Diplomarbeit

Höhere Technische Bundeslehranstalt Leonding
Abteilung für Informatik

HomeDS

Eingereicht von:	**Andrej Sakal, 5CHIF**
	Felix Hofmann, 5CHIF
Datum:	**April 4, 2018**
Betreuer:	**Thomas Stütz**

Declaration of Academic Honesty

Hereby, I declare that I have composed the presented paper independently on my own and without any other resources than the ones indicated. All thoughts taken directly or indirectly from external sources are properly denoted as such.

This paper has neither been previously submitted to another authority nor has it been published yet.

Leonding, April 4, 2018

Andrej Sakal, Felix Hofmann

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorgelegte Diplomarbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Gedanken, die aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.

Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Leonding, am 4. April 2018

Andrej Sakal, Felix Hofmann

Zusammenfassung

Die HTL-Leonding besitzt schon einige Multimedia Systeme verstreut im ganzen Schulgebäude um Projekte, aktuelle News und Änderungen im Unterrichtsablauf anzuzeigen. Doch ein großer Schwachpunkt dieser Multimedia Systeme ist, dass der Prozess vom Erstellen der Anzeige bis zum Zuordnen, welcher Bildschirm welche Information anzeigen soll sehr kompliziert und mühselig ist. So wird oftmals neue Information erst verspätet oder gar nicht angezeigt.

Unsere Diplomarbeit beschäftigt sich mit dem Erschaffen eines gemeinsamen Systems, um einfach neue Supplierungen, Nachrichten oder Eilmeldungen auf allen Bildschirmen der HTL-Leonding anzuzeigen. Diese Systeme werden unter dem Begriff "Digital Signage Systemßusammengefasst.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Ausgangssituation

Die HTL-Leonding besitzt schon einige Multimedia Systeme verstreut im ganzen Schulgebäude um Projekte, aktuelle News und Änderungen im Unterrichtsablauf anzuzeigen. Doch ein großer Schwachpunkt dieser Multimedia Systeme ist, dass der Prozess vom Erstellen der Anzeige bis zum Zuordnen welcher Bildschirm welche Information anzeigen soll, sehr kompliziert und mühselig ist. So werden neue Informationen erst verspätet oder gar nicht angezeigt.

1.2 Ziele

Ziel ist es, dass die Schulverwaltung möglichst schnell überall in der Schule Informationen, Warnungen oder Ankündigungen anzeigen kann. Die verschiedenen Multimediasysteme sollen einheitlich gesteuert und verwaltet werden können, um schnell alle Anzeigen beliebig zu verändern. So ist es auch ein Teilziel festzustellen, ob es möglich ist die derzeitig verwendeten Anzeigesysteme durch den XIBO Server zu ersetzen.

1.3 Problemstellung

Momentan wird um eine Anzeige zu ändern sehr viel Aufwand betrieben. Zum Beispiel wird eine neue Präsentation in Form von Folien oder Video zusammen-

geschnitten. Beispiel dafür ist die Anzeige im Eingangsbereich der Schule. Diese Vorgehensweise ist zeitaufwendig und werden Änderungen vorgenommen, kann die alte Präsentation oder das Video meistens verworfen werden.

Kapitel 2

XIBO-Grundlagen

2.1 Digital Signage

Digital Signage Systeme haben die Aufgabe viele Bildschirme mit Inhalten zu füllen und eventuell auch diese Inhalte zu designen. Damit soll das zeit- oder interaktionsgesteuerte Ändern von Inhalten auf den Bildschirmen einfach und übersichtlich gehalten werden. Weiteres bietet Digital Signage ein breites Spektrum an Anwendungsbereichen. Digital Signage ist vor allem im Marketing Bereich ein sehr beliebtes Mittel um ein neues Produkt oder eine Neuheit zu präsentieren. https://de.wikipedia.org/wiki/Digital$_{S}ignageAnwendungsbeispiele$:$_{2}$ 017.

2.2 Was ist XIBO?

Das XIBO ist ein Open Source Digital Signage System entwickelt von der Spring Signage LTD. Das XIBO-System besteht aus vielen verschiedenen Komponenten. Das XIBO Paket besteht aus einem klassischen Server-Client Konstrukt. Der Server besteht aus drei Komponenten: das Content Managment System welches mithilfe von ZeroMq bei Änderung der Inhalte diese aktualisieren soll, einer Datenbank und einer Weboberfläche, die es dem Benutzer ermöglichen soll das System zu bedienen.

SYSTEM ARCH PLAN eventuell noch über zeromq schreiben

2.3 XIBO - Login

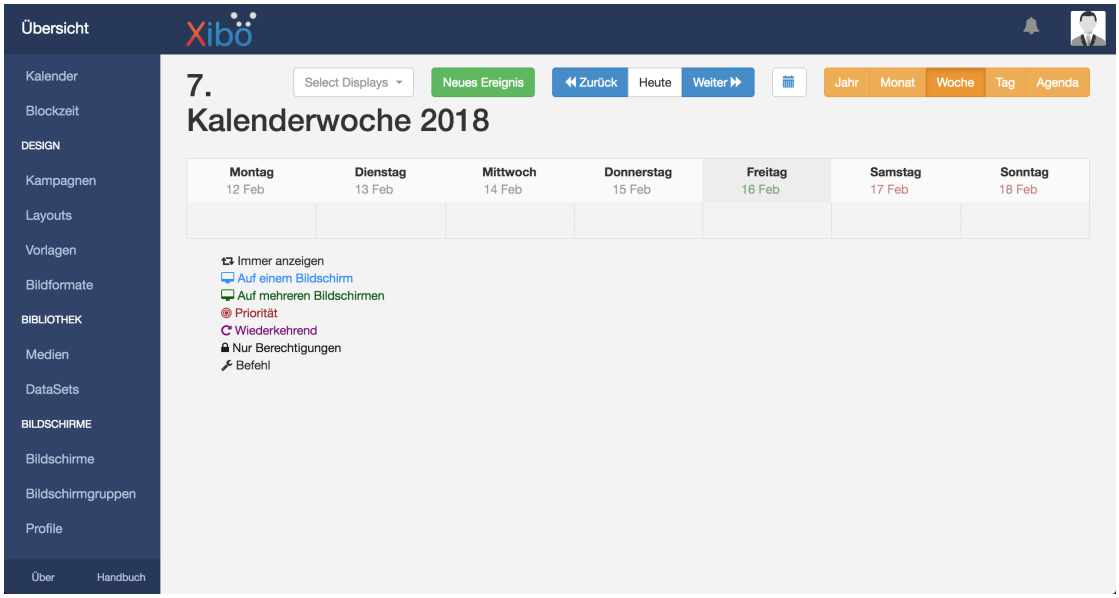
Der verwendete XIBO Server ist auf einem eigenen Dedicated Server installiert. Und nur direkt erreichbar wenn über ein internes Netzwerk der HTL-Leonding zugegriffen wird. Um von außerhalb sich auf den XIBO verbinden zu können muss man sich in das interne Netztwerk tunneln.

