to pages and organize the pages themselves in ontologies using classes [Br12].

[MNS11] proposes the hybrid wiki as a further improvement to the semantic wikis, stating a complicated syntax, unfamiliar concepts, and a missing clear communication of the benefits of structured data as key drawbacks of semantic wikis. The hybrid wiki aims to mitigate those issues and provide a lightweight way to manage data and information in enterprise. Core concepts used in the hybrid wikis are attributes for capturing structured data and type tags, allowing users to express a class-instance relationship between wiki pages and explicit types.

[HKM15] proposes an alternative wiki approach, focusing on a process-oriented view and extending the wiki with structural elements such as attributes, tasks,

and types. The goal of this approach is to provide end-users with a lightweight way to model processes. The approach was implemented as the Darwin Wiki.

1.2. Motivation

Two of the in Section 1.1 proposed extensions of the semantic wiki, the hybrid wiki and approach of the process-oriented Darwin Wiki, are combined into a single system called SocioCortex, the social information hub [SE14].

One of the main ideas behind SocioCortex Platform is to have a core application, the SocioCortex Backend, which provides a REST-API. This REST-API is used by different clients to provide access to SocioCortex for different use cases and end-users. This approach differs from previous monolithic applications.[1]

Due to their specificity, the individual clients can help to provide only the required functionality for a use case, eliminating unnecessary complexity and increasing the usability. This is a fundamentally different approach as opposed to the main predecessors of SocioCortex.

1.3. Approach

The problem at hand is the development of one specific client, the SocioCortex Content Manager. As described, the client consumes the REST-API provided by SocioCortex Backend to support specific use cases. In terms of the content manager, the covered use cases concern mostly the management of data, especially in the form of manipulating data in concepts such as entities, attributes, tasks, and workspaces.

[1] `http://www.sociocortex.com/`

The scope of this thesis includes the implementation of the SocioCortex Content Manager for a set of documented use cases, given certain constraints regarding the design and technologies. The content manager will combine aspects of the hybrid wiki and Darwin Wiki. A part of the approach is also the evaluation of the implemented system by conducting tests with participants familiar with enterprise architecture management (EAM).

Part II.

Related Work

2. Related Work

This chapter outlines the related work of this thesis. It includes the the general wiki concept, semantic wikis, and hybrid wikis. Furthermore, it describes the related work done in the context of SocioCortex Platform, including its clients.

2.1. Wikis

A wiki is a type of web-based software, that allows users to collaboratively edit information displayed as a web page in a web browser. A page in a wiki can contain text and hypertext, linking to other wiki pages [LC01].

MediaWiki [Meda] as shown in Figure 2.1, is a specific implementation of such a wiki system. It is a popular software used in large-scale project such as Wikipedia [Wik]. MediaWiki supports many of the basic functionality expected from a wiki such as the editing of pages, linking between pages, searching for pages and many more.[2]

2.2. Semantic Wikis

Semantic wiki enhance the traditional wiki concept presented in Section 2.1 by adding structured elements and processing capabilities [Br12]. Structured elements are especially represented by classes and properties. Wiki pages themselves can be an instance of a class and contain properties, which are key-value pairs of data that might be restricted by type. Using queries, the

[2] `https://www.mediawiki.org/wiki/Manual:MediaWiki_feature_list`

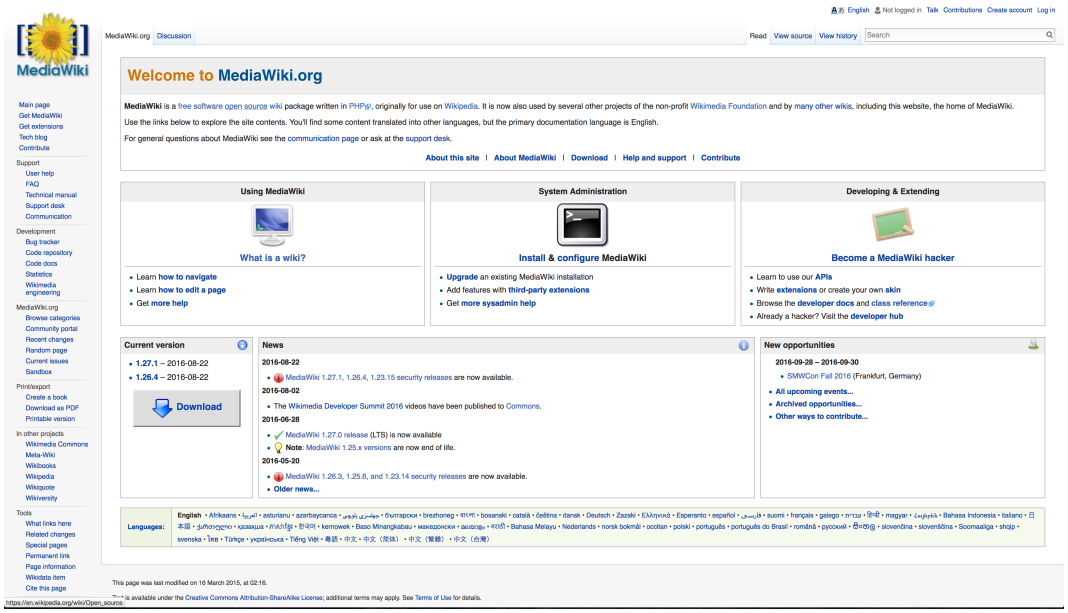


Figure 2.1.: MediaWiki Homepage running on MediaWiki software [Meda]

structured data captured in a semantic wiki can be used for reasoning or inference [Gi13].

Just as for the traditional wiki described in Section 2.1, several implementations of semantic wikis exist, some of which [Br12] evaluate based on various parameters.

A popular semantic wiki is the semantic extension for MediaWiki, called Semantic MediaWiki [Sem]. It extends MediaWiki with semantic elements such as types, properties, templates, and querying capabilities.[3].

2.3. Hybrid Wiki

The hybrid wiki is an enhanced concept of the semantic wiki proposed by [MNS11]. It aims to improve shortcoming of the semantic wiki such as a

[3] `https://www.semantic-mediawiki.org/wiki/Help:User_manual`

complicated syntax, unfamiliar concept, and non-obvious benefits of the additional structure. The hybrid wiki aims to mitigate those issues and provide a lightweight way to manage data and information in enterprise. Core concepts used in the hybrid wikis are attributes for capturing structured data and type tags, allowing users to express a class-instance relationship between wiki pages and explicit types [MNS11].

[Re16] iterated upon the hybrid wiki concept and enhanced the meta model as described in Figure 2.2.

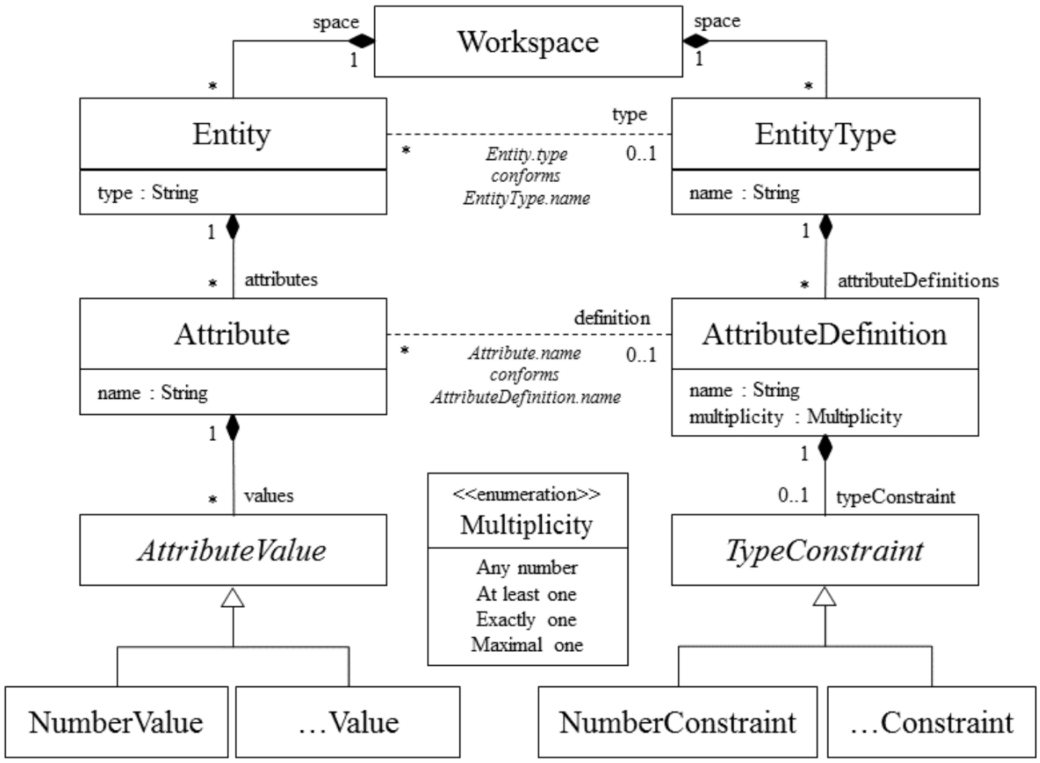


Figure 2.2.: Hybrid wiki meta-model [Re16]

The meta-model defines concepts that relate to instance on the left side and to the model on the right side. Workspaces are the only concept which cannot be clearly assigned to either side as a workspace is related to both. A workspace is a high-level unit of organization and contains entities and entity types. Entities

are a core concept in a hybrid wiki, containing the structured and unstructured information in a hybrid wiki. Entities are instances of entity type, which are defined the model [Re16].

Attributes are key-value pairs that allow entities to store structured data. Every entity has to be part of an entity. Additionally, attributes are defined by the attribute definition, setting constraints of the attribute name and its multiplicity [Re16].

2.4. Darwin

Darwin is a wiki proposed by [HKM15]. It focuses on allowing non-expert users to structure knowledge-intensive processes in a lightweight and user-friendly way. In order to achieve this, it offers several structural elements as opposed to a traditional wiki as described in Section 2.1. The elements it provides include attributes, tasks, and types. Tasks are the most relevant difference from a model perspective as it is an uncommon concept in semantic wikis. The idea behind tasks is that a task can be related to certain attributes. The goal of a task is to fill out the value of the related attributes after which a task is seen as complete [HKM15]. Figure 2.3 shows the user interface of the Darwin system that showcases the structural element with their visualization.

