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 Schulgebï¿ude um Projekte, aktuelle News und ï¿nderungen im Unterrichtsablauf anzuzeigen. Doch ein groï¿er Schwachpunkt dieser Multimedia Systeme ist, dass der Prozess vom Erstellen der Anzeige bis zum Zuordnen welcher Bildschirm welche Information anzeigen soll, sehr kompliziert und mï¿hselig ist. So werden neue Informationen erst verspï¿tet oder gar nicht angezeigt.

1.2 Ziele

Ziel ist es, dass die Schulverwaltung möglichst schnell ï¿berall in der Schule Informationen, Warnungen oder Ankï¿ndigungen anzeigen kann. Die verschiedenen Multimediasysteme sollen einheitlich gesteuert und verwaltet werden kï¿nnen, um schnell alle Anzeigen beliebig zu verï¿ndern. So ist es auch ein Teilziel festzustellen, ob es mï¿glich ist die derzeitig verwendeten Anzeigesysteme durch den XIBO Server zu ersetzen.

1.3 Problemstellung

Momentan wird um eine Anzeige zu ï¿ndern sehr viel Aufwand betrieben. Zum Beispiel wird eine neue Prï¿sentation in Form von Folien oder Video zusammen-

geschnitten. Beispiel dafï¿r ist die Anzeige im Eingangsbereich der Schule. Diese Vorgehensweise ist zeitaufwendig und werden ï¿nderungen vorgenommen, kann die alte Prï¿sentation 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?

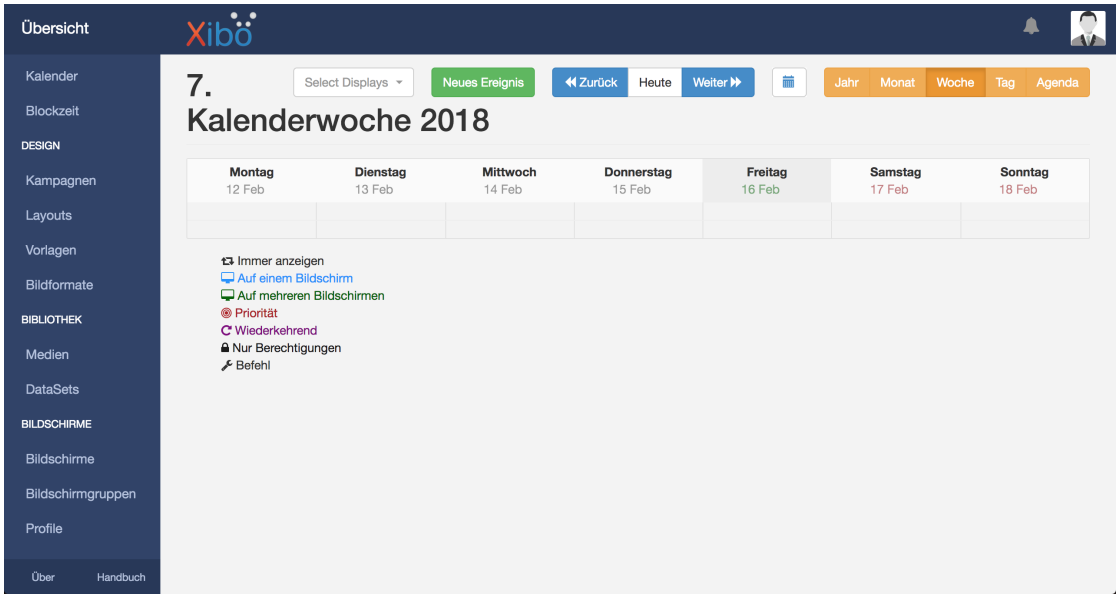
Das XIBO ist ein Open Source Digital Signage System entwickelt von der Spring Signage LTD. Das XIBO-System besteht aus vielen verschiedenen Komponenten. Das XIBO Paket besteht aus einem klassischen Server-Client Konstrukt. Der Server besteht aus drei Komponenten: das Content Managment System welches 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 Weboberfläche des XIBO

Das Steuerungszentrum des ganzen Signage System ist die WeboberflÃache, 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.

