TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Entwicklung einer internen Social Media Plattform mit personalisierbarem Dashboard fur Studierende

Esther Beate 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, 3. November 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, 3. November 2018

Esther Beate Kleinhenz
E-mail: kleinhenzes60188@th-nuernberg.de

Studiengang Media Engineering
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

Abbildungsverzeichnis

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"(vgl. [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. —genauer sagen woher ich mir sicher bin, dass das postfach überlastet ist 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. —Forschungsfrage

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. —zu kurz

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 Serverseitig anwendbar ist. Der Programmaufbau Pythons macht den Code leicht lesbar und der einfache Syntax ermöglicht einen strukturierte Implementierung der Website(vgl. [Ndu17]). Durch den modularen Aufbau ist es selbst für unerfahrene Entwickler möglich die Sprache schnell zu erlernen. Darüber hinaus bringt Python verschiedene Web-Service Tools mit sich, die beim implementieren einer Website viel Zeit sparen und das Aneignen von komplexen Protokollen redundant machen (vgl. [Sol17]). Das dazugehörige Framework Django reduziert den Entwicklungsaufwand eines Prototypen erheblich und ist daher als zielführendes Framework die beste Wahl

2.1 Django

Django ist ein Web-Framework, das eine schnelle, strukturierte Entwicklung ermöglicht und dabei ein einfaches Design beibehält. Der darin enthaltene Model-View-Presenter (MVP) kann, ähnlich wie der Model-View-Controller, die Interaktionen zwischen Model und View, die Auswahl und Ausführung von Befehlen und das Auslösen von Ereignissen steuern (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 vgl. [She09]. Durch die direkte Bindung von Daten und View, geregelt durch den Presenter, wird die Codemenge der Applikation stark reduziert.

Der Prozess vom Anfragen der URL über den Server, bis hin zur fertig gerenderten Website kann wie folgt vereinfacht dargestellt werden.

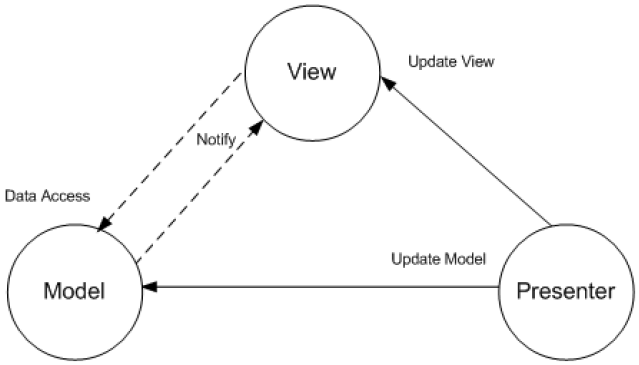

Abbildung 2.1. Vereinfachter MVP ([She09])

Der User gibt eine URL im Browser ein und sendet sie an den Web-Server. Das Interface WSGI am Web-Server verbindet diesen mit dem Web-Framework, indem es den Request zum passenden Objekt weiterleitet. Hier wird der Applikation eine Callback-Funktion zur Verfügung gestellt (vgl. [Kin17]). Außerdem werden folgende Schritte durchgeführt:

- Die Middleware-Klassen aus der settings.py werden geladen
- Die Methoden der Listen Request, View, Response und Excpetion werden geladen
- Die angeforderte URL wird aufgelöst

Der WSGI-Handler fungiert also als Pförtner und Manager zwischen dem Web-Server und dem Django-Projekt. Um die URL, wie weiter oben erwähnt, aufzulösen, benötigt WSGI einen *urlresolver*(vgl.). Durch die explizite Zuweisung der vorhandenen Seiten, kann dieser über die regulären Ausdrücke der url.py-Datei iterieren. Gibt es eine Übereinstimmung, wird die damit verknüpfte Funktion in der View (view.py) aufgerufen. Hier ist die gesamte Logik der Website lokalisiert. Es ist 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 und, wenn nötig, mit einfachen Python-Befehlen angepasst werden. Nun kann das Template, die vom WSGI-Framework zur Verfügung gestellte Callback-Funktion befüllen und einen Response an den Web-Server schicken. Die fertige Seite ist beim Klienten im Browserfenster zum rendern bereit (vgl. [Kin17], Abbildung 2.2.).

