Technische Universität Braunschweig

Institut für Nachrichtentechnik

Bachelor Thesis

Development of an Administrative Web Frontend for Deep Learning Research

Lukas Güldenhaupt

Matrikelnummer: 4571429

01.03.2017

Technische Universität Braunschweig
Institute for Communications Technology
Schleinitzstraße 22 – 38106 Braunschweig

Prüfer: Prof. Dr.-Ing. Tim Fingscheidt
Betreuer: Samy Elshamy, M.Sc.

Erklärung

Hiermit versichere ich die vorgelegte Bachelor Thesis zum Thema

„Development of an Administrative Web Frontend for Deep Learning Research“

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben.

(Sollte in der zutreffenden Prüfungsordnung eine eidesstattliche Versicherung gefordert sein: „Hiermit versichere ich an Eides statt, dass ich die vorliegende Bachelorarbeit zum Thema „Ihr Thema“ selbstständig und nur mit den angegebenen Quellen und Hilfsmitteln erstellt habe.“)

Braunschweig, den 01.03.2017

_____

Lukas Güldenhaupt

Preface (optional)

A preface is optional and can come here.

Abstract

Keywords:

Contents

Content **Page**

1 Architecture

Since the basic idea of this tool is to give a lasting web frontend for the institute a good choice of what software to use is essential. Therefore, for client and server-side code we chose well maintained and well-known frameworks as there are Meteor as a mostly server-side JavaScript framework and Angular as a frontend JavaScript framework. With this an all in all forward-looking webpage is ensured. In this chapter we evaluate why the chosen software fits our purpose and how they work together. We could build the web server and client completely from scratch but Meteor and Angular provide an overall good structure and a solid base for further development. Furthermore, we chose MongoDB as a database.

1.1 Server-side

As mentioned before Meteor is our chosen framework for the server side. It is an opensource full-stack JavaScript platform for web, mobile and desktop development. The power of this platform is its fast learning curve, its usability for any device and its technology integration. What that means is that without knowing much about webservers you can easily create your own application. Meteor also is known for its compatibility since you can use it independently from the platform, no matter if its web, iOS, Android or desktop. In our case we use it as a webserver but with further development of the website it could be optimized for mobile devices or become an app itself if desired.

A big point for Meteor is that you can share code between server, client and the database, what accelerates the development process enormously. This is what makes our application very reactive. Meteor uses data on the wire, sending not HTML to the client but data which gets rendered directly on client-side. With that provided reactivity the client reflects the true state of the data in real time. In combination with our frontend framework Angular no page reloading is necessary to have the latest data.

Behind the easy-to-use platform lies a NodeJS-server. When deploying Meteor-code it generates a stand-alone Node application. And this is the only dependency it has which means everywhere you have NodeJS installed you can run your meteor application.

1.2 Client-side

On client side we chose the JavaScript framework Angular in version 4, developed and maintained by Google. Angular makes client development across all platforms possible. It grants fast speed and good performance and allows us to extend the template language with our own

written components. Nearly every web IDE supports Angular to give the user syntax-highlighting, code-completion and Angular-specific help. In our case it replaces the Meteor-standard blaze-templates. Meteor and Angular work perfectly together on various platforms, when displaying real time data and keeping the reactivity of our application on a very high level.

1.3 Database

1.4 Additional Software

2 Software Design

2.1 Components

2.1.1 Basic Structure

2.2 Typescript

-difference to javascript

-analogy to java

2.3 Data-handling

2.3.1 Data model

-uml diagram

-explain foreign ids

2.3.2 Data services

-MongoDB queries

2.3.3 Observables

-RXJS

2.4 Authentication

3 Users perspective

3.1 Profile/Login

3.2 Projects

3.3 Configurations

3.3.1 Mappings

3.3.2 Filtering

3.3.3 Result monitoring?

4 Developers Perspective

4.1 Developer Tools

4.2 Setting up a development environment

4.3 Coding

4.3.1 Adding Components

4.3.2 Adding Collections

4.3.3 Working with Observables

4.3.4 Extending Functionality

-Npm packages

4.4 Documentation

4.5 Deployment

A Appendix

B Bibliography

Im aktuellen Dokument sind keine Quellen vorhanden.

C List of Figures

Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.

D List of Tables

Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.

E Abbreviations (optional)