Intranet

Prof. Dr.-Ing. Holger Vogelsang

Stand 09.09.2015

Inhaltsverzeichnis

1

REST-Schnittstelle (Hinweise)

Das Projekt „Intranetaccess“ bietet eine öffentliche Schnittstelle in Form einer REST-API.

1.1 Hinweise

Es gibt drei Gruppen von REST-Aufrufen:

- Im Kapitel 2 wird die Schnittstelle vorgestellt, die von allen Fakultäten der Hochschule bzw. die ohne Authentifizierung nutzbar ist. Hier werden recht allgemeine Infirmationen bereitgestellt.
- Das Kapitel 3 beschreibt die Aufrufe, die speziell auf die Informatik-Studiengänge der Fakultät IWI zugeschnitten ist und die teilweise ohne gültige IZ-Accounts der Informatik nicht nutzbar ist.
- Kapitel 4 stellt eine experimentelle Schnittstelle vor, dessen API sich jederzeit ändern kann. Die Schnittstelle kann nur mit gültigem Informatik-Account genutzt werden.

1.2 Aufrufe

Der Zugriff darf sowohl über HTTP als auch HTTPS erfolgen. Alle URLs in den folgenden Unterkapiteln sind Pfade unterhalb von:
`http://www.iwi.hs-karlsruhe.de/Intranetaccess/REST`
bzw.
`https://www.iwi.hs-karlsruhe.de/Intranetaccess/REST`
Beim Zugriff über `HTTP` sendet der Server ein „Redirect“ auf dieselbe URL, allerdings mit HTTPS-Protokoll. Existiert eine URL unterhalb von `/Intranetaccess/REST` nicht, dann wird der HTTP-Status-Code `NOT FOUND` (404) zurückgegeben.

1.3 Passwörter

Einige Aufrufe erfordern die Anmeldedaten des Aufrufers. Diese können als Kombination aus Namen und Passwort per HTTP-Basic-Authentifizierung übergeben werden. Häufig ist es aber sinnvoll, auf dem Client das Passwort nicht direkt speichern zu müssen.

Deswegen bietet der Aufruf 2.1.2 die Möglichkeit, das Passwort auf dem Server verschlüsseln und das verschlüsselte Passwort zurückgeben zu lassen. Dieses verschlüsselte Passwort kann genauso wie das unverschlüsselte in allen Aufrufen verwendet werden, die die HTTP-Basic-Authentifizierung erfordern. Das Passwort wird weder unverschlüsselt noch verschlüsselt auf dem Server gespeichert. Lediglich der Schlüssel ist dort vorhanden. Nach einer Änderung des Schlüssels auf dem Server wird es erforderlich, erneut ein verschlüsseltes Passwort zu erzeugen. Das ist daran zu erkennen, dass trotz eines vormals korrekten verschlüsselten Passwortes ein Authentifizierungsfehler `UNAUTHORIZED` (401) bei Zugriffen auf gesicherte Daten gemeldet wird.

2

REST-Schnittstelle (allgemein)

Dieses Kapitel beschreibt die Aurfufe, die für alle Fakultäten unabhängig vom Informatik-Intranet der Fakultät IWII einsetzbar ist. Möglicherweise sind aber noch nicht alle Funktionen für Anwender anderer Fakultäten freigeschaltet, so dass lediglich Aufrufe ohne Authentifizierungsinformationen nutzbar sein können.

2.1 Anwenderinformationen

Diese Schnittstelle dient dazu zu prüfen, ob die Anmeldeinformationen korrekt sind und welche Rolle der Aufrufer besitzt.

2.1.1 Anmeldeinformation

Der Aufruf liefert als Information, ob die Anmeldeinformation korrekt ist. Im Gegensatz zu allen anderen Aufrufen, die Anmeldeinformationen verlangen, werden diese direkt in der URL als Teile des Pfades angegeben.

Aufruf (GET)	
`/credential/check/<name>/<passwort>`	
Aufruf-Parameter in der URL	
`<name>`	Name des Anwenders.
`<passwort>`	Passwort des Anwenders.
Authentifizierung	
Der Aufruf verlangt keine Authentifizierung mittels „HTTP Basic Authentication“. Diese würde im Fehlerfall einen Statuscode ungleich 200 bewirken.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
Ergebnis (Body als Zeichenkette)	
Das Ergebnis ist `true`, wenn die Anmeldeinformation korrekt ist, ansonsten `false`.	

2.1.2 Verschlüsseltes Passwort

Der Aufruf liefert als Ergebnis das verschlüsselte Passwort des Anwenders zurück.

Aufruf (GET)	
`/credential/encryptedpassword`	
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
Ergebnis (Body als Zeichenkette)	
Das Ergebnis eine Zeichenkette mit den verschlüselten Passwort oder ein Leerstring, falls der Aufrufer weder Student noch Mitarbeiter war.	

2.1.3 Informationen über Anwender

Der Aufruf liefert als Ergebnis Informationen über den Anwender.

Aufruf (GET)	
`/credential/info`	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
Das Ergebnis ist ein JSON-Objekt mit folgendem Aufbau für Studierende.	
`adsName`	Login-Name des Studierenden im ADS.
`mailAddress`	E-Mail-Adresse des Studierenden an der Hochschule.
`firstName`	Vorname des Studierenden.
`lastName`	Nachname des Studierenden.
`faculty`	Kürzel der Fakultät, zu der der Studierende gehört (z.B. „IWI“ für „Informatik und Wirtschaftsinformatik“).
`department`	Fachbereich innerhalb der Fakultät, zu der der Studierende gehört (z.B. „I“ für „Informatik“ oder „WI“ für „Wirtschaftsinformatik“.)
`courseOfStudies`	Kürzel des Studiengang, in dem der Studierende eingeschrieben ist (z.B. „INFB“, „INMF“, „MKIB“). Diese Kürzel werden konsistent innerhalb der REST-Schnittstelle zur Identifizierung der Studiengänge verwendet, siehe auch Abschnitt 3.1.
`courseOfStudiesName`	Name des Studiengangs.
`matrNumber`	Matrikelnummer des Studierenden.
`telephoneNumber`	Telefonnummer des Studierenden, kann `null` sein.
`examination RegulationsNumber`	Nummer der Prüfungsordnung, innerhalb derer der Studierende eingeschrieben ist.

`roleName`	Rollenname des Anwender, ist immer `student`.
`id`	Eindeutige ID des Studierenden, ändert sich innerhalb eines Studiengangs nicht. Erst beim Wechsel vom Bachelor in den Master erhält der Studierende zusammen mit einem neuen ADS-Account auch eine neue ID.
Das Ergebnis ist ein JSON-Objekt mit folgendem Aufbau für Dozenten oder Sekretariatsmitglieder.	
`adsName`	Login-Name des Mitarbeiters im ADS.
`mailAddress`	E-Mail-Adresse des Mitarbeiters an der Hochschule.
`firstName`	Vorname des Mitarbeiters.
`lastName`	Nachname des Mitarbeiters.
`roleName`	Rollenname des Anwender, ist `lecturer` für Dozenten und `office` für Mitglieder des Sekretariats.
`title`	Akademischer Grad des Mitarbeiters.
`faculty`	Kürzel der Fakultät, zu der der Mitarbeiters gehört (z.B. „IWI“ für „Informatik und Wirtschaftsinformatik“).
`department`	Fachbereich innerhalb der Fakultät, zu der der Mitarbeiters gehört (z.B. „I“ für „Informatik“ oder „WI“ für „Wirtschaftsinformatik“.)
`mastered CourseOfStudies`	Array mit Kürzeln der Studiengänge, die der Dozent leitet (in denen er Studiendekan ist)
`id`	Eindeutige ID des Mitarbeiters, ändert sich nicht.

2.1.4 Rolle des Anwenders

Der Aufruf liefert als Ergebnis den Namen der Rolle, die der Anwender besitzt. Das Ergebnis ist der Wert, der in der JSON-Antwort im Aufruf 2.1.3 im Attribut `roleName` aufgeführt ist.

Aufruf (GET)	
`/credential/role`	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
Ergebnis (Body als Zeichenkette)	
Das Ergebnis ist der Name der Rolle, die der Anwender im Intranet besitzt. Zur Zeit werden vergeben: ▪ `student`: Der Aufrufer ist Student. ▪ `lecturer`: Der Aufrufer ist Dozent. ▪ `office`: Der Aufrufer ist Sekretariatsmitglied. Weitere Rollennamen können in Zukunft hinzu kommen.	

2.2 Version

Der Aufruf liefert die Versionsnummer der REST-Schnittstelle.

Aufruf (GET)	
`/version`	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	In jedem Fall.
Ergebnis (Body als String)	
Das Ergebnis ist ein String im Format "*`Major.Minor.Micro`*" (z.B. "`1.2.1`"). Die Schnittstelle ist zur vorherigen Version kompatibel, solange sich die Major-Nummer nicht ändert. Die Micro-Nummer ändert sich bei kleineren Fehlerkorrekturen.	

2.3 Mensapläne

Diese Schnittstelle liefert die Mensapläne vom Server des Studentenwerks. Die Pläne werden für bessere Antwortzeiten intern in einem Cache gehalten und geben deshalb Änderungen auf den Seiten des Studentenwerks um höchstens eine Minute verzögert wieder.

Aufruf (GET)	
`/canteen/<id>/<date>`	
Aufruf-Parameter in der URL	
`<id>`	Nummer der Mensa, erlaubt sind die IDs 1 bis 6: `1` Mensa Am Adenauerring `2` Mensa Moltkestraße `3` Mensa Erzbergerstraße `4` Mensa Schloss Gottesaue `5` Mensa Tiefenbronner Straße `6` Mensa Holzgartenstraße
`<date>`	Datum, für das die Essen ausgelesen werden sollen. Das Datum muss im Format `yyyy-MM-dd` (also z.B. 2013-12-17) angegeben sein.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`BAD REQUEST` (400)	Wenn das Datum ungültig ist.
`NOT FOUND` (404)	Wenn in der URL eine ungültige Mensanummer aufgeführt ist.
Ergebnis (Body als JSON-Objekt)	
Das Ergebnis ist ein JSON-Objekt mit folgendem Aufbau.	
`name`	Lokalisierter Name der Mensa.
`mealGroups`	JSON-Array aller Essen in dieser Mensa, Erklärung folgt weiter unten.

`status`	Der Status dieses Abrufs beschreibt einen möglichen Fehler genauer. Er wird als Zeichenkette abgelegt: ■ `ok`: Der Aufruf erfolgte ohne Fehler. Dieser Status wird beim HTTP-Status-Code `OK` zurückgegeben. ■ `invalid canteen id`: Die angegebene Mensa-Nummer ist ungültig. ■ `invalid request date`: Für das angegebene Datum existiert kein Mensa-Plan. ■ `malformed request date`. Das Datum liegt nicht im geforderten Format vor (`yyyy-MM-dd`). ■ `no mealplan available`: Die Mensa bietet an dem Tag kein Essen an. Dieser Status wird beim HTTP-Status-Code `OK` zurückgegeben.
`date`	Datum, für das die Angaben oben gelten (entspricht dem Datum aus dem Aufruf).

Aufbau der Essensgruppen (z.B. eine Linie in der Mensa am Adenauerring oder ein Wahlessen in der Mensa Moltke), Objekt im Array des Attributs `mealGroups` der Mensa	
`title`	Lokalisierter Name der Essensgruppe wie z.B. „Linie 1“ oder „Wahlessen 1“.
`meals`	Array aller Essen der Gruppe, Erklärung folgt weiter unten.
`message`	Ist eine Essensausgabe geschlossene und damit eine der Gruppen nicht verfügbar, dann steht hier eine Meldung des Studentenwerks. Ansonsten ist der Eintrag `null`.

Aufbau eines Essens einer Gruppe, Objekt im Array des Attributs `meals` der Essensgruppe	
`name`	Textuelle Bezeichnung des Essens ohne Formatierungen.
`identifier`	Eindeutige Identifikation der Mahlzeit, wird für die Mensabewertungen in Abschnitt 2.4 benötigt.
`priceStudent`	Essenspreis für Studierende, Dezimaltrennzeichen ist ein Punkt. Ein Wert kleiner oder gleich 0 kennzeichnet, dass kein Preis angegeben ist.
`priceGuest`	Essenspreis für Gäste, Dezimaltrennzeichen ist ein Punkt. Ein Wert kleiner oder gleich 0 kennzeichnet, dass kein Preis angegeben ist.
`priceEmployee`	Essenspreis für Mitarbeiter, Dezimaltrennzeichen ist ein Punkt. Ein Wert kleiner oder gleich 0 kennzeichnet, dass kein Preis angegeben ist.
`pricePupil`	Essenspreis für Schüler, Dezimaltrennzeichen ist ein Punkt. Ein Wert kleiner oder gleich 0 kennzeichnet, dass kein Preis angegeben ist.
`priceAdditive`	Zusatzhinweis zum Preis (z.B. „pro 100g“, „ab“, . . .). Der Eintrag ist `null`, wenn es sich um Festpreise handelt.

`foodAdditiveNumbers`	Array mit Hinweisnummern zu besonderen Bestandteilen des Essens oder Lebensmittelzusätzen. Laut Legende werden zur Zeit die folgenden Nummern verwendet. In dieser Dokumentation sowie in der REST-Schnittstelle wird weiterhin der alte Begriff „Nummer“ verwendet, weil es sich früher ausschließlich um Zahlen handelte. Inzwischen werden hier Zeichenketten zurückgegeben. `1` mit Farbstoff `2` mit Konservierungsstoff `3` mit Antioxidationsmittel `4` mit Geschmacksverstärker `5` mit Phosphat `6` Oberfläche gewachst `7` geschwefelt `8` Oliven geschwärzt `9` mit Süßungsmittel `10` kann bei übermäßigem Verzehr abführend wirken `11` enthält eine Phenylalaninquelle `12` kann Restalkohol enthalten `14` Kennzeichnung siehe Tageskarte `15` mit kakaohaltiger Fettglasur `27` aus Fischstücken zusammengefügt `93` enthält Rindfleisch `94` enthält Rindfleisch aus artgerechter Tierhaltung `95` enthält Schweinefleisch `96` vegetarisches Gericht (ohne Fleischzusatz) `97` veganes Gericht `98` MSC-zertifizierter Fisch `99` kontrolliert biologischer Anbau mit EU Bio-Siegel / DE-Öko-007 `Gl` Glutenhaltiges Getreide `Nr` Schalenfrüchte / Nüsse `Ei` Eier `Er` Erdnüsse `So` Soja `Sn` Senf `Kr` Krebstiere `Fi` Fisch `ML` Milch / Laktose `Se` Sellerie `Sf` Schwefeldioxid / Sulfit `Sa` Sesam `Lu` Lupine `We` Weichtiere Allerdings verwendet das Studentenwerk als Hinweise zu den einzelnen Essen auch weitere Nummern ohne Erklärung.

Gelegentlich enthalten die Preisangaben `priceGuest`, `priceEmployee` und `pricePupil` den Wert `-1`. Dann kostet das Essen für alle Personengruppen dasselbe, und es gilt der Preis von `priceStudent`. Enthalten alle Preisangaben den Wert `-1`, dann steht der Preis noch nicht fest oder er wird nur durch Aushang in der Mensa bekannt gegeben.

2.4 Mensabeurteilungen

Diese Schnittstelle erlaubt Zugriffe auf die Bewertungen einzelner Mahlzeiten der Mensen. Jede Mahlzeit wird durch eine Identifikation beschrieben. Die Identifikation wird bei jedem Essen als Attribut `identification` zurückgegeben (siehe Abschnitt 2.3).

2.4.1 Eigene Beurteilungen

Über diese Aufrufen können Beurteilungen abgegeben, ausgelesen und gelöscht werden. Für alle Aufrufe in diesem Abschnitt sind Benutzerinformationen erforderlich.

2.4.1.1 Schreiben einer Beurteilung

Der Aufruf schreibt die eigene Beurteilung einer einzelnen Mahlzeit in die Datenbank. Existiert bereits eine Beurteilung für diese Mahlzeit, dann wird sie überschrieben.

Aufruf (POST)	
`/canteenfeedback/rating/private/<id>/<mealId>`	
Aufruf-Parameter in der URL	
`<id>`	Nummer der Mensa, erlaubt sind die IDs 1 bis 6: `1` Mensa Am Adenauerring `2` Mensa Moltkestraße `3` Mensa Erzbergerstraße `4` Mensa Schloss Gottesaue `5` Mensa Tiefenbronner Straße `6` Mensa Holzgartenstraße
`<mealId>`	Eindeutige Identifikation der Mahlzeit, deren Bewertung ausgelesen werden soll (siehe Abschnitt 2.3). Achtung: Diese Identifikation kann Zeichen enthalten, die in einer URL nicht erlaubt sind. Daher müssen diese codiert werden (z.B. „`:`“ → „`%3A`“).
Aufruf-Parameter als JSON-Objekt (Body) Die Informationen über die Beurteilung werden als JSON-Objekt im Request-Body übergeben. Das Objekt hat den folgenden Aufbau:	
`favourite`	`true`, wenn das Essen auf die eigene Favoritenliste gesetzt werden soll, ansonsten `false`.
`ranking`	Eigene Beurteilung des Essens von 1 (ganz mies) bis zu 5 (perfekt).
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`CREATED` (201)	Im Erfolgsfall.
`BAD REQUEST` (400)	Das JSON-Objekt ist nicht vollständig oder korrekt ausgefüllt. Der Fehler tritt ebenfalls bei einem falschen Datumsformat auf.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
`NOT FOUND` (404)	Wenn in der URL eine ungültige Mensanummer aufgeführt ist oder es kein Essen mit der angegebenen Identifikationsnummer gibt.
Ergebnis (Body ist leer)	

2.4.1.2 Lesen einer Beurteilung

Der Aufruf liest die eigene Beurteilung einer einzelnen Mahlzeit aus.

Aufruf (GET)
`/canteenfeedback/rating/private/<id>/<mealId>`
Aufruf-Parameter in der URL

`<id>`	Nummer der Mensa, erlaubt sind die IDs 1 bis 6: 1 Mensa Am Adenauerring 2 Mensa Moltkestraße 3 Mensa Erzbergerstraße 4 Mensa Schloss Gottesaue 5 Mensa Tiefenbronner Straße 6 Mensa Holzgartenstraße
`<mealId>`	Eindeutige Identifikation der Mahlzeit, deren Bewertung ausgelesen werden soll (siehe Abschnitt 2.3). Achtung: Diese Identifikation kann Zeichen enthalten, die in einer URL nicht erlaubt sind. Daher müssen diese codiert werden (z.B. „`:`“ → „`%3A`“).
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`BAD REQUEST` (400)	Wenn in der URL eine ungültige Mensanummer aufgeführt ist.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
`NOT FOUND` (404)	Wenn es kein Essen mit der angegebenen Identifikationsnummer gibt.
Ergebnis (Body als JSON-Objekt) Im Fehlerfall oder wenn keine Bewertungen vorliegen, ist der Rumpf der Antwort leer. Im Erfolgsfall ist das Ergebnis ein JSON-Objekt mit folgendem Aufbau.	
`created`	Datum im Format „`yyyy-MM-dd`“, an die Bewertung geschrieben wurde.
`rankedMeal`	Beschreibung der beurteilten Mahlzeit. Der Aufbau dieses JSON-Objektes wird in der folgenden Tabelle beschrieben.
`ranking`	Eigene Beurteilung des Essens von 1 (ganz mies) bis zu 5 (perfekt).
`favourite`	`true`, wenn es sich um ein Essen auf der eigenen Favoritenliste handelt, ansonsten `false`.

Beschreibung einer Mahlzeit, Attribut `rankedMeal` der Bewertung	
`name`	Vollständiger Name des Essens.
`identifier`	Identifikation der Mahlzeit, entspricht dem Parameter `mealId` in der Aufruf-URL.
`foodAdditiveNumbers`	Array mit den Nummern der Lebensmittelzusätze.
`canteenId`	Nummer der Mensa, entspricht dem Parameter `id` in der Aufruf-URL.

2.4.1.3 Löschen einer Beurteilung

Der Aufruf löscht die eigene Beurteilung einer einzelnen Mahlzeit aus der Datenbank. Existiert die zu löschende Beurteilung nicht, wird kein Fehler gemeldet.

Aufruf (DELETE)
`/canteenfeedback/rating/private/<id>/<mealId>`
Aufruf-Parameter in der URL

`<id>`	Nummer der Mensa, erlaubt sind die IDs 1 bis 6: 1 Mensa Am Adenauerring 2 Mensa Moltkestraße 3 Mensa Erzbergerstraße 4 Mensa Schloss Gottesaue 5 Mensa Tiefenbronner Straße 6 Mensa Holzgartenstraße
`<mealId>`	Eindeutige Identifikation der Mahlzeit, deren Bewertung ausgelesen werden soll (siehe Abschnitt 2.3). Achtung: Diese Identifikation kann Zeichen enthalten, die in einer URL nicht erlaubt sind. Daher müssen diese codiert werden (z.B. „`:`“ → „`%3A`“).
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`BAD REQUEST` (400)	Wenn in der URL eine ungültige Mensanummer aufgeführt ist.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
Ergebnis (Body ist leer)	

2.4.1.4 Lesen der Favoriten

Der Aufruf liest alle eigenen Favoriten aus.

Aufruf (GET) `/canteenfeedback/favourites/private/<id>`	
Aufruf-Parameter in der URL	
`<id>`	Nummer der Mensa, erlaubt sind die IDs 1 bis 6: 1 Mensa Am Adenauerring 2 Mensa Moltkestraße 3 Mensa Erzbergerstraße 4 Mensa Schloss Gottesaue 5 Mensa Tiefenbronner Straße 6 Mensa Holzgartenstraße
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`BAD REQUEST` (400)	Wenn in der URL eine ungültige Mensanummer aufgeführt ist.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
Ergebnis (Body als Array mit JSON-Objekten) Das Ergebnis ist eine Array mit JSON-Objekt des folgenden Aufbaus.	
`created`	Datum im Format „`yyyy-MM-dd`“, an die Bewertung geschrieben wurde.

`rankedMeal`	Beschreibung der beurteilten Mahlzeit. Der Aufbau dieses JSON-Objektes wird in der folgenden Tabelle beschrieben.
`ranking`	Eigene Beurteilung des Essens von 1 (ganz mies) bis zu 5 (perfekt).
`favourite`	Immer `true`, weil hier nur die Favoriten gelesen werden.

Beschreibung einer Mahlzeit, Attribut `rankedMeal` der Bewertung	
`name`	Vollständiger Name des Essens.
`identifier`	Identifikation der Mahlzeit, entspricht dem Parameter `mealId` in der Aufruf-URL.
`foodAdditiveNumbers`	Array mit den Nummern der Lebensmittelzusätze.
`canteenId`	Nummer der Mensa, entspricht dem Parameter `id` in der Aufruf-URL.

2.4.2 Alle Beurteilungen

Mit diesen Aufrufen kann die Meinung anderer Anwender eingeholt werden.

2.4.2.1 Lesen einer Beurteilung

Über diesen Aufruf können die Durchschnittswerte aller Beurteilungen einer Mahlzeit ausgelesen werden. Der Aufruf darf anonym erfolgen.

