|  |
| --- |
| 04. April 2018, 12:00 Uhr |
| IPA Bericht |
| Jusuf Zera |

**Vorwort**

Inhalt

[1 Obligatorischer Teil 4](#_Toc510605580)

[1.1 Kurzfassung 4](#_Toc510605581)

[1.2 Ausgangslage 5](#_Toc510605582)

[1.3 Detaillierte Aufgabenstellung 5](#_Toc510605583)

[1.3.1 Login 6](#_Toc510605584)

[1.3.2 Benutzerprofil 6](#_Toc510605585)

[1.3.2.1 Stammdaten ändern 6](#_Toc510605586)

[1.3.2.2 Passwort ändern 6](#_Toc510605587)

[1.3.3 Testbenutzer für interne Registrierung erstellen 6](#_Toc510605588)

[1.3.4 Testing 7](#_Toc510605589)

[1.3.5 Dokumentation 7](#_Toc510605590)

[1.3.6 Design 7](#_Toc510605591)

[1.3.7 Versionierung 7](#_Toc510605592)

[1.3.8 Mittel und Methoden 8](#_Toc510605593)

[1.3.9 Vorkenntnisse 8](#_Toc510605594)

[1.3.10 Vorarbeiten 8](#_Toc510605595)

[1.3.11 Neue Lerninhalte 8](#_Toc510605596)

[1.3.12 Arbeiten in den letzten 6 Monaten 9](#_Toc510605597)

[1.4 Projektorganisation 10](#_Toc510605598)

[1.5 Zeitplan 11](#_Toc510605599)

[1.6 Arbeitsjournal 12](#_Toc510605600)

[1.6.1 Dienstag, 03. April 2018 12](#_Toc510605601)

[1.6.2 Mittwoch, 04. April 2018 13](#_Toc510605602)

[1.6.3 Donnerstag, 05. April 2018 14](#_Toc510605603)

[1.6.4 Freitag, 06. April 2018 15](#_Toc510605604)

[1.6.5 Montag, 09. April 2018 16](#_Toc510605605)

[1.6.6 Dienstag, 10. April 2018 17](#_Toc510605606)

[1.6.7 Mittwoch, 11. April 2018 18](#_Toc510605607)

[1.6.8 Donnerstag, 12. April 2018 19](#_Toc510605608)

[1.6.9 Freitag, 13. April 2018 20](#_Toc510605609)

[1.6.10 Montag, 16. April 2018 21](#_Toc510605610)

[2 Projektdokumentation 22](#_Toc510605611)

[2.1 Informieren 22](#_Toc510605612)

[2.1.1 Auftrag 22](#_Toc510605613)

[2.1.2 Vorgaben 23](#_Toc510605614)

[2.1.2.1 Python 23](#_Toc510605615)

[2.1.2.2 Django 23](#_Toc510605616)

[2.2 Planen 24](#_Toc510605617)

[2.2.1 Vorgehensweise nach MTV-Pattern 24](#_Toc510605618)

[2.2.1.1 Models 25](#_Toc510605619)

[2.2.1.2 Templates 25](#_Toc510605620)

[2.2.1.3 Views 25](#_Toc510605621)

[2.2.2 Vorgehensweise nach Django-Administratoranwendung 26](#_Toc510605622)

[2.2.3 Problemanalyse 26](#_Toc510605623)

[2.2.3.1 Datenverlust 26](#_Toc510605624)

[2.2.3.2 Programmierprobleme 27](#_Toc510605625)

[2.2.3.3 Zeitprobleme 27](#_Toc510605626)

[2.2.3.4 Browserkompatibilität 27](#_Toc510605627)

[2.2.4 Sicherung 28](#_Toc510605628)

[2.2.4.1 USB-Stick 28](#_Toc510605629)

[2.2.4.2 Github 28](#_Toc510605630)

[2.2.5 Soll/Ist-Vergleich der Systemlandschaft 29](#_Toc510605631)

[2.3 Entscheiden 30](#_Toc510605632)

[2.4 Realisieren 30](#_Toc510605633)

[2.5 Kontrollieren 30](#_Toc510605634)

[2.5.1 Testfälle 30](#_Toc510605635)

[2.6 Auswerten 30](#_Toc510605636)

[2.7 Abschlussbericht 30](#_Toc510605637)

[2.8 Quellenverzeichnis 30](#_Toc510605638)

[2.9 Glossar 30](#_Toc510605639)

# 1 Obligatorischer Teil

## Kurzfassung

Die Schweizerische Bankiervereinigung befasst sich intensiv mit der Digitalisierung und hat sich daher entschieden, das Projekt „M17“ (Migration 2017) zu lancieren. Die heutigen Systeme wurden von der Bankiervereinigung vor 14 Jahren selbst entwickelt und entsprechen nicht dem Standard neuster Technologien. Ausserdem sind alle Module miteinander verbunden, sodass die Unabhängigkeit der einzelnen Systeme nicht gewährleistet ist. Daher hat sich die Geschäftsleitung entschieden, die Systemlandschaft von Grund auf neu aufzubauen.

Die Systemlandschaft wird demnach modular und nicht zentral erstellt. Dafür verantwortlich wird die Autorisierungsplattform einer externen Firma, welche die Berechtigungen für die einzelnen Module regelt. Den Benutzerinnen und Benutzer wird es nach wie vor möglich sein, ihre Daten selbst zu mutieren. Das Selbstverwaltungs-Tool der externen Firma entspricht aber nicht den Anforderungen der Schweizerischen Bankiervereinigung. Somit wird dieser Teil des Projekts „M17“ von mir, Jusuf Zera, entwickelt.

