

Diplomarbeit

Höhere Technische Bundeslehranstalt Leonding
Abteilung für Informatik



Digital Signage System

Eingereicht von: **Andrej Sakal, 5CHIF**
Felix Hofmann, 5CHIF
Datum: **April 4, 2018**
Betreuer: **Prof. Mag. Dr. 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

Abstract

The HomeDS is a compact digital signage system that is tailored to the requirements. This allows the user to access Signage System features without having to read the signage system manual.

The result of this work is a web interface and an Android application for mobile devices. These should fulfill the functional scope described in the task.

Zusammenfassung

Beim HomeDS handelt es sich um ein kompaktes Digital Signage System, das auf die ausgewählten Anforderungen zugeschnitten ist. Dadurch wird ermöglicht, dass der Benutzer auf Funktionen des Signage System zugreifen kann ohne sich in das Handbuch eines Signage System einlesen zu müssen.

Das Ergebnis dieser Arbeit ist eine Weboberfläche und eine Android Applikation für mobile Endgeräte. Diese sollen den Funktionsumfang, welcher in der Aufgabenstellung beschrieben ist, erfüllen.

Danksagung

An dieser Stelle möchten wir uns sehr herzlich bei der HTBLA Leonding für die kompetente Betreuung und Unterstützung der letzten Jahre bedanken. Insbesondere sind wir auch Prof. Mag. Dr. Stütz zu tiefstem Dank verpflichtet, welcher uns während des Projektes tatkräftig zur Seite stand und jederzeit für Fragen erreichbar war. Natürlich möchten wir auch unseren Eltern und Bezugspersonen einen herzlichen Dank aussprechen, welche in den letzten Jahren stets ein offenes Ohr für unsere Probleme hatten und uns auch in schwierigeren Zeiten ein Ansporn waren.

Inhaltsverzeichnis

1 Einleitung	8
1.1 Ausgangssituation	8
1.2 Problemstellung	8
1.3 Aufgabenstellung	9
1.4 Ziele	9
2 Digital Signage & XIBO	12
2.1 Was ist Digital Signage?	12
2.2 Digital Signage Anwendungen	13
2.3 Was ist XIBO?	13
2.4 Weboberfläche des XIBO	14
2.5 Designen mit XIBO	15
3 XIBO-Server	20
3.1 Beschreibung	20
3.2 API	21
3.3 Authentifizierung	22
4 Verwendete Technologien	28
4.1 Git und GitHub	28
4.2 Android	29
4.3 Java Enterprise Edition	29
4.4 JSF - Java Server Faces	29
4.5 IntelliJ IDEA	29
4.6 Android Studio	30
4.7 Draw IO	30
5 HomeDS - Server	33
5.1 Einleitung	33
5.2 Anforderungen an den HomeDS Server	34
5.3 Komponenten des HomeDS Server	35
5.4 Funktionen des JavaEE	36

5.4.1	Weboberfläche des JavaEE	36
5.4.2	Nachrichten-Pakete ändern - HomeDS Web	38
5.4.3	Medien abspielen - HomeDS Web	40
5.4.4	Structure Crawler - HomeDS Web	42
5.4.5	HomeDS Server Slideshow - HomeDS Web	42
5.4.6	Internationalisierung - HomeDS Web	43
5.5	Funktionen des JavaEE - Technischer Hintergrund	43
5.5.1	Nachrichten-Pakete ändern - Technisch	44
5.5.2	Medien abspielen - Technisch	44
5.5.3	JavaEE mit REST	45
5.5.4	Swagger	46
5.5.5	Structure Crawler - Technisch	47
5.5.6	HomeDS Server Slideshow - Technisch	48
6	Android Applikation	51
6.1	Einleitung	51
6.2	Anforderungen	52
6.3	Struktur	53
6.4	Benutzerhandbuch	53
6.4.1	Startseite	53
6.4.2	Abspielen von Medien auf gewünschten Bildschirmen	54
6.4.3	Meldungen Anzeigen	56
6.4.4	Strukturübersicht	59
6.5	MainBottomNavigationActivity und OverviewFragment	60
6.5.1	MainBottomNavigationActivity	60
6.5.2	HomeScreenFragment	63
6.6	DataSet Verwaltung	66
6.7	Mediaplayer	72
6.8	Strukturplan	77
6.9	Request-Helper	80
7	Continuous Integration	86
7.1	Einleitung	86
7.2	Was ist CI?	86
7.3	Wieso CI?	87
7.4	Wie kann CI realisiert werden?	87
7.5	Jenkins installieren	88
7.6	Jenkins CI konfigurieren	90
8	Arbeitsaufteilung	95
8.1	Arbeit von Sakal Andrej	95

8.2 Arbeit von Hofmann Felix	95
8.3 Genaue Aufteilung der Arbeit	95
Abbildungsverzeichnis	102
Codebeispiele	104
HomeDS - Repository	105

Kapitel 1

Einleitung

1.1 Ausgangssituation

Die HTL Leonding ist ausgestattet mit modernen Multimedia Systemen wie einer Videowall, einem riesigem Touchscreen TV und mehreren Bildschirmen. Diese Bildschirme werden nicht sehr effektiv genutzt, da auf jeder Anzeige die Inhalte separat übermittelt werden müssen.

1.2 Problemstellung

Momentan wird ein großer Aufwand betrieben, um eine einfache Slideshow von Bildern auf einem Bildschirm anzuzeigen. Da nur ein Video-Format unterstützt wird, müssen die Fotos manuell in ein Video verarbeitet werden und dann auf den Bildschirm, über einen DVD-Player, abgespielt werden. Des Weiteren können keine dynamischen Daten angezeigt werden. Dieser Prozess ist sehr langwierig und ist in der Praxis viel zu aufwendig.

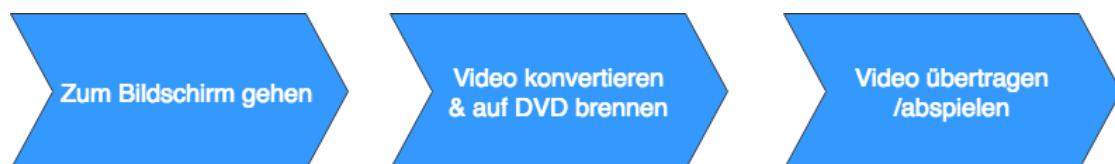


Abbildung 1.1: Prozess des Anzeigens

1.3 Aufgabenstellung

Im Rahmen dieser Arbeit war unsere Aufgabenstellung den bestehenden Signage Server um folgende Funktionen zu erweitern:

1. *Meldungen*: Anzeigen von Meldungen der Schulleitung, um Informationen den Schülern näher zu bringen.
2. *Slideshow*: Schaffen einer Plattform die es ermöglicht Bilder vom vm59.htl-leonding.ac.at Server über das Signage System anzuzeigen.
3. *Medien abspielen*: Dem Benutzer soll es ermöglicht werden, Medien die auf dem Digital Signage Server liegen auf der gewünschten Anzeige abspielen zu können.
4. *Strukturübersicht*: Roh formatierte Ausgabe der Layouts die sich auf dem Signage Server befinden, um eine Übersicht über die Struktur zu erhalten.

1.4 Ziele

Ziel ist es, dass aktuelle Nachrichten oder Informationen den Schülern und Schülerinnen der HTL Leonding näher gebracht werden. Diese Nachrichten können verschiedenster Natur sein. Zum Beispiel Ankündigungen, Warnung oder sonstige hilfreiche Informationen. Der Aufwand für das Erstellen von Slideshows, Informationselementen soll vermindert werden. Es sollen mehrere Datentypen unterstützt werden und auch das Präsentieren von Projekten erleichtert werden.

Kapitel 2

Digital Signage & XIBO

2.1 Was ist Digital Signage?

Digital Signage, in Deutsch Digitales Schild, hat grundsätzlich die Aufgabe Inhalte die meist auf Plakaten oder Schildern angezeigt werden auf Bildschirmen anzugeben. Mithilfe von Digital Signage Systemen soll das zeit- oder interaktionsgesteuerte Ändern von Inhalten auf den Bildschirmen einfach und übersichtlich gehalten werden (siehe Abbildung 2.1). Zusätzlich 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.

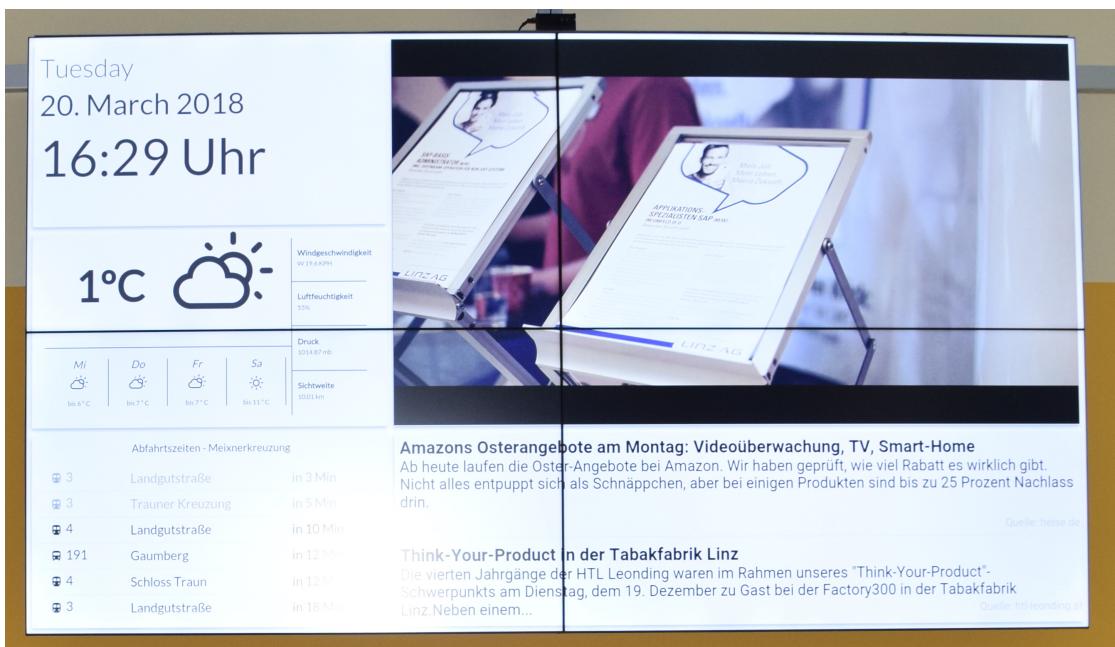


Abbildung 2.1: Digital Signage in der HTL Leonding

2.2 Digital Signage Anwendungen

Digital Signage hat keine Grenzen und kann vielfältig eingesetzt werden. Viele Konzerne nutzen Digital Signage für Marketingzwecke, Produkte zu präsentieren oder oft auch nur als Lockmittel.

2.3 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 Management 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. [1]

2.4 Weboberfläche des XIBO

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

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

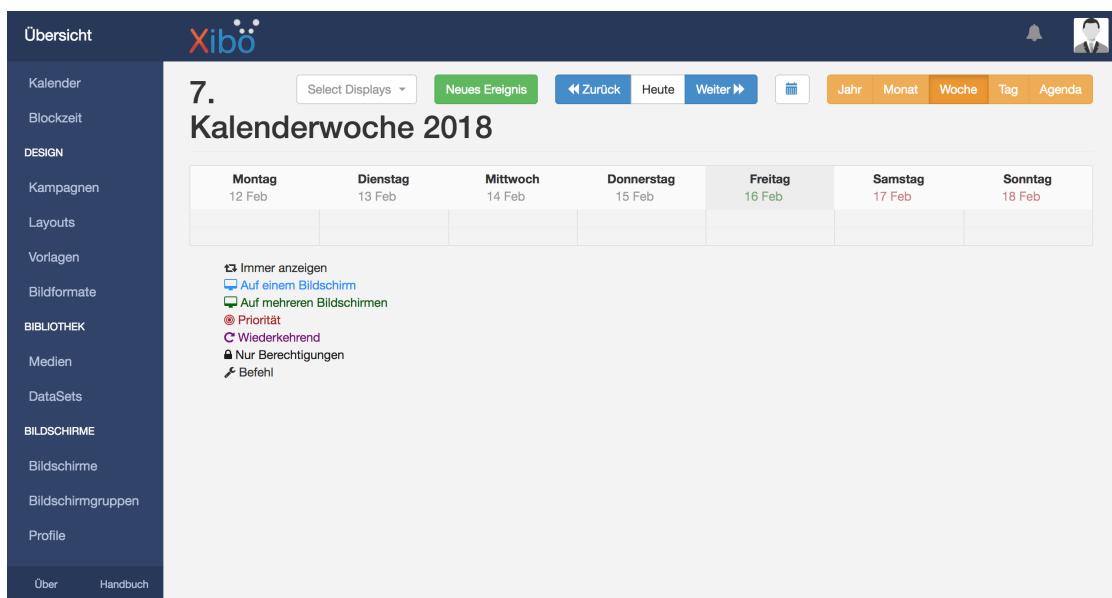


Abbildung 2.2: XIBO - Kalender

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 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 sind einzustellen. Aus den Einstellungen ist auch der CMS geheimer Schlüssel zu finden, der für die Authentifizierung der API Zugriffe zuständig ist herauszulesen.

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

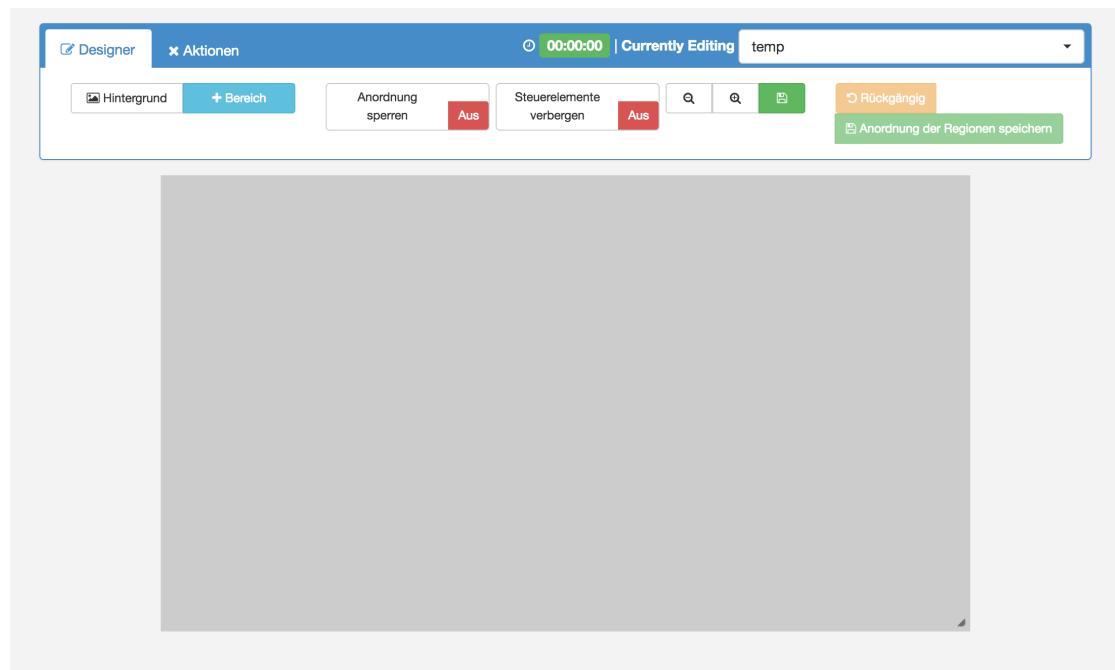


Abbildung 2.3: XIBO-Layout designen

Dem Layout kann nun eine oder mehrere Regionen hinzugefügt werden. Eine Re-

gion kann mehrere Widgets enthalten (siehe Abbildung ??).



Abbildung 2.4: Regions mit Widgets - XIBO

Mit einem Doppelklick auf die Region kann ein Widget hinzugefügt werden. Es gibt viele verschiedene Arten von Widgets:

Bibliothek: Mit diesem Widget können Elemente aus der Medienbibliothek in der Region angezeigt werden. Dabei werden PowerPoint Formate, Video, Bilder und andere Medien Datentypen unterstützt.

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

DataSet: Das DataSet Widget ist sehr wichtig und zeigt einfach gesagt nacheinander Daten aus einem Array, mit Key Value Paaren, an.

Weather: Das Weather Widget, in Deutsch Wetter, zeigt das aktuelle Wetter an. Es kann eingestellt werden ob es anhand von den GPS-Daten des Bildschirmes die Wetterdaten anzeigen soll oder ein vorher definierter Ort für die Daten verwendet werden soll.

Flash: Mit dem Flash Widget können Flash Inhalte abgespielt werden.

HLS: Mit dem HLS Widget können HLS Video Streams angezeigt werden.

Image: Mit dem Image Widget können Bilder entweder aus der XIBO Bibliothek angezeigt oder neue hochgeladen werden.

Local Video: Mit dem Local Video Widget können Videos oder Streams angezeigt werden.

PDF: Mit dem PDF Widget können PDFs entweder aus der XIBO Bibliothek angezeigt oder neue hochgeladen werden.

PowerPoint: Mit dem PowerPoint Widget können PowerPoint Präsentationen entweder aus der XIBO Bibliothek angezeigt oder neue hochgeladen werden.

Text: Mit dem Text Widget können Texte angezeigt werden.

Ticker: Mit dem Ticker Widget können Texte animiert angezeigt werden. Dabei können diese Texte aus einem DataSet oder einem RSS Feed stammen.

Webpage: Mit dem Webpage Widget können Webseiten angezeigt werden.

Nachdem eines der Widgets erstellt wurde, kann das Ergebnis des Layouts mit einer Layout Vorschau kurz überprüft werden.