Aufruf (GET)	
`/canteenfeedback/rating/public/<id>/<mealId>`	
Aufruf-Parameter in der URL	
`<id>`	Nummer der Mensa, erlaubt sind die IDs 1 bis 6: `1` Mensa Am Adenauerring `2` Mensa Moltkestraße `3` Mensa Erzbergerstraße `4` Mensa Schloss Gottesaue `5` Mensa Tiefenbronner Straße `6` Mensa Holzgartenstraße
`<mealId>`	Eindeutige Identifikation der Mahlzeit, deren Bewertung ausgelesen werden soll (siehe Abschnitt 2.3). Achtung: Diese Identifikation kann Zeichen enthalten, die in einer URL nicht erlaubt sind. Daher müssen diese codiert werden (z.B. „:“ → „`%3A`“).
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`BAD REQUEST` (400)	Wenn in der URL eine ungültige Mensanummer aufgeführt ist.
`NOT FOUND` (404)	Wenn es kein Essen mit der angegebenen Identifikationsnummer gibt.
Ergebnis (Body als JSON-Objekt) Das Ergebnis ist ein JSON-Objekt mit folgendem Aufbau.	
`favourites`	Prozentsatz als Double-Zahl der Anwender, die diese Mahlzeit beurteilt und dabei die Mahlzeit zu ihren eigenen Favoriten hinzugefügt haben.
`ranking`	Double-Wert der durchschnittlichen Beurteilung der Mahlzeit von 1 (ganz mies) bis zu 5 (perfekt).
`rankedMeal`	Beschreibung der beurteilten Mahlzeit. Der Aufbau dieses JSON-Objektes wird in der folgenden Tabelle beschrieben.

votesCount	Anzahl an Beurteilungen, die für dieses Essen abgegeben und die somit zur Berechnung der Werte ranking und favourites berücksichtigt wurden.

Beschreibung einer Mahlzeit, Attribut rankedMeal der Bewertung	
name	Vollständiger Name des Essens.
identifier	Identifikation der Mahlzeit, entspricht dem Parameter mealId in der Aufruf-URL.
foodAdditiveNumbers	Array mit den Nummern der Lebensmittelzusätze.
canteenId	Nummer der Mensa, entspricht dem Parameter id in der Aufruf-URL.

2.4.2.2 Lesen der Favoriten

Über diesen Aufruf können die Favoriten aller Anwender, sortiert anhand des Rankings, ausgelesen werden. Es werden dabei nur solche Mahlzeiten berücksichtigt, die mindestens ein Anwender als Favorit markiert hat. Die Treffer werden sortiert zurückgegeben:

1. Je mehr Anwender eine Mahlzeit als Favoriten markiert haben, desto weiter vorne in der Trefferliste steht sie.
2. Bei gleicher Anzahl an Favoritenmarkierungen entscheidet das höhere Ranking.

Der Aufruf darf anonym erfolgen.

Aufruf (GET) /canteenfeedback/favourites/public/<id>/<limit>	
Aufruf-Parameter in der URL	
<id>	Nummer der Mensa, erlaubt sind die IDs 1 bis 6: 1 Mensa Am Adenauerring 2 Mensa Moltkestraße 3 Mensa Erzbergerstraße 4 Mensa Schloss Gottesaue 5 Mensa Tiefenbronner Straße 6 Mensa Holzgartenstraße
<limit>	Maximale Anzahl an Treffern, die ausgelesen werden soll. Fehlt dieser Wert oder ist er negativ, dann werden alle Treffer ausgelesen.
Ergebnis (HTTP-Status-Code)	
OK (200)	Im Erfolgsfall.
BAD REQUEST (400)	Wenn in der URL eine ungültige Mensanummer aufgeführt ist.
Ergebnis (Body als JSON-Objekt) Das Ergebnis ist ein JSON-Objekt mit folgendem Aufbau.	
favourites	Prozentsatz als Double-Zahl der Anwender, die diese Mahlzeit beurteilt und dabei die Mahlzeit zu ihren eigenen Favoriten hinzugefügt haben. Der Wert ist immer 100, weil hier nur die Favoriten gelesen werden.
rankings	Double-Wert der durchschnittlichen Beurteilung der Mahlzeit von 1 (ganz mies) bis zu 5 (perfekt).
rankedMeal	Beschreibung der beurteilten Mahlzeit. Der Aufbau dieses JSON-Objektes wird in der folgenden Tabelle beschrieben.
votesCount	Anzahl an Beurteilungen, die für dieses Essen abgegeben und die somit zur Berechnung der Werte

Beschreibung einer Mahlzeit, Attribut `rankedMeal` der Bewertung	
`name`	Vollständiger Name des Essens.
`identifier`	Identifikation der Mahlzeit, entspricht dem Parameter `mealId` in der Aufruf-URL.
`foodAdditiveNumbers`	Array mit den Nummern der Lebensmittelzusätze.
`canteenId`	Nummer der Mensa, entspricht dem Parameter `id` in der Aufruf-URL.

2.5 Schlüssel-/Werte-Paare

Diese Schnittstelle dient dazu, allgemeine Schlüssel- und Werte-Paare auf dem Server zu speichern und auszulesen. Dabei wird zwischen öffentlichen und privaten Daten unterschieden.

2.5.1 Öffentliche Daten

Um diese Daten zu ändern, werden Schreibrechte benötigt. Lesend darf dagegen jeder Aufrufer zugreifen. Daten werden einzelnen Domänen zugeordnet. Innerhalb einer Domäne sind aus Gründen der der Übersichtlichkeit Sub-Domänen vorgesehen. Die eigentlichen Schlüssel-/Werte-Paare werden in Sub-Domänen abgelegt. Die Rechte werden Domänen-spezifisch vergeben. Das heißt, dass jemand mit Schreibrecht in der Domäne `xyz` alle Sub-Domänen von `xyz` beschreiben darf. Beispiel für den Einsatz: Die Fachschaft erhält die Domäne `studentcouncil`. Fachschaftsmitglieder erhalten das Schreibrecht auf die Domäne. Innerhalb der Domäne wird die Sub-Domäne `store` dazu verwendet, um die Artikel, die die Fachschaft auf Lager hat und verkaufen kann, zu verwalten. Die Längen der Domänen- und Sub-Domänen-Namen sind auf jeweils 20 Zeichen beschränkt. Je Domäne sind maximal 100 Sub-Domänen und je Sub-Domäne maximal 100 Schlüssel möglich. Die Grenze dient dazu zu verhindern, dass in der Datenbank eine unkontrollierbare Datenmenge abgelegt wird.

2.5.1.1 Rechteverwaltung

Durch die Rechteverwaltung können Schreibrechte ermittelt, zugewiesen und entzogen werden.

Ermittlung der eigenen Rechte

Der folgende Aufruf liefert die Namen aller Domänen, für die der Aufrufer Schreibrechte besitzt.

Aufruf (GET)	
`/keyvaluestore/public/writeaccess`	
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.

`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
Ergebnis (Body als JSON-Array)	
Das Ergebnis ist ein JSON-Array mit Strings, wobei jeder String einem Domänen-Namen entspricht. Ist das Array leer, dann besitzt der Aufrufer in keiner Domäne Schreibrechte.	

Ermittlung der Schreibrechte für eine Domäne

Der folgende GET-Aufruf liefert die Namen aller Anwender, die in der übergebenen Domäne Schreibrechte besitzen. Es werden aus Gründen des Datenschutzes nur solche Domänen berücksichtigt, in denen der Aufrufer selbst auch Schreibrecht hat.

Aufruf (GET)	
`/keyvaluestore/public/writeaccess/<domain>`	
Aufruf-Parameter in der URL	
`<domain>`	Name der Domäne, für die die Anwender mit Schreibrecht ermittelt werden sollen.
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
`FORBIDDEN` (403)	Der Aufrufer versucht auf eine Domäne zuzugreifen, innerhalb derer er selbst kein Schreibrecht besitzt.
Ergebnis (Body als JSON-Array)	
Das Ergebnis ist ein JSON-Array mit Strings, wobei jeder String dem ADS-Login-Namen eines Anwenders entspricht, der Schreibrecht in der Domäne besitzt. Ist das Array leer, dann besitzt die Domäne keinen Anwender mit Schreibrechten.	

Gewährung von Schreibrechten

Der folgende POST-Aufruf fügt einer Domäne einen Anwender hinzu, so dass dieser ebenfalls Schreibrecht besitzt. Der Aufrufer kann nur für solche Domänen Schreibrechte vergeben, für die er selbst als Administrator eingetragen ist.

Aufruf (POST)	
`/keyvaluestore/public/writeaccess`	
Aufruf-Parameter als JSON-Objekt (Body)	
`adsAccount`	ADS-Login-Name des Anwenders, der das Schreibrecht erhalten soll.
`domain`	Name der Domäne, für die das Recht gelten soll.
`admin`	Dieser Boole'sche Wert ist `true`, wenn der neue Anwender Administrator-Rechte erhalten soll. Ansonsten ist der Wert `false`. Administrator-Rechte sind erforderlich, um Benutzerrechte vergeben zu können.
Authentifizierung	

Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`BAD REQUEST` (400)	Das JSON-Objekt ist nicht vollständig oder korrekt ausgefüllt.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
`FORBIDDEN` (403)	Der Aufrufer selbst besitzt in der Domäne kein Administrator-Recht.
Ergebnis (Body)	
Der Rumpf der Antwort ist immer leer.	

Entzug von Schreibrechten

Der folgende DELETE-Aufruf entfernt die Schreibrechte eines Anwenders aus einer Domäne. Der Aufrufer darf nur für solche Domänen Schreibrechte entziehen, für die er als Administrator eingetragen ist. Er kann sich auch nicht selbst das Recht entziehen.

Aufruf (DELETE)	
`/keyvaluestore/public/writeaccess`	
Aufruf-Parameter als JSON-Objekt (Body)	
`adsAccount`	ADS-Login-Name des Anwenders, dem das Schreibrecht entzogen werden soll.
`domain`	Name der Domäne, für die das Recht entzogen werden soll.
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`BAD REQUEST` (400)	Das JSON-Objekt ist nicht vollständig oder korrekt ausgefüllt.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
`FORBIDDEN` (403)	Der Aufrufer selbst besitzt in der Domäne kein Administrator-Recht.
Ergebnis (Body)	
Der Rumpf der Antwort ist immer leer.	

2.5.1.2 Lesezugriffe

Alle Lesezugriffe sind öffentlich.

Auslesen aller Schlüssel

Der folgende Aufruf liest alle Schlüssel einer Sub-Domäne aus.

Aufruf (GET)
`/keyvaluestore/public/values/<domain>/<subdomain>`

Aufruf-Parameter in der URL	
`<domain>`	Name der Domäne.
`<subdomain>`	Name der Sub-Domäne.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
Ergebnis (Body als JSON-Array) Das Ergebnis ist ein JSON-Array mit Objekten, die jeweils ein Schlüssel-/Werte-Paar beschreiben. Ist das Array leer, dann sind in der Sub-Domäne keine Paare enthalten, oder aber die Domäne existiert nicht.	
`domain`	Domäne (Gebiet), aus dem der Schlüssel kommt (z.B. `studentcouncil` für die Fachschaft).
`subdomain`	Name der Sub-Domäne (Untergebiet) innerhalb einer Domäne, aus der der Schlüssel stammt (z.B. `stock` für den Warenbestand).
`key`	Nicht lokalisierter Name des Schlüssels (z.B. `mate` für den Matebestand).
`value`	Wert, der dem Schlüssel zugeordnet ist (z.B. `100` für 100 Mate-Flaschen).
`type`	Typ des Werts (z.B. `int`).
`unit`	Einheit des Werts (z.B. `Euro`).
`localizedUnit`	Lokalisierte Einheit des Wertes (z.B. `Flaschen`).
`localizedKey`	Lokalisierter Name des Schlüssels (z.B. `Mate`).
`localized-Description`	Lokalisierte textuelle Beschreibung des Eintrags (beispielsweise `Bestand an Mate-Flaschen in der Fachschaft` oder auch `Preis 1 Euro je Flasche`).

Auslesen eines Schlüssels

Der folgende Aufruf liest einen Schlüssel einer Sub-Domäne aus.

Aufruf (GET) `/keyvaluestore/public/value/<domain>/<subdomain>/<key>`	
Aufruf-Parameter in der URL	
`<domain>`	Name der Domäne.
`<subdomain>`	Name der Sub-Domäne.
`<key>`	Name des Schlüssels.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`NOT FOUND` (404)	Der gesuchte Schlüssel existiert nicht in der Sub-Domäne.
Ergebnis (Body als JSON-Objekt) Das Ergebnis ist ein JSON-Objekt, das ein Schlüssel-/Werte-Paar beschreiben. Dessen Aufbau stimmt mit dem beim Auslesen aller Schlüssel einer Sub-Domäne beschriebenen überein.	
`domain`	Domäne (Gebiet), aus dem der Schlüssel kommt (z.B. `studentcouncil` für die Fachschaft).
`subdomain`	Name der Sub-Domäne (Untergebiet) innerhalb einer Domäne, aus der der Schlüssel stammt (z.B. `stock` für den Warenbestand).
`key`	Nicht lokalisierter Name des Schlüssels (z.B. `mate` für den Matebestand).

`value`	Wert, der dem Schlüssel zugeordnet ist (z.B. `100` für 100 Mate-Flaschen).
`type`	Typ des Werts (z.B. `int`).
`unit`	Einheit des Werts (z.B. `Euro`).
`localizedUnit`	Lokalisierte Einheit des Wertes (z.B. `Flaschen`).
`localizedKey`	Lokalisierter Name des Schlüssels (z.B. `Mate`).
`localized-Description`	Lokalisierte textuelle Beschreibung des Eintrags (beispielsweise `Bestand an Mate-Flaschen in der Fachschaft` oder auch `Preis 1 Euro je Flasche`).

2.5.1.3 Schreibzugriffe

Für Schreibzugriffe sind immer Schreibrechte erforderlich.

Schreiben eines Schlüssels (Domäne in der URL)

Der folgende Aufruf schreibt einen Schlüssel in eine Sub-Domäne. Existiert der Schlüssel bereits in der Sub-Domäne, dann wird er überschrieben. Eine noch nicht vorhandene Sub-Domäne wird automatisch angelegt.

Aufruf (POST) `/keyvaluestore/public/value/<domain>/<subdomain>`	
Aufruf-Parameter in der URL	
`<domain>`	Name der Domäne.
`<subdomain>`	Name der Sub-Domäne.
Aufruf-Parameter als JSON-Objekt (Body) Die Informationen über das zu schreibende Schlüssel-Werte-Paar werden als JSON-Objekt im Request-Body übergeben. Es entspricht im Wesentlichen den Ergebnis-Objekten aus den Leseaufrufen. Lediglich die Domänen- und Sub-Domänen-Namen werden direkt in der URL aufgeführt. Sind sie im Objekt vorhanden, dann werden sie dort ignoriert. Das Objekt hat den folgenden Aufbau:	
`key`	Nicht lokalisierter Name des Schlüssels (z.B. `mate` für den Matebestand). Der Schlüsselname muss innerhalb der Sub-Domäne eindeutig sein, wobei Groß- und Kleinbuchstaben unterschieden werden. Der Schlüssel sollte weder Leer- noch Sonderzeichen enthalten, erlaubt sind sie aber. Die Länge wird intern auf 20 Zeichen beschnitten. Der Schlüssel darf nicht `null` sein.
`value`	Wert, der dem Schlüssel zugeordnet ist (z.B. `100` für 100 Mate-Flaschen). Die Länge wird intern auf 100 Zeichen beschnitten. Der Wert darf `null` sein.
`type`	Typ des Werts (z.B. `int`). Die Länge wird intern auf 20 Zeichen beschnitten. Der Wert darf `null` sein.
`unit`	Einheit des Werts (z.B. `Euro`). Die Länge wird intern auf 20 Zeichen beschnitten. Der Wert darf `null` sein.
`localizedUnit`	Lokalisierte Einheit des Wertes (z.B. `Flaschen`). Die Länge wird intern auf 50 Zeichen beschnitten. Der Wert darf `null` sein.
`localizedKey`	Lokalisierter Name des Schlüssels (z.B. `Mate`). Die Länge wird intern auf 50 Zeichen beschnitten. Der Wert darf `null` sein.

`localized-Description`	Lokalisierte textuelle Beschreibung des Eintrags (beispielsweise `Bestand an Mate-Flaschen in der Fachschaft` oder auch `Preis 1 Euro je Flasche`). Die Länge wird intern auf 100 Zeichen beschnitten. Der Wert darf `null` sein.
Authentifizierung Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`CREATED` (201)	Der Aufruf war erfolgreich.
`BAD REQUEST` (400)	Das JSON-Objekt ist nicht vollständig oder korrekt ausgefüllt.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
`FORBIDDEN` (403)	Der Aufrufer selbst besitzt in der Domäne kein Schreibrecht.
`INSUFFICIENT STORAGE` (507)	Die Obergrenze an möglichen Schüsseln in der Sub-Domäne ist erreicht, ob der aber es können keine weiteren Sub-Domains angelegt werden, weil deren Obergrenze innerhalb der Domäne erreicht wurde.
Ergebnis (Body) Der Rumpf der Antwort ist immer leer.	

Schreiben eines Schlüssels (Domäne im Modell)

Der folgende Aufruf schreibt einen Schlüssel in eine Sub-Domäne. Existiert der Schlüssel bereits in der Sub-Domäne, dann wird er überschrieben. Eine noch nicht vorhandene Sub-Domäne wird automatisch angelegt. Der Aufruf entspricht dem vorherigen, außer dass Domäne und Sub-Domäne aus dem Modell ermittelt werden. Dazu müssen die Modellattribute `domain` und `subdomain` (siehe Lesezugriff) im Modellobjekt enthalten sein.

Aufruf (POST) `/keyvaluestore/public/value`	
Aufruf-Parameter als JSON-Objekt (Body) Die Informationen über das zu schreibende Schlüssel-Werte-Paar werden als JSON-Objekt im Request-Body übergeben. Es entspricht im Wesentlichen den Ergebnis-Objekten aus den Leseaufrufen. Das Objekt hat den folgenden Aufbau:	
`domain`	Domäne (Gebiet), in die der Schlüssel geschrieben werden soll (z.B. `studentcouncil` für die Fachschaft).
`subdomain`	Name der Sub-Domäne (Untergebiet) innerhalb einer Domäne, in die der Schlüssel geschrieben werden soll (z.B. `stock` für den Warenbestand).
`key`	Nicht lokalisierter Name des Schlüssels (z.B. `mate` für den Matebestand). Der Schlüsselname muss innerhalb der Sub-Domäne eindeutig sein, wobei Groß- und Kleinbuchstaben unterschieden werden. Der Schlüssel sollte weder Leer- noch Sonderzeichen enthalten, erlaubt sind sie aber. Die Länge wird intern auf 20 Zeichen beschnitten. Der Schlüssel darf nicht `null` sein.
`value`	Wert, der dem Schlüssel zugeordnet ist (z.B. `100` für 100 Mate-Flaschen). Die Länge wird intern auf 100 Zeichen beschnitten. Der Wert darf `null` sein.

`type`	Typ des Werts (z.B. `int`). Die Länge wird intern auf 20 Zeichen beschnitten. Der Wert darf `null` sein.
`unit`	Einheit des Werts (z.B. `Euro`). Die Länge wird intern auf 20 Zeichen beschnitten. Der Wert darf `null` sein.
`localizedUnit`	Lokalisierte Einheit des Wertes (z.B. `Flaschen`). Die Länge wird intern auf 50 Zeichen beschnitten. Der Wert darf `null` sein.
`localizedKey`	Lokalisierter Name des Schlüssels (z.B. `Mate`). Die Länge wird intern auf 50 Zeichen beschnitten. Der Wert darf `null` sein.
`localized-Description`	Lokalisierte textuelle Beschreibung des Eintrags (beispielsweise `Bestand an Mate-Flaschen in der Fachschaft` oder auch `Preis 1 Euro je Flasche`). Die Länge wird intern auf 100 Zeichen beschnitten. Der Wert darf `null` sein.
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`CREATED` (201)	Der Aufruf war erfolgreich.
`BAD REQUEST` (400)	Das JSON-Objekt ist nicht vollständig oder korrekt ausgefüllt.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
`FORBIDDEN` (403)	Der Aufrufer selbst besitzt in der Domäne kein Schreibrecht.
`INSUFFICIENT STORAGE` (507)	Die Obergrenze an möglichen Schüsseln in der Sub-Domäne ist erreicht, ob der aber es können keine weiteren Sub-Domains angelegt werden, weil deren Obergrenze innerhalb der Domäne erreicht wurde.
Ergebnis (Body)	
Der Rumpf der Antwort ist immer leer.	

Schreiben mehrerer Schlüssel (Domäne in der URL)

Der folgende Aufruf schreibt mehrere Schlüssel innerhalb einer Transaktion in eine Sub-Domäne. Existiert einer der Schlüssel bereits in der Sub-Domäne, dann wird er überschrieben. Eine noch nicht vorhandene Sub-Domäne wird automatisch angelegt.

Aufruf (POST)	
`/keyvaluestore/public/values/<domain>/<subdomain>`	
Aufruf-Parameter in der URL	
`<domain>`	Name der Domäne.
`<subdomain>`	Name der Sub-Domäne.
Aufruf-Parameter als JSON-Array (Body)	
Die Informationen über die zu schreibenden Schlüssel-Werte-Paare werden als Array mit JSON-Objekten im Request-Body übergeben. Der Aufbau der Objekte entspricht im Wesentlichen den Ergebnis-Objekten aus den Leseaufrufen. Lediglich die Domänen- und Sub-Domänen-Namen werden direkt in der URL aufgeführt. Sind sie im Objekt vorhanden, dann werden sie dort ignoriert. Die Objekte haben den folgenden Aufbau:	

`key`	Nicht lokalisierter Name des Schlüssels (z.B. `mate` für den Matebestand). Der Schlüsselname muss innerhalb der Sub-Domäne eindeutig sein, wobei Groß- und Kleinbuchstaben unterschieden werden. Der Schlüssel sollte weder Leer- noch Sonderzeichen enthalten, erlaubt sind sie aber. Die Länge wird intern auf 20 Zeichen beschnitten. Der Schlüssel darf nicht `null` sein.
`value`	Wert, der dem Schlüssel zugeordnet ist (z.B. `100` für 100 Mate-Flaschen). Die Länge wird intern auf 100 Zeichen beschnitten. Der Wert darf `null` sein.
`type`	Typ des Werts (z.B. `int`). Die Länge wird intern auf 20 Zeichen beschnitten. Der Wert darf `null` sein.
`unit`	Einheit des Werts (z.B. `Euro`). Die Länge wird intern auf 20 Zeichen beschnitten. Der Wert darf `null` sein.
`localizedUnit`	Lokalisierte Einheit des Wertes (z.B. `Flaschen`). Die Länge wird intern auf 50 Zeichen beschnitten. Der Wert darf `null` sein.
`localizedKey`	Lokalisierter Name des Schlüssels (z.B. `Mate`). Die Länge wird intern auf 50 Zeichen beschnitten. Der Wert darf `null` sein.
`localized-Description`	Lokalisierte textuelle Beschreibung des Eintrags (beispielsweise `Bestand an Mate-Flaschen in der Fachschaft` oder auch `Preis 1 Euro je Flasche`). Die Länge wird intern auf 100 Zeichen beschnitten. Der Wert darf `null` sein.
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`CREATED` (201)	Der Aufruf war erfolgreich.
`BAD REQUEST` (400)	Das JSON-Objekt ist nicht vollständig oder korrekt ausgefüllt.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
`FORBIDDEN` (403)	Der Aufrufer selbst besitzt in der Domäne kein Schreibrecht.
`INSUFFICIENT STORAGE` (507)	Die Obergrenze an möglichen Schüsseln in der Sub-Domäne ist erreicht, ob der aber es können keine weiteren Sub-Domains angelegt werden, weil deren Obergrenze innerhalb der Domäne erreicht wurde.
Ergebnis (Body)	
Der Rumpf der Antwort ist immer leer.	