ERLÄREN TUNNELING

2.4 Weboberfläche des XIBO

Das Steuerungszentrum des ganzen Signage System ist die Weboberfläche, die ganz einfach über einen Browser unter der Serveraddresse aufgerufen werden kann. Auf der Willkommensseite sind die wichtigsten Funktionen dargestellt:

1. *Kalender:* Mit der Kalender Funktion kann eingetragen werden zu welchem Zeitpunkt, welcher Inhalt, auf welchem Bildschirm angezeigt werden soll. In dem Xibo-Kalender werden auch bereits eingetragene Aktivitäten angezeigt.



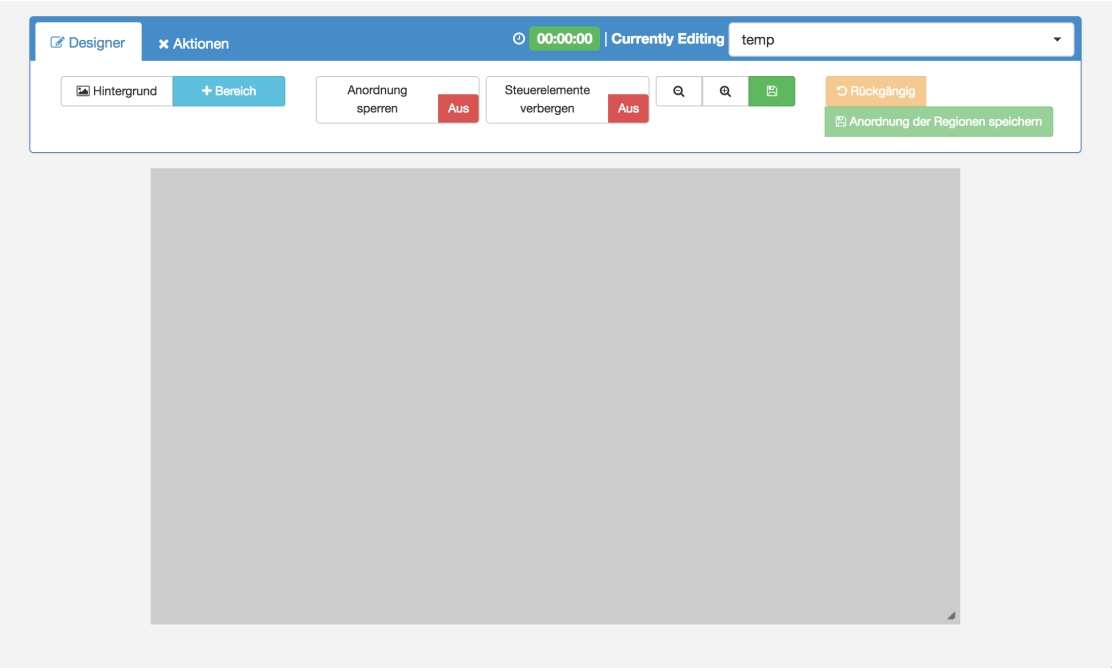
2. *Layouts:* Die Layout-Funktion ist einer der wichtigsten Komponenten des Signage Systems. Es beschäftigt sich mit dem Designen der Inhalte. Auf diese Funktion kommen wir noch einmal zurück

3. *Bibliothek:* Die Bibliothek-Funktion ist zuständig für das Verwalten der Medien. Hier können Sie verschiedene Dateien hochladen. Diese Medien können dann in Layouts eingebunden und angezeigt werden.
4. *Benutzer:* Im Menüpunkt Benutzer können neue Benutzer angelegt werden und bereits bestehende bearbeitet oder gelöscht werden. Dabei gibt es auch ein Rechte-System. Es können auch Datenmengenbegrenzungen pro Benutzer eingestellt werden.
5. *Einstellungen:* Der Menüpunkt Einstellungen gibt dem Nutzer die Möglichkeit, verschiedene Optionen zu wählen. So sind zum Beispiel die richtige Zeitzone, E-Mail Benachrichtigungen, wichtige Einstellungen, die für ein einwandfreies Funktionieren des Xibo-Servers zuständig zuständig sind.

2.5 Designen mit XIBO

Beim Designen von einem neuen Layout im XIBO muss zuerst die Bildschirmauflösung ausgewählt und dem Layout ein passender Name zugewiesen werden, sowie optional auch eine Beschreibung.

Layout Maske



Dem Layout kann nun eine Region oder auch mehrere hinzugefügt werden. Eine Region kann widerrum mehrere Widgets enthalten. Mit einem Doppelklick auf die Region kann ein Widget hinzugefügt werden. Es gibt viele verschiedene Arten von Widgets:

Bibliothek: Mit diesem Widget können Dateien aus der Medienbibliothek in der Region angezeigt werden.

Uhr: Dieser Widgettyp bindet eine Uhr in die Region ein. Es kann entweder eine Uhr im Analog Stil oder Digitalem Stil ausgewählt werden.

Bibliothek: Die Bibliothek Funktion ist zuständig für das Verwalten der Medien. Hier können Sie verschiedene Dateien hochladen. Diese Medien können dann in Layouts eingebunden und angezeigt werden.

Benutzer: Im Menüpunkt Benutzer können neue Benutzer angelegt und bereits bestehende bearbeitet oder gelöscht werden. Dabei gibt es auch ein Rechte-System. Es können auch Datenmengenbegrenzungen pro Benutzer eingestellt werden.

Einstellungen: Der Menüpunkt Einstellung gibt dem Nutzer die Möglichkeit verschiedene Optionen zu wählen. So sind zum Beispiel die richtige Zeitzone, E-Mail Benachrichtigungen wichtige Einstellungen die für ein Einwandfreies funktionieren des Xibo-Servers zuständig sind.

