



UNIVERSITÄT OSNABRÜCK

INSTITUT FÜR INFORMATIK  
AG SOFTWARE ENGINEERING

*Masterarbeit*

# **Anbindung von Messaging-Systemen an Lernmanagementsysteme (am Beispiel von Stud.IP und Matrix)**

Manuel Schwarz

August 2021

Erstgutachter: Dr. Tobias Thelen  
Zweitgutachterin: Prof. Dr. Elke Pulvermüller



## **Zusammenfassung**

Die vorliegende Arbeit entstand in der Arbeitsgruppe Software Engineering an der Universität Osnabrück im Bereich Webentwicklung und befasst sich mit der Anbindung von Messaging-Systemen an Lernmanagementsysteme. Kern der Arbeit ist die Erstellung und sinnvolle Umsetzung eines Anforderungskatalogs für das oben genannte Thema am Beispiel der Messaging-Software Matrix und der Lernplattform Stud.IP.

Einleitend wird eine kurze Motivation des Themas gegeben. Daraufhin folgt ein Grundlagenkapitel mit allen für diese Arbeit relevanten Hintergrundinformationen. Die Anforderungsanalyse bildet das nächste Kapitel, in dem die Erhebung sowie alle daraus entstandenen Anforderungen aufgelistet und beschrieben werden. Anschließend wird die Umsetzung der herausgearbeiteten Anforderungen im Implementationskapitel dargestellt.

Abschließend endet die Arbeit mit einer kurzen, größtenteils technischen Evaluation sowie einem Ausblick, der sich mit möglichen Verbesserungen und Erweiterungen für den Praxiseinsatz beschäftigt.



# Inhaltsverzeichnis

<b>1</b>	<b>Motivation</b>	<b>1</b>
<b>2</b>	<b>Hintergrund</b>	<b>3</b>
2.1	Lernmanagementsysteme . . . . .	3
2.2	Stud.IP . . . . .	4
2.2.1	Stud.IP Plugins . . . . .	5
2.2.2	JSON:API . . . . .	6
2.2.3	Cronjobs . . . . .	7
2.3	Anbindung von Fremdsystemen an Stud.IP . . . . .	7
2.3.1	Meetings Plugin . . . . .	7
2.3.2	Etherpad Plugin . . . . .	10
2.3.3	Moodle-Connect Plugin . . . . .	11
2.3.4	LTI-Tools / Alija Plugin . . . . .	11
2.4	Messenger . . . . .	12
2.4.1	Matrix . . . . .	12
2.4.2	Matrix Schnittstellen . . . . .	12
<b>3</b>	<b>Anforderungsanalyse</b>	<b>13</b>
3.1	Umfrage zur Anforderungsanalyse . . . . .	13
3.2	Grundsatzentscheidung . . . . .	14
3.3	Anforderungen . . . . .	14
3.3.1	Übernahme von Veranstaltungsinformationen Veranstaltungen auf Räume abbilden + Übernahme der Teilnehmendenlisten . . . . .	14
3.3.2	Übernahme von Rollen, Rechten und Status . . . . .	14
3.3.3	kein zusätzliches Login . . . . .	15
3.3.4	Benachrichtigungsfunktion . . . . .	15
3.3.5	Transparenz . . . . .	16
3.3.6	Direktnachrichten . . . . .	16
3.3.7	mehrere Räume pro Kurs . . . . .	16
3.3.8	Kompatibilität von Nachrichten in Blubber und Matrix . . . . .	16
3.3.9	Übernahme des Stud.IP Profilbildes . . . . .	17
3.3.10	Stautsindikator in Stud.IP . . . . .	17
3.3.11	Opensource . . . . .	17

3.3.12	Anbindung von Matrix durch ein Stud.IP-Plugin . . . . .	17
3.3.13	keine zusätzliche API . . . . .	17
3.3.14	Sicherheit . . . . .	17
3.3.15	Robustheit, Ausfallsicherheit . . . . .	17
<b>4</b>	<b>Implementation</b>	<b>19</b>
4.1	Übernahme von Veranstaltungsinformationen Veranstaltungen auf Räume ab- bilden + Übernahme der Teilnehmendenlisten . . . . .	19
4.2	Übernahme von Rollen, Rechten und Status . . . . .	19
4.3	kein zusätzliches Login . . . . .	20
4.4	Benachrichtigungsfunktion . . . . .	20
4.5	Transparenz . . . . .	21
4.6	Direktnachrichten . . . . .	21
4.7	mehrere Räume pro Kurs . . . . .	21
4.8	Kompatibilität von Nachrichten in Blubber und Matrix . . . . .	21
4.9	Übernahme des Stud.IP Profilbildes . . . . .	22
4.10	Stautsindikator in Stud.IP . . . . .	22
4.11	Opensource . . . . .	22
4.12	Anbindung von Matrix durch ein Stud.IP-Plugin . . . . .	22
4.13	keine zusätzliche API . . . . .	23
4.14	Sicherheit . . . . .	23
4.15	Robustheit, Ausfallsicherheit . . . . .	23
<b>5</b>	<b>Evaluation</b>	<b>25</b>
5.1	Technische Tests . . . . .	25
5.1.1	Unit-Tests . . . . .	25
5.1.2	Code Coverage . . . . .	25
5.2	Chancen . . . . .	25
5.3	Probleme . . . . .	25
<b>6</b>	<b>Ausblick</b>	<b>27</b>

# Abbildungsverzeichnis

2.1	LMS Architektur . . . . .	4
2.2	Stud.IP Pluginverwaltung . . . . .	5
2.3	Stud.IP JSON:API Ablaufdiagramm . . . . .	7
2.4	Meetings-Plugin: Dialog bei Raumerstellung . . . . .	8
2.5	Etherpad: Einbettung mit <b>iframe</b> . . . . .	11
3.1	Umfrage: Teilnehmendenverteilung . . . . .	13





# Kapitel 1

## Motivation

Lehren und Lernen im digitalen Zeitalter.

Moderne Kommunikationswege und -mittel studiumsunterstützend einsetzen.

Aktuelle Werkzeuge nutzen, um Studierenden eine möglichst niedrige Einstiegsschwelle bei Fragen oder Unklarheiten zu bieten.

Fortwährender Prozess der Weiterentwicklung und Anpassung der Vermittlung von Informationen.

Mailinglisten, Foren, Instant-Messenger ...

Wie studiert man heute? Wie ist die durchschnittliche Nutzung der Studierenden von WhatsApp und Co.? (Bezug auf die Studien)

Digitalisierung generell hervorheben, speziell in den vergangenen Corona- bzw. Digitalsemestern noch einmal zusätzlich an Relevanz gewonnen.



# Kapitel 2

## Hintergrund

Das folgende Kapitel befasst sich knapp mit den Grundlagen, die dieser Arbeit zu Grunde liegen. Begonnen bei Lernmanagementsystemen und Messaging-Systemen, hin zu den konkreten Implementationen Stud.IP und Matrix sowie Beispielen für die Anbindung anderer Fremdsysteme an die Lernplattform Stud.IP.

