BMBF-Listen Generator

Jörn Tillmanns - Erzeugen des PDF und CSV Verarbeitung
Konfuzzyus - Websocket und Datenbank

23. Januar 2020

Inhaltsverzeichnis

Teil I.

Überblick

1. Anforderung

Die Webanwendung **BMBF-Listen Generator** wird aus Daten, welche der Anwendung übersendet werden, mithilfe einer Vorlage ein PDF-File generieren, welches die Listen für das BMBF darstellen.

1.1. Kriterien

- Automatisierbare Ansteuerung via JSON-Post- und -Get-Requests
- Ausspielen der erzeugten PDF Dateien
- Gruppentrennung auf verschiedene Seiten
- Fehlerkontrolle

2. Produkteinsatz

2.1. Anwendungsbereiche

- Vereine
- gemeinnützige Organisationen

2.2. Betriebsbedingungen

- Server durch Aufrufer via Netzwerk erreichbar
- Notwendige Software ist auf Server installiert

3. Produktumgebung

3.1. Server

- Framework: flask
- Datenbank: MySQL/MariaDB
- Programmiersprachen: python
- Betriebssystem: Linux 64-bit

Sowie folgende Python-Erweiterungen:

- fdfgen
- sh
- pathlib

4. Produktfunktionen

4.1. Event-Basierte Funktionen

F01.00 *Neues Event* Mit Hilfe eines Webaufrufs mit Parametern lässt sich dem Generator ein neues Event hinzufügen.

F02.00 *PDF-Generieren* Die Anforderung und Generierung des PDF kann mittels eines Webaufrufs mit Parametern ausgelöst werden.

4.2. Teilnehmer-basierte Funktionen

F03.00 *Teilnehmer hinzufügen* Mit Hilfe eines Webaufrufs mit Parametern lässt sich dem Generator ein neuer Teilnehmer zu einem bestehenden Event hinzufügen.

F04.00 *Teilnehmer mit Gruppe hinzufügen:* Mit Hilfe eines Webaufrufs mit Parametern lässt sich dem Generator ein neuer Teilnehmer zu einem bestehenden Event mit einer vorhandenen oder neuen Gruppe hinzufügen. Gruppen gruppieren Teilnehmer.

5. Sicherheitsmodell

Zur Datensicherheit werden sogenannte API-Tokens verwendet, hierdurch kann ein Aufrufer nur seine Daten anfordern und auch nur an seinen eigen erstellten Events Teilnehmer hinzufügen oder das PDF generieren. Es gibt keinen Administrator API-Token, daher kann nur der Admin des Webservers alle Daten beeinflussen.

6. Datenbankentwurf

Die Datenbank besteht aus 7 Tabellen. Alle Tabellen haben ein, in der Konfigurationsdatei stehendes, Prefix voran. Bei unten stehenden Tabellen ist dies "bmbf". Hier eine Auflistung der Tabellen:

- bmbf__auth
- bmbf__events
- bmbf__groups
- bmbf__mapping
- bmbf__participants
- bmbf__templates
- bmbf__times

6.1. bmbf__auth

Wird genutzt um einen Aufrufer mittels des übermittelten API-Token zu erkennen. Es wird jedem API-Token auch eine eindeutige Nutzer-ID zugeordnet.

6.2. bmbf__events

In dieser Tabelle werden alle relevanten Daten für ein Event abgelegt. Dies umfasst unter anderem die Organisation, Maßnahmenbeschreibung und die Zeitdau-

er des Events.

6.3. bmbf__groups

In dieser Tabelle werden alle Gruppen, die das System kennt, einem Event zugeordnet. Die angegebenen Gruppennamen sind Eventspezifisch. Gleiche Gruppennamen in verschiedenen Events sind erlaubt.

6.4. bmbf__mapping

In dieser Tabelle erfolgt die Zuordnung von Events zu Nutzer.

6.5. bmbf__participants

In dieser Tabelle stehen die Teilnehmer aller Events, diese werden eindeutig den Events und eventuell vorhandenen Gruppen zugeordnet.

6.6. bmbf__templates

In dieser Tabelle werden durch den Admin des Systems alle auf dem System abgelegten ausfüllbaren PDF’s eingespeichert und können anschließend durch Aufrufer benutzt werden.

6.7. bmbf__times

In dieser Tabelle wird das Anfangs- und Enddatum eines Events im Mysql-Date-Format gespeichert. Es erfolgt eine eindeutige Zuordnung zu einem Event.

7. Schnittstellen

Die API-Schnittstelle wird in "Teil 3 - Implementierung"näher beschrieben.

8. Qualitätszielbestimmung

	sehr wichtig	wichtig	weniger wichtig	unwichtig
Robustheit	**X**			
Zuverlässigkeit		**X**		
Sicherheit		**X**		
Korrektheit		**X**		
Benutzerfreundlichkeit				**X**
Übertragbarkeit			**X**	
Effizienz			**X**	
Attraktivität				**X**
Wartbarkeit		**X**		

Teil II.

