Dokumentation Erwachsenenbildung Rest-API

Einführung 5
Aufruf der Rest-API 5
Response Codes 5
Datenobjekte 6
course (Vollständiges Kursobjekt) 6
courseSmall (Abgespecktes Kursobjekt) 7
courseEssential (Essentielles Kursobjekt) 7
courseMeeting (Vollständiges Kombinationsobjekt aus Kurs und Termin) 8
educationalInstruction (Vollständiges Bildungseinrichtungsobjekt) 8
meeting (Vollständiges Terminobjekt) 8
instructor (Vollständiges Dozentenobjekt) 9
meetingInstructor (Vollständiges Kombinationsobjekt aus Termin und Dozenten) 9
room (Vollständiges Raumobjekt) 10
meetingRoom (Vollständiges Kombinationsobjekt aus Termin und Raum) 10
address (Vollständiges Adressobjekt) 10
holidayCalendar (Vollständiges Ferienkalenderobjekt) 11
category (Vollständiges Kategorieobjekt) 11
eventType (Vollständiges Veranstaltungsartobjekt) 11
cooperation (Vollständiges Kooperationsobjekt) 11
targetGroup (Vollständiges Zielgruppenobjekt) 11
range (Vollständiges Datumsbereichsobjekt) 12
filter (Vollständiges Filterobjekt) 12
Course 13
/rest/course/{uid} 13
/rest/course/{uid}/deleted 13
/rest/course/category/{uid} 13
/rest/course/category/{uid}/all 13
/rest/course/category/{uid}/deleted 14
/rest/course/instructor/{uid} 14
/rest/course/instructor/{uid}/all 14
/rest/course/instructor/{uid}/deleted 14
/rest/course/educationalInstruction/{uid} 15

/rest/course/educationalInstruction/{uid}/all....15
/rest/course/educationalInstruction/{uid}/deleted....15
/rest/course/educationalInstruction/{uid}/complete....15
/rest/course/educationalInstruction/{uid}/complete/all....16
/rest/course/educationalInstruction/{uid}/complete/deleted....16
/rest/course/{apiKey}/switch/{uid}/state/{state}....16
/rest/course/deleted/init....16
/rest/course/search/{searchString}....17
/rest/course/category/{uid}/search/{searchString}....17
/rest/course/{apiKey}/create....18
/rest/course/{apiKey}/create....19
/rest/course/{apiKey}/create/uid....20
/rest/course/{apiKey}/create/uid....21
/rest/course/{apiKey}/update....22
/rest/course/{apiKey}/update....23
/rest/course/{apiKey}/update/uid....24
/rest/course/{apiKey}/update/uid....25
/rest/course/{apiKey}/import....26
/rest/course/{apiKey}/import....27
/rest/course/{apiKey}/import/delete....28
/rest/course/{apiKey}/import/delete....29
/rest/course/{apiKey}/restore/uid....30
/rest/course/{apiKey}/restore/uid....30
/rest/course/filter....30
/rest/course/filter....31
/rest/course/{apiKey}/{courseNumber}....32
/rest/course/{apiKey}/{courseNumber}....32
/rest/course/{apiKey}/{uid}/uid....32
/rest/course/{apiKey}/{uid}/uid....32
EducationalInstruction....33
/rest/educationalInstruction....33
/rest/educationalInstruction/{apiKey}....33
Meeting....34
/rest/meeting....34
/rest/meeting/educationalInstruction/{uid}....34

/rest/meeting/room/{uid} 34
/rest/meeting/room/{uid}/hidden 34
/rest/meeting/instructor/{uid} 35
/rest/instructor/room/{uid}/hidden 35
Instructor 36
/rest/instructor/educationalInstruction/{uid} 36
/rest/instructor/{apiKey}/blocked/date 36
/rest/instructor/{apiKey}/blocked/date 36
/rest/instructor/{apiKey}/blocked/date/multiple 36
/rest/instructor/{apiKey}/blocked/date/multiple 37
/rest/instructor/{apiKey}/create 37
/rest/instructor/{apiKey}/create 37
/rest/instructor/{apiKey}/update/uid 38
/rest/instructor/{apiKey}/update/uid 38
/rest/instructor/{apiKey}/cud/uid 38
/rest/instructor /{apiKey}/cud/uid 39
/rest/instructor /{apiKey}/{uid}/uid 39
/rest/instructor /{apiKey}/{uid}/uid 39
Room 40
/rest/room/educationalInstruction/{uid} 40
/rest/room/{apiKey}/addresses 40
/rest/room/{apiKey}/blocked/date 40
/rest/room/{apiKey}/blocked/date 40
/rest/room/{apiKey}/blocked/date/multiple 41
/rest/room/{apiKey}/blocked/date/multiple 41
/rest/room/{apiKey}/create 41
/rest/room/{apiKey}/create 41
/rest/room/{apiKey}/update/uid 42
/rest/room/{apiKey}/update/uid 42
/rest/room/{apiKey}/cud/uid 42
/rest/room/{apiKey}/cud/uid 43
/rest/room/{apiKey}/{uid}/uid 43
/rest/room/{apiKey}/{uid}/uid 43
HolidayCalendar 44
/rest/holidayCalendar/educationalInstruction/{uid} 44

/rest/holidayCalendar/{apiKey}/date 44
/rest/holidayCalendar/{apiKey}/date 44
/rest/holidayCalendar/{apiKey}/update 44
/rest/holidayCalendar/{apiKey}/date 44
Category 45
/rest/category 45
EventType 45
/rest/eventType 45
Cooperation 45
/rest/cooperation 45
TargetGroup 45
/rest/targetGroup 45
Mail 46
/rest/mail/key 46
/rest/mail/send 46

Einführung

Allgemein	Aufruf der Rest-API

Die Rest-API stellt verschiedene Controller zur Verfügung, die wiederum Methoden nach außen zugänglich machen. Eine solche Methode wird aufgerufen, indem die dafür eindeutig definierte URL angesteuert wird. Diese kann am Beispiel der Abfrage eines Kurses wie folgt aussehen:

http://beispieladresse:beispielport/rest/course/1

Wichtig hierbei ist, dass der Anfragetyp (GET, POST, DELETE) zur Methode passt. Im Beispiel müsste die Anfrage demnach als GET-Anfrage gesendet werden. Innerhalb der Anfragetypen gibt es folgende Unterscheidungen:

- GET-Anfragen werden gestellt, um Daten abzufragen. Im Normalfall werden hierbei benötigte Daten als Parameter in der URL übergeben.
- POST-Anfragen werden gestellt, um Daten zu übermitteln und entsprechend neue Datensätze anzulegen oder bestehende Datensätze zu aktualisieren. Hierbei können Daten ebenfalls als Parameter in der URL übermittelt werden. Allerdings ist die bevorzugte Variante die Übermittlung der Daten im Body der Anfrage. Dabei werden die Daten im ***JSON-Format*** übergeben.
- DELETE-Anfragen werden gestellt, um Daten zu löschen. Wie bei den POST-Anfragen ist sowohl das Übermitteln der Daten als Parameter in der URL als auch das Senden der Daten im Body möglich.

Wie die jeweiligen Parameter der zur Verfügung gestellten Methoden im Detail aussehen, kann im weiteren Verlauf der Dokumentation eingesehen werden.

Allgemein	Response Codes

Die verschiedenen Aufrufe der Rest-API liefern verschiedene Antwortcodes, die davon abhängen, ob bei der Bearbeitung Fehler aufgetreten sind und um welche Fehler es sich handelt. Innerhalb der Anwendung können folgende Response Codes auftreten:

- **200**: Dieser Code bedeutete, dass kein Fehler aufgetreten ist und ein Ergebnis geliefert wurde.
- **404**: Dieser Code wird innerhalb der Anwendung an verschiedenen Stellen genutzt. Er tritt auf, wenn eine Anfrage nicht korrekt ist oder auch, wenn zur Anfrage keine Ressourcen gefunden werden. In jedem Fall wird in der Fehlernachricht der Fehler ausgegeben.
- **501**: Wenn dieser Fehler auftritt, ist bei der Verarbeitung des Aufrufs ein Fehler aufgetreten. Dies kann verschiedene Gründe haben, wie einen falschen Datentyp im Parameter.

Welche Response Codes in welchen Anfragen auftreten, ist für jede Anfrage nochmals aufgeschlüsselt.

Datenobjekte

Objekt	course (Vollständiges Kursobjekt)

Kurse sind der zentrale Teil des Bildungsangebots. Das vollständige Modell enthält dabei alle Informationen, die ein Kurs enthalten kann. Das umfasst zum einen die Daten des Kurses selbst, wie zum Beispiel das Thema, die Start- und Endzeit oder auch Preise. Zum anderen sind Adressdaten und Dozenten enthalten. Letztere werden als vollständiges Objekt geliefert. Selbiges gilt für die Bildungseinrichtung, die Kategorien, die Termine, die Veranstaltungsart, die Kooperationen, und die Zielgruppe. Genauere Informationen zum Aufbau der Unterobjekte sind in den entsprechenden Themenbereichen zu finden.

Objektstruktur:

{

„courseNumber“: „string“,	Kursnummer
„titel“: „string“,	Titel
„description“: „string“,	Beschreibung
„uid“: „int“,	Eindeutige ID des Kurses
„startTime“: „datetime“,	Beginn
„endTime“: „datetime“,	Ende
„externalLink“: „string“,	Externer Link
„learningLink“: „string“,	Learning Link
„price“: „decimal“,	Preis
„examPrice“: „decimal“,	Prüfungspreis
„participantCount“: „int“,	Teilnehmerzahl
„concept“: „string“,	Konzept
„categorys“: [**category**],	Array - Vollständiges Kategorieobjekt
„instructors“: [**instructor**],	Array - Vollständiges Dozentenobjekt
„educationalInstruction“: **educationalInstruction**,	Vollständiges Bildungseinrichtungsobjekt
„meetings“: [**meeting**],	Array - Vollständiges Terminobjekt
„buildingName“: „string“,	Gebäudename
„street“: „string“,	Straße
„zipCode“: „string“,	Stadt
„city“: „string“,	Postleitzahl
„lat“: „decimal“,	Lat
„lng“: „decimal“,	Lng
„totalHours“: „int“,	Gesamtzahl Stunden
„eventType“: **eventType**,	Vollständiges Veranstaltungsartobjekt
„acknowledgement“: „boolean“,	Anerkennung nach dem BFG
„cooperations“: [**cooperation**],	Array - Vollständiges Kooperationsobjekt
„ebg“: boolean,	EBG-Kennzeichen
„targetGroup“: **targetGroup**,	Vollständiges Zielgruppenobjekt
„participantDays“: „int“,	Teilnehmertage
„overnightStays“: „int“,	Übernachtungen
„state“: „int“,	Zustand des Kurses
„hidden“: „boolean“,	Versteckt
„deleted“: „boolean“	Gelöscht

}

Objekt	courseSmall (Abgespecktes Kursobjekt)

Kurse sind der zentrale Teil des Bildungsangebots. Das abgespeckte Modell enthält aber nur die Kerninformationen wir die Kursnummer oder den Titel für Darstellungen in Listen oder ähnlichem.

Objektstruktur:

{

„courseNumber“: „string“,	Kursnummer
„titel“: „string“,	Titel
„description“: „string“,	Beschreibung
„uid“: „int“,	Eindeutige ID des Kurses
„state“: „int“,	Zustand des Kurses
„hidden“: „boolean“,	Versteckt
„deleted“: „boolean“	Gelöscht

}

Objekt	courseEssential (Essentielles Kursobjekt)

Kurse sind der zentrale Teil des Bildungsangebots. Das essentielle Modell enthält dabei die wichtigsten Kursinformationen. Das sind in diesem Fall die Daten des Kurses selbst, wie zum Beispiel das Thema, die Start- und Endzeit oder auch Preise. Unterobjekte oder auswertungsrelevante Daten sind nicht enthalten.

