SE I - Belegabgabe

StuRa-Mitgliederdatenbank

Theresa Schüttig, Lucie Jill Urbons, Helene Uhlig, Benjamin Hempel, Jonathan Vincent Cremer, Stefan Holland, Mauritius Berger, Lukas Hirsch

31. Januar 2020

Inhaltsverzeichnis

Technische Spezifikation

- Vision
- Use Case Model
- System-wide Requirements
- Glossar
- Domänenmodell

1. Vision StuRa Mitgliederdatenbank

1.1. Einführung

Der Zweck dieses Dokuments ist es, die wesentlichen Bedarfe und Funktionalitäten der Mitgliederdatenbank für den StuRa zu sammeln, zu analysieren und zu definieren. Der Fokus liegt auf den Fähigkeiten, die von Stakeholdern und adressierten Nutzern benötigt werden, und der Begründung dieser Bedarfe. Die Details, wie die Mitgliederdatenbank diese Bedarfe erfüllt, werden in der Use-Case und Supplementary Specification beschrieben.

1.1.1. Zweck

Der Zweck dieses Dokuments ist es, die wesentlichen Anforderungen an das System aus Sicht und mit den Begriffen der künftigen Anwender zu beschreiben.

1.1.2. Gültigkeitsbereich (Scope)

Dieses Visions-Dokument bezieht sich auf die StuRa-Mitgliederdatenbank, das vom in dieser Datei aufgelisteten Team entwickelt wird. Das System wird es Mitgliedern der Referate Verwaltung und Präsidium sowie Mitarbeitern des StuRas erlauben, Daten von Mitgliedern des StuRas über eine Webanwendung zu verwalten, um Informationen zu Mitgliedern, Ämtern und Wahlen zu speichern und einzusehen, einen Arbeitsleitfaden für die interne Verwaltung sowie eine Checkliste zu generieren. Im Idealfall sollte die Anwendung auch das hier veröffentlichte Organigramm erzeugen können.

1.1.3. Definitionen, Akronyme und Abkürzungen

siehe Glossar

1.1.4. Referenzen

Aufgabenstellung

1.2. Positionierung

1.2.1. Fachliche Motivation

Derzeit werden Mitgliederdaten des StuRas über eine Excel-Tabelle verwaltet. Die StuRa-Mitgliederdatenbank soll die Verwaltung von Mitgliedern im StuRa erleichtern und übersichtlicher gestalten, indem Mitglieder sowie Referate und Ämter über eine benutzerfreundliche Website verwaltet werden können. Zudem soll über eine Historie ersichtlich sein, wann in der Datenbank von wem welche Änderungen vorgenommen wurden. Für die interne Verwaltung soll das Programm eine Checkliste mit Aufgaben (z.B. zum Mailverteiler hinzufügen, Zugang zum Schlüsselgang einrichten, etc.) erstellen können. Des Weiteren sollen die Software in der Lage sein, Anwesenheitslisten sowie das Organigramm zu generieren.

1.2.2. Problem Statement

Das Problem	Es fehlt die Übersicht, welches Mitglied welche Funktion belegt.
betrifft	die interne Verwaltung
die Auswirkung davon ist	Unwichtige Funktionen bleiben unbesetzt (nicht auffallen), man kann nicht effizient schauen wer in welcher Funktion tätig ist.
eine erfolgreiche Lösung wäre	Zusätzlich anstatt nach Mitgliedern, nach Funktionen zu sortieren.

Das Problem	Es ist nicht möglich, herauszufinden, von wem bestimmte Änderungen vorgenommen wurden
betrifft	die interne Verwaltung
die Auswirkung davon ist	Verantwortliche für inkorrekte Änderungen können nicht ausfindig gemacht werden und Wiederherstellen korrekter Daten kann sich schwierig gestalten
eine erfolgreiche Lösung wäre	das Erstellen einer Historie

Das Problem	Das Erteilen von Zugängerechten ist unorganisiert
betrifft	die interne Verwaltung, StuRa-Mitglieder
die Auswirkung davon ist	die Verwaltung hat einen schlechten Überblick über noch zu erteilende Zugängge und StuRa-Mitglieder müssen auf Zugänge länger warten als nötig
eine erfolgreiche Lösung wäre	das Erstellen eines Arbeitsleitfadens für die interne Verwaltung

1.2.3. Positionierung des Produkts

Für	Organisationseinheit Verwaltung (StuRa)
welchem	das Verwalten der Mitglieder deutlich erleichtert wird
Die Lösung ist eine	Webanwedung zur Mitgliederverwaltung
Die / Das	das Verwalten übersichtlicher gestaltet und eine Historie speichert
Im Gegensatz zu	Excel-Tabelle
Unser Produkt	zeigt nur abgefragte Daten an und ermöglicht das Hinzufügen und Bearbeiten in kürzerer Zeit

1.3. Stakeholder Beschreibungen

1.3.1. Zusammenfassung der Stakeholder

Name	Beschreibung	Verantwortlichkeiten
Verwaltung des StuRas	selbsterklärend	Eintragen von Mitgliedern, Dokumentieren von Wahlen, Einrichten von Zugängen
Präsidium	Das Präsidium ist zuständig für Sitzungen des StuRas	Dokumentieren von Anwesenheit
Mitglieder des StuRas	Studierende der HTW, die vom StuRa in ein Funktion gewählt wurden	Lesen von Daten aller Mitglieder

1.3.2. Benutzerumgebung

Beschreiben Sie die Arbeitsumgebung des Nutzers. Hier sind einige Anregungen:

- Der StuRa hat derzeit 80 Mitglieder, von denen alle auf die Anwendung zugreifen können sollten. Diese Anzahl kann innerhalb der nächsten Jahre variieren.
- Wie lange dauert die Bearbeitung der Aufgabe? Wie viel Zeit wird für jeden Arbeitsschritt benötigt? Ändert sich das?
- Derzeit eingesetze Anwendung: Excel
- Organigramm

2. Use-Case Model StuRa-Mitgliederdatenbank

2.1. Use-Case: Mitglieder verwalten

2.1.1. Kurzbeschreibung

Mitglieder der internen Verwaltung des StuRas
sollen Mitglieder hinzufügen, bearbeiten und entfernen können.

2.1.2. Kurzbeschreibung der Akteure

Interne Verwaltung

Selbsterklärend.

2.1.3. Vorbedingungen

1. Das StuRa-Mitglied ist eingeloggt und verfügt über die benötigten Berechtigungen.

2.1.4. Standardablauf (Basic Flow)

Mitglied hinzufügen

1. Der Use Case beginnt, wenn der Nutzer dem System mitteilt, dass er ein neues Mitglied anlegen möchte.
2. Alle benötigten Mitgliedsdaten werden angegeben
3. Die Daten werden bestätigt.
4. Der Use Case ist abgeschlossen.

Mitglied bearbeiten

1. Der Use Case beginnt, wenn der Nutzer dem System mitteilt, dass er ein ausgewähltes Mitglied bearbeiten möchte.
2. Der Nutzer kann bestehende Mitgliedsdaten einsehen und verändern
3. Die veränderten Daten werden bestätigt
4. Der Use Case ist abgeschlossen.

Mitglied löschen

Der Use Case beginnt, wenn der Nutzer dem System mitteilt, dass er ein ausgewähltes Mitglied löschen möchte. . Der Löschvorgang wird bestätigt. . Der Use Case ist abgeschlossen.

2.1.5. Nachbedingungen

Hinzugefügtes Mitglied

Das neue Mitglied befindet sich nun in der Mitgliedertabelle der Datenbank. Der Nutzer, vom das Erstellen ausging, sowie das Erstelldatum werden in der Historie verzeichnet.

Bearbeitetes Mitglied

Die Attribute in der Mitgliedertabelle wurden aktualisiert und der Ursprungszustand, der Nutzer, von dem die Bearbeitung ausging, sowie das Bearbeitungsdatum der Historie hinzugefügt.