Schreiben mehrerer Schlüssels (Domänen im Modell)

Der folgende Aufruf schreibt mehrere Schlüssel in eine oder mehrere Sub-Domänen. Im Gegensatz zum vorherigen Aufruf dürfen die Modellobjekte in verschiedenen Domänen und Sub-Domänen geschrieben werden.

Aufruf (POST)
`/keyvaluestore/public/values`
Aufruf-Parameter als JSON-Array (Body)
Die Informationen über die zu schreibenden Schlüssel-Werte-Paare werden als Array mit JSON-Objekten im Request-Body übergeben. Der Aufbau der Objekte entspricht im Wesentlichen den Ergebnis-Objekten aus den Leseaufrufen. Die Objekte haben den folgenden Aufbau:

`domain`	Domäne (Gebiet), in die der Schlüssel geschrieben werden soll (z.B. `studentcouncil` für die Fachschaft).
`subdomain`	Name der Sub-Domäne (Untergebiet) innerhalb einer Domäne, in die der Schlüssel geschrieben werden soll (z.B. `stock` für den Warenbestand).
`key`	Nicht lokalisierter Name des Schlüssels (z.B. `mate` für den Matebestand). Der Schlüsselname muss innerhalb der Sub-Domäne eindeutig sein, wobei Groß- und Kleinbuchstaben unterschieden werden. Der Schlüssel sollte weder Leer- noch Sonderzeichen enthalten, erlaubt sind sie aber. Die Länge wird intern auf 20 Zeichen beschnitten. Der Schlüssel darf nicht `null` sein.
`value`	Wert, der dem Schlüssel zugeordnet ist (z.B. `100` für 100 Mate-Flaschen). Die Länge wird intern auf 100 Zeichen beschnitten. Der Wert darf `null` sein.
`type`	Typ des Werts (z.B. `int`). Die Länge wird intern auf 20 Zeichen beschnitten. Der Wert darf `null` sein.
`unit`	Einheit des Werts (z.B. `Euro`). Die Länge wird intern auf 20 Zeichen beschnitten. Der Wert darf `null` sein.
`localizedUnit`	Lokalisierte Einheit des Wertes (z.B. `Flaschen`). Die Länge wird intern auf 50 Zeichen beschnitten. Der Wert darf `null` sein.
`localizedKey`	Lokalisierter Name des Schlüssels (z.B. `Mate`). Die Länge wird intern auf 50 Zeichen beschnitten. Der Wert darf `null` sein.
`localized-Description`	Lokalisierte textuelle Beschreibung des Eintrags (beispielsweise `Bestand an Mate-Flaschen in der Fachschaft` oder auch `Preis 1 Euro je Flasche`). Die Länge wird intern auf 100 Zeichen beschnitten. Der Wert darf `null` sein.
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`CREATED` (201)	Der Aufruf war erfolgreich.
`BAD REQUEST` (400)	Das JSON-Objekt ist nicht vollständig oder korrekt ausgefüllt.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
`FORBIDDEN` (403)	Der Aufrufer selbst besitzt in der Domäne kein Schreibrecht.
`INSUFFICIENT STORAGE` (507)	Die Obergrenze an möglichen Schüsseln in der Sub-Domäne ist erreicht, ob der aber es können keine weiteren Sub-Domains angelegt werden, weil deren Obergrenze innerhalb der Domäne erreicht wurde.
Ergebnis (Body)	
Der Rumpf der Antwort ist immer leer.	

Löschen eines Schlüssels

Der folgende Aufruf entfernt einen Schlüssel aus einer Sub-Domäne. Existiert der Schlüssel nicht in der Sub-Domäne, dann tritt kein Fehler auf. Ist die Sub-Domäne nach dem Löschen des Schlüssels leer, dann wird sie automatisch gelöscht.

Aufruf (DELETE)
`/keyvaluestore/public/value/<domain>/<subdomain>/<key>`

Aufruf-Parameter in der URL	
`<domain>`	Name der Domäne.
`<subdomain>`	Name der Sub-Domäne.
`<key>`	Name des zu löschenden Schlüssels.
Authentifizierung Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
`FORBIDDEN` (403)	Der Aufrufer selbst besitzt in der Domäne kein Schreibrecht.
Ergebnis (Body) Der Rumpf der Antwort ist immer leer.	

2.5.2 Private Daten

Im privaten Bereich können anwenderspezifische Informationen abgelegt werden. Auf diese Informationen hat nur der Anwender selbst lesenden und schreibenden Zugriff. Alle Daten werden in der Datenbank unverschlüsselt abgelegt. Es werden keine Domänen-Namen vergeben, weil der ADS-Login-Name des Anwenders als Domäne dient. Lediglich die Sub-Domänen existieren genau so wie bei den öffentlichen Daten.

2.5.2.1 Lesezugriffe

Lesezugriffe sind nicht öffentlich.

Auslesen aller Schlüssel

Der folgende Aufruf liest alle Schlüssel einer Sub-Domäne aus.

Aufruf (GET) `/keyvaluestore/private/values/<subdomain>`	
Aufruf-Parameter in der URL	
`<subdomain>`	Name der Sub-Domäne.
Authentifizierung Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
Ergebnis (Body als JSON-Array) Das Ergebnis ist ein JSON-Array mit Objekten, die jeweils ein Schlüssel-/Werte-Paar beschreiben. Ist das Array leer, dann sind in der Sub-Domäne keine Paare enthalten.	

`domain`	Domäne (Gebiet), enspricht dem ADS-Login-Namen des Aufrufers.
`subdomain`	Name der Sub-Domäne (Untergebiet) innerhalb einer Domäne, aus der der Schlüssel stammt (z.B. `login`).
`key`	Nicht lokalisierter Name des Schlüssels (z.B. `lastlogin` für den Zeitpunkt der letzten Anmeldung).
`value`	Wert, der dem Schlüssel zugeordnet ist.
`type`	Typ des Werts (z.B. `int`).
`unit`	Einheit des Werts.
`localizedUnit`	Lokalisierte Einheit des Wertes.
`localizedKey`	Lokalisierter Name des Schlüssels.
`localized-Description`	Lokalisierte textuelle Beschreibung des Eintrags.

Auslesen eines Schlüssels

Der folgende Aufruf liest einen Schlüssel einer Sub-Domäne aus.

Aufruf (GET)	
`/keyvaluestore/private/value/<subdomain>/<key>`	
Aufruf-Parameter in der URL	
`<subdomain>`	Name der Sub-Domäne.
`<key>`	Name des Schlüssels.
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
`NOT FOUND` (404)	Der gesuchte Schlüssel existiert nicht in der Sub-Domäne.
Ergebnis (Body als JSON-Objekt)	
Das Ergebnis ist ein JSON-Objekt, das ein Schlüssel-/Werte-Paar beschreiben. Dessen Aufbau stimmt mit dem beim Auslesen aller Schlüssel einer Sub-Domäne beschriebenen überein.	
`domain`	Domäne (Gebiet), enspricht dem ADS-Login-Namen des Aufrufers.
`subdomain`	Name der Sub-Domäne (Untergebiet) innerhalb einer Domäne, aus der der Schlüssel stammt (z.B. `login`)).
`key`	Nicht lokalisierter Name des Schlüssels (z.B. `lastlogin` für den Zeitpunkt der letzten Anmeldung).
`value`	Wert, der dem Schlüssel zugeordnet ist.
`type`	Typ des Werts (z.B. `int`).
`unit`	Einheit des Werts.
`localizedUnit`	Lokalisierte Einheit des Wertes.
`localizedKey`	Lokalisierter Name des Schlüssels.

`localized-Description`	Lokalisierte textuelle Beschreibung des Eintrags.

2.5.2.2 Schreibzugriffe

Auch für Schreibzugriffe sind immer Anmeldedaten erforderlich.

Schreiben eines Schlüssels (Domäne in der URL)

Der folgende Aufruf schreibt einen Schlüssel in eine Sub-Domäne. Existiert der Schlüssel bereits in der Sub-Domäne, dann wird er überschrieben. Eine noch nicht vorhandene Sub-Domäne wird automatisch angelegt.

Aufruf (POST)	
`/keyvaluestore/private/value/<subdomain>`	
Aufruf-Parameter in der URL	
`<subdomain>`	Name der Sub-Domäne.
Aufruf-Parameter als JSON-Objekt (Body)	
Die Informationen über das zu schreibende Schlüssel-Werte-Paar werden als JSON-Objekt im Request-Body übergeben. Es entspricht im Wesentlichen den Ergebnis-Objekten aus den Leseaufrufen. Lediglich der Sub-Domänen-Namen wird direkt in der URL aufgeführt. Ist er im Objekt vorhanden, dann wird er dort ignoriert. Das Objekt hat den folgenden Aufbau:	
`key`	Nicht lokalisierter Name des Schlüssels (z.B. `balance`). Der Schlüsselname muss innerhalb der Sub-Domäne eindeutig sein, wobei Groß- und Kleinbuchstaben unterschieden werden. Der Schlüssel sollte weder Leer- noch Sonderzeichen enthalten, erlaubt sind sie aber. Die Länge wird intern auf 20 Zeichen beschnitten. Der Schlüssel darf nicht `null` sein.
`value`	Wert, der dem Schlüssel zugeordnet ist (z.B. `100`). Die Länge wird intern auf 100 Zeichen beschnitten. Der Wert darf `null` sein.
`type`	Typ des Werts (z.B. `int`). Die Länge wird intern auf 20 Zeichen beschnitten. Der Wert darf `null` sein.
`unit`	Einheit des Werts (z.B. `eur`). Die Länge wird intern auf 20 Zeichen beschnitten. Der Wert darf `null` sein.
`localizedUnit`	Lokalisierte Einheit des Wertes. Die Länge wird intern auf 50 Zeichen beschnitten. Der Wert darf `null` sein.
`localizedKey`	Lokalisierter Name des Schlüssels (z.B. `Euro`). Die Länge wird intern auf 50 Zeichen beschnitten. Der Wert darf `null` sein.
`localized-Description`	Lokalisierte textuelle Beschreibung des Eintrags. Die Länge wird intern auf 100 Zeichen beschnitten. Der Wert darf `null` sein.
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`CREATED` (201)	Der Aufruf war erfolgreich.
`BAD REQUEST` (400)	Das JSON-Objekt ist nicht vollständig oder korrekt ausgefüllt.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
`FORBIDDEN` (403)	Der Aufrufer selbst besitzt in der Domäne kein Schreibrecht.

`INSUFFICIENT STORAGE` (507)	Die Obergrenze an möglichen Schüsseln in der Sub-Domäne ist erreicht, ob der aber es können keine weiteren Sub-Domains angelegt werden, weil deren Obergrenze für den Aufrufer erreicht wurde.
Ergebnis (Body)	
Der Rumpf der Antwort ist immer leer.	

Schreiben eines Schlüssels (Sub-Domäne im Modell)

Der folgende Aufruf schreibt einen Schlüssel in eine Sub-Domäne. Existiert der Schlüssel bereits in der Sub-Domäne, dann wird er überschrieben. Eine noch nicht vorhandene Sub-Domäne wird automatisch angelegt. Der Aufruf entspricht dem vorherigen, außer dass die Sub-Domäne aus dem Modell ermittelt wird. Dazu muss das Modellattribut `subdomain` (siehe Lesezugriff) im Modellobjekt enthalten sein.

Aufruf (POST)	
`/keyvaluestore/public/value`	
Aufruf-Parameter als JSON-Objekt (Body)	
Die Informationen über das zu schreibende Schlüssel-Werte-Paar werden als JSON-Objekt im Request-Body übergeben. Es entspricht im Wesentlichen den Ergebnis-Objekten aus den Leseaufrufen. Das Objekt hat den folgenden Aufbau:	
`subdomain`	Name der Sub-Domäne (Untergebiet), in die der Schlüssel geschrieben werden soll (z.B. `account` für das Konto).
`key`	Nicht lokalisierter Name des Schlüssels (z.B. `balance` für den Kontostand). Der Schlüsselname muss innerhalb der Sub-Domäne eindeutig sein, wobei Groß- und Kleinbuchstaben unterschieden werden. Der Schlüssel sollte weder Leer- noch Sonderzeichen enthalten, erlaubt sind sie aber. Die Länge wird intern auf 20 Zeichen beschnitten. Der Schlüssel darf nicht `null` sein.
`value`	Wert, der dem Schlüssel zugeordnet ist (z.B. `100`). Die Länge wird intern auf 100 Zeichen beschnitten. Der Wert darf `null` sein.
`type`	Typ des Werts (z.B. `int`). Die Länge wird intern auf 20 Zeichen beschnitten. Der Wert darf `null` sein.
`unit`	Einheit des Werts (z.B. `eur`). Die Länge wird intern auf 20 Zeichen beschnitten. Der Wert darf `null` sein.
`localizedUnit`	Lokalisierte Einheit des Wertes (z.B. `Euro`). Die Länge wird intern auf 50 Zeichen beschnitten. Der Wert darf `null` sein.
`localizedKey`	Lokalisierter Name des Schlüssels (z.B. `Kontostand`). Die Länge wird intern auf 50 Zeichen beschnitten. Der Wert darf `null` sein.
`localized-Description`	Lokalisierte textuelle Beschreibung des Eintrags. Die Länge wird intern auf 100 Zeichen beschnitten. Der Wert darf `null` sein.
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`CREATED` (201)	Der Aufruf war erfolgreich.
`BAD REQUEST` (400)	Das JSON-Objekt ist nicht vollständig oder korrekt ausgefüllt.

`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
`FORBIDDEN` (403)	Der Aufrufer selbst besitzt in der Domäne kein Schreibrecht.
`INSUFFICIENT STORAGE` (507)	Die Obergrenze an möglichen Schüsseln in der Sub-Domäne ist erreicht, ob der aber es können keine weiteren Sub-Domains angelegt werden, weil deren Obergrenze für den Aufrufer erreicht wurde.
Ergebnis (Body)	
Der Rumpf der Antwort ist immer leer.	

Schreiben mehrerer Schlüssel (Sub-Domäne in der URL)

Der folgende Aufruf schreibt mehrere Schlüssel innerhalb einer Transaktion in eine Sub-Domäne. Existiert einer der Schlüssel bereits in der Sub-Domäne, dann wird er überschrieben. Eine noch nicht vorhandene Sub-Domäne wird automatisch angelegt.

Aufruf (POST)	
`/keyvaluestore/private/values/<subdomain>`	
Aufruf-Parameter in der URL	
`<subdomain>`	Name der Sub-Domäne.
Aufruf-Parameter als JSON-Array (Body)	
Die Informationen über die zu schreibenden Schlüssel-Werte-Paare werden als Array mit JSON-Objekten im Request-Body übergeben. Der Aufbau der Objekte entspricht im Wesentlichen den Ergebnis-Objekten aus den Leseaufrufen. Lediglich der Sub-Domänen-Name wird direkt in der URL aufgeführt. Ist er im Objekt vorhanden, dann wird er dort ignoriert. Die Objekte haben den folgenden Aufbau:	
`key`	Nicht lokalisierter Name des Schlüssels (z.B. `balance` für den Kontostand). Der Schlüsselname muss innerhalb der Sub-Domäne eindeutig sein, wobei Groß- und Kleinbuchstaben unterschieden werden. Der Schlüssel sollte weder Leer- noch Sonderzeichen enthalten, erlaubt sind sie aber. Die Länge wird intern auf 20 Zeichen beschnitten. Der Schlüssel darf nicht `null` sein.
`value`	Wert, der dem Schlüssel zugeordnet ist (z.B. `100`). Die Länge wird intern auf 100 Zeichen beschnitten. Der Wert darf `null` sein.
`type`	Typ des Werts (z.B. `int`). Die Länge wird intern auf 20 Zeichen beschnitten. Der Wert darf `null` sein.
`unit`	Einheit des Werts (z.B. `eur`). Die Länge wird intern auf 20 Zeichen beschnitten. Der Wert darf `null` sein.
`localizedUnit`	Lokalisierte Einheit des Wertes (z.B. `Euro`). Die Länge wird intern auf 50 Zeichen beschnitten. Der Wert darf `null` sein.
`localizedKey`	Lokalisierter Name des Schlüssels (z.B. `Kontostand`). Die Länge wird intern auf 50 Zeichen beschnitten. Der Wert darf `null` sein.
`localized-Description`	Lokalisierte textuelle Beschreibung des Eintrags. Die Länge wird intern auf 100 Zeichen beschnitten. Der Wert darf `null` sein.
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	

`CREATED` (201)	Der Aufruf war erfolgreich.
`BAD REQUEST` (400)	Das JSON-Objekt ist nicht vollständig oder korrekt ausgefüllt.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
`FORBIDDEN` (403)	Der Aufrufer selbst besitzt in der Domäne kein Schreibrecht.
`INSUFFICIENT STORAGE` (507)	Die Obergrenze an möglichen Schüsseln in der Sub-Domäne ist erreicht, ob der aber es können keine weiteren Sub-Domains angelegt werden, weil deren Obergrenze für den Aufrufer erreicht wurde.
Ergebnis (Body)	
Der Rumpf der Antwort ist immer leer.	

Schreiben mehrerer Schlüssel (Sub-Domänen in den Modellen)

Der folgende Aufruf schreibt mehrere Schlüssel in eine oder mehrere Sub-Domänen. Im Gegensatz zum vorherigen Aufruf dürfen die Modellobjekte in verschiedenen Domänen und Sub-Domänen geschrieben werden.

Aufruf (POST)	
`/keyvaluestore/private/values`	
Aufruf-Parameter als JSON-Array (Body)	
Die Informationen über die zu schreibenden Schlüssel-Werte-Paare werden als Array mit JSON-Objekten im Request-Body übergeben. Der Aufbau der Objekte entspricht im Wesentlichen den Ergebnis-Objekten aus den Leseaufrufen. Die Objekte haben den folgenden Aufbau:	
`subdomain`	Name der Sub-Domäne (Untergebiet), in die der Schlüssel geschrieben werden soll (z.B. `account` für das Konto).
`key`	Nicht lokalisierter Name des Schlüssels (z.B. `balance` für den Kontostand). Der Schlüsselname muss innerhalb der Sub-Domäne eindeutig sein, wobei Groß- und Kleinbuchstaben unterschieden werden. Der Schlüssel sollte weder Leer- noch Sonderzeichen enthalten, erlaubt sind sie aber. Die Länge wird intern auf 20 Zeichen beschnitten. Der Schlüssel darf nicht `null` sein.
`value`	Wert, der dem Schlüssel zugeordnet ist (z.B. `100`). Die Länge wird intern auf 100 Zeichen beschnitten. Der Wert darf `null` sein.
`type`	Typ des Werts (z.B. `int`). Die Länge wird intern auf 20 Zeichen beschnitten. Der Wert darf `null` sein.
`unit`	Einheit des Werts (z.B. `eur`). Die Länge wird intern auf 20 Zeichen beschnitten. Der Wert darf `null` sein.
`localizedUnit`	Lokalisierte Einheit des Wertes (z.B. `Euro`). Die Länge wird intern auf 50 Zeichen beschnitten. Der Wert darf `null` sein.
`localizedKey`	Lokalisierter Name des Schlüssels (z.B. `Kontostand`). Die Länge wird intern auf 50 Zeichen beschnitten. Der Wert darf `null` sein.
`localized-Description`	Lokalisierte textuelle Beschreibung des Eintrags. Die Länge wird intern auf 100 Zeichen beschnitten. Der Wert darf `null` sein.
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	

CREATED (201)	Der Aufruf war erfolgreich.
BAD REQUEST (400)	Das JSON-Objekt ist nicht vollständig oder korrekt ausgefüllt.
UNAUTHORIZED (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
FORBIDDEN (403)	Der Aufrufer selbst besitzt in der Domäne kein Schreibrecht.
INSUFFICIENT STORAGE (507)	Die Obergrenze an möglichen Schüsseln in der Sub-Domäne ist erreicht, ob der aber es können keine weiteren Sub-Domains angelegt werden, weil deren Obergrenze für den Aufrufer erreicht wurde.
Ergebnis (Body)	
Der Rumpf der Antwort ist immer leer.	

Löschen eines Schlüssels

Der folgende Aufruf entfernt einen Schlüssel aus einer Sub-Domäne. Existiert der Schlüssel nicht in der Sub-Domäne, dann tritt kein Fehler auf. Ist die Sub-Domäne nach dem Löschen des Schlüssels leer, dann wird sie automatisch gelöscht.

Aufruf (DELETE)	
/keyvaluestore/private/value/<subdomain>/<key>	
Aufruf-Parameter in der URL	
<subdomain>	Name der Sub-Domäne.
<key>	Name des zu löschenden Schlüssels.
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
OK (200)	Der Aufruf war erfolgreich.
UNAUTHORIZED (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
Ergebnis (Body)	
Der Rumpf der Antwort ist immer leer.	

2.6 Informationen zum Studium

Die Schnittstellen lesen Daten aus dem Noteninformationssystem QIS aus.

2.6.1 Notenblatt

Diese Schnittstelle liefert die Noten aus dem Online-Service der Hochschule.

2.6.1.1 Alle Noten

Der Aufruf liefert die Noten aller Fachprüfungen laut SPO, die Noten der Einzelprüfungen sowie die Mittelwerte laut SPO.

Aufruf (GET)
/grades/all

Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
`INTERNAL SERVER ERROR` (500)	Wenn die Kommunikation mit dem externen Server fehlschlägt oder die QIS-Seite nicht geparst werden kann.
Ergebnis (Body als JSON-Objekt)	
Das Ergebnis ist ein JSON-Objekt mit folgendem Aufbau:	
`basicStudyPeriod`	Der Wert ist `true`, wenn der Studiengang ein Grundstudium besitzt, ansonsten `false`.
`avarageBasic StudyPeriod`	Durchschnittsnote im Grundstudium, mit 100 multipliziert, um eine ganze Zahl zu erhalten. Die Note wird anhand der Gewichtung der Module in der Prüfungsordnung berechnet. Gibt es kein Grundstudium oder wurde dort noch keine Prüfung bestanden, dann ist der Wert `-1`.
`avarageMain StudyPeriod`	Durchschnittsnote im Hauptstudium, mit 100 multipliziert, um eine ganze Zahl zu erhalten. Die Note wird anhand der Gewichtung der Module in der Prüfungsordnung berechnet. Wurde dort noch keine Prüfung bestanden, dann ist der Wert `-1`.
`grades`	JSON-Array mit Noten-Objekten, deren Aufbau im folgenden Abschnitt beschrieben wird. Das Array enthält die Einzelnoten aller Prüfungen sowie die Modulnoten.
`gradesBasic StudyPeriod`	JSON-Array mit Noten-Objekten aus dem Grundstudium. Weist der Studiengang (z.B. Master) keine Unterscheidung zwischen Grund- und Hauptstudium auf, so ist das Array leer. Die Noten in diesem Array entsprechen den Fachprüfungen aus der Studien- und Prüfungsordnung.
`gradesMain StudyPeriod`	JSON-Array mit Noten-Objekten aus dem Hauptstudium. Die Noten in diesem Array entsprechen den Fachprüfungen aus der Studien- und Prüfungsordnung. Weist der Studiengang (z.B. Master) keine Unterscheidung zwischen Grund- und Hauptstudium auf, so befinden sich alle Fachprüfungen in diesem Array.

