

# **User and Project Management for the Online Card Sorting Tool Sortit**

Markus Steininger





# User and Project Management for the Online Card Sorting Tool Sortit

Markus Steininger

## **Bachelor's Thesis**

to achieve the university degree of

Bachelor of Science

Bachelor's Degree Programme: Information and Computer Engineering

submitted to

Graz University of Technology

Supervisor

Ao.Univ.-Prof. Dr. Keith Andrews  
Institute of Interactive Systems and Data Science (ISDS)

Graz, 01 May 2018

© Copyright 2018 by Markus Steininger, except as otherwise noted.

This work is placed under a Creative Commons Attribution 4.0 International (CC BY 4.0) licence.





# **Benutzer- und Projektverwaltung für das online Karten-Sortier-Tool Sortit.**

Markus Steininger

## **Bachelorarbeit**

für den akademischen Grad

Bachelor of Science

Bachelorstudium: Information and Computer Engineering

an der

Technischen Universität Graz

Begutachter

Ao.Univ.-Prof. Dr. Keith Andrews  
Institute of Interactive Systems and Data Science (ISDS)

Graz, 01. Mai 2018

Diese Arbeit ist in englischer Sprache verfasst.



## **Statutory Declaration**

*I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The document uploaded to TUGRAZonline is identical to the present thesis.*

## **Eidesstattliche Erklärung**

*Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Dokument ist mit der vorliegenden Arbeit identisch.*

---

Date/Datum

---

Signature/Unterschrift





## **Abstract**

Card sorting helps designers include the perspective of users when constructing an information hierarchy, by asking users to organise concepts or phrases into groups and then name the groups. Conducting a card sort by hand can be tedious, so web applications are being developed to support this process.

Sortit is an online card sorting tool which aims to support the managers and participants of card sorting projects with an intuitive web-based interface. Sortit uses the full-stack web application framework Meteor together with React and redux to produce a responsive web application. This thesis describes the user and project management elements of Sortit and their implementation.



## **Kurzfassung**

Card sorts helfen Designern die Perspektive von Benutzern in die Erstellung von Informationsarchitektur mit einzubeziehen, indem Benutzer aufgefordert werden Konzepte oder Phrasen in Gruppen zu ordnen und diese dann zu benennen. Einen card sort per Hand auszuführen kann sehr mühsam sein. Deshalb werden Webanwendungen entwickelt um den Prozess zu unterstützen.

Sortit ist ein solches online Cardsortingtool, das darauf abzielt Leiter von card sorts mit geeigneten Analysetools und einer intuitiven Benutzeroberfläche zu unterstützen. Sortit benutzt die Full-Stack-Webanwendung Meteor zusammen mit React und redux um eine responsive online Benutzeroberfläche zu produzieren. Diese Arbeit beschreibt den Benutzer- und Projektmanagement Teil von Sortit und ihre Implementierung.



# Contents

<b>Contents</b>	<b>ii</b>
<b>List of Figures</b>	<b>iii</b>
<b>List of Tables</b>	<b>v</b>
<b>List of Listings</b>	<b>vii</b>
<b>Acknowledgements</b>	<b>ix</b>
<b>Credits</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Card Sorting</b>	<b>3</b>
2.1 Open Card Sorting . . . . .	3
2.2 Closed Card Sorting . . . . .	3
2.3 Tree Testing . . . . .	4
2.4 Conducting a Card Sort . . . . .	4
<b>3 Building Web Applications</b>	<b>5</b>
3.1 Meteor . . . . .	5
3.2 MongoDB . . . . .	6
3.3 React.js . . . . .	6
3.4 Redux . . . . .	6
3.5 Atom Editor . . . . .	7
3.6 ES6 . . . . .	7
3.7 ESLint . . . . .	7
3.8 SASS . . . . .	8
3.9 BEM . . . . .	8
3.10 Reactstrap . . . . .	8
<b>4 Sortit Project</b>	<b>9</b>
4.1 Software Architecture . . . . .	9
4.2 Project Structure . . . . .	12

<b>5</b>	<b>User Management</b>	<b>13</b>
5.1	Access Levels . . . . .	13
5.1.1	Unrestricted Area . . . . .	13
5.1.2	User . . . . .	13
5.1.3	Project Manager . . . . .	13
5.1.4	Admin . . . . .	15
5.2	Setting Up a New User . . . . .	18
5.3	Forgotten Password . . . . .	18
5.4	Changing an Email Address . . . . .	18
<b>6</b>	<b>Future Work</b>	<b>21</b>
6.1	Analyse Data . . . . .	21
6.2	Send Mail . . . . .	21
6.3	Deployment . . . . .	21
6.4	Design System . . . . .	21
<b>7</b>	<b>Concluding Remarks</b>	<b>23</b>
<b>A</b>	<b>Email Setup</b>	<b>25</b>
	<b>Bibliography</b>	<b>27</b>

# List of Figures

3.1	ES6 Implementation Status . . . . .	7
4.1	Sortit Application Stack . . . . .	10
4.2	Sortit Software Architecture . . . . .	11
5.1	Login Page . . . . .	14
5.2	Navigation Bar . . . . .	14
5.3	Projects List . . . . .	14
5.4	Project in Detailed View . . . . .	15
5.5	Manage User Page . . . . .	16
5.6	Manage User Page . . . . .	17
5.7	Login/CreateAccount Page . . . . .	18





# List of Tables

2.1 Properties of Card Sorts . . . . .	4
--	---



# List of Listings

4.1 Sortit Project Structure . . . . . 12



# Acknowledgements

I wish to thank my supervisor, Keith Andrews, for his patience and guidance during the creation of this thesis. I dedicate this thesis to my parents, on whose support I can always count. Special mention goes to all my friends, for patiently proofreading and correcting my English.

Markus Steininger  
Graz, Austria, May 2018



# Credits

I would like to thank the following individuals and organisations for permission to use their material:

- The thesis was written using Keith Andrews' skeleton thesis [Andrews, 2017].
- Figures 4.1 and 4.2 are taken from Ljajić's thesis on the same project and used with kind permission [Ljajić, 2016].
- The Login and Create Account forms (seen in Figures 5.1 5.6 and 5.7) were created by Proniushkin [2015] and published under MIT licence.
- Figure 3.1 was taken from a table generated using the software written by Zaytsev [2017] and published under MIT licence.





# Chapter 1

## Introduction

This thesis describes the online card sorting tool Sortit, and in particular its user management. Chapter 2 describes card sorting in the context of information architecture. Chapter 3 describes the techniques used to build Sortit. Chapter 4 describes the architecture of Sortit and Chapter 5 explains its user management. Chapter 6 discusses possible future work on the project. In Appendix A, the current setup of the email sending process is explained.



## Chapter 2

# Card Sorting

*“ The card sorting method is used to generate information about the associations and grouping of specific data items. Participants in a card sort are asked to organise individual, unsorted items into groups and may, depending on the technique, also provide labels for these groups. ”*

[ UXPA [2009] ]

The aim of card sorting is to involve the end-user in the construction of an information hierarchy. Participants are asked to group items (concepts, words, or phrases) which are traditionally written on physical cards. Card sorting helps construct an information hierarchy in a bottom-up manner, typically one level at a time. There are multiple types of card sorting, which are explained below. Spencer [2009] explains the card sorting technique in detail.

### 2.1 Open Card Sorting

In open card sorting (often simply called card sorting), each user is asked to collect items into groups and then name the groups. Sometimes, the use of subgroups is allowed or even encouraged, but is more often not allowed. The process is repeated with a number of users. For initial exploration of the concept space, five to seven users are probably sufficient. For statistical analysis, 30 to 50 or more users might be required.