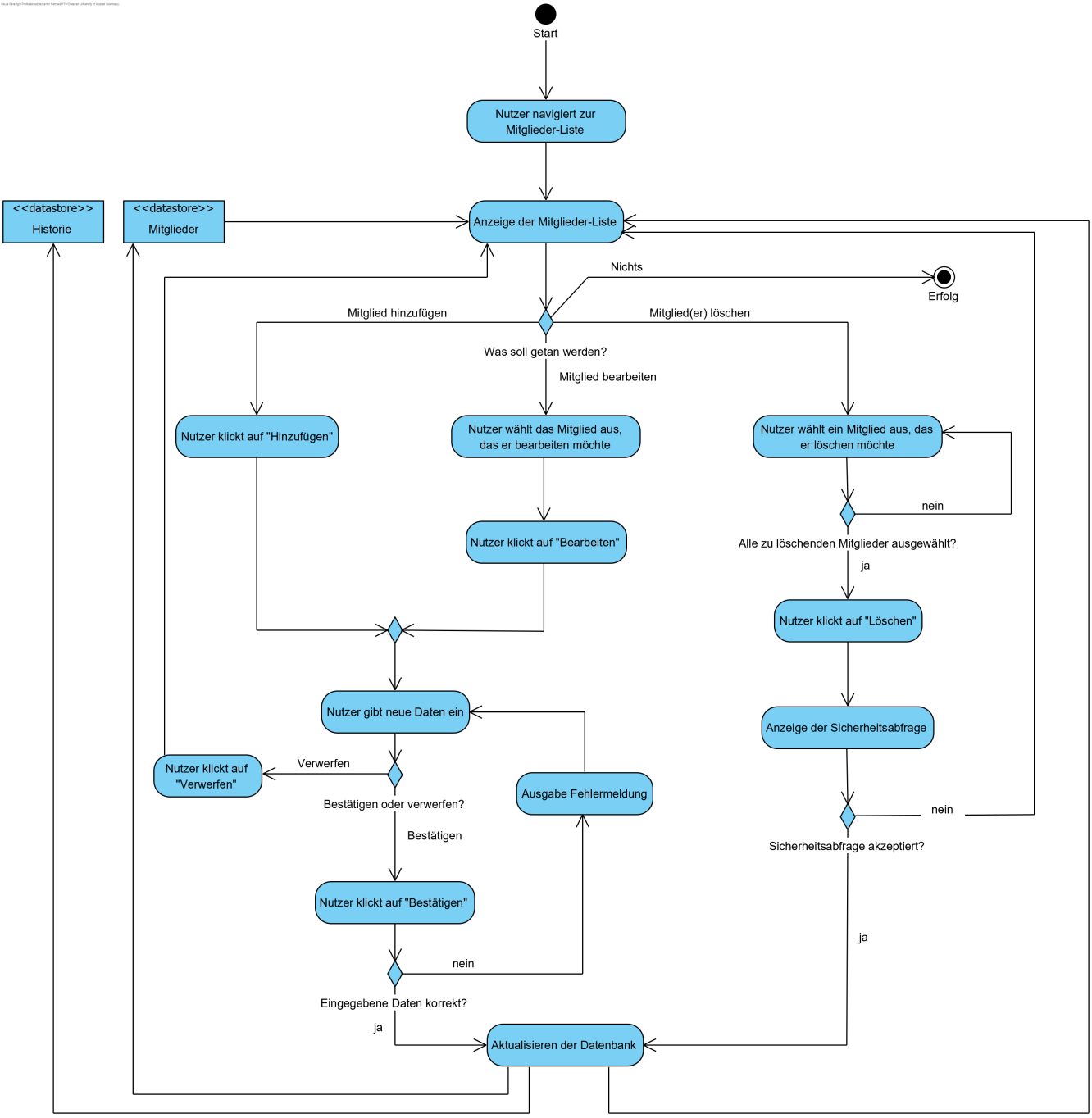
Gelöschtes Mitglied

Das Mitglied wurde aus der Mitgliedertabelle entfernt. Alle personenbezogenen Daten des Mitglieds können nicht mehr wiederhergestellt werden.

2.1.6. Besondere Anforderungen

1. Beim Löschen eines Mitglieds hat eine zusätzliche Sicherheitsabfrage zu erfolgen.

2.1.7. Activity Diagramm



2.2. Use-Case: Ämter verwalten

2.2.1. Kurzbeschreibung

Mitglieder der internen Verwaltung des StuRas können Ämter hinzufügen, bearbeiten und löschen.

2.2.2. Kurzbeschreibung der Akteure

Interne Verwaltung

Selbsterklärend.

2.2.3. Vorbedingungen

Mitglied der internen Verwaltung ist eingeloggt

2.2.4. Standardablauf (Basic Flow)

Funktion hinzufügen

1. Der Use Case beginnt, wenn der Nutzer dem System mitteilt, dass er ein Funktion hinzufügen möchte
2. Eingabe der funktionsspezifischen Daten
3. Bestätigung
4. Der Use Case ist abgeschlossen.

Funktion bearbeiten

1. Der Use Case beginnt, wenn der Nutzer dem System mitteilt, dass er ein Funktion bearbeiten möchte
2. while Nutzer hat Daten noch nicht gespeichert
    - Hinzufügen eines Mitglieds (mit Angabe des Legislaturbeginns) oder Entfernen eines Mitglieds
3. end while
4. Der Use Case ist abgeschlossen.

Funktion löschen

1. Der Use Case beginnt, wenn der Nutzer dem System mitteilt, dass er ein Funktion löschen möchte
2. Bestätigung der Löschung
3. Der Use Case ist abgeschlossen.

2.2.5. Nachbedingungen

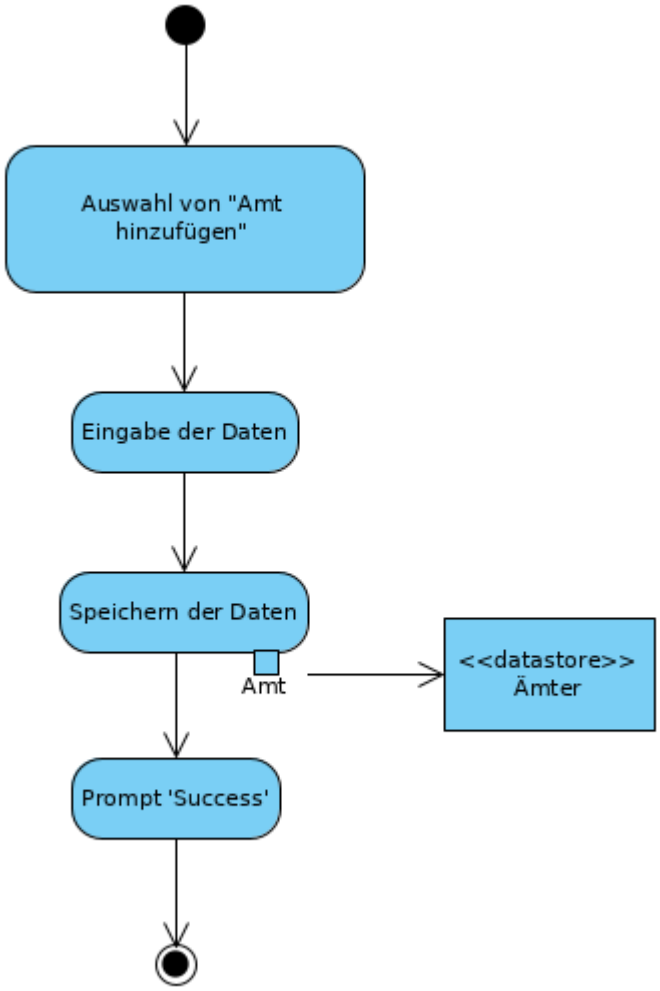
Veränderung der Daten in der Datenbank

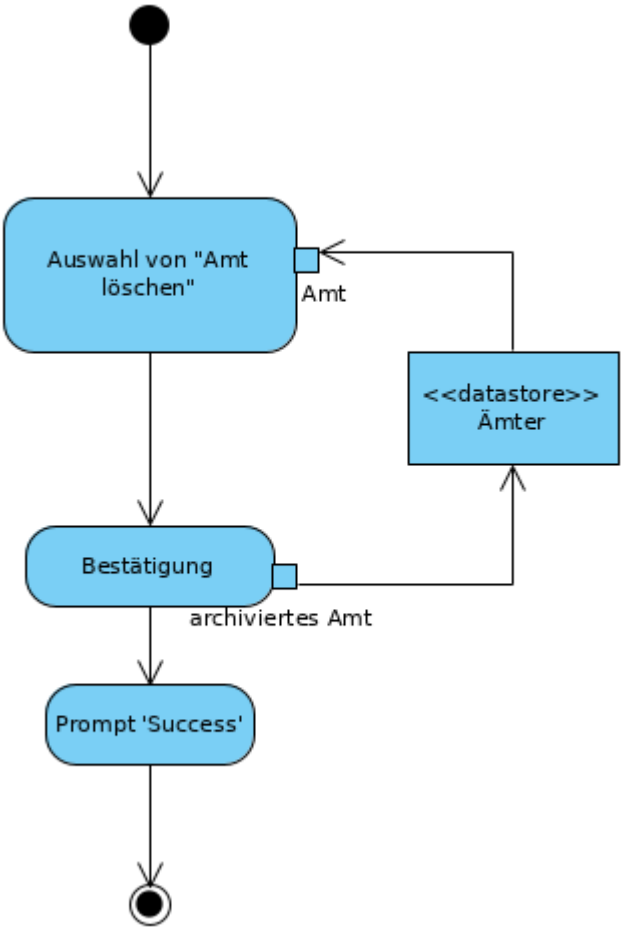
Die Daten werden in der Datenbank aktualisiert. Werden Mitglieder zu Ämtern hinzugefügt oder entfernt, so wird dies auch in der Mitgliedertabelle vermerkt. Gelöschte Ämter befinden sich weiterhin in der Datenbank, werden in der Webanwendung aber nicht mehr angezeigt und können nicht besetzt sein oder werden.

