Projekt Info

Das Projekt ist aus einem Kundenauftrag (AStA) als Teil eines Moduls an der HS-Esslingen entstanden.

Deployment

1. Server Setup
2. Auth Config
3. Domain & Start

FAQ / Common Issues

Siehe: FAQ

Configuration

- Formular Konfiguration
- PDF Export

Deployment - Setup Server

Aufsetzen eines Servers für ein Deployment

Diese Anleitung ist unter anderem aus dem Deployment auf einer VPS der BW-Cloud entstanden. Diese sind teilweise etwas vorkonfiguriert. Wichtige Konfigurationsschritte, welcher bereits erledigt sind, hier erwähnt, allerdings nicht im Detail beschrieben.

Betriebssystem

Diese Anleitung nimmt an, dass es sich bei dem System um eine neue VPS Instanz mit Debian 12 handelt. Das Setup ist sicherlich auch mit anderen Betriebssystemen möglich, jedoch können andere und oder weitere Schritte notwendig sein.

In jedem Falle wird empfohlen, ein aktuelles Betriebssystem tu verwenden.

Updates

```
Apt update
apt upgrade
```

Unattend Upgrades

Bei BW-Cloud per Default aktiviert

ssh login

Die Anmeldung mit Username und Passwort sollte unbedingt Verboten sein. Der Login sollte ausschließlich per SSH key möglich sein.

fail2ban

Um Angriffen zusätzlich entgegenzuwirken, sollte fail2ban verwendet werden.

`apt install fail2ban`

Die Konfiguration ist bei BW-Cloud bereits abgeschlossen. *(Individuell Prüfen)*

Rootless Docker

Installation vorn Docker Rootless siehe: https://docs.docker.com/engine/security/rootless/

Derzeit:

```
sudo apt-get install -y dbus-user-session

sudo apt-get install -y fuse-overlayfs

sudo apt-get install -y slirp4netns

apt-get install -y uidmap

curl -fsSL https://get.docker.com/rootless | sh

reboot
```

Testen mit

`docker run hello-world`

Echte IP's

```
 mkdir -p ~/.config/systemd/user/docker.service.d

 cd ~/.config/systemd/user/docker.service.d

 nano override.conf

 cat override.conf
[Service]
Environment="DOCKERD_ROOTLESS_ROOTLESSKIT_PORT_DRIVER=slirp4netns"
```

Compose

Sihe: https://docs.docker.com/compose/install/linux/#install-using-the-repository

Port Freigabe für nicht root user

add `net.ipv4.ip_unprivileged_port_start=0` to `/etc/sysctl.conf`

run: `sudo sysctl --system`

Install Git

für Continuous Deployment

`apt install git`

Clone Repo

```
cd ~
mkdir Git
cd Git/
git clone https://github.com/MaxTrautwein/AStA-Digital-Forms.git
```

Install nvm & node

see: https://github.com/nvm-sh/nvm?tab=readme-ov-file#installing-and-updating

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash
nvm install node
```

Install Java

Required for openapi-generator-cli

```
sudo apt install default-jre
```

openapi-generator-cl

Zum Generieren der Frontend-API Implementation