When analysing the results of a card sort, Spencer [2009] suggests either exploratory or statistical analysis methods. The first involves a small number of users to gain an initial insight into the concept space and to test assumptions, since participants may find groupings which the creator of the study did not think of. Statistical analysis methods can be applied to the results gained from a larger number of users. The first step in analysis is to group the sorts according to the organising principle used by each user (their “mindset”). For example, some users will group products according to their supermarket shelf location, whereas other users might group them by region of origin. After that, various tools and techniques are used to interpret the results. Some of these can be found in Spencer [2009]. In open card sorting, it is important that each user understands all of the concepts to be sorted.

### 2.2 Closed Card Sorting

In closed card sorting, users are asked to place concepts into predetermined categories, i.e. groups where the group names have already been assigned. Like in open card sorting, users must understand each

Variation	Supervised	Unsupervised	Remote	Face-to-Face	Paper	Electronic	Single User	Collaborative
Online Card Sort		X	X			X	X	
In Person	X			X	X		X	X

**Table 2.1:** Likely properties of card sorts.

concept before starting. A closed card sorting study checks where users would place concepts in a pre-existing information hierarchy.

In a hybrid card sort, users are given some predefined groups, but are allowed to extend (or rename) the groups as needed.

## 2.3 Tree Testing

Tree testing (also called inverse or reverse card sorting) is used to evaluate an existing or proposed information hierarchy. A user is asked to find a specific concept in a given information hierarchy. The information hierarchy and the set of concepts are usually not known to the user in advance.

## 2.4 Conducting a Card Sort

When conducting a card sort, some decisions need to be made regarding the execution of the study. Some of the options include:

- supervised vs. unsupervised
- remote vs. face-to-face
- paper vs. electronic
- single user vs. collaborative

While many combinations are possible, some are more likely than others. Table 2.1 shows two typical setups and their likely options. The online card sorting tool Sortit is intended to support unsupervised remote electronic single-user open card sorting.

## Chapter 3

# Building Web Applications

“Undoubtedly, personal computing has revolutionised the way we live and work today. The Web has further revolutionised the way we use applications. When it was first introduced, the Internet was designed to present information in the form of documents. Later, JavaScript was added, which has been the key ingredient for the innovation we see on the Web today. [...] Because of the importance of the Web and the pivotal role that JavaScript plays in web development, you can find a solution for most technical problems in some open source JavaScript project. Node.js allows you to use all these innovative JavaScript projects on the server the same as on the client browser.”

[Syed, 2014]

The world of web application development is in constant change. What was the norm yesterday might be outdated today. Especially since node.js brought its modularity to the server-side, many new concepts for full-stack development appeared, gained fame, and disappeared again.

Here, only the tools and techniques actually used in the Sortit project are described, since any in-depth comparison would go beyond the scope of the thesis. The decisions as to which technologies to base the project on were taken by project leader Keith Andrews and the first developer Haris Ljajić [Ljajić, 2016]. For the rest of this chapter, it is assumed that the reader is familiar with the three core technologies of web development HTML [Keith and Andrew, 2016; W3C, 2018], CSS [Cederholm, 2015] and JavaScript [Marquis, 2015], the run-time environment Node.js [Cantelon et al., 2013], and the Model-View-Controller (MVC) design pattern [Wikipedia, 2017].

### 3.1 Meteor

Meteor [MDG, 2018a] is a free and open-source full-stack JavaScript web framework, distributed under an MIT licence [MDG, 2017c]. Full-stack means that both frontend and backend are written in the same language, JavaScript, which helps to close the gap between frontend and backend development:

“JavaScript is the de facto language of the Internet. Web servers and server side frameworks are written in many different languages but client side programs are (almost) exclusively JavaScript. This means if you develop an app using a framework written in a language other than JavaScript, you have to learn that language and JavaScript. Switching languages adds an additional cognitive load that is difficult to manage for both beginners and seasoned developers. The solution is simple, use JavaScript for both the client and server.”

[Robinson et al., 2015]

In the popular MVC pattern, Meteor is responsible for both the backend (Model) and frontend (Controller) logic and all the manipulations and data flow in between. Combined with all best practices recommended by Meteor, this enables a web application to be fully reactive. Staltz [2018] explains the term reactive for programmers. In short, it is a programming paradigm which allows web applications to

instantly react to user input. Moreover, Meteor supports fast and automatic rebuilds of the whole project. In fact, only the changed parts are updated and refreshed in the browser which allows for quick and easy development. [DeBergalis, 2012] The best way to get started is to follow one of the well-documented tutorials [MDG, 2017e], user guide [MDG, 2017b] and documentation [MDG, 2017a].

To be able to provide everything needed to build a full web application, Meteor requires only two additional tools: a database and a library for building user interfaces (View). As a database, MongoDB is integrated into Meteor and will be described in Section 3.2. For composing the User Interface, Meteor lets each development team decide which tool to use as, there are many different ones out there. For this project, React.js was chosen, which is described in Section 3.3.

As the demand for fast built high-quality web applications rises, there are numerous other full-stack web frameworks available to support and accelerate the build process, including MEAN [Linnovate, 2017], Sails [TSC, 2017], and DerbyJS [Smith, 2017].

## 3.2 MongoDB

Nowadays, two kinds of database are in general use: Relational databases typically using SQL, such as Oracle [Oracle, 2017b] and MySQL [Oracle, 2017a] and schemaless databases, often called NoSQL databases, such as MongoDB and Cassandra [ASF, 2017].

MongoDB [MongoDB, 2018] is a NoSQL database which stores data in a manner similar to JavaScript Object Notation (JSON) [ECMA, 2017]. It is a free, scalable and schemaless database. In combination with Meteor, it allows developers to store data without having to define a database schema first. This saves a great deal of time, as requirements often change during development. However, older entries may become invalid in the project, while still being considered correct by the database and have to be migrated manually. Meteor's subscriptions API [MDG, 2017d] restricts what an end-user is able to see and obtain from the database. Developers unfamiliar with NoSQL databases, should read the MongoDB documentation [MongoDB, 2017].

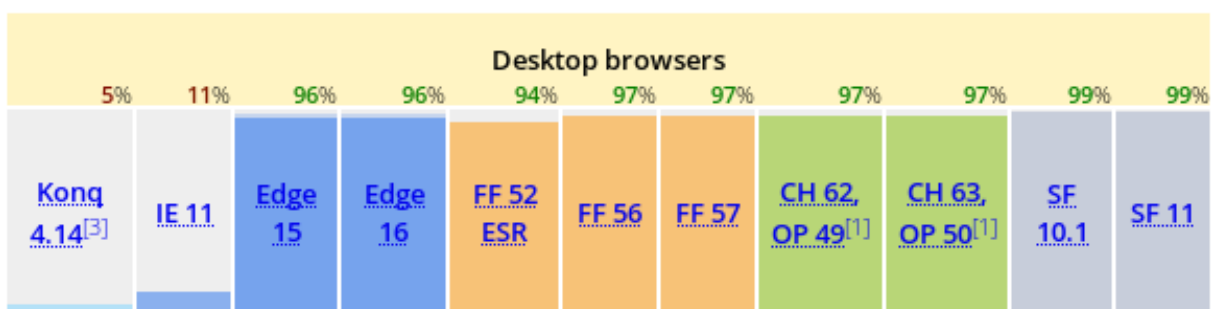
## 3.3 React.js

React.js (or React for short) [Facebook, 2018] is a JavaScript library developed and maintained by Facebook. It is a framework for building scalable user interfaces (UI), which are fast responding and produce reusable code. Once a component has been defined, it can be rendered and reused anywhere in the project. One of the biggest advantages of React is its virtual Document Object Model (DOM) [Rotolo, 2015]. When the recommended top-down data flow is used, React automatically re-renders only the smallest possible part of the UI after a data change has occurred. This ensures a fast responding web application, although the programmer may assume that the whole page is re-rendered on every change of data.