Theoretische Umsetzung

1. Grundlegende Informationen über die Datenbank

1.1. Allgemeines

1.1.1. Anmerkungen

Alle im folgenden beschriebenen Angaben beziehen sich auf die Datenbanksoftware MariaDB. Die darin enthaltenen SQL-Anweisungen beziehen sich auch auf diese spezielle Datenbank.

1.2. Tabellen

Die Daten wurden aus praktischen Gründen in mehrere Tabellen unterteilt. Es existieren einige Tabellen mit Multiple-Foreign-Keys, diese stellen diverse M zu N Beziehungen dar. Einzelbeziehungen werden stets durch referenzierte Foreign-Keys dargestellt. Im Folgenden wird näher auf den Typen eines Feldes und dessen Inhalt eingegangen.

1.2.1. bmbf__auth

1. uid

   Datentyp: Integer

   Besonderheiten: AutoIncrement, Uniqe, NotNull, Primary-Key

Bedeutung: In diese Feld wird jedem Token eine Eindeutige ID zugeordnet.

2. token

   Datentyp: Varchar, Länge 190

   Besonderheiten: NotNull

   Bedeutung: In diesem Feld werden die Strings abgelegt, welche die Aufrufer in ihren Aufrufen zur Authentifikation verwenden. Diese bestehen aus einer Zeichenkette, welche aus Groß- und Kleinbuchstaben sowie Zahlen besteht. Durch das AddNewTool.py-Script welches zur Erzeugung der Token genutzt wird, werden Duplikate verhindert.

Schlüssel

Primärer Schlüssel Der Primärer Schlüssel in dieser Tabelle ist die ID. Da jeder Nutzertoken eindeutig und einfach seinen Events zugeordnet werden kann. Zugleich spart es auch Speicherplatz, in dem aus einem *Varchar* ein Integer wird.

Fremdschlüssel Diese Tabelle hat keine Fremdschlüssel.

1.2.2. bmbf_events

1. id

   Datentyp: Integer

   Besonderheiten: AutoIncrement, PrimaryKey

   Bedeutung: Dies stellt die eindeutige Identifikation des Events dar.

2. organization

   Datentyp: Mediumtext

Besonderheiten: Not Null

Bedeutung: Name der Organisation, für welche das PDF erstellt wird. Auch für den Inhalt des PDFs notwendig.

3. measure

   Datentyp: Mediumtext

   Besonderheiten: Not Null

   Bedeutung: Ist die Bezeichnung der Maßnahme für welche eine Liste erstellt werden soll.

4. template

   Datentyp: int

   Besonderheiten: Not Null

   Bedeutung: Enthalt eine ID aus der der Tabelle *bmbf__templates* mit der ein bestimmtes Template beim Erzeugen der PDF ausgewählt werden kann.

5. measure_periode

   Datentyp: mediumtext

   Besonderheiten: Not Null

   Bedeutung: Ist eine Zeichenkette welche den Maßnahmenzeitraum beziffert. Wird unverändert in das Vorlagen-PDF geschrieben.

Schlüssel

Primärer Schlüssel Der primäre Schlüssel in dieser Tabelle ist die ID. Da jedes Event eindeutig und einfach seinen Teilnehmern und Gruppen zugeordnet werden kann.

Fremdschlüssel

Template Im Feld *template* werden nur Werte akzeptiert, welche auch in *bmbf__templates* als *id*-Wert vorkommen.

1.2.3. bmbf__groups

1. event

   Datentyp: Integer

   Besonderheiten: Not Null

   Bedeutung: Dies ordnet eine Gruppe einem Event zu.

2. group_id

   Datentyp: Mediumtext

   Besonderheiten: Not Null

   Bedeutung: Gruppen-Identifikator des aufrufenden Systems. In diesem System kein eindeutiger Identifikator.

3. ugid

   Datentyp: int

   Besonderheiten: Auto Increment, Primary Key

   Bedeutung: Stellt den eindeutigen Identifikator der Gruppe in diesem System dar.

Schlüssel

Primärer Schlüssel Der primäre Schlüssel in dieser Tabelle ist die UGID. Da jede Gruppe eindeutig und einfach seinen Mitgliedern, welche Teilnehmer eines Events sind, zugeordnet kann.

Fremdschlüssel

event Im Feld *event* werden nur Werte akzeptiert, welche auch in *bmbf__events* als *id*-Wert vorkommen.

1.2.4. bmbf__mapping

1. uid

   Datentyp: Integer

   Besonderheiten: Not Null

   Bedeutung: Dies stellt eine Nutzerzuordnung dar.

2. eid

   Datentyp: Integer

   Besonderheiten: Not Null

   Bedeutung: Dies stellt eine Eventzuordnung dar.

Schlüssel

Primärer Schlüssel Diese Table hat keinen primären Schlüssel.

Fremdschlüssel

uid Im Feld *uid* werden nur Werte akzeptiert, welche auch in *bmbf__auth* als *id*-Wert vorkommen.

eid Im Feld *eid* werden nur Werte akzeptiert, welche auch in *bmbf__events* als *id*-Wert vorkommen.