Das Ergebnis wird eine Webapplikation, die mit der Datenbank (LDAP-Datenbank) der Autorisierungsplattform verbunden ist. Die Benutzerinnen und Benutzer werden sich mit ihrer E-Mail-Adresse und ihrem Passwort anmelden, um anschliessend ihr Profil mit allen Daten zu sehen. Veränderbare Daten können auf einer separaten Seite mutiert und gespeichert werden. Bei Fehleingaben wird den Benutzerinnen und Benutzer ersichtlich, was angepasst werden muss, um den Prozess des Speicherns abzuschliessen. Ausserdem kann das Passwort beliebig geändert werden, nachdem der Benutzende die Voraussetzungen erfüllt hat. Die Abmeldung erfolgt über einen einfachen One-Click. Zur Weiterentwicklung nach der IPA wird eine Registrierungsseite erstellt, die Benutzer in einer internen Datenbank erstellt und nicht in der LDAP-Datenbank der externe Firma. Grund dafür ist, dass die Registrierung seitens Geschäftsleitung direkt über die Autorisierungsplattform erfolgen soll.

## 1.2 Ausgangslage

Der Wandel in die Digitalisierung ist ein wichtiger Schritt in der Zukunft der Schweizerischen Bankiervereinigung (SBVg). Aus diesem Grund wurde 2017 das Projekt „M17“ (Migration 2017) initialisiert. Das Projekt beinhaltet die Migration der internen und externen Systemlandschaft. Die gegenwärtig eingesetzten Systeme entsprechen nicht mehr den neuesten Technologien, zudem ist die Usability der Bewirtschaftungssoftware nicht mehr gewährleistet. Die Struktur des Projekts „M17“ ist modular aufgebaut, damit alle Systeme unabhängig voneinander funktionieren oder allenfalls ersetzt werden können. Die jetzige Software wurde im Jahre 2006 von der SBVg selbst entwickelt.   
Die neue Systemlandschaft beinhaltet u.a. eine webbasierte Collaboration-Plattform, auf der Gremienmitglieder als Vertretung seitens Banken interaktiv zusammenarbeiten. Dazu hat die SBVg bereits jetzt eine Portallösung aufgeschaltet, auf der berechtige Personen (vorzugsweise Bankmitarbeitende) Zugriff haben und entsprechende Informationen vorfinden. Die Portallösung soll im Rahmen des Projekts M17 erneuert werden. Eine Komponente davon bildet die Benutzerverwaltung, d.h. den berechtigen Personen werden entsprechende Berechtigungen erteilt. Um inskünftig die Personen, gegenwärtig deren 22‘000, zu verwalten, wird eine Autorisierungsplattform entwickelt, die die Berechtigungen aller Systeme steuert. Bei der von der SBVg verwendeten Autorisierungsplattform entspricht das Selbstadministrations-Tool nicht den Anforderungen der SBVg. Die fehlenden Komponenten werden von der SBVg entwickelt.

## 1.3 Detaillierte Aufgabenstellung

Das Resultat der IPA soll eine webbasierte Selbstverwaltung für Benutzer beinhalten. Die Änderungen werden in der Mitarbeiter-Datenbank (LDAP-DB) gespeichert. Die Selbstverwaltung der Benutzenden muss an diese Datenbank angebunden werden und kommunizieren können. Die Mutationen werden durch das Speichern in die LDAP-DB übernommen.  
  
Ziel des Projekts ist eine benutzerfreundliche Webapplikation, die auf allen Endgeräten wie Smartphones, Tablets und Computer kompatibel ist und auf dem vorgegebenen Design unserer Kommunikationsabteilung basiert.

User-Modell:  
Die Webapplikation ist für Portaluser der Banken gedacht, die Ihre Daten selbständig und jederzeit ändern wollen. Sie sind in unserer LDAP-DB aufgelistet und Mutationen werden dort direkt übernommen. In der Applikation selbst werden keine Berechtigungen gegeben, welche Module etc. genutzt werden können, da dies von der Autorisierungsplattform gemacht wird.   
Es handelt sich dabei um ein Selfservice-Formular, das jedem Benutzer ermöglicht, die gleichen Daten zu ändern (z.B. Name, Vorname, Strasse). Durch das Login wird sichergestellt, dass jeder Benutzer nur seine eigenen Daten ändern kann. Auf der Profil-Seite sind alle Daten ersichtlich, aber auf der Profil Bearbeiten-Seite nur die Daten, die der Benutzer mutieren kann.

Umfang der Arbeit:

* Startseite mit Login
* Profil-Seite, die alle Daten des Benutzers anzeigt
* Profil bearbeiten-Seite auf der die Stammdaten des Benutzers verändert werden können.
* Passwort ändern-Seite auf dem der Benutzer sein altes Passwort durch ein Neues ersetzen kann.
* Registrierungsseite für Testbenutzer und Weiterentwicklung (nur für Administratoren)

Die Benutzerverwaltung besteht aus folgenden Komponenten:

### 1.3.1 Login

Beim Anmelden muss der Benutzer zwei Felder ausfüllen (E-Mail-Adresse und Passwort). Die Felder werden vom System auf ihre Korrektheit und Übereinstimmung verifiziert. Bei einer Falscheingabe, wird eine Fehlermeldung angezeigt. Mit einem einfachen One-Click meldet sich der Benutzer wieder ab.

### 1.3.2 Benutzerprofil

Nach der Anmeldung wird der Benutzer auf sein Profil weiteregeleitet. 