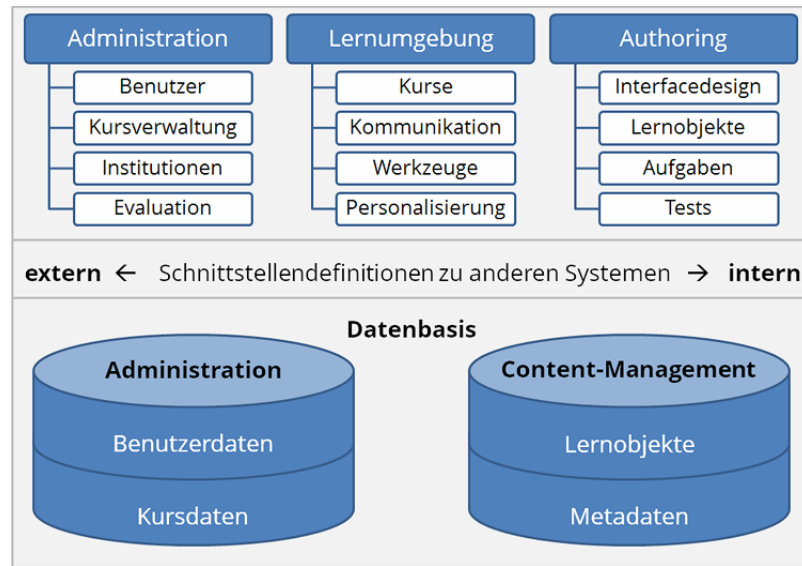
### 2.1 Lernmanagementsysteme

Lernmanagementsysteme, im Folgenden mit LMS abgekürzt, dienen der Unterstützung von Lehr- und Lernprozessen im Kontext von E-Learning sowie der Verwaltung von Lehrmaterialien und Nutzerdaten. Diese Online-Lernplattformen bilden an vielen Hochschulen aber auch anderen Institutionen die Basis einer E-Learning-Infrastruktur.

Die Hauptfunktion, die ein LMS in der Regel erfüllt, ist die Organisation des Lehrbetriebs, insbesondere das Anlegen von Veranstaltungen, die Bereitstellung von Lerninhalten und die Kommunikation zwischen Lehrenden und Lernenden.

Definition nach Schulmeister [3, S. 10]: Als Lernplattformen oder Learning Management Systeme werden im Unterschied zu bloßen Kollektionen von Lehrskripten oder Hypertext-Sammlungen auf Webservern solche Software-Systeme bezeichnet, die über folgende Funktionen verfügen:

- Eine Benutzerverwaltung (Anmeldung mit Verschlüsselung)
- Eine Kursverwaltung (Kurse, Verwaltung der Inhalte, Dateiverwaltung)
- Eine Rollen- und Rechtevergabe mit differenzierten Rechten
- Kommunikationsmethoden (Chat, Foren) und Werkzeuge für das Lernen (Whiteboard, Notizbuch, Annotationen, Kalender etc.)
- Die Darstellung der Kursinhalte, Lernobjekte und Medien in einem netzwerkfähigen Browser.



**Abbildung 2.1:** LMS Architektur [3, S. 11]

Die Abbildung 2.1 zeigt die Architektur eines idealtypischen LMS. Auf der Ebene der Datenbasis bildet meist ein Content-Management-System die Grundlage der Datenverwaltung. Neben Personen- und Kursdaten, können ebenso Lerninhalte archiviert, verteilt und wiederverwendet werden.

Darauf aufbauend ist es möglich je nach Bedarf interne sowie externe Schnittstellen zu anderen Systemen zu definieren.

Auf höchster Abstraktionsebene stellt das Diagramm drei Hauptkategorien der Nutzung dar. Zunächst sollte es eine administrative Ebene mit einer Nutzer-, Institutions-, und Kursverwaltung sowie der Option für Evaluationen geben. Die zweite Kategorie repräsentiert eine Lernumgebung mit ihren Werkzeugen und Kommunikationswegen. Der letzte Block beschreibt das Authoring, dass die konkreten Lernobjekte, Aufgaben und Tests umfasst.

## 2.2 Stud.IP

Stud.IP ist eine Open Source Implementation eines LMS, dass Anfang der 2000er Jahre entwickelt wurde. Mit ihren offenen Schnittstellen erlaubt diese digitale Lernplattform die Integration von externen Systemen und Anwendungen zum Einsatz in der digitalen Lehre. Neben Hochschulen gehören ebenfalls Schulen, Unternehmen, Verbände und Behörden zu den Einsatzgebieten von Stud.IP. Insgesamt gibt es derzeit ca. 70 Installationen mit ungefähr 600.000 Nutzerinnen und Nutzern (Quelle: studip.de).

Stud.IP versucht durch stetige Verbesserungen und Anpassungen den Anforderungen der sich ändernden Gegebenheiten der digitalen Lehre gerecht zu werden. Dabei ist ein kontinuierlicher Dialog zwischen Lehrenden, Lernenden sowie Entwicklern essentiell.

Stud.IP erfüllt alle im vorherigen Abschnitt 2.1 aufgeführten Anforderungen an ein LMS. Neben einer ausgereiften Benutzer- und Kursverwaltung inklusive eines Rollen- und Rechtessystems, existiert ebenfalls ein Forum und ein Chat zur Kommunikation sowie diverse Lernwerkzeuge (Etherpad, Wiki, Courseware, Vips). Stud.IP verfügt über diverse Werkzeuge sowie Schnittstellen zur Integration von externen Systemen und Anwendungen. Im Folgenden werden die für diese Arbeit relevanten Schnittstellen kurz beschrieben.

### 2.2.1 Stud.IP Plugins

Die Lernplattform Stud.IP besitzt eine Plugin-Schnittstelle, mit dessen Hilfe das LMS um beliebige Funktionen und Werkzeuge erweitert werden kann (Quelle: <https://hilfe.studip.de/develop/Entwickler/Plugin>). Der Grund hierfür sind die mitunter sehr individuellen Anforderungen jedes Standorts, an dem Stud.IP betrieben wird. Mit Hilfe von Plugins lässt sich die Software bedarfsgenau konfigurieren ohne direkte Änderungen am Kern-Programmcode vornehmen zu müssen. Plugins werden innerhalb von Stud.IP über den sogenannten Pluginmarktplatz bereitgestellt und lassen sich als Administrator einfach installieren und aktivieren bzw. deinstallieren oder deaktivieren.

Ein Ausschnitt der Administrationsoberfläche für die Pluginverwaltung ist in Abbildung 2.2 zu





Verwaltung von Plugins						28 Plugins (28/0)
Aktiv	Name	Typ	Version	Schema	Position	Aktionen
<input checked="" type="checkbox"/>	ActivityFeed <i>(Kern-Plugin)</i>	PortalPlugin	1.0		<input type="text" value="8"/>	
<input checked="" type="checkbox"/>	Blubber <i>(Kern-Plugin)</i>	CorePlugin, StandardPlugin, StudipModule	3		<input type="text" value="1"/>	
<input checked="" type="checkbox"/>	ContentsWidget <i>(Kern-Plugin)</i>	PortalPlugin	1.0		<input type="text" value="9"/>	
<input checked="" type="checkbox"/>	CoreAdmin <i>(Kern-Plugin)</i>	CorePlugin, StudipModule			<input type="text" value="1"/>	

Abbildung 2.2: Stud.IP Pluginverwaltung

sehen. Zusätzlich zu dem Namen, der Version, dem Status und dem Aktionsmenü wird ebenfalls der Plugintyp angezeigt, von dem es verschiedene gibt, worauf ich im Implementationsteil weiter eingehen werde. Hier sei nur kurz die Besonderheit von Kern-Plugins erwähnt, die bereits bei der Standardinstallation von Stud.IP mitgeliefert und installiert werden.