1.2.5. bmbf__participants

1. id

   Datentyp: Integer

   Besonderheiten: Auto Increment, Primary Key

   Bedeutung: Dieses Feld dient der eindeutigen Identifikation eines Templates.

2. event

   Datentyp: Integer

   Besonderheiten: Not Null

   Bedeutung: Dies stellt eine Eventzuordnung dar.

3. name

   Datentyp: Mediumtext

   Besonderheiten: Not Null

   Bedeutung: Dies ist der Name des Teilnehmers, welcher so auch in das Template-PDF geschrieben wird.

4. university

   Datentyp: Mediumtext

   Besonderheiten: Not Null

   Bedeutung: Dies ist der Universitätsname des Teilnehmers, welcher so auch in das Template-PDF geschrieben wird.

5. grp

   Datentyp: Integer

   Besonderheiten: Kann Null sein

   Bedeutung: Dies stellt eine Gruppenzuordnung dar.

Schlüssel

Primärer Schlüssel Der primäre Schlüssel in dieser Tabelle ist die ID. Sie dient zum eindeutigen Abrufen von Teilnehmern und zur eindeutigen Identifikation von Teilnehmern bei Problemen mit der Zeichenkodierung (Bspw.: Griechische UTF-8 Zeichen)

Fremdschlüssel

event Im Feld *event* werden nur Werte akzeptiert, welche auch in *bmbf__events* als *id*-Wert vorkommen.

grp Im Feld *grp* werden nur Werte akzeptiert, welche auch in *bmbf__groups* als *ugid*-Wert vorkommen.

1.2.6. bmbf__templates

1. id

   Datentyp: Integer

   Besonderheiten: Auto Increment, Primary Key

   Bedeutung: Dieses Feld dient der eindeutigen Identifikation eines Teilnehmers.

2. filename

   Datentyp: Mediumtext

   Besonderheiten: Not Null

   Bedeutung: Dies ist der Dateiname des genutzen Templates, welcher zum Erzeugen der ausgefüllten PDFs benötigt wird.

Schlüssel

Primärer Schlüssel Der primäre Schlüssel in dieser Tabelle ist die ID. Sie dient zum eindeutigen Abrufen von Daten zu einem Template.

Fremdschlüssel Diese Tabelle hat keine Fremdschlüssel.

1.2.7. bmbf__times

1. event

   Datentyp: Integer

   Besonderheiten: Not Null

   Bedeutung: Dies stellt eine Eventzuordnung dar.

2. startdate

   Datentyp: Date

   Besonderheiten: Not Null

   Bedeutung: Stellt das Anfangsdatum eines Events in Maschinenlesbarer weise dar.

3. enddate

Datentyp: Date

Besonderheiten: Not Null

Bedeutung: Stellt das Enddatum eines Events in Maschinenlesbarer weise dar.

Schlüssel

Primärer Schlüssel Diese Table hat keinen primären Schlüssel.

Fremdschlüssel

event Im Feld *event* werden nur Werte akzeptiert, welche auch in *bmbf__events* als *id*-Wert vorkommen.

Teil III.

Implementierung

1. Dateien

1.1. AddNewTool.py

Bedeutung und Besonderheiten Fügt dem BMBF-Listen-Generator einen neuen API-Token hinzu und gibt diesen aus. Aus Anwendungssicht entsteht ein neuer Nutzer.

Besonderheiten Gibt den API-Token direkt auf der Konsole aus. Script bedarf keiner Input Parameter.

Funktionen:

1. *randomStringDigits(stringLength:int)*

   Gibt eine Zeichenkette bestehend aus der Anzahl der in *stringLength* angegebenen Zeichen zurück. Es generiert den eigentlichen API-Token.

2. *__name__=="__main__"*

   Generiert einen API-Token mittels der Funktion *randomStringDigits* und schreibt den erhaltenen Token einerseits in die Datenbank und andererseits gibt es den Token auf Konsole aus.

1.2. bmbf_main.py

Bedeutung und Besonderheiten Liest die Konfigurationsdatei ein und ruft dann *generate_bmbf_list.mainDB(1)* oder *generate_bmbf_list.mainFiles()* auf. Ent-

scheiden wird das mittels der Einstellung *config.use_db*.

Besonderheiten Script bedarf keiner Input Parameter.

Funktionen:

1. *__name__=="__main__"*

   Entscheidet in der oben beschrieben Art und Weise was es aufruft.

1.3. Commands.py

Bedeutung und Besonderheiten Stellt alle über die Web-API aufrufbaren Funktionen dar. Ist der Hauptteil der Websocket Implementierung. Hier wird jede Anfrage beantwortet. Die Funktionen werden von der *webstart.py* aufgerufen.

Besonderheiten Darf nicht direkt aus der Konsole aufgerufen werden. Wird durch anderen Anwendungsteil aufgerufen. Jede Funktion prüft mittels des API-Token ob ein Nutzer berechtigt ist eine Aktion durch zu führen. Dies passiert mithilfe der Funktionen *checkToken(token:string)* oder *checkAccess(uid:Int,eid:Int)*. Sollte ein Nutzer nicht berechtigt sein, so wird Ihm mit einem *"You shall not pass."* geantwortet.