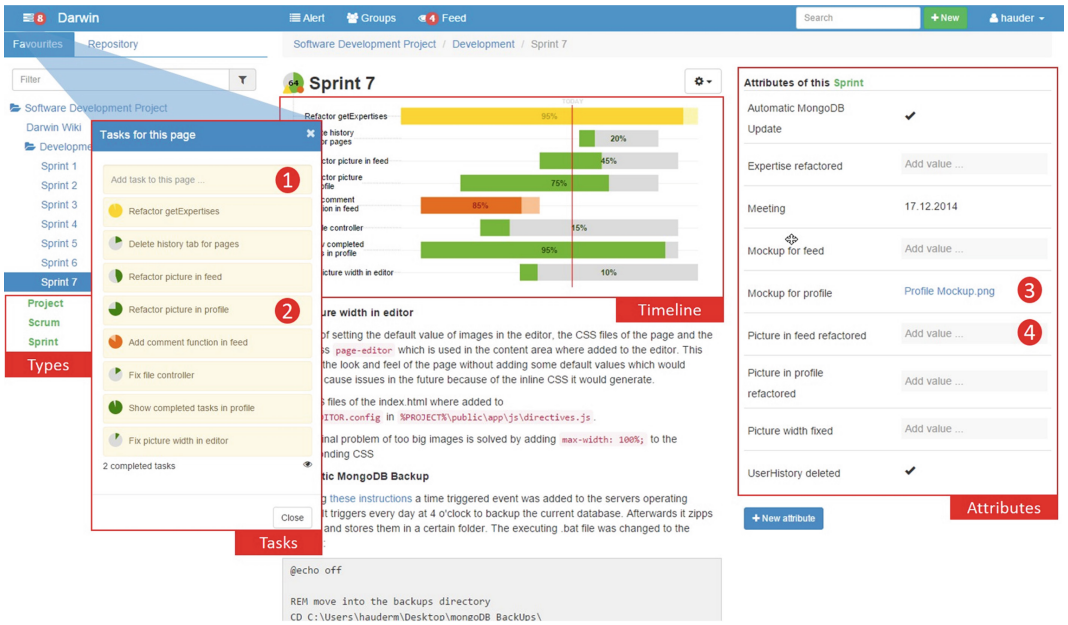


Figure 2.3.: The Darwin Wiki user interface showcasing structural wiki elements and their visualization [HKM15]

2.5. SocioCortex

SocioCortex is a platform to support knowledge-intensive processes. It builds upon the hybrid wiki and the task-oriented Darwin [SE14]. As opposed to its predecessors with a monolithic architecture, the SocioCortex Backend provides a REST-API which different types of clients can leverage.

2.5.1. Conceptual Model of SocioCortex

Figure 2.4 gives an overview about the most important concepts in SocioCortex. On a high level, concepts can be categorized by whether they are part of the model or instance. The following will give a short description of individual concepts and their relation to each other:

- *Workspace*: A workspace is a high-level organizational unit that contains all instance and model data. There can be several workspaces, each containing their individual set of data, including entities, attributes, tasks, entity types, attribute definitions, task definitions, and stages.
- *Entity*: An entity contains unstructured and structured data. The structured data is represented as attributes and tasks.
- *Attribute*: An attribute contains structured data and is part of an entity. Additionally, an attribute can be assigned to a task.
- *Task*: Tasks are a core concept to provide a structured workflow in filling out attributes [HKM15]. Tasks can be contained in entities and in turn have attributes.
- *EntityType*: Each entity is an instance of an entity type. An entity type is like an blueprint for an entity for defining attribute definitions and task definitions.
- *AttributeDefinition*: Like the relationship between an entity and entity type, an attribute definition is like an blueprint for an attribute.
- *TaskDefinition*: Like the relationship between an entity and entity type, an task definition is like an blueprint for a task.
- *Stage*: Task definitions can be assigned to stages, making it possible to structure processes, a concept from Darwin.

Section 3 will elaborate many of the concepts as it describes the system behavior in of the proposed SocioCortex Content Manager in detail.

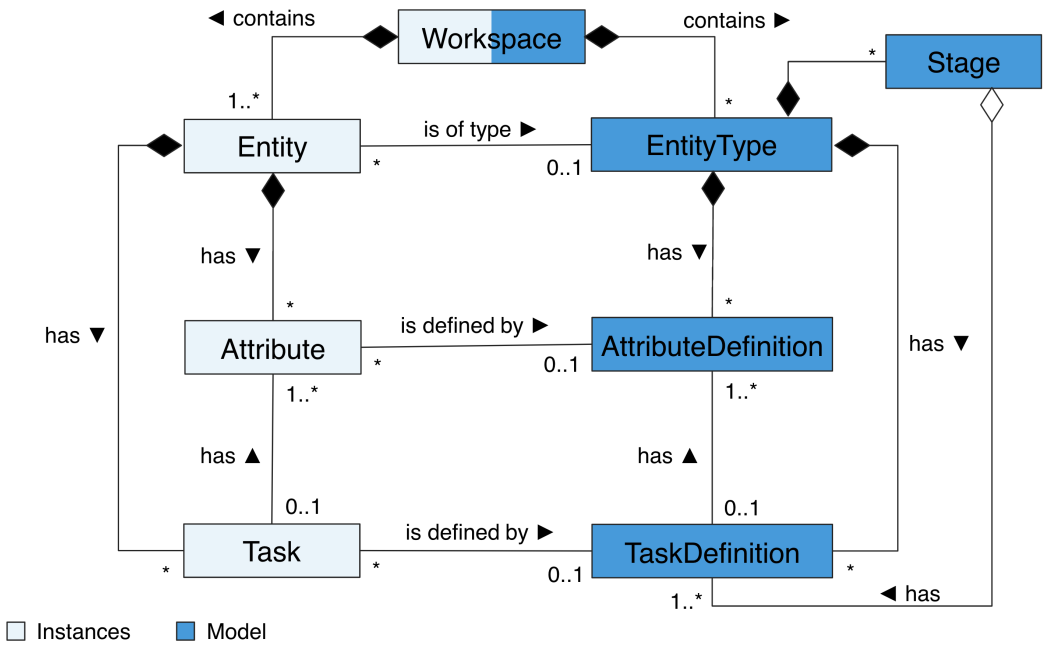


Figure 2.4.: The conceptual model of SocioCortex[4]

2.5.2. Clients for SocioCortex

In this section the three SocioCortex clients from the default client suite are presented. They are all currently in active development.

2.5.2.1. SocioCortex Content Manager

The SocioCortex Content Manager is the topic of this thesis and provides a generic interface to manipulate the data of the SocioCortex Backend. This client is not concerned with the modeling of data, such as the creation of entity types. It supports the editing of data such as CRUD operations on workspaces, entities, attributes, and tasks. It is part of the default client suite in SocioCortex as depicted in Figure 2.5.

[4]http://www.sociocortex.com/
[5]http://www.sociocortex.com/

Figure 2.5.: SocioCortex architecture with content manager[5]

2.5.2.2. SocioCortex Modeler

The SocioCortex Modeler as shown in Figure 2.6 is another client. It is concerned with the data modeling perspective of SocioCortex. The client supports tasks such as the creation an management of entity types, attribute definitions, task definitions, and derived attributes. [Sc16]

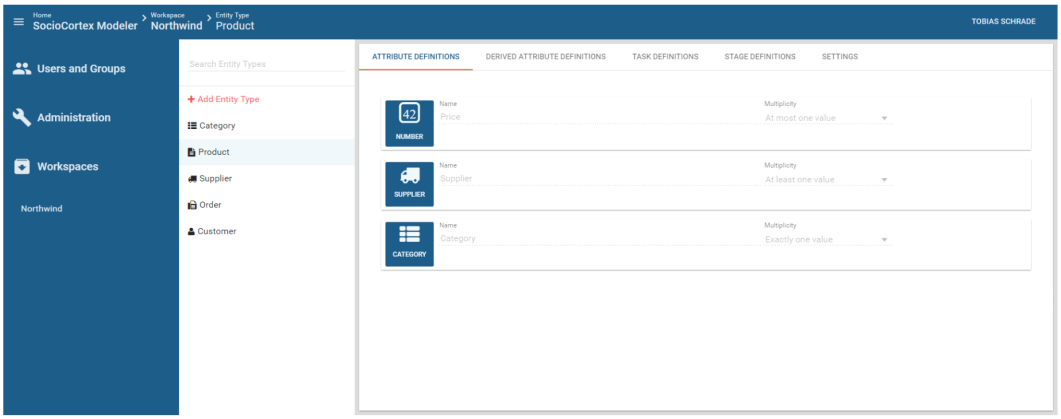


Figure 2.6.: SocioCortex Modeler[Sc16]

2.5.2.3. SocioCortex Visualizer

The SocioCortex Visualizer as shown in Figure 2.7 is a client dedicated to the visualization of data stored in SocioCortex. It is based on a modular dashboard architecture, enabling end-users to quickly analyze data. [Bü15]

2.5.3. Previous Work used as the Foundation of this Thesis

Work done in the context of SocioCortex provided direct input for the development of the content manager. First of all, the bachelor thesis [Ka15] provided the general design guidelines and specific mockups, as shown in Figure 2.8, for many parts of the web interface. Additionally, the initial implementation in the context of the master thesis [Os15] provided the a first implementation of the client with a focus on the task management experience. A student project in the context of the SEBIS chair provided the implementation of the activity feed functionality. Moreover, an abstraction of the SocioCortex REST API called SC-Angular [SCA] provided the basis for the communication of the SocioCortext Content Manager with the SocioCortex Backend.

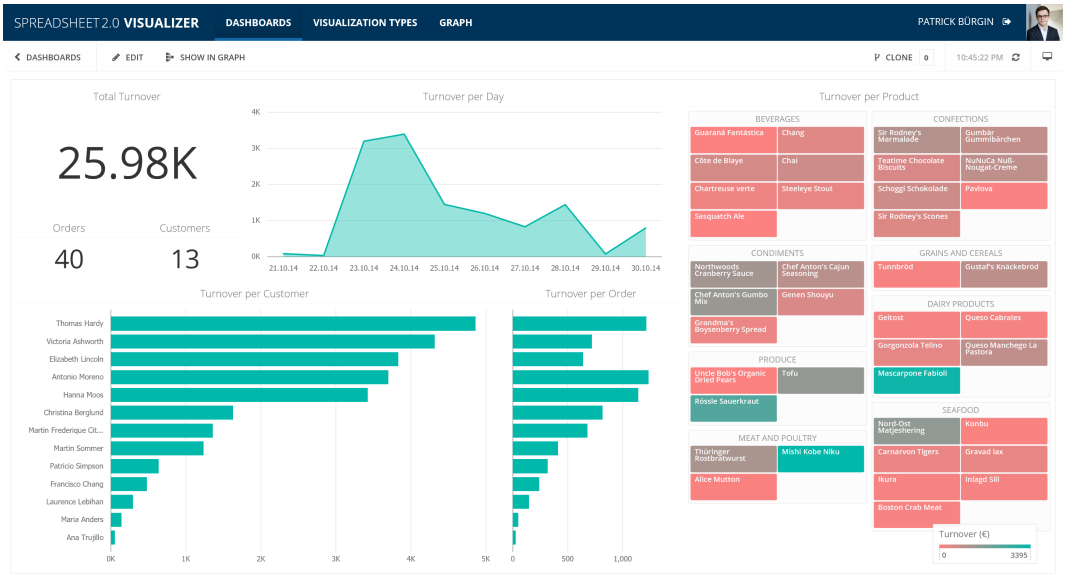


Figure 2.7.: SocioCortex Visualizer[Bü15]

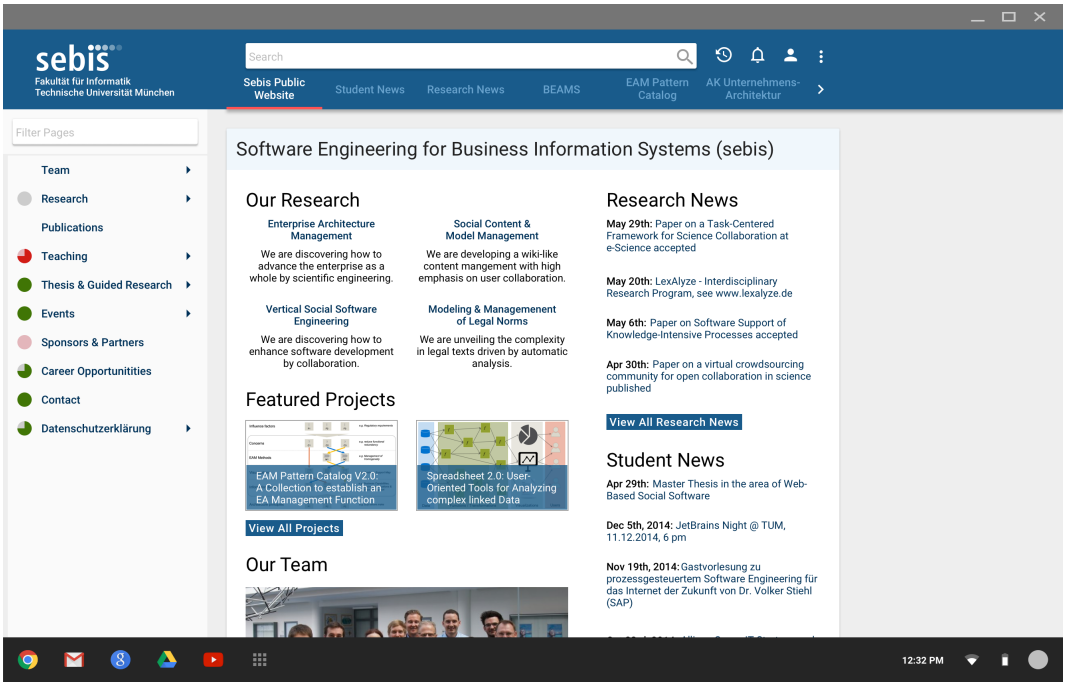


Figure 2.8.: Example Mockup for the SocioCortex Content Manager [Ka15]

Part III.