Objektstruktur:

{

„courseNumber“: „string“,	Kursnummer
„titel“: „string“,	Titel
„description“: „string“,	Beschreibung
„uid“: „int“	Eindeutige ID des Kurses
„startTime“: „datetime“,	Beginn
„endTime“: „datetime“,	Ende
„externalLink“: „string“,	Externer Link
„learningLink“: „string“,	Learning Link
„price“: „decimal“,	Preis
„examPrice“: „decimal“,	Prüfungspreis
„participantCount“: „int“,	Teilnehmerzahl
„concept“: „string“,	Konzept
„buildingName“: „string“,	Gebäudename
„street“: „string“,	Straße
„zipCode“: „string“,	Stadt
„city“: „string“,	Postleitzahl
„lat“: „string“,	Lat
„lng“: „string“,	Lng
„totalHours“: „int“,	Gesamtzahl Stunden
„state“: „int“,	Zustand des Kurses
„hidden“: „boolean“,	Versteckt
„deleted“: „boolean“	Gelöscht

}

Objekt	courseMeeting (Vollständiges Kombinationsobjekt aus Kurs und Termin)

Dieses Objekt enthält eine Kombination aus Terminen und den wichtigsten zugehörigen Kursinformationen.

Objektstruktur:

{	
„titel“: „string“,	Titel des Kurses
„courseNumber“: „string“,	Kursnummer
„startTime“: „datetime“,	Startzeit des Termins
„endTime“: „datetime“,	Endzeit des Termins
„instructor“: „string“,	Ein Dozent des Termins
„city“: „string“,	Ort
„room“: „string“	Raum
}	

Objekt	educationalInstruction (Vollständiges Bildungseinrichtungsobjekt)

Die Bildungseinrichtung ist eine Institution, welche einem Bildungsauftrag nachgeht. Das vollständige Modell enthält alle Angaben zur genauen Identifikation der Bildungseinrichtung, wie die Adresse oder Telefonnummer.

Objektstruktur:

{	
„name“: „string“,	Name
„street“: „string“,	Straße
„city“: „string“,	Stadt
„zipCode“: „string“,	Postleitzahl
„lat“: „decimal“,	Fallback Link
„lng“: „decimal“,	Lat
„fallbackLink“: „string“,	Lng
„tel“: „string“,	Telefonnummer
„logo“: „string“,	Logo
„uid“: „int“	Eindeutige ID der Bildungseinrichtung
}	

Objekt	meeting (Vollständiges Terminobjekt)

Termine stellen dar, wann einzelne Kurse stattfinden und geben zusätzlich an, welcher Dozent die Lehrveranstaltung durchgeführt und in welchem Raum sie stattfindet.

Objektstruktur:

{	
“startTime“: „datetime“,	Beginn
„endTime“: „datetime“,	Ende
„hours“: „int“,	Anzahl der Stunden
„break“: „int“,	Pause in Minuten
„instructors“: [**instructor**],	Array - Vollständiges Dozentenobjekt
„room“: **room**,	Vollständiges Raumobjekt
„uid“: „int“	Eindeutige ID des Termins
}	

Objekt	instructor (Vollständiges Dozentenobjekt)

Das Dozentenobjekt dient der Abbildung der lehrenden Dozenten in der Bildungseinrichtung. Hinterlegt sind dabei alle wichtigen Informationen vom Namen über die E-Mail-Adresse bis hin zur Telefonnummer.

Objektstruktur:

```
{
    „instructorNumber“: „string“,        Dozentennummer
    „gender“: „string“,                  Anrede
    „firstName“: „string“,               Vorname
    „lastName“: „string“,                Nachname
    „email“: „string“,                   E-Mail
    „job“: „string“,                     Beruf
    „mobil“: „string“,                   Handynummer
    „tel“: „string“,                     Telefonnummer
    „fax“: „string“,                     Fax
    „workType“: „string“                 Arbeitstyp (Hauptamtlich/Nebenamtlich)
    „uid“: „int“                         Eindeutige ID des Dozenten
}
```

Objekt	meetingInstructor (Vollständiges Kombinationsobjekt aus Termin und Dozenten)

Dieses Objekt stellt eine Kombination aus Termin und Dozenten dar. Das heißt, dass zu dem Dozenten Terminin-formationen abgebildet sind, welche angeben, in welchem Termin der Dozent geplant ist.

Objektstruktur:

```
{
    „instructorNumber“: „string“,        Dozentennummer
    „gender“: „string“,                  Anrede
    „firstName“: „string“,               Vorname
    „lastName“: „string“,                Nachname
    „email“: „string“,                   E-Mail
    „job“: „string“,                     Beruf
    „mobil“: „string“,                   Handynummer
    „tel“: „string“,                     Telefonnummer
    „fax“: „string“,                     Fax
    „workType“: „string“                 Arbeitstyp (Hauptamtlich/Nebenamtlich)
    „uid“: „int“,                        Eindeutige ID des Dozenten
    „meetingStartTime“: „datetime“,      Startzeit des Termins
    „meetingEndTime“: „datetime“,        Endzeit des Termins
    „meetingUid“: „int“                  Eindeutige ID des Termins
}
```

Objekt	room (Vollständiges Raumobjekt)

Die Räume dienen als Lokalität zum Durchführen der verschiedenen Lehrveranstaltungen. Sie enthalten eine Raumnummer, die Addressdaten, Sitzplätze sowie eine Telefonnummer und sind eindeutig durch die Kombination aus Raumnummer und Gebäudename gekennzeichnet.

Objektstruktur:

```
{
    „roomNumber“: „string“,        Raumnummer
    „zipCode“: „string“,           Postleitzahl
    „city“: „string“,              Ort
    „street“: „string“,            Adresse / Straße
    „buildingName“: „string“,      Gebäudename
    „seats“: „string“,             Anzahl Plätze
    „tel“: „string“,               Telefonnummer
    „uid“: „int“                   Eindeutige ID des Raums
}
```

Objekt	meetingRoom (Vollständiges Kombinationsobjekt aus Termin und Raum)

Dieses Objekt stellt eine Kombination aus Termin und Raum dar. Das heißt, dass zu dem Raum Termininformationen abgebildet sind, welche angeben, in welchem Termin der Raum verwendet wird.

Objektstruktur:

```
{
    „roomNumber“: „string“,        Raumnummer
    „zipCode“: „string“,           Postleitzahl
    „city“: „string“,              Ort
    „street“: „string“,            Adresse / Straße
    „buildingName“: „string“,      Gebäudename
    „seats“: „string“,             Anzahl Plätze
    „tel“: „string“,               Telefonnummer
    „uid“: „int“,                  Eindeutige ID des Raums
    „meetingStartTime“: „datetime“,  Startzeit des Termins
    „meetingEndTime“: „datetime“,    Endzeit des Termins
    „meetingUid“: „int“            Eindeutige ID des Termins
}
```

Objekt	address (Vollständiges Adressobjekt)

Die Adressen gehen aus den Räumen hervor und bilden alle einzigartigen Kombinationen der Adressdaten ab.

Objektstruktur:

```
{
    „zipCode“: „string“,           Postleitzahl
    „city“: „string“,              Ort
    „street“: „string“,            Adresse / Straße
    „buildingName“: „string“       Gebäudename
}
```

Objekt	holidayCalendar (Vollständiges Ferienkalenderobjekt)

Das Ferienkalenderobjekt enthält nicht viele Informationen. Es stellt lediglich einen blockierten Tag anhand des entsprechenden Datums und einer eindeutigen ID dar.

Objektstruktur:

```
{
    „date“: „datetime“,        Datumswert
    „uid“: „int“               Eindeutige ID des Kalendereintrags
}
```

Objekt	category (Vollständiges Kategorieobjekt)

Das Kategorieobjekt beschreibt die für einen Kurs hinterlegbaren Kategorien, welche wiederrum als übergeordnete Themengruppen zu verstehen sind. Beispiele sind Sprachen oder auch Politik.

Objektstruktur:

```
{
    „titel“: „string“,         Titel
    „description“: „string“,   Beschreibung
    „tags“: „string“,          Tags
    „uid“: „int“               Eindeutige ID der Kategorie
}
```

Objekt	eventType (Vollständiges Veranstaltungsartobjekt)

Die Veranstaltungsart dient Auswertungszwecken und enthält lediglich eine vorgeschriebene ID und einen Beschreibungstext.

Objektstruktur:

```
{
    “id“: „int“,               ID der Veranstaltungsart für die Auswertung
    „description“: „string“,   Beschreibung
    „uid“: „int“               Eindeutige ID der Veranstaltungsart
}
```

Objekt	cooperation (Vollständiges Kooperationsobjekt)

Das Kooperationsobjekt dient Auswertungszwecken und enthält lediglich eine vorgeschriebene ID und einen Beschreibungstext.

Objektstruktur:

```
{
    “id“: „int“,               ID der Kooperation für die Auswertung
    „description“: „string“,   Beschreibung
    „uid“: „int“               Eindeutige ID der Kooperation
}
```

Objekt	targetGroup (Vollständiges Zielgruppenobjekt)

Die Zielgruppe dient Auswertungszwecken und enthält lediglich eine vorgeschriebene ID und einen Beschreibungstext.

Objektstruktur:

```
{
    “id“: „int“,               ID der Zielgruppe für die Auswertung
    „description“: „string“,   Beschreibung
    „uid“: „int“               Eindeutige ID der Zielgruppe
}
```

Objekt	range (Vollständiges Datumsbereichsobjekt)

Das Datumsbereichsobjekt dient der Abfrage von Dozenten oder Räumen, die in bestimmten Zeiträumen bereits verplant sind. Dafür sind eine Start- und Endzeit sowie ein auszuschließender Termin enthalten.

Objektstruktur:

```
{
    „startTime“: „datetime“,          Startzeitpunkt
    „endTime“: „datetime“,            Endzeitpunkt
    „meetingUid“: „int“               Eindeutige Termin-ID zum Auslassen (optional)
}
```

Objekt	filter (Vollständiges Filterobjekt)

Das Datumsbereichsobjekt dient der Abfrage von Dozenten oder Räumen, die in bestimmten Zeiträumen bereits verplant sind. Dafür sind eine Start- und Endzeit sowie ein auszuschließender Termin enthalten.

Objektstruktur:

```
{
    „searchString“: „string“,                       Suchstring
    „categoryUid“: „int“,                           Eindeutige ID der zu durchsuchenden Kategorie
    „filter“: {                                     Filterobjekt
        „radiusEnabled“: boolean,                       Positionssuche aktivieren
        „location“: {                                   Positionsfilter
            „lat“: „decimal“,                               Lat
            „lng“: „decimal“                                Lng
        },
        „priceEnabled“: „boolean“,                      Preissuche aktivieren
        „price“: {                                      Preisfilter
            „lower“: „decimal“,                             Untere Preisgrenze
            „upper“: „decimal“                              Obere Preisgrenze
        },
        „educationEnabled“: „boolean“,                  Bildungseinrichtungssuche aktivieren
        „activEducationalInstructions“: [„int“],        Liste der IDs der Bildungseinrichtungen
        „dateEnabled“: „boolean“,                       Datumssuche aktivieren
        „start“: „datetime“,                            Startzeitpunkt
        „end“: „datetime“                           Endzeitpunkt
    }
}
```

Course

GET	/rest/course/{uid}

Gibt den Kurs zur angegebenen eindeutigen ID im Parameter {uid} zurück. Gelöschte Kurse sind hierbei ausgeschlossen und werden nicht geliefert.

Parameter:

uid: „int“ — Eindeutige ID des gesuchten Kurses

Beispiel:

uid = 1

Antwort:

Response Code 200: course (Vollständiges Kursobjekt)
Response Code 404: Es wurde kein Kurs mit der angegebenen ID gefunden

GET	/rest/course/{uid}/deleted

Gibt den Kurs zur angegebenen eindeutigen ID im Parameter {uid} zurück. Kurse werden dabei ebenfalls zurückgeliefert, wenn sie gelöscht sind.

