TECHNISCHE DOKUMENTATION TÜRSTATUS

DAVID CONRAD, SASCHA SEEMANN, ALEXANDER WALTER

22. April 2016

INHALTSVERZEICHNIS

1 EINLEITUNG

Für die öffentliche Darstellung variabler Öffnungszeiten ist es ein gängiges Mittel, den aktuellen Status (geöffnet oder geschlossen) zusammen mit den tagesaktuell gültigen Öffnungszeiten auf einer Website anzuzeigen. Es existieren bereits verschiedene Umsetzungen der Abfrage des Status, zum Beispiel über einen Reed-Kontakt an der Tür, und dessen Weitergabe an verschiedene Webplattformen. [1] [2]
Diese bieten jedoch keine Möglichkeit der Vorprogrammierung der Öffnungszeiten und erlauben darüber hinaus auch keine einfache Programmierung und Anzeige der verbleibenden Öffnungszeit vor Ort.

Für das FabLab Lübeck wurde nun im Rahmen eines Bachelorprojektes ein erweiterter Türstatus konzeptioniert und umgesetzt. Dieser bietet die Möglichkeit der Erstellung und Verwaltung variabler Öffnungszeiten, der öffentlichen Darstellung über die Website sowie das Forum des FabLab und ermöglicht über ein Hardwareinterface das Bearbeiten vor Ort. Der aktuelle Türstatus, die voraussichtlich verbleibende Öffnungszeit und die geplanten Termine werden auf einer Website dargestellt.

Die Realisierung wurde mit einem Einplatinencomputer (Raspberry Pi) durchgeführt, der die Erkennung des aktuellen Status, eine einfache Eingabe der verbleibenden Öffnungszeit vor Ort und das Hosting der Website übernimmt. Der Raspberry ist das Kernstück der Hardware. Um diesen herum wurde ein System entwickelt das im Folgenden genauer erklärt werden soll.

1.1 Dokumentenübersicht

Im Rahmen des Projekts wurde mehrere für die Planung, Entwicklung, Wartung und Nutzung des Systems relevante Dokumente erstellt, die anbei diesem Dokument zu finden sein sollten.

- **Anforderungen:** Umfasst die Kunden- und Entwicklungsanforderungen
- **Benutzersheet:** Kurze Bedienungsanleitung für Erstnutzer
- **Bedienungsanleitung Türstatus:** Bedienungsanleitung (mit allen Use-Cases)
- **Technische Dokumentation Türstatus:** Zur technischen Wartung und Weiterentwicklung des Systems

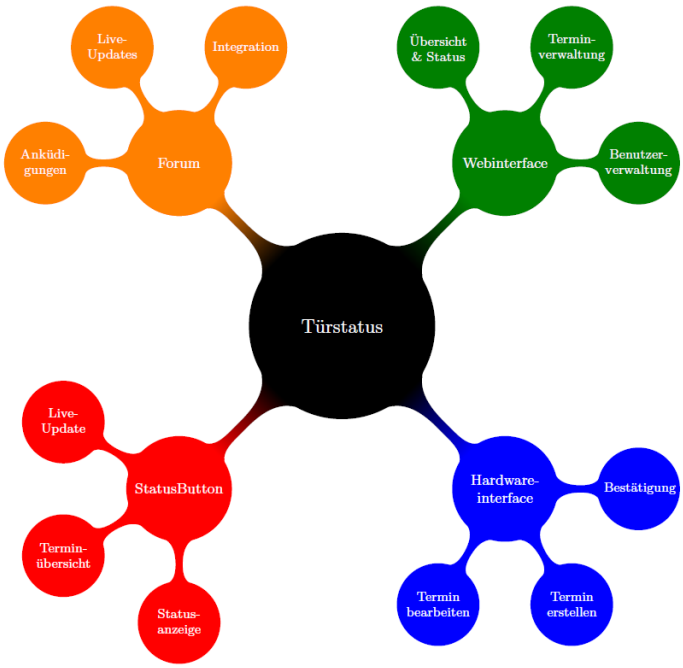
1.2 Termini & Erklärungen

Im nachfolgenden Dokument wird wird der Begriff *Türstatus* verwenden, womit in der Regel das Konzept eines Türstatus, wie vorangegangen erklärt gemeint ist. Vereinzelt ist dies aber auch ein Synonym für das System.
Mit `TODO` sind solche Sachen umschrieben, die noch angedacht sind und der Person ans Herz gelegt werden, welche das System weiter wartet. Sie sind

1 http://status.metameute.de/
2 https://blog.attraktor.org/tuerstatus-faq/

nicht essenziell notwendig, aber sinnvoll.
Alle Dateipfade beziehen sich hierbei auf das Dateisystem des Raspberry Pi.

