Dennis Groß

apolol92@web.de

Exposee

In diesem Dokument findet man die Projektdokumentation des   
DMarkdown-Editors.

DMarkdown

Ein moderner Markdown-Editor

Inhalt

[Einleitung 1](#_Toc481344800)

[Projektanforderungen 2](#_Toc481344801)

[Projektstruktur 3](#_Toc481344802)

[Design 4](#_Toc481344803)

[Softwarearchitektur 6](#_Toc481344804)

[Einflussfaktoren 6](#_Toc481344805)

[Use-Case-Diagramm 6](#_Toc481344806)

[Klassendiagramm 7](#_Toc481344807)

[Aktivitätsdiagramm 7](#_Toc481344808)

[Softwaretests 9](#_Toc481344809)

[Zusammenfassung und Ausblick 10](#_Toc481344810)

Abbildungsverzeichnis

[Abbildung 1 Zu unterstützende Markdown-Zeichen für die IT-Talents Challenge (https://www.it-talents.de/foerderung/code-competition/code-competition-04-2017) 3](#_Toc481344811)

[Abbildung 2 Projektstruktur 4](#_Toc481344812)

[Abbildung 3 Wireframe des Hauptfensters 5](#_Toc481344813)

[Abbildung 4 DMarkdown-Editor 6](#_Toc481344814)

[Abbildung 5 DMarkdown-Logo 6](#_Toc481344815)

[Abbildung 6 Use-Case-Diagramm 7](#_Toc481344816)

[Abbildung 7 Klassendiagramm 8](#_Toc481344817)

[Abbildung 8 Aktivitätsdiagramm 9](#_Toc481344818)

# Einleitung

IT-Talents ist eine Plattform für IT-Nachwuchskräfte. Für die Förderung dieser Nachwuchskräfte bietet sie mit Partnern aus der Industrie monatliche Code-Competitions an. Im März 2017 wurde die Aufgabe gestellt einen Markdown Parser zu entwickeln.

In dieser Projektdokumentation wird auf die Softwarelösung von Dennis Groß eingegangen. Die Lösung wurde mit JavaScript und dem Node Package Electron realisiert. Da es bereits im Bereich der Markdown-Editoren einige Softwarelösungen gibt war es das Ziel einen einfach zu bedienenden, modernen und plattformübergreifenden Editor zu programmieren.

In der vorliegenden Softwaredokumentation wird zuerst auf die Projektanforderungen eingegangen, dann auf die Projektstruktur, das Design des Markdown Editors, auf die Softwarearchitektur und anschließend auf die Softwaretests. Im letzten Kapitel wird ein Ausblick für mögliche Weiterentwicklungen gegeben und das ganze Projekt zusammengefasst.

# Projektanforderungen

Die primären funktionalen Anforderungen für die Challenge waren:

* Die Eingabe eines mit Markdown strukturierten Dokuments zu erlauben. Wobei die Eingabe über einen Textbereich in einer GUI möglich, aber auch das Einlesen einer Datei über eine GUI oder über die Kommandozeile möglich sein sollte.
* Folgende Markdown-Zeichen sollten unterstützt werden

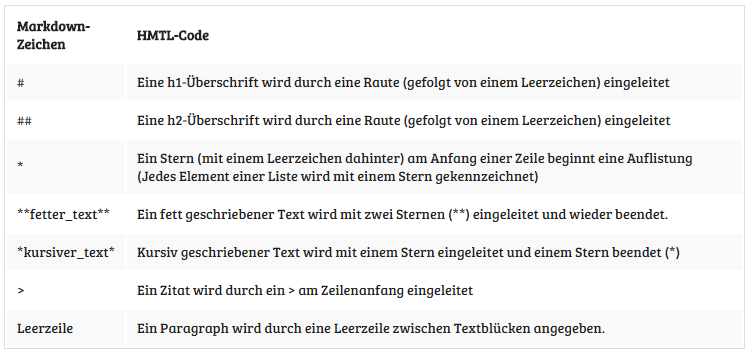


Abbildung 1 Zu unterstützende Markdown-Zeichen für die IT-Talents Challenge  
 (<https://www.it-talents.de/foerderung/code-competition/code-competition-04-2017>)

Weitere funktionale Anforderungen die von Seiten des Entwicklers eingebaut werden sollten:

* Die Markdown-Datei muss exportiert werden können
* Die Markdown-Datei muss gespeichert werden können
* Der Editor soll eine Undo-Redo-Funktion besitzen
* Der Editor muss ein Code-Highlighting besitzen
* Der Editor soll eine Vorschau des Markdown-Codes enthalten
* Die Vorschau soll bei jeder Änderung des Markdown-Codes aktualisiert werden
* Der Editor soll mindestens Deutsch und Englisch unterstützen

Die nicht-funktionalen Anforderungen waren die folgenden:

* Der Editor sollte leicht bedienbar sein
* Der Editor sollte das schnelle Editieren von Markdown-Dateien erlauben
* Der Editor sollte ein responsive Design besitzen

# Projektstruktur

Das Projekt wurde wie in Abbildung 2 gezeigt strukturiert.

|  |  |
| --- | --- |
| Abbildung 2 Projektstruktur | Das Projekt besteht aus vier Teilen. Der Code-Teil beinhaltet jeglichen Programmcode. Der Design-Teil beinhaltet die Designvorlagen des Logos und die Wireframes. Der Documentation-Teil beinhaltet Projektdokumentationen in verschiedenen Sprachen. Node\_modules beinhaltet alle verwendeten Node-Packages.  Der Code-Teil ist wiederrum in einen Assets-, css-, js-, plugins-, spec-Ordner aufgeteilt. Der Einstiegspunkt des Programms ist die main.js-Datei. Hier wird mit Hilfe des Electron-Frameworks eine GUI erstellt und die index.html geladen. Der Name der Applikation und der Haupteinstiegspunkt wird in der package.json-Datei festgelegt.  Der Design-Teil beinhaltet die Wireframes und alle anderen Design-Vorlagen.  Der Documentation-Ordner beinhaltet die Dokumentationen in verschiedenen Sprachen.  Node\_modules hat alle Node-Module als Inhalt. |

# Design

Das graphische Design der Software wurde so gewählt, das die Anwendung aus einem Hauptfenster besteht und der Benutzer beim Aufruf des Programms standardmäßig eine unbenannte Markdown-Datei erhält. Durch die Verwendung der Menüleiste hat er die Möglichkeit andere Markdown-Dateien zu laden, aktuelle Dateien speichern oder zu exportieren. Daneben gibt es in der Menüleiste ein Menüelement das die Undo-Redo-Funktion enthält. Über das Hilfe-Menüelement bekommt der Benutzer Hilfe in seiner jeweiligen Sprache. Im Hauptbereich des Hauptfensters erhält der Benutzer auf der linken Seite die Möglichkeit seinen Markdown-Code einzugeben und auf der rechten Seite eine Vorschau zu erhalten. In der Fußzeile wird die aktuelle Codezeile und Codespalte angegeben.

Die Design-Vorlage als Wireframe sieht man in Abbildung 3.