```
npm install @openapitools/openapi-generator-cli -g
```

allow ligering

für nicht root Nutzer. `sudo loginctl enable-linger $(whoami)`

Deployment - Auth Keycloak Config

General

Es wird davon ausgegangen, dass Keycloak verwendet wird. Solange pkce unterstützt wird, könnten allerdings auch andere Auth Backends verwendet werden. *Einstellungen für andere Auth Backends sind hier nicht näher beschrieben.*

Realm

Es ist ein `DigitalForms` Realm zu erstellen.

Client

In dem Erstellten Realm soll ein `df` client erstellt werden.

Client Config

Es muss auf `pkce` konfiguriert sein. Bei Keycloak ist das derzeit der Standard bei einem neuen client.

`Web origins` sollte auf die Frontend-Domain eingestellt werden. Eine `*` Einstellung ist möglich, stellt aber ein Sicherheitsrisiko dar.

Selbiges gilt für `Valid redirect URIs`

Deployment - Domain & Hosting

Allgemein

Die Verwendung eines Reverce Proxies, zur Sicherheit und Bereitstellung von HTTPS wird dringend empfohlen. *Hier wird dies am Beispiel von Traefik gezeigt.*

Die Script gegen von folgender Ordnerstruktur aus:

~ ~/Deployment ~/Deployment/AUTH_DOMAIN ~/Deployment/BACKEND_DOMAIN ~/Deployment/FRONTEND_DOMAIN ~/Deployment/docker-compose.yaml ~/Deployment/DeployFromGit.sh ~/Git/AStA-Digital-Forms/

Bei Abweichungen müssen Anpassungen gemacht werden

Erstellen von A und AAAA Records

Das Beispiel geht von 3 separaten Domains für Auth, Backend und Frontend aus.

Vorbereiten der Docker-Compose für Traefik

Template unter: docker-compose.yaml

Ersetze `{YOUR_DOMAIN}` mit deiner Domain.

DNS API

Im Beispiel wird die Hetzner API verwendet. Je nach Arbeiter muss `--certificatesresolvers.letsencrypt.acme.dnschall` angepasst werden der API-Key wird unter `../.key/api` relativ von der Docker-Compose erwartet.

DeployFromGit.sh Vorbereitung

Erstelle `FRONTEND_DOMAIN`, `AUTH_DOMAIN` und `BACKEND_DOMAIN`. Der Inhalt sollte die eigene vollständige Domain enthalten.

Lader DeployFromGit.sh herunter.

Note

Es ist eine Anpassung zu machen, sodass die URL der OpenAPI Spezifikation der eigenen BACKEND_DOMAIN‘ gleicht siehe: openapi.yaml#L13

Es ist ebenfalls nötig in den application.properties `spring.security.oauth2.resourceserver.jwt.issuer-uri` und `spring.security.oauth2.resourceserver.jwt.jwk-set-uri` auf die `AUTH_DOMAIN` anzupassen. Sollte das Realm von Keycloak nicht `DigitalForms`, so muss dies ebenfalls angepasst werden. `cors.allowed.origins`sollte `FRONTEND_DOMAIN` erlauben.

Abgesehen davon ist in auth.config.ts `issuer` auf `AUTH_DOMAIN` anzupassen und bei Bedarf das Realm anzupassen. Sollte die `clientId` im Realm nicht `df` entsprechen, so ist dies ebenfalls hier anzupassen.

Deploy

Durch Ausführen von `./DeployFromGit.sh`

FAQ Common Issues

Grandelw failed Permission denied

```
cd Backend
chmod o+x gradlew
```

Auth cant Execute initEnv.sh

```
cd Auth
chmod o+x initEnv.sh
```

OpenAPI Client Generieren

mit

```
openapi-generator-cli generate -i ../Backend/src/main/resources/openapi.yaml -g typescript-angular -o ./
```

Windows

crlf

Unter Windows könnte es zusätzlich erforderlich sein, in Git 'autocrlf' abzuschalten. Um nicht versehentlich bash scipts zu beschädigen.

```
git config core.autocrlf false
```

Patch tut nicht

Sicherstellen, dass die `.patch` Files in UTF-8 formatiert sind.

Docker Support

Unter Windows sind nicht alle Docker Features vollständig unterstützt. Daher kann für die Entwicklung darauf Windows_LocalTest.patch verwendet werden.

Formular Konfiguration

Generell

Formulare müssen derzeit im `DatabaseLoader` konfiguriert werden.

Formular Vorlage

Grundlegende Informationen

```
Form form = new Form();
form.setTemplate(true);
form.setTitel("Mein Titel");
form.setDescription("Meine Passende Bschreibung");
```

Bei der Kategorie kann zwischen `Form.CategoryEnum.ANTRAG` und `Form.CategoryEnum.ABRECHNUNG` gewählt werden.