### 2.2.2 JSON:API

Die JSON:API ist eine Spezifikation, die Konventionen dafür festlegt, wie ein Client den Abruf oder die Änderung von Ressourcen anfordert (Request) und wie ein Server auf diese Anforderungen reagieren soll (Response). Die allgemeine JSON:API-Spezifikation wurde entwickelt, um sowohl die Anzahl der Anfragen als auch die Menge der zwischen Clients und Servern übertragenen Daten zu minimieren. Dieses Effizienzziel wird erreicht, ohne die Lesbarkeit, Flexibilität oder Auffindbarkeit zu beeinträchtigen. (Quelle: <https://jsonapi.org/format/>)

Stud.IP verfügt über eine solche JSON:API-Schnittstelle, welche die oben genannte allgemeine Spezifikation erfüllt und mit deren Hilfe diverse Daten abgerufen sowie dem System hinzugefügt können.

Für Stud.IP existieren verschiedenste JSON:API-Routen, die unter folgender URI erreichbar sind:

`https://<meine.studip.installation.de>/jsonapi.php/v1/<routen>`

Ein Beispielaufruf zur Anforderung der Daten eines bestimmten Semester (eindeutigen ID) hat diese Form:

`https://<meine.studip.installation.de>/jsonapi.php/v1/semester/<ID>`

Die zugehörige Antwort im JSON-Format sieht wie folgt aus:

```

1  {
2    "data": {
3      "type": "semesters",
4      "id": "322f640f3f4643ebe514df65f1163eb1",
5      "attributes": {
6        "title": "SS 2021",
7        "description": "",
8        "token": "",
9        "start": "2021-04-01T00:00:00+02:00",
10       "end": "2021-09-30T23:59:59+02:00",
11       "start-of-lectures": "2021-04-12T00:00:00+02:00",
12       "end-of-lectures": "2021-07-16T23:59:59+02:00",
13       "visible": true
14     },
15     "links": {
16       "self": "\trunk\jsonapi.php\v1\semesters\322f640f3f4643ebe514df65f1163eb1"
17     }
18   }
19 }
```

Der Client bekommt die JSON-Repräsentation eines Stud.IP-Semesters zurückgeliefert, welche alle relevanten Daten der Objekts enthält. Routen repräsentieren die verschiedenen Stud.IP-

Datenstrukturen und werden mit Hilfe einer `RouteMap` von der angefragten URI auf den entsprechenden Programmcode abgebildet. Die Abbildung 2.3 stellt schemenhaft den Ablauf der Abarbeitung eines JSON:API-Requests dar. Nach einem initialen Request folgt die bereits an-



**Abbildung 2.3:** Stud.IP JSON:API Ablaufdiagramm

gesprochene Routen-Zuordnung, welche den Programmcode bestimmt, der ausgeführt werden soll. Der Routen-Handler kümmert sich um eine JSON:API-konforme Antwort. Die folgende Schema-Zuordnung legt fest welche Schemaklasse bestimmte Stud.IP-Objekte in JSON umwandeln kann. Im nächsten Schritt wird eben dieses Schema ausgewählt, welches die konkrete Abbildung eines Stud.IP-Objekts in JSON definiert. Daraufhin kann die JSON:API-Antwort an den Client gesendet werden.

(Quelle: <https://hilfe.studip.de/develop/Entwickler/JSONAPI>)

### 2.2.3 Cronjobs

## 2.3 Anbindung von Fremdsystemen an Stud.IP

Im Kontext des Lernmanagementsystems Stud.IP existieren mehrere Beispiele für die Anbindung oder Einbettung von anderen Systemen an bzw. in die LMS-Software. Im Folgenden werden ein paar solcher Integrationen konkret aufgeführt.

### 2.3.1 Meetings Plugin

Meetings ist ein Stud.IP Videokonferenzplugin (Quelle github), das das LMS mit unterschiedlichen Videokonferenzsoftwarelösungen, wie z.B. BigBlueButton (Quelle: [bigbluebutton.org](https://bigbluebutton.org)), verknüpfen kann. Das Plugin ermöglicht es direkt aus einer Stud.IP-Veranstaltung heraus Videokonferenzräume für z.B. Vorlesungen, Webinare, Vorträge oder Gruppenarbeiten anzulegen, denen die Teilnehmer\*innen der Veranstaltung beitreten können (dank der Rechteverwaltung in Stud.IP). Per Einladungslink ist es auch Personen ohne ein Stud.IP-Nutzerkonto (z.B. Gastredner) möglich an Videokonferenzen teilzunehmen.

Neben den zu erwartenden Kommunikationswerkzeugen wie Mikrofon, Kamera und einer Chatfunktion können zusätzlich gemeinsam Präsentationen angesehen, Texte im integrierten Etherpad kooperativ erstellt und der eigene Bildschirm für alle Teilnehmenden freigeschaltet werden. Das Plugin ist somit eine ideale Erweiterung für dezentrales Lernen und Arbeiten, insbesondere da auch aus einfachen Stud.IP-Studiengruppen heraus Meetings-Räume erstellt werden können, außerhalb des strikten Lehrveranstaltungscontextes. Dies hilft Lehrenden bei der Vernetzung und Studierenden bei der Bildung von Arbeits- und Lerngruppen.

Weiterhin besteht auch die Möglichkeit Videokonferenzen aufzuzeichnen, was die Nacharbeitung oder Wiederholung von Lehrinhalten bei einem verpassten Meeting um ein Vielfaches erleichtert. (Quelle: <https://hilfe.studip.de/help/4.4/de/Basis/Meetings>) Die folgende Abbildung 2.4 zeigt einen Screenshot des „Raum erstellen“-Dialogs in dem Meetings Plugin mit all seinen Optionen.

**Raumkonfiguration**

**Raumname**

**Konferenz Systemeinstellung**

**Konferenzsystem \***

BigBlueButton

**Verfügbare Server \***

Bitte wählen Sie einen Server aus

**Zusätzliche Funktionen**

- ☐ Alle Teilnehmenden haben Moderationsrechte
- ☒ Alle Teilnehmenden initial stumm schalten
- ☒ Jeder Teilnehmer kann die Konferenz starten
- ☐ Nur Moderatoren können Webcams sehen ⓘ
- ☐ Nur Moderatoren können Webcams teilen
- ☐ Nur Moderatoren können Audio teilen
- ☐ Gemeinsame Notizen deaktivieren
- ☐ Private Chats deaktivieren
- ☐ Moderatoren vor Teilnehmendenzutritt fragen ⓘ
- ☐ Zugang via Link ⓘ

Maximale Teilnehmerzahl

50

Minuten Konferenzdauer (Max. Limit: 1440 Minuten) ⓘ

240

Willkommensnachricht ⓘ

Welcome to <b>%%CONFNAME%%</b>!  
For help on using B

**Aufzeichnung**

- ☐ Sitzungen können aufgezeichnet werden. **beta**
- ☒ Aufzeichnungen für Teilnehmende sichtbar schalten ⓘ

**Automatisches hochladen von Materialien ⓘ**

Aktuell ausgewählter Ordner: Kein Ordner

**Name**

- Allgemeiner Dateiordner
- Aufgaben-Plugin
- Martins
- nils
- Ordner tests

Neuer Ordner

**Abbildung 2.4:** Meetings-Plugin: Dialog bei Raumerstellung

Neben den Grundeinstellungen wie **Raumname** und **Systemeinstellung** können noch diverse andere Konfigurationsoptionen ausgewählt werden, die je nach selektiertem Konferenzsystem unterschiedlich ausfallen. Die Abbildung 2.4 verdeutlicht welche Auswahlmöglichkeiten in BigBlueButton zur Verfügung stehen und stellt die Benutzeroberfläche dar, die als Grundlage der im Folgenden beispielhaft beschriebenen BigBlueButton API dient.