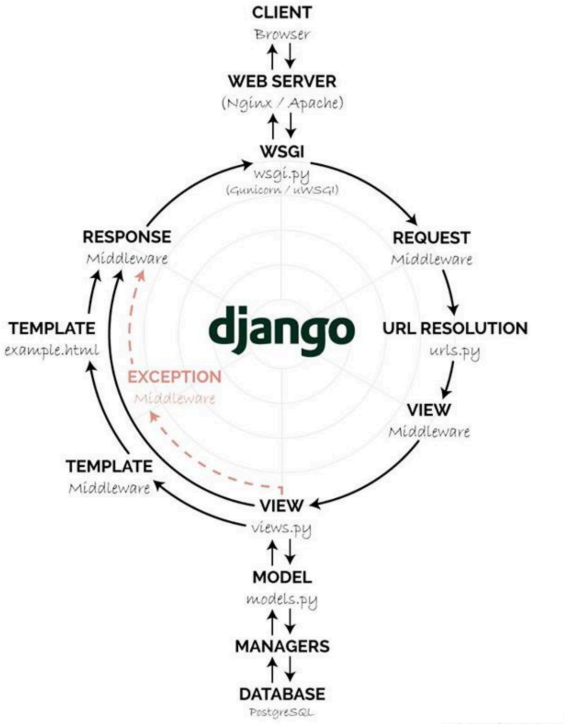

Abbildung 2.2. Request-Response-Kreislauf des Django Frameworks ([Nev15])

2.1.1 Besonderheiten Django's

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 verschiedene Decorator zur Verfügung. Soll eine bestimmte Seite nur von eingeloggten Benutzern besucht werden, so importiert man die Decorator des, von Django zur Verfügung gestellten, Authentifizierungssystems mit

```
from django.contrib.auth.decorators import login_required
```

Vor der Definition der Funktion wird dann folgende Zeile ergänzt:

@login_required

Natürlich lassen sich Decorator auch für andere Zwecke vor Funktionen platzieren. Unter Anderem werden so die Views vor möglichen Angriffen, wie Cross-Site-Scripting, geschützt.

Durch den einfachen Aufbau ist es außerdem möglich diese selbst zu implementieren. Ein einfaches Beispiel wäre das prüfen des, an die Funktion übergebenen, Parameter. Sollen nur positive Zahlen verarbeitet werden, so kann der Decorator alle anderen Eingaben abfangen.

2.1.2 Virtuelle Umgebung

Wird eine prototypische Anwendung gestartet, ist es notwendig, verschiedensten Module zu verwenden und zu testen. Die Versionen dieser spielen hierbei eine entscheidende Rolle, um Konflikte zu vermeiden [Fou18]. Um diesem Problem vorzubeugen, wird eine virtuelle Umgebung implementiert. Diese besitzt einen eigenen Projektpfad, beinhaltet alle nötigen Pakete und Bibliotheken, und lässt sich nach dem Einrichten im Terminal starten. Die folgende Abbildung (2.4) zeigt das Erstellen eines neuen Ordners, das Erstellen der virtuellen Umgebung und den Aktivierungsbefehl. Ist der Name des Environment in Klammern am Kommandozeilenanfang, bedeutet das, diese ist jetzt aktiv.

```
[Esthers-MBP:~ Esthi$ mkdir thesis-test
[Esthers-MBP:~ Esthi$ python3 -m venv test
[Esthers-MBP:~ Esthi$ source test/bin/activate
 (test) Esthers-MBP:~ Esthi$
```

Abbildung 2.3. Erstellen der virtuelle Umgebung im Terminal

Um die Pakete und Module kollisionsfrei zu installieren ist es empfehlenswert einen Package-Manager zu verwenden. Mit pip, „rekursives Akronym für pip Install Packages"(vgl. [Wei17, K. 23.1]), können diese installiert, geupdated und gelöscht werden. Außerdem kann der Manager Abhängigkeiten, wenn nötig, überschreiben und optimieren. Falls ein, sich von der neuesten Version unterscheidendes, Programm installiert werden soll, so ist dies ebenso möglich.

2.1.3 Lightweight Directory Access Protocol

Das ldap, Lightweight Directory Access Protocol, muss als Erweiterung in die hier bearbeitende Bachelor-Arbeit eingebunden werden, um später die Login-Daten im

Hochschulinternen Netz abfragen zu können. Dies ist ein Internetprotokoll, welches die Kommunikation mit dem Active Directory verwaltet. Es wird eingesetzt um Benutzer so schnell und effizient wie möglich durch eine bereits existierende Datenbank abzufragen und zu authentifizieren. Der Aufbau ist mit einem Baum zu vergleichen (vgl. Abbildung 2.4.). Die Wurzel besteht aus sehr allgemeinen Informationen, umso näher man den Blättern kommt, umso spezifischer werden diese. Ein Objekt in der Struktur wird durch einen einmaligen Namen identifiziert, der aus den gesamten hinterlegten Informationen besteht. Der Name für den in der Abbildung 2.4 dargestellten Baum wäre „cn=John Doe, ou=People, dc=sun.com“(vgl. [Sch17]).