Kapitel 3

XIBO-Server

3.1 Beschreibung

Als zentrale Steuereinheit wird ein XIBO-Server verwendet. Um diesen verwenden zu können, war es notwendig sich in die Dokumentation einzulesen und die API-Schnittstelle auszuprobieren. Die Website des Servers diente vorerst als Übungsumgebung. Dadurch wurde es leicht auch die einzelnen Funktionen, inklusive der Vorgangsweise, des Servers zu verstehen. [1]

3.2 API-Schnittstelle

Die API-Schnittstelle des XIBO-Servers ist mittels Swagger dokumentiert. Diese Dokumentation deckt die Grundfunktionalitäten und die Form der Anfragen ab. Da die Schnittstelle des Servers später als wesentliches Verbindungsstück zwischen der eigens entwickelten Steuerungssoftware und dem Server dient, war es nötig, diese gründlich zu testen und auch zu verstehen. Anfangs wurde dafür mit Postman gearbeitet. Um mit Postman die Requests testen zu können musste festgestellt werden, welche Codierung für den Request verwendet wird. Im Falle des XIBO-Servers wird "application/x-www-form-urlencoded" als Codierung verwendet. Die Anfragen an den Server wurden im Java Code durch die "libary" OkHttp3 übernommen. [2] [3] [?]

3.3 Authentifizierung

Es stellte sich heraus, dass die Authentifizierung mittels OAuth2 sehr speziell war, was zu Beginn zu einigen Schwierigkeiten führte. Es benötigte einige Anläufe um herauszufinden, wie und in welcher Reihenfolge die Parameter übergeben werden müssen. Dazu wurde eine Java-Klasse entwickelt, welche die Authentifizierung automatisch übernimmt. [4]

Der Server benötigt zur Authentifizierung mit einem Client eine Client_ID. Diese wird vom Server für jeden Client eindeutig erzeugt. Man bekommt sie direkt von der Website des Servers. Weiteres wird ein Client_Secret benötigt, das ebenso wie die Client_ID vom Server für jede Anwendung ,eindeutig erzeugt wird und auch auf der Website erhältlich ist. Zudem ist ein Parameter in der Form "grant_type=client_credentials" mitzugeben.

Zuerst wird ein Request-Body erstellt. Dieser hat folgende Parameter in der Form: "client_id=<CLIENT_ID>&client_secret=<CLIENT_SECRET>&grant_type=client_credentials" , die im Body mitgegeben werden und als Format 'application/x-www-form-urlencoded' haben. Anschließend werden dem Header noch der content-type" mit dem Wert "application/x-www-form-urlencoded" und der Parameter "cache-control" mit dem Wert "no-cache" hinzugefügt. Als Ergebniss der Anfrage bekommt der Client einen " access_token ", dieser ist nun bei jeder Anfrage notwendig um sich beim Server zu authentifizieren und es dem Client zu ermöglichen Daten abzurufen beziehungsweise weiterzugeben.

Kapitel 4

Verwendente Technologien

4.1 Git und GitHub

Um dynamisch als Team arbeiten zu können, verwenden wir Software zur Versionsverwaltung. Hierbei handelt es sich um Git. Github ist die verwendete Online-Plattform, auf der Benutzer ihre Projekte gratis als Repository speichern. Dies ermöglicht einfaches arbeiten im Team und verhindert in den meisten Fällen Zusammenführungskonflikte. Mittels Git lässt sich auch leicht zurückverfolgen welches, Teammitglied welche Änderungen gemacht hat und im Notfall ist es auch möglich diese Änderungen wieder rückgängig zu machen.

Verwendet wird GitHub für die gesamte Diplomarbeit, sowohl für die Versionierung der Dokumente, als auch die einzelnen Applicationen. Um sicherzustellen, das keine Konflikte durch paralleles arbeiten entstehen, wird in Branches gearbeitet. Diese Branches wurden erstellt, wenn ein neues Arbeitspaket begonnen wurde, zum Beispiel die Android-App.

Bild Github und verweis Git/GitHub

4.2 Android

Android ist ein Betriebssystem für mobile Endgeräte, spezialisiert für Touch-Anwendungen. Ziel ist es das Endgerät möglichst intuitiv und flexibel bedienen zu können. Mit Android ist es möglich open-source Applicationen zu erstellen die ein großes Publikum erreichen. Google stellt auch einen Markt zur Verfügung in dem die Applicationen gratis oder auch gegen Entgelt erworben werden können.

Diese Aspekte: open-source, gratis und großes Publikum, waren ausschlaggebend dafür, dass die Applicationen in Android implementiert wurden. Als Programmiersprache wurde JAVA verwendet.

4.3 Java Enterprise Edition

Java Platform, Enterprise Edition oder abgekürzt auch Java EE ist die technische nähere Beschreibung einer Softwarearchitektur, die programmierte Java Anwendungen ausführt. (weiter ausführen)

QUELLE: https://de.Wikipedia.org/wiki/Java$_{P}latform,_{E}nterprise_{E}dition$

4.4 JSF - Java Server Faces

Kapitel 5

HomeDS - Server

5.1 Einleitung

Im Rahmen der Diplomarbeit wird neben dem XIBO-Server ein weiterer Java Enterprise Edition Server eingesetzt. Aufgrund der hohen Komplexität des Signage-Servers jedoch, relativ dünn dokumentierten API-Schnittstellen und begrenzten technischen Möglichkeiten, muss ein eigens programmierter Server eingesetzt werden. Dieser wird die Kommunikation vom Signage Server zur Android App erleichtern.

5.2 Anforderungen an den HomeDS Server

Der JavaEE Server soll die verschiedenen komplizierten Abläufe des XIBO-Servers vereinfachen. Die verschiedenen Zugriffe mittels REST die eigentlich direkt auf den XIBO laufen sollen, werden über den JavaEE Server verwaltet. Dies hat insofern Vorteile, da die komplizierten und meist mit viel Aufwand verknüpften Authentifizierungen wegfallen. Somit können ohne Probleme neue Funktionen hinzugefügt und ohne Probleme an neue Anforderungen angepasst werden.