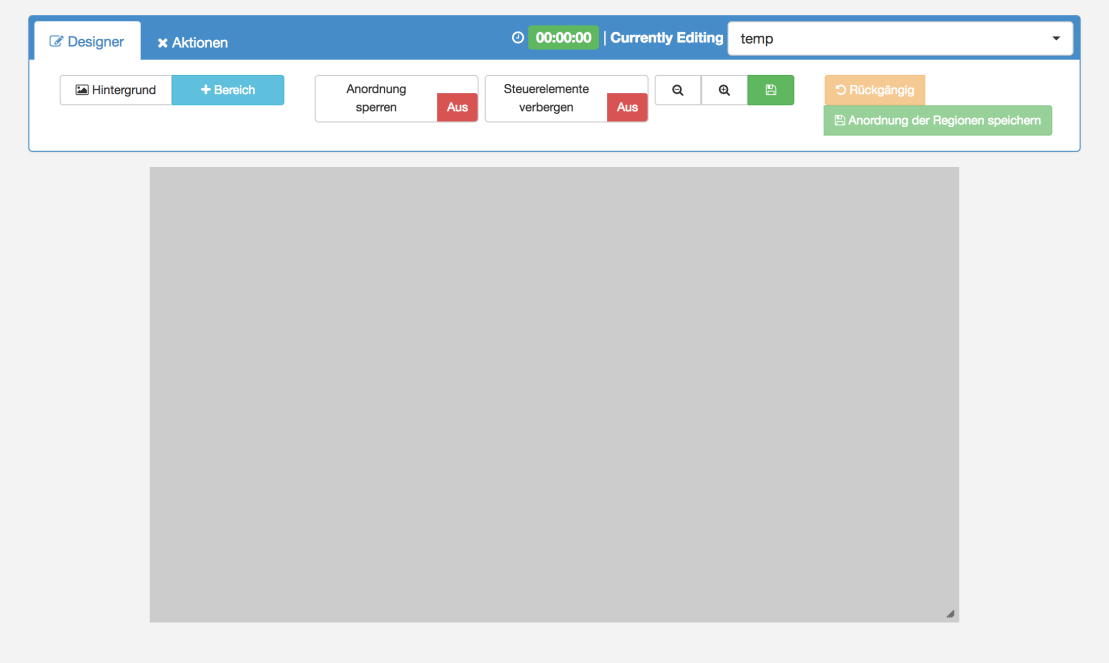
5. *Einstellungen:* Der Menüpunkt Einstellungen gibt dem Nutzer die Möglichkeit, verschiedene Optionen zu wählen. So sind zum Beispiel die richtige

Zeitzone, E-Mail Benachrichtigungen, wichtige Einstellungen, die für ein einwandfreies Funktionieren des Xibo-Servers zuständig zuständig sind.

2.4 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.

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.

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. VERWEIS!!!!!!!!!!!!!!!!!! https://oauth.net/2/

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.

3.4 Request-Helper

Um in weiterer Folge die Anfragen an den Digital Signage 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, auch noch die Methode ëxecuteRequest". Diese übernimmt die Hauptaufgabe der Klasse und fährt die Anfragen an das Signage System durch.

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.
- *Token:* Ist jener Parameter ” access_token ”,der benötigt wird um sich beim Server zu authentifizieren. Der Erhalt dieses Parameters, funktioniert wie bereits im vorigen Unterpunkt Authentifizierung beschrieben.

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=78token=ajdlfjkakawkflkd6545”. 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. Anschließend wird noch die URL mittels HttpUrl.Builder erstellt und ausgegeben. Anschließend wird per Switch-Case dem Request die richtige Art der Anfrage zugewiesen und danach wird auch noch die URL übergeben.

Um die Anfragen noch fertig zu stellen, wird noch das Interface Callback implementiert. Mit den beiden Methoden onFailure und onResponse wird dem Interface zugewiesen was passiert, wenn der Request fehlschlägt oder funktioniert.

Sollte der Request fehlschlagen, wird im Log-Fenster der Responsecode und die Fehlermeldung/Exception ausgegeben. Wird der Request ohne Fehler durchgeführt so wird im Log-Fenster ebenfalls der Responsecode und der Responsebody ausgegeben.

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.

Als Ergebnis der Anfrage, mit den Parametern, in der Form ———–STütz fragen ob bsp für request usw einbauen

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$_{P}latform,_{E}nterprise_{E}dition$

Kapitel 5

HomeDS - Server

5.1 Einleitung

Im Rahmen der Diplomarbeit wird neben dem XIBO-Server ein eigens programmierter 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. !!!1 Lesen 2 moi eigens programmierter serrver eingesetzt

5.2 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$_{P}latform,_{E}nterprise_{E}dition$

5.3 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.

5.4 Struktur des Projekt's

5.5 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.

5.6 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)

5.7 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.

5.8 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/

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" 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

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

Die "MainActivitiy" ist die Einzige "Activity" die in der Android Applikation benötigt wird. Hauptaufgabe der "MainActivity" 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" belegt wurde.

Methoden der "MainActivity":

- *onCreate:* Hier wird die "Main_Activity" 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 DataSet verwaltung

Die beiden Fragment

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.