Kapitel 3

XIBO-Server

3.1 Beschreibung

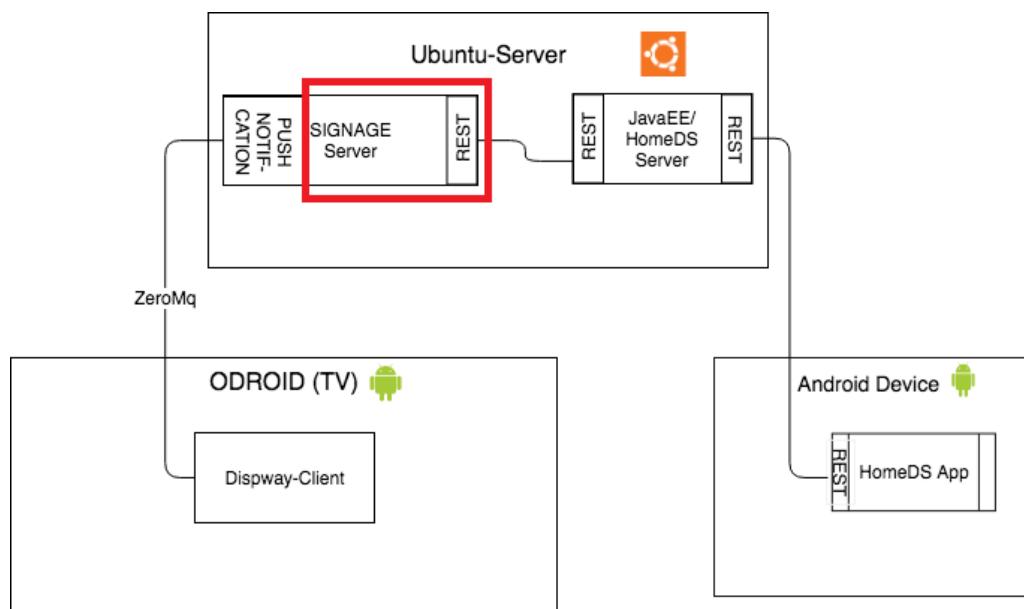


Abbildung 3.1: Systemkomponente XIBO-Server

Als zentrale Steuereinheit wird ein XIBO-Server verwendet. Der XIBO-Server bietet die benötigten Funktionalitäten wie zum Beispiel:

- *Medieninhalte abspielen*
- *Zeitsteuerung der anzuzeigenden Informationen*

- Verteilung der Anzeigedaten an die verbundenen Clients

Es gibt zwei Möglichkeiten den Funktionsumfang des XIBO-Servers zu nutzen. Zum einen über das Server interne Web-Interface, zum anderen hat man die Möglichkeit den Server über die eingebaute REST-Schnittstelle anzusprechen.(Siehe Abbildung 3.2) [1]

3.2 API

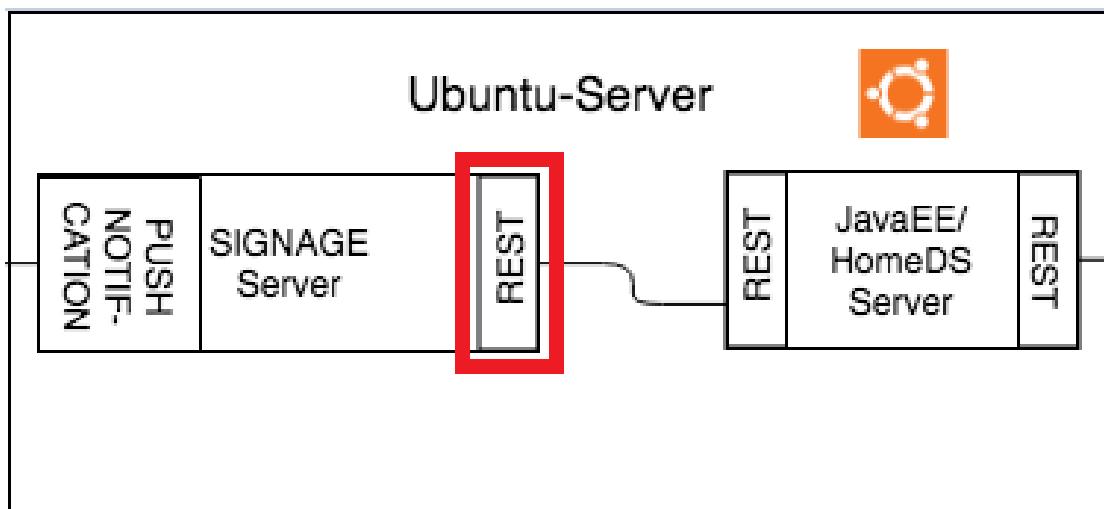


Abbildung 3.2: Gliederung Ubuntu-Server

Die API des XIBO-Servers ist mittels Swagger dokumentiert. Diese Dokumentation deckt die Grundfunktionalitäten und die Form der Anfragen ab. Die Schnittstelle des Servers dient als wesentliches Verbindungsstück zwischen der eigens entwickelten Steuerungssoftware und dem XIBO-Server. Wie der XIBO-Server die verschiedenen Anfragen verarbeitet und entgegennimmt wird, bevor die Implementierung des Java-EE-Servers beginnt, mittels Postman getestet. Diese Vorgehensweise ist nötig um festzustellen, ob das XIBO-System über REST-API ausreichend konfigurierbar und im operativen Betrieb steuerbar ist. Die Anfragen an den Server werden im Java Code durch die "libary" OkHttp3 übernommen.[2][3][5]

3.3 Authentifizierung

Die Authentifizierung einer Client-Applikation per REST-Anfrage am XIBO-Server erfolgt über OAuth2 , also mittels Access Token. [4]

Zunächst ist am XIBO-Server ein "Application"-Objekt im Webinterface zu erstellen. Beim Erstellen des "Application"-Objekts können die Berechtigungen für den Client festgelegt werden. Nachdem das Objekt erstellt wird, stellt dieses ein "Client Secret" zur Verfügung. Dieses ist für jeden Client eindeutig.(Siehe Abbildung 3.3 und 3.4)

The screenshot shows the 'Anwendung bearbeiten' (Edit Application) dialog. It has two tabs: 'Allgemein' (General) and 'Berechtigungen' (Permissions), with 'Allgemein' selected. The 'Anwendungsname' (Application Name) is set to 'Test'. Under 'Client ID', the value is '48AH8BorxHfrMzseMNPs1NVjkgI5FGNNKuJT2MMX'. Under 'Client Secret', the value is 'ePnHrfFMtHN1BijA3T1CQDUJ4p4FZyLH3T1beSu0rnlwUtoiLApJ8X4F4fG48U6tBwYA', with a link 'In die Zwischenablage kopi...' (Copy to clipboard). There are three checkboxes: 'Secret zurücksetzen?' (Reset secret?), 'Autentifizierungscode' (Authorization code), and 'Client Anmelddaten' (Client credentials). The 'Umleitungs URI' (Redirect URI) field is empty. At the bottom, there are buttons for 'Hilfe' (Help), 'Abbrechen' (Cancel), and 'Speichern' (Save).

Abbildung 3.3: Grundeinstellungen einer XIBO-Anwendung

Anwendung bearbeiten

All access
 Media Conversion as a Service

Besitzer

Set the owner of this Application. If you are not an admin you will not be able to reverse this action.

Hilfe Abbrechen Speichern

Abbildung 3.4: Mögliche Berechtigungen für eine XIBO-Anwendung

Damit ein externer Client auf den XIBO-Server per REST-Request zugreifen beziehungsweise Anweisungen an diesen geben kann, wird ein POST-Request mit folgenden Parametern abgesetzt.

Die Parameter:

- *Client_ID*: Wird vom XIBO-Server bereitgestellt
- *Client_Secret*: Wird vom XIBO-Server bereitgestellt
- *grant_type*: Muss in der Form "grant_type=client_credentials" übergeben werden

Die über den POST-Request erhaltene Antwort, liefert einen Access Token, welcher 60 Minuten gültig ist und nach Ablauf erneuert werden muss, um weiter über die API zu kommunizieren. Dieser Token muss bei jedem Request an den XIBO-Server im Header des Requests übergeben werden, damit der XIBO-Server feststellen kann, ob es sich um einen registrierten Client handelt.

Um die Authentifizierung zu automatisieren, wurde eine Java Klasse entwickelt. Diese ist zu finden im HomeDsBackend (HomeDS/HomeDsBackend/src/main/java/at/htl/utils/AuthentificationHandler.java). Funktionsweise dieser wird nachfolgend geschildert:

Um eine Verbindung zum XIBO-Server herstellen zu können, wird eine URL und eine HttpURLConnection deklariert. Im Anschluss wird die URL mit der richtigen

Adresse belegt. Die HttpURLConnection wird über den Befehl "httpURLConnection.openConnection()" angewiesen, eine Verbindung aufzubauen. Durch die Anweisung "httpURLConnection.setDoOutput(true)" wird der Connection mitgeteilt, dass als Antwort Daten erhalten werden. Die Art der Anfrage wird als POST-Request festgelegt. httpurlconnection

```
1 URL obj;
2 HttpURLConnection con = null;
3 try {
4     // Building Connection
5     obj = new URL(new RequestHelper()
6         .BASE_URL + AUTHORIZE_URL);
7     con = (HttpURLConnection) obj.openConnection();
8     con.setRequestProperty("Content-Type",
9         "application/x-www-form-urlencoded");
10    con.setDoOutput(true);
11    con.setRequestMethod("POST");
```

Code 3.1: Erstellen der Verbindung zum Server

Um die für die Authentifizierung am XIBO-Server geforderten Parameter "client_id", "client_secret" und "grant_type" übergeben zu können wird ein "DataOutputStream" deklariert und mit den benötigten Werten versehen. Anschließend wird über den Befehl "DataoutputStrem.flush" dem "DataOutputStream" mitgeteilt, dass er die Daten über die Verbindung senden soll.

```
1 DataOutputStream write
2     = new DataOutputStream(con.getOutputStream());
3
4 String body = "client_id=" + CLIENT_ID
5     + "&client_secret=" + CLIENT_SECRET
6     + "&grant_type=client_credentials";
7
8 write.writeBytes(body);
9 write.flush();
```

Code 3.2: Erstellen und senden des JSON-Body

Um auftretende Fehler besser finden zu können, werden der übergebene Request-Body, die URL über die der Request durchgeführt wurde und der erhaltene Response-Code im Log-Fenster ausgegeben. Die Daten, die anschließend vom XIBO-Server als Antwort erhalten werden, werden durch einen "BufferedReader", dieser bekommt bei der Instanzierung den "InputStream" der "httpURLConnection" übergeben, entgegengenommen. Solange vom Server Daten erhalten werden, wird über einen "StringBuilder" ein String um jene erhaltenen Daten erweitert. Im Anschluss wird die "BufferedReader" Verbindung geschlossen und der erhaltene "Access_Token" wird im Log-Fenster angezeigt.

```

1 System.out.println("nPost-Body: " + body);
2 System.out.println("nSending 'AUTHORIZATION',
3     request to URL : " + AUTHORIZE_URL);
4
5 System.out.println("Response Code :
6     " + con.getResponseCode());
7
8 BufferedReader in = new BufferedReader(
9     new InputStreamReader(con.getInputStream()));
10
11 String output;
12 StringBuilder response = new StringBuilder();
13 while ((output = in.readLine()) != null) {
14     response.append(output);
15 }

```

Code 3.3: Erhalt der Daten vom Server

Als nächstes wird der erhaltene Token per "return" Statement als Ergebnis der Methode übergeben. Abschließend wird im "finally" Block überprüft, ob die "HttpURLConnection" noch geöffnet beziehungsweise vorhanden ist. Sollte dies der Fall sein so wird die Verbindung geschlossen.

```

1     in.close();
2     System.out.println(response.toString());
3
4     return new JSONObject(
5         response.toString()).getString("access_token");
6
7 } catch (MalformedURLException e) {
8     e.printStackTrace();
9 } catch (IOException e) {
10    e.printStackTrace();
11 }
12 finally {
13     if (con != null)
14         con.disconnect();
15 }

```

Code 3.4: Rückgabe des Token und schließen der Verbindung

Kapitel 4

Verwendete Technologien

4.1 Git und GitHub

Um als Team dynamisch 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 als Repository hosten können. Dies ermöglicht einfaches Arbeiten im Team und verhindert in den meisten Fällen Zusammenführungskonflikte. Mittels Git lässt sich auch einfach zurückverfolgen welches Teammitglied welche Änderungen gemacht hat. Bei Bedarf ist es möglich diese Änderungen zurückzusetzen.

Verwendet wird GitHub für die gesamte Diplomarbeit, sowohl für die Versionsverwaltung der Dokumente, als auch die einzelnen Applikationen. Um sicherzustellen, dass 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.

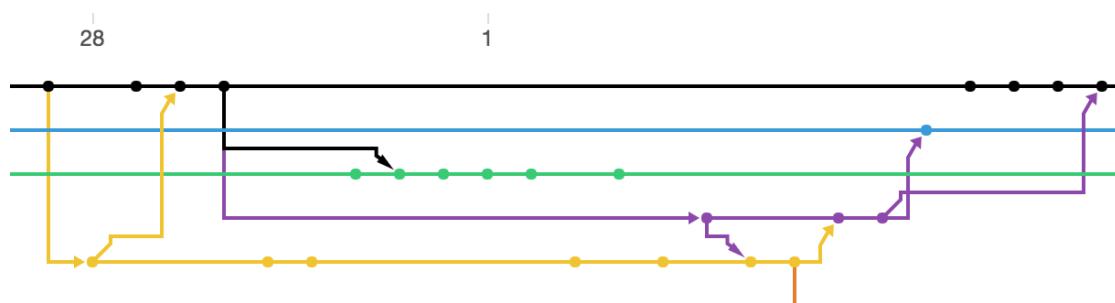


Abbildung 4.1: Github - Repository Network Diagramm

4.2 Android

Android ist ein Betriebssystem für mobile Endgeräte, spezialisiert auf Touch-Anwendungen. Ziel ist es das Endgerät möglichst intuitiv und flexibel bedienen zu können. Mit Android ist es möglich open-source Applikationen zu erstellen, welche ein großes Publikum erreichen. Google stellt hier seinen eigenen "PlayStore" zur Verfügung, in dem die Applikationen gratis oder auch gegen Entgelt erworben werden können. Diese Aspekte open-source, frei erhältlich und das Erreichen großes Publikum, sind ausschlaggebend dafür, dass die Applikationen in Android implementiert wird. Als Programmiersprache wird Java verwendet.

4.3 Java Enterprise Edition

Die Java Platform, Enterprise Edition oder abgekürzt auch Java EE ist die technische nähere Beschreibung einer Softwarearchitektur, die programmierte Java Anwendungen ausführt. JavaEE baut auf JavaSE auf und ist dafür entwickelt worden um zuverlässige, sichere und skalierbare Netzwerkanwendungen erstellen zu können. [7][8]

4.4 JSF - Java Server Faces

JSF ist ein Java Enterprise Edition Framework, welches zur Entwicklung von Webanwendungen verwendet wird. JSF wurde als Nachfolger von JSP eingeführt, um die Mischung von HTML Code und Java Code übersichtlicher zu gestalten. Ziel der JSF Vorgehensweise ist es Anwendungen in Komponenten zu teilen, welche im besten Fall wiederverwendbar sind. [9]

4.5 IntelliJ IDEA

Herausgeber dieser Entwicklungsumgebung ist JetBrains. Durch diese IDE wird die Entwicklung des Java-EE-Backends unterstützt, wobei Java nicht die einzige Programmiersprache ist, die in IntelliJ verwendet werden kann. Andere Programmiersprachen, wie zum Beispiel Groovy und Kotlin, können in der IDE ebenfalls programmiert werden. Des Weiteren gibt es eine Vielzahl an Tools wie zum Beispiel der

direkte Zugang zu diversen Datenbanken, beispielsweise wie MySQL Datenbanken, oder die Integration von "build automation tools" wie bower oder grunt. Zudem sind verschiedenste Versionsverwaltungssysteme wie GitHub mit IntelliJ kompatibel und ebenfalls direkt über die Benutzeroberfläche der IDE verwendbar.[10]

4.6 Android Studio

Android Studio ist eine von Google und JetBrains bereitgestellte integrierte Entwicklungsumgebung, um Android Applikationen zu entwickeln. Grundgerüst für die Entwicklungsumgebung ist die "IntelliJ IDEA". Zudem können über Android Studio Emulatoren verschiedenster Android Geräte heruntergeladen und gestartet werden, damit erstellte Applikationen leicht zu probieren sind. Zusätzlich bietet Android Studio GitHub Integration und einen großen Umfang an Werkzeugen und Testmöglichkeiten. [11]

4.7 Draw IO

Unsere Grafiken wurden mithilfe von dem Online-Tool "Draw.io" erstellt. [12]

Kapitel 5

HomeDS - Server

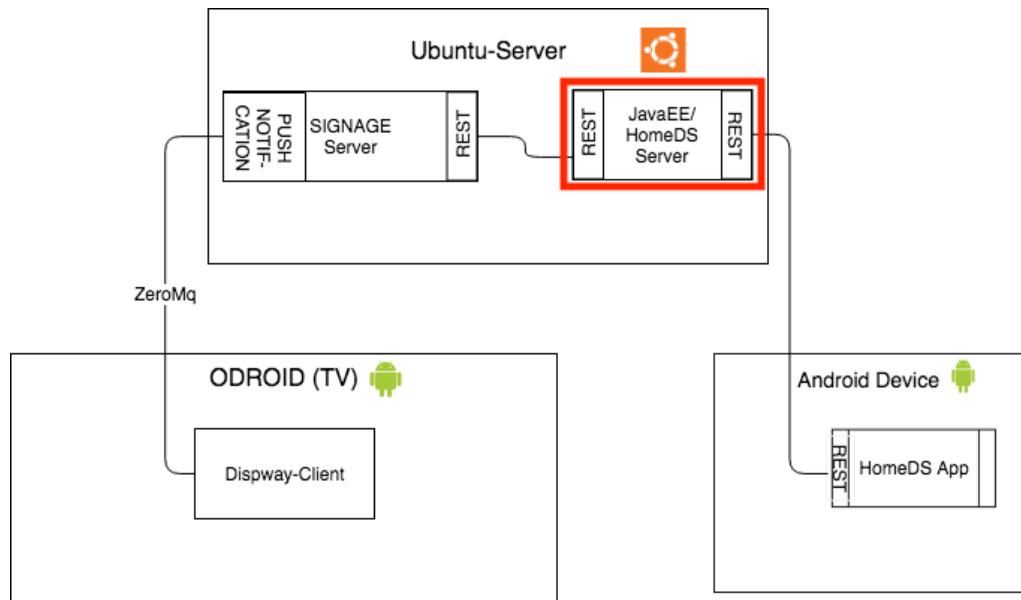


Abbildung 5.1: JavaEE - HomeDS

5.1 Einleitung

Im Rahmen der Diplomarbeit muss neben dem XIBO-Server ein weiterer Server eingesetzt werden. Es handelt sich um einen Java Enterprise Edition Server. Aufgrund der hohen Komplexität des Signage-Servers aber dünn dokumentierten API-Schnittstellen und begrenzten technischen Möglichkeiten, muss ein weiterer Server eingesetzt werden. Dieser wird für die Kommunikation zwischen Signage Server

und Applikation-Client erleichtern. Der JavaEE Server verwendet die Funktionen des Signage Servers und baut diese meist aus, sodass neue Funktionen entstehen können.