While the virtual DOM was first developed for React, many frameworks offer similar optimisation and performance, including Vue [Vue, 2017] and AngularJS [Google, 2017]. A good introduction to React is the video series of LearnCode.academy on YouTube. [LCA, 2016]

## 3.4 Redux

Redux is more of a programming pattern than a tool. It was derived from Facebook's flux [Facebook, 2017] with which it shares many similarities. Redux as a pattern replaces the MVC architecture. MVC has a clear distinction between the three components Model, View, and Controller and a bidirectional data flow between them. To make an application reactive, it is necessary to keep track of every change to



**Figure 3.1:** Implementation status of ES6 in modern browsers. [Image extracted from Zaytsev [2017] and used under the terms of an MIT licence.]

the data regardless of which component is responsible for the change and to centralise the current state. Redux (like Flux), uses a centralised state store. In Redux, state can only be modified via actions, which by convention are defined in one single file. For a more detailed explanation of Redux, the reader is directed to the Redux documentation [Abramov, 2017] or an excellent article from Smashing Magazine, which also compares Redux to Flux and the MVC pattern [Bachuk, 2016].

### 3.5 Atom Editor

The Atom editor [GitHub, 2018] was the text editor of choice for the Sortit project, since it is lightweight, free, open-source and built with node, which makes it adjustable to the point where one might call it hackable. Any other Integrated Development Environment (IDE) or text editing program can also be used. Popular examples include Eclipse [EF, 2017], Visual Studio Code [Microsoft, 2018], and WebStorm [JetBrains, 2017].

### 3.6 ES6

The European Computer Manufacturers Association (ECMA) is the body which standardises JavaScript specifications under the name ECMAScript (ES). Compilers, desktop and mobile browsers, and servers (for example Node.js running on a server) generally follow these specifications for their JavaScript implementations. In 2015, ECMA released its 6<sup>th</sup> specification called ES6, which is already implemented by most modern web browsers, as can be seen in Figure 3.1. For a full list of features in ES6 see Engelschall [2017], for a summary of the most important features see LCA [2015].

While ES6 does have some advantages, it is still not implemented fully in every browser there is. ECMA already published its 7<sup>th</sup> and 8<sup>th</sup> versions of ECMAScript. To use the newest features during development, the code can be transpiled with babel [McKenzie, 2018] into an earlier version of ECMAScript to run on all platforms. Other programming languages which transcompile to JavaScript include CoffeeScript [Ashkenas, 2017] and TypeScript [Microsoft, 2017].

### 3.7 ESLint

ESLint is a “pluggable JavaScript linter” [JSF, 2017]. In the Sortit project, ESLint is used to check the ES6 code for inconsistent or potentially dangerous usage, best practice, and some styling issues. The ‘linter-eslint’ package of the Atom editor is used, so that linting is performed directly in the editor. The rules enforced by ESLint [JSF, 2018] can be considered an implicit form of coding standard.

### 3.8 SASS

Syntactically Awesome Style Sheets (SASS) [Catlin et al., 2018] is a preprocessor which adds some useful programming techniques to CSS, including variables, nesting, import of files, and inheritance. Other popular CSS preprocessors include LESS [Sellier, 2019] and Stylus [Holowaychuk, 2018].

### 3.9 BEM

The Sortit project uses the Block-Element-Modifier (BEM) methodology as a naming convention for CSS classes. BEM provides a simple and easy to remember technique of naming classes. The convention follows this simple syntax: `Block__Element--Modifier`, where a block stands for a single independent entity which can stand on its own, like a form for example. An element is part of a block which has no standalone meaning, for instance an input field. Modifiers describe different states of the entity, like being `onFocus`. To write a CSS class for a selected input field in a form, one would name the class `form__input-field--focus`. These principles are quickly introduced and implemented and keep even large projects maintainable. Strukchinsky and Starkov [2017] provide a good introduction to BEM. Other popular naming conventions for CSS classes include OOCSS [Sullivan, 2013], ACSS [Yahoo, 2017], and SMACSS [Snook, 2017].

### 3.10 Reactstrap

Bootstrap is a popular UI component library [Otto and Thornton, 2017]. Bootstrap v4 supports the responsive web design paradigm. Reactstrap [Hernandez et al., 2018] integrates Bootstrap v4 components into React.

Other frontend web frameworks include Material Design [CEA, 2017; Google, 2018] and Foundation [ZURB, 2017]. While these frameworks help build quality graphical UIs, it is always possible to write one's own styles and functionality directly with HTML, CSS, and JavaScript.