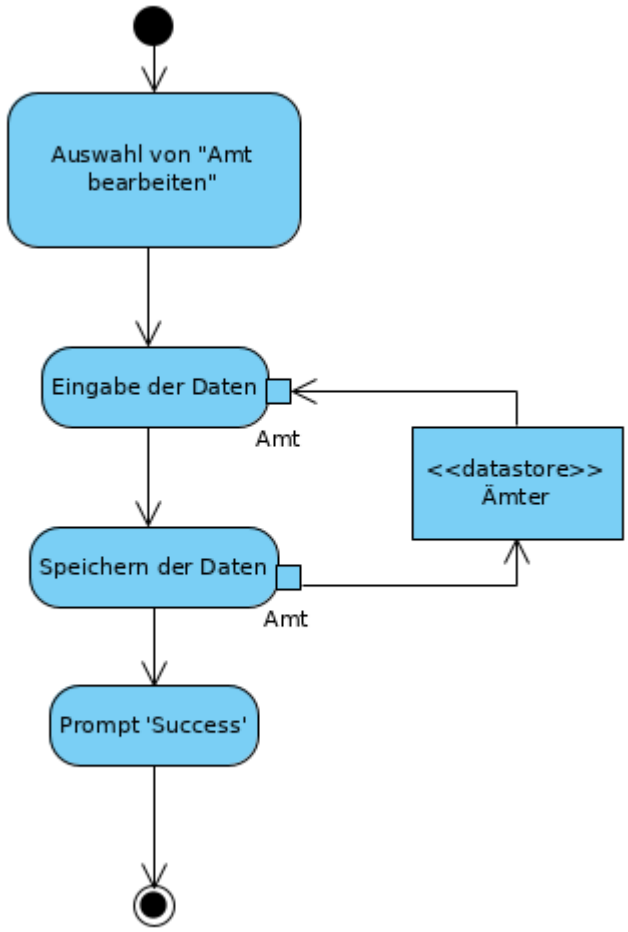
2.2.6. Besondere Anforderungen

1. Beim Löschen eines Amtes hat eine zusätzliche Sicherheitsabfrage zu erfolgen.

2.2.7. Activity Diagramm




Auswahl von "Amt löschen"
Amt
<<datastore>>
Ämter
Bestätigung
archiviertes Amt
Prompt 'Success'



2.3. Use-Case: Systemnutzer verwalten

2.3.1. Kurzbeschreibung

Als Administrator des Systems und Mitglied der internen Verwaltung des StuRas
möchte ich Systemnutzer hinzufügen, bearbeiten und löschen können.

2.3.2. Kurzbeschreibung der Akteure

Administrator des Systems

Der/die Administrator/in/en haben Vollzugriff auf das System und sind für die Verwaltung der Nutzerzugänge verantwortlich.

Interne Verwaltung des StuRas

Eines oder mehrere Mitglieder der internen Verwaltung sind für die Administration des Systems verantwortlich.

2.3.3. Vorbedingungen

- Der/die Administrator/in ist im System eingeloggt.

2.3.4. Standardablauf (Basic Flow)

1. Der Use Case beginnt, wenn der/die Administrator/in die Oberfläche zur Verwaltung der Systemnutzer öffnet.
2. while Administrator/in hat noch nicht alle Änderungen vorgenommen
    a. Dem Akteur wird eine aktuelle Liste aller Systemnutzer angezeigt.
    b. Es wird ausgewählt, ob ein Nutzer hinzugefügt, bearbeitet oder gelöscht werden soll.
    c. Fortsetzung in den jeweiligen Unterabläufen (subflows).
3. end while
4. Der Use Case ist abgeschlossen.

2.3.5. Unterabläufe (subflows)

Hinzufügen eines neuen Nutzers

1. Der Admin wählt aus, dass er einen neuen Nutzer hinzufügen möchte.
2. Der Admin vergibt einen Benutzernamen, ein Passwort (welches beim erstmaligen Login geändert werden muss) und ordnet den Systemnutzer dem entsprechenden StuRa-Mitglied zu.
3. Der Admin bestätigt die Eingabe und es wird ihm/ihr die aktualisierte Systemnutzer-Liste angezeigt.

Ändern eines bestehenden Nutzers

1. Der Admin wählt einen bestehenden Eintrag in der Systemnutzer-Liste aus.
2. Der Admin wählt aus, dass er diesen Systemnutzer ändern möchte.
3. Der Admin ändert den Nutzernamen, setzt das Passwort zurück und/oder ändert die Zuordnung zum entsprechenden StuRa-Mitglied.
4. Der Admin bestätigt die Eingabe und es wird ihm/ihr die aktualisierte Systemnutzer-Liste angezeigt.

Löschen eines bestehenden Nutzers

1. Der Admin wählt einen bestehenden Eintrag in der Systemnutzer-Liste aus.
2. Der Admin wählt aus, dass er diesen Systemnutzer löschen möchte.
3. Der Admin bestätigt die Sicherheitsabfrage und es wird ihm/ihr die aktualisierte Systemnutzer-Liste angezeigt.

2.3.6. Nachbedingungen

Hinzufügen eines neuen Nutzers

1. Falls ein neuer Nutzer hinzugefügt wurde, so wird dieses als neuer Datensatz in der Nutzerdatenbank gespeichert.
2. Der Nutzer hat mit den angegebenen Anmeldedaten Zugriff auf das System.

Änderung eines bestehenden Nutzers

1. Falls ein Nutzer geändert wurde, sind die geänderten Daten in der Nutzerdatenbank gespeichert.
2. Der Nutzer hat mit den geänderten Anmeldedaten Zugriff auf das System, aber nicht mehr mit den alten Daten.

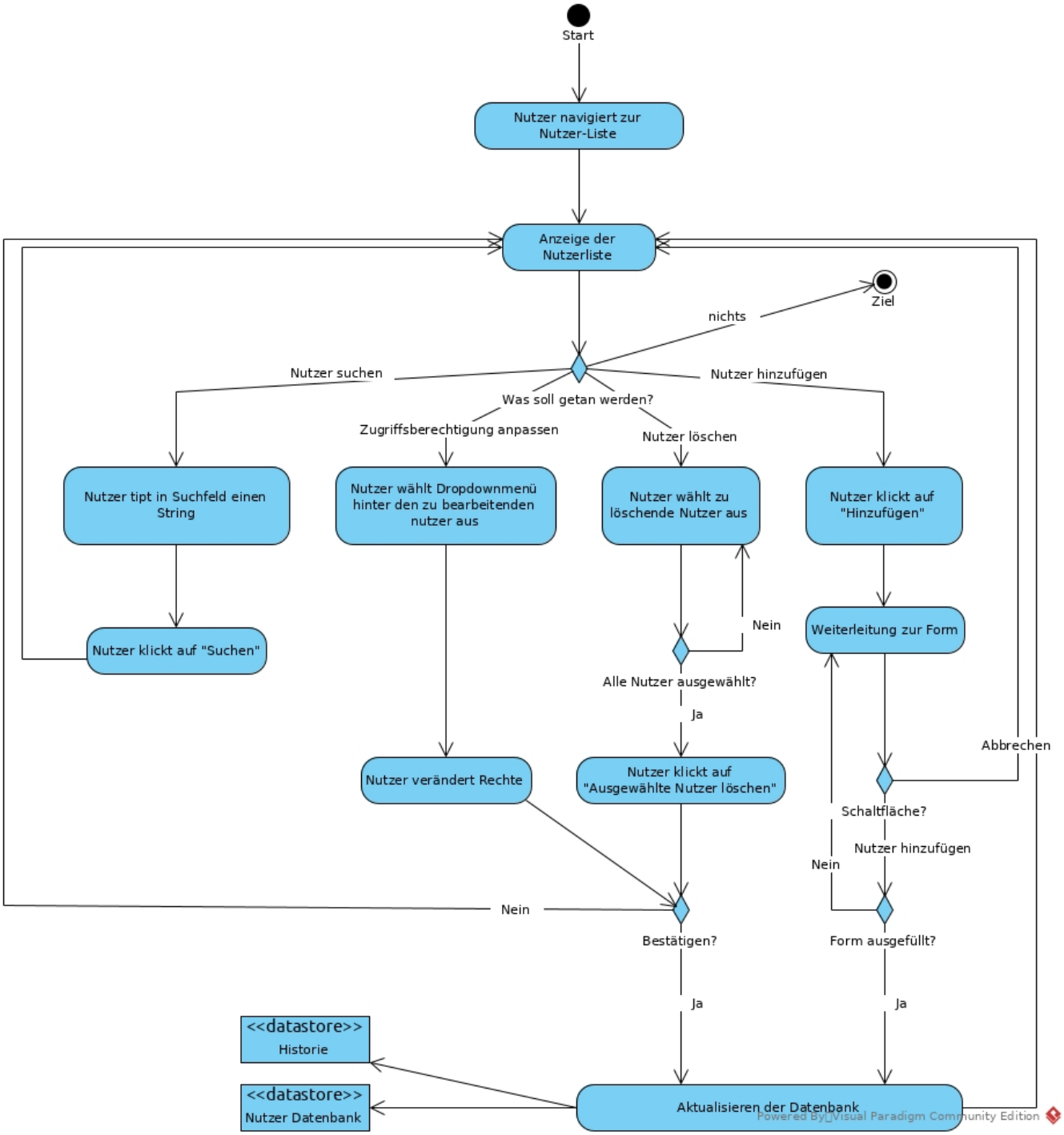
Löschen eines bestehenden Nutzers

1. Falls ein Nutzer gelöscht wurde, so wurde sein Datensatz aus der Nutzerdatenbank entfernt.
2. Der gelöschte Nutzer hat keinen Zugriff mehr auf das System.

2.3.7. Besondere Anforderungen

1. Gespeicherte Anmeldedaten, insbesondere Passwörter, sollten niemals im Klartext in der Nutzerdatenbank vorliegen und es sollte niemals mit Klartextdaten gearbeitet werden.
2. Vor dem Löschen eines Systemnutzers hat eine Sicherheitsabfrage zu erfolgen.

2.3.8. Activity Diagramm



2.4. Use-Case: Arbeitsleitfaden abarbeiten

2.4.1. Kurzbeschreibung

Nach Aufnahme eines neuen Mitglieds im StuRa müssen bestimmte Aufgaben (Emailverteiler, Schlüsselkasten, Zugangsschließsystem, Berechtigung Website/Dateiverwaltung/Aufgabenverwaltung, notwendige Unterschriften) abgearbeitet werden.

2.4.2. Kurzbeschreibung der Akteure

Interne Verwaltung

Mitglieder der internen Verwaltung sind für die Abarbeitung der Checkliste verantwortlich

2.4.3. Vorbedingungen

Das Mitglied der internen Verwaltung ist eingeloggt.