5.3 Struktur des Projekt's

5.4 Funktionen des HomeDS Server

Der JavaEE Server verfügt über eine eigene MySQL Datenbank. Diese wird gebraucht, um die "DataSets" aus dem Signage-Server zwischenzuspeichern. Dies ist insofern erforderlich, da die Aufgabenstellung erfordert die Datensätze entweder nur ab einem bestimmten Datum im Layout anzuzeigen oder die Datensätze bis zu einem bestimmten Datum angezeigt werden dürfen.

5.5 DataSet mit Ablaufdatum

Einer der wichtigsten Funktionen des Servers ist das Hinzufügen, Ändern und Löschen vom DataSet. Da der Digital Signage Server über keine Felder wie Start- und Enddatum verfügen.

Die Logik ist simple. Der Benutzer kann Start- und Enddatum für jeden einzelnen DataSet eingeben. Erst wenn das Datum genau in diesem Zeitintervall inklusive den Grenzen liegt, wird das DataSet an den Xibo mittels API-Schnittstelle weitergegeben.

Listing 5.1: public void doCheckEvery24Hours()

```
if (dataset.isActive() == false &&
   (dataset.getFromDate().minusDays(1)
         .isBefore(LocalDate.now()))) {
    try {
         //if succesfull added then set active true and
            add id
         if ((id=dataSetApi.addDataSetField(dataset)) >
            0) {
             dataset.setDataRowId(id);
             dataset.setActive(true);
             dataSetFieldFacade.save(dataset);
         }
    } catch (NoConnectionException e) {
         // Catch Exception
    }
}
```

Die Überprüfung ob die DataSets aus der Server-Datenbank im Zeitintervall liegen, wird mittels eines "TimeSchedule" jeden Tag um 01:00 Uhr durchgeführt. Falls dieses DataSet im Intervall liegt, wird dieses DataSet an den XIBO Signage Server gesendet. Diese Überprüfung beinhaltet auch das FromDate. Dies löscht bei überschreiten des Bis-Datums das DataSet aus dem Digital Signage heraus.

Die Überprüfung ob das Startdatum heute oder vor dem heutigem Datum liegt, wird mithilfe der Java Annotation @Schedule realisiert. (siehe Codeausschnitt)

5.6 Jave Enterprise Edition mit Android über REST

Ein weiterer essentieller Teil des Server auf Java Enterprise Edition Basis ist die Kommunikation mittels REST. Alle Funktionen des Servers werden auch wieder mit REST für unsere Android Applikation zur Verfügung gestellt.

5.7 Swagger

Swagger wird verwendet um Funktionalität und Möglichkeit einer API übersichtlich zu gestalten. Um die REST Schnittstellen zu dokumentieren und übersichtlich zu visualisieren wird Swagger verwendet. Dabei gibt es zwei verschiedene Arten eine REST-Dokumentation zu erstellen.

Die erste wäre, mit dem Swagger Editor die Dokumentation in der JSON-Ausdruckssprache mit der Hand zu schreiben und immer wieder zu aktualisieren. Natürlich ist dies bei vielen verschiedenen GETs, POSTs, PUTs und DELETEs sehr aufwendig und mühsam.

Bei der zweiten Methode, die auch bei der Diplomarbeit zum Einsatz kommt, werden die API's automatisch von der im Projekt eingebundenen Swagger Engine erkannt. Daraus wird dann auch wieder eine JSON-File generiert, die dann mithilfe von Swagger-UI gut im Browser über eine Webseiten URL erreichbar ist.

Verweis: https://swagger.io/

5.8 Java Server Faces mit JavaEE

Um den Benutzern die Möglichkeit zu geben die Funktionen unseres Servers zu nutzen wurde im Rahmen der Diplomarbeit auch eine Webapplikation erstellt.

Und da JSF also Java Server Faces mit einem Java EE Server harmoniert haben wir uns entschieden eine Webapplikation zu erstellen. Dies hat viele Vorteile wie z. Bsp. die Platformunabhängigkeit und auch die schnellere Entwicklung im Vergleich zu anderen Clients.

Mit JSF wird nämlich über eine Managed Bean direkt im XHTML auf die Funktionen des Servers zugegriffen somit sind keine REST Zugriffe nötig.

Unsere Weboberfläche ist Responsive gestaltet und mithilfe von BootsFaces realisiert worden.Boots Framework stellt fertige Komponenten zur Verfügung wie Buttons, Listen, Forms etc.

BILD VON OBERFLÄCHE

Die Aufgabe der DataSet Webapplikation ist DataSet zu ändern, hinzufügen oder zu löschen. Die Applikation besteht grob gesagt aus 2 Teilen.

Teil 1 kümmert sich um das Anzeigen aller DataSets in einer Responisve Komponente die "DataTable“gennant wird in der Bibliothek vom Bootsfaces Framework. Im DataTable ist es möglich die Anzahl der angezeigten Elemente pro Seite zu begrenzen oder erweitern, die einzelnen Spalten sortieren dabei ***ascending*** oder ***descending*** und durch die verschiedenen Spalten einer Zeile eine Suche durchzuführen.

Es ist auch durch die editierbaren Textfelder und DatePicker möglich die DataSets zu ändern. Durch klicken auf den Speichern Button wird die Änderung des einzelnen DataSets bestätigt. Durch klicken auf den Löschen Button Verweis auf JSF Section,

5.9 HomeDS Server Projekt-Struktur

5.10 HomeDS Server Structure Crawler

Für das verstehen des Aufbaus eines Layouts war es die Aufgabe einen sogennanten StructureCrawler zu erstellen. Dieser soll den JSON Aufbau eines Layouts ausgeben. Durch diese Funktion ist es für uns möglich gewesen das ganze Signage System zu verstehen und zu verwenden.

WEBOBERFLÄCHE VOM CRAWLER

Kapitel 6

Android Application

6.1 Einleitung

Um eine höchst mögliche Reichweite an Endgeräten zu erzielen, wurde eine Android Applikation entwickelt, mit der die wichtigsten Funktionen abgedeckt werden. Zum Beispiel das Wechseln der DataSets des Digital Signage Servers. Wichtig war es die Applikation möglichst einfach und leicht bedienbar zu gestalten, um auch Erstbenutzern die Bedienung zu erleichtern. Prototyp für die aktuelle Applikation war eine Anwendung für Android, die direkt mit dem Digital Signage Server kommuniziert, da sehr schnell klar wurde, dass das Steuern des Servers direkt über eine Applikation auf Android Basis viel zu umständlich ist. Es wurde eine weitere Mobile Applikation entwickelt, welche über das "HomeDsBackend"(Java-EE-Server) kommuniziert, um das Funktionsspektrum zu erweitern und die Applikation möglichst kompakt zu gestalten. Beispielsweise ist es durch diese Aufteilung nicht mehr nötig eine Datenbank in der Applikation zu haben. Somit erleichtert es auch Applikationen für andere Betriebssysteme zu implementieren.