## Chapter 4

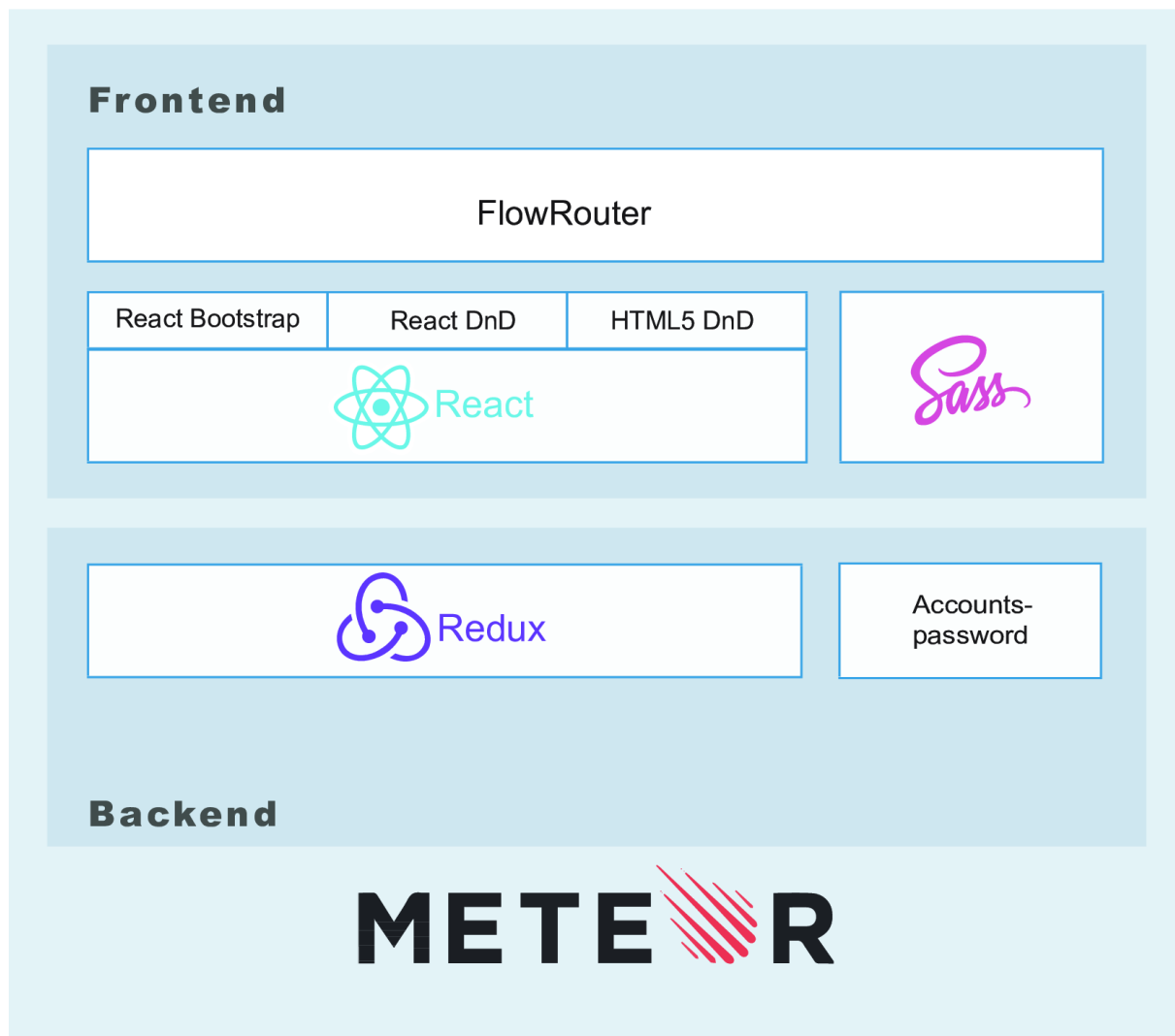
# Sortit Project

Sortit is a free and open-source online card sorting tool designed to assist conducting card sorts. It is written in JavaScript, CSS and HTML using the tools described in Chapter 3. Sortit uses an online, remote, electronic, unsupervised approach, as described in Section 2.4. It supports card sorting studies by making it easy to set up new card sorts, reach many sorters, and automate the analysis of card sorts, although analysis tools have not been implemented at the time of writing this thesis.

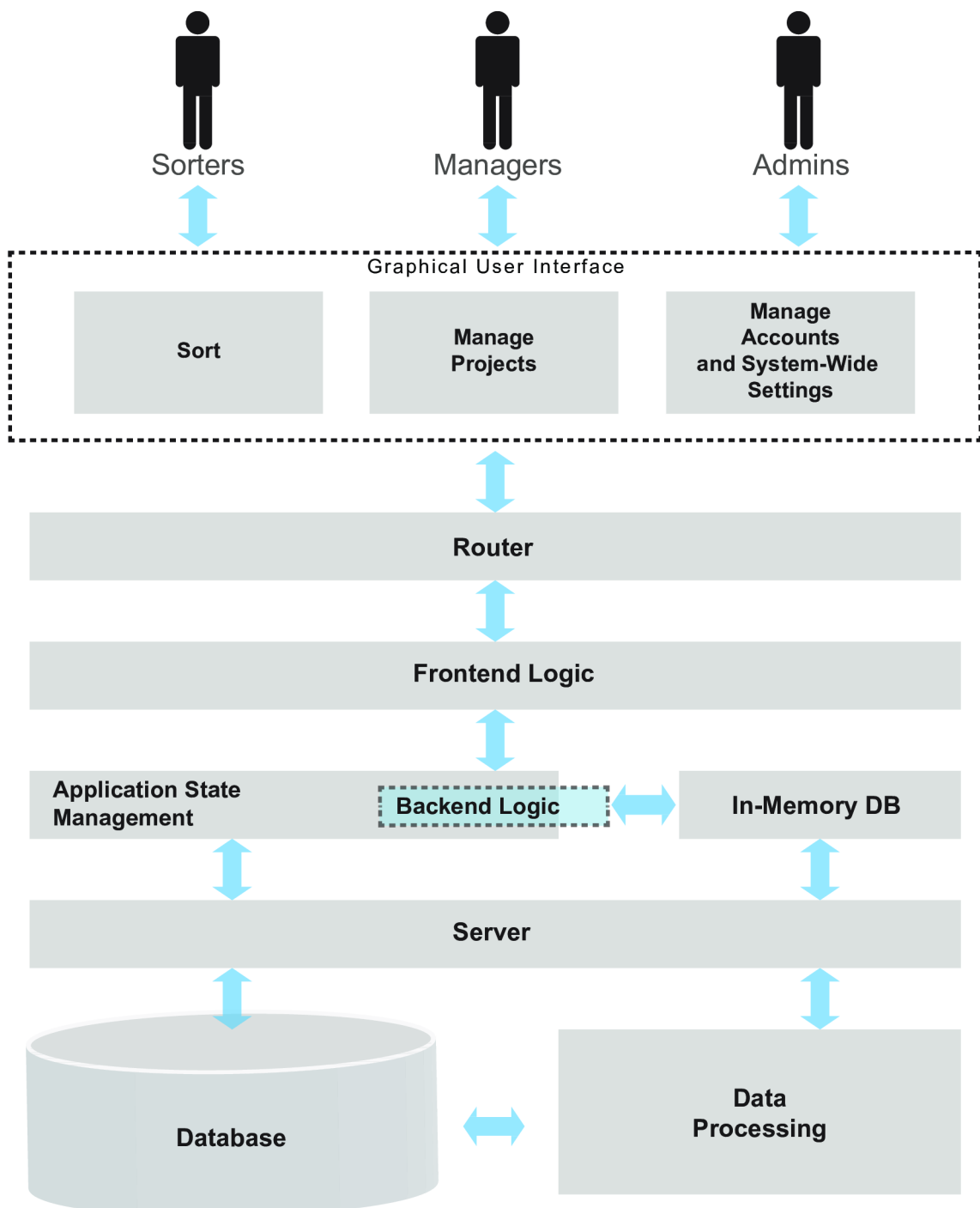
The user interface (UI) of Sortit is divided into four main parts requiring different access levels, which are described in more detail in Chapter 5.

### 4.1 Software Architecture

Ljajić [2016] was responsible for building the backend and designing the general software architecture of Sortit. Since the architecture has not changed, one can read about it in detail in his thesis. Figure 4.1 and Figure 4.2 provide an overview of the application stack and software architecture.



**Figure 4.1:** The application stack of Sortit. [Image taken from Ljajić [2016] and used with kind permission.]



**Figure 4.2:** The software architecture of Sortit. [Image taken from Ljajić [2016] and used with kind permission.]

```

1 sortit
2 |-- doc                # documentation and todos
3 |-- sample-data       # sample data to test basic functionality
4 '-- src                # source code
5   |-- client          # code executed on client side (browser)
6   |   '-- stylesheets # corresponding style files
7   |-- imports         # all static data that can be reused as components
8   |   |-- api         # files concerning the API
9   |   |   |-- redux
10  |   |   |-- server
11  |   |   |-- types
12  |   |   '-- util
13  |   '-- ui          # files concerning the user interface
14  |       |-- components # reuseable components
15  |       |   |-- login
16  |       |   |-- projects
17  |       |   '-- sort
18  |       '-- pages   # pages composing application front-end
19 |-- node_modules     # npm packages (not contained in git project)
20 |-- public           #
21 |   '-- design       # graphic used in the project
22 '-- server           # code only executed on server

```

**Listing 4.1:** The new project structure of Sortit in accordance with Meteor’s recommendation [MDG, 2018b].

## 4.2 Project Structure

The project structure of Sortit was adapted to follow Meteor’s current recommendations [MDG, 2018b]. The current folder structure is shown in Listing 4.1. The source code is divided into code which is executed only on the server, only on the client, or on both machines. This structure helps programmers organise the project with both frontend and backend code part of the same project repository.

## Chapter 5

# User Management

Sortit has a user management system based on Meteor’s built-in account management package [MDG, 2018d], which allows users of different access levels to use different parts of the application. For the Login and Create Account form, an external skeleton [Proniushkin, 2015] is used to include features like email verification and a password strength check.