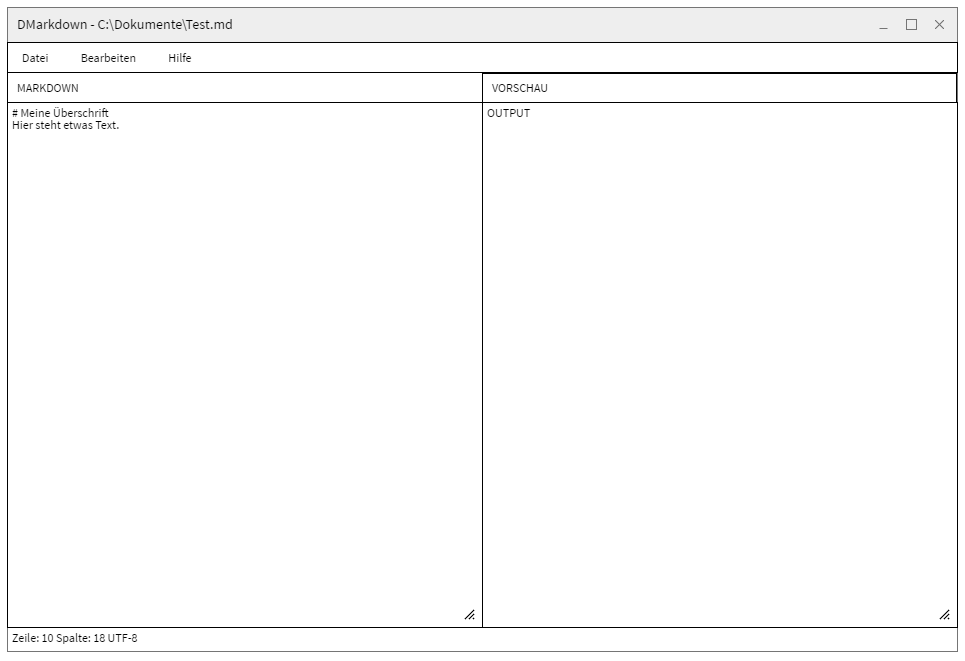


Abbildung 3 Wireframe des Hauptfensters

Die graphische Umsetzung des Wireframes sieht man in Abbildung 4. Die Farben der Anwendung wurden so gewählt, da der Entwickler hiermit sehr zufrieden war.

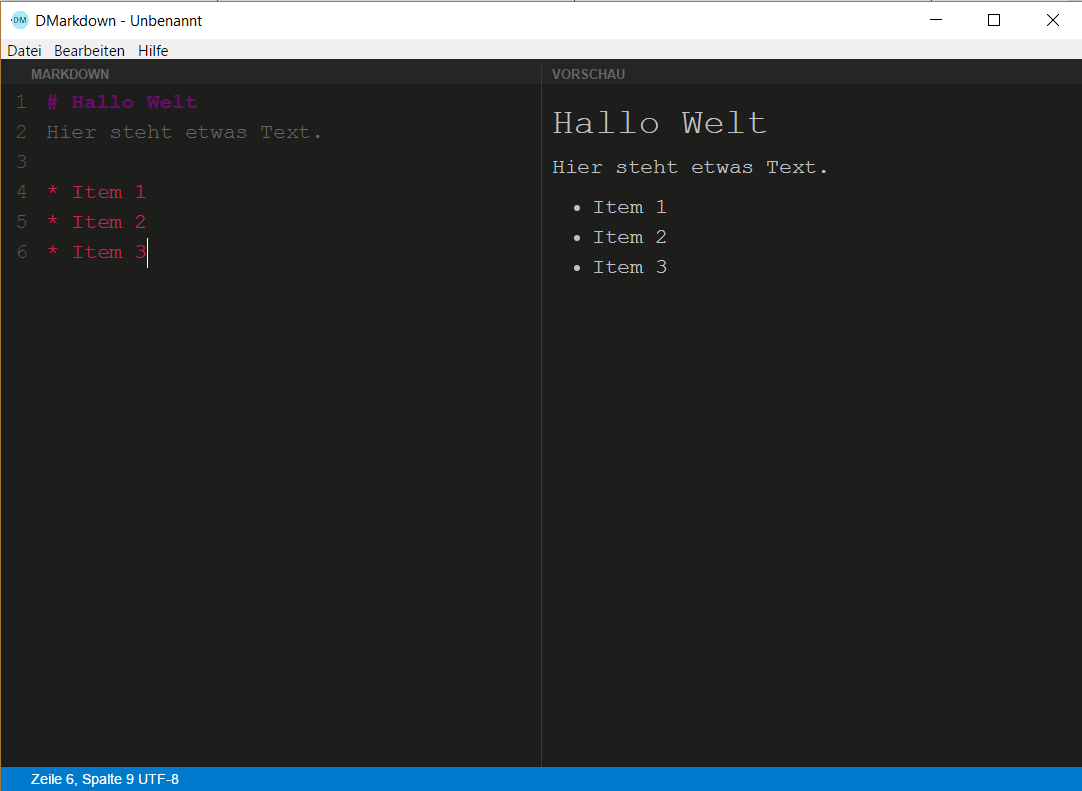


Abbildung 4 DMarkdown-Editor

Das aktuelle Logo in Abbildung 5 ist bisher nur ein Platzhalter und kann noch verändert werden.



Abbildung 5 DMarkdown-Logo

# Softwarearchitektur

In diesem Kapitel wird auf die Softwarearchitektur des Projekts eingegangen. Ziel dieses Kapitels soll es sein, Softwareentwicklern einen Überblick über das Projekt zu verschaffen und ihnen die Möglichkeit zu geben die Softwarelösung für ihre Bedürfnisse weiterzuentwickeln.