6.2 Anforderungen

Die Anforderungen an die Android Applikation weichen von den Anforderungen an das "Backend" leicht ab, da das "Backend" die gesamte Zeitsteuerung- und Datenbankfunktionalität übernimmt. Es besteht die Möglichkeit, den Server über die Website des "Backend" zu steuern, beziehungsweise über die Applikation auf Android Basis.

- *Eilmeldungen:* Dem Benutzer soll es möglich sein Nachrichten in einem Ticker auf den Bildschirmen anzuzeigen.
- *Medien Wiedergabe:* Medien die Lokal am Digital Signage Server liegen sollen abgespielt werden können.
- *Authentifizierung automatisieren (Prototyp Applikation):* Die Authentifizierung am Digital Signage Server soll automatisiert werden, um nicht vom Benutzer durchgeführt werden zu müssen.

6.3 Verwendete Technologien

6.3.1 Android

Android wird mit der API(Englisch: application programming interface / Deutsch: Anwendungsprogrammierschnittstelle), in der Version 26 (Oreo / Android 8.0) verwendet. VERWEIS ANDROID !!!!

6.3.2 OkHttp3

Als Erweiterung für die Http Anfragen an den Digital Signage Server und das HomeDsBackend wird OkHttp3 verwendet.

6.4 Struktur

- *activity:* Alle Activities die für die Anwendung benötigt werden. Als Beispiel die MainActivity
- *adapter:* Hier befinden sich alle Adapter für die RecyclerViews.
- *apiClient:* Beinhaltet die Klasse "RequestHelper" mit der die Anfragen an den Digital Signage Server beziehungsweise an das HomeDsBackend vereinfacht werden.
- *entity:* Jene Klassen die als Models für die Anwendung benötigt werden. STÜTZ!!!
- *enumeration:* Enumerationen welche die Android Basierte Applikation verwendet. Zum Beispiel das "RequestTypeEnum".

- *fragment:* Alle Fragments die für das User Interface benötigt werden. Beispiel hierfür ist das Fragment "NewsOverviewFragment", welches alle DataSets die am Server sind anzeigt.
- *viewholder:* Beinhaltet alle "ViewHolder" die für die Verschiedenen "RecyclerViews" benötigt werden.

6.5 Benutzerhandbuch

6.5.1 Abspielen von Medien auf gewünschten Bildschirmen

Das Abspielen von Medien wird im Navigationstab "Play Media" abgewickelt, des weiteren befindet sich auf der Startseite ein Button über den direkt zu dieser Sektion navigiert werden kann. Startseite mit markierten Navigationselementen

6.6 MainBottomNavigationActivity und OverveiwFragment

6.6.1 MainBottomNavigationActivity

Die "MainBottomNavigationActivity" ist die Einzige "Activity" die in der Android Applikation benötigt wird. Hauptaufgabe der "MainBottomNavigationActivity" ist es, zwischen den einzelnen "Fragments" zu navigieren. Enthalten sind dazu jene Methoden, welche mit dem "SupportFragmentManager" die einzelnen Fragments im "container_main" austauschen. Der "container_main" ist ein "ConstraintLayout" welches in der zur "MainActivity" gehörenden Layout Ressource "activity_main.xml" mit der Identifikationsnummer "container_main" und belegt wurde. Zudem wird die Klasse durch das Interface "AppCompatActivity" erweitert und implementiert die "OnFragmentInteractionListener" der Fragmente die in der Applikation verwendet werden. Im unteren Teil der "Activity" befindet sich eine Navigationsleiste. Diese ist zuständig für das wechseln zwischen den einzelnen Fragmenten mit folgenden Navigationspunkten:

- *Home:* Jener Menüpunkt der die Startseite der Applikation beinhaltet.
- *DataSet:* Ist zuständig für die Verwaltung der "DataSets".

- *Medium abspielen:* Beinhaltet die Fragmente die benötigt werden um Medien auf der gewünschten anzeige abzuspielen.
- *Strukturplan:* Bietet eine Übersicht über die Struktur der am Server liegenden Layouts.

onCreate

Hier wird die "MainBottomNavigationActivity" als "ContentView" gesetzt, zudem wird mittels "SupportFragmentManager" das "HomeScreenFragment" dem "container_main" zugewiesen und angezeigt. Die statische Variable "instance", vom Datentyp "MainActivity", wird auf die aktuelle Instanz der Klasse zugewiesen.

getInstance

Ist der "Getter" für die statische Variable "instance" und übergibt die aktuelle Instanz der Klasse.

Fragment-austausch-Methoden

Sind jene Methoden die für das Austauschen der einzelnen "Fragments" im "container_main" zuständig sind. Dies geschieht mittels "SupportFragmentManager" welcher immer die "Fragments" im "container_main" anzeigt und dem "BackStack", welcher für die Rückwerts Navigation (mittels retour Knopf des Mobilen Endgerätes) in der Applikation zuständig ist, hinzufügt. Manche Methoden übergeben zudem noch ein "Bundle" an das erstellte "Fragment", welches Objekte enthält die in nächsten "Fragment" benötigt werden. Erkennungsmerkmal dieser Methoden ist das englische Verb "open" am beginn des Methodennamens. Als Namensbeispiel hierfür wird die Methode "openNewsEditFragment" herangezogen.

6.6.2 HomeScreenFragment

Der Einstiegspunkt der Applikation ist das "HomeScreenFragment". Es zeigt den Serverstatus an, dieser wird über einen "REST-Request" vom Java-EE-Server abgefragt. Den Hauptanteil des Fragments bilden die beiden Navigations-Buttons "Median abspielen" und "Eilmeldungen anzeigen". Durch klicken dieser Buttons öffnen sich die dazugehörigen Fragmente "NewsOverviewFragment" und "MediaOverviewFragment".