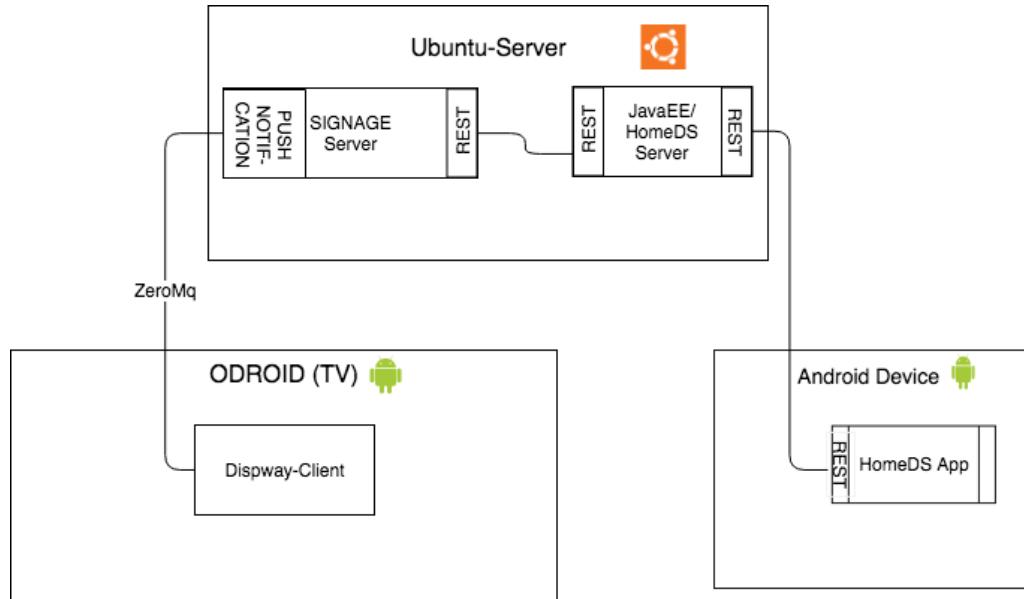


Abbildung 5.2: Systemarchitektur - HomeDS

5.2 Anforderungen an den HomeDS Server

Der JavaEE Server soll die verschiedenen komplizierten Abläufe des XIBO-Servers vereinfachen. Die Zugriffe mittels REST die eigentlich direkt auf den XIBO laufen sollen, werden über den JavaEE Server verwaltet (siehe Abbildung 5.2). Das heißt, der Java Enterprise Server empfängt REST Zugriffe, verarbeitet diese entsprechend und tätigt dann REST Abfragen auf den XIBO Server. Dieser benachrichtigt dann mittels PUSH Notifications über ZeroMQ die Display Clients. Das Einsetzen eines weiteren JavaEE Server hat insofern einen großen Vorteil, da die komplizierte Authentifizierungen wegfällt. Des Weiteren können ohne Probleme neue Funktionen hinzugefügt und neue Anforderungen flexibel erfüllt werden. Aufgrund dieser Vorgehensweise müssen Grundfunktionen des Signage Servers nicht erneut programmiert werden. Es werden lediglich über die möglichen API-Schnittstellen neue Funktionen und Erweiterungen speziell für die benötigten Anwendungsfälle erstellt. (siehe Abbildung 5.3)

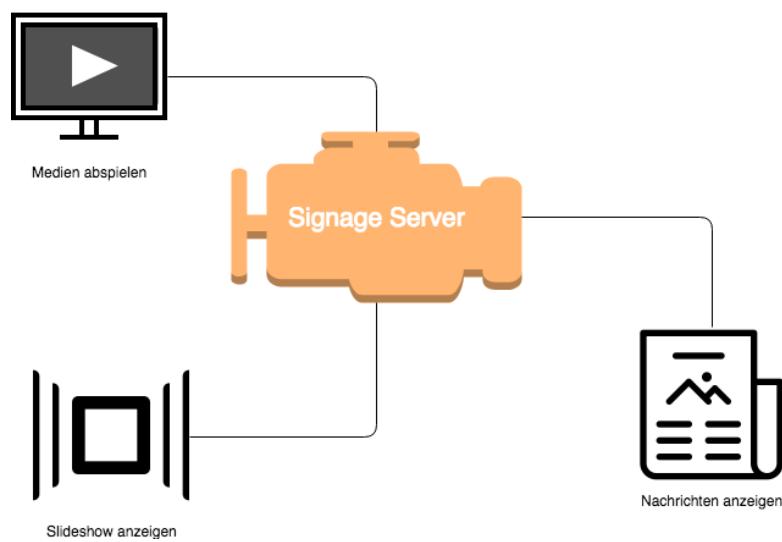


Abbildung 5.3: Signage Server die Engine - HomeDS

5.3 Komponenten des HomeDS Server

Der Server ist in Java programmiert und verfügt über eine eigene MySQL Datenbank. Über REST Schnittstellen sind die Funktionen des Servers verfügbar. Des Weiteren befindet sich auf dem Server auch eine JSF Weboberfläche also Java Server Faces Komponente. Der ganze Server wird auf einem Wildfly Application Server mittels continuous integration deployed.

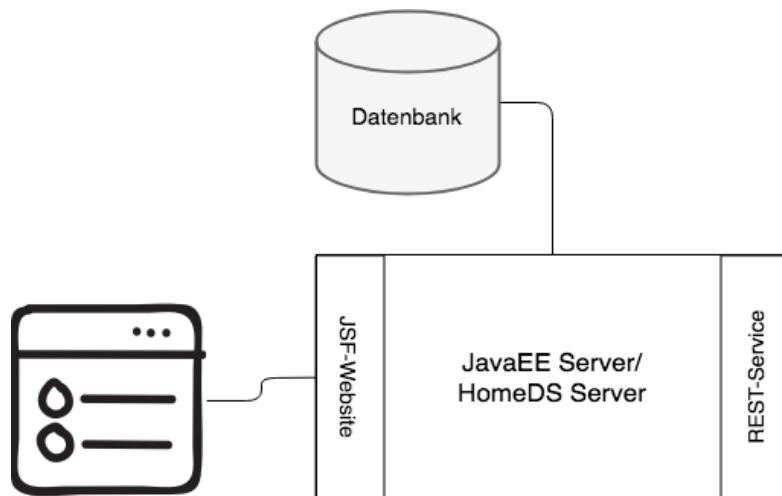


Abbildung 5.4: Komponenten des HomeDS Server

5.4 Funktionen des JavaEE

5.4.1 Weboberfläche des JavaEE

Die HTL Leonding kann mithilfe von einem Signage Server und der HomeDS Webapplikation Projektvideos der Schüler auf einen gewünschten Bildschirm anzeigen. Zum Beispiel kann das Sekretariat der HTL Leonding neuste Nachrichten oder Informationen in kürzester Zeit in der ganzen Schule auf allen Bildschirmen im System anzeigen lassen.

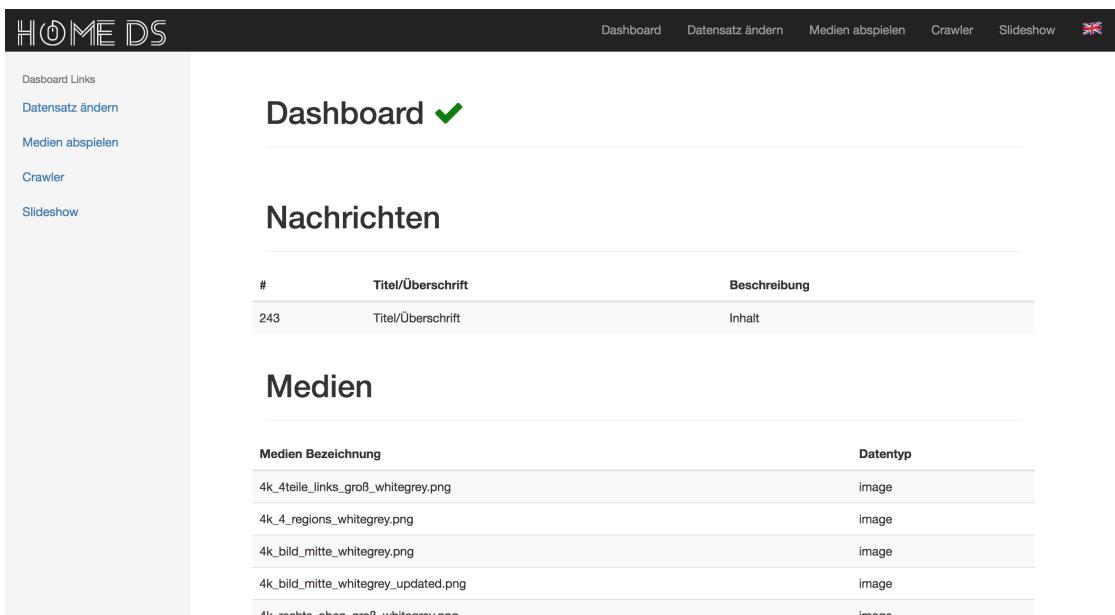


Abbildung 5.5: Startseite - HomeDsWeb

Unsere Weboberfläche ist responsive gestaltet und mithilfe von Java Server Faces mit BootsFaces realisiert worden. Das Bootsfaces Framework stellt fertige Komponenten zur Verfügung wie Buttons, Listen, Forms etc. Dabei wurde Bootstrap als Vorbild hergenommen.

Es war wichtig die Oberfläche so zu gestalten, dass diese übersichtlich und leicht zu bedienen ist. Das Designen wurde in enger Zusammenarbeit mit dem zukünftigen Benutzern der Software ausgearbeitet. Als Vorbild fungiert das Design eines Cockpits in einem Flugzeug wo vielen Funktionen direkt bei der Hand liegen müssen und trotzdem alles übersichtlich gestaltet ist. In der Abbildung 5.5 ist das Dashboard abgebildet. Hier erkennt man nun, dass es sich um die Kommandozentrale der Applikation handelt. Von dieser Seite aus kann man über eine Top-Navigationsleiste oder eine Left Sidebar Navigation zu allen Funktionen der Website gelangen.

Um den Nutzer immer ein Feedback über den aktuellen Status der Verbindung vom XIBO zu JavaEE Server zu geben, wird bei erfolgreicher Verbindung ein grünes Häkchen visualisiert, versehen mit einer Tooltip. Oder bei keiner Verbindung zum Server ein rotes "X" dargestellt (siehe Abbildung 5.6)s.

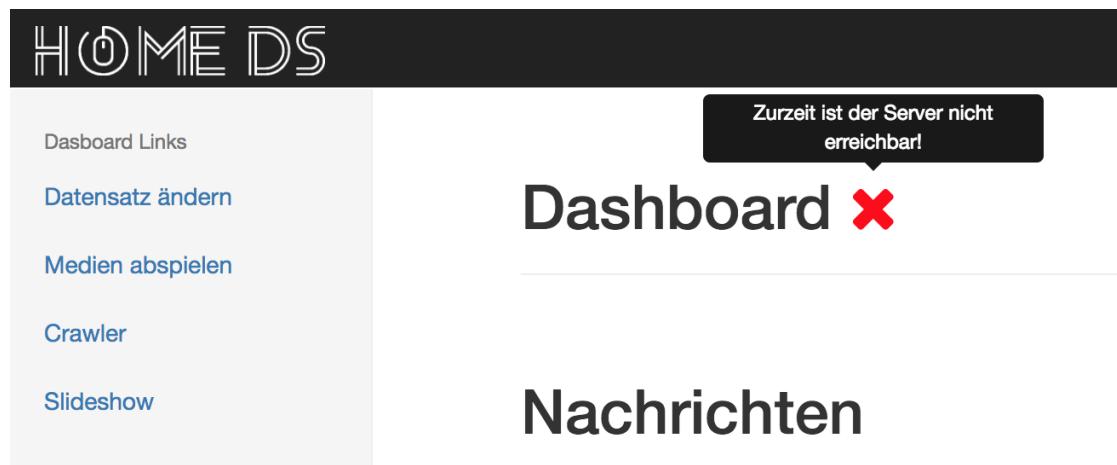


Abbildung 5.6: Keine Verbindung zum XIBO - HomeDsWeb

5.4.2 Nachrichten-Pakete ändern - HomeDS Web

Die "Datensatz ändern" Funktion auch genannt "Nachrichten-Pakete ändern", "DataSet bearbeiten" oder "Nachrichten-Pakete bearbeiten" kann über das Dashboard erreicht werden. Die Aufgabe der "Nachrichten Paket ändern" Funktion ist es ein DataSet zu ändern, hinzuzufügen oder zu löschen. Diese Seite besteht einfach gesagt aus zwei Teilen. Diese zwei Teile können bei Bedarf auf- und zugeklappt werden.

Teil eins kümmert sich um das Anzeigen aller DataSets in einer Responsiv Komponente die "DataTable" genannt 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 zu sortieren. Dabei kann man zwischen ab- oder aufsteigend wählen und durch die verschiedenen Spalten einer Zeile eine Suche durchzuführen.

Abbildung 5.7: Nachrichten ändern - HomeDsWeb

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 roten Löschen Button wird das Nachrichtenpaket gelöscht falls es aktiv war wird es auch aus dem XIBO System entfernt. Die technische Umsetzung ist in der Sektion "Nachrichten-Pakete ändern - Technisch" nochmals genauer erklärt.

Der zweite Teil der "Datensatz ändern" Seite ist das Hinzufügen von Nachrichtenpaketen, dabei ist der Titel und der Inhalt der Nachricht natürlich erforderlich. Bei Nichteingeben vom Start Datum des Nachrichtenpakets wird davon ausgegangen das es sofort in den XIBO gespeichert werden soll und somit auf aktiv gesetzt wird. Andersrum, wenn kein Ablaufdatum eingegeben wird, wird das DataSet solange angezeigt, bis entweder ein Ablaufdatum im Nachhinein hinzugefügt oder es manuell einfach wieder gelöscht wird.

Nach jeder Aktion wird die Webseite mit einer Loading Animation versehen und der Nutzer bekommt dann ein Feedback ob die jeweilige Operation erfolgreich oder fehlgeschlagen ist. (siehe Abbildung 5.8)

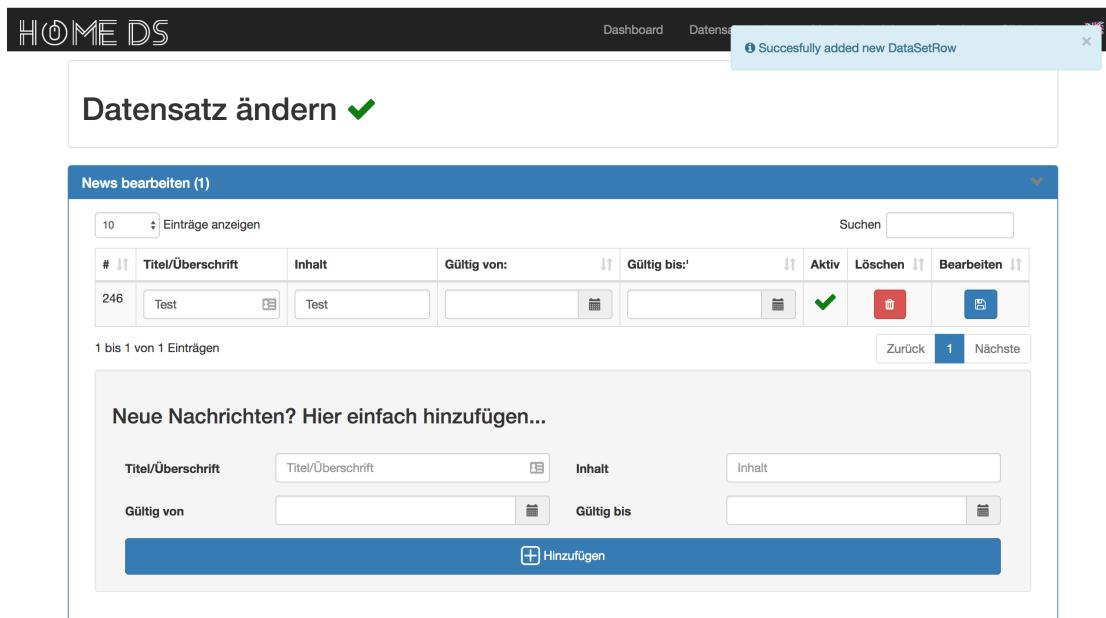


Abbildung 5.8: Feedback - HomeDsWeb

5.4.3 Medien abspielen - HomeDS Web

Eine weitere Funktion ist das Abspielen von Medien die im XIBO hinterlegt sind. Es soll dem Nutzer eine einfache und übersichtliche Oberfläche zur Verfügung gestellt werden, die Medien abspielt. Der Nutzer kann auf dieser Seite dann den Bildschirm auswählen auf den das gewünschte Video abgespielt werden sollen. Des Weiteren soll verhindert werden, dass der Nutzer stundenlang sein gewolltes Video aus der XIBO Bibliothek heraussuchen muss. Diese Herausforderung wurde mithilfe von Schlagworten und einer Schlagwort-Wolke gelöst. Auch werden nur Medien Formate wie Foto und Video aus der XIBO Bibliothek geladen.

Medien Bezeichnung	Datentyp	Abspielen
2007_roboChess.mp4	video	
4k_4_regions_whitegrey.png	image	

Abbildung 5.9: Medien abspielen - HomeDsWeb

Die Medien im XIBO müssen mit einem Schlagwort versehen werden. Diese Schlagwörter können die jeweilige Abteilung sein, also Informatik, Medientechnik oder Elektronik. Es kann sich aber auch um ein Projekt oder sonstiges handeln. Die Wahl der Schlagwörter bleibt dem Nutzer überlassen. (siehe Abbildung 5.9)

Mithilfe dieser Tags wollen wir dem Nutzer die Suche nach dem richtigen Video erleichtern. Der Nutzer bekommt die Möglichkeit einer dieser Tags auf der HomeDs Weboberfläche auszuwählen, daraufhin werden nur die Videos mit dem entsprechendem Tag angezeigt.

Nachdem der Nutzer sein Schlagwort und den gewünschten Bildschirm ausgewählt und auch schon das Video oder Foto gefunden hat, welches er anzeigen möchte, muss er nur den Play-Button drücken. Nach der Meldung, dass dieses Element erfolgreich abgespielt wurde, erscheint es innerhalb weniger Sekunden auf dem gewünschten Bildschirm. Das Video wird bis zum Ende abgespielt und danach wieder das vorangegangene Layout hergestellt. Wenn ein weiteres Element während dem Element eins abgespielt wird, wird das Video eins abgebrochen und das Element zwei abgespielt. Neben dem Abspielen gibt es auch die Stop Funktion. Diese soll sofort wieder das Layout, welches vor dem Abspielen des Videos angezeigt wurde, wiederherstellen.

5.4.4 Structure Crawler - HomeDS Web

Für das Verstehen des Aufbaus eines Layouts war es die Aufgabe einen sogenannten 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. Dabei kann die ID eines Layouts oder der Name eingegeben werden und der Crawler liefert dann alle enthaltenen Entitäten des jeweiligen Layouts im JSON Format. Dieser JSON kann mithilfe der "Copy to Clipboard" Funktion in die Zwischenablage kopiert werden.

Der Crawler ist eigentlich nur ein REST-Zugriff, der sich alle Layouts vom XIBO Server holt und dabei alle anderen Child-Elemente des Layouts mitschickt. Dies ist praktisch, denn es unterstützt beim Arbeiten mit vielen verschiedenen Elementen, die alle eindeutige Schlüssel besitzen.