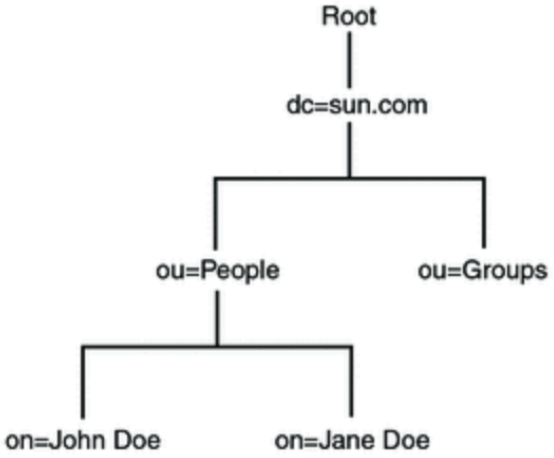

Abbildung 2.4. Beispiel eines LDAP-Trees ([Orc10])

2.2 Erweiterungen

Django bringt viele hilfreiche Erweiterungen mit sich, die mit einem Packagemanager einfach in die virtuelle Umgebung geladen werden können. Um das passende Add-on für ein Projekt zu finden, bietet die Plattform djangopackages.org alle Erweiterungen in übersichtlichen Tabellen mit Eigenschaften und Bewertung an.

2.2.1 Taggable-Manager

Django-taggit ist eine Erweiterung, die das Verwenden von Tags vereinfacht. Der darin enthaltene Taggable Manager verwendet Django's Contenttype Framework, welches per Default verwendet wird, um die Modelle der Applikation zu verfolgen und diese durch generische Beziehungen zu verknüpfen. Die Felder app_label und model machen die Modelle eindeutig zuweisbar. Instanzen des Contenttypes präsentieren und speichern die Informationen und Erstellen automatisch neue Instanzen,

wenn Modelle hinzugefügt werden. Zudem stehen Methoden zur Verfügung, die das Abrufen und Arbeiten mit Instanzen der einzelnen Modelle erleichtern.

Der Taggable-Manager ist jedoch nicht an das Contenttype-Framework gebunden ([Her16]). Durch die Verwendung eines echten Fremdschlüssels, kann zum Beispiel die Performance und Referenzgarantie verwirklicht werden. Dazu ist lediglich die Erstellung einer eigenen lookup-Tabelle notwendig, die die Entitäten zweier Tabellen direkt verlinkt, anstatt diese generische zu verbinden. Weiterführend können Modelle in einem benutzerdefinierten Modell vereint werden, sodass er Zugriff auf diese einheitlich geschieht. AuSSerdem ist es möglich Primary-Keys zu verwenden, die nicht aus ganzen Zahlen bestehen, sondern aus Buchstaben und Wörtern.

Um django-taggit zu installieren wird der folgende Befehl in die Kommandozeile eingefügt([Gay10]):

$ 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 unter INSTALLED_APPS ergänzt werden. Um dem Programm mitzuteilen, 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 dieses jedoch aus mehreren Wörtern bestehen so müssen diese mit Anführungszeichen umfasst werden. StandardmäSSig unterscheidet der Taggable Manager zwischen GroSS- und Kleinschreibung, Tags sind also case sensitive. Ändern kann man das, indem der Boolean TAGGIT_CASE_INSENSITIVE in der settings.py auf True gestellt wird.

2.2.2 Hilfsbibliotheken

—warum habe ich diese bibs gealden (beschreibung evtl in prototyp) Weitere Addons werden geladen um kleinere Funktionen der Website einfach umsetzen zu können. Zu diesen gehört django-taggit-templatetags, welches durch die Einbindung im HTML-File die Tags der Applikation als Liste ausgibt. Außerdem lassen sich die eingepflegten Tags als Cloud visualisieren. Kommen bestimmte Tags öfters vor als andere, so werden sie entsprechend größer dargestellt.