2.4.4. Standardablauf (Basic Flow)

1. Der Use Case beginnt, wenn der Nutzer auswählt, dass er die Checkliste abarbeiten möchte.
2. Der Nutzer kann Aufgaben kennzeichnen, die er abgearbeitet hat.
3. Der Nutzer bestätigt die aktualisierte Checkliste.
4. Der Use Case ist abgeschlossen.

2.4.5. Wesentliche Szenarios

StuRa-Mitglied M1 wurde dem E-Mail-Verteiler hinzugefügt

1. Das Mitglied der internen Verwaltung wählt aus, dass es die Checkliste von M1 bearbeiten will
2. Es bestätigt Verteilung zum E-Mail-Verteiler
3. Die Veränderung wird bestätigt

Falsche Angaben bei Schließsystem von StuRa-Mitglied M2 mit abgearbeiteter Checkliste

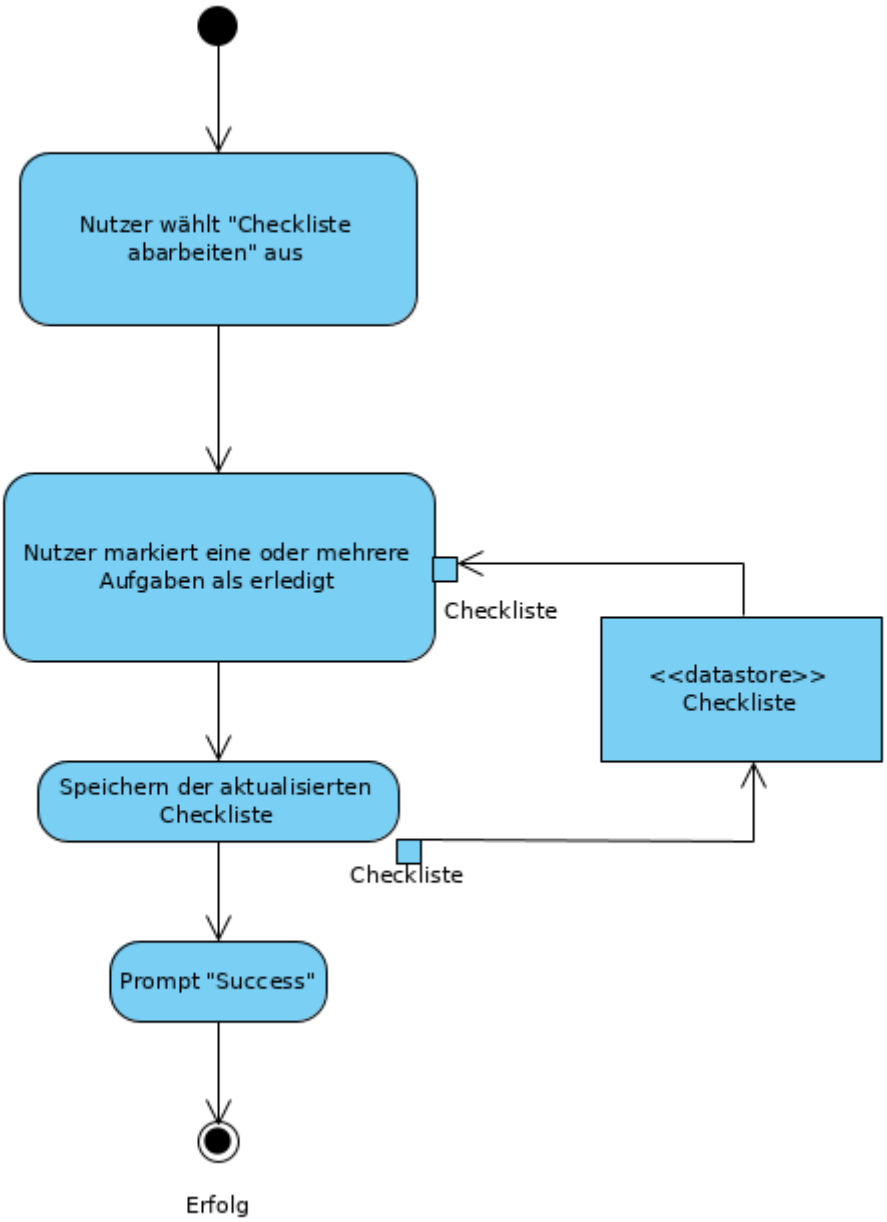
1. Das Mitglied der internen Verwaltung wählt aus, dass es die Checkliste von M2 bearbeiten will
2. Es kennzeichnet die Berechtigung für das Zugangsschließsystem als nicht abgearbeitet
3. Die Veränderung wird bestätigt

2.4.6. Nachbedingungen

Gespeicherte Checkliste

1. die gespeicherte Checkliste wurde in der Datenbank aufgenommen

2.4.7. Activity Diagramm



2.5. Use-Case: Historie einsehen

2.5.1. Kurzbeschreibung

Um Änderungen an der Mitgliederdatenbank nachvollziehen und ggf. rückgängig machen zu können,
möchte ich die Historie aller Änderungen einsehen können.

2.5.2. Kurzbeschreibung der Akteure

Interne Verwaltung

Die Mitglieder der internen Verwaltung sind für die Richtigkeit und Vollständigkeit der Daten in der Mitgliederdatenbank verantwortlich.

2.5.3. Vorbedingungen

1. Der Akteur ist eingeloggt und verfügt über die Berechtigung, die Historie einsehen zu können.

2.5.4. Standardablauf (Basic Flow)

1. Der Use Case beginnt, wenn der Akteur dem System mitteilt, dass er die Historie einsehen möchte.
2. Das System listet die letzten n Einträge der Historie auf.
3. while Akteur hat noch nicht alle Informationen, die er braucht
    - Der Nutzer ändert die Filterkriterien für die Einträge (siehe Unterablauf).
    - ODER
    - Der Nutzer lässt sich ältere Einträge anzeigen, indem er die Seite wechselt.
4. end while
5. Der Use Case ist abgeschlossen.

2.5.5. Unterabläufe (subflows)

Filterkriterien ändern

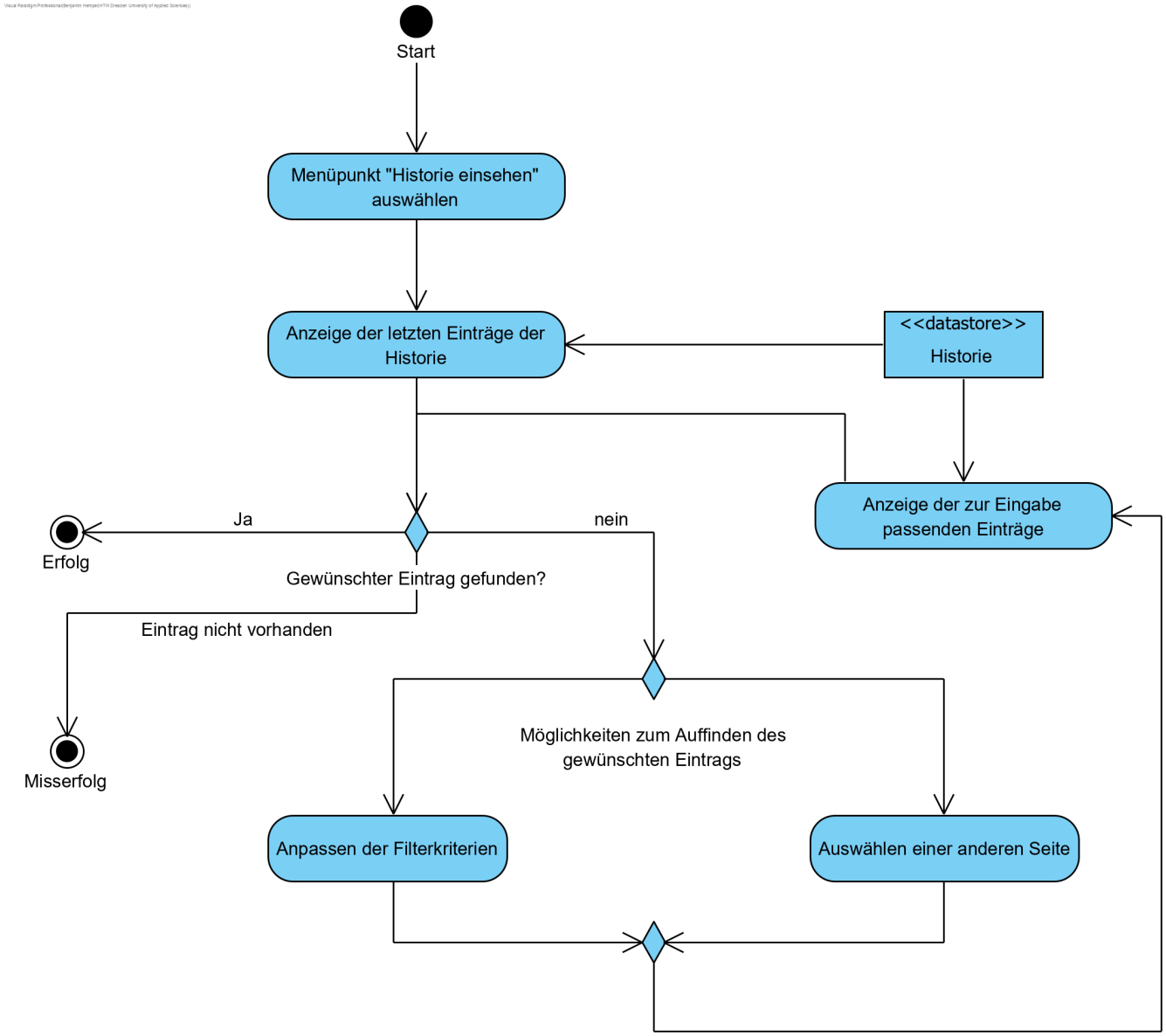
1. Der Akteur entscheidet sich dazu, die Historie nach bestimmten Kriterien zu filtern.
2. Der Akteur gibt die entsprechenden Filterkriterien ein und teilt dem System mit, dass es alle Einträge basierend auf diesen Kriterien auflisten soll.
3. Das System listet die letzten n Einträge der Historie auf, die den Filterkriterien entsprechen.

