TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Entwicklung einer internen Social Media Plattform mit personalisierbarem Dashboard fur Studierende

Esther Kleinhenz

Matrikelnummer: 2649270

Bachelorarbeit

zur Erlangung des akademischen Grades

Bachelor of Engineering

Media Engineering

Erstprüfer: Prof. Dr. Oliver Hofmann
Zweitprüfer: Prof. Dr. Matthias Hopf

Nürnberg, 4. Oktober 2018

Hiermit erkläre ich, dass die vorliegende Arbeit von mir selbständig verfasst und nicht anderweitig für Prüfungszwecke vorgelegt wurde, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche oder sinngemäSSe Zitate als solche gekennzeichnet wurden.

Nürnberg, 4. Oktober 2018

Katja Cornelia Hader
E-mail: haderka56442@th-nuernberg.de

Studiengang Wirtschaftsinformatik
Georg Simon Ohm
Technische Hochschule Nürnberg
KeSSlerplatz 12
90489 Nürnberg
Deutschland

Abstract

Content of this Bachelor thesis is ....

Keywords: bla,bla Georg Simon Ohm, Wirtschaftsinformatik

Inhaltsverzeichnis

Abstract i

Abbildungsverzeichnis 1

1 Einleitung 3
- 1.1 Ausgangssituation . . . 3
- 1.2 Ziel der Arbeit . . . 3

2 Framework 5
- 2.1 Django . . . 5
  - 2.1.1 Besonderheiten . . . 6
- 2.2 Erweiterungen . . . 7
  - 2.2.1 Taggable-Manager . . . 7
- 2.3 Bootstrap . . . 7

3 Prototyp 9
- 3.1 Organisation . . . 9
  - 3.1.1 Verwaltung im Administrator-Backend . . . 9
  - 3.1.2 Berechtigung der User . . . 9
- 3.2 Funktion . . . 9
  - 3.2.1 Abonnieren . . . 9
  - 3.2.2 Filtern . . . 9
  - 3.2.3 Benachrichtigung . . . 9

4 Ergebnis 11
  - 4.0.1 Evaluierung . . . 11

5 Zusammenfassung und Ausblick 13

Referenzen 15

Abbildungsverzeichnis

2.1 Vereinfachter MVP 5
2.2 Request-Response-Kreislauf des Django Frameworks 6

Kapitel 1

Einleitung

Die vorliegende Arbeit beschäftigt sich mit der wachsenden E-Mail-Flut in den Postfächern der Studierenden und wie man diese reduzieren kann. Schon seit geraumer Zeit ist bekannt, dass das Versenden von Informationen über elektronische Post nicht nur Vorteile mit sich bringt. Wie der Spezialist für Gesundheitsprozessberatung in einem Bericht der Mitteldeutschen Zeitung erwähnt, macht es die stets wachsende E-Mail-Menge unmöglich sich vernünftig mit den Informationen zu befassen"([Ver13]). Nicht nur am Arbeitsplatz sondern auch in Hochschulen wird Gebrauch gemacht, weitere Empfänger oder sogar ganze Verteiler mit in die Kopie einer E-Mail zu integrieren. Um die Prioritäten der Informationen besser bilden zu können, sollen Studierende selbst entscheiden, welche Nachrichten relevant sind. Hierfür wird eine Social Media Plattform mit personalisierbarem Dashboard implementiert.

1.1 Ausgangssituation

Alle Informationen der Fakultat Elektrotechnik Feinwerktechnik Informationstechnik, kurz efi, werden uber die globalen Verteiler des Hochschulinternen Postfaches versendet. Viele dieser Daten sind jedoch nur fur eine geringe Schnittmenge der Empfanger relevant und lassen sich nur schwer priorisieren. Das standig uberlastete Postfach muss somit regelmaßig gepflegt werden. Einen massiven Administrativen Aufwand bedeutet es, E-Mails zu filtern und nach persönlichem Ermessen zu verwalten. Zudem leidet die Nachhaltigkeit der Informationen. Mochten die Empfanger altere E-Mails abrufen, mussten diese meist schon entfernt werden um Platz fur den neuen, eintreffenden E-Mail-Verkehr zu schaffen. Diese Situation fuhrt dazu, dass Empfanger die Informationen meist nicht lesen und sofort entfernen. Die Ersteller haben keinerlei Moglichkeiten zu uberprufen ob und wie viele Studierende und Dozenten eingehende Nachrichten offnen und lesen.