Django-hitcount dient zum zählen der Besucher einer Seite ([Tim15]). Dies lässt sich auf drei verschiedene Arten in die Applikation einbinden. Der schnellste Weg ist

die Darstellung der Besuche mit Hilfe eines Template Tags im HTML-File. Möchte man die Anzeige aber individueller gestalten so kann durch das integrieren der HitCountDetailView in views.py die Variable hitcount verwenden und im Frontend ausgeben. Eine weiter Möglichkeit ist das Erweitern oder neu Erstellen eines Models im Django Backend. Von dort kann auf das neue Feld im Django-Admin-Backend zugegriffen werden, ebenso wie in der View und im Template. Die im Add-on integrierten Einstellungen, die in der settings.py ergänzt werden müssen, ermöglichen unter Anderem das begrenzen der Lebensdauer des Zählers, bevor dieser zurück gesetzt wird.

Um das Versenden und Verwalten von E-Mails in Django zu realisieren eignet sich django-post-office ([Ong18]). Nach der Installation kann im Admin-Backend ein E-Mail-Templates angelegt werden, Anhänge verwaltet und das Senden dieser im Log überprüft werden. Die Benachrichtigungen können asynchron versendet werden mit einem integrierten Planungsmanager. Der Inhalt kann Text oder HTML-basiert sein und in mehreren Sprachen hinterlegt werden.

2.3 Bootstrap

Eine umfangreiche Website einheitlich zu gestalten ist oft sehr komplex und zeitaufwendig. Die Entwickler von Twitter haben deshalb, zunächst Firmenintern, an einem neuen Verwaltungswerkzeug gearbeitet, das mehrere Bibliotheken zusammenführen sollte. Sie merkten, das die neue Bibliothek, die daraus entstand, nicht nur auf Ihre eigene Website anwendbar, sondern so flexible ist, dass jede Art von Website davon profitieren könnte (vgl. [Ott11]). Die Open-Source-Bibliothek, die auf GitHub abrufbar ist, wird seitdem von vielen Programmierern weiterentwickelt und ist somit stark gewachsen. Version 2.0 verfügt außerdem über die Fähigkeit Websites responsive auf verschiedenste mobile Endgeräte anzupassen (vgl. [Ott12]).

Das Bootstrap-Paket beinhaltet vorgefertigte Cascading Stylesheets, kurz CSS, die Farben, Schriftarten und viele weitere Stildefinitionen. Zudem befinden sich auch Erweiterungen des JavaScript-Frameworks jQuery in der Bibliothek, die weiterführende Funktionen beinhalten wie zum Beispiel Filter oder Dropdown-Menüs. Das Paket kann einfach eingebunden werden im head-tag einer HTML-Datei (vgl. Abbildung 2.3). Das bedeutet, dass Media-Queries oder ähnliche Methoden nicht mehr nötig sind, nicht nur um eine Website mobilfähig zu machen, sondern auch kompatibel für die verschiedensten Browser (vgl. [Boo12]).

Durch das Einbinden von Bootstrap in einer HTML-Datei werden einige Styles bereits automatisch auf die darin vorkommenden Tags, wie Links und Überschriften,

```
<head>
    <link href="{% static 'bootstrap/css/bootstrap.css' %}" rel="stylesheet">
</head>
```

Abbildung 2.5. Einbindung von Bootstrap in einer HTML-Datei

angewendet. Dies ist jedoch nur ein sehr kleiner Teil den die Bibliothek zur Verfügung stellt. Möchte man Bootstrap umfangreich nutzen so lassen sich die Stildefinitionen mit Klassen oder ID's in diverse HTML-Tags eintragen (vlg. Abbildung 2.4.).

```
<div class="content container">
    <div class="row">
        <div class="col-md-8">
            {% block content %} {% endblock %}
        </div>
    </div>
</div>
```

Abbildung 2.6. Bootstrap-Klassen in HTML-Tag

Möchte man bestimmte gestalterische Eigenschaften von Bootstrap überschreiben muss eine eigens verfasste CSS-Datei nach der Verlinkung von Bootstrap in die Website eingebunden werden. Der Parser liest die Datei von oben nach unten, Links nach Rechts. Liest dieser also zu erst die Bootstrap Bibliothek und speichert diese, so überschreiben die Styles die danach kommen, die bereits gelesenen Eingaben. Die Styles, die inline auf ein Tag angewendet werden sind somit die bestimmenden Eigenschaften. Natürlich sollten Stildefinitionen niemals inline eingepflegt werden, da dies zu einem sehr unübersichtlichen und wartungsintensiven Code führen.