Funktionen:

1. *NewPerson(p)*

   Stellt aus den in *p* gegebenen Daten, welche in bestimmter weise als Array übergeben werden müssen, die Daten zum Anlegen eines neuen Teilnehmers in der Datenbank zusammen. Anschließend ruft es eine Funktion zum Schreiben in die Datenbank auf.

2. *NewEvent(e)*

Stellt aus den in *e* gegebenen Daten, welche in bestimmter weise als Array übergeben werden müssen, die Daten zum Anlegen eines neuen Events in der Datenbank zusammen. Es gibt die Event ID zurück. Besonderheit ist jedoch, das sich die Events mindestens in einem String unterscheiden müssen, damit die Funktion sauber arbeitet. Anschließend ruft es eine Funktion zum schreiben in die Datenbank auf. Auch wird ein Mapping von API-Token zu Event in die Datenbank geschrieben.

3. *GeneratePDF(e)*

   Stellt aus den in *e* gegebenen Daten, welche in bestimmter weise als Array übergeben werden müssen, die Daten zum Erstellen des PDFs zusammen. Anschließend wird das PDF mittels der *generate_bmbf_list.mainDB(eid:Int)* Funktion generiert.

4. *RequestTemplates(token)*

   Stellt mithilfe der in *token* gegebenen Daten, welcher nur ein String sein, die Daten zum den Templates zusammen.

5. *__name__=="__main__"*

   Generiert einen API-Token mittels der Funktion *randomStringDigits* und schreibt den erhaltenen Token einerseits in die Datenbank und andererseits gibt es den Token auf Konsole aus.

1.4. config-sample.py

Bedeutung und Besonderheiten Stellt die Beispielkonfiguration dar. Die genutzte Konfigurationsdatei muss *config.py* heißen und kann eine Kopie dieser Datei darstellen.

Besonderheiten Darf nicht direkt aus der Konsole aufgerufen werden. Wird durch anderen Anwendungsteil aufgerufen, andernfalls beendet es mit Code 1. Alle unten stehenden 'Funktionen' stellen Parameter der Software dar und sind keine Funktionen, sondern Werte.

Funktionen:

1. *use_db*

   Dieser Parameter hat 2 Wertoptionen:

   - **False:** Es wird die Datenbank nicht genutzt.
   - **True:** Es wird die Datenbank genutzt.

1.4.1. Parameter für die Nutzung ohne Datenbank:

Erklärung Bei dieser Nutzungsvariante ist es nicht möglich Teilnehmerdaten aus der Datenbank zu lesen. Es wird ausschließlich aus den Dateiquellen gelesen.

2. *measures_period*

   Stellt den Text dar, welcher in das entsprechende Feld der Template-PDF geschrieben wird. Es ist eine passende Zeichenkette anzugeben, die ohne Änderungen übernommen wird. In diesem Fall wird der Maßnahmenzeitraum definiert.

3. *date*

   Wird nicht in das PDF eingebettet. Stellt das Datum der Erstellung dar.

4. *organization*

   Stellt den Text dar, welcher in das entsprechende Feld der Template-PDF geschrieben wird. Es ist eine passende Zeichenkette anzugeben, die ohne Änderungen übernommen wird. In diesem Fall wird die Organisation definiert.

5. *measure*

   Stellt den Text dar, welcher in das entsprechende Feld der Template-PDF geschrieben wird. Es ist eine passende Zeichenkette anzugeben, die ohne Änderungen übernommen wird. In diesem Fall wird der Maßnahmentitel definiert.

6. *csv_file_name*

   Hier muss die CSV-Datei angegeben werden, in welcher die Teilnehmerdaten zu finden sind. Sie kann relativ angegeben werden.

7. *template*

   Hier muss die Template-PDF angegeben werden, welche verwendet werden soll. Sie kann relativ angegeben werden.

8. *list_dates*

   Hier muss eine Auflistung aller Tage des Events angegeben werden. Die Tage sind in folgender weise anzugeben: *DD.MM.YYYY*. Und in einem Python-Array zu schreiben. Dies muss dann folgendermaßen angegeben werden, durch eine andere Art der Angabe können Fehler entstehen: *["DD.MM.YYYY", "DD.MM.YYYY"]*.

1.4.2. Parameter für die Nutzung mit Datenbank:

Erklärung Bei dieser Nutzungsvariante ist es nicht möglich Teilnehmerdaten aus einer CSV zu lesen. Es wird ausschließlich aus der Datenbank gelesen.

9. *db_host*

   Hier muss der Datenbankhost angegeben werden, dies kann auf 3 Varianten geschehen:

   - **Lokale Datenbank:** Dann kann hier der Wert *localhost* angegeben werden, alternativ kann auch *127.0.0.1* angegeben werden.
   - **DNS-Adresse:** Dies ist folgendem Beispiel ähnlich an zu geben: *example.com*
   - **IP-Adresse:** Dies ist folgendem regulärem Ausdruck entsprechend ähnlich an zu geben: *\d{1,3}.\d{1,3}.\d{1,3}.\d{1,3}*

10. *db_port*

Hier muss ein Integerwert angegeben werden, welcher den Port auf dem die Datenbank erreichbar ist darstellt.