The screenshot shows the HomeDS web application interface. At the top, there is a navigation bar with links for Dashboard, Datensatz ändern, Medien abspielen, Crawler, Slideshow, and a language switcher (UK). Below the navigation bar, there is a header with the text 'Crawler ✓'. Underneath this, there are two main sections: 'Configuration' and 'Output'. The 'Configuration' section contains fields for 'Layout ID' (set to 36) and 'Layout Name' (empty), with a large blue button labeled 'Crawl' below them. The 'Output' section contains a 'Copy to Clipboard' button and a 'See in formater' button. A large text area displays a JSON object representing the layout structure. The JSON starts with an array of regions, each containing a playlist with an ID of 149, no permissions, and a name of 'test-1'. It includes details about widgets (type: 'hls', duration: 60, options: 'mute'), audio (tempId: null), and video (width: 1920, height: 1080, zIndex: 0). There are also region options for region IDs 109 and 110, and a region for test-2 with a duration of 50,000ms, a loop option, and a transition direction. The JSON ends with a region for test-3 with a duration of 50,000ms and a transition type.

```
[{"owner": "lukas", "backgroundColor": "#000", "schemaVersion": 3, "modifiedDt": "2018-03-13 11:04:43", "regions": [{"playlists": [{"playlistId": 149, "permissions": [], "name": "test-1"}, {"playlistId": 149, "calculatedDuration": "60", "widgetId": 413, "displayOrder": "1", "isNew": false, "ownerId": 5, "type": "hls", "duration": 60, "widgetOptions": [{"widgetId": 413, "type": "attrib", "value": "109", "option": "hlsId"}, {"widgetId": 413, "type": "attrib", "value": "0", "option": "mute"}, {"widgetId": 413, "type": "attrib", "value": null, "option": "name"}, {"widgetId": 413, "type": "attrib", "value": "110", "option": "posterId"}, {"widgetId": 413, "type": "attrib", "value": "0", "option": "transparency"}], "displayOrder": null, "ownerId": 5, "layoutId": 36, "duration": "60", "top": "0.0000", "regionOptions": [{"regionId": 109, "left": "0.0000", "permissions": [], "name": "test-1", "width": "1920.0000", "tempId": null, "height": "1080.0000", "zIndex": 0}, {"playlistId": 149, "left": "1920.0000", "permissions": [], "name": "test-2", "displayOrder": "1", "ownerId": 5, "widgets": [], "tags": []}, {"regionId": 162, "permissions": [], "name": "test-2", "displayOrder": "1", "ownerId": 5, "layoutId": 36, "duration": "0", "top": "50.0000", "regionOptions": [{"regionId": 162, "value": "0", "option": "loop"}, {"regionId": 162, "value": "N", "option": "transitionDirection"}, {"regionId": 162, "value": null, "option": "transitionDuration"}, {"regionId": 162, "value": null, "option": "transitionType"}], "regionId": 162, "left": "50.0000", "permissions": [], "name": "test-3"}]}]
```

Abbildung 5.10: Crawler - HomeDsWeb

5.4.5 HomeDS Server Slideshow - HomeDS Web

Eine weitere Funktion des HomeDs Servers ist das Abspielen von Medien ohne diese in den XIBO hochladen zu müssen. Dabei handelt es sich um eine Website die Bilder aus dem lokalen Server Ordner mit einem Wechselintervall von 3 Sekunden anzeigt.

5.4.6 Internationalisierung - HomeDS Web

Die Weboberfläche der HomeDS Anwendung ist vollständig in Englisch und Deutsch übersetzt. Die Sprache kann ganz leicht mit einem Klick entweder auf die englische oder österreichische Flagge entsprechend geändert werden.



Abbildung 5.11: Internationalisierung

5.5 Funktionen des JavaEE - Technischer Hintergrund

Um den Benutzern die Möglichkeit zu geben die Funktionen des Servers zu nutzen, wurde auch eine Webapplikation erstellt. Die Entscheidung mit welcher Technologie die Weboberfläche programmiert werden soll ist schnell auf JSF gefallen. Diese hat nämlich viele Vorteile, wie zum Beispiel die Platzformunabhängigkeit und auch schnelle 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.

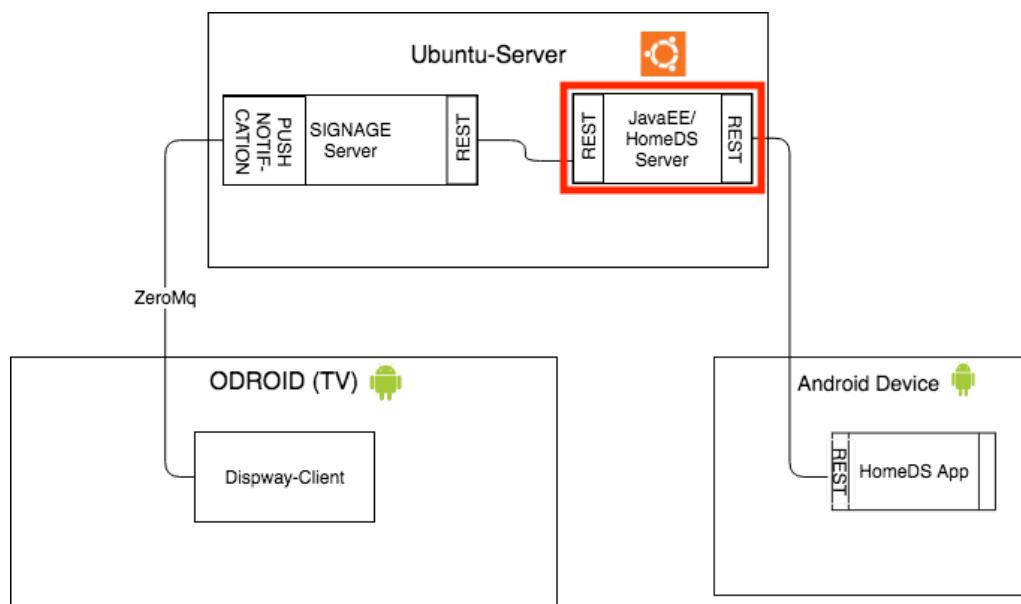


Abbildung 5.12: JavaEE - Technischer Hintergrund

5.5.1 Nachrichten-Pakete ändern - Technisch

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. Es wurde auch so programmiert, dass bei keinem Startdatum das Nachrichten Paket sofort angezeigt wird, oder wenn kein Enddatum festgelegt wird, dass es bis zum manuellen Deaktivieren angezeigt wird.

```
1 if (dataset.isActive() == false &&
     (dataset.getFromDate().minusDays(1)
      .isBefore(LocalDate.now())))
2 {
3     try {
4         //if succesfull added then set active true and add id
5         if ((id=dataSetApi.addDataSetField(dataset)) > 0) {
6             dataset.setDataRowId(id);
7             dataset.setActive(true);
8             dataSetFieldFacade.save(dataset);
9         }
10    } catch (NoConnectionException e) {
11        // Catch Exception
12    }
13 }
```

Code 5.1: public void doCheckEvery24Hours()

Die Überprüfung ob die DataSets aus der Server-Datenbank im Zeitintervall liegen, wird mithilfe der Java Annotation @Schedule jeden Tag um 01:00 Uhr durchgeführt. Falls dieses DataSet im Intervall liegt, wird dieses 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. Solange das DataSet sich im XIBO befindet wird es angezeigt. Nach dem Entfernen aus dem XIBO ist es auf dem Bildschirm nicht mehr sichtbar.

Falls keine Nachrichten-Pakete aktiv oder im System sind, werden aus dem RSS-Feed der HTL Leonding die Neuigkeiten anstatt der Nachrichten-Pakete angezeigt.

5.5.2 Medien abspielen - Technisch

Die Medien werden per REST aus dem XIBO abgefragt. Dabei wird schon der Datentyp und das Schlagwort gefiltert. Es werden auch die Bildschirme, die im XIBO angemeldet sind, abgefragt. Wenn der Nutzer den Bildschirm ausgewählt hat und bei einem der Medien auf den "Play Button" gedrückt hat, wird per REST im XIBO in einem Layout ein Widget vom Typ Bibliothek dem Medium zugewiesen. Falls dies

erfolgreich verlaufen ist, wird daraufhin dieses Layout wieder mittels REST in den Kalender vom ausgewähltem Bildschirm mit einer Priorität von 20 eingeplant. Prioritäten sind dazu da, um bei mehreren Events diese überschreiben zu können. Zum Beispiel wenn zwei Events gleichzeitig eingeplant sind, dann wird das Event, das eine höheren Priorität besitzt, angezeigt.

Das heißt, wenn gerade Nachrichten vom Sekretariat auf einem Bildschirm angezeigt werden und auf diesem Bildschirm dann Medien abgespielt werden, hat das "Medien abspielen" (Priorität 20) gegenüber dem "Nachrichten anzeigen" (Priorität 10) Vorrang und wird abgespielt. Beim Ende oder Beenden des "Medien abspielen" werden dann wieder die Nachrichten angezeigt.

5.5.3 JavaEE mit REST

Der Server stellt Funktionen wie Medien abspielen, Nachrichten Paket ändern, löschen, hinzufügen oder abfragen, Status abfragen und den Crawler per REST zur Verfügung.

Die REST Schnittstellen sind mithilfe von Swagger dokumentiert. (siehe Seite 46)

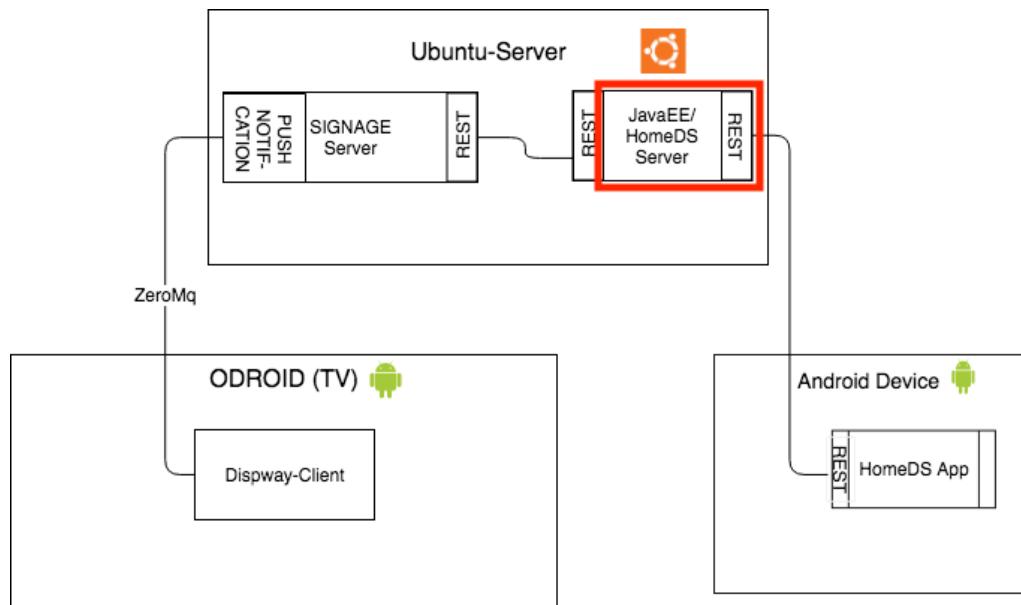


Abbildung 5.13: JavaEE REST-Service

5.5.4 Swagger

Swagger wird verwendet um Funktionalität und Möglichkeit einer API übersichtlich zu gestalten. Swagger dokumentiert welche Response Code's man erwarten kann, wie bei einem POST oder PUT der Body aussehen muss oder welche Pathparams mitgegeben werden müssen oder welche nur optional sind. Dabei gibt es zwei verschiedene Arten eine REST-Dokumentation zu erstellen.

The screenshot shows the Swagger UI interface for a GET request to the endpoint '/crawler'. The endpoint is described as 'Get all Crawled things'. There are two parameters listed: 'layoutId' (integer, query) and 'layout' (string, query). The 'Responses' section shows a successful response (Code 200) with the description 'successful operation'. An example value for the response is shown as 'string'.

Abbildung 5.14: GET Request Dokumentation

Die erste Möglichkeit 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.

The screenshot shows the JavaEE REST Service Swagger documentation. It is organized into sections:

- DataSetDataField - API** (with a dropdown arrow):
 - GET /datasetdatafield** Get all DataSetRows
 - DELETE /datasetdatafield/{dataid}/{datarowid}** Delete DataSetRow
 - POST /datasetdatafield/save** Save DataSetRow
 - PUT /datasetdatafield/edit** Edit DataSetRow
- Display - API** (with a dropdown arrow):
 - GET /display** Get Displays
- Media - API** (with a dropdown arrow):
 - GET /media/play** Play Media
 - GET /media** Get all Medias
- Status - API** (with a dropdown arrow):
 - GET /status** Get server Status

Abbildung 5.15: JavaEE REST- Service

Unsere Swagger-Dokumentation ist unter <http://vm59.html-leonding.ac.at:8080/homeds/swagger/swagger.html> oder 'BASE_URL:PORT/homeds/swagger/swagger.html' zu finden. [2]

5.5.5 Structure Crawler - Technisch

Der Crawler holt sich per REST vom XIBO ein Layout mit allen Sub-Entitäten. Ohne Parameter gibt XIBO aber nur das Model von Layout im JSON zurück. Um aber auch weitere Entitäten mit zu laden, musste mit dem Querparam "embed=" der REST Zugriff modifiziert werden. Dabei müssen mit einem Komma getrennt, die weiteren Entitäten angegeben werden, die mit eingebunden werden sollen.

```

1 HttpURLConnection con = new RequestHelper()
2         .executeRequest(RequestTypeEnum.GET, null,
3             new RequestHelper().BASE_URL +
4                 "api/layout?" +
5                 "embed=regions,playlists" +
6                 ",widgets,widgetOptions" +
7                 query, AuthenticationHandler.getTOKEN());

```

Code 5.2: Crawler GET Request

5.5.6 HomeDS Server Slideshow - Technisch

Bei der Slideshow Funktion müssen mittels einem FTP-Client die Bilder auf den vm59.htm-leonding.ac.at Server in den Ordner "/home/vwall/images/" hochgeladen werden. Danach werden die Medien aus dem Ordner gelesen und in einem Intervall von drei Sekunden nach der Reihe auf einer Website abgespielt. Bei dieser Funktion wurde plain JavaScript, HTML und CSS verwendet.

```
1 function carousel() {  
2     var x = document.getElementsByClassName("mySlides");  
3     for (i = 0; i < x.length; i++) {  
4         x[i].style.display = "none";  
5     }  
6     myIndex++;  
7     if (myIndex > x.length) {  
8         myIndex = 1  
9     }  
10    x[myIndex - 1].style.display = "block";  
11    setTimeout(carousel, 3000); //Change image every 3 seconds  
12 }
```

Code 5.3: Carousel JavaScript

Kapitel 6

Android Applikation

6.1 Einleitung

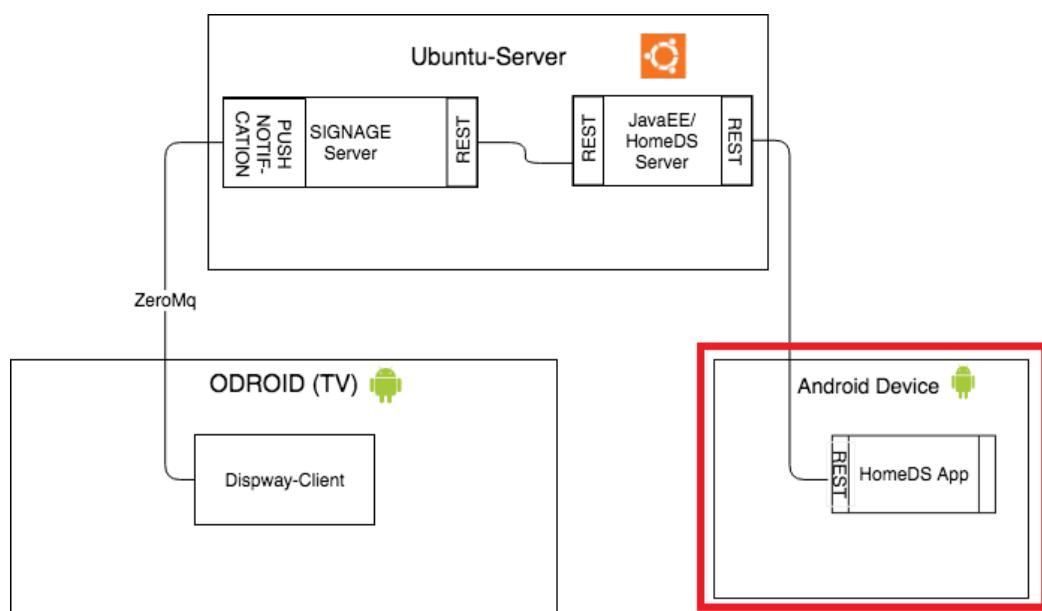


Abbildung 6.1: Die Android Applikation dient zur mobilen Benutzung

Um eine höchstmögliche Reichweite an Endgeräten zu erzielen, wurde eine Android Applikation entwickelt, mit der die wichtigsten Funktionen abgedeckt werden. Zum Beispiel das Wechseln der DataSets des XIBO-Servers. Wichtig ist es die Applikation möglichst einfach und leicht bedienbar zu gestalten, um auch Erstbenutzern

die Bedienung zu erleichtern. Prototyp für die aktuelle Applikation ist eine Anwendung für Android, die direkt mit dem Digital Signage Server kommuniziert. Es stellte sich heraus, dass das Steuern des Servers direkt über eine Applikation auf Android Basis viel zu umständlich ist. Darum wurde eine Browser Applikation entwickelt, welche über das "HomeDsBackend"(Java-EE-Server) kommuniziert, um das Funktionspektrum 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

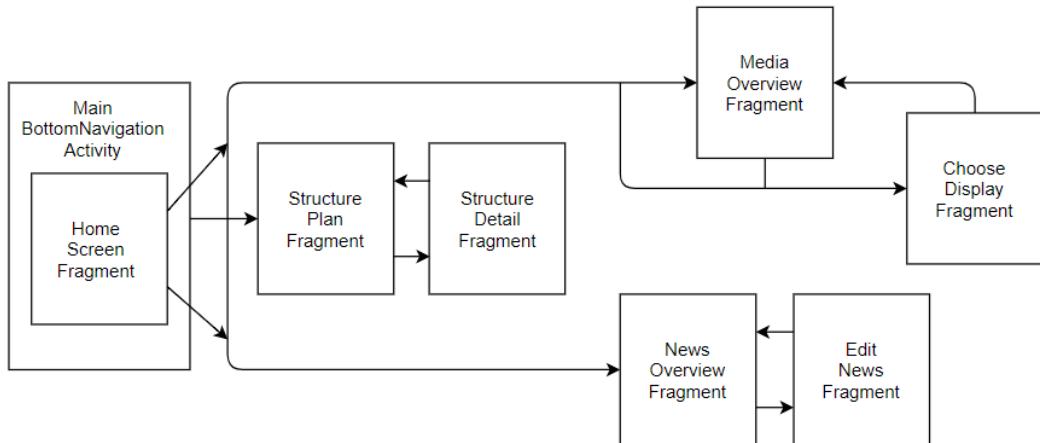


Abbildung 6.2: Fluss Diagramm der Android Anwendung

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 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.
- *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.