1.2 Ziel der Arbeit

Ziel der Arbeit ist es, durch die Einbindung einer Social Media Plattform den Speicheraufwand des Hochschulpostfaches fur Studierende der Efi-Fakultat zu reduzieren. Die Flut

an E-Mails soll durch das Verwenden eines personalisierte Dashboard gedrosselt werden. Hierbei wird zunächst der Fokus auf die grundlegenden Funktionen der Website gelegt. Dazu gehört das Abonnieren, einpflegen von neuen und löschen von alten Nachrichten. Zudem sollen die Autoren benachrichtigt werden, in welchem Umfang die hochgeladenen Informationen bereits abonniert und gelesen wurden.

Kapitel 2

Framework

Um die Website-Erweiterung realisieren zu können, wird zunächst festgelegt welche Programmierschnittstellen verwendet werden. Im Web-Backend fällt die Wahl auf die objektorientierte Sprache Python, die ausschließlich Serverseitig anwendbar ist. Der Programmaufbau Pythons macht den Code leicht lesbar und der einfache Syntax ermöglicht einen strukturierte Implementierung der Website([Ndu17]). Ein entscheidender Vorteil hierbei ist das dazugehörige Framework Django, auf das im folgenden Kapitel genauer eingegangen wird.

2.1 Django

Django ist ein Web-Framework, das auf einer Model-View-Presenter (MVP) Architektur basiert. Ähnlich wie der Model-View-Controller sind die Interaktionen zwischen Model und View die Auswahl und Ausführung von Befehlen und das Auslösen von Ereignissen (vgl. Abbildung 2.1). Da die View aber hier bereits den Großteil des Controllers übernimmt, ist der MVP eine Überarbeitung. Der Teil, der Elemente des Modells auswählt, Operationen durchführt und alle Ereignisse kapselt, ergibt die Presenter-Klasse([She09]). Durch die direkte Bindung von Daten und View, geregelt durch den Presenter, wird die Codemenge der Applikation stark reduziert.

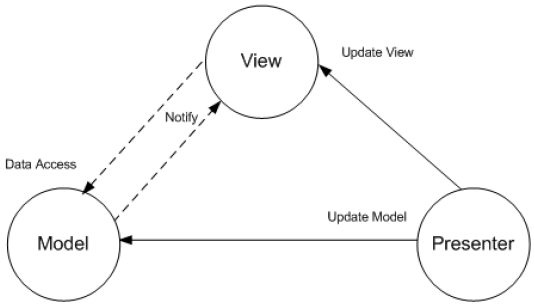

Abbildung 2.1. Vereinfachter MVP

Der Prozess vom Anfragen der URL über den Server, bis hin zur fertig gerenderten

Website kann wie folgt vereinfacht darstellen. Der User gibt eine URL im Browser ein und sendet sie an den Web-Server. Das Interface WSGI am Web-Server leitet den Request zum passenden Objekt einer Applikation weiter. Da das Framework über eine explizite Zuweisung der verschiedenen Seiten verfügt, iteriert der urlresolver über die vorhandene URL-Struktur im Code (url.py). Gibt es eine Übereinstimmung, wird die damit verknüpfte Funktion in der View (view.py) aufgerufen. Hier ist die gesamte Logik der Website lokalisiert. Wie bereits erwähnt, ist es möglich unter Anderem auf die Datenbank der Applikation zuzugreifen und Eingaben des Users über eine Form zu verarbeiten. Nachdem werden die Informationen der View an das Template weitergereicht. Es handelt sich dabei um eine einfache HTML-Seite in der der strukturelle Aufbau im Frontend festgelegt wird. Die Informationen der View können hier zwischen doppelt-geschweiften Klammern eingebunden werden und, wenn nötig, mit einfachen Python-Befehlen anpassen. Nun kann das Template einen Response an den Web-Server schicken und die fertige Seite wird beim Klienten im Browser gerendert (vgl. Abbildung 2.1).