11. *db_scheme*

    Hier muss eine Zeichenkette angegeben werden, welches Datenbank Schema in der Datenbank verwendet werden soll.

12. *db_prefix*

    Hier muss eine Zeichenkette angegeben werden, welches den Tabellennamen, die im verwendeten Schema in der Datenbank zu finden sind, voran steht.

13. *db_user*

    Hier muss eine Zeichenkette angegeben werden, welche den Nutzernamen darstellt, mit welchem die in der Datenbank gearbeitet werden soll.

14. *db_password*

    Hier muss eine Zeichenkette angegeben werden, welche das Passwort zum Nutzernamen darstellt, mit welchem die in der Datenbank gearbeitet werden soll.

15. *dns*

    Hier muss eine Zeichenkette angegeben werden, welche die Adresse unter dem die API verfügbar ist angegeben werden.

1.4.3. Allgemeine Parameter:

16. *empty_sheets*

    Der hier angegebene Integerwert stellt die Anzahl der leeren Seiten pro Tag oder pro Tag und Gruppe dar.

17. *debug*

Es können 2 Werte angegeben werden:

- **False:** Deaktiviert die Debug-Ausgaben.
- **True:** Aktiviert die Debug-Ausgaben.

1.4.4. Funktion:

18. *__name__=="__main__"*

    Stellt sicher, dass die Datei nicht alleine ausgeführt wird und beendet den Aufruf mit Code 1.

1.5. databaseConnect.py

Bedeutung und Besonderheiten Stellt die Verbindung zur Datenbank dar und beinhaltet alle benötigten Funktionen und Variablen oder fasst diese aus der Konfigurationsdatei zusammen.

Besonderheiten Darf nicht direkt aus der Konsole aufgerufen werden. Wird durch anderen Anwendungsteil aufgerufen.

1.5.1. Variablen:

1. *dbConnectCfg*

   Stellt die Zusammenfassung aller Daten, die zum Verbinden zur Datenbank benötigt werden dar. Die Daten werden aus der Konfigurationsdatei gelesen.

2. *alltemplates*

   Stellt den *SQL*-Befehl dar, mittels dessen alle Templates aus der DB gelesen werden können.

3. *allevents*

   Stellt den *SQL*-Befehl dar, mittels dessen alle Events aus der DB gelesen werden können.

1.5.2. Funktionen:

4. *QueryDB(query)*

   Führt die in *query* übergebene Abfrage, welche keine Parameter benötigt, aus und gibt das Ergebnis zurück.

5. *QueryDBParameter(query, par)*

   Führt die in *query* übergebene Abfrage, welche Parameter benötigt, die in *par* gegeben sind, aus und gibt das Ergebnis zurück.

6. *QueryDBParameterWOO(query, par)*

   Führt die in *query* übergebene Abfrage, welche Parameter benötigt, die in *par* gegeben sind, aus und gibt kein Ergebnis zurück.

7. *ReadAllTemplates()*

   Gibt alle Templates welche in der Datenbank zu finden sind zurück.

8. *ReadSpecificTemplate(id)*

   Gibt die Daten zu einem speziellen Template zurück. Die Daten werden mithilfe des *id*-Paramters gesucht.

9. *ReadAllEvents():*

   Gibt alle Events welche in der Datenbank zu finden sind zurück.

10. *GetGroupsByEID(eid)*

    Gibt alle Gruppen, welche in der Datenbank zu einem bestimmten Event zu finden sind zurück. Das Event wird mittels der Event-ID übergeben,

welche im Parameter *eid* zu finden ist.

11. *GetParticipantsByEIDGID(eid, gid)*

    Gibt alle Teilnehmer, welche in der Datenbank zu einem bestimmten Event und in einer bestimmten Gruppe des Events zu finden sind zurück. Das Event wird mittels der Event-ID übergeben, welche im Parameter *eid* zu finden ist. Die Gruppe wird mittels der Gruppen-ID übergeben, welche im Parameter *gid* zu finden ist.

12. *ReadPersonsWOG(eid)*

    Gibt alle Teilnehmer, welche in der Datenbank zu einem bestimmten Event zu finden sind zurück. Das Event wird mittels der Event-ID übergeben, welche im Parameter *eid* zu finden ist.

13. *checkGrp(id)*

    Gibt alle Gruppen, welche in der Datenbank zu einem bestimmten Event zu finden sind zurück. Das Event wird mittels der Event-ID übergeben, welche im Parameter *id* zu finden ist.

14. *ReadEventWG(id)*

    Gibt alle Daten eines Events mit Gruppen mit allen Teilnehmern zurück. Das Event wird mittels der Event-ID übergeben, welche im Parameter *id* zu finden ist.

15. *ReadEventWOG(id)*

    Gibt alle Daten eines Events ohne Gruppen mit allen Teilnehmern zurück. Das Event wird mittels der Event-ID übergeben, welche im Parameter *id* zu finden ist.