Parameter:

uid: „int“ — Eindeutige ID des gesuchten Kurses

Beispiel:

uid = 1

Antwort:

Response Code 200: course (Vollständiges Kursobjekt)
Response Code 404: Es wurde kein Kurs mit der angegebenen ID gefunden

GET	/rest/course/category/{uid}

Gibt alle aktuell aktiven Kurse zur Kategorie mit der angegebenen ID im Parameter {uid} zurück. Gelöschte Kurse sind hierbei ausgeschlossen und werden nicht geliefert.

Parameter:

uid: „int“ — Eindeutige ID der zu durchsuchenden Kategorie

Beispiel:

uid = 1

Antwort:

Response Code 200: [courseSmall] (Array - Abgespecktes Kursobjekt)

GET	/rest/course/category/{uid}/all

Gibt sowohl alle aktuell aktiven sowie alle aktuell inaktiven Kurse zur Kategorie mit der angegebenen ID im Parameter {uid} zurück. Gelöschte Kurse sind hierbei ausgeschlossen und werden nicht geliefert.

Parameter:

uid: „int“ — Eindeutige ID der zu durchsuchenden Kategorie

Beispiel:

uid = 1

Antwort:

Response Code 200: [courseSmall] (Array - Abgespecktes Kursobjekt)

GET	/rest/course/category/{uid}/deleted

Gibt sowohl alle aktuell aktiven sowie alle aktuell inaktiven Kurse zur Kategorie mit der angegebenen ID im Parameter {uid} zurück. Kurse werden dabei ebenfalls zurückgeliefert, wenn sie gelöscht sind.

Parameter:

uid: „int“ Eindeutige ID der zu durchsuchenden Kategorie

Beispiel:

uid = 1

Antwort:

Response Code 200: [courseSmall] (Array - Abgespecktes Kursobjekt)

GET	/rest/course/instructor/{uid}

Gibt alle aktuell aktiven und zum Dozenten mit der angegebenen ID im Parameter {uid} zugeordneten Kurse abgespeckt zurück. Gelöschte Kurse sind hierbei ausgeschlossen und werden nicht geliefert.

Parameter:

uid: „int“ Eindeutige ID des Dozenten

Beispiel:

uid = 1

Antwort:

Response Code 200: [courseSmall] (Array - Abgespecktes Kursobjekt)
Response Code 404: Es wurde kein Dozent zur angegebenen ID gefunden

GET	/rest/course/instructor/{uid}/all

Gibt sowohl alle aktuell aktiven sowie alle aktuell inaktiven und zum Dozenten mit der angegebenen ID im Parameter {uid} zugeordneten Kurse abgespeckt zurück. Gelöschte Kurse sind hierbei ausgeschlossen und werden nicht geliefert.

Parameter:

uid: „int“ Eindeutige ID des Dozenten

Beispiel:

uid = 1

Antwort:

Response Code 200: [courseSmall] (Array - Abgespecktes Kursobjekt)
Response Code 404: Es wurde kein Dozent zur angegebenen ID gefunden

GET	/rest/course/instructor/{uid}/deleted

Gibt sowohl alle aktuell aktiven sowie alle aktuell inaktiven und zum Dozenten mit der angegebenen ID im Parameter {uid} zugeordneten Kurse abgespeckt zurück. Kurse werden dabei ebenfalls zurückgeliefert, wenn sie gelöscht sind.

Parameter:

uid: „int“ Eindeutige ID des Dozenten

Beispiel:

uid = 1

Antwort:

Response Code 200: [courseSmall] (Array - Abgespecktes Kursobjekt)
Response Code 404: Es wurde kein Dozent zur angegebenen ID gefunden

GET	/rest/course/educationalInstruction/{uid}

Gibt alle aktuell aktiven Kurse zur Bildungseinrichtung mit der angegebenen ID im Parameter {uid} abgespeckt zurück. Gelöschte Kurse sind hierbei ausgeschlossen und werden nicht gefunden.

Parameter:

uid: „int“ Eindeutige ID der Bildungseinrichtung

Beispiel:

uid = 1

Antwort:

Response Code 200: [courseSmall] (Array - Abgespecktes Kursobjekt)
Response Code 404: Es wurde keine Bildungseinrichtung zur angegebenen ID gefunden

GET	/rest/course/educationalInstruction/{uid}/all

Gibt sowohl alle aktuell aktiven als auch alle derzeit inaktiven Kurse zur Bildungseinrichtung mit der angegebenen ID im Parameter {uid} abgespeckt zurück. Gelöschte Kurse sind hierbei ausgeschlossen und werden nicht gefunden.

Parameter:

uid: „int“ Eindeutige ID der Bildungseinrichtung

Beispiel:

uid = 1

Antwort:

Response Code 200: [courseSmall] (Array - Abgespecktes Kursobjekt)
Response Code 404: Es wurde keine Bildungseinrichtung zur angegebenen ID gefunden

GET	/rest/course/educationalInstruction/{uid}/deleted

Gibt sowohl alle aktuell aktiven als auch alle derzeit inaktiven Kurse zur Bildungseinrichtung mit der angegebenen ID im Parameter {uid} abgespeckt zurück. Kurse werden dabei ebenfalls zurückgeliefert, wenn sie gelöscht sind.

Parameter:

uid: „int“ Eindeutige ID der Bildungseinrichtung

Beispiel:

uid = 1

Antwort:

Response Code 200: [courseSmall] (Array - Abgespecktes Kursobjekt)
Response Code 404: Es wurde keine Bildungseinrichtung zur angegebenen ID gefunden

GET	/rest/course/educationalInstruction/{uid}/complete

Gibt alle aktuell aktiven Kurse zur Bildungseinrichtung mit der angegebenen ID im Parameter {uid} vollständig zurück. Gelöschte Kurse sind hierbei ausgeschlossen und werden nicht geliefert.

Parameter:

uid: „int“ Eindeutige ID der Bildungseinrichtung

Beispiel:

uid = 1

Antwort:

Response Code 200: [courseEssential] (Array - Essentielles Kursobjekt)
Response Code 404: Es wurde keine Bildungseinrichtung zur angegebenen ID gefunden

GET	/rest/course/educationalInstruction/{uid}/complete/all

Gibt sowohl alle aktuell aktiven als auch alle derzeit inaktiven Kurse zur Bildungseinrichtung mit der angegebenen ID im Parameter {uid} vollständig zurück. Gelöschte Kurse sind hierbei ausgeschlossen und werden nicht geliefert.

Parameter:

uid: „int“ Eindeutige ID der Bildungseinrichtung

Beispiel:

uid = 1

Antwort:

Response Code 200: [courseEssential] (Array - Essentielles Kursobjekt)
Response Code 404: Es wurde keine Bildungseinrichtung zur angegebenen ID gefunden

GET	/rest/course/educationalInstruction/{uid}/complete/deleted

Gibt sowohl alle aktuell aktiven als auch alle derzeit inaktiven Kurse zur Bildungseinrichtung mit der angegebenen ID im Parameter {uid} vollständig zurück. Kurse werden dabei ebenfalls zurückgeliefert, wenn sie gelöscht sind.

Parameter:

uid: „int“ Eindeutige ID der Bildungseinrichtung

Beispiel:

uid = 1

Antwort:

Response Code 200: [courseEssential] (Array - Essentielles Kursobjekt)
Response Code 404: Es wurde keine Bildungseinrichtung zur angegebenen ID gefunden

GET	/rest/course/{apiKey}/switch/{uid}/state/{state}

Setzt den Zustand des über den Parameter {uid} spezifizierten Kurses auf den Zustand im Parameter {state}. Vor der Verwendung der Methode, muss es einen Applikationsschlüssel für die jeweilige Bildungseinrichtung geben

Parameter:

uid: „int“ Eindeutige ID der Bildungseinrichtung

Beispiel:

uid = 1

Antwort:

Response Code 200: [course] (Array - Vollständiges Kursobjekt)
Response Code 404: Kein Kurs zur ID gefunden, keine Bildungseinrichtung für den ApiKey

GET	/rest/course/deleted/init

Sucht alle bereits im System als gelöscht markierten Kurse und initialisiert für alle den Zustand. Als Initialwert wird der Zustand „Beendet“ angenommen.

Parameter: -

Beispiel: -

Antwort:

Response Code 200: -

GET	/rest/course/search/{searchString}

Sucht alle Kurse, die dem Suchstring entsprechen. Hierbei gibt es mehrere Möglichkeiten, wann ein Kurs zurückgeliefert wird. Die erste Möglichkeit ist, dass der Suchstring der Kursnummer entspricht. Alternativ kann der Suchstring auch im Titel oder der Beschreibung enthalten sein. Gelöschte Kurse sind hierbei ausgeschlossen und werden nicht gefunden.

Parameter:

searchString: „string“	Suchanfrage

Beispiel:

searchString = „Mustersuche“

Antwort:

Response Code 200: [courseSmall] (Array - Abgespecktes Kursobjekt)

GET	/rest/course/category/{uid}/search/{searchString}

Sucht alle Kurse innerhalb der Kategorie mit der angegebenen ID im Parameter {uid}, die dem Suchstring entsprechen. Hierbei gibt es mehrere Möglichkeiten, wann ein Kurs zurückgeliefert wird. Die erste Möglichkeit ist, dass der Suchstring der Kursnummer entspricht. Alternativ kann der Suchstring auch im Titel oder der Beschreibung enthalten sein. Gelöschte Kurse sind hierbei ausgeschlossen und werden nicht geliefert.

Parameter:

uid: „int“,	Eindeutige ID der zu durchsuchenden Kategorie
searchString: „string“	Suchanfrage

Beispiel:

uid = 1,
searchString = „Mustersuche“

Antwort:

Response Code 200: [courseSmall] (Array - Abgespecktes Kursobjekt)

POST	/rest/course/{apiKey}/create

Erstellt einen neuen Kurs für die im Parameter {apiKey} spezifizierte Bildungseinrichtung.
Vor der Verwendung der Methode, muss es einen Applikationsschlüssel für die jeweilige Bildungseinrichtung geben. Die Dozenten können auf verschiedene Arten angegeben werden, was sowohl für die Dozenten des Kurses als auch der jeweiligen Termine gilt. Zuerst ist eine Liste von Nummern angebbar, wodurch die Dozenten anhand der Dozentenummern gesucht und zugeordnet werden. Weiterhin ist es möglich, Dozentenobjekte zu übergeben. In diesem Fall wird zuerst geschaut, ob es die Dozentennummer bereits gibt. Falls nicht, wird der Dozent angelegt und dann zugeordnet. Die Räume können ebenso als einfache Nummern oder Objekte angegeben werden. Hierbei ist aber nur die *uid* in den Objekten von Bedeutung. Auch als Nummer wird hierbei die *uid* angegeben, da die Raumnummer nicht eindeutig ist.
Auch die Datumswerte können auf verschiedene Arten angegeben werden (ISO, DIN).
Die Kategorien können als Objekte oder als Schlagworte angegeben werden. Bei der Übergabe als Objekte, wird über die *uid* die entsprechende Kategorie identifiziert und zugeordnet. Andernfalls wird geschaut, ob ein Schlagwort in einer Kategorie vorkommt und danach eine Zuordnung durchgeführt.
Pflichtfelder beim Anlegen sind: Kursnummer, Titel, mindestens eine Kategorie sowie Beginn und Ende.