Approach

3. Use Cases

This chapters gives an overview of the functionality the implemented content manager should provide. This is achieved with descriptions of the systems' use cases. The use cases are grouped by major components or aspects they relate to. Some of them have been described in [Os15] and [Ka15], both of which were used as the main input for this thesis. However, use cases have been added, extended or changed in the process of this work. They are documented to provide an complete overview of the capabilities of the implemented content manager.

3.1. User Authentication

3.1.1. Logging in

Users have the ability to log into the content manager in order to get access to restricted information in the system. If a users wants to log in, he has to provide an e-mail address and password in order to authenticate.

3.1.2. Logging out

If a user wishes to give up the privileges to view restricted information, he can log out of a session and undo a previous authentication.

3.2. Workspaces

Workspaces provide a mechanism to organize information. They act as a high-level container for entities and aid in the retrieval of information. Workspaces behave like folders, meaning entities and entity related models such as attributes, tasks, and files can only be in one workspace at the same time.

3.2.1. Creating a workspace

A precondition for a user being able to create new workspace is to be authenticated and having the appropriate permissions. Then, a user can choose to create a new workspace, having to set the workspace name as a required argument. The creating of a workspace automatically creates an entity in that workspace that acts as the homepage of that workspace.

3.2.2. Deleting a workspace

If a workspace and all contained information such as entities and their data captured in attributes, tasks, files, and unstructured content is no longer needed, it can be deleted. A user with the appropriate permissions can delete a workspace and all its containing information. This operation is destructive and cannot be undone.

3.2.3. Renaming a workspace

Each workspace is required to have have a name. This name can be changed by an user with the appropriate permissions. In order to complete a rename operation, the user has to provide a new name. This name is not allowed to be empty.

3.2.4. Adding a workspace to favorite workspaces

Workspace can contain meta information whether the workspace is a favorite workspace or not. The user is provided with specific ways that allows them to navigate the set of favorite workspaces more easily than others.

3.2.5. Removing a workspace from favorite workspaces

Analogous to adding a workspace to a list of favorite workspaces, a user can remove a workspace from this list if its favorite status is no longer required. This operation can by undone by adding the workspace to a favorite workspace again.

3.2.6. Navigating between workspaces

A user can navigate between workspaces. At all times, only a single workspace can be selected. If a workspace is selected, the user is only presented with the information contained in this workspace. The precondition for navigating to a specific workspace is to have at least read permissions of the workspace.

3.2.7. Editing workspace settings

Workspace settings provide users with a mechanism to change the permissions for the entire workspace. The permissions include the ability to read and write data in the specific workspace.

3.3. Entities

Entities allow users to capture and store structured, semi-structured, and unstructured data. The main way to handle structured and semi-structured data is using attributes of entities. The various types and use cases for attributes are described in detail in Section 3.4 . Rich-text entity content provides the

main approach to handle unstructured data in entities. An entity can also be called a *page* or *wiki page*, a term commonly used in other wiki systems such as MediaWiki [Medb]. The terminology has been changed in the revision of the hybrid wiki meta-model in [Re16].

3.3.1. Creating an entity

A precondition for creating an entity are the appropriate permissions by the user who wants to create the entity. If the correct permissions exist, the user can create a new entity and has to set the name of the entity as a required information. Optionally, the entity type can be set in this step. If the user does not set a specific entity type, the entity will be created with the default entity type *Text Page*. If an existing entity is currently selected, the newly created entity will be a subpage of the selected entity.

3.3.2. Deleting an entity

A user can delete an entity, if the information contained in the entity is no longer required. Deleting an entity deletes all content, attributes, tasks, files, and other associated information. Additionally, all subpages of that entity including all their related data such as content, attributes, tasks, files and other information are deleted too. This is a destructive operation and cannot be undone.

3.3.3. Renaming an entity

The entity name, which has to be set at least initially during creation of an entity, can be edited. To edit an entity name the user has to provide a new valid name for the entity. A valid entity name is any non-empty string. The renaming of an entity does not influence the behavior of references to that entity. For example, if the entity content or attribute of an entity contains a reference to another entity, this reference is still valid after a rename operation.

This reduces the amount of manual work necessary by a user to rename a page in comparison to traditional wiki systems, as individual references do not need to be checked and updated manually.

3.3.4. Duplicating an entity

Any entity can be duplicated. In order to duplicate an entity, the user is required to set a name and parent for the duplicated entity. The system provides the user with default values for the name and parent of the duplicate. This action only duplicates the entity selected for duplication without any child entities.

3.3.5. Moving an entity

Moving an entity allows users to set a new parent for an entity. The new parent is the only attribute required for this operation. This operation moves an entity with all its associated data and child entities.

3.3.6. Editing the content of an entity

The content of an entity plays a central role in capturing and storing unstructured content. Users can capture information in rich-text, meaning they can use a variety of elements and formatting options in order to style and structure the content. The precondition for editing entity content are adequate write permissions on that entity. The most basic element in the content is the text block, which can be inserted, edited and removed. Additionally they can be styled with inline formatting options such as bold, italics, and underline. Another element is the image, which can be inserted by the user. The image can either be uploaded from a locally stored file, a reference of an existing image file, or an externally hosted image file. Furthermore, links can be used to either created references between entities or to an external resource such as a website.

3.3.7. Managing files of an entity

Files of any type can be attached to an entity. A user can upload a one or more locally stored files to an entity. After the files are successfully uploaded, any user with at least read permissions can view all the files attached to the entity and download the files individually. Additionally, files that are not required anymore can be deleted from an entity. As mentioned in the use case in Section 3.3.2, multiple files can be deleted at once if the whole entity is being deleted.

3.3.8. Editing entity settings

Entity settings provide the user with a way to change permissions and other aspects of a specific entity. The permissions include the ability to read and write data. Permissions are inherited to child entities.

3.4. Attributes

Attributes provide a way to capture structured data. There are three classes of attributes: Free attributes, typed attributes, and derived attributes.

The first class of attributes are free attributes. They can be defined in the content manager and are not defined in the type of an entity. The idea behind free attributes is to provide users with the flexibility of capturing structured data without the need to first define it in entity types, thus reducing overhead.

The second class of attributes are typed attributes and can only be defined in the SocioCortex Modeler. However, the value of typed attributes can be edited and is part of the use cases.

The third class of attributes are derived attributes. They are automatically calculate using MxL expressions. Their existence and corresponding MxL expression can only be defined in the SocioCortex Modeler.

3.4.1. Editing an attribute

The value of attributes can be edited. The editing interface adapts to the type of the attribute. For example, an attribute of the type *Date* shows the user a date specific interface as opposed to an attribute of the type *Text*.

3.4.2. Creating a free attribute

The content manager allows users to create free attributes. They are not defined in the model of an entity using the SocioCortex Modeler but by using the content manager itself. Free attributes allow users to the capture of structured data which later might be included into the model as a typed attribute.

3.4.3. Editing a free attribute

Free attributes do not contain information about the type of information they will hold. As such, a user can enter information in an unrestricted way. In the process of entering such information, the content manager is trying to 'guess' the implicit type of the information and support the user appropriately.

3.4.4. Deleting a free attribute

A user can delete a free attribute, if the structured information it represents is no longer required. Deleting a free attribute leads to the deletion of the attribute value as well as the attribute itself. This is a destructive operation and cannot be undone.

3.5. Tasks

Tasks provide a way to structure a workflow. The main concept behind tasks is that they are related to attributes in such a way, that they define whether an attribute or a set of attributes contains information as described by a task.

For example, a workspace might contain an entity called *Master thesis* with a typed attribute *Abstract* and a task *Define abstract*. If the attribute *Abstract* is assigned to the task *Define abstract*, there is a specific relationship between both. An example how this relationship works is that if the attribute *Abstract* contains a value, the task *Define abstract* is automatically completed.

3.5.1. Creating new tasks on an entity

Tasks can either be created using the SocioCortex Modeler or using the SocioCortex Content Manager. A task created as a task definition in the SocioCortex Modeler can be viewed and edited by the SocioCortex Content Manager.

3.5.2. Editing a task

The metadata of a task can be edited. This includes progress of a task expressed in a percent value, a start date and end data expressed as date values which can be edited with a date picker, an owner and expertise of a task. The task progress is automatically calculated if the task contains attributes. For example, if a task contains two attributes, and only one of them contains a value, the task progress is set to 50 percent. The progress can also be manually overridden by a user.

3.5.3. Deleting a task

A task that is defined by a task definition cannot be delete in the content manager. However, if this task if not required, it can be skipped as described in Section 3.5.5. If a task is defined in the content manager, it can be delete by users with access and permissions for that entity.

3.5.4. Completing a task

A task can be completed and there are several way to achieve this. First of all, if a task contains task attributes, it is automatically completed if all attributes contain a value, for example because a user has set the value of the attributes. Additionally, a user can complete a task by using the *Complete* action in the content manager. The last approach to complete a task is to manually set the metadata *progress* of a task to 100 percent.

3.5.5. Skipping a task

A user can skip a task. This action can be manually triggered in the content manager and leads to a state in which the task has not to be completed using any of the approaches as described in the previous section, Section 3.5.4 . The skipped status of the task is recorded by the system.

3.5.6. Assigning an attribute to a task

Any attribute of an entity can be assigned to a task. The connection between the task and an attribute leads to the progress status of a task being connected to the value stored in the attribute. For example, if a task *Define thesis* has two attributes *Define start date* and *Define end data*, the progress status of the task *Define thesis* will be 50 percent, if one of the two attributes contains a date.

3.5.7. Removing an attribute from a task

If a task has one or more attributes assigned to it, those attributes can be removed. Removing an attribute from a task revokes any effect of the relationship as described in Section 3.5.6.

3.6. User Profile

The user profile provides basic information about the currently logged in user. First of all, the user profile displays the profile picture of the user. Additionally, metadata such as email, the full name of the user, expertise, and tasks are presented. The representation of tasks is split into current, future, and completed tasks.

3.7. Data Tables

Data tables provide tabular, spreadsheet-like overview of entities. For navigation purposes, it is possible to sort, filter, and search entities based on their metadata such as the entity name and the values of their attributes. Additionally, it is possible to edit the values of entities directly in the data tables, without having to navigate to the individual entities first. This makes it easy to edit the attributes of several entities without loosing context. Figure 3.1 shows a first implementation of data tables. The development of them is outside the scope of this thesis. However, they will be implemented by Daniel Elsner and subsequently integrated into the content manager.