Um einen API-Aufruf an den BigBlueButton-Server zu senden, stellt man eine HTTPS-Anfrage an den API-Endpunkt des BigBlueButton-Servers (normalerweise bestehend aus dem Hostna-



men des Servers gefolgt von `/bigbluebutton/api`). Alle API-Aufrufe müssen eine Prüfsumme enthalten, die mit Hilfe eines mit dem BigBlueButton-Server geteilten Geheimnis berechnet wird.

Der BigBlueButton-Server gibt daraufhin eine Antwort im XML-Format für alle API-Aufrufe zurück. Beispielhaft ist hier der API-Aufruf zum Erstellen eines Meetings in BigBlueButton aufgeführt (`create`):

`http://mein.server.de/bigbluebutton/api/create?[parameters]&checksum=[checksum]`

(Quelle: <https://docs.bigbluebutton.org/dev/api.html#create>)

Die konkrete Implementation auf Programmcodeebene sieht folgendermaßen aus:

```

1  <?php
2  public function createMeeting(MeetingParameters $parameters)
3  {
4      $params = array(
5          'name' => $parameters->getMeetingName(),
6          'meetingID' => $parameters->getRemoteId() ?: $parameters->getMeetingId(),
7
8          // ...
9      );
10
11     // ...
12     $response = $this->performRequest('create', $params, $options);
13
14     $xml = new \SimpleXMLElement($response);
15     // ...
16     return isset($xml->returncode) && strtolower((string)$xml->returncode) === 'success';
17 }
18
19 private function performRequest($endpoint, array $params = array(), array $options = [])
20 {
21     $uri = 'api/'. $endpoint. '?' . $this->buildQueryString($params);
22
23     //...
24
25     $method = (is_array($options) && count($options)) ? 'POST' : 'GET';
26     $request = $this->client->request($method, $this->url . '/' . $uri, $options);
27     return $request->getBody(true);
28 }
```

Zum besseren Verständnis werden hier nur die relevanten Abschnitte und Methoden gezeigt. Die Funktion `createMeeting()` sammelt zunächst alle Parameter, die zur Erstellung eines Meetings relevant sind und deren Werte zuvor in der Nutzeroberfläche ausgewählt wurden.

In Zeile 12 wird anschließend die Methode `performRequest()` mit dem entsprechenden Schüs-

selwort `create` und den Parametern aufgerufen. Bei einem Erfolg wird `true` zurückgegeben.

Die Funktion `performRequest()` wird ab Zeile 19 beschrieben und zeigt auf, wie die konkrete Serveranfrage mit dem jeweiligen Endpoint entsprechend der oben skizzierten allgemeinen Spezifikation zusammengestellt wird. Am Ende wird der die XML-Antwort des Servers im Response-Body zurückgegeben.

Damit ist die Funktionalität des Stud.IP Meetings Plugins grundsätzlich gezeigt und die Anbindung von BigBlueButton mit Hilfe der API deutlich gemacht.

### 2.3.2 Etherpad Plugin

Die webbasierte Open-Source-Software Etherpad ist ein kollaborativer Echtzeit-Editor, der das Bearbeiten eines Dokuments durch viele Nutzer\*innen gleichzeitig erlaubt (Quelle: <https://github.com/ether/etherpad-lite>).

Jedem Teilnehmenden wird dazu eine eindeutige Farbe beim Editieren des Textes zugewiesen. Die integrierte Versionierung macht es zudem einfach vorherige Änderungen nachzuvollziehen oder auf eine alte Version zurückzugreifen, falls nötig. Damit eignet sich Etherpad insbesondere gut für Brainstorming, Protokolle, Sammlungen, Gruppen- und Seminararbeiten oder ähnliches.

Das Etherpad Plugin ist ein weiteres Modul mit dem das externe Programm Etherpad in Stud.IP eingebunden wird. Das Plugin verbindet die Lernplattform mit einem Etherpad-Lite Server, der in vielen Fällen von der jeweiligen Institution selbst gehostet werden kann.

Dozenten oder Tutoren beliebig viele der sogenannten Pads in einer Stud.IP-Veranstaltung bzw. -Studiengruppe anlegen. Innerhalb einer Veranstaltung können die Teilnehmenden anschließend die bereits existierenden Pads zum gemeinsamen Arbeiten nutzen.

Die technische Umsetzung der Einbettung des Editors in Stud.IP sieht folgendermaßen aus und geschieht mit Hilfe eines `iframe`, dessen Ausmaße auf sinnvolle Weise beschränkt werden. (Quelle: <https://github.com/luniki/studip-plugin-studipad>)

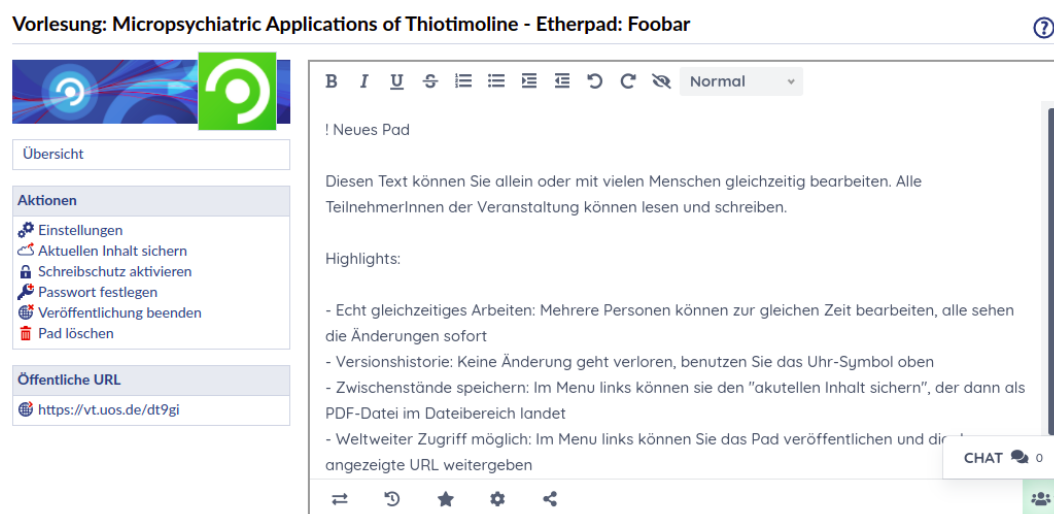
```

1 <iframe id="etherpad"
2     src="<?= $controller->link_for('pads/open', $padid) ?>"
3     style="width:100%">
4 </iframe>
5 <script>
6 jQuery(window).on('message onmessage', function (e) {
7     var msg = e.originalEvent.data;
8     if (msg.name === 'ep_resize') {
9         var width = msg.data.width;
10        var height = msg.data.height;
11        jQuery("#etherpad").height(Math.min(Math.max(height, 400), 1080))
12    }
13 });
14 </script>

```

Die Benutzeroberfläche eines konkreten Pads einer Veranstaltung in Stud.IP ist in Abbildung 2.5

dargestellt. Neben dem eingebetteten Editor mit seinen Formatierungsmöglichkeiten verfügt das



**Abbildung 2.5:** Etherpad: Einbettung mit `iframe`

Plugin über ein Stud.IP-Aktionsmenü mit erweiterten Funktionen wie beispielsweise dem PDF-Export des Textes („Aktuellen Inhalt sichern“) und einer Veröffentlichungsfunktion, wodurch das Pad auch außerhalb von Stud.IP per generiertem Link zu erreichen ist.