## Einflussfaktoren

Damit die Entwicklungsumgebung erfolgreich für weitere Softwareentwickler aufgesetzt werden kann, werden an dieser Stelle alle Einflussfaktoren für das System aufgelistet.

|  |  |
| --- | --- |
| **Randbedingung** | **Erläuterung** |
| Hardware-Infrastruktur | Alle PCs und MACs |
| Software-Infrastruktur | Windows 7/8/8.1/10, Linux und IOS |
| Grafische Oberfläche | Wird benötigt |
| Entwicklungsumgebung | Visual Studio Code Version 1.11.2 |
| Referenzarchitektur | Keine vorhanden |
| Analyse- und Entwurfsmethoden | Objektorientiertes Vorgehen |
| Programmiersprachen | HTML, CSS, JavaScript ES 6, Node.js Version 4.1.2 |
| Node-Packages | Electron Version 1.4.15,  markdown-js Version 0.5.0 |
| Testing-Framework | Jasmine Version 2.5.3 |

Es ist zu empfehlen alle Node-Packages global zu installieren, damit es hier keine Komplikationen gibt. Um das Programm anschließend ausführen zu können muss man in das Code-Verzeichnis wechseln und dort „electron .“ aufrufen. Um das Programm zu kompilieren führt man den Befehl „electron-packager ./ --electron-version=1.4.15 --asar –all“ aus. Dadurch wird für jede Plattform eine Applikation kompiliert.

## Use-Case-Diagramm

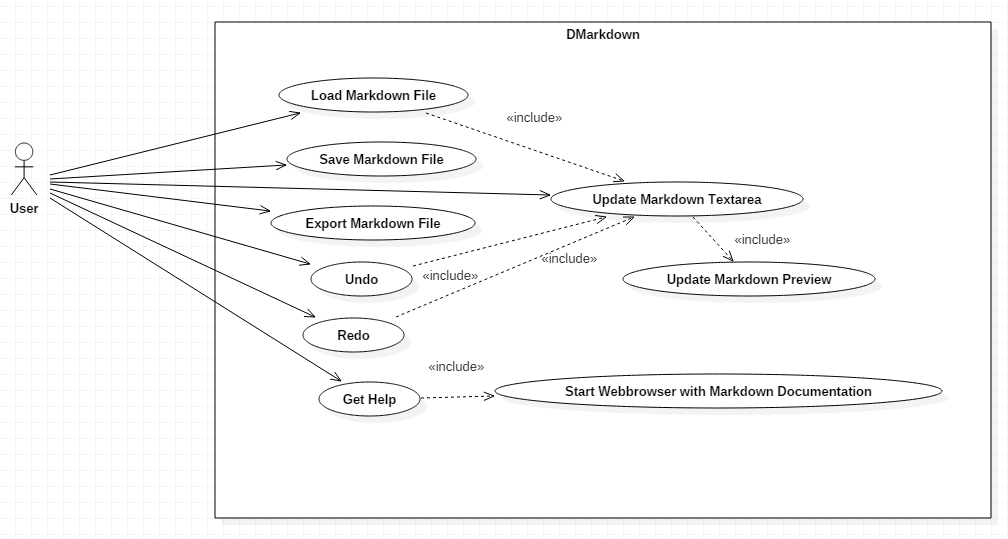
In diesem Use-Case-Diagramm (siehe Abbildung 6) sieht man wie der Benutzer mit dem Markdown-Parser interagiert.  


Abbildung 6 Use-Case-Diagramm

Der Benutzer hat die Möglichkeit Markdown-Dateien zu laden, zu speichern oder zu exportieren. Des Weiteren kann der Benutzer das Textfeld für den Markdown-Code editieren. Über den Undo und Redo Use-Case kann Änderungen an seinem Programmcode wiederholen oder rückgängig machen. Der „Get Help“-Use-Case kann verwendet werden, um den Benutzer Hilfe über das Internet zu verschaffen.

## Klassendiagramm

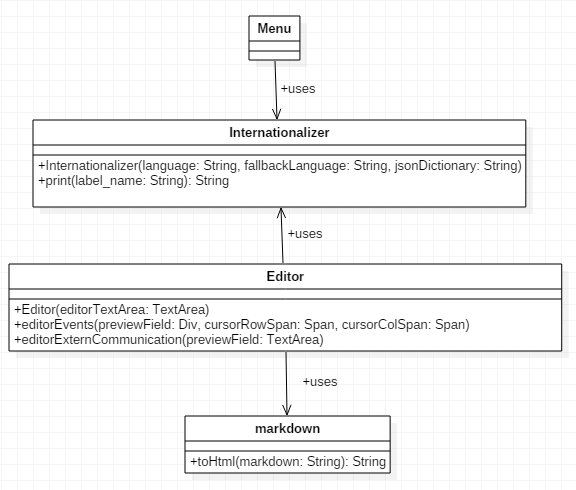
Das Klassendiagramm (siehe Abbildung 7) gibt Auskunft über die verwendeten Klassen und ihrer Zusammenhänge.  