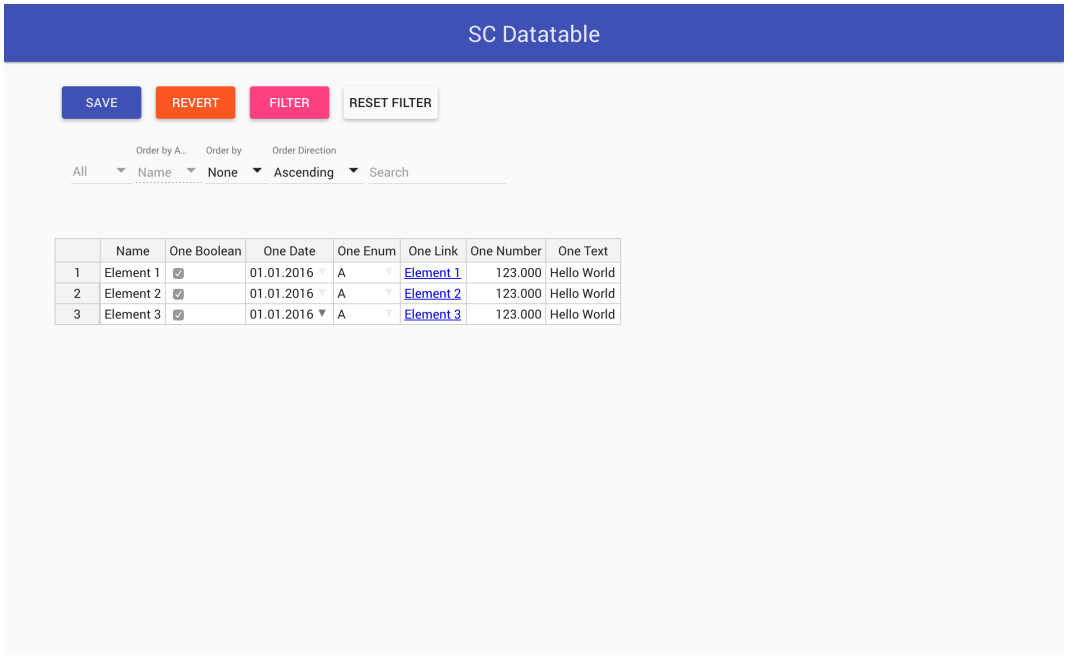


Figure 3.1.: The current implementation of data tables

4. Design Challenges

This chapter documents the design process and challenges. The content is separated by use case specific and general system aspects.

4.1. Use Case Specific Design Challenges

Use case specific design aspects reference the use cases elaborated in Chapter 3 and describe design challenges and considerations for workspaces and tasks.

4.1.1. Workspaces

Users navigate workspaces with a tabbed navigation in the application bar. There is no restriction on the number of workspaces a user can have access to. Thus, if a user has access to a large number of workspaces, they do not fit into the the horizontal space of the application bar. Material design proposes the use of swipe gestures or dropdowns in order to handle this case. However, those solutions don't scale with large numbers of tabs. The suggestion of thesis providing mockups and guidelines [Ka15] to address this issue was by offering a large dropdown inconsistent with the material design language. However, this approach does not scale either. Thus, a different approach is suggested and implemented, the use of a dedicated workspace overview page. In case there are too many workspaces to be displayed in the horizontal space of the application bar, the user can select a new tab called *All Workspaces*. This navigates him to a list of all available workspaces. As it is a dedicated page, it scales well with many workspaces, as it offers vertical scrolling to navigate them. The main drawback is that the user potentially looses the context of the

current task, as a completely new page is loaded. Future improvements might be an incremental search to aid the retrieval of large number of workspaces and different visualizations such as a grid-based over a list-based view of the workspaces.

4.1.2. Tasks

A user can view tasks in a list next to the entity content. Similar to the problem discussed in Section 4.1.1, a large number of tasks provides a design challenge, especially since every tasks contains additional meta data. The previous design was a list of tasks with the metadata responsively hidden in an edit menu as shown in Figure 4.1.

A suggested improvement that has been implemented in the client is different trade-off between vertical and horizontal space as shown in Figure 4.2. The tasks metadata now takes up more vertical space. Additionally, clicking on the task title collapses and expands the metadata, leading to a potentially faster navigation if the interaction is known. The interaction of collapsing and expanding tasks this way can be either suggested with tooltips, an onboarding process, or through a product documentation.

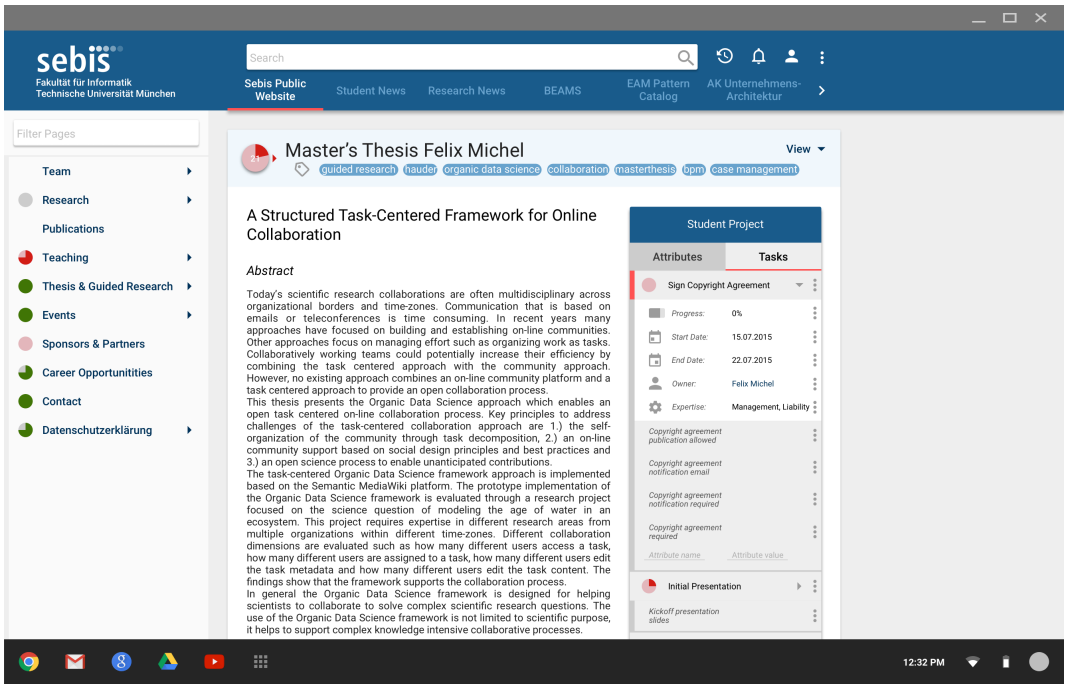


Figure 4.1.: Design of tasks in the ScoioCortex Content Manager [Ka15]

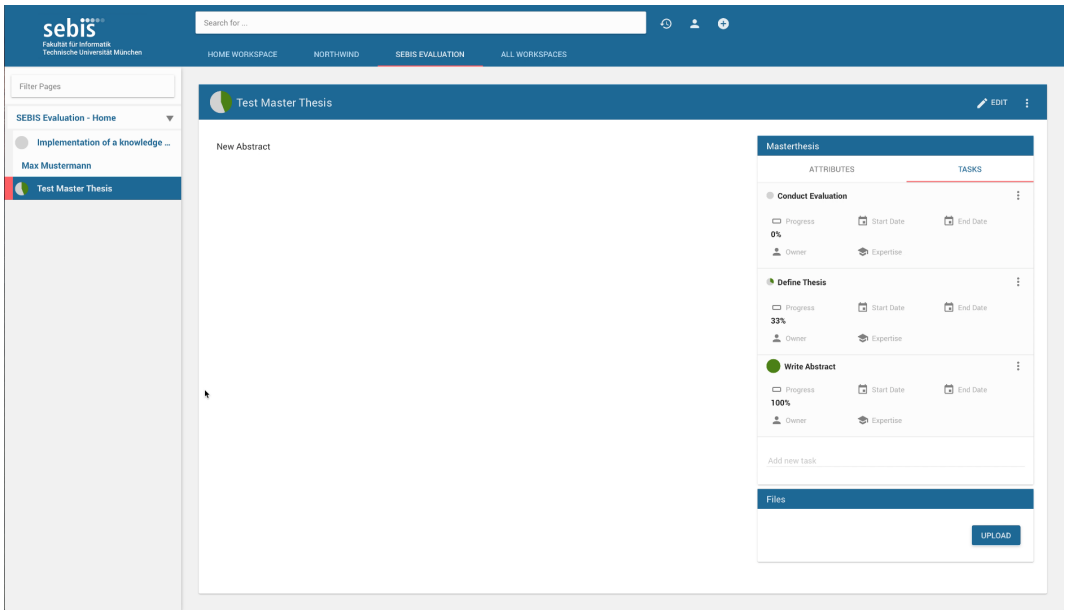


Figure 4.2.: Updated design of tasks in the ScoioCortex Content Manager

4.2. General Application Design Challenges

This section is concerned with the design challenges of the application in general. It focuses on a discussion of the Material Design language and the mockups and design guidelines that have been used as the foundation of this thesis.

4.2.1. Material Design

Material Design is a design language developed by Google [Mat] and offers a set of guidelines for designing the experience and interface of digital products. It is used internally by Google for their ecosystem of digital products and available for third parties.

Material design has been chosen as the design language for the content manager before the implementation of this work started. The mockups and design guidelines [Ka15] that formed the foundation of the content manager adhered to many aspects of the material design language.

There are several challenges when Material Design is applied to the content manager. One challenge is the use of whitespace. Material Design clearly sets guidelines on how much whitespace is added to which element using paddings and margins. Angular Material applies those standards.

This generally leads to nicer aesthetics, but reduces the information density on any given screen. In case of the content manager, which often requires to display lots of information at once, it leads to a reduced overview as users have to scroll more to consume the same amount of information.

Currently, the solution to this problem is a customization of the Angular Material specifications, reducing the amount of margins and paddings when necessary. However, this leads to inconsistencies with the Material Design Language, inconsistencies within the application, and will make it difficult to maintain consistencies across different SocioCortex clients.

4.2.2. Mockups and Design Guidelines

The thesis [Ka15] formed the basis of the technical implementation of the content manager. It provided both the general requirements and specification in terms of high-fidelity mockups. During the initial phase and the development process several shortcomings of the existing work were identified. An exemplary listing of those shortcoming and recommendations are presented in the following in order to aid future work. One shortcoming is the completeness of the mockups with regards to the described use case. For example, a critical use case of the proposed content manager is the WYSIWYG editing interface. While it was named as a requirement, it was not specified via mockups. Due to the missing specification, such use cases had to be designed and implemented nearly ad-hoc and untested. Another shortcoming is the completeness of the mockups and description with regards to the complete user flow. While it is important for a development process to have a complete specification of the user flow for specific tasks, it was often not possible to deduce how the user flow should look due to missing textual descriptions or mockups. In order to increase product quality in the future complete, correct, up-to-date, tested mockups could provide a valuable basis for future development.

5. Technical Implementation

This chapter describes technical implementation of the SocioCortext Content Manager. This includes the general setup and technologies used for the development, an overview of the application architecture, as well as the discussion of general technical challenges and aspects.

5.1. Architecture and Technologies

This section describes the structure and technologies used to build the content manager. Figure 5.1 gives a short overview of the architecture which are explained in further detail in this section.