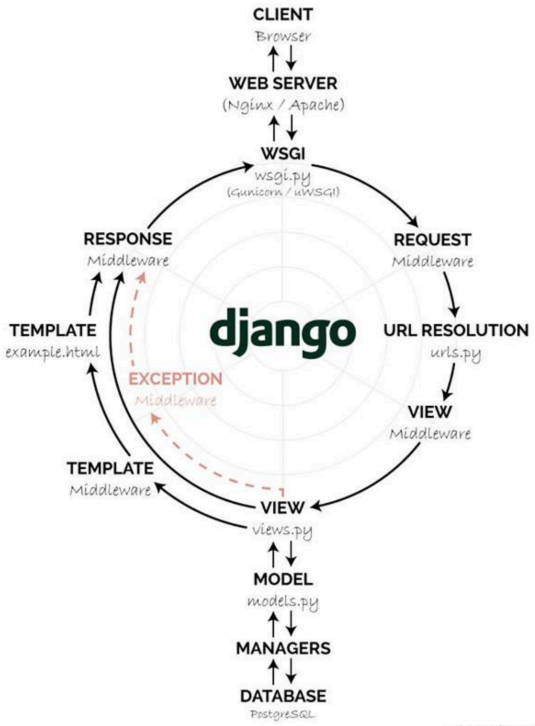

Abbildung 2.2. Request-Response-Kreislauf des Django Frameworks

2.1.1 Besonderheiten

Das Django-Framework bringt einige Besonderheiten mit sich, die beim implementiern des Prototypen von Bedeutung sind. Diese werden im Folgenden beschrieben. Die Administratoroberfläche ist eines der hilfreichsten Werkzeugen des gesamten Frameworks. Es stellt die Metadaten der Modelle aus dem Code visuell dar. Verifizierte Benutzer können die Daten nicht nur schnell erfassen, sondern diese auch editieren und verwalten. Das Recht,

das Admin-Backend uneingeschränkt zu benutzen, ist dem sogenannten superuser vorenthalten. Dieser kann beim erstmaligen zuweisen nur über die Kommandozeile eingerichtet werden. Ist bereits ein superuser vorhanden, kann dieser im Admin-Backend weiteren Benutzern den gleichen Handlungsfreiraum einräumen. Zudem gibt es noch weitere Stufen der Zugangsberechtigungen, Staff- und Active-Status, die für eine breitere Gruppe von Benutzern geeignet ist. Um die gestaffelten Zugangsberechtigungen auch auf der Website umsetzen zu können, stellt Django ... zur Verfügung. Natürlich lassen sich die Dekorator auch für andere Zwecke vor Funktionen paltzieren. Sicherheit

2.2 Erweiterungen

Durch das hohe Ansehen, dass Django in der Branche genießt ist der Pool an Erweiterungen groß.

pip?

allg Erweiterungen und dann genauer Taggable Manager

2.2.1 Taggable-Manager

Django-taggit ist eine Erweiterung von Alex Gaynor, einem Entwickler aus Washingtion DC. Das Add-on ermöglicht das Verwenden von Tags die automatisch mit einem eindeutigen Slug versehen werden. Um dieses zu installieren wird der folgende Befehl in die Kommandozeile eingefügt:

```
$ pip install django-taggit
```

Im model.py wird das Feld tag neu erstellt und als Taggable Manager definiert. Außerdem muss taggit in der settings.py Datei unter INSTALLED_APPS ergänzt werden. Um dem Programm zu sagen, dass nun eine neue Liste der Datenbank hinzugefügt werden muss, werden folgende Befehle in die Kommandozeile eingefügt:

```
$ python3 manage.py makemigrations
$ python3 manage.py migrate
```

Im Admin-Backend kann nun geprüft werden, ob das neue Feld in die Datenbank integriert wurde. Neue Tags können in das Textfeld eingetragen werden. Der Parser verarbeitet jedes Wort, dass durch ein Komma oder ein Leerzeichen getrennt ist als ein Tag. Soll ein dieses jedoch aus mehreren Wörtern bestehen so müssen diese zwischen Anführungszeichen stehen. StandardmäSSig unterscheidet der Taggable Manager zwischen GroSS- und Kleinschreibung, ist also case sensitive. Ändern kann man das, indem der Boolean TAGGIT_CASE_INSENSITIVE in der settings.py auf True gestellt wird.

2.3 Bootstrap

Kapitel 3

Prototyp

3.1 Organisation

3.1.1 Verwaltung im Administrator-Backend

3.1.2 Berechtigung der User

3.2 Funktion

3.2.1 Abonnieren

3.2.2 Filtern

3.2.3 Benachrichtigung

Kapitel 4

Ergebnis

4.0.1 Evaluierung

Kapitel 5

Zusammenfassung und Ausblick

Zusammenfassung...

Referenzen

[Lei13] Ingo Leipner. Stress fur beschaftigte: Wie kann man die e-mail-flut bekampfen. 2013. http://www.mz-web.de/wirtschaft/e-mail-flut-mails-bearbeiten-kommunikation-stress-zeit-sparen.

[Ndu17] Nnenna Ndukwe. Python is the back-end programming language of the future and heres why. 2017. https://medium.com/@nnennahacks/https-medium-com-nnennandukwe-python-is-the-back-end-programming-language-of-the-future-heres-why.

[She09] Alexy Shelest. Model view controller, model view presenter, and model view viewmodel design patterns. 2009. https://www.codeproject.com/Articles/42830/Model-View-Controller-Model-View-Presenter-and-Mod.