```
form.setCategory(Form.CategoryEnum.ABRECHNUNG);
```

Formular Sektionen

Das Formular ist in einzelne Sektionen unterteilt. Diese teilen die einzelnen Elemente sinnvoll auf und tragen so zur Übersichtlichkeit bei.

```
List<FormSection> sections = new ArrayList<>();
```

Erstellen einer neuen Sektion mit

```
sections.add(createFormSection(0,"Sektionen 0"));
```

der erste Parameter ist die Sektionsnummer, der zweite der Name dieser Sektion

mit

```
addFormElement(sections,FormElement.TypeEnum,"Beschreibung"
                ,"hilfe","ID");
```

Kann ein Element der derzeitigen Sektion angefügt werden.

Parameter Form-Element Der zweite Parameter beschreibt den Typen des Elements. Es kann zwischen verschiedenen Typen gewählt werden: * FormElement.TypeEnum.TEXT * FormElement.TypeEnum.ADDRESS * FormElement.TypeEnum.IBAN * FormElement.TypeEnum.DATE * FormElement.TypeEnum.MONEY * FormElement.TypeEnum.TEXTMULTILINE * FormElement.TypeEnum.BOOL

`Beschreibung` beschreibt das Element.

`hilfe` kann zum Anzeigen von Hilfetexten verwendet werden, soll keiner angezeigt werden, so ist es auf `null` zu setzen.

Der letzte Parameter, `ID` identifiziert dieses Feld in diesem Formular eindeutig. Die `ID` wir auch für den Export verwendet.

Note: *Die `ID` darf nur aus `A` bis `Z` (klein und groß) sowie `_` zusammensetzen*

Sind alle Sektionen erstellt, so können diese angefügt werden

```
form.setForm(sections);
```

Anhänge

Anhänge sind zusätzliche Elemente, welche dem Formular beizufügen sind.

```
List<Attachment> attachments = new ArrayList<>();
```

Es kann nun zwischen 3 Typen von Anhängen unterscheiden werden.

User-Attachment Sind, Anhänge, welche eventuell angefügt werden müssen. Dies kann nicht automatisiert festgestellt werden. Erstellen mit:

```
attachments.add(createUserAttachment("ID",
                "Bescheibungen",
                "Hilfe"));
```

Required-Attachment Sind Anhänge, welche immer beizufügen sind. Erstellen mit:

```
attachments.add(createRequierdAttachment("ID",
                "Bescheibungen",
                "Hilfe"));
```

Conditional-Attachment Sind Anhänge, welche unter bestimmten, automatisch feststellbaren Bedingungen benötigt werden.

Erstellen mit:

```
attachments.add(createAttachment("ID","Bescheibungen",
                "Hilfe",
                Attachment.RequiredEnum.CONDITIONAL,"Referenz","Bedingung"));
```

Parameter Erklärung `ID` eindeutige ID des Anhangs innerhalb des Formulars

`Bescheibungen` Beschreibung zu dem Anhang

`Hilfe` Hilfestellung zu dem Anhang, kann, falls nicht, benötigt auf `null` gesetzt werden.

`Referenz` Referenziert eine Element `ID` aus diesem Formular (Element ID)

`Bedingung` definiert die Match-Bedingung zu dem referenzierten Element. Ohne ein spezielles Präfix werden die Werte auf Gleichheit geprüft.

Folgende Präfixe ist implementiert: * `>` Größer * `<` Kleiner * `=` Gleich * `!` Ungleich

Sind alle Anhänge erstellt, so können diese dem Formular beigefügt werden mit:

```
form.setAttachments(attachments);
```

Speichern

Speicher mit:

```
Form newForm = repository.save(form);
```

`newForm` ist die gespeicherte Vorlage. Die Variable wird im folgenden Schritt verwendet, um die ID des Formulars zu erhalten.

Formular Vorlage Export Link

Link zwischen Form ID & PDF Vorlage. Hier wird der name der html benötigt.

Speichern und Erstellen mit:

```
pdfRepository.save(new TemplatePDF(newForm.getId(), "name der html"));
```

`newForm.getId()` greift auf die ID aus dem vorherigen Schritt zu.

`name der html` ist der Name aus PDF Export

Antrag Abrechnungs-Gruppierung

Gruppierung von einem Antrag auf ein oder mehrerer Abrechnungen. Im Falle von mehreren soll eine Begründung/Brechreibung für die einzelnen hinterlegt werden.

Erstellen

```
tg = createTemplateGroup("Titel", "Beschribung", newForm.getId());
```

`Titel` Titel der Gruppierung

`Beschribung` Beschreibung der Gruppierung

`newForm.getId()` ID des Antrags

Abrechnung(en) Beifügen

```
tg.getRechnungen().add(Id_Erstattung);
```

`Id_Erstattung` ID des Erstattungsformulars

Bei mehr als einer möglichen Abrechnung ist eine Wahlbegründung beizufügen:

```
tg.getReasons().add("Begründung");
```

`Begründung` Soll dem Nutzer vermitteln, warum diese Option im vergeblich zu den anderen zu wählen ist.

Speichern

```
tgRepository.save(tg);
```

Änderungen übernehmen

Um Änderungen zu übernehmen, muss die derzeitige Datenbank komplett geleert werden.

Aus dem Repository mit:

```
sudo rm -rf ./DB/data/
```

PDF Export

Einführung

Alle Formulare lassen sich als PDF exportieren. Dazu müssen Vorlagen erstellt werden.

Pfad

Vorlagen müssen unter Backend/src/main/resources/templates/ abgelegt werden.

Format

Die Vorlagen sind als HTML Seiten implementiert, welch auf **Chromium** korrekt dargestellt werden sollen.
Die Darstellung auf anderen Browsern ist irrelevant für den Export

die werte Übergabe erfolgt mithilfe von Thymeleaf mehr dazu unter Thymeleaf werte Übergeben

Name

Der Dateiname einer Vorlage sollte keine Leerzeichen oder anderweitige Whitespace / Sonderzeichen enthalten.

Der Name wird bei der Verlinkung zu einem Formular wieder benötigt. (Formular Vorlage Export link)

Hilfestellungen

Das Erstellen von Templates kann am Anfang etwas überwältigend und zeitintensiv sein. Glücklicherweise wiederholen sich einige Elemente immer wieder.

Einige solcher Elemente sind in der Readme zu finden. Dabei wird auch ein gemeinsames Stylesheet verwendet, welches neben dem Styling für A4 Seiten auch die Stylings der sich wiederholenden Elemente, welche in der Readme beschrieben sind, enthält.

Abgesehen davon kann das orientieren an bereits existierenden Vorlagen hilfreich sein.

Thymeleaf Werte übergeben

Um die Werte von Formularen übergeben zu können wird Thymeleaf eingesetzt.

Das sieht dann etwa so aus:

```
<input class="FirstPageGeneral" type="text" th:value="${bez}">
```

dabei ist für Thymeleaf folgender teil relevant: `th:value="${bez}"`

`bez` ist die variable auf welcher der wert erwartet wird. Technisch muss dieser Wert der `id` des zugehörigen FormElements entsprechen, welche unter Formular Vorlage konfiguriert wird.

Sonderfälle

Nicht alle Element lassen sich so 1:1 gleich abbilden. In welchem Format Sonderfälle erwartet werden, ist in diesem Teil beschrieben

BOOL Bei vielen Formularen ist zwischen Haken für Ja und Nein zu unterscheiden. * `th:checked="${id + _YES}"` für den Ja-Fall * `th:checked="${id + _NO}"` für den Nein-Fall

TEXTMULTILINE `th:text="${id}"`

IBAN Für jedes Feld aufgeteilt:

```
th:value="${id + _1}"
th:value="${id + _2}"
th:value="${id + _3}"
th:value="${id + _4}"
th:value="${id + _5}"
th:value="${id + _6}"
```