2.5.6. Besondere Anforderungen

Unterteilung in Seiten (Pagination)

- Damit nicht immer die gesamte Historie geladen werden muss, sollten zunächst nur die letzten n Einträge angezeigt werden.
- Mittels einer Pagination (Menü zum Wechseln von Seiten) kann der Nutzer auch ältere Einträge sehen, die nur bei Bedarf geladen werden.

2.5.7. Activity Diagramm



2.6. Nach Wichtigkeit sortierte Usecase-Liste

1. Mitglied hinzufügen/löschen/ändern
2. Funktion hinzufügen/löschen/ändern
3. Systemnutzer hinzufügen/löschen/ändern
4. Historie
5. Arbeitsleitfaden

3. Stura-Mitgliederdatenbank System-Wide Requirements Specification

In diesem Dokument werden die systemweiten Anforderungen für das Projekt Stura-Mitgliederdatenbank spezifiziert. Die Gliederung erfolgt nach der FURPS+ Anforderungsklassifikation:

- Systemweite funktionale Anforderungen (F),
- Qualitätsanforderungen für Benutzbarkeit, Zuverlässigkeit, Effizienz und Wartbarkeit (URPS) sowie
- zusätzliche Anforderungen (+) für technische, rechtliche, organisatorische Randbedingungen

Die funktionalen Anforderungen, die sich aus der Interaktion von Nutzern mit dem System ergeben, sind als Use Cases in einem separaten Dokument festgehalten.
[Use-Cases]

3.1. Systemweite funktionale Anforderungen

<F001> Authentifizierung:

- Daten müssen vor Unbefugten geschützt werden

<F002> Zeitsteuerung:

- Die Datenbank muss in regelmäßigen Abständen (Jeden Monat) gesichert werden
- Alte Sicherungen Löschen (Die älter sind als 12 Monate)

<F003> Auditierung:

- Es soll ein Änderungslog geführt werden (Datum-Nutzer-Änderung)

3.2. Qualitätsanforderungen für das Gesamtsystem

3.2.1. Benutzbarkeit (Usability)

<F004> Sprache: Deutsch
<F005> Datumsformat: dd.mm.yyyy
<F006> Währungen: Euro

<F007> Benutzerfreundlichkeit

- Sehr einfaches Erlernen und schnelle Bedienung.
- Daten können nach mehreren Kriterien gefiltert werden
- Das System sollte den Nutzer bei ungültigen Eingaben anleiten, wie diese auszusehen haben

- Einfache administrative Aufgaben soll ein priorisierter nutzer erledigen können

3.2.2. Zuverlässigkeit (Reliability)

<F008> Verfügbarkeit: Die Software ist grundsätzlich verfügbar, solange auch der Server des StuRas diese Bedingung erfüllt. Das Programm soll Wartungsfrei laufen, bzw vom Endnutzer gewartet werden können im laufendem Betrieb.
<F009> Wiederherstellbarkeit: Das Programm soll bei Totalausfall zum letzten Backup hergestellt werden können (in ca 24h?)

3.2.3. Effizienz (Performance)

<F010> Antwortzeiten: Es ist keine spezielle Antwortzeit vorgeschrieben, die Websites sollten aber sich in endlicher zeit aufbauen (Pagination)
<F011> Durchsatz: Es ist kein spezieller Durchsatz gefordert
<F012> Kapazität:

- Mehrbenutzerbetrieb muss gewährleistet sein. → Serialisierung für die Datenbankarbeit

3.2.4. Wartbarkeit (Supportability)

<F013> (Niedrigste Priorität) Anpassbarkeit: Es soll bei freier Kapazität eine API für den Export eines Organigrammes erstellt werden.
<F014> Kompatibilität: Mit allen Betriebssystemen → Browseranwendung. Es muss einbindbar in den Plone sein.
<F015> Konfigurierbarkeit: Die zugrundeliegende Datenbank soll frei gewählt werden (MySQL, MSSQL, SQLite, …)
<F016> Wartbarkeit: Code ausreichend dokumentieren(Sphinx)/struckturieren(PEP8) für eine Weiterentwicklung.

3.3. Zusätzliche Anforderungen

3.3.1. Einschränkungen (Constraints)

<F017> Implementierung: Python (Django), SQLite
<F018> Plattform: Plattformunabhängig → Browserapplikation
<F019> Ressourcenbegrenzungen: Möglichst geringer Energieverbrauch, Speicherplatz (so viel wie im Container vorhanden), Möglichst wenig Traffic
<F020> Legal: strikte Einhaltung des Datenschutzes

3.3.2. Interface Requirements

<F021> Look & Feel: angelehnt an das bekannte Stura/Htw-Design

3.3.3. Organisatorische Randbedingungen

- Anforderungen an Betrieb, Management und Wartung der Anwendung
- zu beachtende Standards, Normen und Regeln

3.3.4. Rechtliche Anforderungen

- Lizensierung der Anwendung (Open Source)
- Datenschutz (mit dem Austritt eines Mitglieds müssen alle Daten dieser Person entfernt werden)

4. Glossar

4.1. Einführung

In diesem Dokument werden die wesentlichen Begriffe aus dem Anwendungsgebiet (Fachdomäne) der StuRa-Mitgliederdatenbank definiert. Zur besseren Übersichtlichkeit sind Begriffe, Abkürzungen und Datendefinitionen gesondert aufgeführt.

4.2. Begriffe

Begriff	**Definition und Erläuterung**	**Synonyme**
Administrator	Eine Person mit Vollzugriff auf das System, welche sich um die Verwaltung desselbigen kümmert.	Admin
Ausschuss	Erfüllung von verschiedenen Aufgaben der studentischen Selbstverwaltung	
Beauftragte	Instrumente für die Beobachtung und zur Intervention zu speziellen wichtigen Belangen, die vom StuRa selbst festgelegt werden	
Beratendes Mitglied	Mitglied, das von StuRa-Mitgliedern in den StuRa gewählt wurde	
Checkliste	Liste mit allen Aufgaben für die interne Verwaltung des StuRas bezüglich der Mitgliederverwaltung	
Exekutives Funktion	Ämter der Referate, Sprecher, Präsidium, Ausschuss, Beauftragte	
Exekutives Mitglied	Mitglied, das in ein exekutives Funktion berufen wurde, höher gestellt als exekutive Mitglieder	
Filterkriterium	Ein Begriff, nach welchem die vorhandenen Datensätze "ausgesiebt" werden	
Gremium	zur Erfüllung einer bestimmten Aufgabe berufene Kommission	

Begriff	Definition und Erläuterung	Synonyme
Legislatives Mitglied	Mitglied, das von der Studentenschaft gewählt wurde	
Legislatur	Zeitraum, in dem ein Mitglied ein bestimmtes Funktion besetzt hat	
Organigramm	Übersicht aller Ämter und Besetzungen	
Plenum	Gesamtheit der StuRa-Mitglieder	
Präsidium	Verantwortliche für StuRa-Sitzungen (Vorbereitung, Vorsitz, Nachbereitung)	
Organisationseinheit	Exekutive des StuRas, verantwortlich für die Umsetzung der Beschlüsse des StuRas	
Sicherheitsabfrage	Vom Nutzer wird für bestimmte Operationen eine zusätzliche Bestätigung dieser angefordert.	
Sprecher	Vertreter der Stundentenschaft bei verschiedenen Veranstaltungen	
Systemnutzer	Eine Person, die auf das Mitgliederdatenbank-System Zugriff hat.	Nutzer, Benutzer
Wahl	Vorgang, bei dem StuRa-Mitglieder ein Mitglied für ein Funktion wählen	
Zugangsschließsystem	System für das Aufschließen (Entriegeln) für den Zugang zu den Räumen des StuRa bzw. der Studentinnen- und Studentenschaft	