Parameter:

course (Vollständiges Kursobjekt ohne *uid, deleted* und *educationalInstruction*)

Beispiel: Übergabe von IDs

```
{
    „courseNumber“: „181-4B04“,
    „titel“: „Musterkurs“,
    „description“: „Musterbeschreibung“,
    „startTime“: „2020-02-28 08:00“,
    „endTime“: „2020-02-28 18:00“,
    „externalLink“: „Musterlink“,
    „learningLink“: „Musterlink“,
    „price“: 50.5,
    „examPrice“: 100,
    „participantCount“: 20,
    „concept“: „Musterkonzept“,
    „categorys“: [„Muster1“, „Muster2],
    „instructors“: [67, 204],
    „meetings“: [{
        „startTime“: „2020-02-28 08:00“,
        „endTime“: „2020-02-28 18:00“,
        „hours“: 10,
        „break“: 0,
        „instructors“: [67],
        „room“: 1
    }],
    „zipCode“: „Musterpostleitzahl“,
    „city“: „Musterstadt“,
    „street“: „Musterstraße“,
    „buildingName“: „Mustergebäude“,
    „totalHours“: 10,
    „eventTyp“: 1,
    „acknowledgement“: true,
    „cooperations“: [1, 5],
    …
```

Beispiel: Übergabe von Objekten

```
{
    „courseNumber“: „181-4B04“,
    „titel“: „Musterkurs“,
    „description“: „Musterbeschreibung“,
    „startTime“: „2020-02-28 08:00“,
    „endTime“: „2020-02-28 18:00“,
    „externalLink“: „Musterlink“,
    „learningLink“: „Musterlink“,
    „price“: 50.5,
    „examPrice“: 100,
    „participantCount“: 20,
    „concept“: „Musterkonzept“,
    „categorys“: [{
        „titel“: „Muster“,
        „description“: „Musterbeschreibung“,
        „tags“: „Mustertag“,
        „uid“: 1
    }],
    „instructors“: [{
        „instructorNumber“: „67“,
        „gender“: „M“,
        „firstName“: „Max“,
        „lastName“: „Mustermann“,
        „email“: „Mustermail“,
        „job“: „Musterberuf“,
        „mobil“: „12345“,
        „tel“: „12345“,
        „fax“: „12345“,
        „workType“: „Hauptamtlich“
    }],
    …
```

```
    „ebg“: true,
    „targetGroup“: 1,
    „participantDays“: 2,
    „overnightStays“: 1,
    „state“: 1,
    „hidden“: false
}
```

```
    „meetings“: [{
        „startTime“: „2020-02-28 08:00“,
        „endTime“: „2020-02-28 18:00“,
        „hours“: 10,
        „break“: 0
        „instructors“: [67],
        „room“: {
            „roomNumber“: „101“,
            „zipCode“: „Musterpostleitzahl“,
            „city“: „Musterort“,
            „street“: „Mutserstraße“,
            „buildingName“: „Mustergebäude“,
            „seats“: 1,
            „tel“: „12345“,
            „uid“: 1
        }
    }],
    „buildingName“: „Mustergebäude“,
    „street“: „Musterstraße“,
    „city“: „Musterstadt“,
    „zipCode“: „Musterpostleitzahl“,
    „totalHours“: 10,
    „eventTyp“: 1,
    „acknowledgement“: true,
    „cooperations“: [1, 5],
    „ebg“: true,
    „targetGroup“: 1,
    „participantDays“: 2,
    „overnightStays“: 1,
    „state“: 1,
    „hidden“: false
}
```


Antwort:

Response Code 200: [course] (Array – Vollständiges Kursobjekt)
Response Code 404: Es existiert bereits ein Kurs, fehlende Angaben, keine Bildungseinrichtung für den ApiKey

OPTIONS	/rest/course/{apiKey}/create

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige POST-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der POST-Abfrage aufgerufen.

POST	/rest/course/{apiKey}/create/uid

Erstellt einen neuen Kurs für die im Parameter {apiKey} spezifizierte Bildungseinrichtung.
Vor der Verwendung der Methode, muss es einen Applikationsschlüssel für die jeweilige Bildungseinrichtung geben. Dozenten und Termine können nur als Objekte angegeben werden. Ist eine *uid* enthalten, wird das Objekt angepasst, sonst wird es erstellt und zugeordnet. Für die Dozenten wird dabei noch eine Prüfung der Dozentennummer durchgeführt, da keine zwei Dozenten dieselbe Nummer haben dürfen. Die restlichen Unterobjekte können sowohl als Objekte als auch als *uid* übergeben werden.
Datumswerte können auf verschiedene Arten angegeben werden (ISO, DIN).
Die Kategorien können als Objekte oder als Schlagworte angegeben werden. Bei der Übergabe als Objekte, wird über die *uid* die entsprechende Kategorie identifiziert und zugeordnet. Andernfalls wird geschaut, ob ein Schlagwort in einer Kategorie vorkommt und danach eine Zuordnung durchgeführt.
Pflichtfelder beim Anlegen sind: Kursnummer, Titel, mindestens eine Kategorie sowie Beginn und Ende.

Parameter:

course (Vollständiges Kursobjekt ohne *uid, deleted* und *educationalInstruction*)

Beispiel:

```
{
    „courseNumber“: „181-4B04“,
    „titel“: „Musterkurs“,
    „description“: „Musterbeschreibung“,
    „startTime“: „2020-02-28 08:00“,
    „endTime“: „2020-02-28 18:00“,
    „externalLink“: „Musterlink“,
    „learningLink“: „Musterlink“,
    „price“: 50.5,
    „examPrice“: 100,
    „participantCount: 20,
    „concept“: „Musterkonzept“,
    „categorys“: [{
        „titel“: „Musterkategorie“,
        „description“: „Musterkategorie“,
        „tags“: „Mustertag“
        „uid“: 1
    }],
    „instructors“: [{
        „instructorNumber“: „67“,
        „gender“: „M“,
        „firstName“: „Max“,
        „lastName“: „Mustermann“,
        „email“: „Mustermail“,
        „job“: „Musterberuf“,
        „mobil“: „12345“,
        „tel“: „12345“,
        „fax“: „12345“,
        „workType“: „Hauptamtlich“,
        „uid“: 1
    }],
    …
```

```
    „meetings“: [{
        „startTime“: „2020-02-28 08:00“,
        „endTime“: „2020-02-28 18:00“,
        „hours“: 10,
        „break“: 0,
        „instructors“: [{
            „instructorNumber“: „67“,
            „gender“: „M“,
            „firstName“: „Max“,
            „lastName“: „Mustermann“,
            „email“: „Mustermail“,
            „job“: „Musterberuf“,
            „mobil“: „12345“,
            „tel“: „12345“,
            „fax“: „12345“,
            „workType“: „Hauptamtlich“,
            „uid“: 1
        }],
        „room“: 1,
        „uid“: 1
    }],
    „zipCode“: „Musterpostleitzahl“,
    „city“: „Musterstadt“,
    „street“: „Musterstraße“,
    „buildingName“: „Mustergebäude“,
    „totalHours“: 10,
    „eventTyp“: {
        „description“: „Musterveranstaltung“,
        „uid“: 1
    },
    „acknowledgement“: true,
    „cooperations“: [{
        „description“: „Musterkooperation“,
        „uid“: 1
    }],
    „ebg“: true,
    „targetGroup“: {
        „description“: „Musterzielgruppe“,
        „uid“: 1
    },
    „participantDays“: 2,
    „overnightStays“: 1,
    „state“: 1,
    „hidden“: false
}
```


Antwort:

Response Code 200: course (Vollständiges Kursobjekt)
Response Code 404: Es existiert bereits ein Kurs, fehlende Angaben, keine Bildungseinrichtung für den ApiKey

OPTIONS	/rest/course/{apiKey}/create/uid

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige POST-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der POST-Abfrage aufgerufen.

POST	/rest/course/{apiKey}/update

Aktualisiert einen Kurs für die im Parameter {apiKey} spezifizierte Bildungseinrichtung anhand der Kursnummer. Vor der Verwendung der Methode, muss es einen Applikationsschlüssel für die jeweilige Bildungseinrichtung geben. Die Dozenten können auf verschiedene Arten angegeben werden, was sowohl für die Dozenten des Kurses als auch der jeweiligen Termine gilt. Zuerst ist eine Liste von Nummern angebbar, wodurch die Dozenten anhand der Dozentenummern gesucht und zugeordnet werden. Weiterhin ist es möglich, Dozentenobjekte zu übergeben. In diesem Fall wird zuerst geschaut, ob es die Dozentennummer bereits gibt. Falls nicht, wird der Dozent angelegt und dann zugeordnet. Die Räume können ebenso als einfache Nummern oder Objekte angegeben werden. Hierbei ist aber nur die *uid* in den Objekten von Bedeutung. Auch als Nummer wird hierbei die *uid* angegeben, da die Raumnummer nicht eindeutig ist.
Auch die Datumswerte können auf verschiedene Arten angegeben werden (ISO, DIN).
Die Kategorien können als Objekte oder als Schlagworte angegeben werden. Bei der Übergabe als Objekte, wird über die *uid* die entsprechende Kategorie identifiziert und zugeordnet. Andernfalls wird geschaut, ob ein Schlagwort in einer Kategorie vorkommt und danach eine Zuordnung durchgeführt.
Pflichtfelder beim Anlegen sind: Kursnummer, Titel, mindestens eine Kategorie sowie Beginn und Ende.

Parameter:

course (Vollständiges Kursobjekt ohne *uid*, *deleted* und *educationalInstruction*)

Beispiel: Übergabe von IDs

```
{
    „courseNumber“: „181-4B04“,
    „titel“: „Musterkurs“,
    „description“: „Musterbeschreibung“,
    „startTime“: „2020-02-28 08:00“,
    „endTime“: „2020-02-28 18:00“,
    „externalLink“: „Musterlink“,
    „learningLink“: „Musterlink“,
    „price“: 50.5,
    „examPrice“: 100,
    „participantCount“: 20,
    „concept“: „Musterkonzept“,
    „categorys“: [„Muster1“, „Muster2],
    „instructors“: [67, 204],
    „meetings“: [{
        „startTime“: „2020-02-28 08:00“,
        „endTime“: „2020-02-28 18:00“,
        „hours“: 10,
        „break“: 0,
        „instructors“: [67],
        „room“: 1
    }],
    „zipCode“: „Musterpostleitzahl“,
    „city“: „Musterstadt“,
    „street“: „Musterstraße“,
    „buildingName“: „Mustergebäude“,
    „totalHours“: 10,
    „eventTyp“: 1,
    „acknowledgement“: true,
    „cooperations“: [1, 5],
    …
```

Beispiel: Übergabe von Objekten

```
{
    „courseNumber“: „181-4B04“,
    „titel“: „Musterkurs“,
    „description“: „Musterbeschreibung“,
    „startTime“: „2020-02-28 08:00“,
    „endTime“: „2020-02-28 18:00“,
    „externalLink“: „Musterlink“,
    „learningLink“: „Musterlink“,
    „price“: 50.5,
    „examPrice“: 100,
    „participantCount“: 20,
    „concept“: „Musterkonzept“,
    „categorys“: [{
        „titel“: „Muster“,
        „description“: „Musterbeschreibung“,
        „tags“: „Mustertag“,
        „uid“: 1
    }],
    „instructors“: [{
        „instructorNumber“: „67“,
        „gender“: „M“,
        „firstName“: „Max“,
        „lastName“: „Mustermann“,
        „email“: „Mustermail“,
        „job“: „Musterberuf“,
        „mobil“: „12345“,
        „tel“: „12345“,
        „fax“: „12345“,
        „workType“: „Hauptamtlich“
    }],
    …
```

```
    „ebg“: true,
    „targetGroup“: 1,
    „participantDays“: 2,
    „overnightStays“: 1,
    „state“: 1,
    „hidden“: false
}
```

```
    „meetings“: [{
        „startTime“: „2020-02-28 08:00“,
        „endTime“: „2020-02-28 18:00“,
        „hours“: 10,
        „break“: 0
        „instructors“: [67],
        „room“: {
            „roomNumber“: „101“,
            „zipCode“: „Musterpostleitzahl“,
            „city“: „Musterort“,
            „street“: „Mutserstraße“,
            „buildingName“: „Mustergebäude“,
            „seats“: 1,
            „tel“: „12345“,
            „uid“: 1
        }
    }],
    „buildingName“: „Mustergebäude“,
    „street“: „Musterstraße“,
    „city“: „Musterstadt“,
    „zipCode“: „Musterpostleitzahl“,
    „totalHours“: 10,
    „eventTyp“: 1,
    „acknowledgement“: true,
    „cooperations“: [1, 5],
    „ebg“: true,
    „targetGroup“: 1,
    „participantDays“: 2,
    „overnightStays“: 1,
    „state“: 1,
    „hidden“: false
}
```


Antwort:

Response Code 200: course (Vollständiges Kursobjekt)
Response Code 404: Kein Kurs gefunden, fehlende Angaben, keine Bildungseinrichtung für den ApiKey

OPTIONS	/rest/course/{apiKey}/update

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige POST-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der POST-Abfrage aufgerufen.