16. *ReadListOfDays(id)*

    Gibt eine Liste der Tage eines Events zurück. Das Event wird mittels der Event-ID übergeben, welche im Parameter *id* zu finden ist.

17. *insertEvent(e)*

Fügt ein neues Event dazu und gibt die Event-ID zuürck. Die dazu benötigten Daten werden aus dem Parameter *e* gelesen. Dieser Parameter muss folgende Elemente beinhalten:

- organization
- measure
- template
- measure_periode
- startdate
- enddate

18. *insertPerson(e)*

    Fügt einen Teilnehmer zu einem Event dazu. Die dazu benötigten Daten werden aus dem Parameter *e* gelesen. Dieser Parameter muss folgende Elemente beinhalten:

    - eid
    - name
    - university
    - gid

19. *GetGroupByEID(eid)*

    Gibt alle Gruppen eines Events zurück. Das Event wird mittels der Event-ID übergeben, welche im Parameter *id* zu finden ist. Die Rückgabewerte sind in einem Array enthalten, welches keine Schlüssel besitzt.

20. *InsertGrp(eid, gid)*

    Fügt eine Gruppe zu einem Event dazu. Die dazu benötigten Daten werden aus dem Parametern *eid* und *gid* gelesen, wobei *gid* die Gruppen-ID des Aufrufers und *eid* die Event-ID darstellt.

21. *InsertTokenToDB(token)*

    Fügt einen neuen Token in die Datenbank ein. Die dazu benötigten Daten werden aus dem Parameter *token* gelesen.

22. *InsertMapping(uid, eid)*

    Fügt einen Zuweisung von Nutzer zu neu erstelltem Event in die Datenbank ein. Die dazu benötigten Daten werden aus dem Parametern *uid* und *eid* gelesen. Wobei *uid* die Nutzer-ID oder Token-ID und *eid* die Event-ID darstellt.

23. *checkAccess(token, eid)*

    Prüft den Zugriff auf ein Event mithilfe des API-Tokens des Aufrufers. Die dazu benötigten Daten werden aus dem Parametern *token* und *eid* gelesen. Wobei *token* den Token des Aufrufers und *eid* die Event-ID darstellt. Der Rückgabe Wert ist einer folgenden 2 Werte:

    - **True:** Der Zugriff auf ein Event ist erlaubt.
    - **False:** Der Zugriff auf ein Event ist nicht erlaubt.

24. *checkToken(token)*

    Prüft den Zugriff mithilfe des API-Tokens des Aufrufers. Die dazu benötigten Daten werden aus dem Parameter *token* gelesen. Wobei *token* den Token des Aufrufers und darstellt. Der Rückgabe Wert ist einer folgenden 2 Werte:

    - **True:** Der Token ist gültig.
    - **False:** Der Token ist ungültig.

25. *getUidFromToken(token)*

    Gibt die User-ID des Aufrufers zurück. Die dazu benötigten Daten werden aus dem Parameter *token* gelesen. Wobei *token* den Token des Aufrufers und darstellt.

26. *__name__=="__main__"*

Stellt sicher, dass die Datei nicht alleine ausgeführt wird und beendet den Aufruf mit Code 1.

1.6. generate_bmbf_list.py

Bedeutung und Besonderheiten Enthält alle Variablen und Funktionen, welche zum erstellen der PDF benötigt werden.

Besonderheiten Darf nicht direkt aus der Konsole aufgerufen werden. Wird durch anderen Anwendungsteil aufgerufen.

1.6.1. Variablen:

1. *form_mapping*

   Stellt eine Liste in einer Liste dar mit den Namen der Felder für die Teilnehmer in der Vorlagen PDF dar.

1.6.2. Funktionen:

2. *readcsv(csv_file_name)*

   Liest die im übergebenen Parameter angegebene Datei als CSV ein und gibt ein Array mit den gelesenen Daten zurück.

3. *generate_pdfs(persons, massnahmenzeitraum, datum, kif_ev, massname, leer_blaetter, template)*

   Generiert das PDF eines Tages in der Nutzungsvariante ohne Datenbank.

4. *generate_pdfsDB(persons, massnahmenzeitraum, datum, kif_ev, massname, leer_blaetter, template, page)*

   Generiert das PDF eines Tages in der Nutzungsvariante mit Datenbank.

5. *mainFiles()*

   Ist die Hauptfunktion dieser Datei, falls das Tool ohne Datenbank verwendet wird und generiert das gewollte PDF.

6. *mainDB(id)*

   Ist die Hauptfunktion dieser Datei, falls das Tool ohne Datenbank verwendet wird und generiert das gewollte PDF des Events, welches mithilfe des *id*-Parameters ausgewählt wird.

7. *__name__=="__main__"*

   Stellt sicher, dass die Datei nicht alleine ausgeführt wird und beendet den Aufruf mit Code 1.

1.7. helper.py

Bedeutung und Besonderheiten Enthält benötigte Hilfsfunktionen für die Anwendung.

Besonderheiten Darf nicht direkt aus der Konsole aufgerufen werden. Wird durch anderen Anwendungsteil aufgerufen.