2 AUFBAU



(a) Systemaufbau

Das System besteht vereinfacht gesagt aus vier Komponenten:

- Dem **Status-Button** (auf der Website des FabLab) für **Besucher** und Interessenten
- Der **Forenintegration** für **Mitglieder** des FabLab
- Dem **Webinterface** für **betreuende Mitglieder** des FabLab
- Dem **Hardwareinterface** für **betreuende Mitglieder** des FabLab

All diese Komponenten haben eine eigene Funktion und bilden gemeinsam ein System.
So gibt der Button auf der Website Auskunft über den Startzeitpunkt des nächsten Termins und dessen Dauer.
Die Forenintegration baut auf der bestehenden Regelung auf, dass Terminankündigungen über das Forum des FabLab bis dato erfolgen und sich die aktuellen Mitglieder nicht umstellen müssen.
Das Webinterface ist das eigentliche Kernstück des Status. Hier können regelmäßige oder Einzeltermine erstellt, geändert und gelöscht werden durch hier angemeldete Mitglieder. Administratoren können zu dem die anderen Benutzer erstellen und verwalten. Auf der Startseite gibt es zu dem eine Übersicht über die nächsten geplanten Termine und den aktuellen Status.
Das Hardwareinterface dient zur Verifikation von Terminen vor Ort, zum Nachbearbeiten und zur eingeschränktem Terminauskunft (nächster geplanter Termin). Im folgenden gehen wir weiter ins Detail.

3 HARDWARE

3.1 Verwendete Komponenten

Dies ist eine kurze Auflistung der gewählten Komponenten und eine Erklärung ihrer Wahl.

3.1.1 *Raspberry Pi*

Verbaut wurde ein Raspberry Pi 2 Modell B, der über ein mitgeliefertes 5V-Netzteil mit Strom versorgt wird. Dieser umfasst ein 16 GB micro SD Karte (Transcend Extreme-Speed Micro SDHC 16GB) als Speicher. Als Betriebssystem wurde Raspbian gewählt.

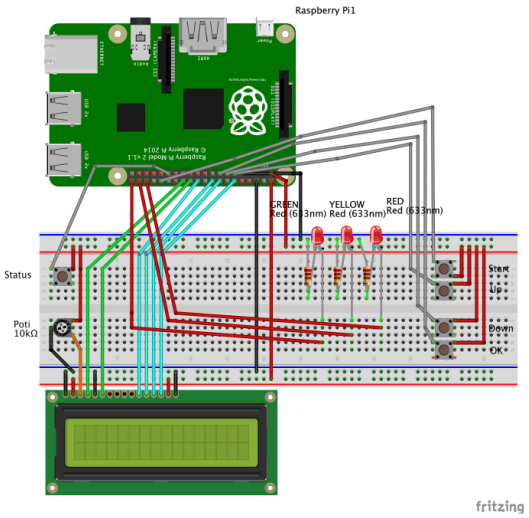
3.1.2 *Display*

Als Display wurde das HD44780 LCD Display (mit blauer Hintergrundbeleuchtung) verbaut dies ist wie im Schaltplan ersichtlich mit den GPIO-Ports des Raspberry Pi verbunden. Bei der Anbindung haben wir uns an einen Webguide gehalten. [3]

3.1.3 *Weitere Komponenten*

- 3 5mm LED (in rot, gelb und grün)
- 5 Drucktaster
- div. Jumperkabel
- 1 Netzteil (230V Wechselstrom zu 5V Gleichstrom)
- 1 10m Ethernetkabel

3.2 Hardwarearchitektur



(a) Schaltplan

3 http://www.schnatterente.net/technik/raspberry-pi-4x20-hd44780-display

Achtung: Der Schaltplan ist grundlegend gleich zu dem hier dargestellten, es wurden nur:

- Die Masse-Leitungen der Taster zusammengefasst zu einer Masseleitung
- Die Masse-Leitungen der LEDs zusammengefasst zu einer Masseleitung
- Die Vorwiderstände (unter dem Schrumpfschlauch) angelötet an die LEDs
- Das Potentiometer durch zwei feste Widerstände ersetzt

Hierdurch ist das Breadboard (Schaltbrett) obsolet geworden.

4 SOFTWARE

4.1 Verwendete Software

Ein kurzer Überblick über die im System verwendeten Softwarekomponenten:

- **Betriebssystem (Raspberry)**: Raspbian Jessie (Release 2015-09-24) - Kernel 4.1.7
  Wurde gewählt da von Raspberry Pi (Firma) mit gepflegt und empfohlen.

- **Laufzeitumgebung:** Python3 [4]
  Da Python auf Raspberry gut dokumentiert & unterstützt.

- **Server:** Apache Server (Version 2.4.10) [5]
  Gängigste HTTP-Webserver Software

- **Framework** Django [6]
  Gut gepflegtes, umfangreiches Webframework in Python
  - Timepicker [7]
    JQuery/JavaScript Skript, welches die Eingabe von Daten und Uhrzeiten beim Eintragen neuer Termine erleichtert. Dieses Skript wurde in die Templates des Webinterfaces integriert.