Aufbau eines Noten-Objektes im Array `grades`:	
`examNumber`	Prüfungsnummer aus dem Modulhandbuch.
`examName`	Name der Prüfung.
`examSemester`	Textuelle Angabe des Semesters, in dem der Studierende an der Prüfung teilgenommen hat oder an dem sie als externe Studienleistung anerkannt wurde. Das Format entspricht „SS 2015“ für das Sommersemester 2015 bzw. „WS 2015“ für das Wintersemester 2015/2016.
`date`	Datum, an dem die Note eingetragen wurde.
`grade`	Note, mit 100 multipliziert. Bei unbenoteten Prüfungen steht die 0 für „bestanden“, die 99 für „nicht bestanden“.
`status`	Status der Prüfung (z.B. „bestanden“).
`externalGrade`	`true`, wenn es sich um eine anerkannte, externe Prüfungsleistung handelt, ansonsten `false`.

`comment`	Zusätzlicher Kommentar zur Prüfung.
`attempt`	Nummer des Versuchs, um die Prüfung zu bestehen bzw. Nummer des Versuchs, mit dem Prüfung bestanden wurde. Die Zählung beginnt mit 1.
`weighting`	Gewichtung dieer Note laut Prüfungsordnung für die Berechnung der Abschlussnote.

2.6.1.2 Noten der Fachprüfungen laut SPO

Der Aufruf liefert die Noten aller Fachprüfungen laut SPO und berechnet den Durchschnittswert. Die Noten der Einzelprüfungen werden nicht ermittelt. Der Aufbau des Ergebnisses entspricht dem aus dem Aufruf in Abschnitt 2.6.1.1.

Aufruf (GET)	
`/grades/examinationregulations`	
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
`INTERNAL SERVER ERROR` (500)	Wenn die Kommunikation mit dem externen Server fehlschlägt oder die QIS-Seite nicht geparst werden kann.
Ergebnis (Body als JSON-Objekt)	
Das Ergebnis ist ein JSON-Objekt mit folgendem Aufbau:	
`basicStudyPeriod`	Der Wert ist `true`, wenn der Studiengang ein Grundstudium besitzt, ansonsten `false`.
`avarageBasic StudyPeriod`	Durchschnittsnote im Grundstudium, mit 100 multipliziert, um eine ganze Zahl zu erhalten. Die Note wird anhand der Gewichtung der Module in der Prüfungsordnung berechnet. Gibt es kein Grundstudium oder wurde dort noch keine Prüfung bestanden, dann ist der Wert `-1`.
`avarageMain StudyPeriod`	Durchschnittsnote im Hauptstudium, mit 100 multipliziert, um eine ganze Zahl zu erhalten. Die Note wird anhand der Gewichtung der Module in der Prüfungsordnung berechnet. Wurde dort noch keine Prüfung bestanden, dann ist der Wert `-1`.
`grades`	Das Array ist immer leer, weil die Einzelnoten nicht zurückgegeben werden.
`gradesBasic StudyPeriod`	JSON-Array mit Noten-Objekten aus dem Grundstudium. Weist der Studiengang (z.B. Master) keine Unterscheidung zwischen Grund- und Hauptstudium auf, so ist das Array leer. Die Noten in diesem Array entsprechen den Fachprüfungen aus der Studien- und Prüfungsordnung.
`gradesMain StudyPeriod`	JSON-Array mit Noten-Objekten aus dem Hauptstudium. Die Noten in diesem Array entsprechen den Fachprüfungen aus der Studien- und Prüfungsordnung. Weist der Studiengang (z.B. Master) keine Unterscheidung zwischen Grund- und Hauptstudium auf, so befinden sich alle Fachprüfungen in diesem Array.

Aufbau eines Noten-Objektes im Array `grades`:

examNumber	Prüfungsnummer aus dem Modulhandbuch.
examName	Name der Prüfung.
examSemester	Textuelle Angabe des Semesters, in dem der Studierende an der Prüfung teilgenommen hat oder an dem sie als externe Studienleistung anerkannt wurde. Das Format entspricht „SS 2015“ für das Sommersemester 2015 bzw. „WS 2015“ für das Wintersemester 2015/2016.
date	Datum, an dem die Note eingetragen wurde.
grade	Note, mit 100 multipliziert. Bei unbenoteten Prüfungen steht die 0 für „bestanden“, die 99 für „nicht bestanden“.
status	Status der Prüfung (z.B. „bestanden“).
externalGrade	true, wenn es sich um eine anerkannte, externe Prüfungsleistung handelt, ansonsten false.
comment	Zusätzlicher Kommentar zur Prüfung.
attempt	Nummer des Versuchs, um die Prüfung zu bestehen bzw. Nummer des Versuchs, mit dem Prüfung bestanden wurde. Die Zählung beginnt mit 1.

2.6.1.3 Noten-Statistik einer Prüfung

Der Aufruf ist veraltet. Statt dessen sollte nur noch 2.6.1.4 verwendet werden.
Der Aufruf liefert die Noten-Statistik aller Teilnehmer einer Prüfung. Aufgrund der sehr langen Antwortzeiten des QIS-Servers werden die Statistiken intern in einem Cache gehalten, der lediglich wöchentlich aktualisiert wird. Bei nachträglichen Notenänderungen im QIS sind die Statistiken also möglicherweise nicht immer ganz aktuell. Aufgrund des Cachings ergeben sich bei Verwendung dieses REST-Aufrufs auch unterschiedliche Antwortzeiten: Liegen die Daten im Cache und sind nicht mehr als eine Woche alt, dann erfolgt die Antwort in wenigen Millisekunden. Beim Zugriff auf den QIS-Server sind Antwortzeiten von 10 Sekunden nicht ungewöhnlich.

Aufruf (GET)	
/grades/statistics/<p-nr>/<p-semester>	
Aufruf-Parameter in der URL	
<p-nr>	Interne Nummer der Prüfung, wie sie im QIS vergeben ist (siehe Attribut examNumber in den Abschnitten 2.6.1.1 und 2.6.1.2).
<p-semester>	Semester, in dem die Prüfung stattfand. (siehe Attribut examSemester in den Abschnitten 2.6.1.1 und 2.6.1.2).
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
OK (200)	Der Aufruf war erfolgreich.
UNAUTHORIZED (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
INTERNAL SERVER ERROR (500)	Wenn die Kommunikation mit dem externen Server fehlschlägt oder die QIS-Seite nicht geparst werden kann.
Ergebnis (Body als JSON-Objekt)	

Liegt keine Statistik für eine angefragte Note vor, so ist das Ergebnis `null`. Ansonsten ist das Ergebnis ist ein JSON-Objekt mit folgendem Aufbau:	
`examNumber`	Interne Nummer der Prüfung, entspricht dem Aufrufparameter `<p-nr>`.
`examName`	Vollständiger Name der Prüfung.
`examSemester`	Semester, in dem die Prüfung abgehalten wurde, entspricht dem Aufrufparameter `<p-semester>`.
`status`	Normalerweise eine leere Zeichenkette. Bei einem internen Serverfehler (Status-Code 500) enthält Status eine genaue Fehlerbeschreibung.
`grading`	Dieses Array enthält Objekte mit den Notenverteilungen in fünf Notenbereichen. Der Aufbau dieser Objekte ist in der Folgetabelle beschrieben. Dabei enthalten diese Indizes die folgenden Werte: ■ `0`: Teilnehmer mit einer sehr guten Note. ■ `1`: Teilnehmer mit einer guten Note. ■ `2`: Teilnehmer mit einer befriedigenden Note. ■ `3`: Teilnehmer mit einer ausreichenden Note. ■ `4`: Teilnehmer mit einer nicht ausreichenden Note.
`attendees`	Anzahl an Teilnehmern an der Prüfung. Der Wert entspricht der Summe der Einträge im Array `grading`.
`averageGrade`	Durchschnittsnote, mit 100 multipliziert.

Aufbau eines Notenbereichs-Objektes im Array `grading`:	
`lowerLimit`	Notenuntergrenze des Bereichs.
`upperLimit`	Notenobergrenze des Bereichs.
`count`	Anzahl an Prüfungsteilnehmern, deren Noten in diesem Bereich liegen.

2.6.1.4 Noten-Statistiken ausgewählter Prüfungen

Der Aufruf liefert die Noten-Statistiken aller Teilnehmer ausgewählter Prüfung. Aufgrund der sehr langen Antwortzeiten des QIS-Servers werden die Statistiken intern in einem Cache gehalten, der lediglich wöchentlich aktualisiert wird. Bei nachträglichen Notenänderungen im QIS sind die Statistiken also möglicherweise nicht immer ganz aktuell. Aufgrund des Cachings ergeben sich bei Verwendung dieses REST-Aufrufs auch unterschiedliche Antwortzeiten: Liegen die Daten im Cache und sind nicht mehr als eine Woche alt, dann erfolgt die Antwort in wenigen Millisekunden. Beim Zugriff auf den QIS-Server sind Antwortzeiten von 10 Sekunden nicht ungewöhnlich.

Aufruf (GET) `/grades/statistics/<p-selektor>`	
Aufruf-Parameter in der URL	
`<p-selector>`	Paare, bestehend aus der internen Nummer der Prüfung als Schlüssel sowie dem Semester, in dem die Prüfung stattfand, als zugehöriger Wert. Alle Paare werden durch `&` voneinander getrennt. Die Prüfungsnummer entspricht der internen des QIS (siehe Attribut `examNumber` in den Abschnitten 2.6.1.1 und 2.6.1.2). Das Semester, in dem die Prüfung stattfand, wird in den Aufrufen 2.6.1.1 und 2.6.1.2 im Attribut `examSemester` zurückgegeben. Beispiel: `IB 210=WS 2014&IB 310=SS 2014&IB 320=SS 2014`
Authentifizierung	

Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
`INTERNAL SERVER ERROR` (500)	Wenn die Kommunikation mit dem externen Server fehlschlägt oder die QIS-Seite nicht geparst werden kann.
Ergebnis (Body als JSON-Objekt) Das Ergebnis ist ein JSON-Array mit Objekten, die den folgenden Aufbau besitzen (liegt für eine angefragte Prüfung keine Statistik vor, so fehlt für diese Prüfung der Statistik-Eintrag im Array):	
`examNumber`	Interne Nummer der Prüfung, entspricht dem Aufrufparameter `<p-nr>`.
`examName`	Vollständiger Name der Prüfung.
`examSemester`	Semester, in dem die Prüfung abgehalten wurde, entspricht dem Aufrufparameter `<p-semester>`.
`status`	Normalerweise eine leere Zeichenkette. Bei einem internen Serverfehler (Status-Code 500) enthält Status eine genaue Fehlerbeschreibung.
`grading`	Dieses Array enthält Objekte mit den Notenverteilungen in fünf Notenbereichen. Der Aufbau dieser Objekte ist in der Folgetabelle beschrieben. Dabei enthalten diese Indizes die folgenden Werte: ▪ `0`: Teilnehmer mit einer sehr guten Note. ▪ `1`: Teilnehmer mit einer guten Note. ▪ `2`: Teilnehmer mit einer befriedigenden Note. ▪ `3`: Teilnehmer mit einer ausreichenden Note. ▪ `4`: Teilnehmer mit einer nicht ausreichenden Note.
`attendees`	Anzahl an Teilnehmern an der Prüfung. Der Wert entspricht der Summe der Einträge im Array `grading`.
`averageGrade`	Durchschnittsnote, mit 100 multipliziert.

Aufbau eines Notenbereichs-Objektes im Array `grading`:	
`lowerLimit`	Notenuntergrenze des Bereichs.
`upperLimit`	Notenobergrenze des Bereichs.
`count`	Anzahl an Prüfungsteilnehmern, deren Noten in diesem Bereich liegen.

2.6.2 Bescheinigungen

Mit diesen Aufrufen kann auf Bescheinigungen aus dem QIS zugegriffen werden. Dazu gehören z.B. die Immatrikulations- sowie die KVV-Bescheinigung. Es werden zwei Zugriffsvarianten unterstützt:

1. Direkter Zugriff auf die Noten im QIS mit dem Aufruf aus 2.6.2.1.
2. Zugriff über einen Cache mit den Aufrufen 2.6.2.2 gefolgt von 2.6.2.3. Diese Variante ermöglicht es gerade mobilen Geräten, mit 2.6.2.3 direkt ohne Authentifizierung über einen erzeugten Hash-Wert auf die Bescheinigungen zuzugreifen.

2.6.2.1 Direkter Zugriff

Dieser Aufruf holt die Bescheinigungen direkt vom QIS-Server ab.

Aufruf (GET)	
`/certificate/live/<typ>`	
Aufruf-Parameter in der URL	
`<type>`	Art der Bescheinigung. Erlaubte Werte sind (Groß- und Kleinschreibung werden nicht unterschieden): ■ `DATA_CONTROL_SHEET`: Datenkontrollblatt ■ `CERTIFICATE_OF_MATRICULATION`: Immatrikulationsbescheinigung (deutsch) ■ `CERTIFICATE_OF_MATRICULATION_ENGLISH`: Immatrikulationsbescheinigung (englisch) ■ `BAFOEG`: BAFÖG-Bescheinigung ■ `KVV`: KVV-Bescheinigung ■ `DURATION_OF_STUDY`: Studienzeitbescheinigung ■ `IZ_ACCESS`: Initiale IZ-Zugangsdaten
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`BAD REQUEST` (400)	Im Typ wurde eine nicht vorhandene Art der Bescheinigung angegeben.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
`INTERNAL SERVER ERROR` (500)	Wenn die Kommunikation mit dem externen Server fehlschlägt oder die QIS-Seite nicht geparst werden kann.
Ergebnis (Body PDF-Dokument)	
Das Ergebnis ist ein PDF-Dokument mit der Bescheinigung.	

2.6.2.2 Erzeugen von Zugriffslinks

Dieser Aufruf liest alle Bescheinigungen eines Studierenden aus dem QIS und schreibt sie in einen internen Cache. Dort stehen die Bescheinigungen eine Stunde lang zum Abholen durch einen oder mehrere Aufrufe von 2.6.2.3 bereit.

Aufruf (GET)	
`/certificates/links`	
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`BAD REQUEST` (400)	Im Typ wurde eine nicht vorhandene Art der Bescheinigung angegeben.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.

INTERNAL SERVER ERROR (500)	Wenn die Kommunikation mit dem externen Server fehlschlägt oder die QIS-Seite nicht geparst werden kann.
Ergebnis (Body als JSON-Array mit Objekten)	
Die JSON-Objekte im Array haben den folgenden Aufbau:	
linkHashCode	Eindeutiger Zugriffs-Hashcode auf das Dokument, wird in Aufruf 2.6.2.3 benötigt.
certificateType	Art der Bescheinigung. Die möglichen Werte sind bereits als Aufrufparameter in 2.6.2.1 vorgestellt.
validUtil	Datum und Uhrzeit im Format yyyy-MM-dd'T'hh:mm:ssX beschreiben den Zeitpunkt, bis zu dem die entsprechende Bescheinigung im Cache gehalten wird.

2.6.2.3 Zugriff über die Links

Dieser Aufruf holt die Bescheinigungen aus dem Cache ab. Es ist keine Authentifizierung erforderlich.

Aufruf (GET) /certificate/cache/<hash>	
Aufruf-Parameter in der URL	
<hash>	Zugriffs-Hashcode aus dem Aufruf 2.6.2.2.
Ergebnis (HTTP-Status-Code)	
OK (200)	Der Aufruf war erfolgreich.
NOT FOUND (404)	Zu dem Hashcode existiert kein Dokument (mehr).
Ergebnis (Body PDF-Dokument) Das Ergebnis ist ein PDF-Dokument mit der Bescheinigung.	

2.7 Informationen der Fachschaft Informatik

Der Aufruf liefert die Informationen aus dem RSS-Feed der Informatik-Fachschaft.

Aufruf (GET) /studentcouncil/news	
Ergebnis (HTTP-Status-Code)	
OK (200)	Der Aufruf war erfolgreich.
SERVICE UNAVAILABE (503)	Der Fachschaftsserver ist zur Zeit nicht erreichbar.
Ergebnis (Body als JSON-Objekt)	
title	Titel des Feeds (zur Zeit „Fachschaft Informatik - Hochschule Karlsruhe news“).
link	URL der Übersichtsseite aller News.
description	Beschreibung des Feeds, zur Zeit immer leer.
date	Veröffentlichungsdatum der neuesten Nachricht im Format yyyy-MM-dd'T'hh:mm:ssX
messages	JSON-Array mit dem eigentliches Nachrichten-Objekten. Der Aufbau der Objekte des Arrays ist in der Folgetabelle beschrieben.

Die JSON-Objekte im Array `messages` haben den folgenden Aufbau:	
`title`	Kurzbeschreibung der Nachricht.
`link`	URL einer Webseite mit weiterführenden Informationen zur Nachricht.
`description`	Vollständige Nachricht.
`date`	Veröffentlichungsdatum der Nachricht im Format `yyyy-MM-dd HH:mm:ss`. Die Zeitzone ist immer GMT+1.

Der Aufruf `/studentcouncil/debugnews` liefert dagegen Nachrichten, die intern im REST-Server gespeichert sind. So kann das Abholen der Nachrichten auch ohne funktionierenden Fachschaftsserver getestet werden.

2.8 Offizielle Termine der Hochschule

Diese Schnittstelle liefert die offiziellen Semestertermine der Hochschule.

Aufruf (GET)	
`/officialcalendar/current`	
`/officialcalendar/next`	
Aufruf-Parameter in der URL	
`current`	Es werden die Termine aus dem aktuellen Semester ausgelesen.
`next`	Es werden die Termine aus dem folgenden Semester ausgelesen.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`NOT FOUND` (404)	Zum angegebenen Semester existieren keine Termine.
Ergebnis (Body als JSON-Array) Die JSON-Objekte im Array haben den folgenden Aufbau:	
`semesterId`	Interne ID des Semesters, aus dem die Termine stammen. Diese ID kann zur Zeit nicht sinnvoll im Client verwendet werden.
`semesterName`	Name des Semesters, Aufbau „SS 2015“ oder „WS 2015“ (also immer „SS“ oder „WS“, gefolgt von einem Leerzeichen und einer vierstelligen Jahreszahl des Jahres, in dem das Semester startete).
`entries`	Nach Datum sortiertes Array mit Objekten für die Termine. Der Aufbau der Objekte des Arrays ist in der Folgetabelle beschrieben.

Die Termine besitzen diesen Aufbau (im Attribut `entries`).	
`date`	Zeichenkette mit einer Freitextangabe des Datum.
`time`	Zeichenkette mit einer Freitextangabe der Uhrzeit.
`description`	Textuelle Beschreibung des Termins.

Die Datums- und Zeitwerte liegen nicht formatiert vor, weil sie von der Webseite der Hochschule eingelesen werden und dort nahezu beliebige Formulierungen verwendet werden.

2.9 Bibliothek

Diese Schnittstellen liefern Informationen wie Öffnungszeiten und freie Arbeitsplätze zu den Bibliotheken.

2.9.1 Namen

Diese Schnittstelle ermittelt die Kurz-Namen und vollständigen Bezeichnungen aller Bibliotheken.

Aufruf (GET)	
`/library/names`	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Immer.
Ergebnis (Body als JSON-Hashtanelle)	
Das Ergebnis ist eine JSON-Hashtabelle, in der der Schlüssel den Kurznamen der Bibliothek und der zugehörige Wert die Bezeichnung der Bibliothek enthält. Die Bezeichnung kann Zeilenumbrüche enthalten.	

2.9.2 Informationen

Diese Schnittstelle ermittelt Informationen zu einzelnen oder mehreren Bibliotheken. Die Attributnamen in den JSON-Objekten entsprechen teilweise nicht den Java-Code-Konventionen, weil hier die Original-Bezeichner des Bibliotheksservers übernommen wurden.

Aufruf (GET)	
`/library/info/<bib-namen>`	
Aufruf-Parameter in der URL	
`<bib-namen>`	Einer oder mehrere Kurznamen von Bibliotheken, zu denen Informationen ausgelesen werden sollen. Die Kurznamen werden als Schlüssel der Ergebnis-Hashtabelle in Abschnitt 2.9.1 zurückgegeben. Werden in diesem Aufruf mehrere Kurznamen angegeben, so sind diese mit Kommata (`%2C`) zu verbinden (Beispiel: `LST%2CFBH`).
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`NOT FOUND` (404)	Wenn in den Aufrufparametern Fehler vorhanden sind.
`BAD GATEWAY` (502)	Es traten Probleme bei der Kommunikation mit dem Bibliotheksserver auf.
Ergebnis (Body als JSON-Objekt)	
Das Ergebnis ist ein JSON-Objekt mit folgendem Aufbau.	
`seatestimate`	JSON-Hashtabelle mit aktueller Sitzplatzbelegung sowie Belegungen aus der Vergangenheit. Der Schlüssel ist dabei der Kurzname der Bibliothek, der Wert ein JSON-Array mit Belegungen. Die Objekte im Array werden in einer der folgenden Tabellen beschrieben.
`location`	JSON-Hashtabelle mit statischen Informationen zu den angeforderten Bibliotheken. Der Schlüssel ist der Kurzname der Bibliothek, der Wert ein Array mit Bibliotheksinformationen. Das Array hat immer die Länge 1.

Aufbau einer Sitzplatzbelegeung (Objekt im Werte-Array des Attributs `seatestimate`).

timestamp	JSON-Objekt mit einem Zeitstempel der folgenden Belegungsangabe. Das Zeitstempel-Objekt ist in einer der folgenden Tabellen beschrieben.
location_name	Kurzname der Bibliothek.
occupied_seats	Anzahl belegter Sitzplätze zu dem Zeitpunkt, der in timestamp definiert ist.
free_seats	Anzahl freier Sitzplätze zu dem Zeitpunkt, der in timestamp definiert ist.