Die Layout Ressource des "HomeScreenFragment" enthält zwei Buttons und ein "ConstaintLayout". Dieses "ConstraintLayout" beinhaltet wiederum eine "TextView" und zwei "ImageViews". In der "onCreate" Methode des Fragments werden als erstes die Anzeigeelemente neu deklarierten Variablen zugewiesen und im "ConstraintLayout" wird der Serverstatus auf View standardmäßig als Offline angezeigt und die Buttons deaktiviert. Im folgenden Schritt wird über einen "GET-Request" die Uhrzeit des XIBO-Servers über den Java-EE-Server abgefragt. Wird eine Uhrzeit vom erhalten, so wird im "ConstraintLayout" des Server Status auf online gesetzt und das dazugehörige Bild angezeigt und die Buttons werden freigegeben. Erhält man keine Antwort bleibt die Anzeige unverändert.

6.7 DataSet Verwaltung

Die beiden Fragmente "NewsOverviewFragment" und "NewsEditFragment" sind für die Verwaltung der "DataSets" zuständig. Im Fragment "NewsOverviewFragment" wird eine Übersicht über alle vorhandenen "DataSets" gegeben. Das "NewsEditFragment" Fragment wird verwendet um vorhandene "DataSets" zu bearbeiten oder neue "DataSets" zu erstellen.

NewsOverviewFragment

Dieses Fragment zeigt alle aktiven "DataSets" an, diese werden vom Server bereitgestellt und per "REST-Request" abgefragt. Über einen "FloatingActionButton" kann ein neues "DataSet" erstellt werden. Durch klicken auf eines der Listen Elemente, öffnet sich eine Detailansicht in der das Bearbeiten, eines des "DataSets" möglich ist.
Die Layout Ressource des Fragments, "fragment_news_overview.xml" enthält eine "RecyclerView" und einen "FloatingActionButton". Die Logik der Anzeige ist in der Java Klasse "NewsOverviewFragment" implementiert. Die meisten Methoden dieser Klasse werden generisch beim Erstellen eines Fragments erzeugt. Die einzige Methode die überschrieben wurde ist die Methode "onCreateView". Beim Ausführen dieser Methode wird zuerst eine "View" erstellt welche das "fragment_news_overview" Layout zugewiesen bekommt. Anschließend wird eine "RecyclerView" und ein "FlaootingActionButton" erstellt und den zugehörigen Anzeige Elementen zugewiesen. Der "FlaootingActionButton" erhält einen "OnClickListener" über diesen wird in der "MainActivity" eine Methode aufgerufen die ein neues "NewsEditFragment" anzeigt, um ein neues "DataSet" zu erstellen.

NewsEditFragment

Das Fragment zeigt alle Details eines "DataSets" an und bietet die Möglichkeit die Detailinformationen bearbeiten zu können. Ebenso wird dieses Fragment dazu verwendet, ein neues "DataSet" zu erstellen, dazu wird das Fragment mit Hinweisen auf die Eingabeoptionen erstellt. Über einen Button können die eingegeben Felder an den "Java-EE-Server" übermittelt werden. Eingabe Felder:

- *Titel:* Das Feld "Titel" beinhaltet die Überschrift der anzuzeigenden Information.
- *Beschreibung:* Hier wird der eigentliche Informationstext eingefügt.
- *Datum-Von-Bis:* Diese Felder werden über ein "DatePickerDialog" befüllt, welcher einen Kalender öffnet und zeigen an in welchen Zeitraum das "DataSet" angezeigt wird.
- *Uhrzeit-Von-Bis:* Die anzeige Elemente geben Auskunft über die Zeitspanne in der das "DataSet" an den einzelnen Tagen angezeigt wird. Das Feld kann mittels "TimePickerDialog" befüllt werden.

Anzeige Ressourcen für dieses Fragment werden in der Datei "fragment_news_edit.xml" bereitgestellt. Diese enthält die benötigten Tags, um die im oberen Teil beschrieben Eingabefelder zur Verfügung zu stellen. Die Verwendung und Belegung dieser Felder wird in der Klasse "NewsEditFragment" implementiert. Auch in dieser Klasse wurde nur die Methode "onCreateView" mit Quellcode versehen. Zu Beginn wird der deklarierten "View" die "fragment_news_edit.xml" als Ressource zugewiesen. Um Daten aus dem "NewsOverviewFragment" zu empfangen wird ein "Bundle" befüllt. Wenn das befüllte "Bundle" Informationen beinhaltet, dann wird die Funktion "setArguments" der Basisklasse Fragment(android.support.v4.app.Fragment) mit dem Parameter "bundle" aufgerufen, um das Feld "mArguments" der Basisklasse Fragment mit Daten zu versehen(Vererbung). Die zuvor deklarierten Variablen (benötigte "Views") werden jetzt initialisiert. Das Fragment hat zwei verschiedene Vorgehensweisen. Zum einen werden, sollte ein "DataSet" in Form eines "Bundels" übergeben werden, die Anzeigeelemente mit den übergebenen Werten befüllt. Andernfalls werden die Felder mit keinen Werten versehen, es werdend Hinweise auf die Eingabeoptionen im aktuellen Fragment angezeigt. Die benötigten "onClickListener" für die Datums und Uhrzeit eingaben werden im Anschluss implementiert und mittels "DatePickerDialog" beziehungsweise "TimePickerDialog" mit Daten versehen. Durch Drücken des Buttons(Speicher Button) wird ein Event ausgelöst. Dieses Event kann zwei verschiedene Ausgänge haben: Wenn das im Event erhaltene "Bundle" eine "ID" besitzt, lässt sich daraus eindeutig schließen, ob es sich hierbei um ein Dataset handelt, das vom User erstellt wurde, oder um eines, das

bereits zuvor am Java-EE-Server vorhanden war, indem man überpfrüft, ob die im Bundle enthaltene Information "ID" den Wert null hat, oder nicht. Falls dies nämlich der Fall ist, sind es eindeutig vom Benutzer eingegebene Daten. Somit muss deswegen am Server anschließend ein "POST-Request" durchgeführt werden, um die neu erstellten Informationen zu senden. Wenn es ein vom Java-EE-Server erhaltenes "DataSet" ist, wird stattdessen ein "PUT-Request" durchgeführt, der die veränderten Daten übermittelt.

6.8 Mediaplayer

Um auf einer gewünschten Anzeige ein, sich auf dem Server befindliches, Medium abzuspielen wurden die beiden Fragments "MediaOverViewFragment" und "ChooseDisplayFragment" implementiert. Das gewünschte Medium wird auf dem ausgewählten Display sofort Abgespielt. Ist noch kein Display ausgewählt wird man zuerst auf das "ChooseDisplayFragment"gelitet um einen anzeige Display zu wählen.