### 5.1 Access Levels

Sortit can be roughly divided into four parts, which correspond to the four different access levels given to users of the application: unrestricted, User, Project Manager, and Admin.

#### 5.1.1 Unrestricted Area

The unrestricted area is accessible without an account. It contains the Login page shown in Figure 5.1 and an About page with general information about Sortit. Any attempt to access a restricted area is blocked and the visitor is redirected to the login page. If a user forgets his or her login credentials, a link is provided to reset the password. The workflow is described in detail in Section 5.3. Before being able to enter any of the other areas, a user has to verify his or her email address by following a link, which is sent to them automatically.

#### 5.1.2 User

Every person participating in a card sort via Sortit needs to have an account. Therefore, users are required to create an account when first receiving a link to a sort. Users are able to change their login credentials and email address. After a change of email address, the login token becomes invalid, which means that a new verification process needs to be initiated (see Section 5.4).

#### 5.1.3 Project Manager

The Project Manager role is a role which is added to a user account. It provides the functionality needed by the manager of a card sorting study. The additional pages can be accessed via the navigation bar illustrated in Figure 5.2. The default Project Manager view shows all projects belonging to the currently logged in Project Manager, as can be seen in Figure 5.3. Each project contains a small summary, such as the one shown in Figure 5.4, as well as a detailed view with further options. The detailed view is displayed on a separate page, which is described in more detail by Stefan [2018]. Currently, it is necessary to manually send the invitation link to each participant in a card sorting study.

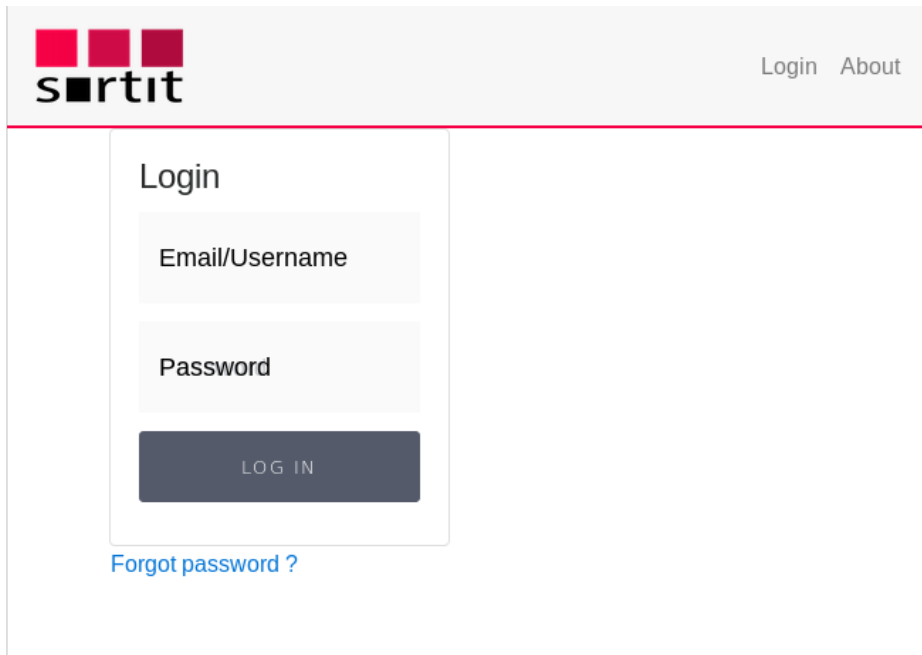


Figure 5.1: Sortit's Login page.

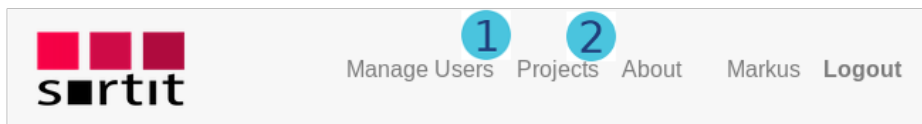
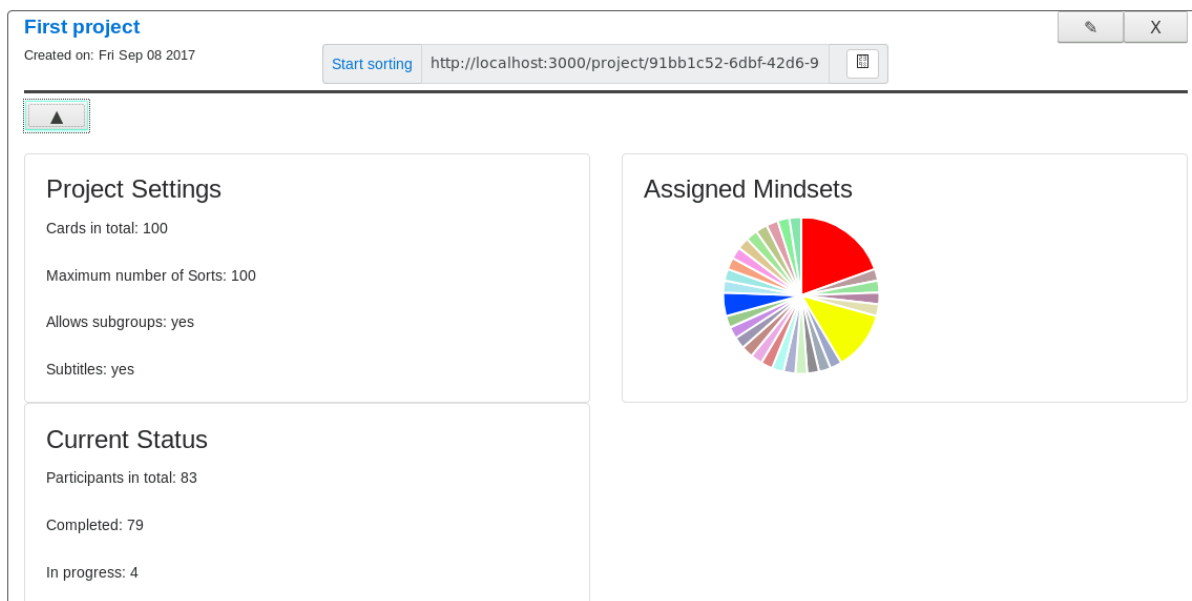


Figure 5.2: Sortit's navigation bar. Admins see the additional menu item Manage Users ①. Project Managers see the additional menu item Projects ②.



Figure 5.3: List of projects belonging to a project manager.