### 2.3.3 Moodle-Connect Plugin

Mit Hilfe des Moodle-Connect Plugins lassen sich Veranstaltungen aus Stud.IP mit Veranstaltungen des Lernmanagementsystems Moodle verknüpfen (Quelle github).

Genauer gesagt lassen sich Stud.IP Veranstaltungen auf Moodle Veranstaltungen abbilden.

- Veranstaltung aus Stud.IP wird in Moodle übernommen
- Übernahme der Teilnehmer
- Screenshot(s)
- Button in Stud.IP
- Funktionsweise (Moodle + Stud.IP konfigurieren, API Nutzung, Tokengenerierung)
- mit Till sprechen!

### 2.3.4 LTI-Tools / Alija Plugin

Dieses Plugin bietet einen einfachen Mechanismus für eine Weiterleitung in eine externe Anwendung mit automatischer Anmeldung unter dem gleichen Account (Single-Sign-On) (Quelle svn Repo/Stud.IP Hilfe). Unterstützt werden dabei LTI-Tools (Version 1.x der Schnittstelle) sowie Stud.IP-Installationen (entweder über LTI oder das Alija-Protokoll). Das Plugin kann auf

Veranstaltungs- oder Einrichtungsebene aktiviert werden. Ähnlich wie bei der "Freien Informationsseite" kann der Titel des Reiters sowie ein freier Text angegeben werden, der den Nutzern zur Erläuterung angezeigt wird.

Als Gegenstück zu diesem Plugin benötigt man auf der anderen Seite entweder ein mit dem LTI-Standard 1.x kompatibles LTI-Tool oder ein Stud.IP mit dem StudipAuthLTI oder StudipAuthAlia Auth-Plugin.

- Screenshot
- Einbettung per iframe oder Verlinkung zur jeweiligen externen Anwendung.
- evtl. mit Elmar sprechen.

## 2.4 Messenger

Was ist das?

### 2.4.1 Matrix

Konkretes Beispiel eines Messenger-Backends. Matrix ist ein offener Standard für interoperable, dezentralisierte Echtzeitkommunikation über IP. Er kann für Instant Messaging, VoIP/WebRTC-Signalisierung, Kommunikation im Internet der Dinge oder überall dort eingesetzt werden, wo eine Standard-HTTP-API für die Veröffentlichung und das Abonnieren von Daten bei gleichzeitiger Verfolgung des Gesprächsverlaufs benötigt wird.

Matrix definiert den Standard und stellt Open-Source-Referenzimplementierungen von Matrix-kompatiblen Servern, Client-SDKs und Anwendungsdiensten zur Verfügung, die Ihnen helfen, neue Kommunikationslösungen zu entwickeln oder die Fähigkeiten und Reichweite bestehender Lösungen zu erweitern.

### 2.4.2 Matrix Schnittstellen

<https://matrix.org/docs/guides/client-server-api>

## Kapitel 3

# Anforderungsanalyse

Kernpunkt der Arbeit, auf dem der Hintergrund und die Implementation aufbauen. Empirischer Teil: Anforderungsanalyse mit Hilfe eines (Mini-) Fragebogens.

### 3.1 Umfrage zur Anforderungsanalyse

Was trifft am ehesten auf Sie zu?

20 responses

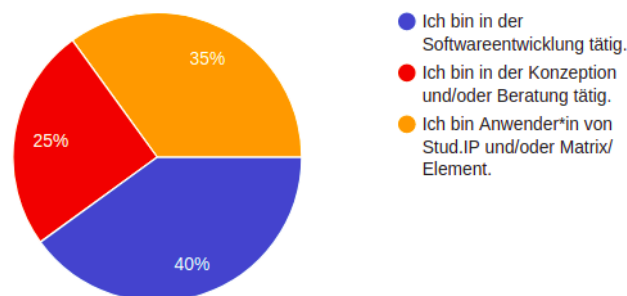


Abbildung 3.1: Umfrage: Teilnehmendenverteilung

Unterkapitel für jede herausgearbeitete Anforderung (funktional + nicht funktional). **Support-Hotline-Raum** Eine Support-Hotline-Funktion bei der man eine Gruppe von Personen erreichen kann, um ein Anliegen zu besprechen und der Chat beendet wird, nach dem das Anliegen erledigt ist bzw. nur der Kunde aus dem Chat entfernt wird.

## 3.2 Grundsatzentscheidung

Möglichkeit der beiden entwickelten Konzepte darstellen und die Entscheidung begründen. Konzept 1: vollständige Bridge (möglichst voller Funktionsumfang) / eigener Client - Aufwandsabschätzung, hohe Komplexität (element-web github Statistiken heranziehen) - hoher Pflegeaufwand - mit Element ist ein guter Client vorhanden, wieso alles nachbauen? Konzept 2: Lose Kopplung, Anbindung statt Integration - sinnvolle Diskussion, wie so eine Anbindung aussehen könnte - zwei klar getrennte Systeme (Transparenz) es wurden einige Anforderungen genannt, die nicht behandelt werden müssen, da diese bereits im Element-Client vorhanden sind Nachrichten senden und empfangen (Basisfunktionen) Übersichtliches UI Klare Übersicht (Nachrichten und Bedienung) undabhängig von OS/Browser/Screen size (Portabilität). Gute Usability, wenig Klicks zum Bilden von Arbeitsgruppen, Direktchats, usw. und ein modernes Erscheinungsbild. Matrix/Element in Blubber verlinken: Chat/Matrix/Element-Icon in Blubber anzeigen, mit dem man zu dem Uni-Matrix/Element-Raum gelangt. **Matrix als Hauptmessagingdienst** Weg von Blubber. Entscheidung der Dozenten, was genutzt wird Blubber abschalten, müssen die Betreiber entscheiden **Stummschalten von (Veranstaltungs-) Räumen** In Stud.IP ist es bisher nicht vorgesehen einzelne Blubber-Nachrichten als gelesen zu markieren.