- **Datenbank:** SQLite3 [8]
  Ist schnell und einfach zu benutzen und wird von Django direkt in der Standardkonfiguration bereits unterstützt. Außerdem beansprucht SQLite3 wenig Ressourcen, da nicht ständig ein Datenbankprozess bzw. -server laufen muss.

- **Website-Button:** HTML5, CSS & Javascript
  Eingebettet in ein iFrame

4 https://www.python.org/
5 https://httpd.apache.org/
6 https://www.djangoproject.com/
7 https://jonthornton.github.io/jquery-timepicker
8 https://www.sqlite.org/

4.2 Softwarearchitektur

4.2.1 *Konzept*

Wie in der Graphik (im Anhang S.14) ersichtlich ist das Kernstück der Softwarearchitektur der Raspberry Pi welcher mit Raspbian als Betriebssystem läuft. Als Hauptapplikationen existieren das Web- und das Hardwareinterface, welche beide in Python geschrieben sind. Als Datenbank fungiert SQLite3 und als Framework Django. Extern zugänglich ist der Pi über das SSH-Gate und der Apache Webserver.

4.3 Realisierung

Das Gesamtsystem (siehe S.15) besteht aus zwei wesentlichen Komponenten: Dem Webinterface und dem Hardware-Controller.
Der Hardware-Controller steuert alle In- und Output-Komponenten, die im FabLab-Terminal, welches direkt im FabLab hängt, verbaut sind. In einer Endlosschleife wird über die GPIO-Schnittstelle die Eingabe der Taster abgefragt und abhängig vom aktuellen Zustand werden verschiedene Aktionen ausgeführt. Dabei wird ebenfalls über GPIO das Display angesprochen und die anzuzeigenden Daten des aktuellen Zustands werden ausgegeben. Auch wird an einigen Stellen auf die Datenbank zugegriffen, wenn an den gespeicherten Terminen Änderungen vorgenommen wurden, z.B. wenn der Termin bestätigt oder der Typ von SelfLab auf OpenLab geändert wurde.
Das Webinterface besteht dagegen aus deutlich mehr Softwarekomponenten. Das Kernstück des Webinterface bildet das Webframework Django. Die Implementierung des Webinterfaces ist ähnlich dem Model-View-Controller(MVC)-Modell. Die "ViewsKomponente ist hierbei allerdings nicht wie man vermuten könnte die View aus dem MVC-Modell. Die eigentlichen Ansichten werden hauptsächlich in den HTML-Templates über die Django Template Language definiert. Die "Views-Komponente dient hier eher als der Controller, denn an dieser Stelle werden die Seitenaufrufe und Benutzereingaben verarbeitet. Dabei wird auf das Model, welches die Klasse "Dateënthält, und Djangos Benutzerverwaltung zugegriffen. Die Datenbankzugriffe die hinter den Operationen mit den Models und Benutzern stecken, werden von Django verwaltet. Die berechneten Daten, die angezeigt werden sollen, wird an das geladene Template weitergeleitet. Das Template definiert klar, in welcher Form diese Daten letztendlich angezeigt werden. Damit die Website und Django überhaupt funktionieren können kommunizieren der Webserver Apache und Django über die WSGI-Schnittstelle, damit bei URL-Aufrufen die richtigen Skripte ausgeführt werden.

4.4 Datenbank

Die im Web- und Hardwareinterface dargestellten Informationen zu Terminen und Benutzern sind in einer Datenbank gespeichert. Hierbei handelt es sich um eine SQLite3-Datenbank. Sie befindet sich im angegebenen Dateipfad. Die wichtigste Tabelle ist in der nachgestellten Tabellenübersicht genauer erklärt.

4.4.1 *Allgemeine Informationen*

- Eingesetzte Software: SQLite 3
- Pfad zur Datenbankdatei: /var/www/db.sqlite3
- Zugriff auf die Datenbank:
  - **Webinterface-Dateien** (in /var/www): Zugriff über Django-Models
  - **Discourse-Anbindung** (in /var/www/tuerstatus/views.py): Zugriff über Django-Models
  - **FabLab-Terminal** (/var/www/main.py): Zugriff über Python SQLite3-Library
  - **Website-Button**: Zugriff über Python SQLite3-Library

4.4.2 *Tabellenübersicht „tuerstatus_date“*

Django erstellt einige Tabellen für die allgemeine Verwaltung wie Benutzer, Gruppen, Sessions usw. Für alle Klassen in „models“ erstellt Django jeweils eine neue Tabelle mit entsprechenden Spalten zu den Attributen. Die Tabelle für das eigene Modell „Date“ heißt „tuerstatus_date“ und besteht aus folgenden Spalten:

- **id** (integer): Primary Key
- **start** (datetime): Startzeitpunkt für den Termin
- **type** (integer): SelfLab (1) oder OpenLab (2)
- **user** (varchar): Benutzername des Users, der den Termin erstellt hat
- **end** (datetime): Endzeitpunkt für den Termin
- **repeat** (integer): Typ der Wiederholung: Keine Wiederholung (0), Wöchentlich (1)
- **link** (integer): Gibt eine eindeutige ID für zusammenhängende, regelmäßige Termine. Alle Einzeltermine, die zu einem regelmäßigen Termin gehören, haben als „link“ die „id“ des ersten Termins dieser Reihe.
- **started** (bool): Gibt an, ob dieser Termin gestartet bzw. über das Terminal im FabLab bestätigt wurde.
- **deleted** (bool): Statt den Eintrag zu löschen, wird dieser mit „true“ markiert. Er wird benötigt, wenn zu diesem Termin ein Ankündigungsbeitrag im FabLab-Forum erstellt wurde, um eine entsprechende Aktualisierung dort vorzunehmen.

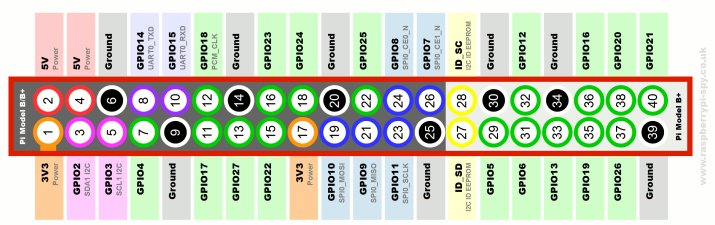
- **edit** (bool): Wird wie „deleted“ für Änderungen für das Forum verwendet. Sobald ein Termin über das Webinterface bearbeitet wurde, ist dieser wert „true“ und wird wieder auf „false“ zurückgesetzt, wenn entsprechende Änderungen in das Forum geschrieben wurden.
- **topic_id** (integer): Gibt an, unter in welchem Thema im Forum Informationen zu diesem Termin stehen. Dies wird ebenfalls für Änderungen an Terminen und im Forum benötigt.

5 USER-INTERFACE-KOMPONENTEN

In den folgenden Unterkapiteln sind die vier verschiedenen Komponenten erläutert, die dem Benutzer (User) Interaktionen mit dem System ermöglichen.

5.1 Hardwareinterface

Das Hardwareinterface wird durch eine einzelne Routine gesteuert. Diese befindet sich unter:
/home/pi/FabLabTuer/main.py
Sie greift mittels der GPIO-Ports auf die Taster, LEDs und das LCD-Display zu (Schaltplan siehe S. 16). Die Portbelegung des Raspberry Pi 2 lautet:



(a) GPIO-Ports (Raspberry Pi 2 Modell B)[9]

Mittels regelmäßiger Datenbankabfragen wird immer der aktuellste Termin ermittelt.
Für genauer Information zur Funktionsweise der Routine lesen sie bitte die ausführlichen Kommentare im Quellcode.
Die Routine wird beim Systemstart mittels eines so genannten **Cronjob** gestartet. Für genauer Details hierzu lesen sie bitte den Abschnitt Cronjob.

5.2 Webinterface

Alle hier angegebenen Dateien befinden sich im Pfad */var/www*.

5.2.1 *Benutzerverwaltung*

Da nur zwei verschiedene Arten von Benutzern existieren, wurden die zwei Standardbenutzertypen und Django verwendet. Benutzer, die als

Admin erstellt werden, werden in der „views.py“ in der „addUser“-Funktion als Superuser erstellt und haben somit alle Rechte. Da die Adminoberfläche von Django nicht in der „urls.py“ eingebunden ist, kann hier auch kein großer Schaden entstehen.
Wenn bestimmte Zugriffe oder Funktionen nur von Admins verfügbar sind, wird einfach nur überprüft, ob der zurzeit eingeloggte Benutzer vom Typ „Superuser“ ist. Für Aktionen, die nur bestimmte Benutzer, aber nicht nur Admins ausführen können, kann der Benutzername abgefragt werden. Diese Abfragen sind sowohl in der „views.py“ als auch in den HTML-Templates über Djangos Template-Sprache[10]

5.2.2 *Dateiübersicht und Erklärung*

- **db.sqlite3** - Datenbank
- **manage.py** – Von Django mitgeliefert. Projekt kann darüber verwaltet werden, z.B. um Datenbankstruktur anzupassen, wenn an den Models etwas geändert wurde.
  Mehr Informationen: `https://docs.djangoproject.com/en/1.9/ref/django-admin/`

/var/www/FabLab:

- **settings.py** – Konfigurationsdatei für Django. Enthält eigene Konstante für den Pfad des Logs für das Webinterface. Zu finden unter „/var/www/tuerstatus/log/log.txt“.
  Mehr Informationen zur settings.py: `https://docs.djangoproject.com/en/1.9/ref/settings/`
- **urls.py** – URL-Konfigurationsdatei. Bindet nur die „urls.py“ aus „tuerstatus“
- **wsgi.py** – Konfigurationsdatei für WSGI-Schnittstelle von Apache

/var/www/tuerstatus:

- **discourse.py** – Schnittstelle zum erstellen von neuen Themen und Beiträgen von Discourse
- **forum_update.py** – Skript, welches regelmäßig ausgeführt wird und Beiträge über Änderungen und neue Termine im Discourse-Forum in der SelfLab-Ankündigungs-Kategorie erstellt.
- **models.py** – Modelliert die Klasse „Date“. Django übernimmt hier die Datenverwaltung in der Datenbank. Die Funktion „delete()“ wird hier überschrieben, um das Löschen von Einträgen zu verhindern. Stattdessen werden die Einträge nur als gelöscht und editiert markiert, damit **forum_update.py** Informationen über gelöschte Termine erhalten kann.
- **urls.py** – Enthält alle möglichen URLs und einen Verweis auf die entsprechende Funktion in „views.py“. URLs bezüglich der Passwort-Zurücksetzen-Funktion verweisen auf die von Django entsprechend mitgelieferten Funktionen, wobei allerdings eingene Templates verwendetet wurden.
- **views.py** – Enthält einen Großteil der Logik für die einzelnen Views, also die Erstellung der einzelnen fertigen Seiten, die im

10 `https://docs.djangoproject.com/en/1.9/ref/templates/language/`

Browser angezeigt werden. Dabei nutzt jede View ihr eigenes Template (alle zu finden unter „/var/www/tuerstatus/templates“). Für den dynamischen Teil der Seite finden in der „views.py“ verschiedene Datenbankzugriffe statt. Zum Teil sind diese Zugriffe statisch, teilweise aber auch abhängig von den POST-Daten, die von den Benutzereingaben kommen. Das Webinterface ist über die Zugriffe über die Models in Django vor SQL-Injections sicher.[11]
Sind irgendwelche Eingaben ungültig, werden in einer Variable „error“ Fehlermeldungen als String an die Templates übergeben. Ebenfalls übergeben werden alle anderen relevanten Teile für die Seite wie z.B. Listen von Daten.

/var/www/tuerstatus/log:

- **log.txt** – Hält alle Änderungen an den Daten und Benutzern und Logins fest.

/var/www/tuerstatus/templates (Es werden nur die wichtigen aufgelistet):

- **pw** (Verzeichnis) – Enthält alle Templates, die für die Passwort-Zurücksetzen-Funktion benötigt werden.
- **head.html** – Enthält alle head-Daten und und das Navigationsmenü
- **head_base.html** – Einfache Head-Datei ohne Navigationsmenü
- **foot.html** – Wird in allen Templates am Ende eingebunden.

Hinweise zu den Templates:

- Jedes Template bindet am Anfang entweder „head.html“ oder „head_base.html“ ein.
- Die Templates bestimmen, wie die übergebenen Daten dargestellt werden und binden auch irrelevante Daten nicht ein, wenn z.B. nicht genügend Rechte vorhanden sind. Dies geschieht über spezielle Tags, die von Django interpretiert werden. Mehr dazu: https://docs.djangoproject.com/en/1.9/ref/templates/language/

/var/www/tuerstatus/static/tuerstatus:

- **style.css** – Legt den allgemeinen Style für das Webinterface fest.
- **tuer.js** – Allgemeine JavaScript-Funktionen für das Webinterface
- **img** (Verzeichnis): Enhält alle genutzten Bilder
- Die restlichen hier enthaltenen Dateien sind für die erleichterte Datum- und Zeiteingabe für die Termine (siehe jquery.timepicker in „Verwendete Software“)

5.3 Website-Button

Auf der Website des FabLab http://www.fablab-luebeck.de/index.php/oeffnungszeiten/ befindet sich eins der Herzstücke des Projekts: Der Button, weclher den Status des FabLab anzeigt.
Dieser besteht drei Teilen:

- **HTML Website**: bestehend aus einer HTML (mit Javascript) und einer CSS Datei

11 https://docs.djangoproject.com/en/1.9/topics/security/#sql-injection-protection

- **FTP-Verzeichnis** mit XML-Datei als Puffer für etwaige Hardwareabstürze im FabLab
- **XML File Handler** welcher die besagte XML-Datei erstellt, hochläd und bei Änderungen aktualisiert.

5.3.1 *HTML Website*

Die HTML-Website befindet sich unter `http://openinghours.fablab-luebeck.de/` und ist mittels eines iFrames auf der eigentlichen Website des FabLab [12] eingebunden worden.
Auf die HTML- und CSS-Datei welche in einem FTP-Server-Verzeichnis liegen kann mittels eines FTP-Clients [13] zu gegriffen werden. Der Javascript-Code ist in die HTML-Datei eingebettet. Dieser greift auf eine XML-Datei Names „dates.xml“ zu, welche im Unterordner des FTP-Verzeichnissen liegen. In dieser Datei sind die nächsten fünf Termine gepuffert, so dass das die Website teilweise unabhängig ist gegenüber Systemausfällen des Raspberry Pi.