Aufbau der statischen Bibliotheksinformationen (Objekt im Werte-Array des Attributs location).	
timestamp	JSON-Objekt mit einem festen Zeitstempel, dessen Bedeutung nicht erkennbar ist. Das Zeitstempel-Objekt ist in einer der folenden Tabellen beschrieben.
name	Kurzname der Bibliothek.
long_name	Vollstänige Bezeichnung der Bibliothek.
url	Link zur Bibliotheksseite.
building	Kürzel des Gebäudes, in dem sich die Bibliothek befindet. Das Kürzel entspricht dem Namen des Gebäudes aus dem Aufruf 3.6.
level	Etage im Gebäude, in der sich die Bibliothek befindet.
room	Raumnummer der Bibliothek, kann null sein.
geo_coordinates	GEO-Koordinaten des Gebäudes, in dem sich die Bibliothek befindet.
available_seats	Gesamtanzahl an Arbeitsplätzen in der Bibliothek (belegte und freie).
opening_hours	JSON-Objekt mit den Öffnungszeiten der Bibliothek an den einzelnen Wochentagen. Der Aufbau des Objektes ist in einer der folgenden Tabellen beschrieben.
super_location	Wenn die Bibliothek Teil einer anderen Bibliothek ist, dann enthält dieses Attribut den Kurznamen der übergeordneten Bibliothek. Teilweise werden synthetische Namen wie ALLBIBS für die Menge aller Bibliotheken verwendet.
sub_locations	Array mit den Namen der Bibliotheken, die Bestandteil dieser übergeordneten Bibliothek sind.

Aufbau eines Zeitstempels (Objekt in verschiedenen Attributen genutzt, z.B. seatestimate und location).	
date	Datum im Format yyyy-MM-dd HH:mm:ss bezogen auf die Zeitzone CET.
timezone_type	Art der Zeitzone, Bedeutung ist unbekannt.
timezone	Name der Zeitzone.

Aufbau einer Öffnungszeit (im Attribut opening_hours).	
base_timestamp	Zeitstempel im bekannten Format. Alle folgenden Datumsangaben beziehen sich auf diesen Zeitstempel. Um also den Wochentag einer Datumsangabe zu ermitteln, muss der Wert dieses Zeitstempels vom Zeitstempel der Angabe im folgenden Attribut weeklyOpeningHours abgezogen werden. Momentan starten alle Tagesangaben im Array weeklyOpeningHours immer montags. Ob das immer so sein wird, ist nicht bekannt.

`weekly_opening_hours`	Array der Länge 1, das wiederum ein Array enthält. Das innere Array besitzt die Zeitstempel mit den Öffnungs- und Schließzeiten der Bibliothek. Werte an ungeraden Indices enthalten die Öffnungszeit, gefolgt von der Schließzeit am darauf folgenden geraden Index. Auch sollte man sich nicht auf die Indizes verlassen, sondern die Zeitstempel auf Basis des `base_timestamp` auswerten.

3

REST-Schnittstelle (IWII)

Dieses Kapitel beschreibt die REST-Schnittstelle, die nur sinnvoll von Mitgliedern des Informatik-Fachbereichs bzw. von Studierenden aus diesem Fachbereich genutzt werden kann.

3.1 Studiengänge

Diese Aufrufen liefern verschiedene Informationen zu den im Intranet verwalteten Studiengängen.

3.1.1 Informationen

Der Aufruf liefert Informationen zu allen zur Zeit angebotenen Studiengängen, in denen noch Studierende immatrikuliert sind.

Aufruf (GET)	
`/courseofstudies/all`	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	In jedem Fall.
Ergebnis (Body als JSON-Array) Das Ergebnis ist ein Array mit JSON-Objekten des folgenden Aufbaus, wobei jeder Eintrag einen Studiengang beschreibt.	
`id`	Eindeutiger Kurzname des Studiengangs.
`currentExamination-RegulationsNumber`	Nummer der aktuellen Prüfungsordnung.
`idHead`	Eindeutige ID des Dozenten, der den Studiengang leitet. Die ID entspricht derjenigen, die z.B. beim Abholen der Dozentensprechzeiten (siehe 3.3.3) zurückgegeben wird.
`name`	Klartextname des Studiengangs.
`semesters`	Regelstudienzeit in Anzahl Semester.
`department`	Kürzel des Fachbereichs, zu dem der Studiengang gehört (z.B. „I“ für „Informatik“ und „WI“ für „Wirtschaftsinformatik“).
`faculty`	Kürzel der Fakultät, zu der der Studiengang gehört (z.B. „IWI“ für „Informatik und Wirtschaftsinformatik“).

basicStudyPeriod-Length	Länge des Grundstudiums in Anzahl Semester. Gibt es in dem Studiengang kein Grundstudium, dann ist der Wert -1.
creditPoints	Anzahl ECTS-Punkte, die in diesem Studiengang vergeben werden.
specializations	Beschreibt, ob der Studiengang Vertiefungsrichtungen besitzt. Hat specializations den Wert null, dann gibt es in diesem Studiengang keine Vertiefungsrichtungen. Werden dagegen in dem Studiengang Vertiefungen angeboten, dann ist specializations ein Array mit den Beschreibungen der Vertiefungsrichtungen

Vertiefungsrichtungen specializations	
id	Eindeutige ID der Vertiefungsrichtung.
courseOfStudies	Studiengang, zu der die Vertiefung gehört. Der Wert entspricht der ID des Studiengangs und ist daher hier in der Schnittstelle irrelevant.
name	Name der Vertiefungsrichtung.
examinationRegula-tionsNumber	Nummer der Prüfungsordnung im Studiengang, in der die Vertiefung angeboten wird. Diese Angabe ist vorhanden, weil nicht in jeder Prüfungsordnung eines Studiengangs eine Vertiefungsrichtung vorhanden sein muss.

3.1.2 Arbeitsbelastung je Semester

Der Aufruf liefert die Arbeitsbelastung (ECTS-Punkte sowie SWS) je Semester eines Studiengangs zurück. Die Arbeitsbelastung je Modul kann durch den Aufruf 3.1.3 ermittelt werden.

Aufruf (GET)	
/semesterworkload/<stg>/<po>/<vert>	
Aufruf-Parameter in der URL	
<stg>	ID eines Studiengangs (siehe Abschnitt 3.3.5)
<po>	Nummer der Prüfungsordnung bzw. 0 für die aktuelle.
<vert>	Nummer der Vertiefungsrichtung, 0 für „keine“. Die Vertiefungsrichtung wird erst für den Master-Studiengang ab dem Wintersemester 2014/2015 benötigt. Weitere Informationen zu den Vertiefungsrichtungen sind in Abschnitt 3.1 zu finden.
Ergebnis (HTTP-Status-Code)	
OK (200)	Im fehlerfreien Fall.
NOT FOUND (404)	Wenn es den übergebenen Studiengang nicht gibt.
INTERNAL SERVER ERROR (500)	Wenn ein Fehler in der Verarbeitung im Server auftritt.
Ergebnis (Body als JSON-Array)	
Das Ergebnis ist ein Array mit JSON-Objekten des folgenden Aufbaus, wobei jeder Eintrag die Arbeitsbelastung in einem Semester des Studiengang beschreibt.	
semester	Semester, für das die Arbeitsbelastung beschrieben wird. Der Eintrag entspricht immer dem um eins erhöhten Index, an dem sich der Eintrag im Array befindet.
contactHours	Anzahl an Semesterwochenstunden in dem Semester. Extern durchgeführte Veranstaltungen wie das Praxissemester oder die Abschlussarbeit werden mit 0 SWS gezählt.

creditPoints	Anzahl an ECTS-Punkten, die in diesem Semester vergeben werden. Nur dieser Wert beschreibt die Arbeitslast aus Studierendensicht.

3.1.3 Arbeitsbelastung je Modul

Der Aufruf liefert die Arbeitsbelastung (ECTS-Punkte sowie SWS) je Module eines Studiengangs zurück. Die Arbeitsbelastung je Semester kann durch den Aufruf 3.1.2 ermittelt werden.

Aufruf (GET)	
/moduleworkload/<stg>/<po>/<vert>	
Aufruf-Parameter in der URL	
<stg>	ID eines Studiengangs (siehe Abschnitt 3.3.5)
<po>	Nummer der Prüfungsordnung bzw. 0 für die aktuelle.
<vert>	Nummer der Vertiefungsrichtung, 0 für „keine“. Die Vertiefungsrichtung wird erst für den Master-Studiengang ab dem Wintersemester 2014/2015 benötigt. Weitere Informationen zu den Vertiefungsrichtungen sind in Abschnitt 3.1 zu finden.
Ergebnis (HTTP-Status-Code)	
OK (200)	Im fehlerfreien Fall.
NOT FOUND (404)	Wenn es den übergebenen Studiengang nicht gibt.
INTERNAL SERVER ERROR (500)	Wenn ein Fehler in der Verarbeitung im Server auftritt.
Ergebnis (Body als JSON-Array)	
Das Ergebnis ist ein Array mit einem Eintrag je Semester. Jeder Eintrag enthält ein Array aller Modulaufwände in diesem Semester. Jeder Modulaufwand ist durch ein JSON-Objekten des folgenden Aufbaus beschrieben.	
semester	Semester, für das die Arbeitsbelastung beschrieben wird. Der Eintrag entspricht immer dem um eins erhöhten Index, an dem sich der Eintrag im Haupt-Array befindet.
contactHours	Anzahl an Semesterwochenstunden in dem Semester. Extern durchgeführte Veranstaltungen wie das Praxissemester oder die Abschlussarbeit werden mit 0 SWS gezählt.
creditPoints	Anzahl an ECTS-Punkten, die in diesem Semester vergeben werden. Nur dieser Wert beschreibt die Arbeitslast aus Studierendensicht.
internalName	Interner Name des Moduls, entspricht dem Namen der Prüfung laut Studien- und Prüfungsordnung.

3.2 Anmeldepflichtige Arbeiten

Die REST-Schnittstelle erlaubt einen eingeschränkten Zugriff auf den Status angemeldeter Arbeiten sowie auf den Workflow, der sich hinter solchen Arbeiten verbirgt.

3.2.1 Liste aller Arbeiten

Es können alle Arbeiten eines Studiengangs ermittelt werden, deren Anmeldungen über das Intranet erfolgt.

Aufruf (GET)	
`/courseofstudies/allcourses/<stg>`	
Aufruf-Parameter in der URL	
`<stg>`	ID eines Studiengangs (siehe Abschnitt 3.3.5)
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`NOT FOUND` (404)	Wenn der angegebene Studiengang nicht existiert.
Ergebnis (Body als JSON-Array)	
Das Ergebnis ist ein Array mit JSON-Objekten des folgenden Aufbaus, wobei jeder Eintrag für eine anmeldepflichtige Veranstaltung steht.	
`id`	Eindeutige ID der Veranstaltung.
`type`	Eindeutiger Veranstaltungsname, Beispiel: `seminar_bachelor`
`group`	Name der Veranstaltungsgruppe, zu der die Veranstaltung gehört, kann in allen Aufrufen verwendet werden, die den Gruppennamen erwarten (z.B. in 3.2.3). Beispiel: `seminar`.
`idHead`	Eindeutige ID des Dozenten, der für diese Veranstaltung verantwortlich ist.
`name`	Lokalisierter Klartextname der Veranstaltung.
`shortName`	Kurzform des Klartextnames der Veranstaltung.

3.2.2 Liste aller Arbeiten mit REST-Anmeldung

Durch den Aufruf werden alle Arbeiten eines Studiengangs ermittelt, deren Anmeldungen über das Intranet mit Hilfe der REST-Schnittstelle erfolgen kann.

Aufruf (GET)	
`/courseofstudies/restcourses/<stg>`	
Aufruf-Parameter in der URL	
`<stg>`	ID eines Studiengangs (siehe Abschnitt 3.3.5)
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`NOT FOUND` (404)	Wenn der angegebene Studiengang nicht existiert.
Ergebnis (Body als JSON-Array)	
Das Ergebnis entspricht dem des Aufrufs aus Abschnitt 3.2.1.	

3.2.3 Workflow

Der Aufruf ermittelt eine Beschreibung des internen Workflows einer Arbeit. Er verlangt den Namen des Studiengangs sowie die Gruppe, zu der die Veranstaltung gehört.

Aufruf (GET)	
`/application/workflow/<stg>/<veranst-gruppe>/<no>`	
Aufruf-Parameter in der URL	
`<stg>`	ID eines Studiengangs (siehe Abschnitt 3.3.5)

`<veranst-gruppe>`	Gruppe, zu der die Veranstaltung gehört: ▪ `academic_writing`: Wissenschaftliches Schreiben ▪ `seminar`: Seminar ▪ `thesis`: Abschlussarbeit ▪ `student_research_project`: Projektarbeit im Bachelor-Studiengang ▪ `research_project_master`: Wissenschaftliches und projektbasiertes Arbeiten unter Anleitung im Master-Studiengang ▪ `team_teaching`: Team-Teaching, Wahlpflichtfach in den Bachelor-Studiengängen Informatik und Medien-und Kommunikationsinformatik ▪ `externship`: Praxissemester
`<no>`	Gibt es mehr als eine Veranstaltung einer Gruppe in einem Studiengang, dann gibt die Nummer deren Index an. Beispiel: Im Master-Studiengang wird im ersten und zweiten Semester jeweils eine Projektarbeit angeboten. Der Index unterscheidet zwischen beiden Arbeiten, indem die Indizes, mit 0 beginnend, in Semesterreihenfolge aufsteigend vergeben werden. Also würde die erste Projektarbeit den Index 0, die zweite den Index 1 erhalten. Gibt es nur eine Arbeit einer Gruppe in einem Studiengang, dann kann die Index-Nummer entfallen, und es wird automatisch 0 angenommen.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`NOT FOUND` (404)	Wenn in der URL Parameter nicht vorhandene Studienänge oder Veranstaltungsgruppen erscheinen.
Ergebnis (Body als JSON-Array) Das Ergebnis ist ein JSON-Objekt mit folgendem Aufbau.	
`workflow`	Eine Liste mit sechs Strings, die die sechs Phasen des Workflows der Arbeit beschreiben. Der genaue Aufbau des Strings ist unter diese Tabelle beschrieben.
`resetPoints-InWorkflow`	Das Array mit Indizes enthält die Positionen im Workflow `workflow`, an die bei einer Ablehnung zurückgeprungen wird. Beispiel: Befindet sich der Workflow `workflow` zur Zeit am Index 3 („Prüfung der hochgeladenen Dokumente durch einen Dozenten“) und lehnt der Dozent die Ausarbeitung ab, dann wird der Workflow an den nächstkleineren Index aus `resetPointsInWorkflow` zurückgesetzt.
`applicationHint`	Hinweise zur Anmeldung der Arbeit, reiner ASCII-Texte ohne Zeilenumbrüche und Formatanweisungen.
`uploadHint`	Hinweise zum Hochladen der Ergebnisse der Arbeit, reiner ASCII-Texte ohne Zeilenumbrüche und Formatanweisungen.
`noteHint`	Hinweise zur Notenfestsetzung, reiner ASCII-Texte ohne Zeilenumbrüche und Formatanweisungen.
`application-DeadlineHint`	Hinweise zu den Anmeldefristen, reiner ASCII-Texte ohne Zeilenumbrüche und Formatanweisungen.
`uploadDeadlineHint`	Hinweise zu den Fristen zum Hochladen der Zwischenstände und Ergebnisse, reiner ASCII-Texte ohne Zeilenumbrüche und Formatanweisungen.
`handOverDeadline-Hint`	Hinweise zu den Abgabefristen, reiner ASCII-Texte ohne Zeilenumbrüche und Formatanweisungen.

Die Strings mit der Beschreibung des Workflows in Attribut `workflow` haben folgenden Aufbau:

0. Anmeldephase durch den Studierenden: Der Student meldet sich an, wobei eventuell noch Vorbedingungen geprüft werden müssem. Dazu gehört beispielsweise, dass eine Anmeldung zum Seminar erst dann möglich ist, wenn vorher das „Wissenschaftliche Schreiben“ erfolgreich abgeschlossen wurde.
1. Genehmigungsphase durch Dozenten und Sekretariat. Stimmt hier eine der beteiligten Personen nicht zu, dann startet der Workflow wieder in Phase 0 (siehe auch Attribut `resetPointsInWorkflow`).
2. Hochladen von Ergebnissen durch den Studierenden bzw. durch das Sekretariat, wenn sich ein Studierender bei einer Abschlussarbeit bereits exmatrikuliert hat.
3. Prüfung der hochgeladenen Dokumente durch einen Dozenten. Stimmt der Dozent nicht zu, dann wird der Workflow in Phase 2 fortgesetzt (siehe auch Attribut `resetPointsInWorkflow`).
4. Noteneingabe durch Dozenten oder Sekretariat. Diese Phase wird wiederholt solange durchlaufen, bis alle beteiligten Personen der Note zugestimmt haben.
5. Notenansicht durch den Studierenden

Jeder String setzt sich aus einer Folge von Zeichen zusammen, wobei jedes Zeichen für die Rolle einer Person steht. Ist eine Phase bei einer Arbeit irrelevant, dann erhält der String für die Phase den Wert `null`.
Erläuterung der Rollenkürzel:

- `A`: Anmeldung durch einen Studierenden (**a**pplication)
- `P`: Erstgutachter (**p**rimary)
- `S`: Zeitgutachter (**s**econdary)
- `O`: Sekretariat (**o**ffice)
- `H`: Leiter einer Veranstaltung (**h**ead)
- `U`: Hochladen von Ergebnissen (**u**pload)
- `N`: Note kann vom Studierenden eingesehen werden (**n**ote)

Bei den Rollen `P`, `S`, `H` und `O` gibt es noch die Unterscheidung zwischen Groß- und Kleinbuchstaben:

- Großbuchstabe: Die Person mit der entsprechenden Rolle muss zustimmen.
- Kleinbuchstabe: Die Person mit der entsprechenden Rolle muss zur Kenntnis nehmen.

Diese Abbildung des Workflows stammt so fast direkt aus der Datenbank und ist ohne spezielle Tools ziemlich schnell erkennbar. Einzelne Strings im Workflow können leer sein, wenn Arbeiten diese Schritte nicht benötigen.

3.2.4 Workflowstatus

Diese Schnittstelle liefert Informationen zum aktuellen Status einer angemeldeten Arbeit.

Aufruf (GET)	
`/application/state/<veranst-gruppe>/<no>`	
Aufruf-Parameter in der URL	
`<veranst-gruppe>`	Gruppe, zu der die Veranstaltung gehört: ▪ `academic_writing`: Wissenschaftliches Schreiben ▪ `seminar`: Seminar ▪ `thesis`: Abschlussarbeit ▪ `student_research_project`: Projektarbeit im Bachelor-Studiengang ▪ `research_project_master`: Wissenschaftliches und projektbasiertes Arbeiten unter Anleitung im Master-Studiengang ▪ `team_teaching`: Team-Teaching, Wahlpflichtfach in den Bachelor-Studiengängen Informatik und Medien-und Kommunikationsinformatik ▪ `externship`: Praxissemester
`<no>`	Gibt es mehr als eine Veranstaltung einer Gruppe in einem Studiengang, dann gibt die Nummer deren Index an. Beispiel: Im Master-Studiengang wird im ersten und zweiten Semester jeweils eine Projektarbeit angeboten. Der Index unterscheidet zwischen den Arbeiten, indem die Indizes, mit 0 beginnend, in Semesterreihenfolge aufsteigend vergeben werden. Die erste Projektarbeit erhält den Index 0, die zweite den Index 1. Gibt es nur eine Arbeit einer Gruppe in einem Studiengang, dann kann die Index-Nummer entfallen, und es wird automatisch 0 angenommen.
Authentifizierung Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Der Studiengang wird anhand der Anmeldedaten des Aufrufers ermitteln. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen. Der Anwender kann nur die eigenen Anmeldeinformationen auslesen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`UNAUTHORIZED` (401)	Es wurden keine oder falsche Anmeldeinformationen übergeben.
`FORBIDDEN` (403)	Der Aufrufer ist kein Student. Weiterhin tritt der Fehler auf, wenn der aufrufende Student die Vorbedingungen nicht erfüllt. So ist beispielsweise für das Seminar im Bachelor-Studiengang der Besuch der Veranstaltung „Wissenschaftliches Schreiben“ Voraussetzung.
`NOT FOUND` (404)	Die Veranstaltungsgruppe existiert nicht oder wird nicht im Studiengang des Aufrufers angeboten.
Ergebnis (Body als JSON-Array) Das Ergebnis ist ein JSON-Objekt mit folgendem Aufbau.	
`topic`	Thema der Arbeit. Der Wert kann `null` sein, wenn bei der Anmeldung kein Thema eingetragen wird (z.B. Praxissemester).
`attempt`	Zähler für die Anzahl Versuche, um diese Prüfung abzulegen. Die Zählung beginnt bei `1`.
`lecturer`	Name des Dozenten, der die Arbeit betreut.
`idLecturer`	ID des Dozenten, der die Arbeit betreut.

`secondExaminer`	Name des Korreferenten der Arbeit. Der Eintrag kann `null` sein, wenn die Arbeit nur von einem Dozenten betreut wird oder noch kein Korreferent zugewiesen ist. Zur Zeit besitzt nur die Thesis einen zweiten Dozenten.
`idSecondExaminer`	ID des Korreferenten der Arbeit. Der Eintrag kann `-1` sein, wenn die Arbeit nur von einem Dozenten betreut wird oder noch kein Korreferent zugewiesen ist.
`dateOfIssue`	Anmeldedatum der Arbeit.
`deadline`	Spätestes Datum, an dem die Arbeit abgegeben werden muss.
`overtime`	Genehmigte Verlängerung der Abgabefrist in Tagen (wird zur Zeit nur bei der Thesis verwendet). Ist keine Verlängerung genehmigt, oder wird bei bestimmten Veranstaltungstypen ohne Verlängerung gearbeitet, dann ist der Wert `0`.
`dateOfHandOver`	Datum, an dem die Arbeit abgegeben wurde. Läuft die Arbeit noch, dann ist der Wert `null`.
`resultFileUpload-State`	Bei allen Arbeiten müssen die Ergebnisse hochgeladen und vom Betreuer akzeptiert werden. Dieser Eintrag zeigt den Akzeptanzzustand dieser Dokumente an: ▪ `-`: Es wurde noch kein Dokument hochgeladen. ▪ `D`: Der Betreuer hat die Dokumente abgelehnt. ▪ `A`: Der Betreuer hat die Dokumente akzeptiert. ▪ `O`: Der Betreuer hat die hochgeladenen Dokumente noch nicht beurteilt.
`workFlowState`	Der Eintrag beschreibt den aktuellen Stand des Workflows der Arbeit als String, bestehend aus zwei Bestandteilen. Der genaue Aufbau des Strings ist unter diese Tabelle beschrieben.