MediaOverviewFragment

Im diesem Fragment werden alle Medien angezeigt, die sich am XIBO-Server in der Bibliothek befinden und mit dem Richtigen Tag markiert sind. Eine Sortierung der Liste ist über ein "Spinner" Element möglich. Des weiteren wird im rechten oberen teil des Fragments der Display angezeigt auf dem das gewählte Medium abgespielt werden soll. Eine Auswahl des Displays ist über Navigation zum "ChooseDisplayFragment" möglich, dieses wird über den Button "Auswählen" geöffnet. Hat der Benutzer noch keinen keinen Display ausgewählt wird er zuerst auf das "ChooseDisplayFragment" weitergeleitet.
Durch die Ressource Datei "fragment_media_overview.xml" können die Elemente in der View angezeigt werden. Dieses Layout beinhaltet neben einer "RecyclerView" zur Darstellung der abzuspielenden Medien, zwei Buttons eine TextView und einen Spinner zur Sortierung der angezeigten Liste. Einstiegspunkt der Methode "onCreateView" ist die zuweisung der Layout Ressource. Weiters werden die in der Klasse deklarierten Variablem mit den Anzeigeelementen verknüpft. Die Textview wird im Anschluss mit dem Namen des, für die Wiedergabe gewählten, Displays belegt. Um die Liste der Medien nach nach belieben zu sortieren wird ein ArrayAdapter initialisiert. Diesem werden die Ressourcen "tag_array", beinhaltet eine Liste mit Sortiermöglichkeiten für die angezeigten Medien, und "simple_spinner_item" um das Design für die angezeigten "Spinner" Elemente,

übergeben. Es erfolgt eine Zuweisung des Adapters an das Spinner Element. Um die Sortierung der Medien Liste zu realisieren wird ein "OnItemSelectedListener" implementiert der die Medien gefiltert nach ausgewähltem Tag anzeigt. Um zu beginn alle inhalte anzizeigen wird dem Spinner das erste Element des "tag_array" als Standardsortierung zugewiesen. Um zwischen den Einzelnen Displays zu wählen wird dem "Auswählen" Button ein "onClickListener" zugewiesen welcher die Navigation zum "ChooseDisplayFragment" einleitet. Des weiteren wurde die Methode "setRecyclerView" erstellt um die Daten nach Tag sortiert vom Server mittels "GET-Request" zu erhalten und diese auf der View anzeigen zu können.

ChooseDisplayFragment

Die Auswahl des Bildschirmes auf dem das Medium angezeigt werden soll erfolgt über dieses Fragment. Dazu werden die verfügbaren Displays vom Java-EE-Server abgefragt und in einer Liste angezeigt. Durch klicken auf den "Aüswählen" Button in einem der in der Liste angezeigten Display Elementen, wird dieser Display ausgewählt und Medien werden dann auf diesem Abgespielt.
Die RecyclerView Ressource um die zu wählenden Displays anzuzeigen ist in der Datei "fragment_choose_display.xml" angelegt. Zuweisen der Layout ressource auf die aktuelle View, initialisieren der Klassen Attribute und aufrufen der Methode "getDisplays" sind die einzigen Schritte der "onCreateView" Funktion."getDisplays" ist jene Methode die alle Displays die mit dem XIBO-Server verbunden sind per "GET-Request" abfragt beziehungsweise aufbereitet und der RecyclerView übergibt um diese auf der View anzuzeigen.

6.9 Strukturplan

Die am XIBO-Server liegenden Layouts werden in der Andoid Applikation formatiert als "JSON-Sting" angezeigt. Damit ist gewährleistet, dass der App-Benutzer eine Übersicht über die Struktur der einzelnen Layouts und deren Unterelemente zur Verfügung hat.

StructurePlanFragment

Dieses Fragment zeigt eine Übersicht der einzelnen Layouts in Form einer Liste an. Durch klicken der einzelnen Listen Elemente navigiert die Applikation zu einer Detailansicht über das gewählte Layout. Die Ressource zur gestaltung der Anzeige beinhaltet eine "RecyclerView" und heißt "fragment_structure_plan.xml". Zu

beginn der ”onCreateView” Methode wird zuerst eine ”View” erstellt welche das ”fragment_structure_plan.xml” Layout zugewiesen bekommt. Die ”RecyclerView” wird anschließend initialisiert und über einen ”GET-Request”, dessen Response die einzelnen Layouts und deren unterstrukturen,des Xibo-Servers in form eines JSON-Strings übermittelt, befüllt. Somit ist auf der Anzeige eine Liste mit auswählbaren Layouts vorhanden über die zu einer Detailansicht navigiert werden kann.

StructureDetailFragment

Ein Layout wird in diesem Fragment in Form eines ”JSON-Strings” mit allen dazugehörigen Unterelementen angezeigt. Wichtig hierbei ist es den Angezeigten ”JSON-String” richtig zu formatieren um die Übersichtlichkeit bei zu behalten.
In der Ressource Datei ”fragment_structure_detail.xml” befindet sich eine ”Text-View” als einziges Element. Die ”onCreateView” Methode der Klasse ”Structure-DetailFragment”, wird nach zuweisen der Layout Ressource der Struktur beschreibende ”JSON-String” aus dem ”Bundle” gelesen, dies geschieht über die Methode ”setArguemnts” der Basisklasse Fragment und deren Feld ”mArguments”, und dem Anzeigefeld zugewiesen. Anschließend wird dem Fragment noch die fähigkeit gegeben sich Scrollen zu lassen und die View wird zurückgegeben.

6.10 Request-Helper

Um in weiterer Folge die Anfragen an den Java-EE-Server einfach und einheitlich durchzuführen gibt es die Klasse ”RequestHelper” . In dieser Klasse gibt es neben den beiden Parametern ”responseBody” und ”responseCode”, welche zur Fehlerausgabe und zum Erhalt der Daten aus der Anfrage vorhanden sind, die Methode ”executeRequest”. Diese übernimmt die Hauptaufgabe der Klasse und führt die Anfragen an das Signage System durch. Werden Daten als Antwort der anfrage erwartet so kann die Methode mit ”Callback” Parameter aufgerufen werden, ist dies nicht der Fall so gibt es die Möglichkeit diese Methode ohne ”Callback” aufzurufen, dabei wird dieser Parameter mit dem Wert null überschrieben.
Die Parameter dieser Methode lauten wie folgt:

- *RequestTypeEnum:* Der Parameter vom Typ Enum wird genutzt um Herauszufinden welche Http Anfrage vorliegt. Mögliche Werte sind hierbei GET, POST, PUT und DELETE.