Abbildung 7 Klassendiagramm

Die Menu- und Editor-Klasse verwenden beide den Internationalizer, um die Texte in die jeweiligen Ländersprachen zu übersetzen. Der Editor verwendet ebenfalls die markdown-Klasse, um den Markdown-Code in HTML zu parsen.

## Aktivitätsdiagramm

In diesem Aktivitätsdiagramm (siehe Abbildung 8) sieht man ein Highlevelablauf des Programms. Zuerst wird das GUI-Fenster erstellt. Anschließend kann man über die einzelnen Menüitems bestimmte Aktionen durch anklicken hervorrufen. Sobald man etwas in das Markdown-Textfeld hineinschreibt, wird das Vorschaufenster aktualisiert. Durch klicken auf das Kreuz am oberen Fensterrand, wird das Programm beendet.

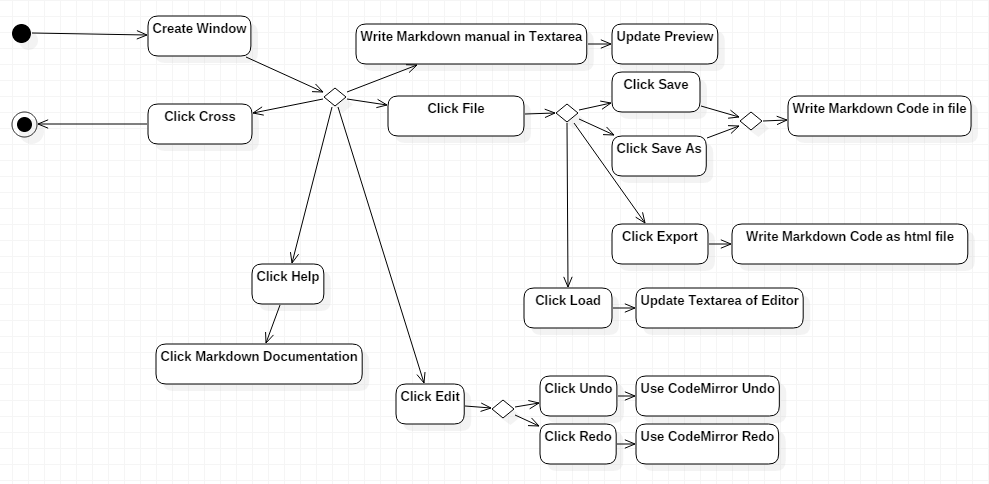


Abbildung 8 Aktivitätsdiagramm

# Softwaretests

Für die Unit-Tests wurde das Testing-Framework Jasmine verwendet. Jasmine bietet für JavaScript, Node.Js und einige weitere Programmiersprachen die Möglichkeit seine Software zu testen. Bei diesem Programm wurde vor allem die Internationalizer-Klasse getestet, da diese eine fundamentale Bedeutung für die korrekte Darstellung der Texte und Menüitems spielte.

Getestet wurden folgende Dinge:

* Konnte das JSON-Wörterbuch korrekt eingelesen werden
* Konnte mit Hilfe des JSON-Wörterbuch eine korrekte Übersetzung geboten werden
* Der Fall, wenn ein Label falsch eingegeben wurde
* Die Fallbacksprache (wenn eine Sprache ein Label nicht unterstützt, die Fallbacksprache jedoch)
* Was wenn eine Sprache nicht unterstützt wurde?

Alle Tests waren erfolgreich und können im Code-Verzeichnis unter ‘spec‘ wiederholt werden.

# Zusammenfassung und Ausblick

Der Markdown-Editor erlaubt es dem Benutzer eine Markdown-Datei zu bearbeiten, zu laden, zu speichern und zu exportieren. Des Weiteren unterstützt der Editor eine Undo-Redo-Funktion. Syntax-Highlighting, Zeilennummern, aktuelle Cursorposition und eine Echtzeitvorschau des Markdown-Codes wird ebenfalls unterstützt. Das Editordesign ist modern und übersichtlich.

Der Editor kann mit weiteren Features wie zum Beispiel “Schriftgrößen ändern“ etc. erweitert werden.