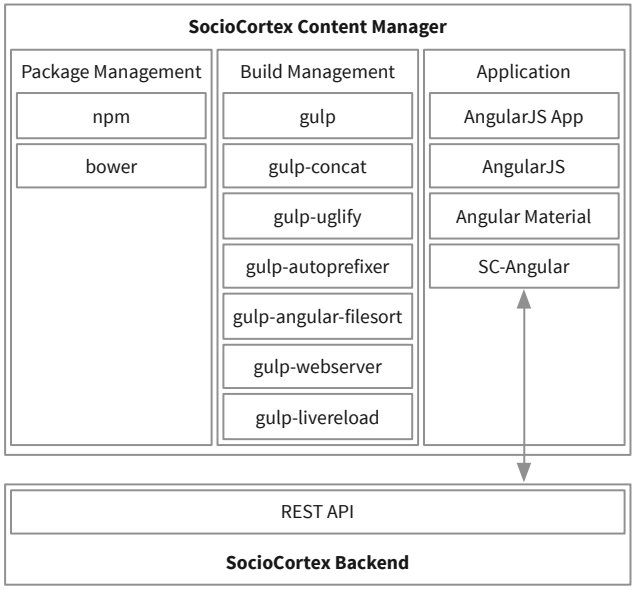


Figure 5.1.: Overview of the architecture and technologies of the content manager

5.1.1. Package Management

Two package managers are used for the client, *NPM* [NPM] and *Bower* [Bow].

NPM, the Node Package Manager, is an open source package manager in the NodeJS ecosystem [NPM]. In this thesis it is used to install and manage dependencies necessary for the build process. The build process is described in detail in the next section, Section 5.1.2.

Bower is an open source package manager for client-side libraries [Bow]. Important dependencies managed with *Bower* are described in Section 5.1.3.

5.1.2. Build Management

Gulp is an open source streaming build system being used for the build management of this project [Gula]. Several gulp plugins are being used in order to automate and optimize parts of the build process.

5.1.2.1. gulp-concat

For the production build, gulp-concat [Guld] concatenates all JavaScript files into a single file and all compiled CSS files into a single file. This reduces the number of individual HTTP requests in order to increase the page loading speed.

5.1.2.2. gulp-uglify

For the production build, gulp-uglify [Gulg] minifies JavaScript files in order to increase the page loading speed by having to server smaller files.

5.1.2.3. gulp-autoprefixer

Gulp-autoprefixer [Gulc] automatically adds browser-specific CSS prefixes in order to increase the cross-browser compatibility of an application.

5.1.2.4. gulp-angular-filesort

The concatenation and minification of AngularJS file in the wrong order often leads to errors. Gulp-angular-filesort [Gulb] automatically sets the correct order in which AngularJS files and dependencies have to be in for the concatenation and minification to work correctly.

5.1.2.5. gulp-inject

Gulp-inject [Gule] automatically injects CSS and JavaScript files into a specified HTML. As this works fully automatic, potential errors by including or missing to include files are reduced. Additionally, it saves time not having to update dependent CSS and JavaScript files manually.

5.1.2.6. gulp-webserver

In development, gulp-webserver [Gulh] allows the local serving of the application. As this plugin can be configured with other gulp commands, operations such as injecting CSS files into the HTML can be done automatically before serving the HTML files.

5.1.2.7. gulp-livereload

In development, gulp-livereload [Gulf] allows the automatic reloading of served HTML, CSS, and JavaScript files. This intends to speed up development, because a reload does not have to be triggered manually once a change to a source file has been made.

5.1.3. Important Front-end Libraries

5.1.3.1. AngularJS

AngularJS [Angb] is a client-side JavaScript framework for building single page applications. The framework offers constructs such as directives, services, and controllers to build and manage large-scale applications. Version 1 of the framework is used in the implementation of the content manager, as the new and improved version is not released as a stable version yet.

5.1.3.2. Angular Material

Angular Material [Anga] is an open source JavaScript library that implements large parts of the Google Material Design Language [Mat] as reusable AngularJS components. This makes it fast and easy to adapt the Google Material Design Language to an AngularJS application without having to translate the design guidelines manually. Most of the functionality of Angular Material is provided via directives which can be used in the application.

5.1.3.3. SC-Angular

SC-Angular [SCA] is a JavaScript library that provides an abstraction of the SocioCortex Backend REST-API for AngularJS 1 applications. Instead of interacting with the API directly by sending HTTP request or using a general purpose abstraction layer, SC-Angular provides additional convenience when interacting with SocioCortex.

The library consists of eight services and two directives. The services are:

- *scAuth*, which provides functions for user authentication such as logging a user in and out.
- *scData*, which allows to manipulate data objects such as workspaces or entities. Due to the high dominance of manipulating data in the content

manager, this is one of the most importance services provided by SC-Angular in the context of this thesis.

- *scModel*, which provides access to objects such as entity types and attribute definitions. As the SocioCortex Modeler is responsible for allowing the manipulation of such data, it is not relevant for the content manager.
- *scMxl*, which provides functionality regarding the MxL query language.
- *scPrincipal*, which provides functionality regarding user accounts such as retrieving meta data about the currently logged in user. This is especially used for the user profile component of the content manager.
- *scRoute*, which provides routing functionality.
- *scSearch*, which allows access to the global search of SocioCortex. This service is used to the global search of the content manager.
- *scUtil*, which provides access to various auxiliary functions.

The two directives *scHref* and *scSrc* are also available. They are used for rendering images and links in the interface.

5.2. Overview of the Application Architecture

This section gives an overview application architecture by describing major components in the system. AngularJS 1, the version used for the implementation of the content manager, does not support a component-based application architecture. However, some system functionality has been grouped by using AngularJS specific constructs as controllers, directives, and services. Furthermore, folders and naming conventions are used to map this structure to the application in order to mimic a component-based architecture. Figure 5.2 gives an overview of the described components.

Figure 5.2.: Overview of the components of the content manager

Main Navigation is concerned with the tree-based navigation of pages in a sidebar. It includes the ability to search for pages by title in the scope of this navigation.

Main Content is concerned with rendering and providing functionality for entities. It contains the general layout of an entity with its unstructured and structured content.

Workspaces includes all workspace related functionality, for example CRUD operations on the workspaces or workspace settings.

User Management contains general user-related functionality such as logging a user in and out.

User Profile includes all user profile related functionality such as the rendering of the user profile page with the meta data and tasks of a user.

Search provides functionality for the global search. As opposed to the local search in the *Main Navigation* component, it searches full-text across all workspaces.

Attributes and Tasks includes the functionality for attributes and tasks of entities. They are grouped together, as they are closely related in state and behavior.

Files includes all file-related functionality such as uploading and downloading files in various places of the application.

Feed includes all feed-related behavior and has been implemented before this thesis. It is integrated in application as it's own highly separated component.

Data Tables will include the implementation of the data tables once they are integrated from another project that is dedicated to the development of the data tables.

5.3. Behavioral Model

This section gives a brief overview of the SocioCortex integrated model and its relationship to the content manager. Figure 5.3 shows the integrated model of the SocioCortex Backend. Grey concepts especially refer to the SocioCortex Content Manager, green ones to the SocioCortex Modeler, and white ones to the SocioCortex Visualizer or other aspects. As the content manager only focuses on providing a specific set of functionality instead of mapping all of the functionality of the SocioCortex Backend, only a part of the model is relevant for the content manager. Concepts with a high relevance for the

content manager are workspaces, entities, files, attributes, and tasks. Concepts such as entity type, attribute definition, task definition, have less relevance for the content manager as most of their behavior is mapped to SocioCortex Modeler Client. Table 5.1 gives an overview of certain operations and their results on the model.

Concept	Operation	Results
Workspace	Create	A workspace is created with only a single entity of type *text page* that acts as the homepage of the workspace.
Workspace	Delete	All associated data, including entities, attributes, tasks, entity types, attribute definitions, and task definitions are deleted.
Entity	Create	An entity with a relationship to the respective workspace is created.
Entity	Move	All associated files, attributes, tasks, and subpages are moved.
Entity	Duplicate	All associated files, attributes, tasks, are copied. Subpages are not copied.
Entity	Delete	All associated files, attributes, tasks, and subpages are deleted.
Attribute	Create	The attribute is created and associated with the entity it was created on. Only free attributes can be created using the content manager.
Attribute	Delete	The file is deleted and not associated to an entity anymore.
File	Create	The file is created and associated with the entity it was created on.
File	Delete	The file is deleted and not associated to an entity anymore.
Task	Complete	The progress metadata is set to a value of 100.
Task	Skip	The status about the skipping is recorded in the task.
Task	Delete	Associated attributes are not deleted.
Entity Type	Delete	Even though deleting an entity type can only be done in the SocioCortex Modeler, it can have effect in the content manager. If an entity type is delete and there is a entity of that type accessible in the content manager, this entity fill get the default type *text page*.
Attribute Type	Delete	If an instance of the attribute definition exists, it is converted to a free attribute instead of a typed attribute. This is an operation that can only be done in the SocioCortex modeler.

Table 5.1.: Table about results of selected operations on the SocioCortex integrated model relevant for the content manager

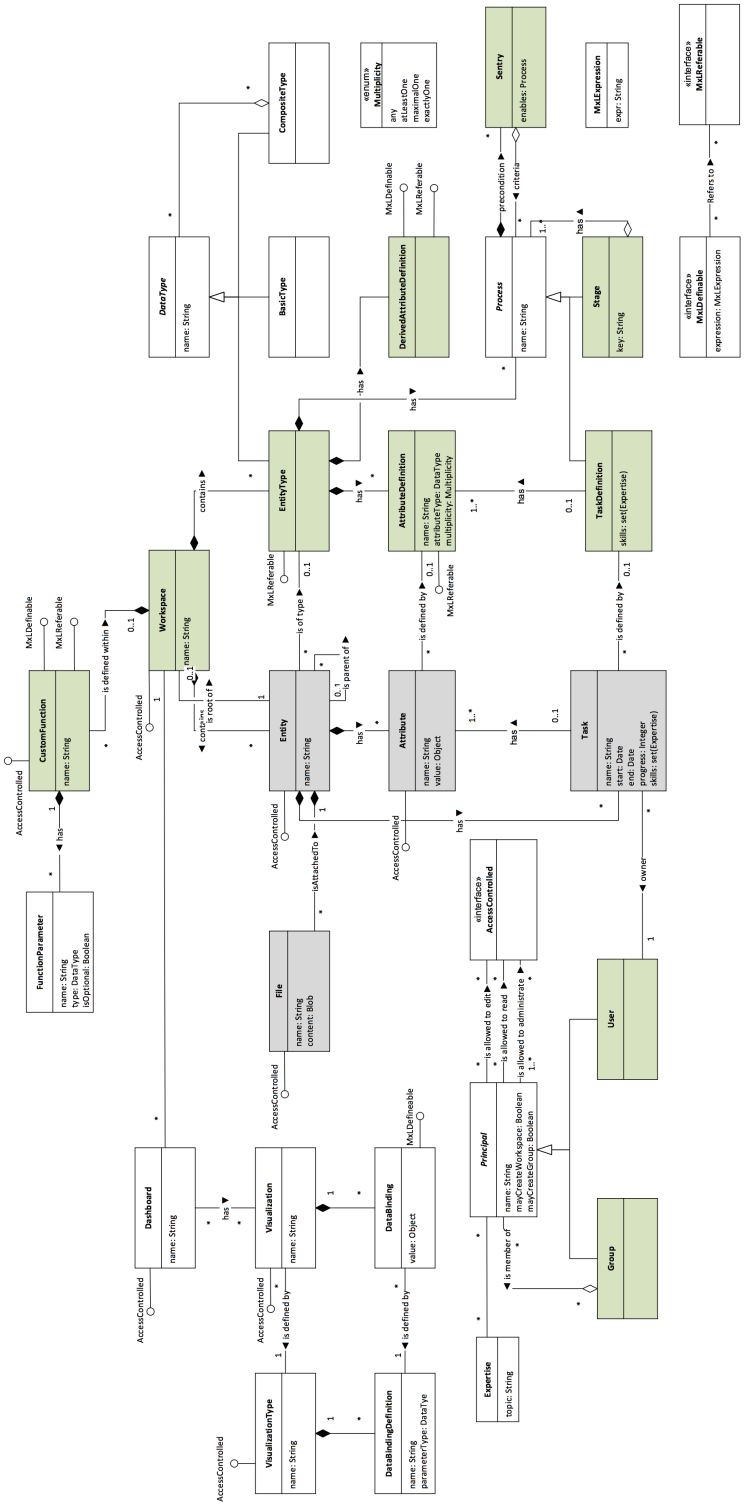


Figure 5.3.: The behavioral model of SocioCortex. Grey denotes concepts predominantly relevant for the content manager, green for modeler, white for the visualizer and other clients.