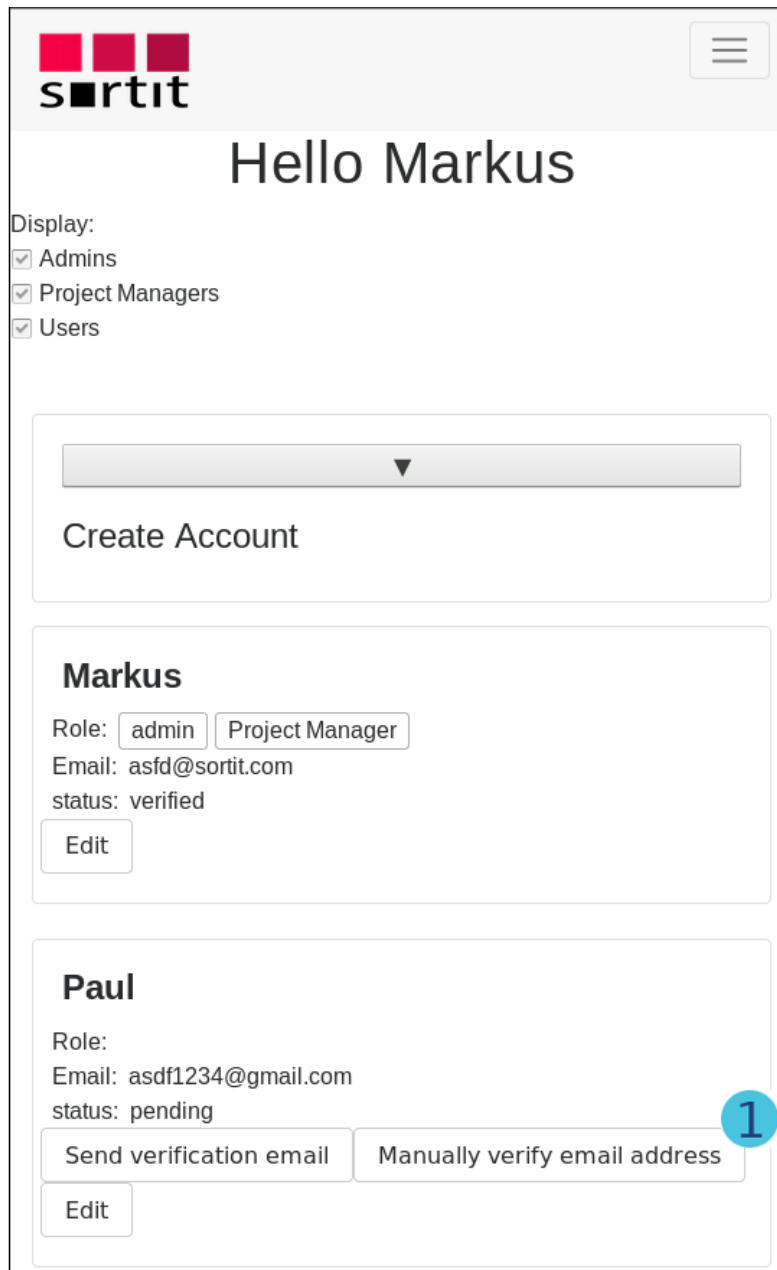


**Figure 5.4:** Clicking on the arrow opens up the project summary.

#### 5.1.4 Admin

The Admin role grants a user special permissions. An Admin has an additional page which can be accessed via the navigation bar (see Figure 5.2). On this page all user entities are listed and can be edited, deleted or new ones created, as shown in Figures 5.5 and 5.6. Each time a new user is created, it triggers an automatic email to verify the account. Additionally, it is possible to send a verification email manually.

Whenever Sortit is started with an empty user collection in its database, an Admin account is created. Its login credentials are hard-coded in the code. It is therefore of utmost importance to change the default admin password once the project goes into production. Moreover, in order not to lose their access rights by accident, Admins are prohibited from altering their own user entity. To alter an Admin account another one must be created first. If a user having the role Project Manager is deleted, it is possible to transfer their projects to the currently logged in Admin account.



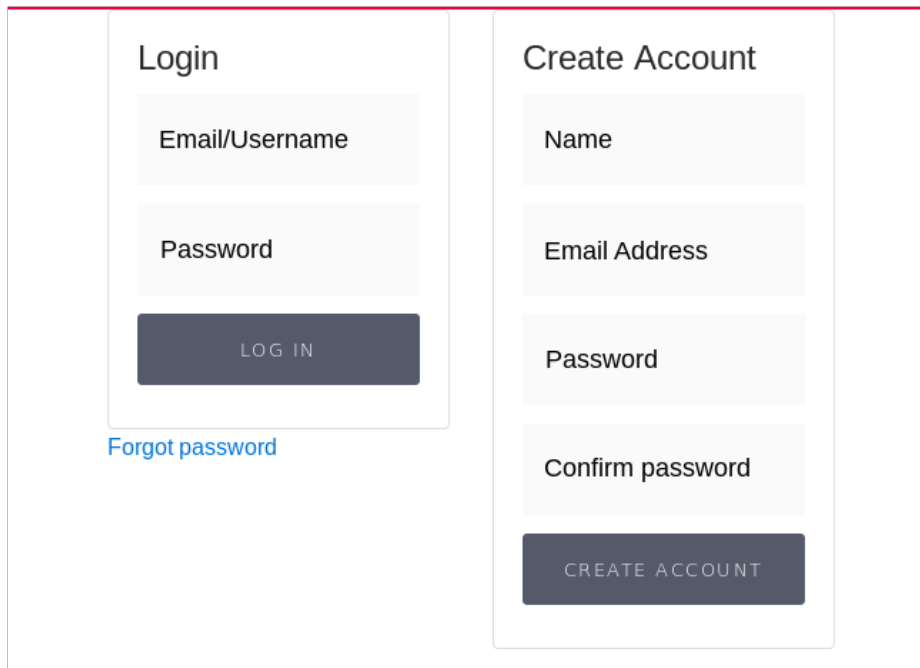
The screenshot shows the Sortit user management interface. At the top left is the Sortit logo (three red squares above the text 'sortit'). At the top right is a hamburger menu icon. Below the logo, the text 'Hello Markus' is displayed. Underneath, there is a 'Display:' section with three checked checkboxes: 'Admins', 'Project Managers', and 'Users'. Below this is a search bar with a downward arrow and a 'Create Account' button. The main content area displays two user profiles. The first profile is for 'Markus', with a role of 'admin' and 'Project Manager', email 'asfd@sortit.com', and status 'verified'. There is an 'Edit' button below his profile. The second profile is for 'Paul', with a role of 'admin', email 'asdf1234@gmail.com', and status 'pending'. There are two buttons below his profile: 'Send verification email' and 'Manually verify email address'. A blue circle with the number '1' is placed over the 'Manually verify email address' button. There is also an 'Edit' button below Paul's profile.

**Figure 5.5:** List of users as seen by the Admin. User “Paul” currently has an unverified email address. ① shows the button to manually verify an email address, as described in Section 5.4.



The image shows a web interface for user management. At the top left is the 'sortit' logo with three red squares above it. To the right is a hamburger menu icon. Below the logo, the text 'Hello Markus' is displayed. Underneath, there is a 'Display:' section with three checked checkboxes: 'Admins', 'Project Managers', and 'Users'. Below this is a 'Create Account' form with three input fields: 'Username', 'Email Address', and 'Password'. A dark grey button labeled 'CREATE ACCOUNT' is at the bottom of this form. A blue circle with the number '1' is positioned to the right of the top of the 'Create Account' form. Below the 'Create Account' form is a user profile for 'Markus'. The name 'Markus' is in a large input field. Below it, the 'Role:' section contains two buttons: 'admin | x' and 'Project Manager | x', followed by an 'Add Role..' button with a dropdown arrow. Below the roles is an 'Email:' field containing 'asfd@sortit.com'. Below the email is the text 'status: verified'. At the bottom of the profile are two buttons: 'Save' and 'Delete'. A blue circle with the number '2' is positioned to the right of the top of the user profile section.

**Figure 5.6:** ① shows the panel to create a new user, which is adapted from Proniushkin [2015]. ② shows user “Markus” in edit mode.



The image shows a web form titled 'Invitation page' with two main sections: 'Login' and 'Create Account'. The 'Login' section contains a text input field for 'Email/Username', a text input field for 'Password', a dark blue button labeled 'LOG IN', and a blue link labeled 'Forgot password'. The 'Create Account' section contains a text input field for 'Name', a text input field for 'Email Address', a text input field for 'Password', a text input field for 'Confirm password', and a dark blue button labeled 'CREATE ACCOUNT'.