Der Aufbau des Strings mit der Beschreibung des Zustands des aktuellenen Workflows in Attribut `workFlowState` bezieht sich auf den Workflow aus Abschnitt 3.2.3. Beispiel: Zustand „`1:OO`“ bei einer Workflowdefinition von „`POHo`“
Die Anmmeldung befindet sich im Workflow in der Phase `1`, das Sekretariat muss zustimmen. Da der Eintrag einer Personenrolle unter Umständen mehrfach im Workflow vorkommen kann, gibt der folgende Index `0` an, dass es sich um das erste Vorkommen des Sekretariats im Anmeldeworkflow handelt. Die Benutzerrollen sind bereits in Abschnitt 3.2.3 vorgestellt. Der Zustand beginnt immer mit der Nummer der Phase, gefolgt von einem Doppelpunkt. Nach dem Doppelpunkt folgt der Zustand des Worflows innerhalb der Phase. Genereller Aufbau des String, bezogen auf die sechs Workflow-Phasen:

0. Anmeldephase: Befindet sich der Workflow in dieser Phase, dann hat sich der Student noch nie für diese Arbeit angemeldet. Der String hat den festen Aufbau „`0:-`“. Sollte jemand bei der Arbeit durchgefallen sein und möchte sich erneut anmelden, dann wird als Status die Note des missglückten Versuchs (Phase 5) angezeigt. Sollte der vorherige Anmeldeversuch dagegen abgelehnt worden sein, dann befindet sich der Workflow in der Phase `1` mit dem Hinweis auf den Ablehnungsgrund. Eine Neuanmeldung ist auch dann möglich, wenn der Ablehnungsgrund behoben ist.
1. Genehmigungsphase: Der String kann folgenden Aufbau besitzen:
   - „`1:`*`RolleIndex`*“: Als nächstes muss die Person mit der Rolle *`Rolle`* der Anmeldung zustimmen. Weil der Eintrag einer Personenrolle unter Umständen mehrfach im Workflow vorkommen kann (siehe Einleitung oben), gibt der fol-

gende Index *Index* an, das wievielte Vorkommen der Personenrolle im Anmeldeworkflow gemeint ist.
Zwei Beispiele für die Workflowdefinition „POHo“:
„1:O0“ verweist auf das erste, große O im Workflow, was bedeutet, dass das Sekretariat zustimmen muss.
„1:o1“ verweist auf das zweite, kleine o im Workflow. Hier muss das Sekretariat zur Kenntniss nehmen.

- „1:D*Grund*“: Die Anmeldung wurde mit dem Grund *Grund* abgelehnt. Mögliche Gründe sind:
  0. Die Abmeldung erfolgt auf Wunsch des Studierenden.
  1. Das Thema war nicht im Vorfeld abgestimmt worden.
  2. Die Anmeldefrist war bereits verstrichen.
  3. Das Thema ist nicht akzeptabel.
  4. Die Aufgabenbeschreibung ist nicht akzeptabel oder ausreichend.
  5. Eine Vorbedingung der Prüfungsordnung ist nicht erfüllt.
  6. Die Firma betreut schon zu viele Arbeiten oder ist nicht geeignet.
  7. Es ist eine Rücksprache mit dem Betreuer erforderlich.
  8. Es fehlen noch Unterlagen bzw. die vorgelegten sind lückenhaft.

  Beispiel: „1:D2“, die Arbeit wurde wegen einer bereits abgelaufenen Anmeldefirst abgelehnt. Es ist eine Neuammeldung möglich.

2. Hochladen von Ergebnissen: Der String hat den festen Aufbau „2:-“.
3. Prüfung der hochgeladenen Dokumente: Mögliche Stringinhalte sind:
   - „3:O“: Die Dokumente wurden noch nicht geprüft (**o**pen).
   - „3:D“: Die Dokumente wurden abgelehnt (**d**enied).

   Wurden hochgeladene Dokumente akzeptiert, dann ist das dadurch erkennbar, dass sich der Workflow im Zustand 4 oder 5 befindet.
4. Noteneingabe: Hochgeladenen Dokumenten wurden akzeptiert, die Note aber noch nicht eingetragen. Mögliche Zustände:
   - „4:-“: Die Note ist noch nicht eingetragen, der Workflow findet außerhalb des Programms statt. Das ist beispielsweise bei der Thesis der Fall. Hier gibt es bei der Abgabe noch eine mündliche Prüfung, bei der die Gesamtnote in einem Protokoll erfasst und erst bei Beantragung der Abschlussnote vom Sekretariat eingetragen wird.
   - „4:*RolleIndex*“: Die Person mit der Rolle *Rolle* an Position *Index* im Workflow muss der Note zustimmen.
5. Notenansicht: Die Note ist eingetragen und von den beteiligten Personen akzeptiert. Der String hat den festen Aufbau „5:*HIS-Note*“. Die Note wird im internen Format von HIS abgelegt. Für benotete Prüfung entspricht das dem Wert der Note, mit 100 multipliziert (170 für die Note 1,7). Bei unbenoteten Prüfungen steht die 0 für eine bestandene, die Note 99 für eine nicht bestandene Prüfung.

3.2.5 Eigenschaften

Diese Schnittstelle liefert alle Dozenten, die eine bestimmte Arbeit betreuen dürfen. Weiterhin enthält das Ergebnis zusätzliche Anmeldeinformationen.

Aufruf (GET)	
`/application/properties/<stg>/<veranst-gruppe>/<no>`	
Aufruf-Parameter in der URL	
`<stg>`	ID eines Studiengangs (siehe Abschnitt 3.3.5)
`<veranst-gruppe>`	Veranstaltungsgruppe (siehe Abschnitt 3.2.3)
`<no>`	Index der Veranstaltung (siehe Abschnitt 3.2.3)
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`NOT FOUND` (404)	Wenn der angegebene Studiengang oder die Veranstaltungsgruppe nicht existiert.
Ergebnis (Body als JSON-Array)	
Das Ergebnis ist ein JSON-Objekt mit folgendem Aufbau.	
`lecturers`	Es handelt sich um ein Array mit Dozenten-Objekten, wie sie in Abschnitt 3.3.3 beschrieben werden. Alle Dozenten in diesem Array dürfen als Betreuer für die Veranstaltung `<veranst-gruppe>` ausgewählt werden.
`topic`	`topic`: Ist der Wert `true`, dann muss bei der Anmeldung in Abschnitt 3.2.9 ein Thema übergeben werden. Ist der Wert `false`, dann sollte beim Anwender kein Thema abgefragt werden, da es ignoriert wird.

3.2.6 Anleitung

Der Aufruf lädt ein XHTML-Fragment mit einer Anleitung zu einer Arbeit herunter. Für einige Arbeiten existieren weiterführende Informationen im PDF-Format (siehe Abschnitt 3.2.8).

Aufruf (GET)	
`/application/hint/<stg>/<veranst-gruppe>/<no>`	
Aufruf-Parameter in der URL	
`<stg>`	ID eines Studiengangs (siehe Abschnitt 3.3.5)
`<veranst-gruppe>`	Veranstaltungsgruppe (siehe Abschnitt 3.2.3)
`<no>`	Index der Veranstaltung (siehe Abschnitt 3.2.3)
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`NOT FOUND` (404)	Wenn der angegebene Studiengang oder die Veranstaltungsgruppe nicht existiert.
Ergebnis (Body als XHTML-Fragment)	
Das Ergebnis ist ein XHTML-Fragment, in dem nur einige Tags vorkommen können. Die Tags enthalten keine weiteren Attribute, das Fragment besitzt weder einen XHTML-Kopf noch -Rumpf. Folgende Tags können vorhanden sein:	
`h4`	für Überschriften
`ol`	für nummerierte Aufzählungen

ul	für nicht nummerierte Aufzählungen
li	für einzelne Aufzählungspunkte
a	für Links. Dabei kann das target-Attribut mit dem Wert _blank vorhanden sein.

3.2.7 Alle Anleitungsdokumente

Der Aufruf lädt ein PDF-Dokument mit allen Anleitungen zu einer Arbeit herunter. Nicht für alle Arbeiten sind jedoch Anleitungen auf dem dem Server hinterlegt.

Aufruf (GET)	
/application/hintdocument/<stg>/<veranst-gruppe>/<no>	
Aufruf-Parameter in der URL	
<stg>	ID eines Studiengangs (siehe Abschnitt 3.3.5)
<veranst-gruppe>	Veranstaltungsgruppe (siehe Abschnitt 3.2.3)
<no>	Index der Veranstaltung (siehe Abschnitt 3.2.3)
Ergebnis (HTTP-Status-Code)	
OK (200)	Im Erfolgsfall.
NOT FOUND (404)	Wenn der angegebene Studiengang oder die Veranstaltungsgruppe nicht existiert bzw. kein Dokument hochgeladen wurde.
INTERNAL SERVER ERROR (500)	Wenn bei der internen Verarbeitung des Dokuments ein Fehler auftrat.
Ergebnis (Body als PDF-Dokument)	
Die PDF-Dokumente sind nicht durch Passwörter geschützt.	

3.2.8 Einzelnes Anleitungsdokument

Der Aufruf lädt ein PDF-Dokument mit einer Anleitung zu einer Arbeit herunter. Nicht für alle Arbeiten sind jedoch Anleitungen auf dem dem Server hinterlegt.

Aufruf (GET)	
/application/hintdocument/<stg>/<veranst-gruppe>/<no>/<dateiname>/	
Aufruf-Parameter in der URL	
<stg>	ID eines Studiengangs (siehe Abschnitt 3.3.5)
<veranst-gruppe>	Veranstaltungsgruppe (siehe Abschnitt 3.2.3)
<no>	Index der Veranstaltung (siehe Abschnitt 3.2.3)
<dateiname>	Name der PDF-Datei
Ergebnis (HTTP-Status-Code)	
OK (200)	Im Erfolgsfall.
INTERNAL SERVER ERROR (500)	Wenn bei der internen Verarbeitung des Dokuments ein Fehler auftrat.
Ergebnis (Body als PDF-Dokument)	
Die PDF-Dokumente sind nicht durch Passwörter geschützt. Existiert das Dokument nicht, dann ist das Ergebnis null.	

Dieser Aufruf ist eigentlich nur zur internen Verwendung durch die Info-Seiten gedacht, weil es zur Zeit keine Möglichkeit gibt, die Namen der Dateien über die REST-Schnittstelle zu ermitteln. Der abschließende `/` in der URL ist wichtig und darf nicht weggelassen werden.

3.2.9 Anmeldung

Zu allen Arbeiten, die kein Hochladen von PDF-Dokumenten verlangen, kann die Anmeldung über die REST-Schnittstelle erfolgen. Solche Arbeiten lassen sich über den Aufruf, der in Abschnitt 3.2.2 beschrieben ist, dynamisch ermitteln. Die Anmeldung erfolgt in drei Schritten, die in genau der folgenden Reihenfolge durchlaufen werden.

3.2.9.1 1. Schritt: Liste der Veranstaltungen

Im ersten Schritt wird die Liste der Veranstaltungen ausgelesen, in der sich der Studierende überhaupt anmelden darf.
Diese Schnittstelle liefert Informationen zum aktuellen Status einer angemeldeten Arbeit.

Aufruf (GET) `/application/applicablecourses`	
Authentifizierung Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Der Studiengang wird anhand der Anmeldedaten des Aufrufers ermitteln. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`UNAUTHORIZED` (401)	Es wurden keine oder falsche Anmeldeinformationen übergeben.
`FORBIDDEN` (403)	Der Aufrufer ist kein Student.
`INTERNAL SERVER ERROR` (500)	Wenn bei der internen Verarbeitung ein Fehler auftrat.
Ergebnis (Body als JSON-Array) Das Ergebnis ist ein Array mit JSON-Objekten, von denen jedes Objekt für eine Veranstaltung steht. Das Array ist leer, wenn sich der Aufrufer an keiner weiteren Veranstaltung anmelden kann. Die Objekte im Array haben den folgenden Aufbau.	
`type`	Eindeutiger Veranstaltungsname, Beispiel: `seminar_bachelor`.
`localizedType`	Lokalisierter Klartextname der Veranstaltung, lokalisiert anhand der eingestellten Sprache im Aufruf.
`group`	Name der Veranstaltungsgruppe, zu der die Veranstaltung gehört, kann in allen Aufrufen verwendet werden, die den Gruppennamen erwarten (z.B. in 3.2.3). Beispiel: `seminar`.
`localizedGroup`	Lokalisierter Klartextname der Veranstaltungsgruppe, zu der die Veranstaltung gehört, lokalisiert anhand der eingestellten Sprache im Aufruf.
`templateURL`	URL, um in Schritt 2 der Anmeldung das Template mit den erforderlichen Daten auszulesen.
`templateMethod`	HTTP-Request-Methode für den Aufruf im Schritt 2 der Anmeldung (immer `GET`).

3.2.9.2 2. Schritt: Template für die Anmeldung

Im zweiten Schritt werden die zur Anmeldung erforderlichen Informationen ausgelesen.

Aufruf (GET)	
`/application/applicationtemplate/<veranst-gruppe>/<no>`	
Aufruf-Parameter in der URL	
`<veranst-gruppe>`	Veranstaltungsgruppe (siehe Abschnitt 3.2.3)
`<no>`	Index der Veranstaltung (siehe Abschnitt 3.2.3)
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Der Studiengang wird anhand der Anmeldedaten des Aufrufers ermitteln. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`UNAUTHORIZED` (401)	Es wurden keine oder falsche Anmeldeinformationen übergeben.
`FORBIDDEN` (403)	Der Aufrufer ist kein Student. Der Fehler tritt auch auf, wenn sich der Aufrufer an der Veranstaltung nicht anmelden darf, weil z.B. nicht alle Vorbedingungen erfüllt sind.
`NOT FOUND` (404)	Es gibt die angegebene Veranstaltung gar nicht.
Ergebnis (Body als JSON-Array)	
Das Ergebnis ist ein JSON-Objekt mit den Informationen, die zur Anmeldung erforderlich sind.	
`propertyInfos`	Array mit JSON-Objekten, von denen jedes Objekt eine Information beschreibt, die zur Anmeldung erforderlich ist. Aufbau: ▪ `type`: Typ der Information. Zur Zeit gibt es `String`, `int` und `id` als Hinweis darauf, dass diese Zahl die ID eines Objektes ist. ▪ `name`: Name der Information, nicht lokalisiert. ▪ `localizedName`: Lokalisierter Name der Information, die Sprache wird anhand der Sprache im Aufruf ermittelt. ▪ `constraints`: Werteinschränkungen der Information. Die Einschränken werden durch Kommata voneinander getrennt und sind in der Tabelle nach dieser Aufzählung beschrieben.
`localizedType`	Lokalisierter Klartextname der Veranstaltung, lokalisiert anhand der eingestellten Sprache im Aufruf.
`validLecturers`	Array mit JSON-Objekten der Dozenten, die diese Veranstaltung anbieten und bei denen sich der Studierende anmelden darf. Dieses Array ist leer, wenn zur Anmeldung kein Dozent angegeben werden muss. Aufbau der Objekte: ▪ `id`: ID des Dozenten. ▪ `name`: Name des Dozenten.
`applicationURL`	URL, um in Schritt 3 die Anmeldung durchzuführen.
`applicationMethod`	HTTP-Request-Methode für den Aufruf im Schritt 3 der Anmeldung (immer `POST`).

Zur Zeit vorhandene Property-Info-Daten (Attribut `type` des Property-Info-Objektes)	
`telephoneNumber`	Telefonnummer des Studenten.
`matriculationNumber`	Matrikelnummer des Studenten.

index	Nummer der Veranstaltung innerhalb derselben Gruppe (normalerweise 0, bei mehreren Veranstaltungen desselben Typs fortlaufend nummeriert)
semester	Studiensemester des Studenten.
topic	Thema der Arbeit.
lecturerPrimary	Betreuender Dozent der Arbeit.

Werte- und Bereichseinschränkungen der erforderlichen Informationen (Attribut constraints des Property-Info-Objektes). Mehrere Einschränkungen je Information werden durch Kommata voneinander getrennt. Anhand der Information ist zur Zeit nicht erkennbar, auf welchen Typ sich eine ID bezieht. Momentan wird die ID nur für die Dozenten eingesetzt. Sie bezieht sie sich auf das Attribut id eines Objektes im Array validLecturers.	
String	minlength(*laenge*): Der String muss die Mindestlänge laenge besitzen.
int	min(*wert*): Die int-Zahl muss den Mindestwert wert besitzen.
int	optional: Die int-Zahl darf leer (null) sein.
id	min(*wert*): Die ID muss den Mindestwert wert besitzen.
id	optional: Die ID darf leer (null) sein.

3.2.9.3 3. Schritt: Anmeldung

Im dritten Schritt wird die eigentliche Anmeldung durchgeführt.

Aufruf (POST) /application/	
Aufruf-Parameter als JSON-Objekt (Body)	
telephoneNumber	Telefonnummer des Studenten als Freitext-String mit einer Mindestlänge von fünf Zeichen.
matriculationNumber	Matrikelnummer des Studenten als int-Zahl.
course	Zeichenkette mit dem Namen der Veranstaltungsgruppe, zu der sich der Student anmelden möchte (z.B. seminar).
index	Gibt es mehrere Veranstaltungen einer Gruppe (eines Typs) in einem Studiengang, dann werden sie fortlaufend, mit 0 beginnend, durchnummeriert. Das ist bisher nur für Projektarbeiten im Master-Studiengang erforderlich. Ansonsten ist index immer 0. Der Wert ist eine int-Zahl.
semester	Aktuelles Studiensemester des Studenten als int-Zahl.
topic	Zeichenkette mit dem Thema der Arbeit. Dieses Feld wird für einige Arbeiten wie team_teaching und academic_writing nicht ausgefüllt. Enthält es trotzdem Daten, so werden diese ignoriert. Durch den in Abschnitt 3.2.5 vorgestellten Aufruf lässt sich vorab herausfinden, ob das Thema erforderlich ist.
idLecturer	int-Zahl mit der eindeutigen ID des Betreuers der Arbeit. Alle zulässigen Betreuer werden im Schritt 2 der Anmeldeprozedur zurückgegeben.
debugMode	Hat debugMode den Wert true, dann wird keine echte Anmeldung durchgeführt. Stattdessen werden lediglich die Aufrufparameter geprüft.
Authentifizierung	

Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Der Studiengang wird anhand der Anmeldedaten des Aufrufers ermitteln. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`CREATED` (201)	Im Erfolgsfall.
`BAD REQUEST` (400)	Es sind Übergabeparameter im JSON-Objekt falsch.
`UNAUTHORIZED` (401)	Es wurden keine oder falsche Anmeldeinformationen übergeben.
`FORBIDDEN` (403)	Der Aufrufer ist kein Student. Der Fehler tritt auch auf, wenn sich der Aufrufer an der Veranstaltung nicht anmelden darf, weil z.B. nicht alle Vorbedingungen erfüllt sind.
`NOT FOUND` (404)	Es gibt die angegebene Veranstaltung gar nicht.
Ergebnis-String (Body) Das Ergebnis ist im Fehlerfall eine nähere Erläuterung des Fehlers in Form einer Zeichenkette. Im Erfolgfalls ist der Rumpf leer.	
`invalid course`	Den Veranstaltungstyp gibt es nicht.
`invalid matriculationnumber`	Die Matrikelnummer kann nicht stimmen.
`invalid telephonnumber`	Die Telefonnummer ist leer oder zu kurz.
`invalid semester`	Das Studiensemester liegt außerhalb des gültigen Bereichs.
`invalid topic`	Das Thema der Arbeit ist leer.
`invalid lecturer`	Es gibt den Dozenten nicht bzw. der Dozent darf diese Art von Arbeit nicht betreuen.
`application not allowed`	Die Anmeldung ist zur Zeit nicht zulässig (z.B. weil diese Arbeit bereits angemeldet ist).
`rest application not allowed`	Die Anmeldung darf nicht über die REST-Schnittstelle erfolgen (z.B. weil Dateien hochgeladen werden müssen).
`precondition not fulfilled`	Es sind nicht alle Vorbedingungen für diese Art der Arbeit erfüllt.

3.3 Nachrichten

Die Nachrichten werden vom Sekretariat oder von einzelnen Dozenten eingepflegt. Sie erscheinen auf den Informationsseiten der Fakultät, sind aber auch über die REST-Schnittstelle auslesbar.

3.3.1 Aktuelles

Die Liste aller Nachrichten unter „Schwarzes Brett“ eines Studiengangs oder aller Studiengänge wird über diesen Aufruf ermittelt.

Aufruf (GET)
`/newsbulletinboard/<stg>`
Aufruf-Parameter in der URL

<stg> (optional)	ID eines Studiengangs (siehe Abschnitt 3.3.5), um Nachrichten dieses Studiengangs auszulesen. Wird die ID weggelassen, dann werden die Nachrichten aller Studiengänge ausgelesen.
Ergebnis (HTTP-Status-Code)	
OK (200)	Im Erfolgsfall.
NOT FOUND (404)	Wenn in der URL eine ungültige Studiengangs-ID aufgeführt ist.
Ergebnis (Body als JSON-Objekt) Das Ergebnis ist ein Array von JSON-Objekten mit dem folgenden Aufbau.	
id	eindeutige ID der Meldung
title	Hauptüberschrift der Nachricht
subTitle	Kurzzusammenfassung der Nachricht
courseOfStudies	Liste mit einer Einschränkung auf die Studiengänge, für die die Nachricht relevant ist. Ist der Wert null, dann betrifft die Nachricht alle Studiengänge (keine Einschränkung). Die Namen der Studiengängen entsprechen denen aus dem Aufruf.
content	eigentlicher Nachrichtentext in reinem ASCII-Format mit manuellen Zeilenumbrüchen
links	Links zu weiterführenden Informationen, durch Kommata getrennt. Der Wert null zeigt an, dass es keine Links gibt.
type	Art der Nachricht: ▪ announcement: allgemeine Ankündigung ▪ rearrangement: zeitliche Verlegung einer Veranstaltung ▪ cancellation: Absage einer Veranstaltung ▪ room_change: Raumänderung einer Veranstaltung In Zukunft können werden Typen hinzukommen.

3.3.1.1 Atom-Feed

Die Liste der Nachrichten kann auch als Atom-Feed ausgelesen werden.

Aufruf (GET) /atomfeed/newsbulletinboard/<stg>
Aufruf-Parameter in der URL
<stg> (optional) — ID eines Studiengangs (siehe Abschnitt 3.3.5), um Nachrichten dieses Studiengangs auszulesen. Wird die ID weggelassen, dann werden die Nachrichten aller Studiengänge ausgelesen.
Ergebnis (HTTP-Status-Code)
OK (200) — Immer, bei einer ungültigen Studiengangs-ID werden die Nachrichten aller Studiengänge ausgelesen.
Ergebnis im Atom-Feed-Format im Body der Antwort Es werden dieselben Attribute wie beim REST-Aufruf „Aktuelles“ 3.3.1 zurückgegeben.

3.3.1.2 RSS-Feed

Die Schnittstelle unterstützt neben Atom- auch RSS-Feeds.

Aufruf (GET) /rssfeed/newsbulletinboard/<stg>

Aufruf-Parameter in der URL	
`<stg>` (optional)	ID eines Studiengangs (siehe Abschnitt 3.3.5), um Nachrichten dieses Studiengangs auszulesen. Wird die ID weggelassen, dann werden die Nachrichten aller Studiengänge ausgelesen.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Immer, bei einer ungültigen Studiengangs-ID werden die Nachrichten aller Studiengänge ausgelesen.
Ergebnis im RSS-Feed-Format im Body der Antwort	
Es werden dieselben Attribute wie beim REST-Aufruf „Aktuelles“ 3.3.1 zurückgegeben.	

3.3.2 Blockveranstaltungen

Diese Schnittstelle liefert Informationen zu den Blockveranstaltungen.