Funktionen:

1. *sha256name(filename)*

   Gibt die SHA256-Summe der angegeben Datei zurück.

2. *renameFile(old)*

   Benennt die Datei welche in *old* benannt ist um. Der neue Dateiname wird mithilfe der *sha256name*-Funktion bestimmt und anschließend zurück gegeben.

3. *assimilatePersons(personList)*

Gleicht die Daten aus der Datenbank für den von der KIF e.V. übernommen Teil an.

4. *MySQLEscape(input)*

   Kapselt die SQL-Escape Funktion.

5. *GarbageCollect(files)*

   Löscht alle im *files*-Paramter angegebenen Dateien.

6. *RenumberPersons(event)*

   Nummeriert die Liste der Teilnehmer neu, da die Datenbank IDs der Teilnehmer nicht bei 1 beginnen muss.

7. *__name__=="__main__"*

   Stellt sicher, dass die Datei nicht alleine ausgeführt wird und beendet den Aufruf mit Code 1.

1.8. webstart.py

Bedeutung und Besonderheiten Stellt die Startdatei für den Websocket dar.

Besonderheiten Es werden keine Startparameter benötigt.

Funktionen:

1. *test()*

   Ist eine Testfunktion für das Senden von API-Requests, benötigt keine Authentifikation mittels API-Token.

2. *bmbf()*

Stellt die Aufteilung der Anforderung die Mittels API-Request gestellt wird dar.

3. *get_pdf(id=None)*

   Sendet das PDF dem Aufrufer zurück. Es stellt keinen API-Aufruf dar.

4. *__name__=="__main__"*

   Startet das Websocket.

Teil IV.

API-Schnittstellenbeschreibung

1. Request

1.1. Event anlegen

Bedeutung Dient zum erstellen eines neuen Events. Alle übergebenen Daten werden der Datenbank hinzugefügt.

```
{
        "token" : "Put Your Token here",
        "type" : "NE",
        "organization" : "Your Organization",
        "measure" : "Your Measure Titel",
        "template" : 2,
        "measure_periode" : "dd.mm-dd.mm.yyyy",
        "startdate" : "yyyy-mm-dd",
        "enddate" : "yyyy-mm-dd"
}
```

1.1.1. Parameter

token: Hier bitte den API-Token des Aufrufers einfügen.

type: Muss hier unbedingt "NEßein.

organization: Hier bitte den Namen der Organisation hineinschreiben.

measure: Hier bitte den Maßnahmentitel hineinschreiben.

template: Hier bitte die ID des gewünschten Templates einfügen.

measure_periode: Hier bitte den Maßnahmenzeitraum angeben.

startdate: Hier bitte den Starttag angeben.

Besondertheit Muss im Format *YYYY-MM-DD* angegeben werden.

enddate: Hier bitte den Endtag angeben.

Besondertheit Muss im Format *YYYY-MM-DD* angegeben werden.

1.2. Templates abrufen

Bedeutung Sendet dem Aufrufer alle Daten zu den verfügbaren Templates zu.

```
{
        "token" : "Put Your Token here",
        "type" : "RT"
}
```

1.2.1. Parameter

token: Hier bitte den API-Token des Aufrufers einfügen.

type: Muss hier unbedingt "RTßein.

1.3. Neue Person mit Gruppe anlegen

Bedeutung Legt einen Teilnehmer mit Gruppe im angegebenen Event an.

```
{
        "token" : "Put Your Token here",
        "type" : "NP",
        "eid" : 1,
        "Name" : "Mia Killing",
        "University" : "Freie Universität Mordor",
        "gid" : 7
}
```

1.3.1. Parameter

token: Hier bitte den API-Token des Aufrufers einfügen.

type: Muss hier unbedingt "NPßein.

eid: Hier bitte die Event-ID des Events einfügen, zu dem ein Teilnehmer hinzugefügt werden soll.

Name: Hier bitte den Namen des Teilnehmers einfügen.

University: Hier bitte den Namen der Universität des Teilnehmers einfügen.

gid Hier bitte die Gruppe des Teilnehmers im gewünschten Event hinzufügen.

1.4. Neue Person ohne Gruppe anlegen

Bedeutung Legt einen Teilnehmer ohne Gruppe im angegebenen Event an.

```
{
        "token" : "Put Your Token here",
        "type" : "NP",
        "eid" : 1,
        "Name" : "Mia Killing",
        "University" : "Freie Universität Mordor"
}
```

1.4.1. Parameter

token: Hier bitte den API-Token des Aufrufers einfügen.

type: Muss hier unbedingt "NPßein.

eid: Hier bitte die Event-ID des Events einfügen, zu dem ein Teilnehmer hinzugefügt werden soll.

Name: Hier bitte den Namen des Teilnehmers einfügen.

University: Hier bitte den Namen der Universität des Teilnehmers einfügen.