POST	/rest/course/{apiKey}/update/uid

Aktualisiert einen Kurs für die im Parameter {apiKey} spezifizierte Bildungseinrichtung anhand der *uid*.
Vor der Verwendung der Methode, muss es einen Applikationsschlüssel für die jeweilige Bildungseinrichtung geben. Dozenten und Termine können nur als Objekte angegeben werden. Ist eine *uid* enthalten, wird das Objekt angepasst, sonst wird es erstellt und zugeordnet. Für die Dozenten wird dabei noch eine Prüfung der Dozentennummer durchgeführt, da keine zwei Dozenten dieselbe Nummer haben dürfen. Die restlichen Unterobjekte können sowohl als Objekte als auch als *uid* übergeben werden.
Datumswerte können auf verschiedene Arten angegeben werden (ISO, DIN).
Die Kategorien können als Objekte oder als Schlagworte angegeben werden. Bei der Übergabe als Objekte, wird über die *uid* die entsprechende Kategorie identifiziert und zugeordnet. Andernfalls wird geschaut, ob ein Schlagwort in einer Kategorie vorkommt und danach eine Zuordnung durchgeführt.
Pflichtfelder beim Anlegen sind: Kursnummer, Titel, mindestens eine Kategorie sowie Beginn und Ende.

Parameter:

course (Vollständiges Kursobjekt ohne *deleted* und *educationalInstruction*)

Beispiel:

```
{
    „courseNumber“: „181-4B04“,
    „titel“: „Musterkurs“,
    „description“: „Musterbeschreibung“,
    „startTime“: „2020-02-28 08:00“,
    „endTime“: „2020-02-28 18:00“,
    „externalLink“: „Musterlink“,
    „learningLink“: „Musterlink“,
    „price“: 50.5,
    „examPrice“: 100,
    „participantCount: 20,
    „concept“: „Musterkonzept“,
    „categorys“: [{
        „titel“: „Musterkategorie“,
        „description“: „Musterkategorie“,
        „tags“: „Mustertag“
        „uid“: 1
    }],
    „instructors“: [{
        „instructorNumber“: „67“,
        „gender“: „M“,
        „firstName“: „Max“,
        „lastName“: „Mustermann“,
        „email“: „Mustermail“,
        „job“: „Musterberuf“,
        „mobil“: „12345“,
        „tel“: „12345“,
        „fax“: „12345“,
        „workType“: „Hauptamtlich“,
        „uid“: 1
    }],
    …
```

```
    „meetings“: [{
        „startTime“: „2020-02-28 08:00“,
        „endTime“: „2020-02-28 18:00“,
        „hours“: 10,
        „break“: 0,
        „instructors“: [{
            „instructorNumber“: „67“,
            „gender“: „M“,
            „firstName“: „Max“,
            „lastName“: „Mustermann“,
            „email“: „Mustermail“,
            „job“: „Musterberuf“,
            „mobil“: „12345“,
            „tel“: „12345“,
            „fax“: „12345“,
            „workType“: „Hauptamtlich“,
            „uid“: 1
        }],
        „room“: 1,
        „uid“: 1
    }],
    „zipCode“: „Musterpostleitzahl“,
    „city“: „Musterstadt“,
    „street“: „Musterstraße“,
    „buildingName“: „Mustergebäude“,
    „totalHours“: 10,
    „eventTyp“: {
        „description“: „Musterveranstaltung“,
        „uid“: 1
    },
    „acknowledgement“: true,
    „cooperations“: [{
        „description“: „Musterkooperation“,
        „uid“: 1
    }],
    „ebg“: true,
    „targetGroup“: {
        „description“: „Musterzielgruppe“,
        „uid“: 1
    },
    „participantDays“: 2,
    „overnightStays“: 1,
    „state“: 1,
    „hidden“: false,
    „uid“: 1
}
```


Antwort:

Response Code 200: course (Vollständiges Kursobjekt)
Response Code 404: Kein Kurs gefunden, fehlende Angaben, keine Bildungseinrichtung für den ApiKey

OPTIONS	/rest/course/{apiKey}/update/uid

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige POST-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der POST-Abfrage aufgerufen.

POST	/rest/course/{apiKey}/import

Importiert eine Menge an Kursen für die über den Parameter {apiKey} spezifizierte Bildungseinrichtung anhand der Kursnummer. Bereits vorhandene Kurse werden aktualisiert und neue Kurse angelegt.
Vor der Verwendung der Methode, muss es einen Applikationsschlüssel für die jeweilige Bildungseinrichtung geben. Die Dozenten können auf verschiedene Arten angegeben werden, was sowohl für die Dozenten des Kurses als auch der jeweiligen Termine gilt. Zuerst ist eine Liste von Nummern angebbar, wodurch die Dozenten anhand der Dozentenummern gesucht und zugeordnet werden. Weiterhin ist es möglich, Dozentenobjekte zu übergeben. In diesem Fall wird zuerst geschaut, ob es die Dozentennummer bereits gibt. Falls nicht, wird der Dozent angelegt und dann zugeordnet. Die Räume können ebenso als einfache Nummern oder Objekte angegeben werden. Hierbei ist aber nur die *uid* in den Objekten von Bedeutung. Auch als Nummer wird hierbei die *uid* angegeben, da die Raumnummer nicht eindeutig ist.
Auch die Datumswerte können auf verschiedene Arten angegeben werden (ISO, DIN).
Die Kategorien können als Objekte oder als Schlagworte angegeben werden. Bei der Übergabe als Objekte, wird über die *uid* die entsprechende Kategorie identifiziert und zugeordnet. Andernfalls wird geschaut, ob ein Schlagwort in einer Kategorie vorkommt und danach eine Zuordnung durchgeführt.
Pflichtfelder beim Anlegen sind: Kursnummer, Titel, mindestens eine Kategorie sowie Beginn und Ende.

Parameter:

[course] (Array - Vollständiges Kursobjekt ohne *uid*, *deleted* und *educationalInstruction*)

Beispiel: Übergabe von IDs

```
[{
    „courseNumber“: „181-4B04“,
    „titel“: „Musterkurs“,
    „description“: „Musterbeschreibung“,
    „startTime“: „2020-02-28 08:00“,
    „endTime“: „2020-02-28 18:00“,
    „externalLink“: „Musterlink“,
    „learningLink“: „Musterlink“,
    „price“: 50.5,
    „examPrice“: 100,
    „participantCount“: 20,
    „concept“: „Musterkonzept“,
    „categorys“: [„Muster1“, „Muster2],
    „instructors“: [67, 204],
    „meetings“: [{
        „startTime“: „2020-02-28 08:00“,
        „endTime“: „2020-02-28 18:00“,
        „hours“: 10,
        „break“: 0,
        „instructors“: [67],
        „room“: 1
    }],
    „zipCode“: „Musterpostleitzahl“,
    „city“: „Musterstadt“,
    „street“: „Musterstraße“,
    „buildingName“: „Mustergebäude“,
    „totalHours“: 10,
    „eventTyp“: 1,
    „acknowledgement“: true,
    „cooperations“: [1, 5],
    …
```

Beispiel: Übergabe von Objekten

```
[{
    „courseNumber“: „181-4B04“,
    „titel“: „Musterkurs“,
    „description“: „Musterbeschreibung“,
    „startTime“: „2020-02-28 08:00“,
    „endTime“: „2020-02-28 18:00“,
    „externalLink“: „Musterlink“,
    „learningLink“: „Musterlink“,
    „price“: 50.5,
    „examPrice“: 100,
    „participantCount“: 20,
    „concept“: „Musterkonzept“,
    „categorys“: [{
        „titel“: „Muster“,
        „description“: „Musterbeschreibung“,
        „tags“: „Mustertag“,
        „uid“: 1
    }],
    „instructors“: [{
        „instructorNumber“: „67“,
        „gender“: „M“,
        „firstName“: „Max“,
        „lastName“: „Mustermann“,
        „email“: „Mustermail“,
        „job“: „Musterberuf“,
        „mobil“: „12345“,
        „tel“: „12345“,
        „fax“: „12345“,
        „workType“: „Hauptamtlich“
    }],
    …
```

```
    „ebg“: true,
    „targetGroup“: 1,
    „participantDays“: 2,
    „overnightStays“: 1,
    „state“: 1,
    „hidden“: false
}]
```

```
    „meetings“: [{
        „startTime“: „2020-02-28 08:00“,
        „endTime“: „2020-02-28 18:00“,
        „hours“: 10,
        „break“: 0
        „instructors“: [67],
        „room“: {
            „roomNumber“: „101“,
            „zipCode“: „Musterpostleitzahl“,
            „city“: „Musterort“,
            „street“: „Mutserstraße“,
            „buildingName“: „Mustergebäude“,
            „seats“: 1,
            „tel“: „12345“,
            „uid“: 1
        }
    }],
    „buildingName“: „Mustergebäude“,
    „street“: „Musterstraße“,
    „city“: „Musterstadt“,
    „zipCode“: „Musterpostleitzahl“,
    „totalHours“: 10,
    „eventTyp“: 1,
    „acknowledgement“: true,
    „cooperations“: [1, 5],
    „ebg“: true,
    „targetGroup“: 1,
    „participantDays“: 2,
    „overnightStays“: 1,
    „state“: 1,
    „hidden“: false
}]
```


Antwort:

Response Code 200: Liste der Fehlermeldungen
Response Code 404: Keine Bildungseinrichtung für den ApiKey

OPTIONS	/rest/course/{apiKey}/import

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige POST-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der POST-Abfrage aufgerufen.

POST	/rest/course/{apiKey}/import/delete

Importiert eine Menge an Kursen für die über den Parameter {apiKey} spezifizierte Bildungseinrichtung anhand der Kursnummer. Bereits vorhandene Kurse werden aktualisiert und neue Kurse angelegt. Nicht enthaltene Kurse werden aus dem System entfernt.
Vor der Verwendung der Methode, muss es einen Applikationsschlüssel für die jeweilige Bildungseinrichtung geben. Die Dozenten können auf verschiedene Arten angegeben werden, was sowohl für die Dozenten des Kurses als auch der jeweiligen Termine gilt. Zuerst ist eine Liste von Nummern angebbar, wodurch die Dozenten anhand der Dozentenummern gesucht und zugeordnet werden. Weiterhin ist es möglich, Dozentenobjekte zu übergeben. In diesem Fall wird zuerst geschaut, ob es die Dozentennummer bereits gibt. Falls nicht, wird der Dozent angelegt und dann zugeordnet. Die Räume können ebenso als einfache Nummern oder Objekte angegeben werden. Hierbei ist aber nur die *uid* in den Objekten von Bedeutung. Auch als Nummer wird hierbei die *uid* angegeben, da die Raumnummer nicht eindeutig ist.
Auch die Datumswerte können auf verschiedene Arten angegeben werden (ISO, DIN).
Die Kategorien können als Objekte oder als Schlagworte angegeben werden. Bei der Übergabe als Objekte, wird über die *uid* die entsprechende Kategorie identifiziert und zugeordnet. Andernfalls wird geschaut, ob ein Schlagwort in einer Kategorie vorkommt und danach eine Zuordnung durchgeführt.
Pflichtfelder beim Anlegen sind: Kursnummer, Titel, mindestens eine Kategorie sowie Beginn und Ende.