5.4. Technical implementation of selected use cases and requirements

This section gives an overview of the technical implementation of selected use cases and requirements implemented in the content manager, that provided specific challenges. The documentation of those challenges will aid in the continuous development of the content manager.

5.4.1. Content Editing Interface

Editing unstructured entity content requires a web-based editor. There are already good existing solutions in the form of open source web-based editors. There are several requirements that have to be fulfilled by such an editor in the context of the requirements of the content editor:

Open Source The editor has to be released under and open source license. The goal is to reduce cost and make the technical implementation easier to understand as the source code and technical discussions are easily available. Additionally, this requirement permits the content manager itself to be open source.

Rich-Text The unstructured content of an entity should be able to include various formatted media such as formatted text or images. Thus, the editor has to provide such rich-text formatting capabilities.

WYSIWYG and HTML View The editor has to provide an WYSIWYG editing interface. In such an interface the objects can be manipulated directly similar to MS Word, providing an easy to use editing experience. Additionally, it should be possible to switch between the WYIWYG editing interface and an HTML source view, providing the ability to manipulate HTML directly.

Extendability As the editor is supposed to be extended with SocioCortex-specific functionality such as creating links between entities and integrating with the SocioCortex API to upload and download files and images. Thus, the editor it needs to be easily extendable.

Compatibility with AngularJS While any web-based editor can be potentially used in an AngularJS project, it might require an effort to make it work within the paradigms and constraints of AngularJS. Thus, the editor should be easy to integrate.

Documentation The editor should be well documented to make integration into the content manager and the subsequent customization and extension as easy and fast as possible.

Cross-Browser Support The editor should work in as many browser as possible in order to provide a high coverage of browsers in which the whole SocioCortex Content Manager might be used.

Active Development The editor should be actively developed to ensure continuous cross-browser support, fixing of known issues, and the potential extension with new features that might be required in the future.

Several editor alternative have been evaluated, with TinyMCE [Tin], Trix [Tri], Quill [Qui] fulfilling all the requirements to varying degrees. In the end, TinyMCE has been selected due to its good integration with AngularJS and documentation with regards to the official documentation and existing information in various online communities. The integration between AngularJS and TinyMCE is provided by the library ui-tinymce [**?**], wrapping the editor in an AngularJS directive.

5.4.2. Images in Entity Content

Entities are able to contain references to images in their unstructured content. Those images are supposed to be rendered inline within the editor content. This content is serialized and persisted as a string of HTML. If the application wants to render the unstructured content, there are several step involved. The following shows an example of rendering an the entity content of a requested entity, with an visual overview given in Figure 5.4:

1. The content manager requests an entity using SC-Angular.
2. SC-Angular makes the request to the SocioCortex API in order to retrieve the entity.
3. The API returns the entity object, which includes a serialized string of HTML. However, images are not included in this string as an HTML *img* tag with a *src* attribute set to the path of the image. They are included as an *img* tag with a custom *sc-src* attribute that contains the ID to the image file.
4. For each image in the HTML string, another request has to be made using the encoded ID in order to retrieve the image and set the resulting data to the *img* tag. This is simplified with the *scSrc* directive provided by SC-Angular.
5. The SC-Angular *scSrc* directive request the individual images.

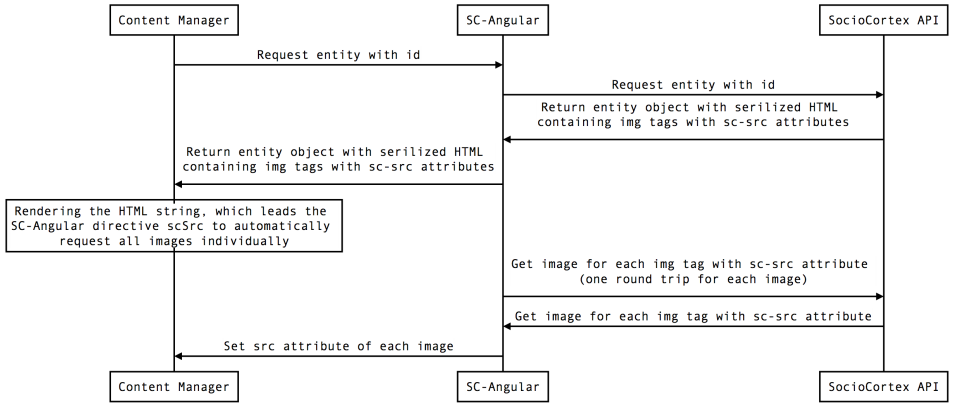


Figure 5.4.: Overview of the steps involved in rendering images inside entity content

The reason why only the IDs are encoded in the *img* tag is that images can have permissions. When individual request have to be made using the API, the permission can be checked on each image, ensuring only users with the correct permissions will be able to get the images.

A major problem with this approach is the usage in conjunction with the content editor. On load, the editor TinyMCE expects a string of HTML to render in it's editable interface. It does not provide functionality for preprocessing the HTML before rendering. Thus, before initializing TinyMCE, the HTML string needs to be processed in such a way, that it contains HTML image tags with the final *src* attribute set.

An approach that failed was the usage of the AngularJS *$compile* service. The idea behind using the *$compile* service was to use it to compile the entity content HTML string with the SC-Angular *scSrc* directive and setting the resulting HTML string as the input for the TinyMCE editor. However, there was no reliable way to load the editor only once string was compiled.

The chosen solution was to manually parse the HTML string, extract the IDs from the *sc-src* attributes, make request for the resources and generate the final HTML string which is then passed to the editor. The same functionality and encoding and decoding is used for modifying images inside the content

editor, the the string that is saved has to contain the IDs in the same format as the result form the API.

5.4.3. Links in Entity Content

Links in the entity content behave like images as described in Section 5.4.2. Each link is serialized as an HTML anchor tag and has the ID to the linked resource encoded as a custom HTML attribute *sc-href*. The directive *scHref* is provided by SC-Angular in order to request the actual URL to the linked resource. The problems of this approach were solved in the same way as for images, manually extracting the IDs.

5.5. Discussion of general technical implementation aspects

This section discusses general technical implementation aspects which are not use cases specific. This includes the discussion of the SocioCortex API and AngularJS as the main framework used in the development of the content manager.

5.5.1. SocioCortex API

A common problem with the use of the SocioCortex API was the general REST problem of over-fetching of data. In most implemented views, only a small set of data returned from API endpoints is needed. Unneeded data sent over the network reduces the performance of the application on limited network connections such as on mobile devices. Approaches such as GraphQL try to solve this problem and might be an alternative to solve this issue in the future.

Another problem with using the API documentation during development was the inability to search it effectively. For example, it was a common use case

to search for specific properties returned in different endpoints. This was only manually possible due to a lack of search. This is a shortcoming of the Open Source API documentation framework Swagger, which has not implemented such as functionality yet. However, there is already a pull request to support this.This should be added as soon as possible to the SocioCortex API documentation, as it becomes available.

5.5.2. AngularJS

Usage of version 1 of the AngularJS framework was a requirement of the implementation for the content manager. Additionally, the existing code was was written in the same version. There are many drawbacks in AngularJS 1 that are supposed to be addressed in the second version or in other comparable JavaScript frameworks such as ReactJS. Such drawbacks include the lack of support for component-based architecture or performance issues in large scale applications with many data bindings. As a switch to AngularJS 2.0 once it is stable or a comparable JavaScript framework will likely require a complete rewrite of the application, it should be evaluate if such a change is appropriate before implementing more functionality in the content manager.

Part IV.

Evaluation

6. Evaluation

An evaluation was conducted in order to gain initial insights into the general usability of the content manager in terms of task completion and satisfaction. The goal was to identify major usability issues and areas of improvement. This evaluation was conducted together with the evaluation of SocioCortex Modeler as part of the master thesis [Sc16]. The same participants were evaluating first the modeler client and then the content manager. While the setup of the evaluation was designed for two independent evaluations, the evaluation of the modeler might have influenced the results of the content manager. This possible effect was considered and neglected due to the estimations that effects might not impede the goal of uncovering major issues.

6.1. Methodology

The methodology of the evaluation consisted of two parts, a usability test and a post-test questionnaire. The usability test focused on participants using the content manager to solve task from a fictitious scenario. The post-test questionnaire combined a standardized test with custom questions.

6.1.1. Usability Test

The usability test consisted of five predefined tasks in a scenario the user had to solve in a live system of the content manager. During the test, the think aloud method [LR93] was used, meaning the test participant was asked to formulate observations and issues verbally for the moderator. In some cases, the participant was probed during the test with follow-on question in order to

gain maximum insight into encountered issues or potential improvements. A drawback of the think aloud method and the probing questions are their impact on the cognitive processes of the participant and quantitative measurements such as the task completion time. However, as the test was designed to identify major usability issues and not measure or improve quantitative metrics, those drawbacks were neglected.

6.1.2. Questionnaire

An post-test questionnaire formed the second part of the evaluation. In this part, a standardized usability questionnaires was combined with three custom open ended questions. A standardized usability questionnaires was used over a fully custom one in order to potentially gain more reliable results and get insights to into the usability compared to other systems. Several standardized questionnaires have been evaluated, with the system usability scale (SUS) [Bo96] being chosen for it's simplicity, duration and widespread use. According to [Bo96], the questions of the SUS are:

1. I think that I would like to use this system frequently
2. I found the system unnecessarily complex
3. I thought the system was easy to use
4. I think that I would need the support of a technical person to be able to use this system
5. I found the various functions in this system were well integrated
6. I thought there was too much inconsistency in this system
7. I would imagine that most people would learn to use this system very quickly
8. I found the system very cumbersome to use
9. I felt very confident using the system
10. I needed to learn a lot of things before I could get going with this system

Finally, three custom open ended question were used to elicit previously unknown feedback and gain insights about the relative usability changes in comparison the the predecessor system Tricia. The complete questionnaire is documented in Appendix A.

6.2. Participants

A set of six participants were selected for the evaluation. The number of participants was intentionally kept small in order to quickly identify the most pressing issues, fix them, and continue testing with the improved version. The participants were part of two groups, members of the SEBIS chair and industry partners. All participants had experience with Enterprise Architecture Management (EAM).

6.3. Scenario

The participants were given a scenario with five tasks. The scenario revolved around the administration of masters theses with the participant being set in the role of a university faculty member. The tasks included the following small subset of basic operations in the system:

6.3.1. Creating entities

Participants are tasked to perform basic CRUD operation on entities. Part of the first task is the creation of such an entity of a specific type, a master thesis, in the scope of a predefined workspace.