Aufruf (GET)	
`/blockcourses/<stg>`	
Aufruf-Parameter in der URL	
`<stg>` (optional)	ID eines Studiengangs (siehe Abschnitt 3.3.5), um die Blockveranstaltungen dieses Studiengangs auszulesen. Wird die ID weggelassen, dann werden die Blockveranstaltungen aller Studiengänge ausgelesen.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`NOT FOUND` (404)	Bei Angabe einer ungültigen Studiengangs-ID.
Ergebnis (Body als JSON-Array)	
Das Ergebnis ist ein Array mit JSON-Objekten des folgenden Aufbaus, wobei jeder Eintrag eine Blockveranstaltung beschreibt.	
`id`	Eindeutige ID der Blockveranstaltung.
`courseOfStudies`	Array mit einer Einschränkung auf die Studiengänge, in denen die Blockveranstaltung angeboten wird. Ist der Wert `null`, dann wird die Blockveranstaltung in allen Studiengänge angeboten (keine Einschränkung). Die Namen der Studiengänge entsprechen denen aus dem Aufruf.
`lectureName`	Name der Blockveranstaltung
`lecturerNames`	Dozent oder Dozenten, die die Veranstaltung leiten
`dates`	Frei formatierbare Angabe der Tage mit Kommentaren, an denen die Veranstaltungen stattfinden. Der Text kann durch Zeilenumbrüche und Leerzeichen formatiert sein.
`times`	Uhrzeiten inklusive Kommentare, an denen die Veranstaltungen stattfinden. Der Text kann durch Zeilenumbrüche und Leerzeichen formatiert sein. Eine logische Zuordnung zu den Tagen ist nicht vorhanden. In der Regel formatieren die Sekretärinen `dates` und `times` aber so, dass die Zeile `x` aus `dates` derselben Zeilennummer aus `times` entspricht.
`buildingAndRoom`	Eine oder mehrere Raumangaben, die in der Regel durch Zeilenumbrüche getrennt werden. Auch hier ist eine logische Zuordnung zu den Tagen nicht explizit vorhanden. In der Regel formatieren aber die Sekretärinen `dates` und `buildingAndRoom` so, dass die Zeile `x` aus `dates` derselben Zeilennummer aus `buildingAndRoom` entspricht.

3.3.3 Sprechzeiten aller Dozenten

Der Aufruf liest alle Dozentensprechzeiten aus.

Aufruf (GET)	
`/lecturersconsultationhours/`	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Immer.
Ergebnis (Body als JSON-Array)	
Das Ergebnis ist ein alphabetisch nach dem Nachnamen des Dozenten sortiertes Array von JSON-Objekten mit folgendem Aufbau, wobei jeder Eintrag eine Dozentensprechstunde beschreibt.	
`id`	Eindeutige ID der Sprechstunde.
`firstname`	Vorname des Dozenten.
`lastname`	Nachname des Dozenten.
`title`	„Prof.“ und akademischer Grad.
`adsAccount`	ADS-Account des Dozenten.
`mailAddress`	Mailadresse des Dozenten, setzt sich immer aus dem ADS-Account und `@hs-karlsruhe.de` zusammen.
`consultationDay`	Wochentag, an dem die Sprechstunde stattfindet. Die Zählung beginnt bei `0` für den Montag.
`consultationTime`	Sprechzeit sowie eventuell Kommentare dazu.
`room`	Raum, in dem die Sprechstunde abgehalten wird.
`building`	Gebäude, in dem der Raum liegt.
`department`	Kürzel des Fachbereichs, zu dem der Dozent gehört (z.B. „I“ für „Informatik“ und „WI“ für „Wirtschaftsinformatik“).
`faculty`	Kürzel der Fakultät, zu der der Dozent gehört (z.B. „IWI“ für „Informatik und Wirtschaftsinformatik“). Mitarbeiter des Informationszentrums werden der Pseudo-Fakultät „IZ“ zugeordnet.

3.3.4 Sprechzeiten eines Dozenten

Es kann auch die Sprechzeit eines einzelnen Dozenten ausgelesen werden.

Aufruf (GET)	
`/lecturersconsultationhours/<lecturer>`	
Aufruf-Parameter in der URL	
`<lecturer>`	ADS-Account eines Dozenten.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgfall.
`NOT FOUND` (404)	Wenn der angegebene Dozent nicht existiert.
Ergebnis (Body als JSON-Array)	
Das Ergebnis ist ein JSON-Objekt mit folgendem Aufbau, das eine Dozentensprechstunde beschreibt. Das Objekt ist dasselbe wie in Abschnitt 3.3.3.	
`id`	Eindeutige ID der Sprechstunde.
`firstname`	Vorname des Dozenten.
`lastname`	Nachname des Dozenten.

`title`	„Prof.“ und akademischer Grad.
`adsAccount`	ADS-Account des Dozenten.
`mailAddress`	Mailadresse des Dozenten, setzt sich immer aus dem ADS-Account und `@hs-karlsruhe.de` zusammen.
`consultationDay`	Wochentag, an dem die Sprechstunde stattfindet. Die Zählung beginnt bei `0` für den Montag.
`consultationTime`	Sprechzeit sowie eventuell Kommentare dazu.
`room`	Raum, in dem die Sprechstunde abgehalten wird.
`building`	Gebäude, in dem der Raum liegt.
`department`	Kürzel des Fachbereichs, zu dem der Dozent gehört (z.B. „I“ für „Informatik“ und „WI“ für „Wirtschaftsinformatik“).
`faculty`	Kürzel der Fakultät, zu der der Dozent gehört (z.B. „IWI“ für „Informatik und Wirtschaftsinformatik“). Mitarbeiter des Informationszentrums werden der Pseudo-Fakultät „IZ“ zugeordnet.

3.3.5 Tutorien

Über diesen Aufruf kann die Liste aller Tutorien eines Studiengangs gelesen werden.

Aufruf (GET) `/tutorials/<stg>`	
Aufruf-Parameter in der URL	
`<stg>` (optional)	ID eines Studiengangs, um die Tutorien dieses Studiengangs auszulesen. Gültige Studiengänge sind: ▪ `INFB`: Informatik Bachelor ▪ `INFM`: Informatik Master ▪ `MKIB`: Medien- und Kommunikationsinformatik Bachelor Wird die ID weggelassen, dann werden die Tutorien aller Studiengänge ausgelesen.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`NOT FOUND` (404)	Bei Angabe einer ungültigen Studiengangs-ID.
Ergebnis (Body als JSON-Array) Das Ergebnis ist ein Array mit JSON-Objekten des folgenden Aufbaus, wobei jeder Eintrag ein Tutorium beschreibt. Gibt es in dem Studiengang kein Tutorium, dann ist das Ergebnis ein leeres Array.	
`id`	Eindeutige ID des Tutoriums.
`courseOfStudies`	Array mit einer Einschränkung auf die Studiengänge, in denen das Tutorium angeboten wird. Ist der Wert `null`, dann wird das Tutorium in allen Studiengängen angeboten (keine Einschränkung). Die Namen der Studiengänge entsprechen denen aus dem Aufruf.
`lectureName`	Name des Tutoriums, in der Regel der Name der Vorlesung, für die das Tutorium angeboten wird.
`tutorName`	Name des Tutors.
`day`	Wochentag, an dem das Tutorium stattfindet. Die Zählung beginnt bei `0` für den Montag.
`time`	Zeitangabe, eventuell mit zusätzlichen Hinweisen, zum Tutorium.
`room`	Raum, in dem das Tutorium abgehalten wird.

building	Gebäude, in dem der Raum liegt.
startDate	String mit dem Datum, an dem das Tutorium erstmalig stattfindet, im Format yyyy.MM.dd.

3.3.6 Wahlpflichtfächer

Die Liste aller Wahlpflichtfächer eines Studiengangs wird über diesen Aufruf ermittelt.

Aufruf (GET)	
/compulsoryoptionalsubjects/<stg>	
Aufruf-Parameter in der URL	
<stg>	ID eines Studiengangs (siehe Abschnitt 3.3.5), um die Wahlpflichtfächer dieses Studiengangs auszulesen. Wird die ID weggelassen, dann werden die Wahlpflichtfächer aller Studiengänge ausgelesen.
Ergebnis (HTTP-Status-Code)	
OK (200)	Im Erfolgsfall.
NOT FOUND (404)	Bei Angabe einer ungültigen Studiengangs-ID.
Ergebnis (Body als JSON-Array)	
Das Ergebnis ist ein Array mit JSON-Objekten des folgenden Aufbaus, wobei jeder Eintrag ein Wahlpflichtfachmodul. beschreibt. Gibt es in dem Studiengang keine Wahlpflichtfächer, dann ist das Ergebnis ein leeres Array.	
courseOfStudies	Studiengang, dem das Modul mit der Veranstaltung zugeordnet ist. Der Aufbau entspricht dem des Aufrufparameters <stg>
idModule	ID des Wahlpflichtfachmoduls.
moduleName	Vollständiger und lokalisierter Name des Wahlpflichtfachmoduls.
compulsoryOptional-Subjects	Array mit Objekten, die die Wahlpflichtfächer in dem Modul beschreiben. Der Aufbau der Objekte wird im Folgenden beschrieben.

Beschreibung eines Wahlpflichtfachs, Objekt im Array compulsoryOptionalSubjects	
id	Eindeutige ID des Fachs (kann momentan noch -1 sein).
lectureName	Name des Fachs.
exam	Art der Prüfung.
internalName	Interne Prüfungsnummer der Veranstaltung.
lecturerName	Durch Komma getrennte Aufzählung der Namen der Dozenten, die die Veranstaltung durchführen.
contactHours	Anzahl Semesterwochenstunden.
creditPoints	Anzahl ECTS-Punkte, die für diese Veranstaltung vergeben werden.

3.4 Stundenpläne

Diese Schnittstelle liefert die Stundenplaninformationen. Momentan stammen sie noch direkt aus dem LSF.

Aufruf (GET)
/timetable/<stg>/<po>/<vert>/<sem>

Aufruf-Parameter in der URL	
`<stg>`	ID eines Studiengangs (siehe Abschnitt 3.3.5).
`<po>`	Nummer der Prüfungsordnung.
`<vert>`	Nummer der Vertiefungsrichtung, `0` für „keine“. Die Vertiefungsrichtung wird erst für den Master-Studiengang ab dem Wintersemester 2014/2015 benötigt.
`<sem>`	Nummer des Semesters (1 bis 7 in den Bachelor-Studiengängen, 1 bis 3 in den Master-Studiengängen)
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`NOT FOUND` (404)	Wenn der angeforderte Stundenplan nicht existiert.
Ergebnis (Body als JSON-Objekt) Das Ergebnis ist ein JSON-Objekt mit folgendem Aufbau.	
`courseOfStudies`	ID des Studiengangs, zu dem der Stundenplan gehört.
`semester`	Studiensemester des Stundenplans.
`semesterName`	Name des Semester (z.B. „WS 2014“).
`timetables`	Array mit JSON-Arrays, von denen der 0. Eintrag für den Montag, der 1. für den Dienstag usw. stehen. Jeder Eintrag ist selbst wiederum ein Array mit dem Namen `entries`, das einzelne JSON-Objekte, die für jeweils eine Veranstaltung an diesem Tag stehen, enthält. Finden an einem Tag keine Vorlesungen statt, dann ist das Aray für diesen Tag leer.
`blockcourses`	JSON-Array mit Veranstaltungs-Objekte für Blockveranstaltungen, kann leer sein.

Aufbau der Veranstaltungs-Objekte im Stundenplan (Attribut `timetables`) oder in den Blockveranstaltungen (Attribut `blockcourses`) innerhalb des JSON-Arrays.	
`startTime`	Startzeitpunkt der Veranstaltung, angegeben in Minuten seit Mitternacht.
`endTime`	Endzeitpunkt der Veranstaltung, angegeben in Minuten seit Mitternacht.
`interval`	Häufigkeit, mit der diese Veranstaltung stattfindet. Erlaubte Werte sind: ▪ `WEEKLY`: wöchentlich ▪ `FORTNIGHTLY`: 14-tägig ▪ `DAILY`: täglich ▪ `MONTHLY`: monatlich ▪ `SINGLE`: Einzelveranstaltung ▪ `BLOCK`: Blockveranstaltung
`lectureName`	Name der Veranstaltung.
`internalName`	Interne Prüfungsnummer der Veranstaltung.
`group`	Sollte es mehrere parallele Züge geben, dann steht hier der Name des Zuges, ansonsten ist der Eintrag `null`.
`locations`	Es handelt sich hierbei um ein Array mit den Orten, an denen diese Veranstaltung stattfindet. Es können 0 bis maximal zwei Orte angegeben sein. Ein Ort ist ein Objekt, dessen Aufbau weiter unten vorgestellt wird.

`lecturerNames`	Array mit den Namen der durchführenden Dozenten, kann leer sein, wenn im LSF die Dozentenangabe fehlt. Mehr als zwei Dozenten sind im Array nie vorhanden. Bei Seminaren, Projektarbeiten usw. steht hier „Professoren“ als einziger Eintrag im Array. Die Namen von Mitarbeitern als Übungsgruppenleiter stimmen in der Regel nicht, weil hier im Stundenplan anonymisierte Namen verwendet werden.
`duration`	Textuelle Angabe zur Dauer der Veranstaltung (Startdatum oder Startdatum sowie Enddatum, ...) oder ein Leerstring, wenn die Dauer nicht angegeben ist.
`comment`	Hinweistext zur Veranstaltung.
`cancellation`	Hinweis auf den Ausfall einer einzelnen Veranstaltung, ansonsten ein Leerstring,
`contactHours`	Anzahl an Semesterwochenstunden, die die Vorlesung umfasst.
`creditPoints`	Anzahl an ECTS-Punkten, die durch den Besuch der Vorlesung erworben werden können.

Aufbau eines Ortes, an dem eine Veranstaltung stattfindet (Objekt im Array des Attributs `locations`).	
`building`	Gebäude, in dem die Veranstaltung stattfindet.
`room`	Raum, in dem die Veranstaltung stattfindet.

3.5 Freie Räume

Diese Schnittstelle liefert alle unbelegten Hörsäle und Pool-Räume zu ausgesuchten Zeitpunkten. Es werden hierbei nur die eigenen Räume des Fachgebiets berücksichtigt, weil die Belegung fremder Räume im Intranet nicht bekannt ist.

3.5.1 Aktuelle Ansicht

In der aktuellen Sicht werden die freien Räume zum Zeitpunkt des Aufrufs ausgelesen.

Aufruf (GET)	
`/unoccupiedrooms/<type>/now`	
Aufruf-Parameter in der URL	
`<type>`	Art des gesuchten Raums. Erlaubt sind die Werte `lecturehalls` für Hörsäle und `computerpools` für Computer-Poolräume.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`NOT FOUND` (404)	Wenn im angegebenen Zeitraum keine Raumbelegung erfasst ist. So finden beispielsweise abends gelegentlich externe Veranstaltungen statt, die nicht im Intranet bekannt sind.
Ergebnis (Body als JSON-Objekt)	
Das Ergebnis ist ein JSON-Objekt mit folgendem Aufbau.	
`startTime`	Startzeitpunkt der Zeitraums, angegeben in Minuten seit Mitternacht.
`endTime`	Endzeitpunkt der Zeitraums, angegeben in Minuten seit Mitternacht.
`locations`	Es handelt sich hierbei um ein Array mit den Angaben zu nicht belegten Räumen (Orten). Ein Ort ist ein Objekt, dessen Aufbau weiter unten vorgestellt wird. Der Aufbau entspricht dem aus dem Stundenplan.

Aufbau eines Ortes (Objekt im Array des Attributs `locations`).	
`building`	Gebäude, in dem sich der Raum befindet.
`room`	Nummer des freien Raums

3.5.2 Tagesansicht

In der Tagesansicht werden die freien Räume alle Zeitslots am Tag des Aufrufs ausgelesen.

Aufruf (GET)	
`/unoccupiedrooms/<type>/today`	
Aufruf-Parameter in der URL	
`<type>`	Art des gesuchten Raums. Erlaubt sind die Werte `lecturehalls` für Hörsäle und `computerpools` für Computer-Poolräume.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`NOT FOUND` (404)	Wenn im angegebenen Zeitraum keine Raumbelegung erfasst ist. So sind z.B. für Sonntage keine Belegungen vorhanden.
Ergebnis (Body als JSON-Objekt)	
Das Ergebnis ist ein JSON-Objekt des folgenden Aufbaus.	
`entries`	Array mit allen Zeitangaben (Zeitslots) und den freien Räumen in dem Slot. Die einzelnen Einträge des Arrays sind bereits im Aufruf 3.5.1 vorgestellt worden.

3.5.3 Wochenansicht

In der Wochenansicht werden die freien Räume alle Zeitslots aller Tage der Woche ausgelesen.

Aufruf (GET)	
`/unoccupiedrooms/<type>/week`	
Aufruf-Parameter in der URL	
`<type>`	Art des gesuchten Raums. Erlaubt sind die Werte `lecturehalls` für Hörsäle und `computerpools` für Computer-Poolräume.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`NOT FOUND` (404)	Wenn im angegebenen Zeitraum keine Raumbelegung erfasst ist. So sind z.B. für Sonntage keine Belegungen vorhanden.
Ergebnis (Body als JSON-Array)	
Das Ergebnis ist ein Array mit JSON-Objekten.	
`entries`	Jedes Objekt im Array beinhaltet die freien Räume für einen Wochentag, wobei an Index 0 der Montag steht. Die einzelnen Objekte im Arrays sind bereits im Aufruf 3.5.2 vorgestellt worden.

3.6 Gebäude

Diese Schnittstelle liefert die Informationen zu allen Einrichtungen der Hochschule.

Aufruf (GET) `/buildings/all`	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Immer.
Ergebnis (Body als JSON-Array) Das Ergebnis ist ein Array mit JSON-Objekten des folgenden Aufbaus, wobei jeder Eintrag für eine ein Gebäude auf dem Campus steht	
`id`	Eindeutige ID des Gebäudes.
`name`	Name des Gebäudes wie z.B. „E“ oder „Mensa“.
`entry`	Name eines Eingangs zum Gebäude, wenn die Gebäudeeingänge hier separat eingetragen sind.
`departments`	JSON-Array aller Einrichtungen oder Fakultäten im Gebäude. Der Aufbau der Objekte im Array ist weiter unten erläutert.
`information`	Informationen zum Gebäude (z.B. eine aktuelle Veranstaltung).
`latitude`	Breitengrad, auf dem sich das Gebäude befindet.
`longitude`	Längengrad, auf dem sich das Gebäude befindet.
`altitude`	Höhenlage des Gebäudes über dem Meeresspiegel.

Die Einrichtungen und Fakultäten besitzen diesen Aufbau (Attribut `departments`).	
`id`	Eindeutige ID der Einrichtung.
`name`	Name der Einrichtung wie z.B. „Informationszentrum“.
`shortName`	Kurzname der Einrichtung wie z.B. „IZ“.
`openingTimes`	Öffnungszeiten der Einrichtung, mehrere Einträge sind durch Zeilenumbrüche getrennt.
`information`	Informationen zur Einrichtung.

Sowohl im Gebäude als auch in den Einrichtungen können in der Datenbank im Attribut `information` Platzhalter verwendet werden. Diese werden beim Ausliefern der Informationen durch dynamische Inhalte ersetzt. Platzhalter können in weiteren HTML-Code eingebettet sein.

- `#{canteen}`: Dieser Platzhalter wird durch eine Kurzbeschreibung des Mensaessens am Tag des Aufrufs ersetzt. Aufbau der Beschreibung (die Kommentare dienen hier nur der Verdeutlichung und werden nicht ausgeliefert):

```
<div class="meals">
  <!-- Je Essensausgabe eine Gruppe -->
  <div class="meals group">
    <!-- Die Nachricht wird nur ausgeliefert, wenn wirklich
         eine Nachricht vorhanden ist (z.B Ausgabe
         geschlossen) -->
    <div class="meals group message">Nachricht<div>
    <!-- Name der Essensausgabe -->
    <div class="meals group title">Ausgabename</div>
    <!-- Je Essen an dieser Ausgabe ein Eintrag -->
```

```
    <div class="meals group meal">Essen 1</div>
    <div class="meals group meal">Essen 2</div>
  </div>
</div>
```

- `#{consultationTimes}`: Dieser Platzhalter wird durch eine Beschreibung der Dozentensprechzeiten am Tag des Aufrufs ersetzt. Nur diejenigen Dozenten sind aufgeführt, die an diesem tag ihre Sprechstunden haben. Aufbau der Beschreibung (die Kommentare dienen hier nur der Verdeutlichung und werden nicht ausgeliefert):

```
<div class="consultationtimes">
  <!-- Je Dozent ein Eintrag -->
  <div class="consultationtimes lecturer">
    <!-- Name des Dozenten -->
    <div class="consultationtimes lecturer name">
      Prof. Dr.-Ing. Holger Vogelsang
    </div>
    <!-- Raumnummer des Dozenten -->
    <div class="consultationtimes lecturer room">
      207
    </div>
    <!-- Sprechzeit -->
    <div class="consultationtimes lecturer time">
      10:00 - 11:00 und nach Vereinbarung
    </div>
  </div>
</div>
```

3.7 Praxisbörse

Diese Schnittstelle liefert Angebote und Firmeninformationen aus der Praxisbörse. Die Anmeldeinformationen des Aufrufers müssen bei allen Aufrufen per Basic-Authentication übergeben werden (siehe auch Abschnitt 3.2.4). Der Studiengang des Aufrufers wird aus seinen Anmeldedaten ermittelt. Die Treffer werden auf diesen Studiengang automatisch eingeschränkt.

3.7.1 Angebote

Die folgenden Aufrufe ermitteln die Angebote eines Typs (wie z.B. für das Praxissemester oder für eine Abschlussarbeit). Die Angabe von Schlüsselwörtern, anhand derer die Treffer gefiltert werden, ist möglich.

Aufruf (GET)	
`/joboffer/offers/<typ>/<schlüssel>/<start>/<anzahl>` `/joboffer/offers/<typ>/<start>/<anzahl>`	
Aufruf-Parameter in der URL	
`<typ>`	Art des Angebots. Folgende Typen werden unterstützt: - `internship`: Praxissemester - `joboffer`: Angebot für eine Mitarbeit nach dem Studium. - `workingstudent`: Angebot für eine Mitarbeit als Werkstudent innerhalb des Studiums. - `thesis`: Angebot für eine Abschlussarbeit. Groß- und Kleinschreibung werden nicht unterschieden.