**Figure 5.7:** The Invitation page which is shown to a user after following an invitation link to a card sort.

## 5.2 Setting Up a New User

Whenever participants receive an invitation link to a card sort, they are sent to the Invitation page shown in Figure 5.7, where they can either log in or create a new account. In addition, an Admin has the ability to add a new user manually. After creation, an automatic email is sent with a verification link, which gives users access and prompts them to change their password. Currently, Sortit uses Mailgun (see Appendix A) to send emails. Emails can only be sent to email addresses which have been pre-registered with Mailgun. This email setup is sufficient during the development of Sortit, but will later need to be replaced with a more sustainable alternative such as sendmail [proofpoint, 2018]. Whenever a link to a card sort is sent to new participants, they will be redirected to the Login or Create Account page first.

## 5.3 Forgotten Password

When users forget their password, they can reset the password by following the [Forgot password?](#) link on the Login page (see Figure 5.1), enter their email address, and receive a link they can follow. Internally, this link is intercepted in the `router.js` file and a new random password is generated, which should be changed immediately.

## 5.4 Changing an Email Address

Whenever an email address is changed, which can be done by users themselves, it is necessary to verify the new email address. Internally, a Meteor call `changeAccountEmail` is sent to `meteor.js` which triggers a call to `sendVerificationEmail` to initiate the automatic email send process. Currently, it is possible to

send emails only to registered addresses, as no email server is set up (the current setup is described in Appendix A). Therefore, an Admin has the possibility to manually verify that an email address is correct (see Figure 5.5).



# Chapter 6

## Future Work

The Sortit web application is not yet finished. Numerous features and enhancements have still to be developed. No analysis tools have yet been implemented.

### 6.1 Analyse Data

In future, Sortit will also support Project Managers in the analysis of the collected data. Spencer [2009] explains in detail how to analyse card sorting data.

First, within a mindset, the group names are standardised and the sorts are expressed using standardised group names. Their correlation counts are overview statistics can be calculated. Finally, a co-occurrence matrix allows a best-fit-grouping to be derived for the mindset.

### 6.2 Send Mail

The process of sending verification and invitation emails could be better automated. Appendix A describes the current status. One possible improvement to the invitation email is implementing either the option to send an invitation to all users of Sortit, or Sortit users selected from an interface. An external email list would be a favourable feature as it would offer the possibility to contact people who have not been registered yet.

### 6.3 Deployment

Information about how to deploy Sortit to a production server can be found in the Meteor documentation [MDG, 2018c]. Ljajić [2016] also summarises the necessary steps.

### 6.4 Design System

One internal task is to unify all popup messages, so that the same UI component is used. This component should be modular enough to fit all purposes. Currently, popups are needed in `imports/ui/components/sort/UsernameInput.js`, `imports/ui/components/UserComponent.js`, `imports/ui/pages/App.js`, and `imports/ui/pages/EditProjectPage.js`. Even better, a design system could be employed to standardise and maintain UI components.



## **Chapter 7**

# **Concluding Remarks**

This thesis described the online card sorting tool Sortit, and in particular its user management. Chapter 2 introduced card sorting in the context of information architecture. Chapter 3 described the utilised technologies. Chapter 4 presented the setup of Sortit and Chapter 5 explained its user management. Chapter 6 discussed possible future work on the project.





# Appendix A

## Email Setup

To be able to test Sortit's email functionality, a basic email setup is required. It was decided to use Mailgun [MT, 2018] internally, since it is free and easy to handle. Mailgun allows the sending of emails via their simple mail transfer protocol (SMTP) server and provides enough reputation to guarantee that the emails will not be blocked by other SMTP servers along the way. However, in the free plan, it is required to register each email address to Mailgun before being able to send emails to them. The following email addresses are currently registered with Mailgun:

### 1. GMX Accounts

- address: sortit.iicm@gmx.at
- address: user1.sortit@gmx.at
- address: user2.sortit@gmx.at

### 2. Mailgun Account

- user: sortit.iicm@gmx.at
- backup telephone number: 0650-3868484

### 3. GMAIL Account

- address: sortit.iicm@gmail.com
- backupemail: sortit.iicm@gmx.at

Since this thesis will be available publicly, no passwords are given here. All files regarding the setup and configuration of the email templates in the project are in the server folder.

To add a new email address to Mailgun, log in, scroll to the bottom of the list and click `Authorized Recipients`. Each new email account has to accept to receive emails from Mailgun.



# Bibliography

- Abramov, Dan [2017]. *Redux Documentation*. 28 Nov 2017. <https://redux.js.org/> (cited on page 7).
- Andrews, Keith [2017]. *Writing a Thesis: Guidelines for Writing a Master's Thesis in Computer Science*. Graz University of Technology, Austria. 18 May 2017. <http://ftp.iicm.edu/pub/keith/thesis/> (cited on page xi).
- ASF [2017]. *Cassandra*. Apache Software Foundation. 16 Dec 2017. <https://cassandra.apache.org/> (cited on page 6).
- Ashkenas, Jeremy [2017]. *CoffeeScript*. 10 Dec 2017. <http://coffeescript.org/> (cited on page 7).
- Bachuk, Alex [2016]. *An Introduction to Redux*. 28 Jun 2016. <https://smashingmagazine.com/2016/06/an-introduction-to-redux/> (cited on page 7).
- Cantelon, Mike, Marc Harter, TJ Holowaychuk, and Nathan Rajlich [2013]. *Node.js in Action*. Manning Publications, 28 Nov 2013. ISBN 1617290572 (cited on page 5).
- Catlin, Hampton, Natalie Weizenbaum, and Chris Eppstein [2018]. *Sass*. 22 Feb 2018. <https://sass-lang.com/> (cited on page 8).
- CEA [2017]. *Material-UI*. Call-Em-All. 16 Dec 2017. <http://material-ui.com/> (cited on page 8).
- Cederholm, Dan [2015]. *CSS3 for Web Designers, 2nd Ed*. A Book Apart, 24 Feb 2015. ISBN 1937557200 (cited on page 5).
- DeBergalis, Matt [2012]. *Hot Code Pushes*. 9 Feb 2012. <https://blog.meteor.com/hot-code-pushes-c3d49c43094b/> (cited on page 6).
- ECMA [2017]. *JSON*. Dec 2017. <http://ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf> (cited on page 6).
- EF [2017]. *Eclipse*. Eclipse Foundation. 16 Dec 2017. <https://eclipse.org/> (cited on page 7).
- Engelschall, Ralf S. [2017]. *ES6 Features*. 28 Nov 2017. <http://es6-features.org/> (cited on page 7).
- Facebook [2017]. *Flux*. 16 Dec 2017. <https://facebook.github.io/flux/> (cited on page 6).
- Facebook [2018]. *React.js*. 11 Feb 2018. <https://reactjs.org/> (cited on page 6).
- GitHub [2018]. *Atom*. 11 Feb 2018. <https://atom.io/> (cited on page 7).
- Google [2017]. *AngularJS*. 16 Dec 2017. <https://angularjs.org/> (cited on page 6).
- Google [2018]. *Material Design*. 11 Feb 2018. <https://material.io/> (cited on page 8).
- Hernandez, Eddy, Chris Burrell, and Evan Sharp [2018]. *Reactstrap*. 11 Feb 2018. <https://reactstrap.github.io/> (cited on page 8).
- Holowaychuk, TJ [2018]. *Stylus*. 2 Mar 2018. <http://stylus-lang.com/> (cited on page 8).