6.3.2. Editing attributes

Next, participants have to edit the values of several attributes with different types (e.g. a text attribute and a date attribute) of the entity that was created by them.

6.3.3. Editing entity content

Additionally, participants are asked to edit the rich-text content of the master thesis entity using the editor. An underlying goal of the task is to create and save a small amount of text, an abstract of the master thesis.

6.3.4. Completing tasks

Predefined tasks of the entity were to be completed by the participant. This could be achieved in several ways such as setting triggering the action explicitly.

6.3.5. Skipping tasks

A single task is shown as not necessary. The participant has to skip the task instead of completing it as the task before.

6.3.6. Uploading files

Participant are tasked to upload a PDF file found locally on the computer of the evaluation to a specified entity.

6.3.7. Adding and editing free attributes

After completing the upload, the participant has to create a new free attribute supposed to hold a reference to the PDF file that was uploaded in the previous task.

6.3.8. Searching entities

Another task is to find an entity not created by the participant. Several ways are possible to achieve this, but the intended goals of the task was to test the usability of the global search.

6.3.9. Renaming entities

After finding a specific entity using search, the participant is supposed to rename the title of the entity.

The complete scenario and list of tasks are documented in Appendix A.

6.4. Results

This chapter summarizes the results of both evaluation parts, the usability test and the questionnaire.

6.4.1. Usability Test

The complete results of the usability test are documented in Appendix A. The following will give a short summary of the most important issues.

Creating entities The implemented interaction design to create a new entity is to use the material design specific *more* icon in the application bar and choosing new entity from the appearing context menu. Most participants discovered the way to create an entity on their own. However, most of them did not expect the action at the current location and took a lot of time to explore the interface in order to find the action. Another problem was the naming of menu item to create an new entity. In the first three tests, the naming was *Create Subpage*, while the participants expected *Create Entity*. It was changed after the first three tests, leading to the following participants not having any problems understanding it.

Editing attribute values The implemented interaction design for editing attribute values is to put each value that is supposed to be editing in an explicit edit mode using a context menu located at every attribute. Participants consistently expected an *edit-in-place* behavior of the attribute value. They tried to click on the attribute name and/or value and expected to be able to edit it. All of the participants managed to find the context menu but expressed their desire to edit the attributes inline.

Renaming entities The implemented interaction design for renaming entities requires users to navigate to a entity and chose the action *rename* from a dropdown menu. However, most participants navigated to an entities, entered the edit mode, and tried to click on the title. They were expecting an *edit-in-place* like behavior, allowing them the edit the entity title inline. After this approach did not work, most figured out using the intended drop down menu after some exploration.

Adding values to a free attribute The implemented interaction design for adding a value to a free attribute of an entity is to put the attribute in the edit mode and using the autocomplete menu to enter the desired value, with the type of the free attribute being guessed. Only one of the participants managed to complete the task by himself. All off the participants either expressed

their desire to declare the type of the free attribute manually first or have the behavior of the system better communicated in the user interface.

6.4.2. Questionnaire

The complete results of the questionnaire can be seen in appendix A. The final SUS score, the measure by which the usability of a system according to SUS is determined, ranges between 0 and 100 [Bo96]. The content manager scored on average across all 6 participants 65.4 points. According to two studies this is slightly below the average of 68 or 68.2 points respectively [Sa11] [BKM09].

The most recent study, [Sa11], took 500 evaluations into account and determined an average score of 68 [Sa11].

Another study, [BKM09], determined an average of 69.5 points across 3463 evaluations. However, this includes evaluations of several user interface types. For the evaluations regarding the *web* interface type, the type most relevant in the context of this evaluations, the average SUS score was 68.2 points across 1433 evaluations [BKM09].

6.5. Potential Improvements

This section shows several potential improvements that could be made on the content manager. The improvements are mere suggestions that should be validated and tested before potentially implementing them.

6.5.1. Improving the entity creation process

Currently, creating an entity was a difficult task for most participants. If creating an entity is selected as a primary action that users should be able to perform easily and quickly, there are several ways this could be improved. One

	Question	Average SUS Scale	SUS Contribution
1	I think that I would like to use this system frequently	3.50	2.50
2	I found the system unnecessarily complex	2.33	2.67
3	I thought the system was easy to use	3.33	2.33
4	I think that I would need the support of a technical person to be able to use this system	2.67	2.33
5	I found the various functions in this system were well integrated	3.00	2.00
6	I thought there was too much inconsistency in this system	2.17	2.83
7	I would imagine that most people would learn to use this system very quickly	3.00	2.00
8	I found the system very cumbersome to use	1.83	3.17
9	I felt very confident using the system	4.00	3.00
10	I needed to learn a lot of things before I could get going with this system	1.67	3.33
		Total Average:	**65.42**

Table 6.1.: Table with SUS results

suggestion is the use of a floating action button as shown in Figure 6.1. The prominent color, placement, and consistency with the material design language could provide an improved solution.

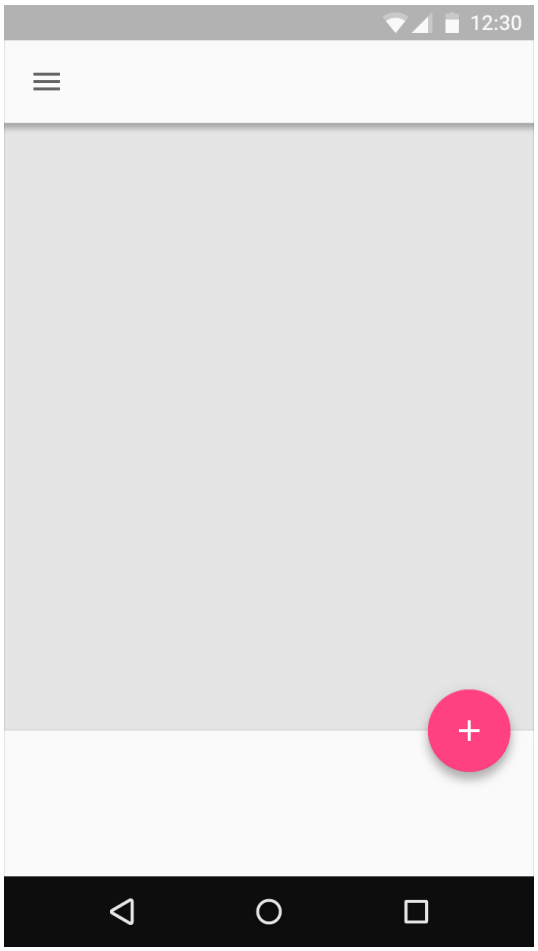


Figure 6.1.: The floating action button pattern from material design language [FAB]

6.5.2. Improving the editing of attributes

Editing attribute values was often seen as an inconvenience by the participants. A common suggestion by the participants was to change the way an attribute is set into an *edit* mode, by allowing an *edit-in-place* interaction de-

sign. The participants noted that editing several attributes this way might be too cumbersome.

Currently the user flow for successfully editing an attribute is the following:

1. Navigate to the entity that contains the attribute that should be edited
2. Click on a button next to attribute
3. Select the action *edit* from the dropdown
4. Edit the value of the attribute
5. Confirm the edit with a click on a *done* button next to the attribute

With an *edit-in-place* design, the interaction would be the following:

1. Navigate to the entity that contains the attribute that should be edited
2. Click directly on the attribute name or value
3. Edit the value of the attribute
4. Click outside the value of the attribute to confirm the edit

Another alternative is a combination of both approaches:

1. Navigate to the entity that contains the attribute that should be edited
2. Click on a button that sets all attributes of that entity in an *edit* state
3. Edit the value of one or more attributes
4. Confirm the edit with a click on a *done* button, saving the values of all attributes

The three alternative approaches should be evaluated and tested separately in order to make a decision or find a better alternative. Currently, the third alternative shows the biggest promise as the interaction is clearly communicated via dedicated buttons, potentially making the interaction easier to understand and more predictable for user not familiar with the *edit-in-place* interaction pattern.

6.5.3. Improving the naming of entities

A naming of entities should be agreed upon and consistently used throughout the content manager and potentially other clients. The participants of this evaluation were more familiar with the term *entity* as opposed to *page* or *subpage*. However, all participants had knowledge of concepts used in the predecessor systems Tricia. If future users should be able to understand the concept without prior knowledge it should be evaluated which term to use. Additionally, the terms and concept could be explained in the application or supporting resources such as training material.

6.6. Discussion

As a general note, the results from the SUS are eventually invalid. During the test, participants consistently noticed in several cases that they have filled out the wrong answer in the SUS due to wrong understanding or remembering of the scale. While a common template for the SUS was used, in the future the design of the test could be improved by placing the description of the scale at every question or using other means to make the scale more obvious. Furthermore, participants could be made aware to check their answers with regard to the questionnaire scale before they hand it in. Additionally, there is an issue in applying the test to non-native English speakers. Several participants did not understand the word *cumbersome*. This issue has already been identified in [Fi06], with the recommended solution being the substitution of the word *cumbersome* with *awkward* or *cumbersome/awkward*. During the test, several participants asked to clarify the meaning of the the word *cumbersome* and it was explained to the participant. While it might not have influenced the test outcome, it is recommended to use the changed version of the test for non-native English speakers in the future. As a general remark, some of the issues identified could have been identified and iterated upon without a functional prototype, but with the use of low- and high-fidelity prototyping which would have taken less effort and potentially increased product quality faster.

Part V.

Conclusion

7. Conclusion and Outlook

7.1. Conclusion

The aim of this thesis was to implement a specific client for SocioCortex, the SocioCortex Content Manager. The client combines concepts from the hybrid wiki with the task-oriented approach of the Darwin Wiki. It provides the user with a generic interface to manipulate data from SocioCortex. Overall, the client was implemented following the constraints regarding use cases, design, and technologies.

The use cases were collected, defined, and elaborated before the implementation of the client. During the development, relevant design and technical challenges have been solved or documented for future work.

Finally, the implemented version was evaluated with 6 participants in a usability test with a post-test questionnaire. The standardized questionnaire which was used, the system usability scale (SUS), resulted in a score indicating a slightly below average usability.

7.2. Outlook

The result of this work, the SocioCortex Content Manager, can be expanded and improved in several aspects. First of all, the potential improvements suggested based on the evaluation in 6.5 should be evaluated, tested, and implemented. Fixing the identified problems will help to increase the current usability score.

The SocioCortex Content Manager should also be integrated with the independently developed project for the data tables. This will provide a better way to navigate entity instances and edit multiple attribute values on the same screen, benefiting from the advantages of a spreadsheet-like user interface.

As described in Section 5.5.2, the currently used front-end JavaScript framework AngularJS 1.x has major drawbacks which are solved by other JavaScript frameworks and will be potentially solved by the next iteration of AngularJS. A migration to a different framework or the next version of AngularJS will likely require a major rewrite of large parts of the application. Because of this, it should be evaluated if continued development with the current framework is the best choice as the migration effort for a larger code base will only increase.

Additionally, it should be evaluated to use a more user-centered design process. This will help to discover and fix potential usability problems before the cost for the development of a functional prototype has already been invested.

Another area of potential improvement is a better integration among the SocioCortex Content Manager and the SocioCortex Modeler. Currently, the two clients are not integrated. However, some users might need to use both clients in order to fulfill certain tasks. An example for an improvement is a better navigation between the entity instances in the SocioCortex Content Manager and the entity types in the SocioCortex Modeler. A specific use case for using both clients is a single user having the task to model a certain workflow and adding the data himself.