5.3.2 *FTP-Verzeichnis*

Hier liegen die HTML-, CSS- und XML-Datei.

5.3.3 *XML File Handler*

Dieser XML File Handler wird beim Starten des Raspberry per Cronjob gestartet und läuft dann dauerhaft. Er lädt dabei in regelmäßigen Intervallen (momentan alle 10 Sek.) die nächsten fünf Termine aus der Datenbank und schreibt diese in eine XML-Datei und lädt diese in das FTP-Verzeichnis in den Ordner „xml“ hoch. Danach läuft der Abfrage der Datenbank kontinuierlich weiter. Immer dann, wenn es zu Änderungen kommt wird die geänderte XML-Datei hoch geladen und die alte Version überschrieben. (Achtung: im Quellcode ist noch eine 2. inaktive Implementierung für einen Upload in eine Dropbox)

5.4 Forum (Discourse-Integration)

Nach dem Erstellen eines neuen SelfLab-Termins wird automatisch ein neues Thema in der entsprechenden Kategorie im FabLab-Discourse-Forum erstellt. Die Texte bzw. die Templates für die Texte sind festgelegt in „views.py“ in der Funktion „forumUpdate“. Bei Änderungen müssen dort die entsprechenden Zeilen abgeändert werden. Dabei müssen aber die von Discourse vorgegebenen Mindestlängen für diese Texte eingehalten werden, da sonst Fehler auftreten.
Das ganze wird über die Discourse-API [14] realisiert.
Da der Zugriff auf die Daten über Python bzw. Django abläuft, wurde für die http-Schnittstelle der Discourse-API eine einfache API in Python implementiert, die neue Themen erstellen und bearbeiten und neue Beiträge verfassen kann. Diese Funktionen sind in der „discourse.py“ zu finden. Die „forumUpdate“ Funktion aus der „views.py“

12 http://www.fablab-luebeck.de/
13 https://filezilla-project.org/download.php
14 https://meta.discourse.org/t/discourse-api-documentation/22706

übernimmt die Logik, welche Termine beachtet und ins Forum geschrieben werden sollen.

5.4.1 *Konfiguration*

Am Anfang der „forumUpdate“ Funktion in der „views.py“ gibt es einige Zeilen zur Konfiguration:
HOST: Adresse des Servers inkl. „http://“ z.B. „http://example.com“. Hier wird auch der Port angegeben, wenn dieser von 80 abweicht, z.B. „ http://example.com:8080“ für Port 8080.
API_KEY: Der API-Key von Discourse. Dieser kann in Discourse für beliebige Benutzer generiert werden.
CATEGORY_ID: ID der Kategorie, unter der die neuen Beiträge erscheinen sollen. Diese Zahl wird von Discourse verteilt, wenn die Kategorie erstellt wurde. Die Kategorie für SelfLab-Ankündigungen hat die ID 9.
THRESHOLD: Anzahl der Tage in der Zukunft, die für neue Posts für zukünftige Termine berücksichtigt werden. 14 würde bedeuten, dass nur Termine in den nächsten zwei Wochen berücksichtigt werden, d.h. neue Termine werden frühestens 14 Tage vorher im Forum angekündigt.
SCHEDULE_LINK: Der Standardtext verlinkt auf das Webinterface mit den nächsten Daten. SCHEDULE_LINK ist der Link zu dieser Seite.

6 WARTUNGSINFORMATIONEN

Hier finden sie verschiedene Informationen zu Hilfsmitteln, welche für die Wartung des Systems nutzlich sind.

6.1 SSH-Gate

Ein SSH-Gate dient dem externen Zugriff auf die Konsole einer Rechners oder Servers über einen Netzwerk. Hier wurde ein SSH-Gate eingerichtet.

6.2 Cronjob

Unter Linux können regelmäßig auszuführende Skripte mittels so genannter Cronjobs (bzw. Crontabs) gestartet werden. Diese können ein Skript in Intervallen starten oder zu bestimmt Zeitpunkten z.B. zum Systemstart. Anlegen kan man diese in der Konsole wie folgt:

sudo su
crontab -e
Hier wird nun wie in der Datei beschrieben Cronjobs angelegt. Anschließend die Datei speichern und schließen. Abschließend noch die neuen Cronjobs starten.
/etc/init.d/cron start

6.3 Netzwerkkofiguration

6.3.1 *Portweiterleitung*

Damit das Webinterface über das Internet erreichbar ist, muss bei einem Routerwechsel oder Ähnlichem eine Portweiterleitung für Port 80 (HTTP-Webserver) sowie Port 22 (SSH-Gate) auf den Raspberry Pi eingerichtet werden. Dies erfolgt über die jeweilige Administration-Oberfläche des Routers.