6.4 Benutzerhandbuch

6.4.1 Startseite

Auf der Einstiegsseite der Applikation ist ein Feld mit dem Serverstatus zu sehen. Sollte dieses Rot eingefärbt sein gibt es Verbindungsprobleme zum Server(die Applikation kann keine Daten vom Server erhalten oder Daten an den Server senden). Andernfalls kann mit der Benutzung fortgefahren werden, der Status wird grün angezeigt. Die Funktion der beiden Buttons wird im weiteren Verlauf des Benutzerhandbuchs erklärt.(Siehe Abbildung 6.3)

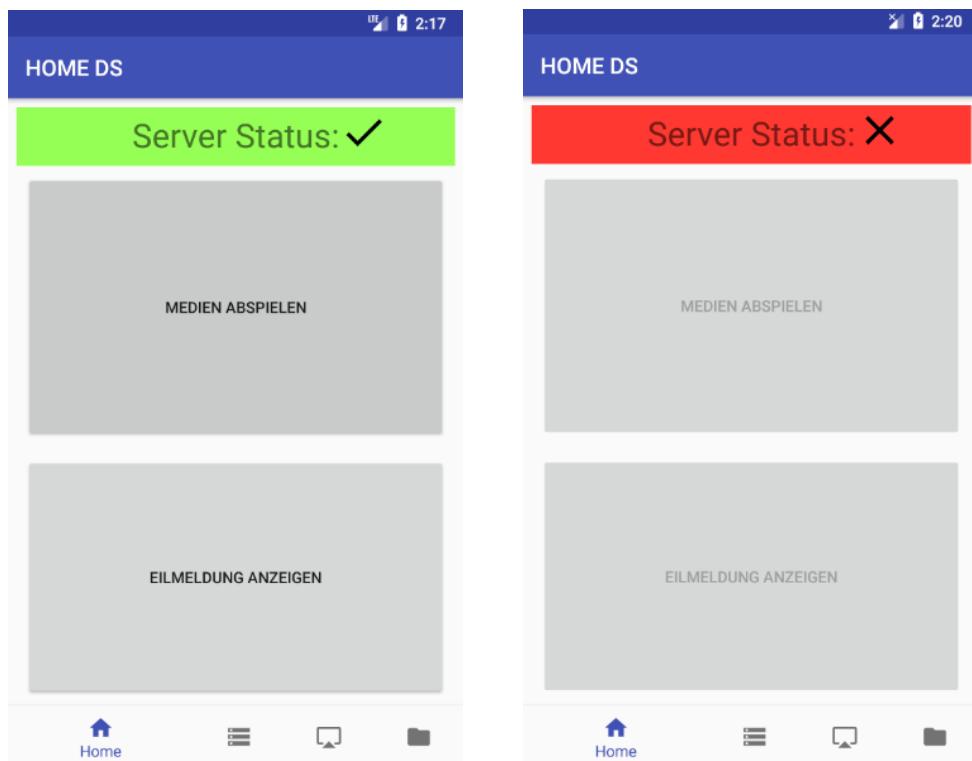


Abbildung 6.3: Startseite mit Status Element Online/Offline

6.4.2 Abspielen von Medien auf gewünschten Bildschirmen

Das Abspielen von Medien wird im Navigationstab "Play Media" abgewickelt. Zusätzlich befindet sich auf der Startseite ein Button über den direkt zu dieser Sektion navigiert werden kann.(Siehe Abbildung 6.26)



Abbildung 6.4: Startseite mit markierten Navigationselementen

Wurde noch kein Display gewählt, auf dem das Medium abgespielt werden soll, erscheint zu Beginn eine Auswahlmaske für die Bildschirme die verfügbar sind. Nach Auswahl einer Anzeige navigiert die Applikation automatisch zur Medienübersicht. (Siehe Abbildung 6.5)

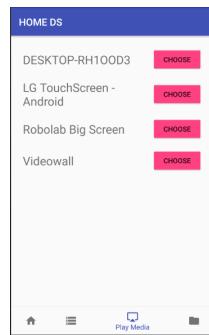


Abbildung 6.5: Liste mit verfügbaren Displays

Ist bereits ein Bildschirm ausgewählt wird direkt eine Übersicht der abspielbaren Medien angezeigt. Im rechten oberen Teil der Ansicht ist der gewählte Bildschirm zu sehen und über einen Button wird die Bildschirmauswahl angezeigt.(Siehe Abbildung 6.6)



Abbildung 6.6: Ausgewählter Bildschirm und Button für die Auswahl

Sortieren der angezeigten Medien ist über einen Spinner im linken oberen Teil der Medienübersicht möglich. Durch Berührungen der Anzeigefläche erscheint die Auswahl der Sortievorschläge.(Siehe Abbildung 6.7)

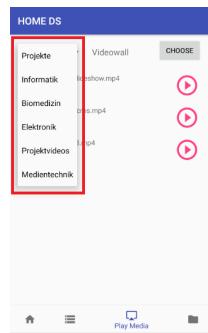


Abbildung 6.7: Spinner mit Tags zur Sortierung

Um ein in der Liste angezeigtes Medium abzuspielen, muss der Play-Button im rechten Teil des Listenelements gedrückt werden. Wird der Button betätigt spielt die gewählte Anzeige das Medium ab .(Siehe Abbildung 6.8)

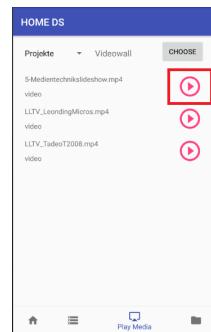


Abbildung 6.8: Play Button um Medium abzuspielen

6.4.3 Meldungen Anzeigen

Um eine Meldung anzuzeigen, muss zum "DataSet" Tab gewechselt werden. Dies ist auch über eine Schaltfläche auf der Startseite möglich. (Siehe Abbildung 6.9)



Abbildung 6.9: Startseite mit Navigationselementen

Die in der Ansicht zu sehende Liste enthält alle aktiven und inaktiven Meldungen, die auf den Anzeigen wiedergegeben werden beziehungsweise werden können. (Siehe Abbildung 6.10)

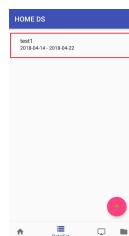


Abbildung 6.10: Am HomeDsBackend vorhandene DataSets

Durch auswählen eines der Listenelemente öffnet sich eine Detailansicht über die die Informationen der Meldung verändert werden können zum Beispiel der Titel oder der Anzeigezeitraum. Wird der FloatingActionButton im unteren rechten Teil der Ansicht gedrückt, so öffnet sich ein Detailfenster welches noch keine Daten eingetragen hat um eine neue Meldung zu erstellen. (Siehe Abbildung 6.11)

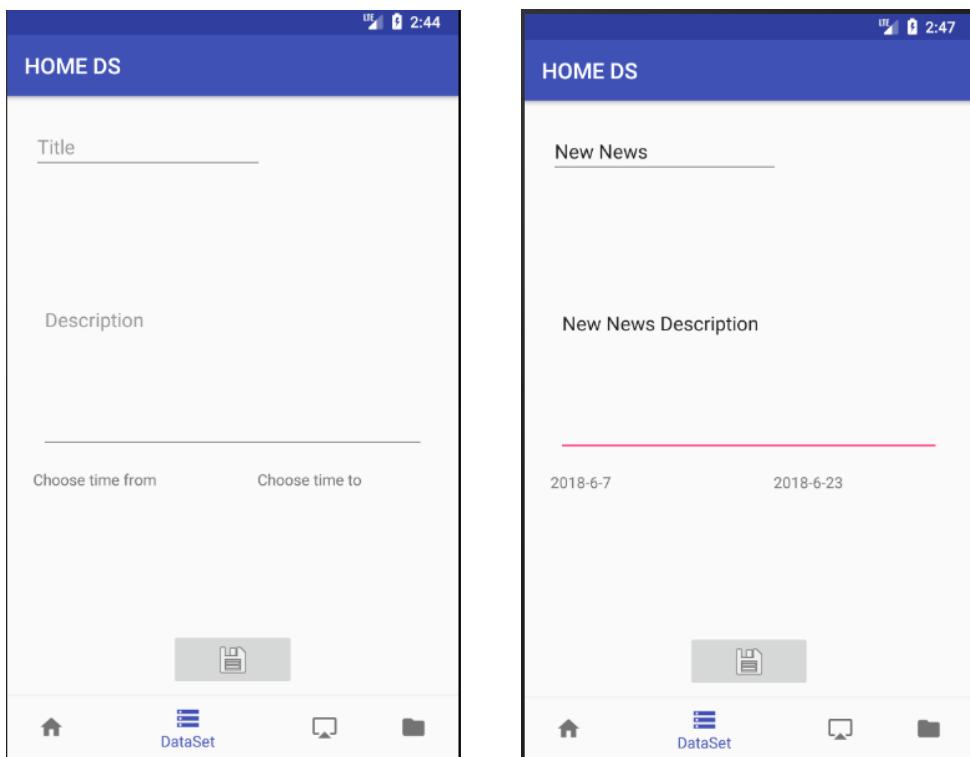


Abbildung 6.11: Anzeige für ein neues/zu veränderndes DataSet

Vorgenommene Änderungen oder neu erstellte Meldungen können mittels Save Button übernommen werden. Dieser befindet sich im unteren Teil der Detailansicht. Nach dem Speichern einer Meldung navigiert die Applikation wieder auf die Übersicht der Meldungen.(Siehe Abbildung 6.13)



Abbildung 6.12: Markierter Speichern Button

6.4.4 Strukturübersicht

Ist es nötig eine detaillierte Übersicht über die am Server liegenden Layouts zu bekommen, kann dies über den Navigationstab "Structure Plan" getan werden.(Siehe Abbildung 6.13)



Abbildung 6.13: Markierte Liste mit Layouts und Navigationstab

Die Liste zeigt alle am Server befindlichen Layouts. Durch Klicken einer der Listen-elemente navigiert die Applikation zu einer Detailansicht des Layouts.(Siehe Abbildung 6.14)

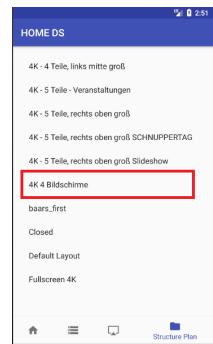


Abbildung 6.14: Markiertes Element um auf die Detailansicht zu gelangen

Die Detailansicht eines Layouts zeigt die Grunddaten eines Layouts wie zum Beispiel die ID oder die Besitzer ID. Des Weiteren werden alle Unterelemente beispielsweise Widget's angezeigt. Dargestellt wird der JSON-String des Layouts. (Siehe Abbildung 6.15)



Abbildung 6.15: Markierte Details des Layouts

6.5 MainBottomNavigationActivity und OverviewFragment

6.5.1 MainBottomNavigationActivity

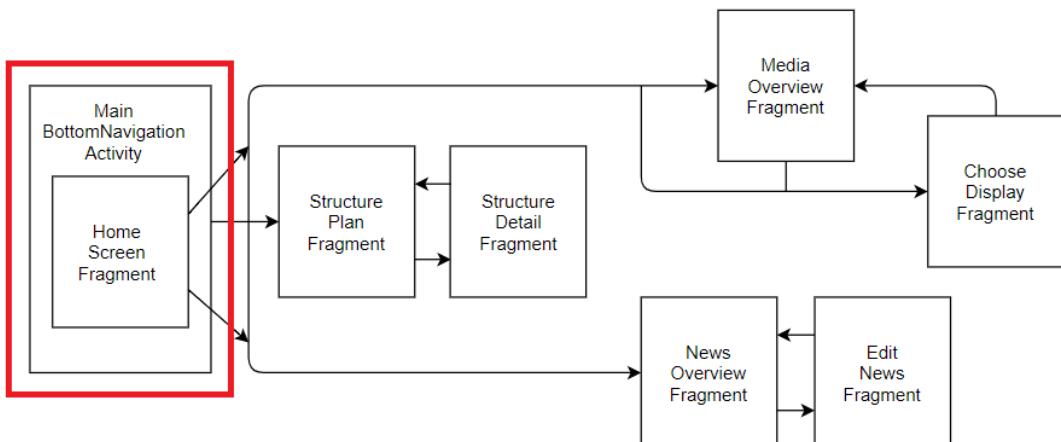


Abbildung 6.16: Stellung des MainBottomNavigationFragment in der Android Applikation

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 "con-

tainer_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. 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 anzeigen 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. Das Feld Display wird neu instantiiert. Die statische Variable "instance", vom Datentyp "MainBottomNavigationActivity", wird auf die aktuelle Instanz der Klasse zugewiesen. Die "BottomNavigationBar" wird mit Werten belegt und ein "OnNavigationItemSelectedListener" gesetzt, um durch die Applikation navigieren zu können.

```

1 setContentView(R.layout.
2     activity_main_bottom_navigation);
3 mainActivityBottomNavigation = this;
4 openHomeScreenFragment();
5 display = new Display();
6
7 navbar = findViewById(R.id.navigation_bar);
8 navbar.setOnNavigationItemSelected(
9     new BottomNavigationView.
10        OnNavigationItemSelected() {
11            @Override
12            public boolean onNavigationItemSelected(
13                @NonNull MenuItem item) {
14                item.setChecked(true);
15                switch (item.getItemId()) {
16                    case R.id.editDatasetNavBar:

```

```

17         openNewsOverview();
18         break;
19     case R.id.playMediaNavBar:
20         if (display.getDisplay() == null){
21             openChooseDisplayFragment();
22         }else {
23             openMediaOverviewFragment();
24         }
25         break;
26     case R.id.homeScreenNavBar:
27         openHomeScreenFragment();
28         break;
29     case R.id.structurePlanNavBar:
30         openStructurePlanFragment();
31         break;
32     }
33     return false;
34 }
35 });

```

Code 6.1: Zuweisung von Variablen und Navigationbar

getInstance

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

```

1 public static MainActivityBottomNavigation
2     getInstance() {
3         return mainActivityBottomNavigation;
4     }

```

Code 6.2: Getter für aktuelle Instanz der Activity

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 Mobilien Endgerätes) in der Applikation zuständig ist, hinzufügt. Manche Methoden übergeben zudem noch ein "Bundle" an das erstellte "Fragment", welches Objekte enthält die im nächsten

"Fragment" benötigt werden. Erkennungsmerkmal dieser Methoden ist das englische Verb "open" am Beginn des Methodennamens. Als Namensbeispiel hierfür wird die Methode "openNewsEditFragment" herangezogen.

```

1 public void openEditNewsFragment(DataSetDataField news) {
2     Bundle bundle = new Bundle();
3     bundle.putSerializable("data", news);
4     NewsEditFragment newsEditFragment =
5         new NewsEditFragment();
6     newsEditFragment.setArguments(bundle);
7     FragmentManager fm = getSupportFragmentManager();
8     fm.beginTransaction().replace(
9         R.id.container_display,
10        newsEditFragment, "actEdit")
11        .addToBackStack("actEdit").commit();
12 }
```

Code 6.3: Beispiel für Fragment-Austausch-Methode

6.5.2 HomeScreenFragment

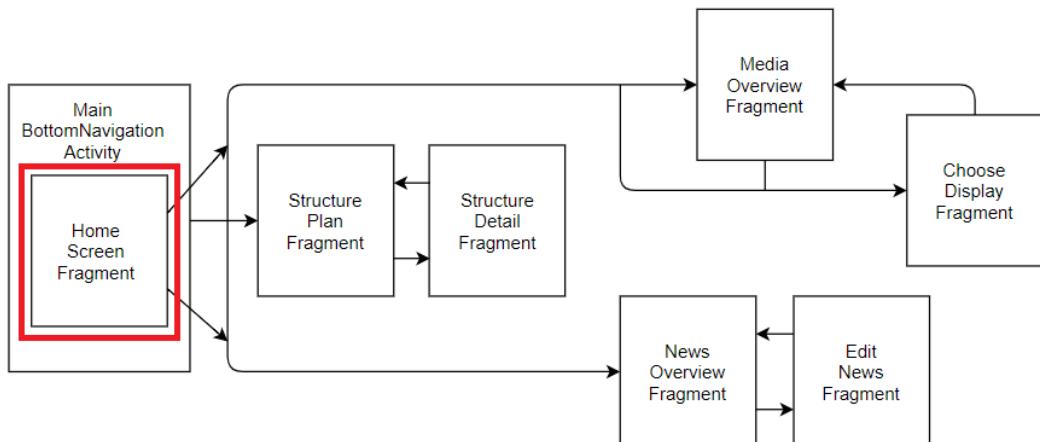


Abbildung 6.17: Stellung des HomeScreenFragment in der Android Applikation

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 "ConstraintLayout". 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.

```

1 Button btPlayMedia = v.findViewById(R.id.btPlayMedia);
2 Button btShowNews = v.findViewById(R.id.btShowNews);
3
4 btPlayMedia.setEnabled(false);
5 btShowNews.setEnabled(false);
6
7 ImageView ivUp = v.findViewById(R.id.ivServerUp);
8 ivUp.setVisibility(View.INVISIBLE);
9
10 ImageView ivDown = v.findViewById(R.id.ivServerDown);
11 ivDown.setVisibility(View.VISIBLE);
12
13 ConstraintLayout clServerStatus =
14     v.findViewById(R.id.cl_serverStatus);
15 clServerStatus.setBackgroundColor(
16     ContextCompat.getColor(
17         this.getContext(), R.color.serverDown));
18 }
```

Code 6.4: Initialisieren der Variablen im HomeScreenFragment

Im folgenden Schritt wird über einen "GET-Request" die Uhrzeit des XIBO-Servers über den Java-EE-Server abgefragt. Wird die Uhrzeit erhalten, so wird im "ConstraintLayout" der Server Status auf online gesetzt und das dazugehörige Bild angezeigt und die Buttons werden freigegeben. Erhält man keine Antwort bleibt die Anzeige unverändert.

```

1 rh.executeRequest(RequestTypeEnum.GET, null,
2     MainActivityBottomNavigation.getInstance()
3     .url + "/status/", () -> {
4         MainActivityBottomNavigation.getInstance()
5         .runOnUiThread(() -> {
6             if (rh.getResponseCode() == 200) {
7
8                 ivUp.setVisibility(View.VISIBLE);
9                 ivDown.setVisibility(View.INVISIBLE);
10
11                 clServerStatus.setBackgroundColor(
12                     ContextCompat.getColor(this.getContext(),
13                     R.color.serverUp));
14
15                 btPlayMedia.setEnabled(true);
16             }
17         });
18     });
19 }
```

```
16         btShowNews.setEnabled(true);
17
18     }else {
19         ivDown.setVisibility(View.VISIBLE);
20
21         clServerStatus.setBackgroundColor(
22             ContextCompat.getColor(this.getContext(),
23             R.color.serverDown));
24     }
25 }
26});
```

Code 6.5: Status Request im HomeScreenFragment

Um die Navigation über die erstellten Buttons zu ermöglichen wird, ein "OnClickListener" implementiert.

```
1 btPlayMedia.setOnClickListener(
2     new View.OnClickListener() {
3
4         @Override
5         public void onClick(View view) {
6             MainActivityBottomNavigation.
7                 getInstance().navbar
8                     .setSelectedItemId(R.id.playMediaNavBar);
9         }
10    });
11});
```

Code 6.6: OnClickListener für die direkte Navigation über Buttons im HomeScreenFragment

6.6 DataSet Verwaltung

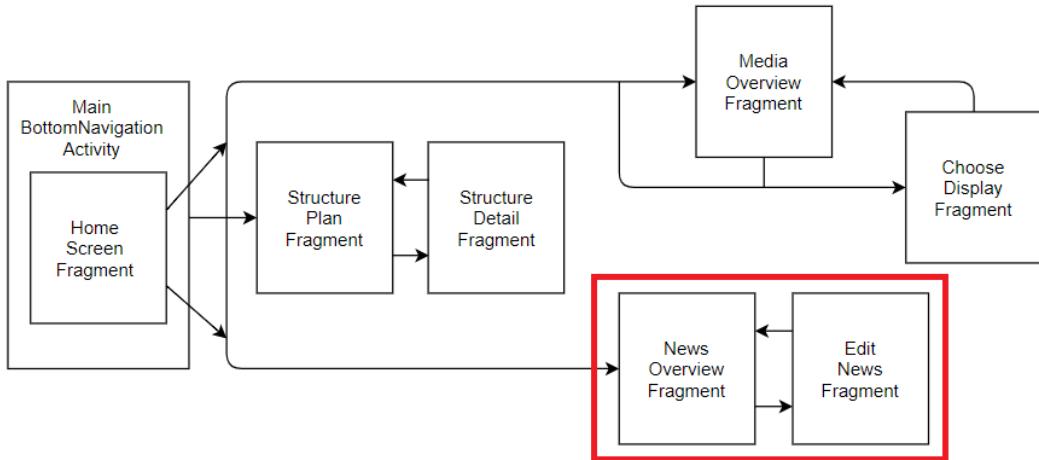


Abbildung 6.18: Stellung der DataSet Verwaltung in der Android Applikation

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.