Parameter:

[**course**] (Array - Vollständiges Kursobjekt ohne *uid*, *deleted* und *educationalInstruction*)

Beispiel: Übergabe von IDs

```
[{
    „courseNumber“: „181-4B04“,
    „titel“: „Musterkurs“,
    „description“: „Musterbeschreibung“,
    „startTime“: „2020-02-28 08:00“,
    „endTime“: „2020-02-28 18:00“,
    „externalLink“: „Musterlink“,
    „learningLink“: „Musterlink“,
    „price“: 50.5,
    „examPrice“: 100,
    „participantCount“: 20,
    „concept“: „Musterkonzept“,
    „categorys“: [„Muster1“, „Muster2],
    „instructors“: [67, 204],
    „meetings“: [{
        „startTime“: „2020-02-28 08:00“,
        „endTime“: „2020-02-28 18:00“,
        „hours“: 10,
        „break“: 0,
        „instructors“: [67],
        „room“: 1
    }],
    „zipCode“: „Musterpostleitzahl“,
    „city“: „Musterstadt“,
    „street“: „Musterstraße“,
    „buildingName“: „Mustergebäude“,
    „totalHours“: 10,
    „eventTyp“: 1,
    „acknowledgement“: true,
    „cooperations“: [1, 5],
    …
```

Beispiel: Übergabe von Objekten

```
[{
    „courseNumber“: „181-4B04“,
    „titel“: „Musterkurs“,
    „description“: „Musterbeschreibung“,
    „startTime“: „2020-02-28 08:00“,
    „endTime“: „2020-02-28 18:00“,
    „externalLink“: „Musterlink“,
    „learningLink“: „Musterlink“,
    „price“: 50.5,
    „examPrice“: 100,
    „participantCount“: 20,
    „concept“: „Musterkonzept“,
    „categorys“: [{
        „titel“: „Muster“,
        „description“: „Musterbeschreibung“,
        „tags“: „Mustertag“,
        „uid“: 1
    }],
    „instructors“: [{
        „instructorNumber“: „67“,
        „gender“: „M“,
        „firstName“: „Max“,
        „lastName“: „Mustermann“,
        „email“: „Mustermail“,
        „job“: „Musterberuf“,
        „mobil“: „12345“,
        „tel“: „12345“,
        „fax“: „12345“,
        „workType“: „Hauptamtlich“
    }],
    …
```

```
    „ebg“: true,
    „targetGroup“: 1,
    „participantDays“: 2,
    „overnightStays“: 1,
    „state“: 1,
    „hidden“: false
}]
```

```
    „meetings“: [{
        „startTime“: „2020-02-28 08:00“,
        „endTime“: „2020-02-28 18:00“,
        „hours“: 10,
        „break“: 0
        „instructors“: [67],
        „room“: {
            „roomNumber“: „101“,
            „zipCode“: „Musterpostleitzahl“,
            „city“: „Musterort“,
            „street“: „Mutserstraße“,
            „buildingName“: „Mustergebäude“,
            „seats“: 1,
            „tel“: „12345“,
            „uid“: 1
        }
    }],
    „buildingName“: „Mustergebäude“,
    „street“: „Musterstraße“,
    „city“: „Musterstadt“,
    „zipCode“: „Musterpostleitzahl“,
    „totalHours“: 10,
    „eventTyp“: 1,
    „acknowledgement“: true,
    „cooperations“: [1, 5],
    „ebg“: true,
    „targetGroup“: 1,
    „participantDays“: 2,
    „overnightStays“: 1,
    „state“: 1,
    „hidden“: false
}]
```


Antwort:

Response Code 200: Liste der Fehlermeldungen
Response Code 404: Keine Bildungseinrichtung für den ApiKey

OPTIONS	/rest/course/{apiKey}/import/delete

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige POST-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der POST-Abfrage aufgerufen.

POST	/rest/course/{apiKey}/restore/uid

Stellt einen gelöschten Kurs für die im Parameter {apiKey} spezifizierte Bildungseinrichtung wieder her. Vor der Verwendung der Methode, muss es einen Applikationsschlüssel für die jeweilige Bildungseinrichtung geben. Das Übergabeobjekt basiert auf einem Kurs, allerdings sind nur eine *uid*, die Kursnummer, ein Enddatum und die Termine, die ebenfalls wiederhergestellt werden sollen, relevant. Für das Wiederherstellen der Termine reicht dabei ebenfalls eine *uid*, es können aber auch vollständige Terminobjekte übergeben werden.
Das Enddatum kann auf verschiedene Arten angegeben werden (ISO, DIN).

Parameter:

```
{
    „courseNumber“: „string“,        Kursnummer
    „uid“: „int“,                    Eindeutige ID des Kurses
    „endTime“: „datetime“,           Ende
    „meetings“: [{                   Liste der IDs der Termine
        „uid“: „int“                     Eindeutige ID des Termins
    }]
}
```

Beispiel:

```
{
    „courseNumber“: „181-4B04“,
    „uid“: 1,
    „endTime“: „2020-02-28 18:00“,
    „meetings“: [{
        „uid“: 1
    }]
}
```

Antwort:

Response Code 200: [course] (Array – Vollständiges Kursobjekt)
Response Code 404: Es existiert bereits ein Kurs, keine Bildungseinrichtung für den ApiKey

OPTIONS	/rest/course/{apiKey}/restore/uid

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige POST-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der POST-Abfrage aufgerufen.

POST	/rest/course/filter

Durchsucht die Kurse anhand des übergebenen Filters. Hierbei können verschiedene Angaben getätigt werden. Ist nur ein Suchstring angegeben, werden alle Kurse gesucht, in denen der Suchstring der Kursnummer entspricht. Alternativ kann der Suchstring auch im Titel oder der Beschreibung des Kurses enthalten sein. Ist eine ID angegeben, werden die Kurse zu der Kategorie mit der angegebenen ID gesucht. Ist ein Filter angegeben, wird basierend auf diesem Filterobjekt gesucht. Es ist hierbei auch möglich, Kombinationen aller Suchmöglichkeiten anzuwenden. Wichtig bei der Angabe von Filtern ist, welche Angaben zusammenhängen:

- Ist die Positionssuche aktiviert, müssen auch die Breiten- und Längengrade angegeben werden.
- Ist die Preissuche aktiviert, müssen auch die untere und obere Grenze angegeben werden.
- Ist die Suche nach einer Bildungseinrichtung aktiviert, muss auch mindestens eine ID in der Liste mitgegeben werden.
- Ist die Datumssuche aktiviert, müssen auch ein Start- und ein Enddatum angegeben werden.

Parameter:

filter (Vollständiges Filterobjekt)

Beispiel: Suche mit Suchstring

```
{
    „searchString“: „181-4B04“
}
```

Beispiel: Suche nach Kategorie

```
{
    „categoryUid“: 1
}
```

Beispiel: Suche mit Positionsfilter

```
{
    „filter“: {
        „radiusEnabled“: true,
        „location“: {
            „lat“: 45,
            „lng“: 45
        }
    }
}
```

Beispiel: Suche mit Preisfilter

```
{
    „filter“: {
        „priceEnabled“: true,
        „price“: {
            „lower“: 0,
            „upper“: 100
        }
    }
}
```

Beispiel: Suche mit Datumsfilter

```
{
    „filter“: {
        „dateEnabled“: true,
        „start“: „1970-01-01T00:00“,
        „end“: „1970-01-01Z00:00“
    }
}
```

Beispiel: Suche nach Bildungseinrichtungen

```
{
    „filter“: {
        „educationEnabled“: true,
        „activEducationalInstructions“: [1]
    }
}
```

Beispiel: Kombinierte Suche 1

```
{
    „searchString“: „181-4B04“,
    „filter“: {
        „priceEnabled“: true,
        „price“: {
            „lower“: 0,
            „upper“: 100
        }
    }
}
```

Beispiel: Kombinierte Suche 2

```
{
    „categoryUid“: 1,
    „filter“: {
        „educationEnabled“: true,
        „activEducationalInstructions“: [1],
        „dateEnabled“: true,
        „start“: "1970-01-01T00:00“,
        „end“: „1970-01-01Z00:00“
    }
}
```

Antwort:

Response Code 200: [{
    „courseNumber“: „string“,
    „titel“: „string“,
    „description“: „string“,
    „uid“: „int“
}]

Rückgabe bei Erfolg
- Kursnummer
- Titel
- Beschreibung
- Eindeutige ID des Kurses

OPTIONS	/rest/course/filter

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige POST-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der POST-Abfrage aufgerufen.

DELETE	/rest/course/{apiKey}/{courseNumber}

Löscht den Kurs mit der im Parameter {courseNumber} übergebenen Kursnummer für die über den Parameter {apiKey} spezifizierte Bildungseinrichtung. Außerdem werden die Termine gelöscht, die dem Kurs zugeordnet sind.

Parameter:

„apiKey“ = „string“,	Applikationsschlüssel der Bildungseinrichtung
„courseNumber“ = „string“	Kursnummer

Beispiel:

„apiKey“ = „Mustereinreichtung“,
„courseNumber“ = „181-4B04“

Antwort:

Response Code 200: -
Response Code 404: Es gibt keinen Kurs zur Nummer, keine Bildungseinrichtung für den ApiKey

OPTIONS	/rest/course/{apiKey}/{courseNumber}

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige DELETE-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der DELETE -Abfrage aufgerufen.

DELETE	/rest/course/{apiKey}/{uid}/uid

Löscht den Kurs mit der im Parameter {uid} übergebenen ID. Außerdem werden die Termine gelöscht, die dem Kurs zugeordnet sind.

Parameter:

„apiKey“ = „string“,	Applikationsschlüssel der Bildungseinrichtung
„uid“ = „integer“	Eindeutige ID

Beispiel:

„apiKey“ = „Mustereinreichtung“,
„uid“ = „1“

Antwort:

Response Code 200: -
Response Code 404: Es gibt keinen Kurs zur Nummer, keine Bildungseinrichtung für den ApiKey

OPTIONS	/rest/course/{apiKey}/{uid}/uid

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige DELETE-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der DELETE -Abfrage aufgerufen.

EducationalInstruction

GET	/rest/educationalInstruction
Gibt alle Bildungseinrichtungen zurück.	
Parameter: -	
Beispiel: -	
Antwort: **Response Code 200**: [**educationalInstruction**] (Array - Vollständiges Bildungseinrichtungsobjekt)	

GET	/rest/educationalInstruction/{apiKey}
Gibt den Name und die eindeutige ID für die über den Parameter {apiKey} spezifizierte Bildungseinrichtung zurück.	
Parameter: apiKey = „string“	Applikationsschlüssel der Bildungseinrichtung
Beispiel: apiKey = „Musterkey“	
Antwort: **Response Code 200**: **educationalInstruction** (Vollständiges Bildungseinrichtungsobjekt)	

Meeting

GET	/rest/meeting
Gibt alle Termine mit Kursinformationen zurück.	
Parameter: -	
Beispiel: -	
Antwort: **Response Code 200**: [courseMeeting] (Array - Vollständiges Kombinationsobjekt aus Kurs und Termin)	

GET	/rest/meeting/educationalInstruction/{uid}
Gibt die Termine zur im Parameter {uid} spezifizierten Bildungseinrichtung mit Kursinformationen zurück.	
Parameter: uid = „int“	Eindeutige ID der Bildungseinrichtung
Beispiel: uid = 1	
Antwort: **Response Code 200**: [courseMeeting] (Array - Vollständiges Kombinationsobjekt aus Kurs und Termin)	

GET	/rest/meeting/room/{uid}
Gibt die in der Zukunft liegenden Termine mit Kursinformationen zurück, die in dem im Parameter {uid} spezifizierten Raum stattfinden. Es werden keine Termine zurückgegeben, die in gelöschten oder versteckten Kursen hinterlegt sind.	
Parameter: uid = „int“	Eindeutige ID des Raums
Beispiel: uid = 1	
Antwort: **Response Code 200**: [courseMeeting] (Array - Vollständiges Kombinationsobjekt aus Kurs und Termin)	

GET	/rest/meeting/room/{uid}/hidden
Gibt die in der Zukunft liegenden Termine mit Kursinformationen zurück, die in dem im Parameter {uid} spezifizierten Raum stattfinden. Es werden keine Termine zurückgegeben, die in gelöschten Kursen hinterlegt sind. Termine in versteckten Kursen werden zurückgegeben.	
Parameter: uid = „int“	Eindeutige ID des Raums
Beispiel: uid = 1	
Antwort: **Response Code 200**: [courseMeeting] (Array – Vollständiges Kombinationsobjekt aus Kurs und Termin)	

GET	/rest/meeting/instructor/{uid}
Gibt die in der Zukunft liegenden Termine mit Kursinformationen zurück, die von dem im Parameter {uid} spezifizierten Dozenten angeboten werden. Es werden keine Termine zurückgegeben, die in gelöschten oder versteckten Kursen hinterlegt sind.	
Parameter: uid = „int“	Eindeutige ID des Raums
Beispiel: uid = 1	
Antwort: **Response Code 200**: [courseMeeting] (Array - Vollständiges Kombinationsobjekt aus Kurs und Termin)	

GET	/rest/instructor/room/{uid}/hidden
Gibt die in der Zukunft liegenden Termine mit Kursinformationen zurück, die von dem im Parameter {uid} spezifizierten Dozenten angeboten werden. Es werden keine Termine zurückgegeben, die in gelöschten Kursen hinterlegt sind. Termine in versteckten Kursen werden zurückgegeben.	
Parameter: uid = „int“	Eindeutige ID des Raums
Beispiel: uid = 1	
Antwort: **Response Code 200**: [courseMeeting] (Array - Vollständiges Kombinationsobjekt aus Kurs und Termin)	

Instructor

GET	/rest/instructor/educationalInstruction/{uid}

Gibt alle Dozenten zur Bildungseinrichtung mit der angegebenen ID im Parameter {uid} zurück.