#### 1.3.2.1 Stammdaten ändern

Auf der „Profil bearbeiten“-Seite kann der Benutzer seine Stammdaten wie Name, Adresse etc. beliebig mutieren. Die Änderungen werden beim Speichern in die LDAP-DB übernommen.

#### 1.3.2.2 Passwort ändern

Auf der Seite „Passwort ändern“ kann der Benutzer sein Passwort ändern. Dazu muss er sein altes Passwort und das neue Passwort doppelt eingeben. Nach einer erfolgreichen Überprüfung des alten Passworts wird das neue Passwort beim Speichern in die DB übernommen. 

### 1.3.3 Testbenutzer für interne Registrierung erstellen

Um die Benutzerveränderungen und Validierung der Felder zu testen, wird zusätzlich eine Registrierungsseite erstellt. Der Benutzer fügt seine E-Mai-Adresse, Name, Vorname und Passwort (2x) etc. ein. Es wird ein neuer Benutzer erstellt. Dieser wird jedoch nicht in der LDAP-DB abgespeichert, sondern in einer SQLite3-DB, um Konflikte mit den Benutzern in der LDAP-DB zu vermeiden. Die Portaluser der Banken werden sich nicht über die Webapplikation registrieren. Dafür ist erneut die Autorisierungsplattform zuständig. Die Webapplikation kann aber nach der IPA weiterentwickelt werden, sodass sich zukünftig Benutzer ihr Konto direkt auf der Webseite erstellen können.   
Diese Funktion hilft sowohl den Usern als auch den Administratoren Probleme und Fehler zu beseitigen und eine einwandfreie Anmeldung und Mutation der Daten zu gewährleisten.   
Die Registrierungsseite wird auf der Startseite zu finden sein und ein benutzerfreundliches Formular enthalten.

### 1.3.4 Testing

Das Testen der Funktionen ist ein wichtiger Teil des Projekts. Um die Funktionalität prüfen zu können, werden die Tests folgend ablaufen:  
  
- Eingabe  
- Erwartetes Resultat  
- Tatsächliches Resultat  
  
In der Webapplikation selbst werden Fehleingaben (z.B. nicht übereinstimmende Passwörter) mit rot-markierten Error-Anzeigen hervorgehoben.  
Die Testfälle werden übersichtlich in Tabellen mit Titeln und Nummern dargestellt. Datum, Zeit und Testperson werden ebenfalls eingetragen, um alles korrekt zu protokollieren. Um die Kompatibilität zu gewährleisten, werden die Tests in verschieden Browsern (Google Chrome, Mozilla Firefox, Internet Explorer) ausgeführt. 

### 1.3.5 Dokumentation

Die IPA-Dokumentation wird auf Basis der Projektmanagement-Methode IPERKA (Informieren, Planen, Entscheiden, Realisieren, Kontrollieren, Ausführen) strukturiert. Die Agilität der Methode steuert unter anderem dazu bei, das Projekt erfolgreich umzusetzen. Ausserdem werden die Testfälle der Applikation eingefügt. Zudem geht die Dokumentation darauf ein, mit welchen zusätzlichen Optionen die Applikation nach der Arbeit erweitert werden könnte.

### 1.3.6 Design

Das Design soll die SBVg repräsentieren und gleich strukturiert sein, wie alle anderen Applikationen. Aus diesem Grund gibt die Kommunikationsabteilung das Grunddesign vor. Um das Design technisch zu implementieren, wird eine Bootstrap-Vorlage verwendet. Damit die Applikation auf allen Endgeräten benutzerfreundlich funktionieren soll, wird sie responsive programmiert.

### 1.3.7 Versionierung

Zur Sicherung des Projekts wird ein GIT-Repository erstellt.

### 1.3.8 Mittel und Methoden

* Python
* Django-Framework
* HTML
* CSS
* Bootstrap
* Atom (Editor)
* UCS - Univention Corporate Server mit LDAP-DB
* GitHub

### 1.3.9 Vorkenntnisse

* HTML & CSS: 3 Jahre
* Python: 3 Monate
* Django: 3 Monate
* LDAP-DB: Basiswissen

### 1.3.10 Vorarbeiten

* Bereitstellung der Serverumgebung (UCS-Server, LDAP-DB, Webserver etc.)
* Vorlage für die Dokumentation der IPA, damit schon ab dem ersten Tag dokumentiert werden kann
* Repository erstellen
* Arbeitsplatz einrichten

### 1.3.11 Neue Lerninhalte

Um Erfahrungen mit der Programmiersprache Python zu sammeln, durfte der Kandidat sich in die Thematik einlesen und vieles durch Üben lernen. Ausserdem wird für die Applikation das Django-Framework verwendet, das neu kennengelernt werden musste. Wie eine LDAP-DB genau aufgebaut ist, gehörte ebenfalls zum Lernprozess, damit das Grundwissen vorhanden ist.

Da die Benutzerverwaltung ein Teil des Projekts „M17“ ist, wurde dem Kandidat ersichtlich, wie man sich bei der Mitarbeit eines solchen Projekts verhält. Dies beinhaltet auch das selbstständige Arbeiten und Evaluieren verschiedener Software.   
Als Quellen standen alle Mitarbeitenden zur Verfügung, die sowohl technisch als auch im Alltag unterstützen. Selbstverständlich wurde auch Hilfe aus dem Internet und der Literatur geholt, um den Radius des Wissens zu vergrössern.