4.3. Abkürzungen und Akronyme

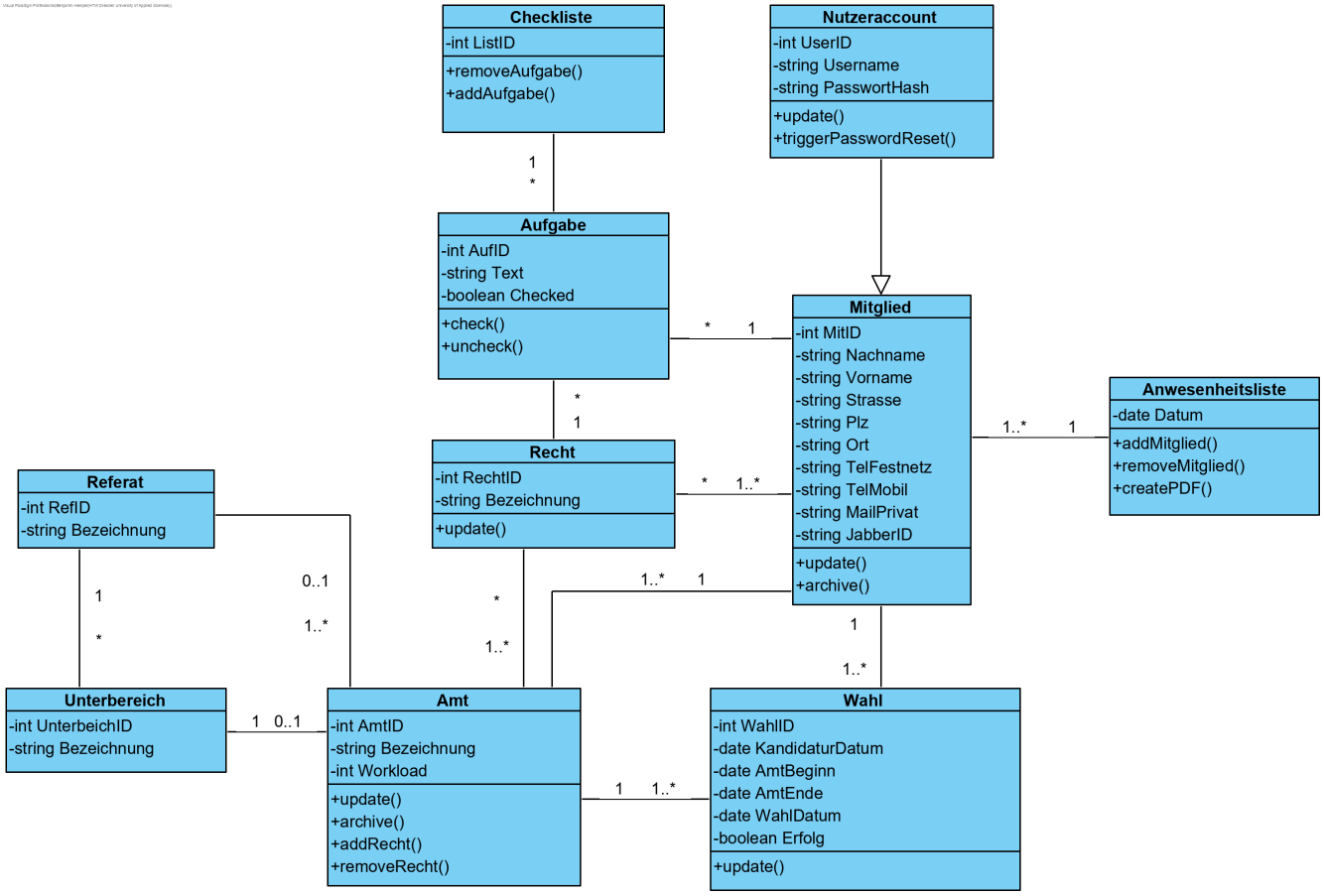
Abkürzung	Bedeutung	Erläuterung
AE	Aufwandsentschädigung	Bei besonders hohem Zeitaufwand als Vergütung
UP	Unified Process	Vorgehensmodell für die Softwareentwicklung

Abkürzung	Bedeutung	Erläuterung
Plone	—	Website des STURA
LXC	Linux Containers	Virtualisierung auf Betriebssystemebene, indem mehrere Systeme isoliert gleichzeitig laufen sich aber denselben Kernel teilen.

4.4. Verzeichnis der Datenstrukturen

Bezeichnung	Definition	Format	Gültigkeitsregeln	Aliase
Anmeldedaten	Zusammensetzung von Benutzername und Passwort.	String	Emailadresse muss @-Zeichen und Punkt enthalten.	Login
Mitgliedsdaten	Zusammensetung von Name, Vorname, Legislatur(Funktion), Adresse	Klasse	belegte Funktion muss vorhanden sein	xx
Daten (funktionsspezifisch)	Referenzierung der Organisationseinheit und des Unterbereiches, der die Funktion untergeordnet ist. Zusammensetung von Bezeichnung, Workload, Maximalmitgliedern	Klasse	xx	xx

5. Domänenmodell



Projektdokumentation

- Projektplan
- Risikoliste
- Iteration Plan (für zwei ausgewählte Iterationen)

6. Projektplan StuRa Mitgliederdatenbank

6.1. Einführung

Der Projektplan enthält Informationen über den allgemeinen Ablauf des Projektes.

6.2. Projektorganisation

6.2.1. Team Mitglieder

Name	Vorname	Primärrolle	Sekundärrolle
Berger	Mauritius	Project Manager	Developer Backup
Urbons	Lucie Jill	Analyst	---
Uhlig	Helene	Analyst	---
Cremer	Jonathan Vincent	Architect	Developer Backup
Schüttig	Theresa	Developer	Tester Backup
Hempel	Benjamin	Developer	---
Hirsch	Lukas	Tester	Technical Writer
Holland	Stefan	Deployment Eng.	Analyst Backup
Michel	Marc	Architect	Developer Backup

6.3. Praktiken und Bewertung

- Wir arbeiten in Iterationen, welche 2 Wochen andauern.
- Aus dem ersten Treffen hat sich herausgestellt, dass wir folgende Tools nutzen wollen:
  - Protokollierung:
    - Pentapad
  - Versionsverwaltung:
    - Github
  - Kommunikation:
    - Whatsapp
    - Discord
  - Tickets aka. WorkItemList:
    - Github Issues
    - Trello

6.4. Meilesteine und Ziele

Iteration	Primary objectives (risks and use case scenarios)	Scheduled start or milestone	Target velocity
Iteration 1	• Aufgaben und Rollen verstehen • Einschätzen ob es sinvoll und machbar ist • Stakeholder kennenlernen • LCO • Erste Modellierung (Use Cases)	01.12.2019 - 20.12.2020	-
Iteration 2	• Veranschaulichung (Wireframes)	06.01.2020 - 17.01.2020	-
Iteration 3	• LCA	20.01.2020 - 31.01.2020	-
Wintersemester Ende	-	-	-
Sommersemester Beginn	-	-	-
Iteration 4	-	16.03.2020 - 29.03.2020	-
Iteration 5	-	30.03.2020 - 12.04.2020	-
Iteration 6	-	13.04.2020 - 26.04.2020	-
Iteration 7	-	27.04.2020 - 10.05.2020	-
Iteration 8	-	11.05.2020 - 24.05.2020	-
Iteration 9	-	25.05.2020 - 07.06.2020	-
Iteration 10	-	08.06.2020 - 21.06.2020	-
Iteration 11	-	22.06.2020 - 05.07.2020	-
Iteration 12	-	06.07.2020 - 19.07.2020	-
Iteration 13	-	20.07.2020 - 02.08.2020	-
Iteration 14	-	03.08.2020 - 14.08.2020	-

6.5. Deployment

- Administratoren festlegen und aktiv in das Projekt mit einbeziehen → Support Faehigkeit aufbauen
- Treffen mit allen noetigen Personen zur klaerung saemtlicher technischer Details/Besonderheiten,geplante Entwickungsstadien
- Termine der wichtigen Phasen vereinbaren wie Testphase,Load testing,Release(Releaseplanung)…
- reibungsloser Rollout

6.6. Erkenntnisse (Lessons learned)

- Das Anwendung des Vier-Augen-Prinzip (d.h. jedes Dokument wird von einem anderen Team-Mitglied überprüft) ist sinnvoll, um die Korrektheit und Vollständigkeit der Dokumente zu sichern.
- Stagnationen des Fortschritts sind vor allem darauf zurückzuführen, dass Stakeholder nicht sofort zum Gespräch zu Verfügung stehen können oder darauf, dass Aufgaben im Team nicht rechtzeitig erledigt werden.
- Als hilfreich hat sich herausgestellt, Aufgaben in Teilaufgaben aufzuteilen und diese jeweils einer Person zuzuordnen.

7. Risikoliste -Projektthema-

In diesem Dokument sind die wesentlichen Risiken des Projekts aufgeführt. Dabei werden folgende Attribute verwendet:

1. **Typ:** Ressourcen, Geschäftlich, Technisch, Zeitlich
2. **Auswirkung (IMP):** Wert zwischen 1 (niedrig) und 5 (hoch), der die Auswirkungen auf das Projekt angibt, wenn das Risiko eintritt
3. **Wahrscheinlichkeit (PRB):** Prozentangabe für die Eintrittswahrscheinlichkeit des Risikos
4. **Stärke (MAG):** Produkt aus Auswirkung und Wahrscheinlichkeit (damit kann die Liste sortiert werden)

Die Risiken sind in folgender Tabelle dargestellt. Das Datum des Dokuments oben gibt an, wann die Risikoliste zuletzt aktualisiert wurde.

type	**priority (0-10)**	**chance (0-10)**	**priority*chance (0-100)**
Ausfall durch Krankheit oder andere wichtige Gruende	7	6	42
Dauerhafter Ausfall	10	1	10
Zeitmangel durch Pruefungsvorbereitung	5	9	45
Stakeholder sind unzufrieden	8	5	40
Github ist offline (kurzfristig)	8	2	16
Github wird komplett offline genommen (langfristig)	10	1	10
Kein Treffen mehr möglich	10	9	90
Deployment funktioniert nicht wie erwartet	7	6	42
Internetprobleme	8	5	40

8. Iterationsplan Iteration 2

8.1. Meilensteine

Meilenstein	Datum
Beginn der Iteration	06.01.2020
Fertigstellen der Aufgaben	-
Ende der Iteration	17.01.2020

8.2. Wesentliche Ziele

- Wireframes ausarbeiten
- Ausgearbeitete Wireframes Stakeholdern vorführen

8.3. Aufgabenzuordnung

Die in dieser Iteration geplanten Aufgaben sind in der Work Items List dargestellt (Link zur Work Items List).

alternativ: Die folgenden Aufgaben werden in dieser Iteration bearbeitet:

Aufgabe bzw. Beschreibung	Priorität	Schätzung der Größe (Punkte)	Status	Referenzen	Zugewiesen (Name)	Gearbeite Stunden	Schätzung der verbleibenden Stunden
Essence Navigator	niedrig	2	fertig	Essence Navigator Bilder	Jill3, Helene 3e	1	1
Wireframes definieren	hoch	3	fertig	Wireframes	alle	3	3
Wireframes ausarbeiten	mittel	8	in Arbeit	Wireframes, Issue #11	alle	3	8
Architecture notebook	mittel	5	in Arbeit	Architecture Notebook	creerm	?	5
Iteration 3 Plan ausarbeiten	hoch	3	in Arbeit	Iteration 3 Plan	mribrgr	0	3

Aufgabe bzw. Beschreibung	Priorität	Schätzung der Größe (Punkte)	Status	Referenzen	Zugewiesen (Name)	Gearbeite Stunden	Schätzung der verbleibenden Stunden
Glossar fuellen	mittel	2	aussteh end	Glossar	Jill3, Helene 3e, Unterst ützung von thrs791 36	0.5	2
Aktivitaetsdiagramme erstellen	mittel	3	fertig	Aktivitätsdiagra mme	thrs791 36	3	3
Nichtfunktionale Anforderungen	mittel	3	in Arbeit	x	Stefano -96	1.5	3
Risiko Liste	niedrig	2	fertig	Risiko Liste	mribrgr	2	2

8.4. Bewertungskriterien

- Essence Navigator Bilder wurden erstellt
- Wireframes erhalten positive Rückmeldung vom Stackholder
- Aufgaben abgeschlossen ja/nein

8.5. Assessment

Assessment Ziel	Alle Aufgaben der Iteration 2 fertigstellen
Assessment Datum	17.01.2020
Teilnehmer	alle
Projektstatus	grün

- Beurteilung im Vergleich zu den Zielen
  - Ziel wurde größtenteils erreicht, wenige Aufgaben werden in Iteration 3 übernommen
- Geplante vs. erledigte Aufgaben
  - Es wurden fast alle Aufgaben gelöst bis auf: (Diese sind jedoch in Arbeit)
    - Architecture notebook
    - Glossar füllen
    - Nichtfunktionale Anforderungen

- Beurteilung im Vergleich zu den Bewertungskriterien
  - Bilder wurden 100%ig erstellt und hochgeladen
  - Wireframes wurden ausgearbeitet und haben auch Zustimmung von Stakeholdern bekommen
  - Alle Aufgaben bis auf die o.g. wurden erfüllt
- Andere Belange und Abweichungen
  - Treffen mit Stakeholdern
  - Couch gab als Tipp, dass die Qualität überprüft werden muss, Project Manager und Deployment setzen sich mal zusammen

9. Iterationsplan Iteration 3

9.1. Meilensteine

Meilenstein	Datum
Beginn der Iteration	20.01.2020
Fertigstellen der Aufgaben	-
Ende der Iteration	31.01.2020

9.2. Wesentliche Ziele

- Essence Navigator weiterhin auf dem Laufenden halten
- Vorbereitung für Implementation fertigstellen

9.3. Aufgabenzuordnung

Die in dieser Iteration geplanten Aufgaben sind in der Work Items List dargestellt (https://github.com/mribrgr/StuRa-Mitgliederdatenbank/issues).

alternativ: Die folgenden Aufgaben werden in dieser Iteration bearbeitet:

Aufgabe bzw. Beschreibung	Priorität	Schätzung der Größe (Punkte)	Status	Referenzen	Zugewiesen (Name)	Gearbeite Stunden	Schätzung der verbleibenden Stunden
Essence Navigator	niedrig	2	ausstehend	Essence Navigator Bilder, Issue #25, Issue #26	Jill3, Helene 3e	0	2
Iterationsplan 4 erstellen	hoch	3	ausstehend	x	mribrgr	0	3
Glossar fuellen	mittel	2	ausstehend	Glossar, Issue #24	Jill3, Helene 3e, Unterstützung von thrs79136	0.5	2

Aufgabe bzw. Beschreibung	Priorität	Schätzung der Größe (Punkte)	Status	Referenzen	Zugewiesen (Name)	Gearbeite Stunden	Schätzung der verbleibenden Stunden
Architecture notebook	mittel	5	in Arbeit	Architecture Notebook	creerm	?	5
Nichtfunktionale Anforderungen	mittel	3	in Arbeit	x	Stefano-96	1.5	3
Projektplan abgabefertig machen	hoch	2	in Arbeit	tba	mribrgr	1	2
Test Cases ausarbeiten	hoch	3	in Arbeit	tba	Yoshinonxkun	0	5
Abgabe vorbereiten	hoch	2	in Arbeit	tba	Yoshinonxkun	0	3

9.4. Probleme (optional)

Problem	Status	Notizen
Zeitdruck durch andere Module	trifft zu	versuchen das Beste draus zu machen

9.5. Bewertungskriterien

- Aufgaben wurden erfüllt oder nicht
- Bewertung durch Prof (in dem Sinne, dass es eine Bewertung der Iteration gibt, dadurch, dass die Vorbereitung der Abgabe in dieser Iteration enthalten ist)

9.6. Assessment

Assessment Ziel	Iteration 3 fertigstellen
Assessment Datum	31.01.2020
Teilnehmer	alle
Projektstatus	Zum Beispiel ausgedrückt als rot, gelb oder grün.

- Beurteilung im Vergleich zu den Zielen Aufgaben sind in Arbeit.
- Geplante vs. erledigte Aufgaben Derzeit liegen wir gut in der Zeit.
- Beurteilung im Vergleich zu den Bewertungskriterien Bisher 70% erfüllt.

Entwurfsdokumentation

- Architektur-Notizbuch
- Test Cases
- Design

10. Architecture Notebook StuRa-Mitgliederdatenbank

10.1. Zweck

Dieses Dokument beschreibt die Philosophie, Entscheidungen, Nebenbedingungen, Begründungen, wesentliche Elemente und andere übergreifende Aspekte des Systems, die Einfluss auf Entwurf und Implementierung haben.

10.2. Architekturziele und Philosophie

- Plattformunabhängigkeit
- Nutzung von überall durch Webserverzugriff
- Unterschiedliche Benutzergruppen mit unterschiedlichen Rechten
- einfache Architektur

10.3. Annahmen und Abhängigkeiten

- (wenige Nutzer) ohne Informatikerausbildung oder Ähnlichem
- Stura Server vorhanden
- LXC Container wird bereitgestellt

10.4. Architektur-relevante Anforderungen

- 3.2.1
  - Daten können nach mehreren Kriterien gefiltert werden.
  - Das System sollte den Nutzer bei ungültigen Eingaben anleiten, wie diese auszusehen haben.
- 3.2.3
  - Mehrbenutzerbetrieb muss gewährleistet sein.
    - Serialisierung für die Datenbankarbeit.
- 3.2.4
  - Es soll bei freier Kapazität eine API für den Export eines Organigrammes erstellt werden.

10.5. Entscheidungen, Nebenbedingungen und Begründungen

1. Python seitens Auftraggeber gewünscht → Forderung akzeptiert, da Sinnhaftigkeit dieser Wahl erkannt. Phython ist weit verbreitet und gut geeignet zum Programmieren von Webanwendungen.

2. SQL Lite damit..
    - kein eigener Server aufgesetzt werden muss und zur zusätzlichen Sicherung, denn die Daten liegen nicht auf einem Datenbankserver vor?
    - einfach zu implementieren und zu verstehen
    - Projektumfang eher klein → SQL Lite reicht aus
3. Django als Webframework, da ..
    - SQL Lite ist Standard
    - in Python geschrieben - > leichte Anbindung
    - populär → Communityunterstützung, viele Entwickler können daran weiterarbeiten

10.6. Architekturmechanismen

1. Webserver : Speichern der Daten auf einem Server auf den von außen zugegriffen werden kann. Gängiger Architekturmechanismus für die Lagerung von Datenbanken.
2. Relationales DBS: Am weitesten Verbreitet. Bringt die meisten Vorteile zur Verwaltung von gängigen Daten mit.
3. Container (LXC): Arbeitet direkt mit dem vorinstallierten Linux-Kernel zusammen, benötigt demzufolge keine zusätzliche virtuelle Maschine. Lässt sich deshalb effizient und einfach einrichten.

10.7. Wesentliche Abstraktionen

1. Datenhaltung in einer relationalen Datenbank
2. robuste Verwaltung von Mitgliedern und Ämtern
3. Schutz vor Änderungen Unbefugter

10.8. Schichten oder Architektur-Framework

Die Webanwendung wird auf dem Webframework Django basieren. Es wird sich um eine Komponentenarchitektur handeln. Die Objekte der einzelnen Klassen werden in einer Datenbank mittels SQLite gespeichert. Die Webanwendung soll auf einem LXC Container des Stura-Webservers laufen.