2.4 Cron

Kapitel 3

Prototyp

Um zu beweisen, wie eine Spcial Media Plattform die Mail-Flut der Efi-Fakultät reduzieren kann, wird ein Prototyp implementiert.

3.1 Organisation

Grundlegender Aufbau der Website, Verwaltung der Daten evlt auf nochmal Taggable-Manager (ManyToMany) ...

3.1.1 Verwaltung im Administrator-Backend

Näher auf CustomUserModel eingehen, Diagramm erstellen und einbinden

3.1.2 Berechtigung der User

Welche Berechtigungen gibt es im Prototyp, welche werden vom Active Directory übernommen?

3.2 Funktionen

Nötige Funktionen

3.2.1 Abonnieren

Tags als eingeloggter User abonnieren und verwalten Front-end und Admin-Backend?

3.2.2 Filtern

Tag-map? Filtern nach abonnierten Posts, alle Posts und Posts mit bestimmten Tags

3.2.3 Benachrichtigung

Mail-Benachrichtigung wöchentlich

Kapitel 4

Ergebnis

4.0.1 Evaluierung

Eine weitere hilfreiche Erweiterung ist pylint. Das Tool sucht nicht nur nach Fehlern im Code, sondern versucht diesen sauber und einheitlich zu gestalten. Hierbei wird auf den Code-Standard PEP-8 geprüft [Dix18]. Die folgende Liste zeigt eine Kurzfassung der wichtigsten Regeln:

- Einrückung, meist 4 Leerzeichen
- Maximale Zeichenanzahl pro Zeile
- Zwei Leerzeile zwischen Klassen und Funktionen
- Eine Leerzeile zwischen Methoden innerhalb einer Klasse
- Leerzeichen in Ausdrücke und Anweisungen vermeiden
- Die Reihenfolge der Importe: Standartbibliotheken, Drittanbieterbibliotheken, Lokale Anwendungen
- Konventionen der Namensgebung von Funktionen, Modulen usw.

Natürlich sind dies Vorgaben, die eingehalten werden können, aber nicht notwendig sind um den Code fertig kompilieren und ausgeben zu lassen.

Kapitel 5

Zusammenfassung und Ausblick

Zusammenfassung...

Referenzen

[BA11] Twitter Inc Bootstrap Authors. Bootstrap repository. 2011. https://github.com/twbs/bootstrap.

[Coo10] Oracle Cooperation. About ldap. 2010. https://docs.oracle.com/cd/E19182-01/820-6573/6nht2e5a4/index.html.

[Dix18] Chitrank Dixit. Pep-8 tutorial: Code standards in python. 2018. https://www.datacamp.com/community/tutorials/pep8-tutorial-python-code.

[FMS17] Andreas Donner Frank-Michael Schlede, Thomas Bär. Was ist ldap (lightweight directory access protocol)? 2017. https://www.ip-insider.de/was-ist-ldap-lightweight-directory-access-protocol-a-581204/.

[Fou18] Python Software Foundation. Virtual environments and packages. 2018. https://docs.python.org/3/tutorial/venv.html.

[Gay10] Alex Gaynor. Exploring django-taggits data model. 2010. https://django-taggit.readthedocs.io/en/latest/getting_started.

[Her16] Stephan Herzog. Model view controller, model view presenter, and model view viewmodel design patterns. 2016. https://medium.com/sthzg/a-short-exploration-of-django-taggit-bb869ea5051f.

[Kin17] Adam King. Django middlewares and the request/response cycle. 2017. https://medium.com/zeitcode/django-middlewares-and-the-request-response-cycle-fcbf8efb903f.

[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.

[Ong18] Selwin Ong. django-post_office git repository. 2018. https://github.com/ui/django-post_office/blob/master/AUTHORS.rst.

[Ott11] Mark Otto. Bootstrap from twitter. 2011. https://blog.twitter.com/developer/en$_u$s/a/2011/*bootstrap – twitter.html*.

[Sha09] Shabda. Understanding decorators. 2009. https://www.agiliq.com/blog/2009/06/understanding decorators/.

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

[Sol17] Mindfire Solutions. Advantages and disadvantages of python programming language. 2017. https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-python-programming-language-fd0b394f2121.

[Tim15] Damon Timm. django-hitcount documentation. 2015. https://django-hitcount.readthedocs.io/en/latest/overview.html.

[Wei17] Michael Weigend. *Python GE-PACKT*. 2017. Kapitel 23.1.