### Arbeiten in den letzten 6 Monaten

Zu den hauptsächlichen Arbeiten gehörten:

* Mitarbeit im Projekt „M17“
* Administration der Datenbank von der SBVg
* Mitarbeit beim Initialisieren der Autorisierungsplattform
* Unterstützung bei diversen Programmierarbeiten
* Evaluation diverser Software
* Unterstützung bei Supportanfragen

Der erste Auftrag war ein 2-tägiger Workshop mit einer externen Firma, die bei uns vor Ort war. Ziel des Workshops war zum einen eine Testumgebung der Autorisierungsplattform aufzusetzen und zum anderen die Anforderungen unsererseits an das LDAP zu definieren, damit die externe Firma das endgültige LDAP-Design erstellen kann.   
  
Der zweite Auftrag ist die Mitarbeit im Projekt „M17“, das laufend weitergeführt wird. Dazu gehört die Strukturdarstellung der neuen Systemlandschaft mit dem Programm Visio, die aufzeigt, wie alle Systeme inskünftig miteinander kommunizieren werden. In diesem Zusammenhang durfte der Kandidat auch das Evaluieren von Software und Anbietern vornehmen.

Tools, die verwendet wurden:

* Word
* Excel
* Powerpoint
* Visio
* UCS – Univention Corporate Server

## Projektorganisation

Als Auftraggeber des Projekts wird die Unternehmensleitung bestimmt, da die Benutzerverwaltung ein Teil des Projekts „M17“ ist. Der Lenkungsausschluss des gesamten Projekts besteht aus Herr Nicolas Hug (Leiter Operations), Herr Lionel Albrecht (Leiter ICT) und Frau Alexandra Arni (Leiterin Online Solutions).

Die technische Projektleitung wird von Herr Detlef Nünnighoff übernommen. Als technischer Mitarbeiter bin ich, Jusuf Zera, für die Planung, Steuerung und Umsetzung der Arbeit verantwortlich, damit keine ungeplanten Komplikationen entstehen.

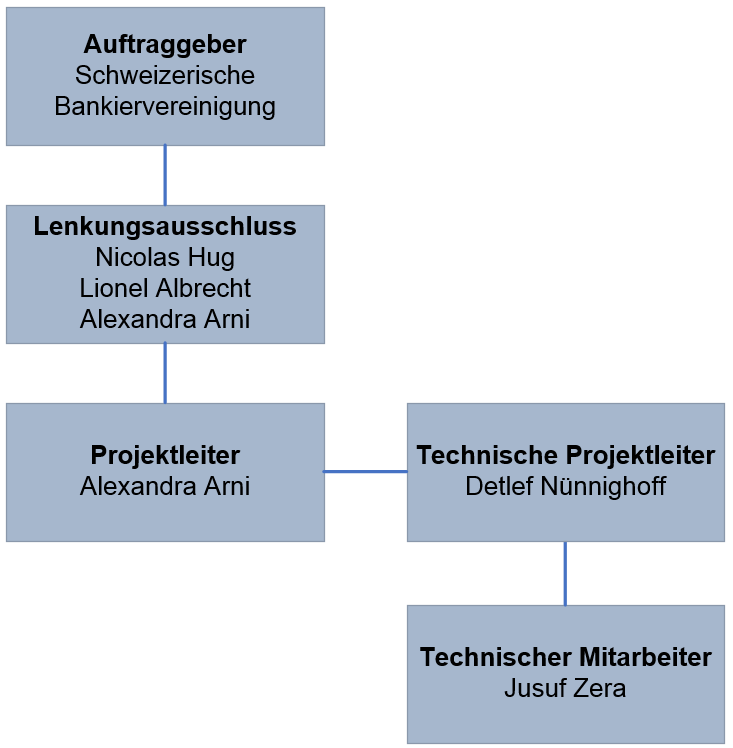


Abbildung: Projektorganisation

Diese Projektorganisation wird auch als „autonome Projektorganisation“ bezeichnet und bietet viele Vorteile für mein Projekt. Die eindeutigen Verantwortlichkeiten, die einfachen Kommunikationswege und das geringe Konfliktpotential der Aufgabenverteilung sind nur einige davon. Als Kontaktperson kann ich mich ohne Umwege an die Projektleitung wenden, um allfällige Fragen zu klären und im Zeitplan zu bleiben.

Da die Applikation von mir realisiert wird, kann es zu keinen grösseren Problemen innerhalb der Organisation kommen. Durch die gut eingeschätzte Zeitplanung beuge ich vor, auf keine Überlastungen während der Arbeitszeit zu stossen.

## 1.5 Zeitplan

Abbildung: Zeitplan der IPA

## 1.6 Arbeitsjournal

### 1.6.1 Dienstag, 03. April 2018

|  |  |
| --- | --- |
| Arbeitszeit | 08:00 Uhr – 10:00 Uhr  Pause  10:05 Uhr – 12:00 Uhr  Mittagspause  12:45 Uhr – 16:50 Uhr |
| Erfolge | * Obligatorischer Teil dokumentiert * Projektorganisation grafisch darstellt * Informationen beschafft * Planung: Vorgehensweisen und Problemanalyse erstellt |
| Misserfolge | Keine |
| Ungeplante Arbeiten | Keine |
| Probleme | Keine |
| Unterstützung Dritter | Keine |
| Soll-/Ist-Vergleich | Der Zeitplan wurde genau eingehalten. |
| Sonstiges | Keine |
| Visum |  |

### 1.6.2 Mittwoch, 04. April 2018

|  |  |
| --- | --- |
| Arbeitszeit |  |
| Erfolge |  |
| Misserfolge |  |
| Ungeplante Arbeiten |  |
| Probleme |  |
| Unterstützung |  |
| Soll-/Ist-Vergleich |  |
| Sonstiges |  |
| Visum |  |