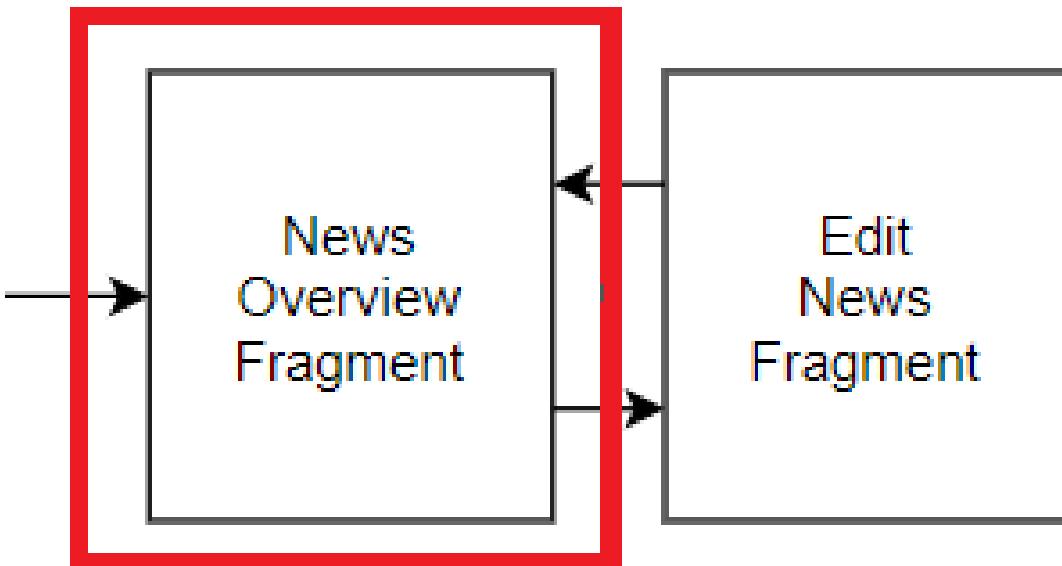


Abbildung 6.19: Stellung des NewsOverViewFragment in der Android Applikation

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 "FlaoingActionButton" erstellt und den zugehörigen Anzeige Elementen zugewiesen.

```

1 View v = inflater.inflate(
2     R.layout.fragment_news_overview, container, false);
3 RecyclerView rv = v.findViewById(R.id.rvNews);
4
5 FloatingActionButton fabAddNews =
6     v.findViewById(R.id.fabAddNews);
7
8 \end{listings}
9 Der ''FlaoingActionButton'', erh lt einen ''OnClickListener'',
10    ber diesen wird in der ''MainActivity'', eine Methode
11    aufgerufen die ein neues ''NewsEditFragment'', anzeigt, um ein
12    neues ''DataSet'', zu erstellen.
13 \begin{listings}[language=Java, caption={}
14 fabAddNews.setOnClickListener(
15     new View.OnClickListener() {
16         @Override
17         public void onClick(View view) {

```

```
15         MainActivityBottomNavigation.  
16             getInstance().  
17             openEditNewsFragment(  
18                 new DataSetDataField());  
19             }  
20         );
```

Code 6.7: Zuweisungen von Variablen und FloatingActionButton-OnClickListener des NewsOverviewFragment

Um in der "RecyclerView" Elemente anzeigen zu können werden diese über einen REST-Request an den Server abgefragt und der "RecyclerView" übergeben.

```
1 rh.executeRequest(RequestTypeEnum.GET, null,  
2     MainActivityBottomNavigation  
3     .getInstance().url  
4     + "/datasetdatafield/", () -> {  
5  
6         MainActivityBottomNavigation.getInstance()  
7         .runOnUiThread(() -> {  
8             //Belegung der angezeigten Elemente  
9         });  
10    });
```

Code 6.8: Anzeigen der abgefragten Elemente im NewsOverviewFragment

NewsEditFragment

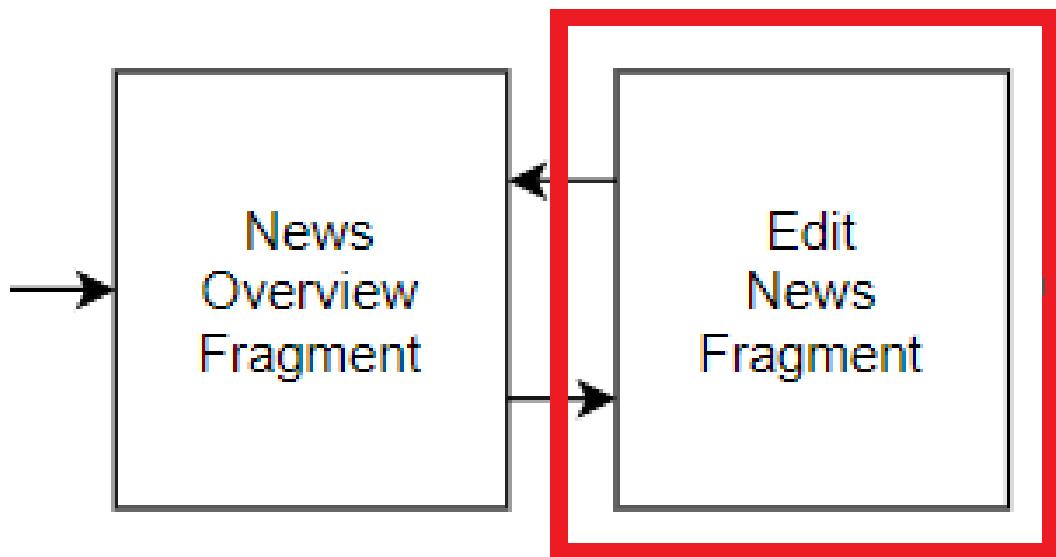


Abbildung 6.20: Stellung des NewsEditFragment in der Android Applikation

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 welchem 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

wird 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.

```

1 Bundle bundle = getArguments();
2 Log.d("BUNDLADATA", String.valueOf(bundle));
3 if (bundle != null){
4     this.setArguments(bundle);
5 }
6
7 DataSetDataField news =
8     (DataSetDataField) bundle.getSerializable("data");
9 ImageButton ibSaveNews =
10    v.findViewById(R.id.ibSaveNews);
11
12 title = v.findViewById(R.id.etTitle);
13 description = v.findViewById(R.id.etDescription);
14
15 title.setText(news.getTitle());
16 description.setText(news.getValue());
17
18 tvTimeFrom = v.findViewById(R.id.tvTimeFrom);
19 tvTimeTo = v.findViewById(R.id.tvTimeTo);

```

Code 6.9: Übernahme der übergebenen Werte und initialisieren der Variablen im NewsEditFragment

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 werden Hinweise auf die Eingabeoptionen im aktuellen Fragment angezeigt.

```

1 if (news.getToDate() != null
2     && news.getFromDate() != null) {
3     tvTimeFrom.setText(news.getFromDate().toString());
4     tvTimeTo.setText(news.getToDate().toString());
5     dateTo = news.getToDate();
6     dateFrom = news.getFromDate();
7 }

```

Code 6.10: Bedingung für Fragment-Recycling im NewsEditFragment

Die benötigten "onClickListener" für die Datum- und Uhrzeiteingaben werden im Anschluss implementiert und mittels "DatePickerDialog" beziehungsweise "TimePickerDialog" mit Daten versehen.

```

1 tvTimeTo.setOnClickListener(new View.OnClickListener() {
2     @Override
3     public void onClick(View view) {
4         final Calendar c = Calendar.getInstance();
5         year = c.get(Calendar.YEAR);
6         month = c.get(Calendar.MONTH);
7         day = c.get(Calendar.DAY_OF_MONTH);
8
9         DatePickerDialog datePickerDialog =
10            new DatePickerDialog(
11                MainActivityBottomNavigation.getInstance()
12                , new DatePickerDialog.OnDateSetListener() {
13                    @Override
14                    public void onDateSet(
15                        DatePicker datePicker,
16                        int year, int month, int day) {
17                        tvTimeTo.setText(year + "-"
18                                + (month + 1)
19                                + "-" + day);
20                        dateTo = LocalDate.
21                            of(year, month, day);
22                    }
23                }, year, month, day);
24                datePickerDialog.show();
25            }
26        });

```

Code 6.11: OnClickListener für Datum-Auswahl im NewsEditFragment

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 überprüft, 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.

```

1 if (news.getId() != null && news.getDataSetId()
2     != null && news.getDataRowId() != null) {
3     params.put("id", news.getId().toString());

```

```

4     params.put("dataSetId", news.getDataSetId().toString());
5     params.put("dataRowId", news.getRowId().toString());
6 }
7
8 if (news.getId() == null) {
9     Long n = -1L;
10    params.put("dataRowId", n.toString());
11    rh.executeRequest(RequestTypeEnum.POST, params,
12        MainActivityBottomNavigation
13        .getInstance().url + "/datasetdatafield/save/",
14        () -> {
15            //Toast ausgabe
16        });
17 } else {
18    rh.executeRequest(RequestTypeEnum.PUT, params,
19        MainActivityBottomNavigation
20        .getInstance().url + "/datasetdatafield/edit/",
21        () -> {
22            //Toast ausgabe
23        });
24 }

```

Code 6.12: Unterscheidung der Art der Anfrage im NewsEditFragment

6.7 Medioplayer

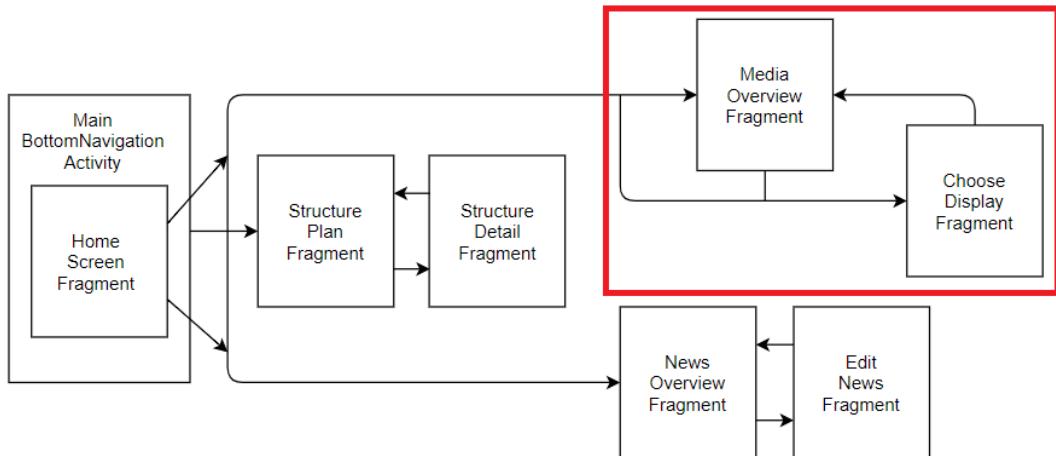


Abbildung 6.21: Stellung der Media Navigation in der Android Applikation

Um auf einer gewünschten Anzeige ein, sich auf dem Server befindliches, Medium abzuspielen, werden 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" geleitet, um einen Bildschirm 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 Display ausgewählt wird er zuerst auf das "ChooseDisplayFragment" weitergeleitet.

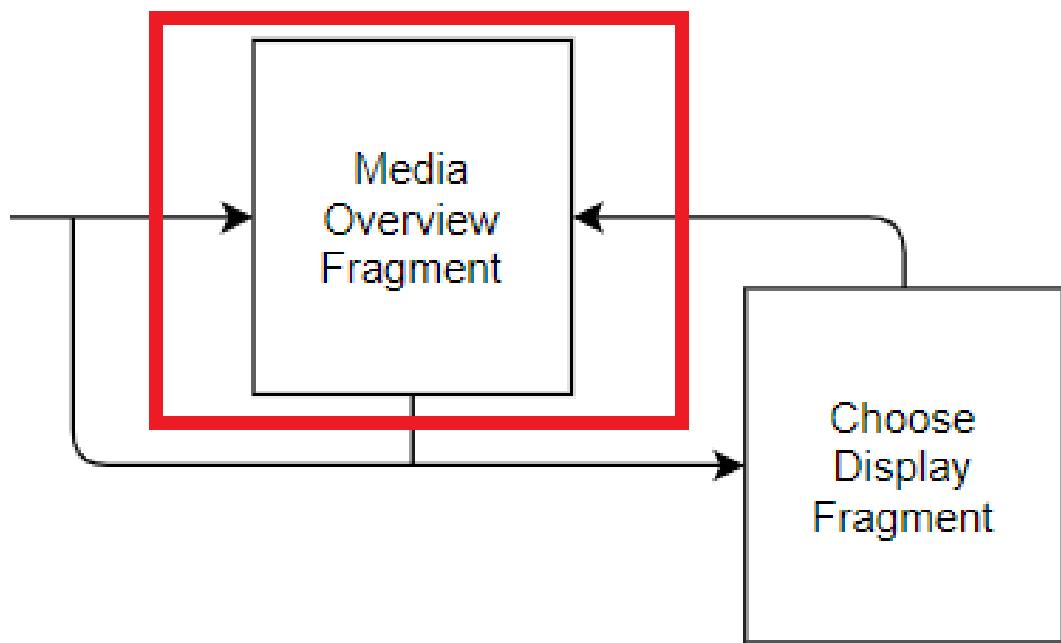


Abbildung 6.22: Stellung des MediaOverViewFragment in der Android Applikation

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. Zusätzlich werden die in der Klasse deklarierten Variablen mit den Anzeigeelementen verknüpft. Die TextView wird im Anschluss mit dem Namen des, für die Wiedergabe gewählten Displays, belegt.

```

1 View v = inflater.inflate(
2     R.layout.fragment_media_overview, container, false);
3 rvMedia = v.findViewById(R.id.rvMedia);
4 medias = new LinkedList<>();
5 spTagChoise = v.findViewById(R.id.spTagChoise);
6 btCooseDisplay = v.findViewById(R.id.btChooseDisplay);
7 tvDisplayToPlay = v.findViewById(R.id.tvDisplayToPlay);
8 tvDisplayToPlay.setText(
9     MainActivityBottomNavigation.getInstance()
10    .getDisplay().getDisplay());

```

Code 6.13: Instantiieren der benötigten Variablen im MediaOverviewFragment

Um die Liste der Medien zu sortieren, wird ein ArrayAdapter initialisiert. Diesem werden die Ressourcen "tag_array", beinhaltet eine Liste mit Sortermöglichkeiten 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 anzuzeigen, wird dem Spinner das erste Element des "tag_array" als Standardsortierung zugewiesen.

```

1 ArrayAdapter<CharSequence> tagAdapter =
2     ArrayAdapter.createFromResource(
3         MainActivityBottomNavigation.getInstance()
4             .getApplicationContext(), R.array.tag_array,
5             android.R.layout.simple_spinner_item);
6
7 tagAdapter.setDropDownViewResource(
8     android.R.layout.simple_spinner_dropdown_item);
9
10 spTagChoise.setAdapter(tagAdapter);
11 spTagChoise.setOnItemSelectedListener(
12     new AdapterView.OnItemSelectedListener() {
13         @Override
14         public void onItemSelected(AdapterView<?> adapterView,
15             View view, int i, long l) {
16             mediaTag = adapterView
17                 .getItemAtPosition(i).toString();
18             setRecyclerView(mediaTag, rvMedia);
19         }

```

```

20     @Override
21     public void onNothingSelected(AdapterView<?> adapterView)
22     {
23         spTagChoise.setSelection(0);
24         mediaTag =
25             adapterView.getItemAtPosition(0).toString();
26         setRecyclerView(mediaTag, rvMedia);
27     });
28     spTagChoise.setSelection(0);
29 mediaTag = spTagChoise.getItemAtPosition(0).toString();

```

Code 6.14: Erstellen des Spinner und der dazugehörigen Event-Listener MediaOverviewFragment

Um zwischen den einzelnen Displays zu wählen, wird dem "Auswählen" Button ein "onClickListener" zugewiesen welcher die Navigation zum "ChooseDisplayFragment" einleitet.

```

1 btCooseDisplay.setOnClickListener(new View.OnClickListener() {
2     @Override
3     public void onClick(View view) {
4         MainActivityBottomNavigation
5             .getInstance().openChooseDisplayFragment();
6
7         tvDisplayToPlay.setText(
8             MainActivityBottomNavigation.getInstance()
9                 .getDisplay().getDisplay());
10    }
11 });

```

Code 6.15: OnClickListener für Displayauswahl im MediaOverviewFragment

Des Weiteren wird 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.

```

1 HashMap<String, String> params = new HashMap<>();
2 params.put("start", "1");
3 params.put("length", "10");
4 params.put("tags", mediaTag);
5
6 rh.executeRequest(RequestTypeEnum.GET, params,
7     MainActivityBottomNavigation.getInstance()
8         .url + "/media/", () -> {
9             MainActivityBottomNavigation.getInstance()
10                .runOnUiThread(() -> {
11                    //Belegung der angezeigten Elemente

```

```
11     } );
```

Code 6.16: Anfordern und anzeigen der Daten vom Server im MediaOverviewFragment

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 "Auswählen" Button in einem der in der Liste angezeigten Display Elementen, wird dieser Display ausgewählt und Medien werden dann auf diesem abgespielt.