6.3.2 *Dynamisches DNS*

Damit auch bei einem IP-Wechsel der externen IP des Raspberry Pi das Webinterface noch erreichbar ist, wurde im Router ebenfalls eine Domain [15] mit dynamischen DNS eingerichtet.
`TODO:Eigenen-DDNS-Einrichten`

6.4 Hinweise zur Wartung

Das Webinterface und somit der Raspberry Pi ist über das Internet erreichbar. Deshalb muss die Software aktuell gehalten werden, um Sicherheitslücken zu schließen und weniger verwundbar gegen Angriffe zu sein. Dies ist besonders wichtig für Apache und Django.

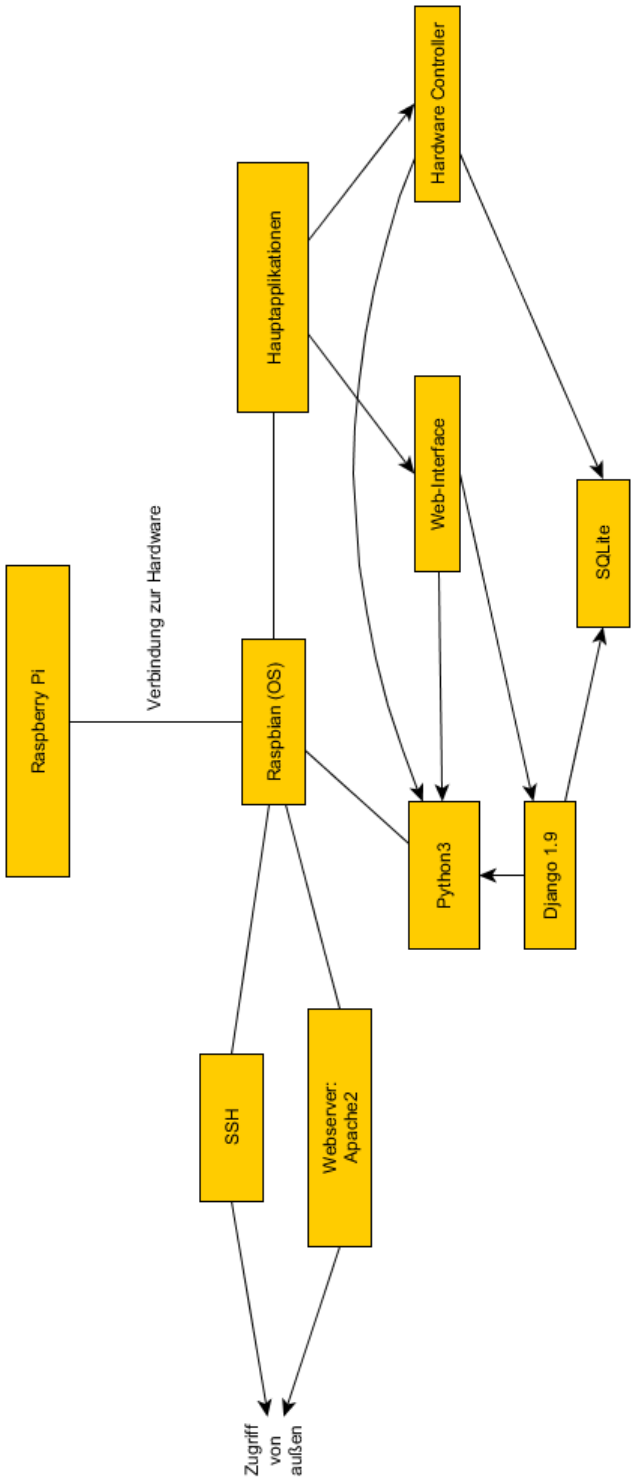
Wichtiges:

- Aktualisieren von Django [16]
- Die verwendete Release-Version 1.9 wird noch mit Sicherheitsupdates bis April 2017 unterstützt.[17]
- Das Aktualisieren von Apache kann über das „apt-get“-Kommando geschehen.

15 http://fablabtuer.ddns.net/
16 https://docs.djangoproject.com/en/1.9/howto/upgrade-version
17 https://www.djangoproject.com/download/#supported-versions