### 1.6.3 Donnerstag, 05. April 2018

|  |  |
| --- | --- |
| Arbeitszeit |  |
| Erfolge |  |
| Misserfolge |  |
| Ungeplante Arbeiten |  |
| Probleme |  |
| Unterstützung |  |
| Soll-/Ist-Vergleich |  |
| Sonstiges |  |
| Visum |  |

### 1.6.4 Freitag, 06. April 2018

|  |  |
| --- | --- |
| Arbeitszeit |  |
| Erfolge |  |
| Misserfolge |  |
| Ungeplante Arbeiten |  |
| Probleme |  |
| Unterstützung |  |
| Soll-/Ist-Vergleich |  |
| Sonstiges |  |
| Visum |  |

### 1.6.5 Montag, 09. April 2018

|  |  |
| --- | --- |
| Arbeitszeit |  |
| Erfolge |  |
| Misserfolge |  |
| Ungeplante Arbeiten |  |
| Probleme |  |
| Unterstützung |  |
| Soll-/Ist-Vergleich |  |
| Sonstiges |  |
| Visum |  |

### 1.6.6 Dienstag, 10. April 2018

|  |  |
| --- | --- |
| Arbeitszeit |  |
| Erfolge |  |
| Misserfolge |  |
| Ungeplante Arbeiten |  |
| Probleme |  |
| Unterstützung |  |
| Soll-/Ist-Vergleich |  |
| Sonstiges |  |
| Visum |  |

### 1.6.7 Mittwoch, 11. April 2018

|  |  |
| --- | --- |
| Arbeitszeit |  |
| Erfolge |  |
| Misserfolge |  |
| Ungeplante Arbeiten |  |
| Probleme |  |
| Unterstützung |  |
| Soll-/Ist-Vergleich |  |
| Sonstiges |  |
| Visum |  |

### 1.6.8 Donnerstag, 12. April 2018

|  |  |
| --- | --- |
| Arbeitszeit |  |
| Erfolge |  |
| Misserfolge |  |
| Ungeplante Arbeiten |  |
| Probleme |  |
| Unterstützung |  |
| Soll-/Ist-Vergleich |  |
| Sonstiges |  |
| Visum |  |

### 1.6.9 Freitag, 13. April 2018

|  |  |
| --- | --- |
| Arbeitszeit |  |
| Erfolge |  |
| Misserfolge |  |
| Ungeplante Arbeiten |  |
| Probleme |  |
| Unterstützung |  |
| Soll-/Ist-Vergleich |  |
| Sonstiges |  |
| Visum |  |

### 1.6.10 Montag, 16. April 2018

|  |  |
| --- | --- |
| Arbeitszeit |  |
| Erfolge |  |
| Misserfolge |  |
| Ungeplante Arbeiten |  |
| Probleme |  |
| Unterstützung |  |
| Soll-/Ist-Vergleich |  |
| Sonstiges |  |
| Visum |  |

# 2 Projektdokumentation

## 2.1 Informieren

### 2.1.1 Auftrag

Der Auftrag besteht darin, das Projekt „M17“ mit meiner Komponente im Rahmen der IPA zu vervollständigen. Die Benutzerverwaltung wird es allen Benutzerinnen und Benutzer möglich machen, ihre Daten selbstständig und beliebig zu mutieren, sobald diese sich über das Login angemeldet haben. Dabei orientiere ich mich an die jetzige Portallösung auf der Hauptseite der Schweizerischen Bankiervereinigung.



Abbildung: Login der SBVg

Das Login wird auch künftig gleich aufgebaut. Da jeder Benutzer nur eine E-Mail-Adresse pro Konto besitzt, muss er sich mit der E-Mail-Adresse und dem dazugehörigen Passwort anmelden. Einen Benutzernamen wird es nach wie vor nicht geben, da die E-Mail-Adresse als solche fungiert.



Abbildung: Mutation der Personendaten im Portal der SBVg

Bei der Benutzerverwaltung sind die einzelnen Felder und Anforderungen, die verbessert werden müssen, zu beachten. Jedes Eingabefeld wird in der künftigen Webapplikation so definiert, dass die Benutzerinnen und Benutzer ohne unnötige Überlegungen wissen, was genau eingegeben werden muss.

### 2.1.2 Vorgaben

Dadurch, dass die Systemlandschaft komplett neu aufgebaut wird, muss ich mich an bestimmte Vorgaben halten.

#### 2.1.2.1 Python

Die Programmiersprache Python wurde aus logischen Gründen ausgewählt. Der grösste Teil der neuen Systeme basiert auf Python, somit wurde festgelegt, dass ich meine Komponente in der gleichen Programmiersprache in die Systemlandschaft einpflege. Da Python schnelle Programmierung möglich macht (wenig Code), kann der Zeitplan sehr gut beibehalten werden.

Vorteile:

* Neue Systeme der SBVg verwenden Python
* Objektorientiert
* Modern
* Lesbar
* Schnelles Programmieren möglich (Zeitplan kann eingehalten werden)
* Ideal für Webentwicklung
* Cross-Plattform (Windows, Linux, MacOS)

Nachteile gibt es wenige bis gar keine, vor allem in Bezug auf mein Projekt ist Python die ideale Programmiersprache, um an eine Lösung zu kommen.