- *Params:* Hier liegt eine HashMap vor, die als Key-Value Paare alle benötigten Parameter für den RequestBody beinhaltet. Beispielsweise: "LayoutID":"78", hierbei ist "LayoutID" der Key und "78" das Value.
- *Url:* Beinhaltet die URL unter der die Anfrage erreichbar ist.
- *Callback:* Dieser Parameter wird benötigt, um auf eine Antwort des ausgeführten "REST-Request" zu warten. Wird ein "Response" erhalten und die Daten werden in dem angezeigten Fragment benötigt, können diese über eine "Lamda-Expression" in die gewünschten Anzeigeelemente eingefügt werden.

Zu Beginn der Methode wird anhand des Parameters RequestTypeEnum unterschieden, um welche Http Anfrage es sich handelt. Wird GET oder DELETE geliefert wird durch die HashMap iteriert und die einzelnen Key-Value Paare als QeryParameter in der URL einfügt. Beispielsweise:"<URL>/layout?layoutID=78"
.
Handelt es sich um eine POST oder PUT Anfrage so werden die Key-Value Paare im Body mitgegeben und im Format äpplication/x-www-form-urlencodedcodiert, andernfalls werden die Parameter als "Query-Parameter" übergeben, Anschließend wird die URL mittels HttpUrl.Builder erstellt und ausgegeben. Des Weiteren wird per Switch-Case dem Request die richtige Art der Anfrage zugewiesen und danach die URL übergeben.
Um die REST-Anfragen fertig zu stellen, wird das Interface Callback implementiert. Mit den beiden Methoden onFailure und onResponse wird dem Interface zugewiesen was passiert, wenn der Request fehlschlägt oder funktioniert.
onFailure: Sollte der Request fehlschlagen, wird im Log-Fenster der Responsecode und die Fehlermeldung/Exception ausgegeben. **onResponse:** Wird der Request ohne Fehler durchgeführt so wird im Log-Fenster ebenfalls der Responsecode und der Responsebody ausgegeben. Letzter schritt beim Erhalt des gewünschten "Response" ist es dem im Methoden Kopf übergebenen "Callback" Parameter auszuführen, dieser enthält zum Beispiel siehe Abbildung. Bild!!!!!!!!!!!
Der letzte Schritt ist es dem OkHttpClient mitzuteilen, dass er einen neuen Call ausführen soll. Als Parameter wird der Zusammengestellte Request mitgegeben. Über .enqueue wird dem Client gesagt er soll auf einen Response warten. Parameter für diese Methode ist das erstellte Interface Callback. [5]
Um Daten aus den Requests zu erhalten beziehungsweise Fehlerausgaben anzeigen zu können gibt es Getter zu den Feldern "responseBody" und "responseCode". Verläuft der Request fehlerfrei so werden die geforderten Werte in die variablen übertragen und können im weiteren verlauf durch die Methoden "getResponseCode" beziehungsweise "getResponseBody" (Getter) ausgelesen werden. Im Fehlerfall wird lediglich der "responseCode" mit dem Fehlercode belegt und kann ausgelesen werden.

Antworten des Servers werden in Form von JSON-Strings erhalten. Das Aufbereiten dieser Daten wird den Einzelnen Fragmenten, welche die Anfragen in Auftrag geben, überlassen, da die Erhaltenen Informationen von Fragment zu Fragment verschiedene Inhalte aufweisen.
Um klarzustellen wie das Aufbereiten der Daten von statten geht wird anhand des Beispiels "StructurePlanFragment" gezeigt was passiert sollte die Anfrage eine positive Antwort erhalten. Zu beachten ist, dass die Zuweisung der erhaltenen Daten in einer Lamda-Expression durchgeführt wird. Als erstes werden eine Liste von JSON-Objekten und ein JSONArray deklariert, um das Anzeigen und Durchlaufen der Informationen zu ermöglichen. Das Array wird hierbei zum Durchlaufen benötigt. Die LinkedList bildet die Quelle für die Anzeigedaten des Fragments. Da JSON-Strings im Folgenden Abschnitt geparst werden ist ein try-catch block nötig in dem Mögliche Fehlerfälle behandeln und ausgegeben zu können. Die übermittelten Daten werden über den "responseBody" Getter ausgelesen und in das JSONArray übernommen. Über Iteration durch das JSONArray werden die einzelnen JSONObjekte ausgelesen und der LinkedList angehängt. Nicht immer ist eine Iteration durch den Erhaltenen JSON-String nötig, beispielsweise bei der Anfragen in der der XIBO-Server-Status festgestellt wird. Im Anschluss werden die Anzeigeelemente mit den ausgelesenen Werten versehen, dabei ist zu beachten, dass dieser schritt über den UI-Thread ausgeführt wird da nach erzeugen der View die Anzeigeelemente verändert werden.

Literaturverzeichnis

[1] Xibo Open Source Digital Signage URL: https://xibo.org.uk/

[2] Swagger official Website URL: https://swagger.io/

[3] Postman official Website URL: https://www.getpostman.com/

[4] OAuth2 official Website URL: https://oauth.net/2/

[5] okHttp3 Dokumentation URL: https://square.github.io/okhttp/3.x/okhttp/

[6] Java-EE-Lamda Expressions Oracle start Page. URL: http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/Lambda-QuickStart/index.html

Kapitel 7

Summary

Here you give a summary of your results and experiences. You can add also some design alternatives you considered, but kicked out later. Furthermore you might have some ideas how to drive the work you accomplished in further directions.

Abbildungsverzeichnis

Tabellenverzeichnis

Project Log Book

Date	Participants	Todos	Due

Anhang A

Additional Information

If needed the appendix is the place where additional information concerning your thesis goes. Examples could be:

- Source Code
- Test Protocols
- Project Proposal
- Project Plan
- Individual Goals
- ...

Again this has to be aligned with the supervisor.

Anhang B

Individual Goals

This is just another example to show what content could go into the appendix.