Bibliography

[Anga] *Angular Material*. `https://material.angularjs.org/latest/`. Last accessed on September 2016.

[Angb] *AngularJS*. `https://angularjs.org/`. Last accessed on September 2016.

[BKM09] Bangor, A.; Kortum, P.; Miller, J.: *Determining what individual SUS scores mean: Adding an adjective rating scale. Journal of usability studies*. 4(3):114–123. 2009.

[Bo96] Brooke, J.; others: *SUS-A quick and dirty usability scale. Usability evaluation in industry*. 189(194):4–7. 1996.

[Bow] *Bower Package Manager*. `https://bower.io/`. Last accessed on September 2016.

[Br12] Bry, F.; Schaffert, S.; Vrandečić, D.; Weiand, K.: *Semantic wikis: Approaches, applications, and perspectives*. In *Reasoning Web International Summer School*. pages 329–369. Springer. 2012.

[Bü15] Bürgin, P.: *Design and Prototypical Implementation of a Dashboard System for Visualizing Semi-Structured Data in a Traceable Way Design and Prototypical Implementation of a Dashboard System for Visualizing Semi-Structured Data in a Traceable Way Design and Prototypical Implementation of a Dashboard System for Visualizing Semi-Structured Data in a Traceable Way*. Master's thesis. 2015.

[FAB] *Material Design Floating Action Button*. `https://material.google.com/components/buttons-floating-action-button.html`. Last accessed on September 2016.

[Fi06] Finstad, K.: *The system usability scale and non-native english speakers. Journal of usability studies.* 1(4):185–188. 2006.

[GBD09] Gantz, J.; Boyd, A.; Dowling, S.: *Tackling Information Overload At the Source. IDC White Papers.* 2009.

[Gi13] Gil, Y.; Knight, A.; Zhang, K.; Zhang, L.; Sethi, R.: *An Initial Analysis of Semantic Wikis.* In *Proceedings of the Companion Publication of the 2013 International Conference on Intelligent User Interfaces Companion.* IUI '13 Companion. pages 109–110. New York, NY, USA. 2013. ACM.

[Gula] *gulp.* `http://gulpjs.com/`. Last accessed on September 2016.

[Gulb] *gulp-angular-filesort.* `https://github.com/klei/gulp-angular-filesort`. Last accessed on September 2016.

[Gulc] *gulp-autoprefixer.* `https://github.com/sindresorhus/gulp-autoprefixer`. Last accessed on September 2016.

[Guld] *gulp-concat.* `https://github.com/contra/gulp-concat`. Last accessed on September 2016.

[Gule] *gulp-inject.* `https://github.com/klei/gulp-inject`. Last accessed on September 2016.

[Gulf] *gulp-livereload.* `https://github.com/vohof/gulp-livereload`. Last accessed on September 2016.

[Gulg] *gulp-uglify.* `https://github.com/terinjokes/gulp-uglify`. Last accessed on September 2016.

[Gulh] *gulp-webserver.* `https://github.com/schickling/gulp-webserver`. Last accessed on September 2016.

[HKM15] Hauder, M.; Kazman, R.; Matthes, F.: *Empowering End-Users to Collaboratively Structure Processes for Knowledge Work.* In *International Conference on Business Information Systems.* pages 207–219. Springer. 2015.

[Ka15] *Optimizing the User Experience of a Social Content Manager for Casual Users*. 2015.

[LC01] Leuf, B.; Cunningham, W.: *The Wiki Way: Quick Collaboration on the Web*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA. 2001. 0-201-71499-X.

[LR93] Lewis, C.; Rieman, J.: *Task-centered user interface design. A Practical Introductio*. 1993.

[Mat] *Google Material Design Language*. `https://material.google.com/`. Last accessed on September 2016.

[Meda] *MediaWiki - Editing Wiki Pages*. `https://www.mediawiki.org/wiki/MediaWiki`. Last accessed on September 2016.

[Medb] *MediaWiki - Editing Wiki Pages*. `https://en.wikipedia.org/wiki/Wiki`. Last accessed on September 2016.

[MNS11] Matthes, F.; Neubert, C.; Steinhoff, A.: *Hybrid Wikis: Empowering Users to Collaboratively Structure Information. ICSOFT (1)*. 11:250–259. 2011.

[NPM] *Node Package Manager*. `https://www.npmjs.com/`. Last accessed on September 2016.

[Os15] Ostner, M.: *Design and implementation of a task-centric social content management application for end-users Design and implementation of a task-centric social content management application for end-users Design and implementation of a task-centric social content management application for end-users*. 2015.

[Qui] *Quill Editor*. `https://github.com/quilljs/quill`. Last accessed on September 2016.

[Re16] Reschenhofer, T.; Bhat, M.; Hernandez-Mendez, A.; Matthes, F.: *Lessons learned in aligning data and model evolution in collaborative information systems*. In *Proceedings of the 38th International Conference on Software Engineering Companion*. pages 132–141. ACM. 2016.

[Sa11] Sauro, J.: *Measuring usability with the system usability scale (SUS)*. 2011.

[Sc16] Schrade, T.: *Implementing a Web Client for Integrated Data, Role, Function, and Task Modelling*. Master's thesis. 2016.

[SCA] *sc-angular*. `https://github.com/sebischair/sc-angular`. Last accessed on September 2016.

[SE14] SEBIS: *Software Engineering for Business Information Systems*. 2014.

[Sem] *Semantic MediaWiki*. `https://www.semantic-mediawiki.org/`. Last accessed on September 2016.

[Tin] *TinyMCE Editor*. `https://github.com/quilljs/quill`. Last accessed on September 2016.

[Tri] *Trix Editor*. `https://github.com/basecamp/trix`. Last accessed on September 2016.

[Wik] *Wikipedia*. `https://www.wikipedia.org`. Last accessed on September 2016.

A. Appendix

Evaluation Scenario Introduction

Scenario Introduction

In the scenario for this evaluation you are a PhD. Student at the SEBIS chair. You have access to a new knowledge management system at the chair, SocioCortex.

You are in charge of modeling data for the system as well as using it to support you in your activities such as overseeing students doing their thesis at the chair.

The following task will cover activites regarding the modeling of data as well as entering data for a specific use case.

Evaluation Scenario

Scenario 2 - Generic Client

Task 1: Create a new master thesis

A student is starting a new master thesis at the SEBIS chair. You are responsible in overseeing his work and want to enter the information in the system to keep other people at the SEBIS chair informed about the progress.

Create a new master thesis for a student and add the information: title, student name, and start date.

Task 2: Add an abstract to the newly created master thesis

The student sends you the abstract of his thesis and you want to add it to the page content. After you have added it to the page, mark the task “Write abstract” as completed.

Task 3: Skip the task “Conduct evaluation” of the master thesis

You decide together with your master thesis student that an evaluation is not necessary for the thesis and you want to skip it in the system.

Task 4: Upload the file “Company NDA.pdf” to the master thesis

The thesis you are overseeing requires a signed company NDA. You possess the NDA as a PDF file and want to attach it in the generic client so that other people at the SEBIS chair can see it. Upload it to the thesis. You find the PDF file on the local computer under “SEBIS Evaluation”.

Additionally, you want to create a new attribute for the master thesis that links to the PDF file you uploaded.

Task 5: Find and rename the master thesis “Implementation of a knowledge management tool”

A colleague asks you to rename a master thesis in the system for him. The master thesis in question is called "Implementation of a knowledge management **tool**" and he would like you to rename it to "Implementation of a knowledge management **system**"

Evaluation Questionnaire

Questionnaire

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5

10. I needed to learn a lot of things before I could get going with this system	1 2 3 4 5

Overall, how does this system compare to Tricia?

Overall, what would you like to change in the system?

Is there any other feedback?

Evaluation Complete SUS Results

SEBIS 1			
#	Question	User's Scale Position	SUS Contribution
1	I think that I would like to use this system frequently:	3	2
2	I found the system unnecessarily complex:	2	3
3	I thought the system was easy to use:	3	2
4	I think that I would need the support of a technical person to be able to use this system:	3	2
5	I found the various functions in this system were well integrated:	4	3
6	I thought there was too much inconsistency in this system:	1	4
7	I would imagine that most people would learn to use this system very quickly:	1	0
8	I found the system very cumbersome to use:	1	4
9	I felt very confident using the system:	5	4
10	I needed to learn a lot of things before I could get going with this system:	2	3
	Individual SUS Score		67.5
SEBIS 2			
#	Question	User's Scale Position	SUS Contribution
1	I think that I would like to use this system frequently:	4	3
2	I found the system unnecessarily complex:	2	3
3	I thought the system was easy to use:	3	2
4	I think that I would need the support of a technical person to be able to use this system:	4	1
5	I found the various functions in this system were well integrated:	2	1
6	I thought there was too much inconsistency in this system:	3	2
7	I would imagine that most people would learn to use this system very quickly:	2	1
8	I found the system very cumbersome to use:	3	2
9	I felt very confident using the system:	3	2
10	I needed to learn a lot of things before I could get going with this system:	2	3
	Individual SUS Score		50
SEBIS 3			
#	Question	User's Scale Position	SUS Contribution
1	I think that I would like to use this system frequently:	3	2
2	I found the system unnecessarily complex:	2	3
3	I thought the system was easy to use:	3	2
4	I think that I would need the support of a technical person to be able to use this system:	4	1
5	I found the various functions in this system were well integrated:	3	2
6	I thought there was too much inconsistency in this system:	3	2
7	I would imagine that most people would learn to use this system very quickly:	3	2
8	I found the system very cumbersome to use:	3	2
9	I felt very confident using the system:	4	3
10	I needed to learn a lot of things before I could get going with this system:	2	3
	Individual SUS Score		55

IDP 1			
#	Question	User's Scale Position	SUS Contribution
1	I think that I would like to use this system frequently:	3	2
2	I found the system unnecessarily complex:	4	1
3	I thought the system was easy to use:	3	2
4	I think that I would need the support of a technical person to be able to use this system:	2	3
5	I found the various functions in this system were well integrated:	2	1
6	I thought there was too much inconsistency in this system:	2	3
7	I would imagine that most people would learn to use this system very quickly:	4	3
8	I found the system very cumbersome to use:	2	3
9	I felt very confident using the system:	3	2
10	I needed to learn a lot of things before I could get going with this system:	2	3
	Individual SUS Score		57.5
IDP 2			
#	Question	User's Scale Position	SUS Contribution
1	I think that I would like to use this system frequently:	5	4
2	I found the system unnecessarily complex:	1	4
3	I thought the system was easy to use:	4	3
4	I think that I would need the support of a technical person to be able to use this system:	1	4
5	I found the various functions in this system were well integrated:	4	3
6	I thought there was too much inconsistency in this system:	2	3
7	I would imagine that most people would learn to use this system very quickly:	4	3
8	I found the system very cumbersome to use:	1	4
9	I felt very confident using the system:	5	4
10	I needed to learn a lot of things before I could get going with this system:	1	4
	Individual SUS Score		90
IDP 3			
#	Question	User's Scale Position	SUS Contribution
1	I think that I would like to use this system frequently:	3	2
2	I found the system unnecessarily complex:	3	2
3	I thought the system was easy to use:	4	3
4	I think that I would need the support of a technical person to be able to use this system:	2	3
5	I found the various functions in this system were well integrated:	3	2
6	I thought there was too much inconsistency in this system:	2	3
7	I would imagine that most people would learn to use this system very quickly:	4	3
8	I found the system very cumbersome to use:	1	4
9	I felt very confident using the system:	4	3
10	I needed to learn a lot of things before I could get going with this system:	1	4
	Individual SUS Score		72.5

List of Figures

List of Tables