#### 2.1.2.2 Django

Die Nutzung eines Frameworks war mir überlassen. Dabei entschied ich mich für das Django-Framework, da die Benutzerverwaltung eine Webapplikation wird und Django für die Webentwicklung sehr geeignet ist. Die Kompatibilität mit Python ist ebenfalls gewährleistet, sodass problemlos programmiert werden kann.

Ein weiterer Aspekt sind die „Django Batteries“, die in einem Package gespeichert sind und für die Entwicklung genutzt werden können. Der wichtigste „Batterie“ ist der auth-Batterie, das Django-Authentifizierungsframework. Dieses wird mir über die ganze Realisierung der Webapplikation soweit unterstützen, um an das Ziel zu gelangen.

Weitere Vorteile:

* Viele Pakete
* Aktive Weiterentwicklung
* Sicherheit
* Sehr gut dokumentiert

## 2.2 Planen

In der Planung werde ich mich hauptsächlich mit den verschiedenen Vorgehensweisen beschäftigen. Dabei geht es nicht um die Erstellung der Webapplikation, sondern um die sukzessive Art an das Ziel zu gelangen. Die Erstellung selbst, wird in der Realisierungsphase genau erläutert.

Problemanalysen werden ebenfalls erstellt, damit ich bei überraschenden Situationen während der Realisierungsphase richtig reagieren kann und somit im Zeitplan bleibe.

Ein Soll/Ist-Vergleich soll darlegen, wo die Benutzerverwaltung im Projekt „M17“ im Vergleich zur jetzigen Situation implementiert wird.

### 2.2.1 Vorgehensweise nach MTV-Pattern

Django basiert auf einen modifizierten Model-View-Controller (MVC). MVC ist ein Software-Design-Pattern, das eine Applikation in drei miteinander verbundene Teile aufteilen soll.

In Django ist diese folgendermassen aufgebaut.

* Model: Schnittstelle zur Datenbank
* Template: Schnittstelle zwischen Benutzer und der Django-Applikation
* View: Verwaltet Daten, Anwendungslogik und Messaging

Es handelt sich hierbei also um ein MTV-Pattern:

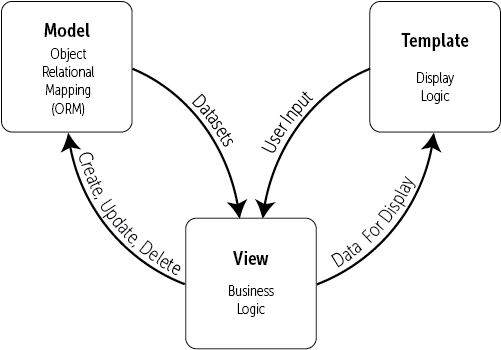


Abbildung: Model-Template-View (MTV) Pattern in Django

#### 2.2.1.1 Models

Django-Models bieten ein Object Relational Mapping (ORM) für die zugrunde liegende Datenbank. ORM ist eine leistungsfähige Programmiertechnik, die das Arbeiten mit Daten und relationalen Datenbanken erheblich vereinfacht.

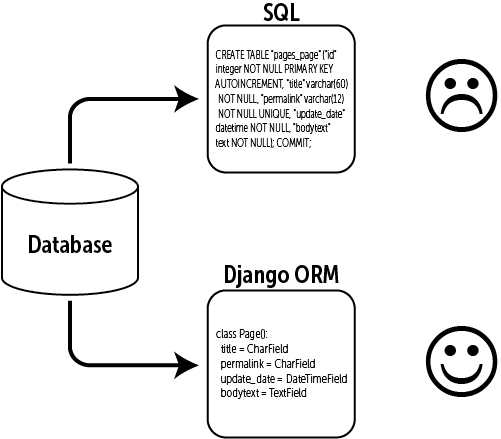


Abbildung: Django-ORM

Basierend auf der heutigen Portallösung müssen die Eingabefelder mit dem Model-Feldern übereinstimmen.

2.2.1.2 Templates

Django-Templates sind üblicherweise HTML-Dateien, um Daten möglichst einfach in einem Webbrowser ersichtlich zu machen. Ich werde mich an folgende Schlüsselargumente halte, damit die Webapplikation einwandfrei läuft.

* Programmlogik von Design trennen
* Redundanzen vermeiden
* Sicherheit gewährleisten

Das Design wird nach Vorgaben der Kommunikationsabteilung der Schweizerischen Bankiervereinigung angepasst.

#### 2.2.1.3 Views

In einer View werden Daten aus einer Datenbank (z.B. LDAP-Datenbank) abgerufen und an ein Template gesendet. Bei einer Webanwendung liefert die View den Inhalt an die Templates der Webseite.

Die Views treffen Entscheidungen darüber, welche Daten an die Templates geliefert werden. Diese können entweder durch Eingaben des Benutzers oder auf andere Prozesse reagieren.

Jede Django-View führt eine bestimmte Funktion aus und besitzt ein zugeordnetes Template. Views werden entweder durch eine Python-Funktion oder eine Methode einer Python-Klasse dargestellt.

### 2.2.2 Vorgehensweise nach Django-Administratoranwendung

Die Django-Administratoranwendung kann meine Models verwenden, um automatisch eine Webseite zu erstellen, mit der ich Datensätze erstellen, anzeigen, aktualisieren und löschen kann (create, view, update, und delete records). Dies spart viel Zeit bei der Entwicklung, sodass ich meine Models einfach testen kann.