## 3.3 Anforderungen

### 3.3.1 Übernahme von Veranstaltungsinformationen Veranstaltungen auf Räume abbilden + Übernahme der Teilnehmendenlisten

Kurse, Studengruppen, ... sollten (automatisch) auf Räume in Matrix abgebildet werden. Reiter: Chat mit konfigurationseinstellungen - nach Aktivierung nur für Doz und Tutoren sichtbar, die sollten die Erzeugung des Matrixraums anstoßen, nur, wenn eine Parametrisierung notwendig, sonst mit Pluginaktivierung automatisch machen - was passiert beim Löschen in Stud.IP und beim erneuten Anlegen? - wenn Raum erzeugt wurde, ist der Reiter auch für Studierende sichtbar mit Anleitung (Einfach halten, "Platzhalter"), + Knopf für Profilerstellung/Übertragung zu Matrix

### 3.3.2 Übernahme von Rollen, Rechten und Status

Keine automatische Übernahme der Teilnehmerliste, Datenschutz, einzeln Übernahme anstoßen pro Teilnehmer, Sichtbarkeitseinstellungen in Matrix nicht abbildbar. (mit Elmar sprechen!, Axel, Jens) manuelle Nutzerverwaltung in Matrix sollte nicht genutzt werden, zwischen Doz und Tutoren nicht unterscheiden. Userstatus spielt in der Praxis kaum eine Rolle. Synchen der User. User = autoren Anbindung von Fremdsystemen mit einem anderen Rollen-Rechte-Konzept. Feingranulare Berechtigungen für Studierende ist nur eingeschränkt umsetzbar, da es in Matrix nur 3 Rechtstufen gibt. Denkbar wäre grob: Dozenten -> Moderator/Admin und Studierende -> Default. In Matrix lassen sich zusätzlich benutzerdefinierte Rechtstufen erstellen, eventuell

lassen sich damit alle in Stud.IP verfügbaren Rechtestufen abbilden.

Der eigene Status sollte leicht einzustellen sein oder aus Stud.IP übernommen werden (sichtbar/nicht sichtbar) Man könnte in Matrix auch manuelle Rollen anlegen Datenschutzaspekt: Ich möchte unsichtbar sein, in Matrix schwierig. Datenschutzrechtliche Begründung, warum steht der Name der Studenten in der Raumliste? beibehalten der Kommunikationskultur, keine anonymen Beiträge möglich in Matrix raumbezogene Einstellung: Teilnehmerliste ist sichtbar oder nicht in synchronisierten Räumen wir die Teilnehmerliste nicht angezeigt, kann Matrix das? Diskussion, was mache ich jetzt? 1. ich halte fest, es ist ein offenes Problem, Umsetzung trotz nicht datenschutzkonform 2. Leute, die sich unsichtbar geschaltet haben, werden nicht mitsynchronisiert, die bekommen evtl nichts mehr mit 3. Leute, die in min 1 VA unsichtbar sind -> mit Synonym sichtbar machen. - jeder Student muss von Stud.IP aus seine Zustimmung geben, das weitergeben des Nutzers an Matrix aktivieren (ist das sinnvoll?) gibt es Alternativen? ich sehe keinen guten Grund, das eine oder andere zu machen. -> Workshop mit Entwicklern und Didaktikern -> Ergebnisse -> Aus technischer Sicht gibt es keinen Grund, das eine oder das andere zu nehmen, Workshop hat ergeben: folgendes Ergebnis Beim Austragen oder Ändern der Rechtestufe in Matrix synchron zu halten. -> Cronjob einmal stündlich Lose Kopplung, nur einmal anlegen, was danach mit dem Raum passiert ist offen

### 3.3.3 kein zusätzliches Login

Nur ein Login für beide Systeme/Dienste. Single Sign-On Stud.IP Nutzer sollten ohne zusätzliches Login in einen Matrix Raum gelangen können.

### 3.3.4 Benachrichtigungsfunktion

**Synchronizität** Zum einen sollten Matrix und Blubber zeitlich synchron sein. Zum anderen wäre es wünschenswert, wenn die in einem System als gelesen markierte Nachrichten auch im anderen System als gelesen markiert würden.

Man möchte in Matrix über relevante Ereignisse informiert werden. (-> später, was ist eigentlich relevant, Tabelle: Relevanzabschätzung für Ereignisse, was ist relevant für alle? persönliche Präferenzen kann man mit dem hier entschiedenen Modell nicht berücksichtigen, deshalb sind folgende Konfiguration pro Veranstaltung sinnvoll? Welche Ereignisse sollen erscheinen, oder nicht? Diskussion, was spricht dafür, was spricht dagegen? Aus studentensicht evtl. verwirrend -> Argument gegen Konfig. Elmar oder Marcus -> Notifications, wenn z.B. eine Datei hochgeladen wird. NotificationCenter::addObserver Richtung Stud.IP -> Matrix, oder Matrix -> Stud.IP (nicht genannt, aber der Vollständigkeit halber dennoch genannt) braucht man eine Konfigurationsmöglichkeit dafür? wichtige Infos über die Kopplung hinweg. Informationsveranstaltung mit uniübergreifenden Infos, denkbar. (Eher ungeeignet, nach unten): unibezogene/veranstaltungsübergreifende Informationen, wie z.B. Rückmeldefristen, Unischließung, ... (dies wäre ein reiner Infochannel, in dem alle Studierenden sein müssten, keine Antwortmöglichkeit -> one-way Kommunikation -> eMail/Stud.IP Benachrichtigung wie bisher ist da

vielleicht besser) Besser: veranstaltungsbezogene Informationen, z.B. eine Datei wurde hochgeladen", "Termin fällt aus"

### 3.3.5 Transparenz

(wird tendenziell nicht umgesetzt -> nur Konzept) Verknüpfung zwischen Stud.IP und Matrix/Element deutlich machen. Welche Konsequenzen hat das eigene Handeln in Matrix, wenn ich in Stud.IP etwas mache? "Diese Aktion wird in Matrix erscheinen. Wie bekommen die Studierenden mit, dass es den Kanal gibt? Studierende sollten irgendwie informiert werden. Reiter: Messenger -> Infos + Kurzanleitung + evtl. Konfiguration Kann man deutlich machen, wie wäre es denn, wenn ich in Stud.IP sehen kann, wenn in Element etwas passiert. was heißt es eigentlich, wenn man zwei verschiedene Dienste hat? Was soll wo passieren? Hochladen einer Datei -> Dialog: erscheint auch im Matrix-Kanal, Schalter (Analyse Idee, schreiben, nicht umsetzen)? (Implementation: gibt Stud.IP)

### 3.3.6 Direktnachrichten

1-zu-1 Chats

- ist bereits in Matrix vorhanden, deshalb in der Umsetzung nicht vorgesehen - man könnte die Stud.IP Direktchats mit Matrix verbinden

**Sprechstundenfunktion** bei der man für eine kurze Zeit einen 1-zu-1 Chat mit einer Person führt und den Chat danach auch automatisch verlässt. Andere Personen sind in der Zeit in einer Warteschlange.

### 3.3.7 mehrere Räume pro Kurs

(keine Umsetzung, nur Konzept) Es sollten aus einer Stud.IP Veranstaltung mehrere Matrix-Räume erstellt werden können, die dann als Community zusammengefasst werden könnten. Mögliche Erweiterung (sinnvoll oder nicht?)

### 3.3.8 Kompatibilität von Nachrichten in Blubber und Matrix

Stud.IP (Blubber) sollte als regulärer Client agieren.