- JetBrains [2017]. *WebStorm*. 16 Dec 2017. <https://jetbrains.com/webstorm/> (cited on page 7).
- JSF [2017]. *ESLint Homepage*. JS Foundation. 28 Nov 2017. <https://eslint.org/> (cited on page 7).
- JSF [2018]. *ESLint Rules*. JS Foundation. 11 Feb 2018. <https://eslint.org/docs/rules/> (cited on page 7).
- Keith, Jeremy and Rachel Andrew [2016]. *HTML5 for Web Designers, 2nd Ed*. A Book Apart, 17 Feb 2016. ISBN 1937557243 (cited on page 5).
- LCA [2015]. *Javascript ES6 Cheatsheet - the best of JS ES6*. 9 Sep 2015. <https://youtu.be/AfWY08t7ed4/> (cited on page 7).
- LCA [2016]. *React JS Tutorials*. 2 Feb 2016. <https://youtu.be/MhkGQAoc7bc/> (cited on page 6).
- Linnovate [2017]. *Mean.io*. 16 Dec 2017. <http://mean.io/> (cited on page 6).
- Ljajić, Haris [2016]. “sortit: Web Card Sorting Backend”. Master’s Thesis. Graz University of Technology, 23 Aug 2016. <http://ftp.iicm.tugraz.at/pub/theses/hljajic-2016-msc.pdf> (cited on pages xi, 5, 9–11, 21).
- Marquis, Mat [2015]. *JavaScript for Web Designers*. A Book Apart, 28 Sep 2015. ISBN 1937557464 (cited on page 5).
- McKenzie, Sebastian [2018]. *Babel is a JavaScript compiler*. 14 Feb 2018. <https://babeljs.io/> (cited on page 7).
- MDG [2017a]. *Meteor Documentation*. Meteor Development Group. 28 Nov 2017. <https://docs.meteor.com/> (cited on page 6).
- MDG [2017b]. *Meteor Guide*. Meteor Development Group. 28 Nov 2017. <https://guide.meteor.com/> (cited on page 6).
- MDG [2017c]. *Meteor Licence*. Meteor Development Group. 6 Jan 2017. <https://github.com/meteor/meteor/blob/master/LICENSE/> (cited on page 5).
- MDG [2017d]. *Meteor Publish and Subscribe*. Meteor Development Group. 28 Nov 2017. <https://docs.meteor.com/api/pubsub.html> (cited on page 6).
- MDG [2017e]. *Meteor Tutorial*. Meteor Development Group. 28 Nov 2017. <https://meteor.com/tutorials/> (cited on page 6).
- MDG [2018a]. *Meteor*. Meteor Development Group. 11 Feb 2018. <https://meteor.com/> (cited on page 5).
- MDG [2018b]. *Meteor Application Structure*. Meteor Development Group. 11 Feb 2018. <https://guide.meteor.com/structure.html#javascript-structure> (cited on page 12).
- MDG [2018c]. *Meteor Deployment and Monitoring*. Meteor Development Group. 19 Feb 2018. <https://guide.meteor.com/deployment.html> (cited on page 21).
- MDG [2018d]. *Meteor Users and Accounts*. Meteor Development Group. 13 Feb 2018. <https://guide.meteor.com/accounts/> (cited on page 13).
- Microsoft [2017]. *TypeScript*. 16 Dec 2017. <https://www.typescriptlang.org/> (cited on page 7).
- Microsoft [2018]. *Visual Studio Code*. 11 Feb 2018. <https://code.visualstudio.com/> (cited on page 7).
- MongoDB [2017]. *MongoDB Shell Methods*. 28 Nov 2017. <https://docs.mongodb.com/manual/reference/method/> (cited on page 6).
- MongoDB [2018]. *MongoDB*. 11 Feb 2018. <https://mongodb.com/> (cited on page 6).

- MT [2018]. *The Email Service for Developers*. Mailgun Technologies. 20 Feb 2018. <https://mailgun.com/> (cited on page 25).
- Oracle [2017a]. *MySQL*. 16 Dec 2017. <https://mysql.com/> (cited on page 6).
- Oracle [2017b]. *Oracle Database*. 16 Dec 2017. <https://oracle.com/database/index.html> (cited on page 6).
- Otto, Mark and Jacob Thornton [2017]. *Bootstrap*. 11 Feb 2017. <https://getbootstrap.com/> (cited on page 8).
- Proniushkin, Mikhail [2015]. *React Signup Form*. 9 Mar 2015. <https://github.com/mikepro4/react-signup-form/> (cited on pages xi, 13, 17).
- proofpoint [2018]. *Sendmail Open Source*. 1 Mar 2018. <https://proofpoint.com/us/open-source-email-solution/> (cited on page 18).
- Robinson, Josh, Aaron Gray, and David Titarenco [2015]. *Introducing Meteor*. Apress, 2015. ISBN 1430268360 (cited on page 5).
- Rotolo, Jarrod [2015]. *The Virtual DOM vs The DOM*. 9 Mar 2015. <https://revelry.co/the-virtual-dom/> (cited on page 6).
- Sellier, Alexis [2019]. *less*. 1 Mar 2019. <http://lesscss.org/> (cited on page 8).
- Smith, Nate [2017]. *Derby*. 16 Dec 2017. <http://derbyjs.com/> (cited on page 6).
- Snook, Jonathan [2017]. *Scalable and Modular Architecture for CSS*. 16 Dec 2017. <https://smacss.com/> (cited on page 8).
- Spencer, Donna [2009]. *Card Sorting: Designing Usable Categories*. Rosenfeld Media, 2009. ISBN 1933820020. <http://rosenfeldmedia.com/books/card-sorting/> (cited on pages 3, 21).
- Staltz, André [2018]. *The introduction to Reactive Programming you've been missing*. 3 Mar 2018. <https://gist.github.com/staltz/868e7e9bc2a7b8c1f754/> (cited on page 5).
- Stefan, Matthias [2018]. "The Project Management and Sorting Interfaces for Sortit". Project Report at Graz University of Technology. 27 Jan 2018 (cited on page 13).
- Strukchinsky, Vsevolod and Vladimir Starkov [2017]. *BEM - Block Element Modifier*. 28 Nov 2017. <http://getbem.com/> (cited on page 8).
- Sullivan, Nicole [2013]. *Object-Oriented CSS*. 26 Sep 2013. <http://oocss.org/> (cited on page 8).
- Syed, Basarat Ali [2014]. *Beginning Node.js*. Apress, 3 Dec 2014. ISBN 1484201884 (cited on page 5).
- TSC [2017]. *Sails*. The Sails Company. 16 Dec 2017. <https://sailsjs.com/> (cited on page 6).
- UXPA [2009]. *Usability Body of Knowledge*. The User Experience Professionals' Association. Jun 2009. <http://usabilitybok.org/card-sorting> (cited on page 3).
- Vue [2017]. *Vue.js*. 16 Dec 2017. <https://vuejs.org/> (cited on page 6).
- W3C [2018]. *HTML 5.3*. World Wide Web Consortium. 6 Feb 2018. <https://w3.org/TR/html53/> (cited on page 5).
- Wikipedia [2017]. *Model-view-controller*. 16 Dec 2017. <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller> (cited on page 5).
- Yahoo [2017]. *Atomic CSS*. 16 Dec 2017. <https://acss.io/> (cited on page 8).

Zaytsev, Juriy [2017]. *ES Compatibility Table*. 16 Dec 2017. <http://kangax.github.io/compat-table/es6/> (cited on pages xi, 7).

ZURB [2017]. *Foundation*. 16 Dec 2017. <https://foundation.zurb.com/> (cited on page 8).