Parameter:

uid: „int“ Eindeutige ID der Bildungseinrichtung

Beispiel:

uid = 1

Antwort:

Response Code 200: [instructor] (Array - Vollständiges Dozentenobjekt)

POST	/rest/instructor/{apiKey}/blocked/date

Sucht innerhalb der über den Parameter {apiKey} spezifizierten Bildungseinrichtung nach Dozenten, die im übergebenen Zeitraum bereits verplant sind, und gibt diese mit Informationen zu den jeweiligen Terminen zurück. Startet eine Planung dabei genau zum übergebenen Endzeitpunkt, gilt der Dozent nicht als verplant. Selbiges gilt für den Startzeitpunt. Wird über den Parameter *meetingUid* ein Termin spezifiziert, wird in der Ergebnismenge kein Dozent dieses Termins geliefert.
Bevor diese Methode verwendet werden kann, muss zuerst ein Applikationsschlüssel für die jeweilige Bildungseinrichtung hinterlegt werden.

Parameter:

[range] (Array - Vollständiges Datumsbereichsobjekt)

Beispiel:

```
{
    „startTime“: „2020-02-28 16:00“,
    „endTime“: „2020-02-28 18:00“,
    „meetingUid“: 1
}
```

Antwort:

Response Code 200: [meetingInstructor] (Array - Vollständiges Kombinationsobjekt aus Termin und Dozenten)

OPTIONS	/rest/instructor/{apiKey}/blocked/date

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige POST-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der POST-Abfrage aufgerufen.

POST	/rest/instructor/{apiKey}/blocked/date/multiple

Sucht innerhalb der über den Parameter {apiKey} spezifizierten Bildungseinrichtung nach Räumen, die in einem der übergebenen Zeiträume bereits belegt sind, und gibt diese mit Informationen zu den jeweiligen Terminen zurück. Startet eine Belegung dabei genau zum übergebenen Endzeitpunkt, gilt der Raum nicht als belegt. Selbiges gilt für den Startzeitpunt. Wird über den Parameter *meetingUid* ein Termin spezifiziert, wird in der Ergebnismenge kein Raum dieses Termins geliefert.
Bevor diese Methode verwendet werden kann, muss zuerst ein Applikationsschlüssel für die jeweilige Bildungseinrichtung hinterlegt werden.

Parameter:

[range] (Array - Vollständiges Datumsbereichsobjekt)

Beispiel:

```
[{
    „startTime“: „2020-02-28 16:00“,
    „endTime“: „2020-02-28 18:00“,
    „meetingUid“: 1
}]
```

Antwort:

Response Code 200: [meetingInstructor] (Array - Vollständiges Kombinationsobjekt aus Termin und Dozenten)

OPTIONS	/rest/instructor/{apiKey}/blocked/date/multiple

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige POST-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der POST-Abfrage aufgerufen.

POST	/rest/instructor/{apiKey}/create

Erstellt einen Dozenten für die über den Parameter {apiKey} spezifizierte Bildungseinrichtung.
Bevor diese Methode verwendet werden kann, muss zuerst ein Applikationsschlüssel für die jeweilige Bildungseinrichtung hinterlegt werden. Wichtig beim Anlegen eines Dozenten ist hierbei, dass die Dozentennummer eindeutig sein muss. Das bedeutet, dass keine zwei Dozenten die gleiche Nummer aufweisen dürfen. Weitere Pflichtangaben sind der Nachname und der Arbeitstyp des Dozenten. Letzteres wird für Auswertungszwecke benötigt. Allerdings macht es Sinn neben den genannten Angaben so viele Informationen wie möglich anzugeben. Besonders die E-Mail-Adresse kann für das Anlegen eines zugehörigen Benutzers wichtig sein.

Parameter:

instructor (Vollständiges Dozentenobjekt ohne *uid*)

Beispiel:

```
{
    „instructorNumber“: „40“,
    „gender“: „Herr“,
    „firstName“: „Max“,
    „lastName“: „Mustermann“,
    „email“: „max.mustermann@muster.de“,
    „job“: „Musterjob“,
    „tel“: „012345“,
    „mobil“: „012345“,
    „workType“: „Hauptamtlich“,
    „fax“: „012345“
}
```

Antwort:

Response Code 200: [instructor] (Array - Vollständiges Dozentenobjekt)
Response Code 404: Bereits vorhandene Dozentennummer, fehlende Angaben oder ungültiger Key

OPTIONS	/rest/instructor/{apiKey}/create

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige POST-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der POST-Abfrage aufgerufen.

POST	/rest/instructor/{apiKey}/update/uid

Aktualisiert einen Dozenten für die über den Parameter {apiKey} spezifizierte Bildungseinrichtung. Der Dozent wird dabei über die *uid* identifiziert.
Bevor diese Methode verwendet werden kann, muss zuerst ein Applikationsschlüssel für die jeweilige Bildungseinrichtung hinterlegt werden. Wichtig beim Anlegen eines Dozenten ist hierbei, dass die Dozentennummer eindeutig sein muss. Das bedeutet, dass keine zwei Dozenten die gleiche Nummer aufweisen dürfen. Weitere Pflichtangaben sind der Nachname und der Arbeitstyp des Dozenten. Letzteres wird für Auswertungszwecke benötigt. Allerdings macht es Sinn neben den genannten Angaben so viele Informationen wie möglich anzugeben. Besonders die E-Mail-Adresse kann für das Anlegen eines zugehörigen Benutzers wichtig sein.

Parameter:

instructor (Vollständiges Dozentenobjekt)

Beispiel:

```
{
    „instructorNumber“: „40“,
    „gender“: „Herr“,
    „firstName“: „Max“,
    „lastName“: „Mustermann“,
    „email“: „max.mustermann@muster.de“,
    „job“: „Musterjob“,
    „tel“: „012345“,
    „mobil“: „012345“,
    „fax“: „012345“,
    „workType“: „Hauptamtlich“,
    „uid“: 1
}
```

Antwort:

Response Code 200: [**instructor**] (Array - Vollständiges Dozentenobjekt)
Response Code 404: Bereits vorhandene Dozentennummer, ungültiger Key oder ungültiger Dozent

OPTIONS	/rest/instructor/{apiKey}/update/uid

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige POST-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der POST-Abfrage aufgerufen.

POST	/rest/instructor/{apiKey}/cud/uid

Erstellt, aktualisiert oder löscht einen Dozenten für die über den Parameter {apiKey} spezifizierte Bildungseinrichtung. Die Methode unterscheidet dabei selbst, wie mit einem Datensatz zu verfahren ist. Erstellt wird ein Dozent, wenn er keine *uid* enthält. Ist eine *uid* vorhanden, wird der zugehörige Dozent aktualisiert. Ist zusätzlich zur vorhandenen *uid* noch ein Parameter *deleted* gesetzt, wird der Dozent gelöscht.
Bevor diese Methode verwendet werden kann, muss zuerst ein Applikationsschlüssel für die jeweilige Bildungseinrichtung hinterlegt werden. Wichtig ist hierbei, dass die Dozentennummer eindeutig sein muss. Das bedeutet, dass keine zwei Dozenten die gleiche Nummer aufweisen dürfen. Weitere Pflichtangaben sind der Nachname und der Arbeitstyp des Dozenten.

Parameter:

[**instructor**] (Array - Vollständiges Dozentenobjekt zuzüglich *deleted*)

Beispiel:

```
[{
    „instructorNumber“: „40“,
    „gender“: „Herr“,
    „firstName“: „Max“,
    „lastName“: „Mustermann“,
    „email“: „max.mustermann@muster.de“,
    „job“: „Musterjob“,
    „tel“: „012345“,
    „mobil“: „012345“,
    „fax“: „012345“,
    „workType“: „Hauptamtlich“,
    „uid“: 1,
    „deleted“: false
}]
```

Antwort:

Response Code 200: [instructor] (Array - Vollständiges Dozentenobjekt)
Response Code 404: Bereits vorhandene Dozentennummer, ungültiger Key oder ungültiger Dozent

OPTIONS	/rest/instructor /{apiKey}/cud/uid

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige POST-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der POST-Abfrage aufgerufen.

DELETE	/rest/instructor /{apiKey}/{uid}/uid

Löscht den Dozenten mit der im Parameter {uid} übergebenen ID. Außerdem wird der Dozent aus allen Terminen und Kursen gelöscht, denen er zugeordnet wurde.

Parameter:

„apiKey“ = „string“,	Applikationsschlüssel der Bildungseinrichtung
„uid“ = „integer“	Eindeutige ID des Dozenten

Beispiel:

```
„apiKey“ = „Mustereinreichtung“,
„uid“ = „1“
```

Antwort:

Response Code 200: -
Response Code 404: Es gibt keinen Dozenten zur Nummer, keine Bildungseinrichtung für den ApiKey

OPTIONS	/rest/instructor /{apiKey}/{uid}/uid

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige DELETE-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der DELETE -Abfrage aufgerufen.

Room

GET	/rest/room/educationalInstruction/{uid}

Gibt alle Räume zur Bildungseinrichtung mit der angegebenen ID im Parameter {uid} zurück.

Parameter:

uid: „int“ — Eindeutige ID der Bildungseinrichtung

Beispiel:

uid = 1

Antwort:

Response Code 200: [room] (Vollständiges Raumobjekt)

GET	/rest/room/{apiKey}/addresses

Gibt alle einzigartigen Adressen für die über den Parameter {apiKey} spezifizierte Bildungseinrichtung zurück. Die einzelnen Adressen gehen dabei aus den definierten Räumen hervor und betrachten alle einzigartigen Kombinationen der Adressdaten. Deshalb ist beim Anlegen von Räumen darauf zu achten, keine unterschiedlichen Schreibweisen für dieselbe Adresse zu verwenden.

Parameter:

uid: „int“ — Eindeutige ID der Bildungseinrichtung

Beispiel:

uid = 1

Antwort:

Response Code 200: [address] (Array - Vollständiges Adressobjekt)

POST	/rest/room/{apiKey}/blocked/date

Sucht innerhalb der über den Parameter {apiKey} spezifizierten Bildungseinrichtung nach Räumen, die im übergebenen Zeitraum bereits belegt sind, und gibt diese mit Informationen zu den jeweiligen Terminen zurück. Startet eine Belegung dabei genau zum übergebenen Endzeitpunkt, gilt der Raum nicht als belegt. Selbiges gilt für den Startzeitpunt. Wird über den Parameter *meetingUid* ein Termin spezifiziert, wird in der Ergebnismenge kein Raum dieses Termins geliefert.
Bevor diese Methode verwendet werden kann, muss zuerst ein Applikationsschlüssel für die jeweilige Bildungseinrichtung hinterlegt werden.