[diagramm layers] | *../files/svg/diagramm_layers.svg*

10.9. Architektursichten (Views)

Folgende Sichten werden empfohlen:

10.9.1. Logische Sicht

[diagramm package classes] | *../files/svg/diagramm_package_classes.svg*

10.9.2. Physische Sicht (Betriebssicht)

1. Webserver → LXC Container → Webanwendung (Django)

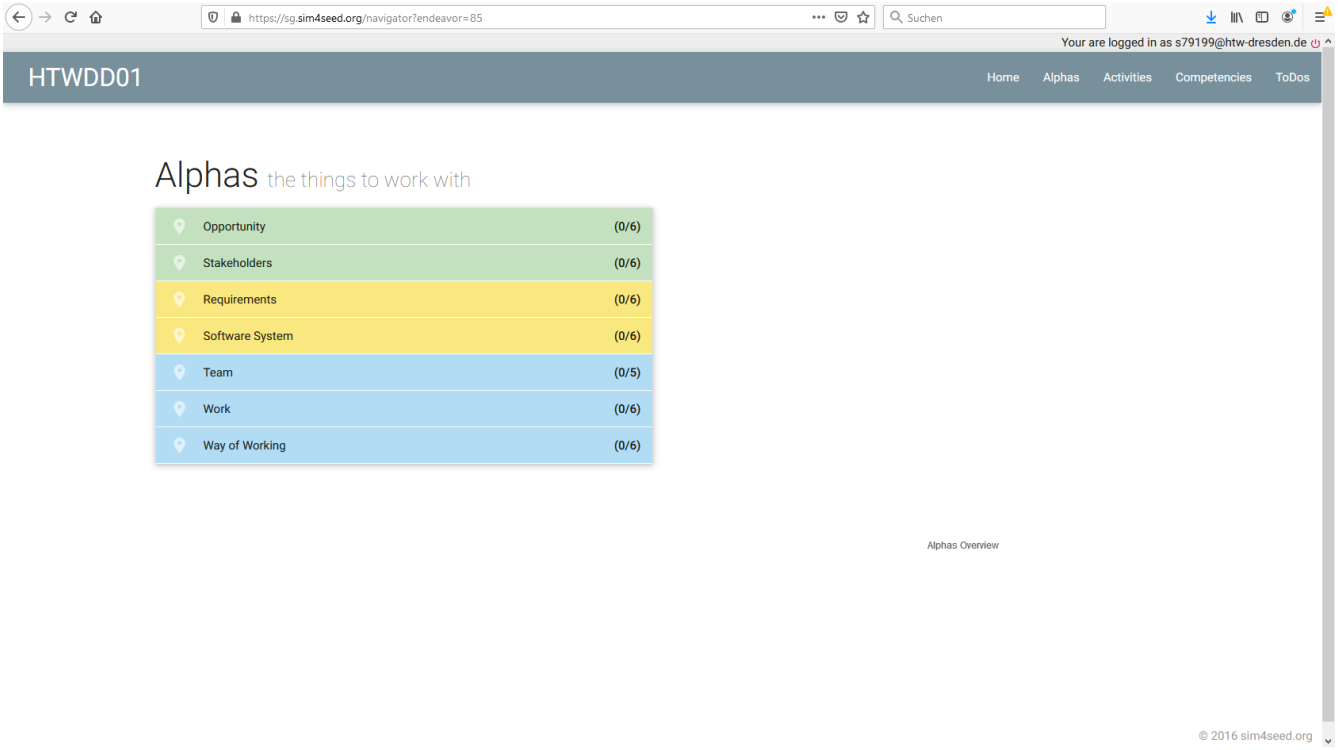
10.9.3. Use cases

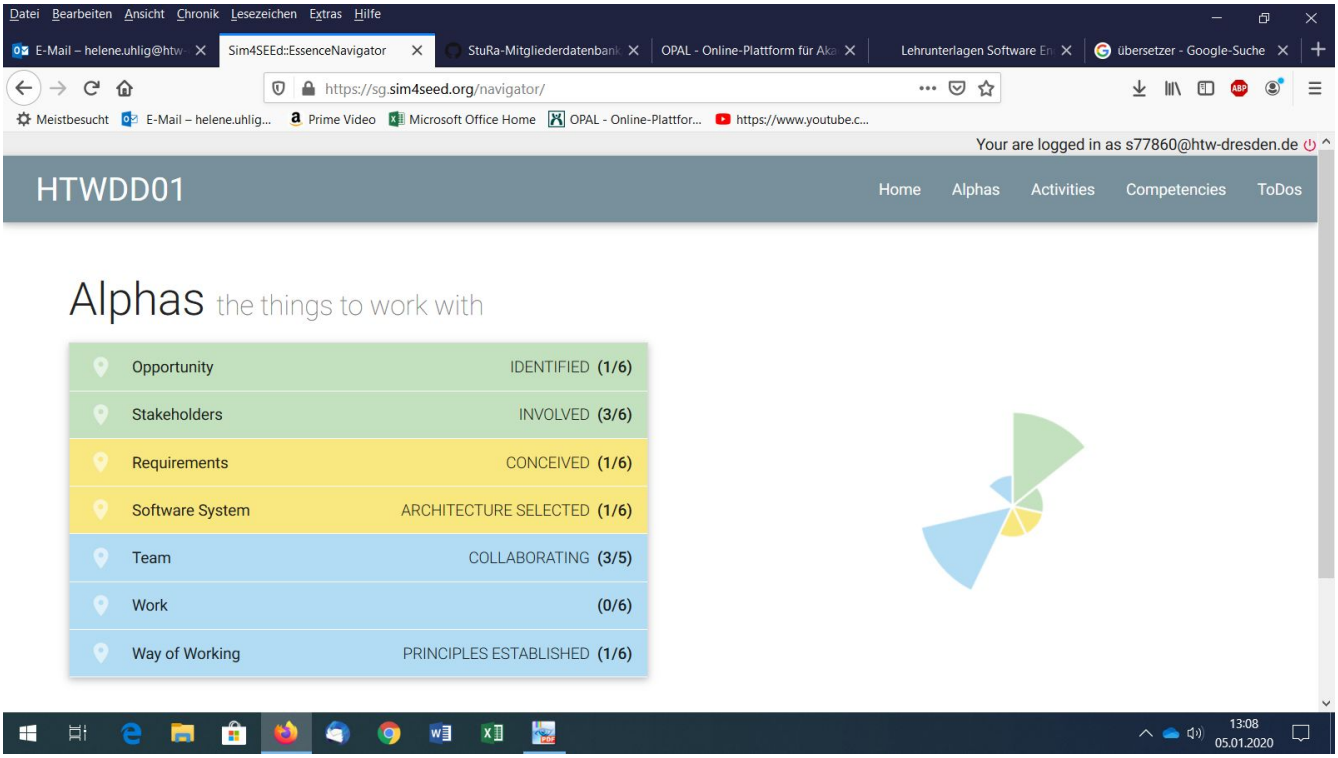
- 2.1 Mitglieder verwalten
- 2.4 Arbeitsleitfaden abarbeiten

11. Test-Cases

Zu den Testcases in der Developer Dokumentation

Essence Navigator



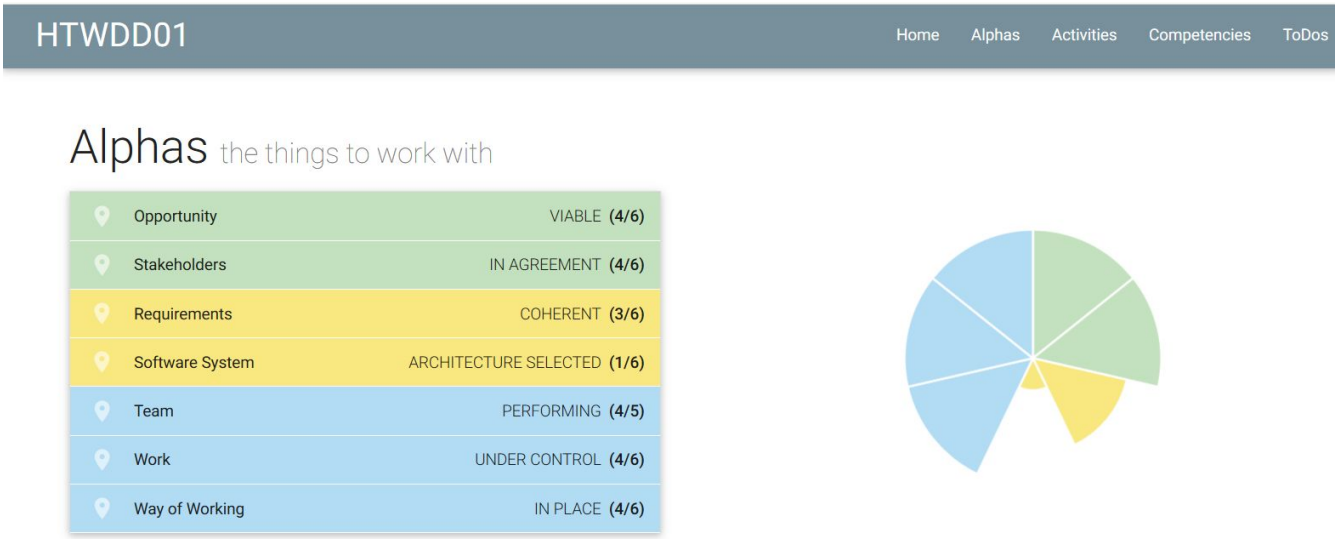


Alphas the things to work with

Opportunity	SOLUTION NEEDED	**(2/6)**
Stakeholders	IN AGREEMENT	**(4/6)**
Requirements	COHERENT	**(3/6)**
Software System	ARCHITECTURE SELECTED	**(1/6)**
Team	PERFORMING	**(4/5)**
Work	UNDER CONTROL	**(4/6)**
Way of Working	IN PLACE	**(4/6)**

Alphas Overview

12. neuste:


HTWDD01
Home
Alphas
Activities
Competencies
ToDos
Alphas the things to work with
Opportunity VIABLE (4/6)
Stakeholders IN AGREEMENT (4/6)
Requirements COHERENT (3/6)
Software System ARCHITECTURE SELECTED (1/6)
Team PERFORMING (4/5)
Work UNDER CONTROL (4/6)
Way of Working IN PLACE (4/6)