**Dateien versenden** Verlinken von Inhalten in Stud.IP Eventuell zu aufwändig.

**Emoticons korrekt anzeigen**

**Nachrichten sollten editier- und löschar sein**



### 3.3.9 Übernahme des Stud.IP Profilbildes

Personenbezogene Daten. Datenschutz. Sobald Matrixkopplung, dann auch Bild übertragen (im Reiter, s.o.), aktive Zustimmung verweist auf Profil->Einstellungen->Privatsphäre->Eigenes Bild Stud.IP: Wie darf mein Bild verwendet werden? "Darf auch extern angezeigt werden" gibt die API das überhaupt her, kann man auch weglassen, wenn es aus Zeitgründen nicht mehr klappt

### 3.3.10 Stautsindikator in Stud.IP

(umsetzen) Icon für Reiter in Stud.IP. objectsUserVisit Tabelle - Wie viele Nachrichten gab es in Matrix seit dem letzten Besuch (Anzeige durch Icon + Tooltip + Benachrichtigung auf dem Reiter) - SQL-Abfragen dauern evtl. lange, wenn das externe System nicht antwortet - evtl nicht individuell pro Student, sondern: Wie viele Nachrichten gab es HEUTE, aktualisieren mit CronJob - Diskussion, ob das sinnvoll ist

### 3.3.11 Opensource

### 3.3.12 Anbindung von Matrix durch ein Stud.IP-Plugin

Eine der nicht funktionalen Anforderungen stellt somit die Anbindung des Matrix-Chats an Stud.IP in Form eines Plugins dar. So kann jeder Standort für sich entscheiden, ob er diese Anbindung überhaupt installieren und anbieten möchte.

### 3.3.13 keine zusätzliche API

### 3.3.14 Sicherheit

### 3.3.15 Robustheit, Ausfallsicherheit

was passiert, wenn der Matrix Server für 3 Std. ausfällt? Werden Messages gepuffert? Implementation direkt über eine Que laufen lassen, DB mit Ergebnissen, die noch so Matrix geschickt werden müssen + Cronjob 1x Std., gibt es noch Ereignisse, die nicht abgearbeitet wurden?



# Kapitel 4

## Implementation

Hier die Anforderungskapitelüberschriften kopieren und Stück für Stück erörtern. Sinnhaftigkeit, Umsetzung, Mögliche Umsetzung oder keine Umsetzung diskutieren.

### 4.1 Übernahme von Veranstaltungsinformationen Veranstaltungen auf Räume abbilden + Übernahme der Teilnehmerlisten

Kurse, Studengruppen, ... sollten (automatisch) auf Räume in Matrix abgebildet werden.

Reiter: Chat mit konfigurationseinstellungen - nach Aktivierung nur für Doz und Tutoren sichtbar, die sollten die Erzeugung des Matrixraums anstoßen, nur, wenn eine Parametrisierung notwendig, sonst mit Pluginaktivierung automatisch machen - was passiert beim Löschen in Stud.IP und beim erneuten Anlegen? - wenn Raum erzeugt wurde, ist der Reiter auch für Studierende sichtbar mit Anleitung (Einfach halten, "Platzhalter"), + Knopf für Profilerstellung/Übertragung zu Matrix

### 4.2 Übernahme von Rollen, Rechten und Status

Keine automatische Übernahme der Teilnehmerliste, Datenschutz, einzeln Übernahme anstoßen pro Teilnehmer, Sichtbareinstellungen in Matrix nicht abbildbar. (mit Elmar sprechen!, Axel, Jens) manuelle Nutzerverwaltung in Matrix sollte nicht genutzt werden, zwischen Doz und Tutoren nicht unterscheiden. Userstatus spielt in der Praxis kaum eine Rolle. Synchen der User. User = autoren Anbindung von Fremdsystemen mit einem anderen Rollen-Rechte-Konzept. Feingranulare Berechtigungen für Studierende ist nur eingeschränkt umsetzbar, da es in Matrix nur 3 Rechtstufen gibt. Denkbar wäre grob: Dozenten -> Moderator/Admin und Studierende -> Default. In Matrix lassen sich zusätzlich benutzerdefinierte Rechtstufen erstellen, eventuell lassen sich damit alle in Stud.IP verfügbaren Rechtstufen abbilden.

Der eigene Status sollte leicht einzustellen sein oder aus Stud.IP übernommen werden (sichtbar/nicht sichtbar) Man könnte in Matrix auch manuelle Rollen anlegen Datenschutzaspekt: Ich möchte unsichtbar sein, in Matrix schwierig. Datenschutzrechtliche Begründung, warum steht der Name der Studenten in der Raumliste? beibehalten der Kommunikationskultur, keine anonymen Beiträge möglich in Matrix raumbezogene Einstellung: Teilnehmerliste ist sichtbar oder nicht in synchronisierten Räumen wir die Teilnehmerliste nicht angezeigt, kann Matrix das? Diskussion, was mache ich jetzt? 1. ich halte fest, es ist ein offenes Problem, Umsetzung trotz nicht datenschutzkonform 2. Leute, die sich unsichtbar geschaltet haben, werden nicht mitsynchronisiert, die bekommen evtl nichts mehr mit 3. Leute, die in min 1 VA unsichtbar sind -> mit Synonym sichtbar machen. - jeder Student muss von Stud.IP aus seine Zustimmung geben, das weitergeben des Nutzers an Matrix aktivieren (ist das sinnvoll?) gibt es Alternativen? ich sehe keinen guten Grund, das eine oder andere zu machen. -> Workshop mit Entwicklern und Didaktikern -> Ergebnisse -> Aus technischer Sicht gibt es keinen Grund, das eine oder das andere zu nehmen, Workshop hat ergeben: folgendes Ergebnis Beim Austragen oder Ändern der Rechtestufe in Matrix synchron zu halten. -> Cronjob einmal stündlich Lose Kopplung, nur einmal anlegen, was danach mit dem Raum passiert ist offen

### 4.3 kein zusätzliches Login

Nur ein Login für beide Systeme/Dienste. Single Sign-On Stud.IP Nutzer sollten ohne zusätzliches Login in einen Matrix Raum gelangen können.

### 4.4 Benachrichtigungsfunktion

**Synchronizität** Zum einen sollten Matrix und Blubber zeitlich synchron sein. Zum anderen wäre es wünschenswert, wenn die in einem System als gelesen markierte Nachrichten auch im anderen System als gelesen markiert würden.

Man möchte in Matrix über relevante Ereignisse informiert werden. (-> später, was ist eigentlich relevant, Tabelle: Relevanzabschätzung für Ereignisse, was ist relevant für alle? persönliche Präferenzen kann man mit dem hier entschiedenen Modell nicht berücksichtigen, deshalb sind folgende Konfiguration pro Veranstaltung sinnvoll? Welche Ereignisse sollen erscheinen, oder nicht? Diskussion, was spricht dafür, was spricht dagegen? Aus studentensicht evtl. verwirrend -> Argument gegen Konfig. Elmar oder Marcus -> Notifications, wenn z.B. eine Datei hochgeladen wird. NotificationCenter::addObserver Richtung Stud.IP -> Matrix, oder Matrix -> Stud.IP (nicht genannt, aber der Vollständigkeit halber dennoch genannt) braucht man eine Konfigurationsmöglichkeit dafür? wichtige Infos über die Kopplung hinweg. Informationsveranstaltung mit uniübergreifenden Infos, denkbar. (Eher ungeeignet, nach unten): unibezogene/veranstaltungsübergreifende Informationen, wie z.B. Rückmeldefristen, Unischließung, ... (dies wäre ein reiner Infochannel, in dem alle Studierenden sein müssten, keine Antwortmöglichkeit -> one-way Kommunikation -> eMail/Stud.IP Benachrichtigung wie bisher ist da

vielleicht besser) Besser: veranstaltungsbezogene Informationen, z.B. "eine Datei wurde hochgeladen", "Termin fällt aus"