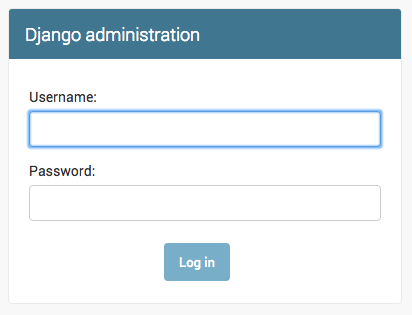
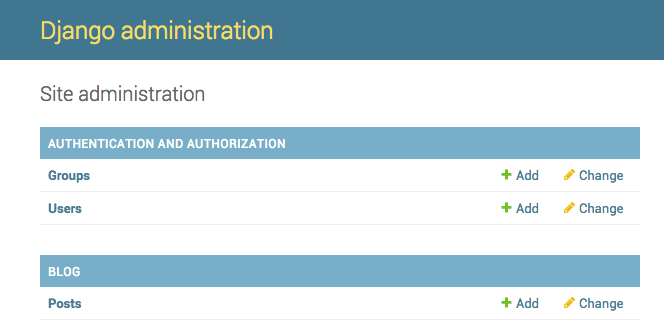
Der Nachteil liegt beim Design der Administratoranwendung. Dieses kann nicht geändert werden. Ich kann lediglich Funktionen und Models hinzufügen.

Abbildung: Design der Django-Administratoranwendung

### 2.2.3 Problemanalyse

#### 2.2.3.1 Datenverlust

Ein Datenverlust während der Arbeit kann viele Ursachen haben und dazu führen, nicht mehr im Zeitplan zu sein.

Ursache von Datenverlust:

* Hardwareprobleme (z.B. PC stürzt ab)
* Softwareproblem (z.B. Editor stürzt ab)
* Unachtsamkeit (z.B. Beim Schliessen von Programmen)

Um dies vorzubeugen, werde ich zwei Mal am Tag und an verschiedenen Orten platzierte Backups erstellen. Damit gemeint sind USB-Stick-Backups und ein Repository auf Github. Die Wahrscheinlichkeit einen grösseren Datenverlust einzufangen ist somit sehr gering.

Nicht nur der Code, sondern auch die Dokumentation muss abgesichert und versioniert werden, damit der Arbeitsablauf bei einem Datenverlust ersichtlich ist.

#### 2.2.3.2 Programmierprobleme

Während der Implementierungsphase ist die Wahrscheinlichkeit hoch, viel Zeit durch Überlegungen und Debuggen des Codes zu verlieren. Da alle Komponenten einen Einfluss auf das Endprodukt haben, muss ich schrittweise arbeiten und nichts überspringen. Der Zeitplan hilft mir dabei, die Übersicht zu behalten, um das Produkt pünktlich zu erstellen.

Bei für mich unlösbaren Probleme oder Sackgassen, kann ich die sehr gut dokumentierten Python- und Django-Dokumentationen nachlesen und diese in meinem Projekt anwenden. Die Community von Python wächst täglich und somit wird auch das Wissen mehrerer Personen geteilt. Foren können dazu beitragen, auf Lösungen für bestimmte Probleme zu stossen und diese in mein Projekt anzuwenden.

#### 2.2.3.3 Zeitprobleme

Bei einer begrenzten Zeit von 10 Arbeitstagen kann es zu Problemen während des Projekts kommen, sodass der Zeitplan nicht eingehalten werden kann. Falls es dazu kommen würde, habe ich Vorgehensweisen, die den Schaden so minimieren, dass das Projekt nicht unvollständig abgegeben wird.

* Prioritäten setzten
* Dokumentation hat Vorrang

Ausserdem ist der Zeitplan sowohl in der Realisierung, als auch in der Dokumentation sehr grosszügig geplant, um Zeitprobleme zu vermeiden.

#### 2.2.3.4 Browserkompatibilität

Da die Webapplikation von rund 22‘000 Benutzerinnen und Benutzer künftig gebraucht wird, muss das Design für alle gängigen Browser kompatibel sein. Jeder Browser ist verschieden und zeigt somit verschiedene Design an.

Das kann ich auch zu meinem Vorteil nutzen, solang die Webseite benutzerfreundlich aussieht und alle Funktionen ausführbar sind.

Die Gegenmassnahme ist die Verwendung eines Frameworks. Das Bootstrap-Framework löst grösstenteils alle Probleme über die verschieden Browser. Ich werde die Vorlage so verwenden, dass das Design der Schweizerischen Bankiervereinigung im Vordergrund steht. Bootstrap wird lediglich die Struktur der Webanwendung darstellen.

Durch die Testfälle in der Kontrollphase wird die Kompatibilität auf allen gängigen Browsern getestet und ersichtlich gemacht, wo noch Anpassungen notwendig sind. Diese können folgend in einer zweiten Version implementiert werden.

### 2.2.4 Sicherung

Das Sichern der Dokumentation und des Sourcecodes muss sichergestellt werden, sodass die Dateien jederzeit abrufbar sind. Dabei muss ich beachten, dass die Struktur der Versionierung für alle Beteiligten klar ist. Die Backups werden zum einen per USB-Stick und zum anderen auf der Plattform Github gespeichert.

#### 2.2.4.2 Arbeitsplatz

Der Arbeitsplatz ist während der gesamten Durchführung des Projekts so gesichert, dass eine konstante Internetverbindung besteht und keine Unberechtigte Person Zugriff auf den von mir benutzen Computer hat.

#### 2.2.4.2 USB-Stick

Das Sichern der Dateien auf einen USB-Stick wird von mir zwei Mal am Tag erstellt. Dabei wird die Naming-Convention der Schweizerischen Bankiervereinigung verwendet.