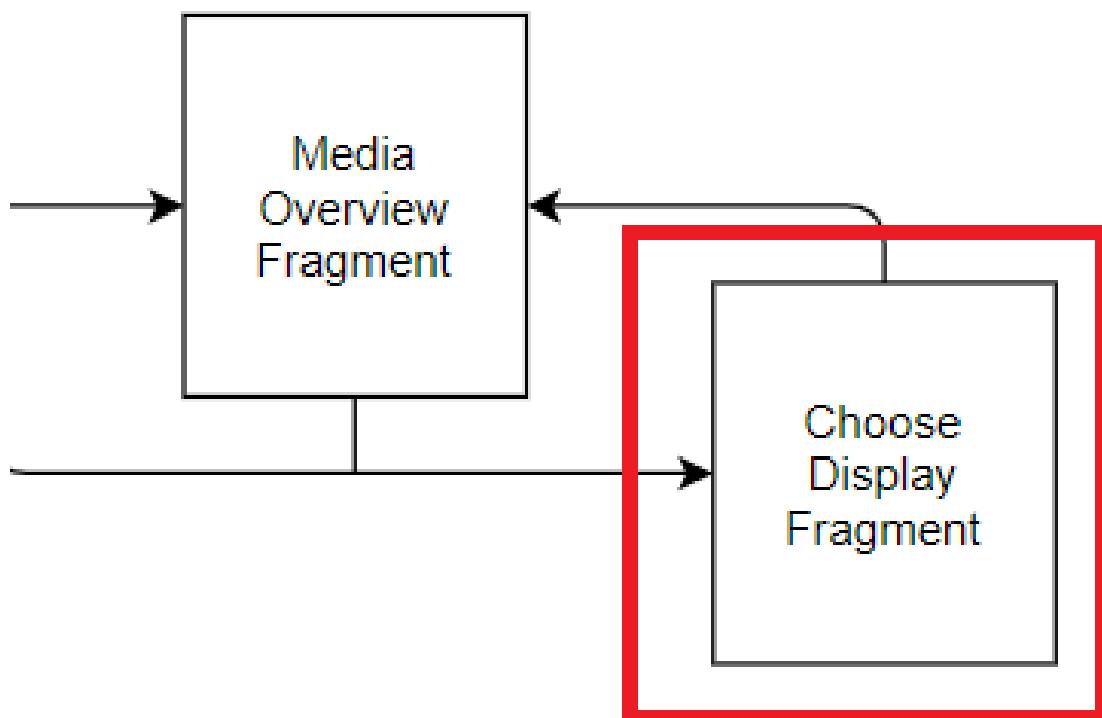


Abbildung 6.23: Stellung des ChooseDisplayFragment in der Android Applikation

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 anzuseigen.

```
1 rh.executeRequest(RequestTypeEnum.GET, null,  
                    MainActivityBottomNavigation.getInstance().  
2 url + "/display/", () -> {  
3     //Belegung der angezeigten Elemente  
4});
```

Code 6.17: Abfragen und anzeigen der Displays im ChooseDisplayFragment

6.8 Strukturplan

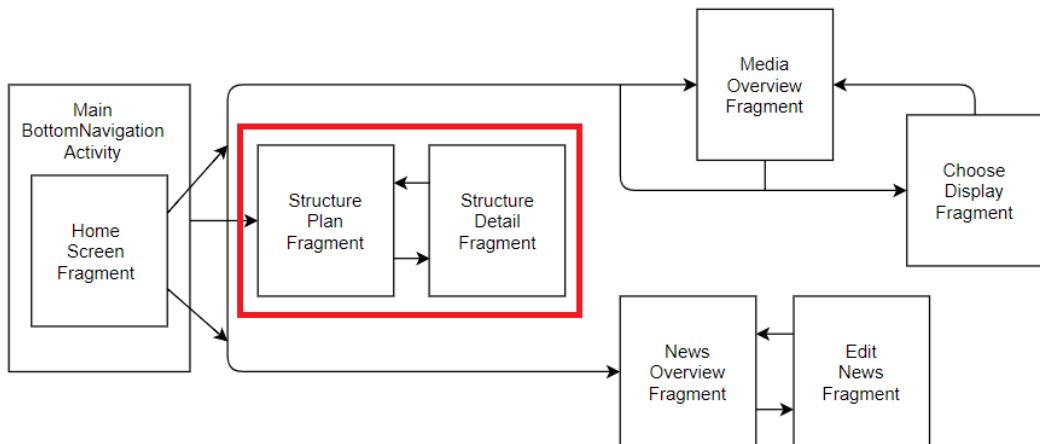


Abbildung 6.24: Stellung der Strukturverwaltung in der Android Applikation

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

tailansicht über das gewählte Layout.

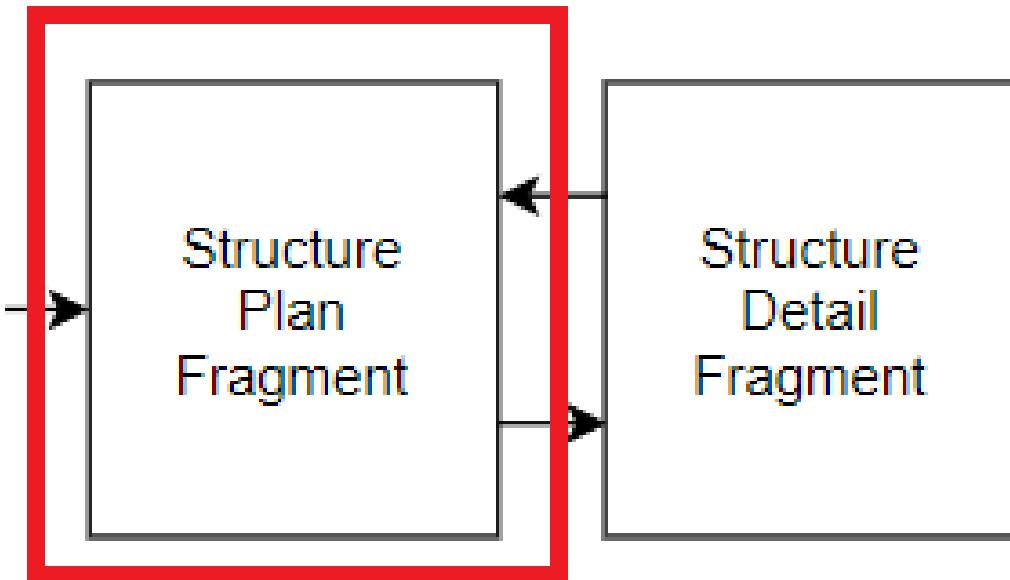


Abbildung 6.25: Stellung des StructurePlanFragment in der Android Applikation

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.

```
1 RecyclerView rvStructurePlan =  
    v.findViewById(R.id.rvStructurePlan);  
2 HashMap<String ,String > params = new HashMap<>();  
3 params.put("layoutId","-1");  
4  
5 rh.executeRequest(RequestTypeEnum.GET,params  
6   ,MainActivityBottomNavigation.getInstance()  
7   .url + "/crawler/", ()->{  
8     //Belegung der angezeigten Elemente  
9   }
```

Code 6.18: Abfrage und anzeigen der Daten im StructurePlanFragment

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.

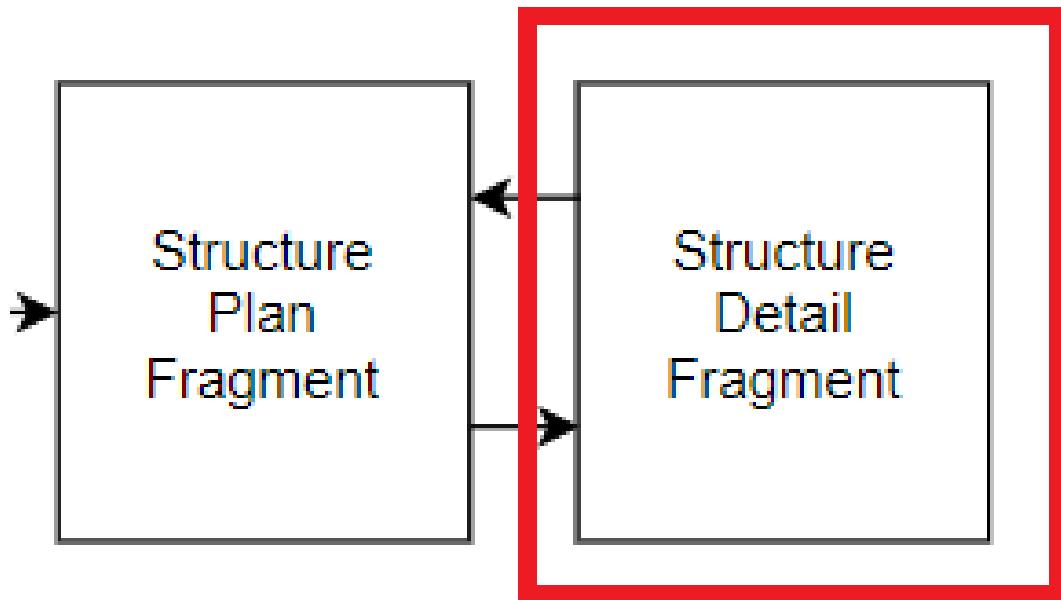


Abbildung 6.26: Stellung des StructureDetailFragment in der Android Applikation

In der Ressource Datei "fragment_structure_detail.xml" befindet sich eine "Text-View" als einziges Element. Die "onCreateView" Methode der Klasse "StructureDetailFragment", wird nach Zuweisen der Layout Ressource, der Struktur beschreibende "JSON-String" aus dem "Bundle" gelesen. Dies geschieht über die Methode "setArguments" der Basisklasse Fragment und deren Feld "mArguments", und dem Anzeigefeld. Anschließend wird dem Fragment noch die Fähigkeit gegeben sich Scrollen zu lassen und die View wird zurückgegeben.

```
1 View v = inflater.inflate(  
2     R.layout.fragment_structure_detail, container, false);  
3 Bundle bundle = this getArguments();  
4 TextView tvStructureDetailString =  
    v.findViewById(R.id.tvStructureDetailString);  
5
```

```

6 String detail = bundle.getString("actStructure");
7
8 tvStructureDetailString.setText(detail.toString());
9     tvStructureDetailString.setMovementMethod(
10         new ScrollingMovementMethod());

```

Code 6.19: Übernahme und anzeigen der Daten im StructureDetailFragment

6.9 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 QueryParameter in der URL einfügt. Beispielsweise:<URL>/layout?layoutID=78".

```

1 if (params != null && params.size() > 0) {

```

```

2     Iterator it = params.entrySet().iterator();
3     while (it.hasNext()) {
4         Map.Entry p = (Map.Entry) it.next();
5         urlBuilder.addQueryParameter(
6             p.getKey().toString(), p.getValue().toString());
7     }
8 }
```

Code 6.20: GET oder DELETE Request

Handelt es sich um eine POST oder PUT Anfrage, so werden die Key-Value Paare im Body mitgegeben und im Format application/x-www-form-urlencoded codiert.

```

1 String stringbody = "{}";
2 if (params != null && params.size() > 0) {
3     Iterator it = params.entrySet().iterator();
4     while (it.hasNext()) {
5         Map.Entry p = (Map.Entry) it.next();
6         stringbody += "\\" + p.getKey().toString() + ":" + p.getValue().toString() + ",";
7     }
8 }
13 stringbody = stringbody.substring(0, stringbody.length() - 1);
15 stringbody += "}";
16 body = RequestBody.create(
17 MediaType.parse("application/json")
18 , stringbody);
```

Code 6.21: POST oder PUT Request

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.

```

1 URL finalUrl = urlBuilder.build().url();
2 Log.i(LOGTAG, "FinalUrl: " + finalUrl.toString());
3 Request.Builder rb = new Request.Builder();
4 switch (executeType) {
5     case GET:
6         rb = rb.get();
7         break;
8     case PUT:
9         rb = rb.put(body);
10        break;
11    case POST:
12        rb = rb.post(body);
```

```

13         break;
14     case DELETE:
15         rb = rb.delete();
16         break;
17     }

```

Code 6.22: Zuweisung der Art der Anfrage

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.

Der letzte Schritt ist 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 Objekt vom Typ 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.

```

1 public int getResponseCode() {
2     return responseCode;
3 }
4
5 public void setResponseCode(int responseCode) {
6     this.responseCode = responseCode;
7 }
8
9 public String getResponseBody() {
10    return responseBody;
11 }
12
13 public void setResponseBody(String responseBody) {

```

```
14     this.responseBody = responseBody;  
15 }
```

Code 6.23: Getter und Setter für erhaltene Daten

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 funktioniert 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.

```
1 MainActivityBottomNavigation.getInstance().runOnUiThread(() ->{  
2     LinkedList<JSONObject> structureparts = new LinkedList<>();  
3     JSONArray jsonArray = null;
```

Code 6.24: Erstellen der benötigten Variablen

Das Array wird hierbei zum Durchlaufen benötigt. Die LinkedList bildet die Quelle für die Anzeigedaten des Fragments. Da JSON-Strings im folgenden Abschnitt konvertiert werden ist ein try-catch block nötig, in dem mögliche Fehlerfälle behandeln und ausgegeben werden können. Die übermittelten Daten werden über den "responseBody" Getter ausgelesen und in das JSONArray übernommen. Über Iteration durch das JSONArray werden die einzelnen JSONObjecte ausgewählt und der LinkedList angehängt.

Kapitel 7

Continuous Integration

7.1 Einleitung

Während des Verlaufes der Diplomarbeit wurde es notwendig unsere Applikation auf einen Server der HTL Leonding zu deployen. Genauer gesagt musste das JavaEE Backend auf einen Application Sever deployed werden. Um jedoch nicht immer per Hand bei der kleinsten Änderung das Projekt neu zu compilieren und dann manuell zu deployen, wurde entschieden diesen Prozess mithilfe von continuous integration zu optimieren.

7.2 Was ist CI?

Continuous integration (CI) ist ein Teil der modernen Software Entwicklung. CI stellt den Prozess dar, der das Bauen und Testen einer Anwendung abbildet. Mit Hilfe von CI lassen sich Fehler schneller finden und beheben. Die Idee der kontinuierlichen Integration ist es, dass die Entwickler frühzeitig und regelmäßig Änderungen in das Versionsmanagement einchecken. Diese Änderungen sollten funktionsfähig sein, sodass die gesamte Applikation auf Integrationsprobleme geprüft werden kann.

Es ist somit die Verfügbarkeit einer lauffähigen Version gegeben, die dann zum Beispiel für anderweitige Testzwecke oder Vertriebszwecke genutzt werden kann. Eine typische Anwendung sind sogenannte Nightly Builds, bei denen zu einer vorgegebenen Uhrzeit der aktuelle Programmcode übersetzt wird und dabei Tests mit der erstellten Software automatisch ausgeführt werden. Bei gefundenen Problemen

kann ein Entwickler dann zum Beispiel direkt per Mail über das gefundene Problem informiert werden.

7.3 Wieso CI?

Continuous integration hat das Ziel, die Qualität der Software über permanente Integration ihrer einzelnen Bestandteile zu steigern. Statt die Software nur in sehr großen Zeitabständen kurz vor der Auslieferung zu erstellen, wird sie in kleinen Zyklen immer wieder erstellt und getestet. Es ist auch ein Zeitgewinn vorhanden da nicht nach jedem Zyklus die Software per Hand sondern auf Knopfdruck ausgeliefert und getestet wird.

7.4 Wie kann CI realisiert werden?

Folgende Tools kamen für diese Herausforderung in Frage:

Bamboo Bamboo ist ein continuous integration server von Atlassian, den Entwicklern von JIRA, Confluence and Crowd.

Travis CI

Travis CI ist ein open-source gehosteter, continuous integration Service, der sehr stark in GitHub integriert ist.

Jenkins

Jenkins ist ein webbasiertes Open Source continuous integration System. Es ist in Java geschrieben und plattformunabhängig. Die Basis von Jenkins unterstützt zahlreiche Werkzeuge darunter SVN, Ant, Maven sowie JUnit. Durch die Community können weitere Funktionen mithilfe von Plug-Ins hinzugefügt werden. Somit lässt sich Jenkins für jedes Projekt individuell anpassen. Auch für Projekte mit anderen Sprachen/Technologien wie zum Beispiel PHP, Ruby oder .NET ist Jenkins geeignet. Testwerkzeuge lassen sich mittels Plug-Ins über die intuitive Benutzeroberfläche integrieren. Builds können durch verschiedene Auslöser gestartet werden: zum Beispiel Änderung des CVS oder Zeitplan (zum Beispiel Nightly Builds). Nightly Builds sind besonders bei Open Source Projekten zu finden und bedeutet, dass die Applikation nachts gebaut und getestet wird.

Aufgrund der hohen Anpassungsmöglichkeit, der großen Community und der sehr

genauen Dokumentation, wird Jenkins als Tool für die continuous integration ausgewählt.

7.5 Jenkins installieren

Auf dem Ubuntu 16.04 Server wurde Jenkins mittels Console installiert und es musste das Jenkins Apt-Repository dem Server apt-repository mit folgendem Befehl: "wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -" und "sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'" hinzugefügt werden.

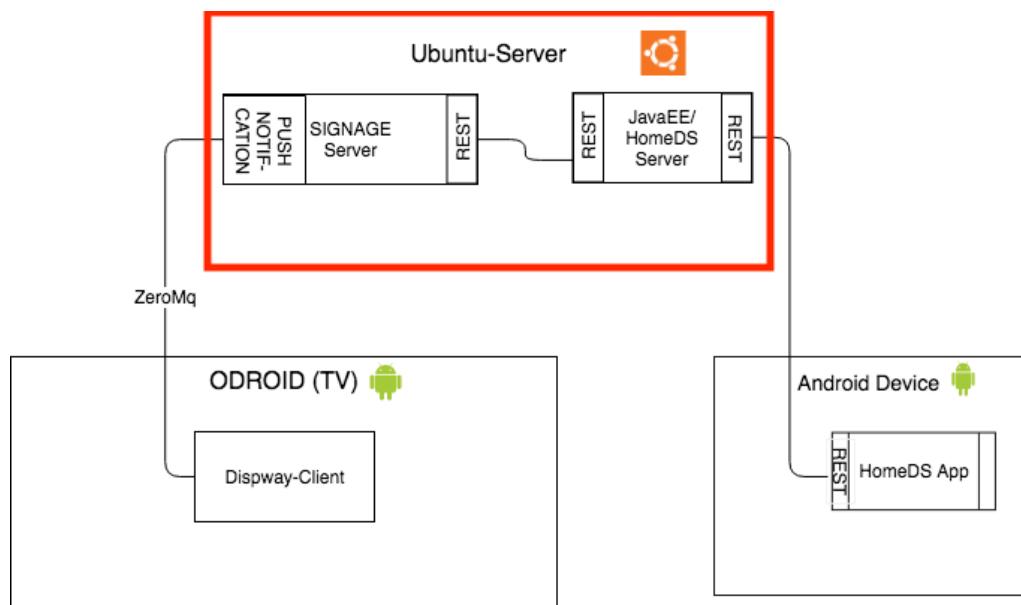


Abbildung 7.1: Jenkins - Ubuntu

Bevor Jenkins installiert wird, wird das Ubuntu Apt-Repository aktualisiert mit "sudo apt-get update" und dann kann Jenkins installiert werden mit "sudo apt-get install jenkins".