## 4.5 Transparenz

(wird tendenziell nicht umgesetzt -> nur Konzept) Verknüpfung zwischen Stud.IP und Matrix/Element deutlich machen. Welche Konsequenzen hat das eigene Handeln in Matrix, wenn ich in Stud.IP etwas mache? "Diese Aktion wird in Matrix erscheinen. Wie bekommen die Studierenden mit, dass es den Kanal gibt? Studierende sollten irgendwie informiert werden. Reiter: Messenger -> Infos + Kurzanleitung + evtl. Konfiguration Kann man deutlich machen, wie wäre es denn, wenn ich in Stud.IP sehen kann, wenn in Element etwas passiert. was heißt es eigentlich, wenn man zwei verschiedene Dienste hat? Was soll wo passieren? Hochladen einer Datei -> Dialog: erscheint auch im Matrix-Kanal, Schalter (Analyse Idee, schreiben, nicht umsetzen)? (Implementation: gibt Stud.IP)

## 4.6 Direktnachrichten

1-zu-1 Chats

- ist bereits in Matrix vorhanden, deshalb in der Umsetzung nicht vorgesehen - man könnte die Stud.IP Direktchats mit Matrix verbinden

**Sprechstundenfunktion** bei der man für eine kurze Zeit einen 1-zu-1 Chat mit einer Person führt und den Chat danach auch automatisch verlässt. Andere Personen sind in der Zeit in einer Warteschlange.

## 4.7 mehrere Räume pro Kurs

(keine Umsetzung, nur Konzept) Es sollten aus einer Stud.IP Veranstaltung mehrere Matrix-Räume erstellt werden können, die dann als Community zusammengefasst werden könnten. Mögliche Erweiterung (sinnvoll oder nicht?)

## 4.8 Kompatibilität von Nachrichten in Blubber und Matrix

Stud.IP (Blubber) sollte als regulärer Client agieren.

**Dateien versenden** Verlinken von Inhalten in Stud.IP Eventuell zu aufwändig.

**Emoticons korrekt anzeigen**

**Nachrichten sollten editier- und löschar sein**

## 4.9 Übernahme des Stud.IP Profilbildes

Personenbezogene Daten. Datenschutz. Sobald Matrixkopplung, dann auch Bild übertragen (im Reiter, s.o.), aktive Zustimmung verweist auf Profil->Einstellungen->Privatsphäre->Eigenes Bild Stud.IP: Wie darf mein Bild verwendet werden? "Darf auch extern angezeigt werden" gibt die API das überhaupt her, kann man auch weglassen, wenn es aus Zeitgründen nicht mehr klappt

## 4.10 Stautsindikator in Stud.IP

(umsetzen) Icon für Reiter in Stud.IP. objectsUserVisit Tabelle - Wie viele Nachrichten gab es in Matrix seit dem letzten Besuch (Anzeige durch Icon + Tooltip + Benachrichtigung auf dem Reiter) - SQL-Abfragen dauern evtl. lange, wenn das externe System nicht antwortet - evtl nicht individuell pro Student, sondern: Wie viele Nachrichten gab es HEUTE, aktualisieren mit CronJob - Diskussion, ob das sinnvoll ist

## 4.11 Opensource

## 4.12 Anbindung von Matrix durch ein Stud.IP-Plugin

Wie bereits im Anforderungskapitel 3.3.12 angesprochen soll die Anbindung von Matrix an Stud.IP mit Hilfe eines Plugins erfolgen, was die Option des direkten Einbaus in den Kern-Programmcode von vorn herein ausschließt.

Für die Implementation sind somit grundsätzlich zwei Varianten denkbar. Die erste Variante wäre die Umsetzung in einem Plugin und die zweite Variante wären mehrere Plugins, die anschließend interagieren. Wie bereits in Abbildung 2.2 zu sehen war, existieren in Stud.IP verschiedene Typen von Plugins, wobei ein konkretes Plugin mehrere Typen haben kann. Der Typ **StandardPlugin** ermöglicht es zum Beispiel eigene Navigationspunkte in Veranstaltungen bzw. Einrichtungen anzuzeigen. Allerdings wird dieser Plugintyp auch nur im Veranstaltungs- und Einrichtungskontext geladen. Das heißt auf der Startseite oder im globalen Adminbereich steht es nicht zur Verfügung und es lassen sich dort auch keine Konfigurationen vornehmen. Deshalb kann es nötig sein, dass ein Plugin mehrere Typen besitzt.

Die verschiedenen Typen werden durch Interfaces repräsentiert und können durch die jeweilige Plugin-Klasse implementiert werden. Aufgrund dieser verschiedenen Plugintypen kann es sinnvoll sein mehrere statt nur ein einziges Plugin für eine neue Funktion zu implementieren.

Die Vorteile einer solchen eher modularisierteren Implementation durch mehrere Plugins sind die bessere Übersichtlichkeit, der Programmcode kann strukturierter aufgebaut werden und eine Wiederverwendung ist gegebenenfalls einfacher. - Beispiel: Schwarzes Brett (<https://gitlab.studip.de/uol/plugin/blob/master/plugin.manifest>) CODE: Pluginmanifest - mehrere Plugins können im Manifest

---

zusammengefasst werden - hier könnte man folgendes auslagern: Adminkonfiguration, NotificationCenter

## 4.13 keine zusätzliche API

Die Vorhandenen APIs von Stud.IP sowie Matrix können genutzt werden.

## 4.14 Sicherheit

## 4.15 Robustheit, Ausfallsicherheit

was passiert, wenn der Matrix Server für 3 Std. ausfaellt? Werden Messages gepuffert? Implementation direkt über eine Que laufen lassen, DB mit Ergebnissen, die noch so Matrix geschickt werden müssen + Cronjob 1x Std., gibt es noch Ereignisse, die nicht abgearbeitet wurden?





# Kapitel 5

## Evaluation

Da vermutlich keine Zeit für eine empirische Evaluation mit einem IRL Test + Umfrage bleibt, werden hier technische Tests hinreichen müssen.

### 5.1 Technische Tests

#### 5.1.1 Unit-Tests

Testbeschreibung

#### 5.1.2 Code Coverage

### 5.2 Chancen

### 5.3 Probleme



## Kapitel 6

# Ausblick

Potentielle Nutzung in der Zukunft und Weiterentwicklung?

Beispieltext. Siehe auch [1, 3, 4]. URLs gehen auch: [2].

# Literaturverzeichnis

- [1] BESL, P.; MCKAY, N.: A Method for Registration of 3-D Shapes. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14 (1992), Februar, Nr. 2, S. 239–256
- [2] S. ASSMANN: *You(r) Study - Eigensinnig Studieren im 'digitalen Zeitalter'*. – <https://your-study.info/>
- [3] SCHULMEISTER, R.: *Lernplattformen für das virtuelle Lernen: Evaluation und Didaktik*. München : Oldenbourg Wissenschaftsverlag, 2005
- [4] THRUN, S.; FOX, D.; BURGARD, W.: A Real-Time Algorithm for Mobile Robot Mapping With Applications to Multi-Robot and 3D Mapping. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000



# Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Osnabrück, August 2021