Parameter:

[range] (Array - Vollständiges Datumsbereichsobjekt)

Beispiel:

```
{
    „startTime“: „2020-02-28 16:00“,
    „endTime“: „2020-02-28 18:00“,
    „meetingUid“: 1
}
```

Antwort:

Response Code 200: [meetingRoom] (Array - Vollständiges Kombinationsobjekt aus Termin und Raum)

OPTIONS	/rest/room/{apiKey}/blocked/date

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige POST-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der POST-Abfrage aufgerufen.

POST	/rest/room/{apiKey}/blocked/date/multiple

Sucht innerhalb der über den Parameter {apiKey} spezifizierten Bildungseinrichtung nach Räumen, die in einem der übergebenen Zeiträume bereits belegt sind, und gibt diese mit Informationen zu den jeweiligen Terminen zurück. Startet eine Belegung dabei genau zum übergebenen Endzeitpunkt, gilt der Raum nicht als belegt. Selbiges gilt für den Startzeitpunt. Wird über den Parameter *meetingUid* ein Termin spezifiziert, wird in der Ergebnismenge kein Raum dieses Termins geliefert.
Bevor diese Methode verwendet werden kann, muss zuerst ein Applikationsschlüssel für die jeweilige Bildungseinrichtung hinterlegt werden.

Parameter:

[**range**] (Array - Vollständiges Datumsbereichsobjekt)

Beispiel:

```
[{
    „startTime“: „2020-02-28 16:00“,
    „endTime“: „2020-02-28 18:00“,
    „meetingUid“: 1
}]
```

Antwort:

Response Code 200: [**meetingRoom**] (Array - Vollständiges Kombinationsobjekt aus Termin und Raum)

OPTIONS	/rest/room/{apiKey}/blocked/date/multiple

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige POST-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der POST-Abfrage aufgerufen.

POST	/rest/room/{apiKey}/create

Erstellt einen Raum für die über den Parameter {apiKey} spezifizierte Bildungseinrichtung.
Bevor diese Methode verwendet werden kann, muss zuerst ein Applikationsschlüssel für die jeweilige Bildungseinrichtung hinterlegt werden. Wichtig beim Anlegen eines Raumes ist hierbei, dass die Raumnummer für das Gebäude eindeutig sein muss. Es dürfen also keine zwei Räume in einem Gebäude die gleiche Nummer aufweisen. Außerdem werden Adressdaten benötigt und sollten immer auf die gleiche Weise geschrieben werden.

Parameter:

room (Vollständiges Raumobjekt ohne *uid*)

Beispiel:

```
{
    „roomNumber“: „101“,
    „zipCode“: „12345“,
    „city“: „Musterort“,
    „street“: „Musterstraße“,
    „buildingName“: „Mustergebäude“,
    „tel“: „12345“,
    „seats“: 1
}
```

Antwort:

Response Code 200: [**room**] (Array - Vollständiges Raumobjekt)
Response Code 404: Bereits vorhandene Raumnummer im Gebäude, fehlende Angaben oder ungültiger Key

OPTIONS	/rest/room/{apiKey}/create

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige POST-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der POST-Abfrage aufgerufen.

POST	/rest/room/{apiKey}/update/uid

Aktualisiert einen Raum für die über den Parameter {apiKey} spezifizierte Bildungseinrichtung. Der Raum wird dabei über die *uid* identifiziert.
Bevor diese Methode verwendet werden kann, muss zuerst ein Applikationsschlüssel für die jeweilige Bildungseinrichtung hinterlegt werden. Wichtig ist, dass die Raumnummer für das angegebene Gebäude eindeutig sein muss. Außerdem werden Adressdaten benötigt und sollten immer auf die gleiche Weise geschrieben werden.

Parameter:

room (Vollständiges Raumobjekt)

Beispiel:

```
{
    „roomNumber“: „101“,
    „zipCode“: „12345“,
    „city“: „Musterort“,
    „street“: „Musterstraße“,
    „buildingName“: „Mustergebäude“,
    „tel“: „12345“,
    „seats“: 1
    „uid“: 1
}
```

Antwort:

Response Code 200: [**room**] (Array - Vollständiges Raumobjekt)
Response Code 404: Bereits vorhandene Raumnummer im Gebäude, ungültiger Key oder ungültiger Raum

OPTIONS	/rest/room/{apiKey}/update/uid

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige POST-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der POST-Abfrage aufgerufen.

POST	/rest/room/{apiKey}/cud/uid

Erstellt, aktualisiert oder löscht einen Raum für die über den Parameter {apiKey} spezifizierte Bildungseinrichtung. Die Methode unterscheidet dabei selbst, wie mit einem Datensatz zu verfahren ist. Erstellt wird ein Raum, wenn er keine *uid* enthält. Ist eine *uid* vorhanden, wird der zugehörige Raum aktualisiert. Ist zusätzlich zur vorhandenen *uid* noch ein Parameter *deleted* gesetzt, wird der Raum gelöscht.
Bevor diese Methode verwendet werden kann, muss zuerst ein Applikationsschlüssel für die jeweilige Bildungseinrichtung hinterlegt werden. Wichtig ist, dass die Raumnummer für das angegebene Gebäude eindeutig sein muss. Außerdem werden Adressdaten benötigt und sollten immer auf die gleiche Weise geschrieben werden.

Parameter:

[**room**] (Array – Vollständiges Raumobjekt zuzüglich *deleted*)

Beispiel:

```
[{
    „roomNumber“: „101“,
    „zipCode“: „12345“,
    „city“: „Musterort“,
    „street“: „Musterstraße“,
    „buildingName“: „Mustergebäude“,
    „tel“: „12345“,
    „seats“: 1
    „uid“: 1,
    „deleted“: false
}]
```

Antwort:

Response Code 200: [room] (Array - Vollständiges Raumobjekt)
Response Code 404: Bereits vorhandene Raumnummer im Gebäude, ungültiger Key oder ungültiger Raum

OPTIONS	/rest/room/{apiKey}/cud/uid

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige POST-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der POST-Abfrage aufgerufen.

DELETE	/rest/room/{apiKey}/{uid}/uid

Löscht den Raum mit der im Parameter {uid} übergebenen ID. Der Raum wird allerdings nicht gelöscht, falls er bereits in Terminen verwendet wird.

Parameter:

„apiKey“ = „string“,	Applikationsschlüssel der Bildungseinrichtung
„uid“ = „integer“	Eindeutige ID des Raumes

Beispiel:

„apiKey“ = „Mustereinreichtung“,
„uid“ = „1“

Antwort:

Response Code 200: -
Response Code 404: Es gibt keinen Raum zur Nummer, keine Bildungseinrichtung für den ApiKey

OPTIONS	/rest/room/{apiKey}/{uid}/uid

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige DELETE-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der DELETE -Abfrage aufgerufen.

HolidayCalendar

GET	/rest/holidayCalendar/educationalInstruction/{uid}

Gibt alle Kalendereinträge für eine Bildungseinrichtung zurück. Die dabei gelieferten Einträge sind alle Tage, die als blockiert gekennzeichnet wurden, also Feiertage, Ferien oder Brückentage.

Parameter: -

Beispiel: -

Antwort:

Response Code 200: [holidayCalendar] (Array - Vollständiges Ferienkalenderobjekt)

POST	/rest/holidayCalendar/{apiKey}/date

Gibt alle Kalendereinträge für eine Bildungseinrichtung zurück, welche im angegebenen Datumsbereich liegen. Die dabei gelieferten Einträge sind alle Tage, die als blockiert gekennzeichnet wurden, also Feiertage, Ferien oder Brückentage. Es ist auch möglich, nur einen Start- oder Endzeitpunkt anzugeben.

Parameter:

{
„startTime“: „datetime“, Startzeitpunkt
„endTime“: „datetime“, Endzeitpunkt
}

Beispiel:

{
„startTime“: „2020-02-28 16:00“,
„endTime“: „2020-02-28 18:00“,
}

Antwort:

Response Code 200: [holidayCalendar] (Array - Vollständiges Ferienkalenderobjekt)
Response Code 404: Ungültiger Key

OPTIONS	/rest/holidayCalendar/{apiKey}/date

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige DELETE-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der DELETE -Abfrage aufgerufen.

POST	/rest/holidayCalendar/{apiKey}/update

Speichert die übergebenen Datumswerte als neue blockierte Kalendertage. Wichtig ist hierbei, dass immer alle blockierten Tage in der Liste enthalten sein müssen. Nicht enthaltene Tage werden wieder freigegeben.

Parameter:

[„date“] Liste aller blockierten Tage

Beispiel:

[„2020-02-28 16:00“, „2020-02-28 18:00“]

Antwort:

Response Code 200: -
Response Code 404: Fehlerhafte Datumswerte oder ungültiger Key

OPTIONS	/rest/holidayCalendar/{apiKey}/date

Diese Abfrage soll nicht direkt von außen angesprochen werden. Sie ist dafür da, für die zugehörige DELETE-Methode die CORS-Richtlinien zu prüfen und wird von modernen Browsern vor der DELETE -Abfrage aufgerufen.

Category

GET	/rest/category
Gibt alle Kategorien zurück.	
Parameter: -	
Beispiel: -	
Antwort: **Response Code 200**: [category] (Array - Vollständiges Kategorieobjekt)	

EventType

GET	/rest/eventType
Gibt alle im System hinterlegten Veranstaltungsarten zurück.	
Parameter: -	
Beispiel: -	
Antwort: **Response Code 200**: [eventType] (Array - Vollständiges Veranstaltungsartobjekt)	

Cooperation

GET	/rest/cooperation
Gibt alle im System hinterlegten Kooperationen zurück.	
Parameter: -	
Beispiel: -	
Antwort: **Response Code 200**: [cooperation] (Array - Vollständiges Kooperationsobjekt)	

TargetGroup

GET	/rest/targetGroup
Gibt alle im System hinterlegten Zielgruppen zurück	
Parameter: -	
Beispiel: -	
Antwort: **Response Code 200**: [targetGroup] (Array - Vollständiges Zielgruppenobjekt)	

Mail

GET	/rest/mail/key

Holt einen Schlüssel zum Versenden einer E-Mail vom Server. Dieser muss der Methode *send* zum Senden einer E-Mail übergeben werden und ist eine halbe Stunde gültig.

Parameter: -

Beispiel: -

Antwort:

Response Code 200: {
„key“: „string“,
}

Rückgabe bei Erfolg
Schlüssel

POST	/rest/mail/send

Sendet eine E-Mail an info@erwachsenenbildung-lsa.de und übermittelt dabei die im Parameter angegebenen Werte. Hierbei ist ein gültiger Key zu übergeben, der über die Methode *key* geholt werden kann.

Parameter:

{

„key“: „string“,	Schlüssel als Base64-String
„name“: „string“,	Vor- und Nachname
„company“: „string“,	Firma
„companyphone“: „string“,	Firma Telefonnummer
„phone“: „string“,	Telefonnummer
„email“: „string“,	E-Mail
„subject“: „string“,	Thema
„content“: „string“,	Inhalt

}

Beispiel:

```
{
    „key“: „MTU5NjQ1Mzc4Ng==“,
    „name“: „Max Mustermann“,
    „company“: „Musterfirma“,
    „companyphone“: „+49 12345“,
    „phone“: „+49 56789“,
    „email“: „Mustermann@Mustermail.com“,
    „subject“: „Musterthema“,
    „content“: „Musterinhalt“,
}
```

Antwort:

Response Code 200: Die E-Mail wurde erfolgreich versendet
Response Code 404: Fehlende Eingaben oder ungültiger Key