Beispiel:

* 03042018\_1200\_IPA\_Dokumentation\_JZE.pdf
* 03042018\_1645\_IPA\_Dokumentation\_JZE.pdf

Die Naming-Convention besteht aus Datum, Uhrzeit, Titel und dem Kürzel des Erstellers. Die Übersicht und Nachvollziehbarkeit ist gewährleistet.

#### 2.2.4.3 Github

Das Erstellen eines Git-Repositorys ist vor allem bei der Entwicklung eine ideale Lösung den Sourcecode nicht nur zu speichern, sondern auch den Ablauf durch genaue Bezeichnungen nachzuvollziehen.

Das Repository kann bei Github jederzeit wieder lokal hergestellt werden. Auch ältere Versionen können als einzelnes Projekt wiederverwendet werden, um Abläufe bei der Programmierung darzustellen.

Bei meinem Projekt liegt der Vorteil bei der Weiterentwicklung. Falls diese nicht von mir persönlich vorgenommen wird, kann derjenige Entwickler dank sogenannte „Commits“ nachvollzeihen, wie ich vorgegangen bin. Commits sind Versionen, die einzelne Titel der jeweiligen Änderung aufweisen. Sie werden in der Eingabeaufforderung (cmd.exe) ausgeführt.

Bespiel:

* git commit –m „CSS-Datei angepasst“

Der Entwickler muss demnach nicht Zeit mit suchen verschwenden und kann die angesprochene Datei zum Editieren öffnen.

### 2.2.5 Soll/Ist-Vergleich der Systemlandschaft

Der Soll/Ist-Vergleich zeigt auf, was sich im gesamten Projekt „M17“ verändern wird. Der Hauptfokus liegt dabei auf die „Self-Service-Plattform“, also der Benutzerverwaltung, die von mir im Rahmen der IPA programmiert wird.

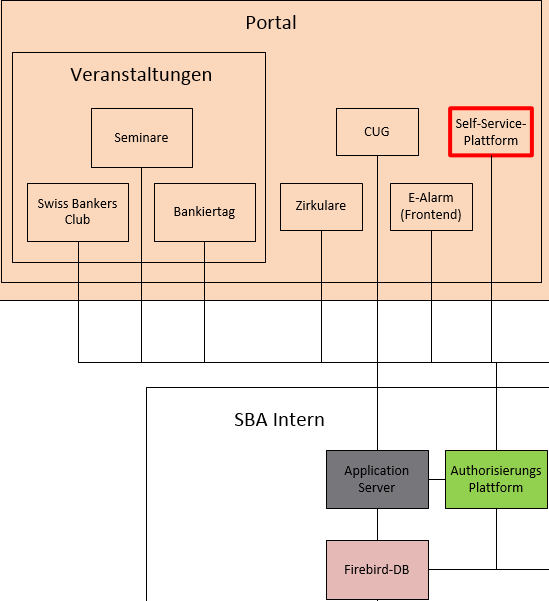


Abbildung: Ist-Situation der Schweizerischen Bankiervereinigung

Durch die Abbildung wird klar, dass alle Module zentral im Portal der Bankiervereinigung eingepflegt wurden. Dies wird sich nun so ändern, dass die „Self-Service-Plattform“ ein eigenständiges Modul ausserhalb des Portals implementiert und an der Autorisierungsplattform angebunden ist.

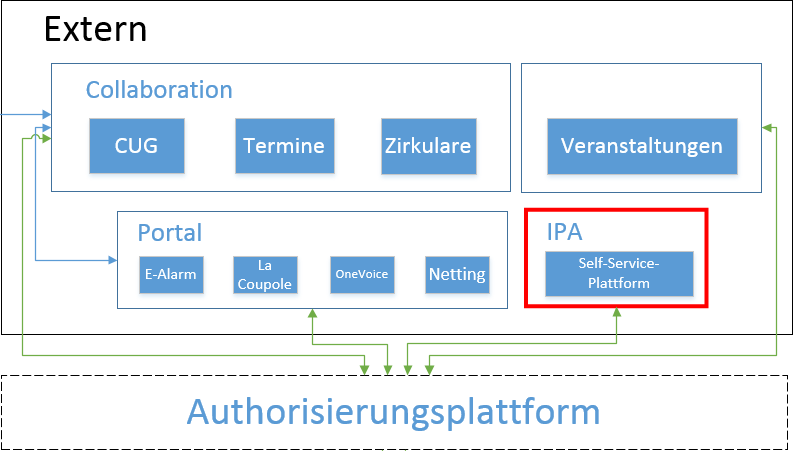


Abbildung: Soll-Situation der Schweizerischen Bankiervereinigung

Die Schweizerische Bankiervereinigung legt grossen Wert auf die Unabhängigkeit der Module, um künftig schneller und günstiger zu reagieren, falls die Geschäftsleitung entscheidet, dass ein Modul ersetzt werden muss.

## 2.3 Entscheiden

## 2.4 Realisieren

## 2.5 Kontrollieren

### 2.5.1 Testfälle

|  |  |
| --- | --- |
| Bezeichnung |  |
| Eingabe |  |
| Erwartetes Resultat |  |
| Tatsächliches Resultat |  |
| Testperson |  |
| Datum und Uhrzeit |  |

## 2.6 Auswerten

## 2.7 Abschlussbericht

## Quellenverzeichnis

* https://erfolgreich-projekte-leiten.de/projektorganisation/
* https://djangobook.com/wp-content/uploads/MTV-Pattern.png
* https://djangobook.com/wp-content/uploads/Django-Models.png

## 2.9 Glossar