Nachdem der Installer fertig ist, wird wie in der Abbildung 7.2 ein initialAdmin-Password angezeigt. Dieses wird benötigt um die weiteren Schritte von der Jenkins Installation zu autorisieren.

```
*****
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

2bef3b1f67c54242b604c70be0a06df2

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*****
```

Abbildung 7.2: Konsolen Ausgabe - Jenkins

Nachdem das initialAdminPassword ausgegeben wurde, kann Jenkins unter der ServerUrl und dem Standard Port 8080 erreicht werden.

Jenkins entsperren

Um sicher zu stellen, dass Jenkins von einem autorisierten Administrator sicher initialisiert wird, wurde ein zufällig generiertes Passwort in das Log([wo ist das?](#)) und diese Datei auf dem Server geschrieben:

`/var/jenkins_home/secrets/initialAdminPassword`

Bitte kopieren Sie dass Passwort von einer dieser Quellen und fügen Sie es unten ein.

Administrator-Passwort

.....

Abbildung 7.3: Anmelden - Jenkins

Wie in 7.3 muss das Passwort jetzt eingegeben werden um die Installation fortzusetzen. In den nächsten Ansichten muss ein Admin User angelegt werden und die benötigten Plug-Ins für den Jenkins ausgewählt werden. Hier reichen die Standard Plug-Ins völlig aus.

Jetzt kann Jenkins verwendet werden.

7.6 Jenkins CI konfigurieren

Um jetzt eine JavaEE Anwendung mithilfe vom Jenkins auf einen Wildfly Application Server zu deployen, müssen vorher noch die Environment Variablen, JDK's, Maven und der Application Server auf dem Server konfiguriert werden. Die Environment Variablen, JDK's und Maven werden vom Jenkins eingerichtet. Dann muss nur in den Einstellungen die jeweilige Version konfiguriert werden. Zusätzlich es muss ein Oracle Login hinterlegt werden, damit sich Jenkins das Java JDK herunterladen kann und das GitHub Konto in welchem sich die Repositories befinden. Jedoch muss der Wildfly manuell heruntergeladen und konfiguriert werden.

Nachdem alle Entwicklungskits installiert sind, kann nun eine Pipeline im Jenkins eingerichtet werden. Dabei wird im Jenkins ein FreeStyle Softwareprojekt erstellt. Des Weiteren muss das Repository und der Branch angegeben werden. Damit Builds von außerhalb von Jenkins ausgeführt werden können, muss die Option "Builds von außerhalb starten (z.B. skriptgesteuert)", aktiviert und ein Authentifizierungstoken eingetragen werden. Das Builden ist das Erstellen einer neuen Version von einem Programm. Damit nach jedem "Commit" ein neuer Build gestartet wird muss die Option "GitHub hook trigger for GITScm polling" ebenfalls aktiviert werden. Bei der Funktion "Build" muss der Pfad zur POM des Servers angegeben und welches Maven-Goal ausgeführt werden soll, also 'install'. Mit dem "install" Maven-Goal werden direkt beim Build die Unit Tests ausgeführt.

Nachdem "Builden" muss die fertige ".war" File auf dem Wildfly Application Server deployed werden. Dieses Skript wird nach dem erfolgreichem Build vom Jenkins ausgeführt. Das Skript verwendet die Jboss-CLI. Dabei muss man sich auf den Application Server verbinden und den Befehl "deploy" ausführen, mit dem Pfad zur ".war" File. (siehe Code Beispiel 7.1)

```
1 bash /home/vwall/wildfly/bin/jboss-cli.sh
      --controller=vm59.htl-leonding.ac.at:9990 --connect -u=USER
      -p=PASSWORD --command="deploy --force
      /var/lib/jenkins/workspace/HomeDsSystems_Backend/
2   HomeDsBackend/target/homeds.war"
```

Code 7.1: Deploy war-File

Somit ist die Konfiguration des "FreeStyle" Projektes fertig.

Jenkins

Maven-Projekt HomeDsSystems_Backend

Zurück zur Übersicht

Status

Änderungen

Arbeitsbereich

Jetzt bauen

Maven-Projekt löschen

Konfigurieren

Module

GitHub Hook Log

Embeddable Build Status

Build-Verlauf

Trend

suchen

#57 27.03.2018 23:36

#56 27.03.2018 23:35

#55 27.03.2018 01:16

#54 26.03.2018 20:07

#53 26.03.2018 20:06

#52 21.03.2018 00:57

Beschreibung hinzufügen

Projekt deaktivieren

Arbeitsbereich

Letzte Änderungen

Letztes Testergebnis (Kein Test fehlgeschlagen.)

Letztes Testergebnis (Kein Test fehlgeschlagen.)

count

Trend der Testergebnisse

(Nur Fehlschläge anzeigen)

Vergroßern

Permalinks

- Letzter Build (#57), vor 1 Tag 13 Stunden
- Letzter stabiler Build (#57), vor 1 Tag 13 Stunden
- Letzter erfolgreicher Build (#57), vor 1 Tag 13 Stunden
- Neuester abgeschlossener Build (#57), vor 1 Tag 13 Stunden

vm59.html-leonding.ac.at:9090/job/HomeDsSystems_Backend/47/testReport/

Abbildung 7.4: Dashboard - Jenkins

Nun wird nach jedem "Commit" auf den "master" Branch ein Build ausgeführt. Dabei kann ein Build entweder erfolgreich, instabil oder fehlgeschlagen sein. Nur bei erfolgreichen Builds wird die ".war" File deployed.

Build #57 (27.03.2018 23:36:05)

Changes
1. changes added ([detail](#) / [githubweb](#))

Started by GitHub push by SakalAndrei

Revision: Odadfd203623e0584e45256819962cc231460ab10

- refs/remotes/origin/master

Testergebnis (Kein Test fehlgeschlagen.)

Modul-Builds

HomeDsBackend 11 Sekunden

Abbildung 7.5: Erfolgreicher Build - Jenkins

Bei instabilen Builds, sind entweder Bibliotheken veraltet oder die Unit Tests fehlgeschlagen.

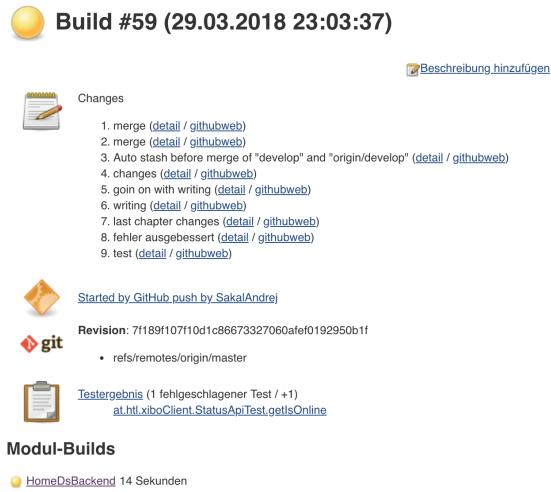


Abbildung 7.6: Instabiler Build - Jenkins

Bei einem fehlgeschlagenen Build, ist das Programm nicht kompilierbar.

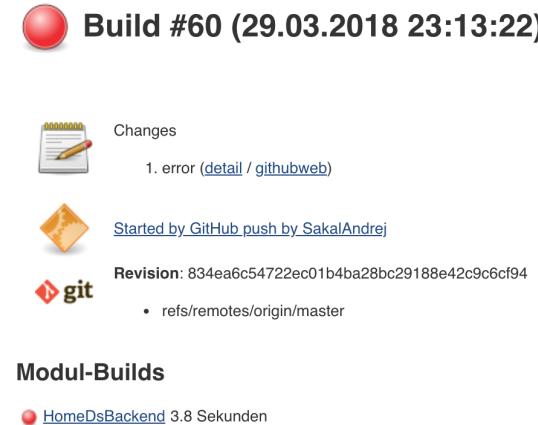


Abbildung 7.7: Fehlgeschlagener Build - Jenkins

Kapitel 8

Arbeitsaufteilung

8.1 Arbeit von Sakal Andrej

Sakal Andrej's Arbeit im Zuge der Diplomarbeit bestand darin, den Teil des JavaEE Servers zu übernehmen. Er kümmerte sich um die gesamte serverseitige Anwendung, diese enthält einen REST-Service, die Kommunikation mit der MySQL-Datenbank und eine Java Server Faces Weboberfläche. Zusätzlich kümmerte er sich auch um die continuous integration, die mithilfe von Jenkins realisiert wurde.

8.2 Arbeit von Hofmann Felix

Hofmann Felix's Teil im Zuge der Diplomarbeit bestand darin den gesamten Teil der Android Applikation zu übernehmen. Er kümmerte sich um die REST-Zugriffe, die Authentifizierung und das Design von der Android Applikation.

8.3 Genaue Aufteilung der Arbeit

Die folgende Tabelle zeigt eine genaue Übersicht der Arbeitsaufteilung der schriftlichen Arbeit. Hier werden die Kapitel den Arbeiter genau zugeteilt. Dabei werden die Bearbeiter mithilfe ihrer Initialen abgekürzt (SA = Sakal Andrej, HF = Hofmann Felix).

Kapitel	Bearbeiter
1 Einleitung	
1.1 Ausgangssituation	HF
1.2 Problemstellung	SA
1.3 Aufgabenstellung	HF
1.4 Ziele	SA
2 Digital Signage & XIBO	
2.1 Was ist Digital Signage?	SA
2.2 Digital Signage Anwendungen	SA
2.3 Was ist XIBO?	SA
2.4 Weboberfläche des XIBO	SA
2.5 Designen mit XIBO	SA
3 XIBO-Server	
3.1 Beschreibung	HF
3.2 API	HF
3.3 Authentifizierung	HF
4 Verwendete Technologien	
4.1 Git und GitHub	HF
4.2 Android	HF
4.3 Java Enterprise Edition	SA
4.4 JSF - Java Server Faces	HF
4.5 IntelliJ IDEA	HF
4.6 Android Studio	HF
4.7 Draw IO	SA
5 HomeDS - Server	
5.1 Einleitung	SA
5.2 Anforderungen an den HomeDS Server	SA
5.3 Komponenten des HomeDS Server	SA
5.4 Funktionen des JavaEE	SA
5.5 Funktionen des JavaEE - Technischer Hintergrund	SA
6 Android Applikation	
6.1 Einleitung	HF
6.2 Anforderungen	HF
6.3 Struktur	HF
6.4 Benutzerhandbuch	HF
6.5 MainBottomNavigationActivity und OverviewFragment	HF
6.6 DataSet Verwaltung	HF
6.7 Medioplayer	HF
6.8 Strukturplan	HF
6.9 Request-Helper	HF

7 Continuous Integration	
7.1 Einleitung	SA
7.2 Was ist CI?	SA
7.3 Wieso CI?	SA
7.4 Wie kann CI realisiert werden?	SA
7.5 Jenkins installieren	SA
7.6 Jenkins CI konfigurieren	SA
8 Arbeitsaufteilung	
8.1 Arbeit von Sakal Andrej	SA
8.2 Arbeit von Hofmann Felix	SA
8.3 Genaue Aufteilung der Arbeit	SA

Literaturverzeichnis

- [1] Xibo Open Source Digital Signage (abgerufen am 26.03.2018) URL: <https://xibo.org.uk/>
- [2] Swagger Dokumentation (abgerufen am 09.03.2018) URL: <https://swagger.io/docs/>
- [3] Postman Dokumentation (abgerufen am 12.02.2018) URL: <https://www.getpostman.com/docs/v6/www.getpostman.com/docs/v6/>
- [4] OAuth2 Official Website (abgerufen am 15.03.2018) URL: <https://oauth.net/2/>
- [5] okhttp3 Dokumentation (abgerufen am 17.01.2018) URL: <https://square.github.io/okhttp/3.x/okhttp/>
- [6] HttpURLConnection Dokumentation (abgerufen am 15.03.2018) URL: <https://developer.android.com/reference/java/net/HttpURLConnection.html>
- [7] Unterschied JavaEE und JavaSE (abgerufen am 28.03.2018) URL: <https://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html>
- [8] JavaEE (abgerufen am 21.03.2018) URL: https://de.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition
- [9] Java Server Faces (abgerufen am 28.03.2018) URL: https://de.wikipedia.org/wiki/JavaServer_Faces
- [10] IntelliJ IDEA (abgerufen am 27.03.2018) URL: https://de.wikipedia.org/wiki/IntelliJ_IDEA
- [11] Android Studio (abgerufen am 28.03.2018) URL: <https://developer.android.com/studio/features.html>
- [12] draw.io - Grafik Online-Tool (abgerufen am 01.03.2018) URL: <https://www.draw.io/>

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

Abbildungsverzeichnis

1.1	Prozess des Anzeigens	8
2.1	Digital Signage in der HTL Leonding	13
2.2	XBIO - Kalender	14
2.3	XIBO-Layout designen	15
2.4	Regions mit Widgets - XIBO	16
3.1	Systemkomponente XIBO-Server	20
3.2	Gliederung Ubuntu-Server	21
3.3	Grundeinstellungen einer XIBO-Anwendung	22
3.4	Mögliche Berechtigungen für eine XIBO-Anwendung	23
4.1	Github - Repository Network Diagramm	28
5.1	JavaEE - HomeDS	33
5.2	Systemarchitektur - HomeDS	34
5.3	Signage Server die Engine - HomeDS	35
5.4	Komponenten des HomeDS Server	36
5.5	Startseite - HomeDsWeb	37
5.6	Keine Verbindung zum XIBO - HomeDsWeb	38
5.7	Nachrichten ändern - HomeDsWeb	39
5.8	Feedback - HomeDsWeb	40
5.9	Medien abspielen - HomeDsWeb	41
5.10	Crawler - HomeDsWeb	42
5.11	Internationalisierung	43
5.12	JavaEE - Technischer Hintergrund	43
5.13	JavaEE REST-Service	45
5.14	GET Request Dokumentation	46
5.15	JavaEE REST- Service	47
6.1	Die Android Applikation dient zur mobilen Benutzung	51
6.2	Fluss Diagramm der Android Anwendung	52
6.3	Startseite mit Status Element Online/Offline	54

6.4	Startseite mit markierten Navigationselementen	54
6.5	Liste mit verfügbaren Displays	55
6.6	Ausgewählter Bildschirm und Button für die Auswahl	55
6.7	Spinner mit Tags zur Sortierung	56
6.8	Play Button um Medium abzuspielen	56
6.9	Startseite mit Navigationselementen	57
6.10	Am HomeDsBackend vorhandene DataSets	57
6.11	Anzeige für ein neues/zu veränderndes DataSet	58
6.12	Markierter Speichern Button	58
6.13	Markierte Liste mit Layouts und Navigationstab	59
6.14	Markiertes Element um auf die Detailansicht zu gelangen	59
6.15	Markierte Details des Layouts	60
6.16	Stellung des MainBottomNavigationFragemt in der Android Applikation	60
6.17	Stellung des HomeScreenFragmnet in der Android Applikation	63
6.18	Stellung der DataSet Verwaltung in der Android Applikation	66
6.19	Stellung des NewsOverViewFragment in der Android Applikation	67
6.20	Stellung des NewsEditFragment in der Android Applikation	69
6.21	Stellung der Media Navigation in der Android Applikation	72
6.22	Stellung des MediaOverViewFragment in der Android Applikation	73
6.23	Stellung des ChooseDisplayFragment in der Android Applikation	76
6.24	Stellung der Strukturverwaltung in der Android Applikation	77
6.25	Stellung des StructurePlanFragment in der Android Applikation	78
6.26	Stellung des StructureDetaiFragment in der Android Applikation	79
7.1	Jenkins - Ubuntu	88
7.2	Konsolen Ausgabe - Jenkins	89
7.3	Anmelden - Jenkins	89
7.4	Dashboard - Jenkins	91
7.5	Erfolgreicher Build - Jenkins	91
7.6	Instabiler Build - Jenkins	92
7.7	Fehlgeschlagener Build - Jenkins	92

Codebeispiele

3.1	Erstellen der Verbindung zum Server	24
3.2	Erstellen und senden des JSON-Body	24
3.3	Erhalt der Daten vom Server	25
3.4	Rückgabe des Token und schließen der Verbindung	25
5.1	public void doCheckEvery24Hours()	44
5.2	Crawler GET Request	47
5.3	Carousel JavaScript	48
6.1	Zuweisung von Variablen und Navigationbar	61
6.2	Getter für aktuelle Instanz der Activity	62
6.3	Beispiel für Fragment-Austausch-Methode	63
6.4	Initialisieren der Variablen im HomeScreenFragment	64
6.5	Status Request im HomeScreenFragment	64
6.6	OnClickListener für die direkte Navigation über Buttons im HomeScreenFragment	65
6.7	Zuweisungen von Variablen und FloatingActionButton-OnClickListener des NewsOverviewFragment	67
6.8	Anzeigen der abgefragten Elemente im NewsOverviewFragment	68
6.9	Übernahme der übergebenen Werte und initialisieren der Variablen im NewsEditFragment	70
6.10	Bedingung für Fragment-Recycling im NewsEditFragment	70
6.11	OnClickListener für Datum-Auswahl im NewsEditFragment	71
6.12	Unterscheidung der Art der Anfrage im NewsEditFragment	71
6.13	Instantiiieren der benötigten Variablen im MediaOverviewFragment	74
6.14	Erstellen des Spinner und der dazugehörigen Event-Listener MediaOverviewFragment	74
6.15	OnClickListener für Displayauswahl im MediaOverviewFragment	75
6.16	Anfordern und anzeigen der Daten vom Server im MediaOverview-Fragment	75
6.17	Abfragen und anzeigen der Displays im ChooseDisplayFragment	77
6.18	Abfrage und anzeigen der Daten im StructurePlanFragment	78
6.19	Übernahme und anzeigen der Daten im StructureDetailFragment	79
6.20	GET oder DELETE Request	80

6.21 POST oder PUT Request	81
6.22 Zuweisung der Art der Anfrage	81
6.23 Getter und Setter für erhaltene Daten	82
6.24 Erstellen der benötigten Variablen	83
7.1 Deploy war-File	90

HomeDS - Repository

<https://github.com/SakalAndrej/HomeDS>



CD-Datenträger



USB-Datenträger