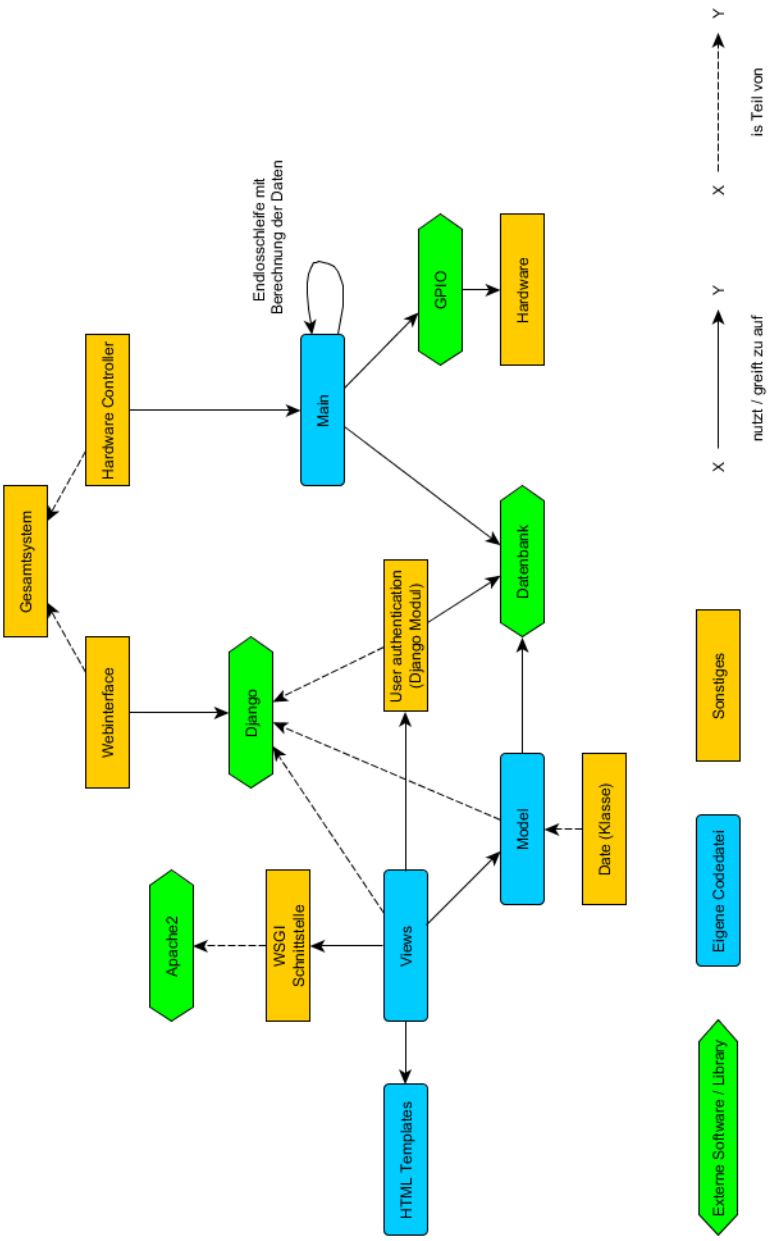
7 ANHANG

7.0.1 *Konzept Softwarearchitektur*



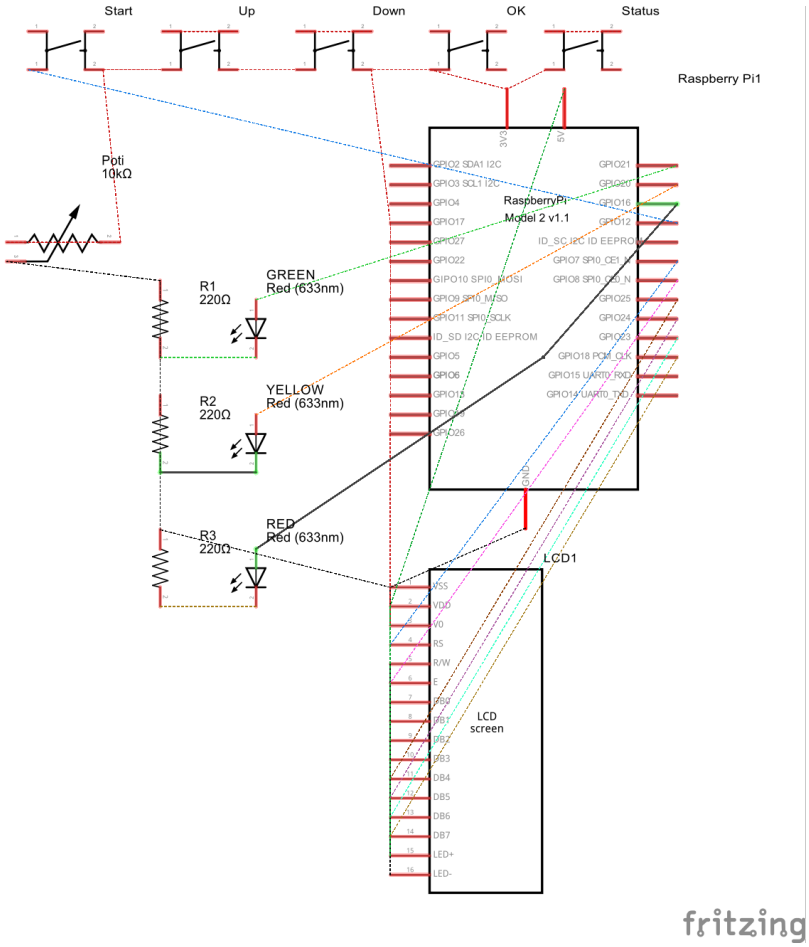
(a) Konzept Softwarearchitektur

7.0.2 *Tatsächliche Softwarearchitektur*



(a) Tatsächliche Softwarearchitektur

7.0.3 *Schaltplan*



(a) Hardware Schaltplan