1.5. PDF generieren

Bedeutung Legt einen Teilnehmer ohne Gruppe im angegebenen Event an.

```
{
        "token" : "Put Your Token here",
        "type" : "GP",
        "eid" : 1
}
```

1.5.1. Parameter

token: Hier bitte den API-Token des Aufrufers einfügen.

type: Muss hier unbedingt "GPßein.

eid: Hier bitte die Event-ID des Events einfügen, zu dem das PDF generiert werden soll.

2. Reply

2.1. Event anlegen

Bedeutung Ist die Antwort auf die Anfrage zum anlegen eines neuen Events.

```
{
        "enddate" : "yyyy-mm-dd",
        "id": 12,
        "measure" : "Your Measure Titel",
        "measure_periode" : "dd.mm-dd.mm.yyyy",
        "organization" : "Your Organization",
        "startdate" : "yyyy-mm-dd",
        "template": 2,
        "type": "NE"
}
```

2.1.1. Parameter

organization: Hier steht der Namen der Organisation.

measure: Hier steht der Maßnahmentitel.

template: Hier steht die ID des gewünschten Templates.

measure_periode: Hier steht der Maßnahmenzeitraum.

startdate: Hier steht der Starttag.

Besondertheit Ist im Format *YYYY-MM-DD* angegeben.

enddate: Hier steht der Endtag.

Besondertheit Ist im Format *YYYY-MM-DD* angegeben.

id: Hier steht die Id des neu erstellten Events.

2.2. Templates abrufen

Bedeutung Ist die Antwort auf die Abfrage aller Templates.

```
[
        {
                "filename": "xxx.pdf",
                "id": 1
        },
        {
                "filename": "xxx.pdf",
                "id": 2
        }
]
```

2.2.1. Parameter

filename: Hier steht der Dateiname des PDF-Templates.

id: Muss steht die ID des PDF-Templates.

2.3. Neue Person mit Gruppe anlegen

Bedeutung Ist die Antwort auf die Anfrage zum erstellen eines neuen Teilnehmers mit Gruppe.

```
{
        "result": null
}
```

2.3.1. Parameter

result: Hier steht das Ergebnis der SQL-Abfrage.

Besonderheit Der Wert *null* ist der zu erwartende Wert.

2.4. Neue Person ohne Gruppe anlegen

Bedeutung Ist die Antwort auf die Anfrage zum erstellen eines neuen Teilnehmers ohne Gruppe.

```
{
        "result": null
}
```

2.4.1. Parameter

result: Hier steht das Ergebnis der SQL-Abfrage.

Besonderheit Der Wert *null* ist der zu erwartende Wert.

2.5. PDF generieren

Bedeutung Stellt die Antwort auf die Anfrage zum generieren eines PDFs dar.

```
{
"result": "generated",
"File" : "URL"
}
```

2.5.1. Parameter

result: Ist der Status des Documentes.

Besonderheit Der zu erwartende Wert ist *generated*.

File: Hier steht der Web-Pfad zum Download des generierten PDFs.

Teil V.

Glossar

Array: Programmstruktur; eine eindimensionale Reihe von Zellen

AutoIncrement: die Zahl wird vom System bei jedem neuen Anlegen hochgezählt

Boolean/boolscher Wert: Datentyp; nimmt stehts nur einen von zwei möglichen Werten an, meist wahr oder falsch

CamelCase: Schreibweise in Quellcode, bei der Worttrennung mittels Großbuchstaben in einem Wort erfolgt; einBeispiel statt ein Beispiel

Client: Nutzerprogramm

Get Methode/Getter: Methode, die das Lesen einer privaten Variable ermöglicht

GUI: Grafische Benutzeroberläche (Graphic User Interface)

hash/hashing: ein effizienter Algorithmus zur Speicherung und Suche von Daten auf Tabellen

ID: Identifikationsnummer

Integer: Datentyp; eine Ganzzahl

MEDIUMTEXT: MySQL spezifischer Datentyp; wie Varchar kann aber bis ca 16 mio Zeichen enthalten

NotNull: gibt an, dass ein Feld/eine Zelle einen Eintrag haben muss

persistent: dauerhaft

private Variable: Eintrag, der nur in einem kleinem (privaten) Umfeld gelesen oder beschrieben werden kann

public Variable: Eintrag der in einem weiten (öffentlichen) Umfeld gelesen oder beschrieben werden kann

Set Methode/Setter: Methode die das Schreiben in einer privaten Variable ermöglicht

SQL Foreign-Key/Fremdschlüssel: Fremdschlüssel; Eintrag um auf Elemente aus anderen Tabellen einer Datenbank zu verweisen

SQL Key: Schlüsseleintrag um einen Datenbankelement zu identifizieren

SQL Varchar: Datentyp; enthält eine Menge von Buchstaben, Sonderzeichen o.ä. Die Zahl in der Klammer gibt die maximale Länge an

String: Datentyp; wie Varchar aber nicht SQL spezifisch, Länge beliebig

Unique: (hier) ein Eintrag in einer Datenbankspalte, der sich von allen anderen unterscheidet

Primary Key/Schlüssel/Primärer Schlüssel: Bezeichnet ein Feld, mithilfe dessen sich ein Datensatz eindeutig Identifizieren lässt.