<schlüssel> (optional)	Folge von Schlüsselwörtern, die im Angebot vorkommen müssen. Die Schlüsselwörter werden nur berückichtigt, wenn Sie mindestens zwei Zeichen lang sind. Mehrere Schlüsselwörter in der Anfrage werden durch Leerzeichen, Tabulatoren, Punkte, Kommata, Doppelpunkte, Semikolon oder Zeilenumbrüche von einander getrennt. Fehlt der optionale Parameter, dann werden alle Angebote eines Typs berücksichtigt.
<start>	Nummer des ersten Treffers in der Liste aller Treffer. Damit kann Clientseitig ein „Paging“ gebaut werden, um nicht sofort alle Treffer auslesen zu müssen. Die Zählung beginnt mit Index 0.
<anzahl>	Anzahl der Treffer, die maximal ausgelesen werden soll. Der Wert `-1` steht für „alle Treffer“.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
Ergebnis (Body als JSON-Array) Das Ergebnis ist ein JSON-Objekt mit folgendem Aufbau:	
`totalHits`	Gesamtanzahl der Treffer, die zum Suchkriterium passen. Die Anzahl kann Größer als die Anzahl zurückgegebener Angebote sein, wenn mit <anzahl> die Summe der Treffer, die zurückgegeben werden soll, eingeschränkt wurde. Der Wert von `totalHits` dient also dazu, beim „Paging“ das Nachladen weiterer Treffer zuzulassen.
`offers`	Array mit allen Angebotsobjekten. Diese werden weiter unten beschrieben.
`companies`	Hash-Tabelle mit all den Firmen, von denen Angebote als Treffer zurückgegeben werden. Der Schlüssel ist die ID der Firma, der Wert das eigentliche Firmenobjekt.
`offerTypes`	Hash-Tabelle mit Informationen zu allen Angebotstypen, die in den Treffern vorhanden sind. Der Schlüssel ist die ID des Typs, der Wert das eigentliche Objekt.
`countries`	Hash-Tabelle mit Informationen zu allen Ländern, in denen Angebote im Treffer vorhanden sind. Der Schlüssel ist die ID des Landes, der Wert das eigentliche Objekt.
`industrialSectors`	Hash-Tabelle mit Informationen zu allen Industrie-Branchen der Firmen, deren Angebote in der Trefferliste vorhanden sind. Der Schlüssel ist die ID der Branche, der Wert das eigentliche Objekt.

Die Angebote aus dem Array `offers` haben folgenden Aufbau:	
`id`	Eindeutige ID, ändert sich innerhalb eines Angebots nicht.
`validDate`	Datum, bis zu dem das Angebot noch gültig ist. Danach wird es automatisch vom Server gelöscht. Es wird das Format `yyyy.MM.dd` verwendet.
`contact`	Das Objekt beinhaltet die Kontaktperson der Firma für genau dieses Angebot. Der Aufbau wird weiter unten beschreiben.
`shortDescription`	Kurzbeschreibung der ausgeschriebenen Stelle.
`description`	Vollständige Beschreibung der ausgeschriebene Stelle, kann Zeilenumbrüche enthalten.
`filename`	Name der PDF-Datei mit weiteren Informationen zur Stelle. Ist der Wert `null`, dann gibt es keine Zusatzinformationen. Die PDF-Datei kann zur Zeit nicht über die REST-Schnittstelle heruntergeladen werden.
`zipCode`	Postleitzahl der Arbeitsstätte für diese Stelle.

usageSite	Einsatzort der Arbeitsstätte für diese Stelle.
dateAdded	Datum, an dem das Angebot veröffentlicht wurde. Es wird das Format yyyy.MM.dd verwendet.
typeId	ID des Angebotstyps (siehe weiter unten). Das Typobjekt kann aus der Hash-Tabelle offerTypes der Antwort ausgelesen werden.
countryId	ID des Landes des Sitzes der Firma, die das Angebot veröffentlicht hat. Das Länderobjekt kann aus der Hash-Tabelle countries der Antwort ausgelesen werden.
companyId	ID der Firma, die das Angebot veröffentlicht hat. Das Firmenobjekt kann aus der Hash-Tabelle companies der Antwort ausgelesen werden.
onNotepad	true, wenn das Angebot sich auf dem internen Merkzettel des Aufrufers befindet, ansonsten false.

Aufbau des Objektes, das die Kontaktperson (Attribut contact einer Firma und eines Angebots beschreibt:	
formOfAddress	Anrede der Kontaktperson.
firstName	Vorname der Kontaktperson.
secondName	Nachname der Kontaktperson.
telephone	Telefonnummer der Kontaktperson, kann null sein.
fax	Faxnummer der Kontaktperson, kann null sein.
email	E-Mail-Adresse der Kontaktperson, kann null sein.

Die Verwendung von IDs der Firmen usw. mit separater Auflistung in der Rückgabe hat den Vorteil, dass bei mehreren Treffern einer Firma die Firmenbeschreibung nur einmal im Ergebnis erscheint und somit die Datenmenge reduziert wird.

Die Firmenobjekte aus der Hash-Tabelle companies haben folgenden Aufbau:	
id	Eindeutige ID, ändert sich nicht.
companyName	Name der Firma.
website	URL mit der Web-Seite der Firma, kann null sein.
description	Kurze Beschreibung der Firma, kann null sein.
street	Straße des Firmensitzes.
postofficeBox	Postfach des Firmensitzes, kann null sein.
zipCode	Postleitzahl des Firmensitzes, kann null sein.
city	Ort des Firmensitzes, kann null sein.
contact	Das Objekt beinhaltet die generelle Kontaktperson der Firma. Der Aufbau wurde bereits oben beschreiben.
numberOfEmployees	Anzahl Mitarbeiter der Firma, kann null sein.
countryId	ID des Landes, in dem sich der Firmensitz befindet. Das Länderobjekt kann aus der Hash-Tabelle countries der Antwort ausgelesen werden.
numberOfOffers	Anzahl an Angeboten, die diese Firma in der Praxisbörse veröffentlicht hat.

`industrialSectorIds`	Array mit IDs der Branchen, in denen die Firma vertreten ist. Die Branchen-Objekte selbst können aus der Hash-Tabelle des Rückgabewertes `industrialSectors` der Antwort ausgelesen werden.

Die Objekte mit den genauen Beschreibungen der Angebotstypen aus der Hash-Tabelle `offerTypes` haben folgenden Aufbau:	
`id`	Eindeutige ID, ändert sich nicht.
`name`	Lokalisierter Name des Typs (z.B. „Praxissemester“).
`shortname`	Nicht lokalisierter, interner Name des Typs. Er entspricht dem Wert des Parameters `<typ>` aus dem Aufruf. Daher existieren diese Kurznamen: ■ `internship`: Praxissemester ■ `joboffer`: Angebot für eine Mitarbeit nach dem Studium. ■ `workingstudent`: Angebot für eine Mitarbeit als Werkstudent innerhalb des Studiums. ■ `thesis`: Angebot für eine Abschlussarbeit. Weitere Namen können in Zukunft hinzu kommen.

Die Objekte mit den Beschreibungen der Länder aus der Hash-Tabelle `countries` haben folgenden Aufbau:	
`id`	Eindeutige ID, ändert sich nicht.
`code`	Ländercode gemäß ISO 3166 (siehe Java-Klasse `Locale`, beispielsweise „DE“ für „Deutschland“).
`language`	Sprachcode gemäß ISO 639 (siehe Java-Klasse `Locale`, z.B. „de“ für „deutsch“).
`name`	Lokalisierter Name des Landes (z.B. „Deutschland“).

Die Objekte mit den vollständigen Beschreibungen der Branche aus der Hash-Tabelle `industrialSectors` haben folgenden Aufbau:	
`id`	Eindeutige ID, ändert sich nicht.
`name`	Lokalisierter Name des Landes (z.B. „Automatisierung“).

3.7.2 Merkzettel

Studierende können sich Angebote auf einem internen Merkzettel auf dem Server speichern und bei Bedarf wieder abrufen. Laufen die Angebote ab, dann werden sie auch automatisch vom Merkzettel gelöscht.

3.7.2.1 Lesezugriffe

Die folgenden Aufrufe suchen Angebote, die sich auf dem eigenen Merkzettel befinden.

Aufruf (GET)

`/joboffer/notepad/<typ>/<schlüssel>/<start>/<anzahl>`
`/joboffer/notepad/<typ>/<start>/<anzahl>`
`/joboffer/notepad/<start>/<anzahl`

Aufruf-Parameter in der URL

`<typ>` (optional)	Art des Angebots. Folgende Typen werden unterstützt: ■ `internship`: Praxissemester ■ `joboffer`: Angebot für eine Mitarbeit nach dem Studium. ■ `workingstudent`: Angebot für eine Mitarbeit als Werkstudent innerhalb des Studiums. ■ `thesis`: Angebot für eine Abschlussarbeit. Groß- und Kleinschreibung werden nicht unterschieden. Fehlt der optionale Typ, dann werden alle Angebotstypen berücksichtigt.
`<schlüssel>` (optional)	Folge von Schlüsselwörtern, die im Angebot vorkommen müssen. Die Schlüsselwörter werden nur berückichtigt, wenn Sie mindestens zwei Zeichen lang sind. Mehrere Schlüsselwörter in der Anfrage werden durch Leerzeichen, Tabulatoren, Punkte, Kommata, Doppelpunkte, Semikolon oder Zeilenumbrüche von einander getrennt. Fehlt der optionale Parameter, dann werden alle Angebote eines Typs auf dem Merkzettel berücksichtigt.
`<start>`	Nummer des ersten Treffers in der Liste aller Treffer. Damit kann Clientseitig ein „Paging“ gebaut werden, um nicht sofort alle Treffer auslesen zu müssen. Die Zählung beginnt mit Index 0.
`<anzahl>`	Anzahl der Treffer, die maximal ausgelesen werden soll. Der Wert `-1` steht für „alle Treffer“.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
Ergebnis (Body als JSON-Array)	
Das Ergebnis ist ein JSON-Objekt, das genau dem Ergebnis-Objekt aus Abschnitt 3.7.1 entspricht.	

3.7.2.2 Schreibzugriffe

Es gibt mehrere unterschiedliche Möglichkeiten, den Merkzettel zu verändern.

Ein Angebot auf dem Merkzettel speichern

Der folgende Aufruf speichert ein Angebot auf dem Merkzettel. Es muss nicht vorab geprüft werden, ob das Angebote bereits auf dem Merkzettel liegt.

Aufruf (POST)	
`/joboffer/notepad/offer`	
Aufruf-Parameter als String (Body)	
Im Request-Body wird die ID des zu merkenden Angebots gespeichert.	
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`BAD REQUEST` (400)	Wenn kein Angebot mit der übergebenen ID existiert.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
Ergebnis (Body)	
Der Rumpf der Antwort ist immer leer.	

Mehrere Angebote auf dem Merkzettel speichern

Der folgende Aufruf speichert mehrere Angebote auf dem Merkzettel. Es muss nicht vorab geprüft werden, ob die Angebote bereits auf dem Merkzettel liegen.

Aufruf (POST)	
`/joboffer/notepad/offers`	
Aufruf-Parameter als JSON-Array (Body)	
Im Request-Body werden die IDs der zu merkenden Angebote als JSON-Array gespeichert.	
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`BAD REQUEST` (400)	Wenn für mindestens eine der übergebenen IDs kein Agebot existiert. Im Fehlerfall wird keine Änderung auf dem Merkzettel vorgenommen.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
Ergebnis (Body)	
Der Rumpf der Antwort ist immer leer.	

Ein Angebot vom Merkzettel löschen

Der folgende Aufruf löscht ein Angebot vom Merkzettel. Ist das zu löschende Angebot nicht auf dem Merkzettel gespeichert, dann wird der Aufruf ignoriert.

Aufruf (DELETE)	
`/joboffer/notepad/offer/<id>`	
Aufruf-Parameter in der URL	
`<id>`	ID des vom Merkzettel zu löschenden Angebots.
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
Ergebnis (Body)	
Der Rumpf der Antwort ist immer leer.	

3.7.3 Firmen

Firmen lassen sich anhand unterschiedlicher Kriterien durchsuchen.

3.7.3.1 Alle Firmen durchsuchen

Die folgenden Aufrufe lesen entweder alle Firmen oder aber Firmen, deren Namen mit einer bestimmten Zeichenfolge anfangen, aus.

Aufruf (GET)	
`/joboffer/companies/all/<name>` `/joboffer/companies/all`	
Aufruf-Parameter in der URL	
`<name>` (optional)	Buchstabenfolge, mit der der Firmenname anfangen muss. Groß- und Kleinbuchstaben werden nicht unterschieden. Fehlt der Namensanfang, dann werden alle Firmen ausgelesen.
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
Ergebnis (Body als JSON-Array)	
Das Ergebnis ist ein Array mit Firmenobjekten, deren Aufbau bereits in Abschnitt 3.7.1 vorgestellt wurde. Das Array ist leer, wenn es keine passenden Treffer gibt.	

3.7.3.2 Firmen mit Angeboten suchen

Die folgenden Aufrufe lesen entweder alle Firmen oder aber Firmen, deren Namen mit einer bestimmten Zeichenfolge anfangen, aus. Dabei werden nur die Firmen berücksichtigt, die Angebote in die Praxisbörse eingestellt haben.

Aufruf (GET)	
`/joboffer/companies/withoffers/<name>` `/joboffer/companies/withoffers`	
Aufruf-Parameter in der URL	
`<name>` (optional)	Buchstabenfolge, mit der der Firmenname anfangen muss. Groß- und Kleinbuchstaben werden nicht unterschieden. Fehlt der Namensanfang, dann werden alle Firmen mit Angeboten ausgelesen.
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
Ergebnis (Body als JSON-Array)	
Das Ergebnis ist ein Array mit Firmenobjekten, deren Aufbau bereits in Abschnitt 3.7.1 vorgestellt wurde. Das Array ist leer, wenn es keine passenden Treffer gibt.	

3.7.3.3 Eine Firma auslesen

Der folgende Aufruf liest die Informationen zu der Firma aus, deren ID übergeben wurde.

Aufruf (GET)	
`/joboffer/company/<firmen-id>`	
Aufruf-Parameter in der URL	
`<firmen-id>`	ID der gesuchten Firma.
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
`NOT FOUND` (404)	Es existiert keine Firma mit der angegebenen ID.
Ergebnis (Body als JSON-Objekt)	
Das Ergebnis ist ein Firmenobjekt, dessen Aufbau bereits in Abschnitt 3.7.1 vorgestellt wurde.	

3.7.4 Angebotsarten

Der folgende Aufruf ermittelt möglichen alle Arten von Job-Angeboten in der Praxisbörse.

Aufruf (GET)	
`/joboffer/offertypes/all`	
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
Ergebnis (Body als JSON-Objekt)	
Das Ergebnis ist ein Array von Objekten, deren Aufbau bereits in Abschnitt 3.7.1 als Wert der Hash-Tabelle `offerTypes` vorgestellt wurde.	

3.7.5 Industrielle Branchen

Der folgende Aufruf ermittelt möglichen alle Arten industrieller Branchen, denen eine Firma in der Praxisbörse angehören kann.

Aufruf (GET)
`/joboffer/industrialsectors/all`
Authentifizierung

Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
Ergebnis (Body als JSON-Objekt)	
Das Ergebnis ist ein Array von Objekten, deren Aufbau bereits in Abschnitt 3.7.1 als Wert der Hash-Tabelle `industrialSectors` vorgestellt wurde.	

3.7.6 Länder

Der folgende Aufruf ermittelt möglichen alle Länder, aus denen Firmen in der Praxisbörse bisher stammen können.

Aufruf (GET)	
`/joboffer/countries/all+`	
Authentifizierung	
Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Sollte die URL also ohne Anmeldedaten aufgerufen werden, dann fordert der Server zunächst Anwender-Namen und Passwort an. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Der Aufruf war erfolgreich.
`UNAUTHORIZED` (401)	Die Anmeldeinformationen des Aufrufers sind nicht korrekt.
Ergebnis (Body als JSON-Objekt)	
Das Ergebnis ist ein Array von Objekten, deren Aufbau bereits in Abschnitt 3.7.1 als Wert der Hash-Tabelle `countries` vorgestellt wurde.	

3.8 Modulhandbuch

Diese Schnittstellen liefern Informationen zu den Modulen der einzelnen Studiengänge.

3.8.1 Modul-Abhängigkeiten

Diese Schnittstelle ermittelt alle inhaltlichen Abhängigkeiten zwischen den Modulen eines Studiengangs.

Aufruf (GET)	
`/mhb/modules/dependencies/<stg>/<po>/<vert>`	
Aufruf-Parameter in der URL	
`<stg>`	ID eines Studiengangs (siehe Abschnitt 3.3.5).
`<po>`	Nummer der Prüfungsordnung bzw. `0` für die aktuelle Prüfungsordnung.

<vert>	Nummer der Vertiefungsrichtung, 0 für „keine“. Die Vertiefungsrichtung wird erst für den Master-Studiengang ab dem Wintersemester 2014/2015 benötigt. Weitere Informationen zu den Vertiefungsrichtungen sind in Abschnitt 3.1 zu finden.
Ergebnis (HTTP-Status-Code)	
OK (200)	Im Erfolgsfall.
NOT FOUND (404)	Wenn einer der Parameter ungültig ist, es also die Vertiefung, die Prüfungsordnung oder den Studiengang nicht gibt.
Ergebnis (Body als JSON-Objekt) Das Ergebnis ist ein JSON-Objekt mit folgendem Aufbau.	
courseOfStudies	ID des Studiengangs, zu dem der Stundenplan gehört (entspricht dem Aufrufparameter <stg>.
examination-RegulationsNumber	Nummer der Prüfungsordnung.
semesterName	Name des Semester (z.B. „WS 2014“).
specialization	Name der Vertiefungsrichtung oder "", wenn es keine Vertiefungsrichtung gibt. Weitere Informationen zu den Vertiefungsrichtungen sind in Abschnitt 3.1 zu finden.
idSpecialization	ID der Vertiefungsrichtung oder 0, wenn es keine Vertiefungsrichtung gibt. Weitere Informationen zu den Vertiefungsrichtungen sind in Abschnitt 3.1 zu finden.
modules	JSON-Array mit einer Kurzbeschreibung aller Module des Studiengangs. Der Aufbau der einzelnen Module ist in der folgenden Tabelle beschrieben.

Aufbau eines Modul-Objekts im JSON-Array der Modulabhängigkeiten (Attribut modules).	
id	Interne ID des Moduls. Achtung: Diese IDs sind die einzigen im Intranet, die negativ sein können. Das ist ein Relikt aus alten Zeiten.
internalName	Namenskürzel des Moduls laut Prüfungsordnung.
name	Vollständiger Name des Moduls.
lecturer	Name des für das Modul verantwortlichen Dozenten.
idLecturer	ID des für das Modul verantwortlichen Dozenten.
internalName	Interne Prüfungsnummer der Veranstaltung.
dependentOn	JSON-Array mit den IDs der Module, von denen dieses inhaltlich abhängig ist.

3.8.2 Module

Diese Schnittstelle ermittelt die Beschreibung eines Moduls.

Aufruf (GET) /mhb/module/<stg>/<int-name>	
Aufruf-Parameter in der URL	
<stg>	ID eines Studiengangs (siehe Abschnitt 3.3.5).
<int-name>	Interner Name des Moduls laut Prüfungsordnung.
Ergebnis (HTTP-Status-Code)	

`OK` (200)	Im Erfolgsfall.
`NOT FOUND` (404)	Wenn es in dem angegebenen Studiengang kein Modul mit dem Namen gibt.
Ergebnis (Body als JSON-Objekt) Das Ergebnis ist ein JSON-Objekt mit folgendem Aufbau.	
`id`	Interne ID des Moduls. Achtung: Diese IDs sind die einzigen im Intranet, die negativ sein können. Das ist ein Relikt aus alten Zeiten.
`longName`	Name des Moduls.
`semester`	Nummer des Studiensemesters, in dem das Modul angeboten wird.
`internalName`	Interner Kurzname des Moduls laut Studien- und Prüfungsordnung, entspricht dem Aufrufparameter `int-name`.
`contactHours`	Anzahl an Semesterwochenstunden, die das Modul belegt.
`creditpoints`	ECTS-Punkte, die für das Modul vergeben werden.
`precondition`	Formale Vorbedigungen, die vor dem Besuch des Moduls erfüllt sein müssen. Die Vorbedingungen werden in einer der folgenden Tabelle beschrieben.
`skills`	Beschreibung der durch das Modul vermittelten Kompetenzen. Der Eintrag kann neben textuellen Angaben die folgenden XHTML-Tags (ohne Attribute) enthalten: `li`, `ul`, `ol`, `p`, `br`, `table`, `thead`, `tbody`, `tfoot`, `td`, `tr`, `th`, `div`, `title`, `em`, `b`, `blockquote`, `cite`, `code`, `dd`, `dl`, `dt`, `dfn`, `dir`, `i`, `kbd`, `pre`, `q`, `s`, `samp`, `span`, `strong`, `abbr`, `acronym`, `sub`, `sup`, `tt`, `u`. Weiterhin kann der Eintrag Steuerzeichen für Tabulatoren (`\t`) und Zeilenumbrüche (`\r`, `\n`) enthalten.
`specialization`	Beschreibung des Vertiefungsgebiets, zu dem das Modul gehört. Die Beschreibung ist in Abschnitt 3.1 zu finden. Ist der Eintrag `null`, dann gehört das Modul zu keinem Vertiefungsgebiet oder zu allen Vertiefungsgebieten eines Studiengangs.
`examName`	Name der Prüfung des Moduls laut Studien- und Prüfungsordnung.
`lecturer`	Name und akademischer Grad des Dozenten, der für das Modul verantwortlich ist.
`idLecturer`	Eindeutige ID des Dozenten, der für das Modul verantwortlich ist.
`exam`	Beschreibung der Art der Prüfung des Moduls, wenn es für alle Veranstaltungen des Moduls eine gemeinsame Prüfung gibt („Modulprüfung“). Werden die Veranstaltungen des Moduls dagegen einzeln geprüft, dann ist dieser Eintrag `null`. Der Aufbau dieses Objektes ist in einer der Folgetabellen näher beschrieben.

Aufbau einer Modul-Vorbedingung (Attribut `precondition`).	
`needsFreeText`	Freitextangabe einer Vorbedingung wie z.B. „x ECTS-Punkte aus dem Vorstudium“.
`needsPreDegree`	`true`: Ein abgeschlossenes Vorstudium ist erforderlich. `false`: Das Vorstudium muss nicht abgeschlossen sein.
`needsModule`	`true`: Es muss vorher das Modul mit der ID `idModuleNeeded` abgeschlossen sein. `false`: Der Eintrag `idModuleNeeded` ist nicht relevant, da kein Modul Vorbedingung ist.
`idNeededModule`	ID des Moduls, das vorher erfolgreich besucht werden muss. Der Eintrag ist nur dann relevant, wenn das Attribut `needsModule` den Wert `true` besitzt.

Aufbau einer Prüfungsleistung (Attribut `exam`). Die Werte der Attribute können zu einer Ausgabe zusammengesetzt werden: <Wert von `type`> `length` <Wert von `unit`>.	
`length`	Dauer der Prüfung.
`type`	Art der Prüfung. Die möglichen Werte sind in Abschnitt 3.8.5 beschrieben.
`unit`	Einheit, in der die Dateuer angegeben ist. Die möglichen Werte sind in Abschnitt 3.8.6 beschrieben.

3.8.3 Veranstaltung

Diese Schnittstelle ermittelt die Beschreibung einer Veranstaltung eines Moduls.

Aufruf (GET) `/mhb/lecture/<stg>/<int-name>`	
Aufruf-Parameter in der URL	
`<stg>`	ID eines Studiengangs (siehe Abschnitt 3.3.5).
`<int-name>`	Interner Name der Veranstaltung laut Prüfungsordnung.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`NOT FOUND` (404)	Wenn es in dem angegebenen Studiengang keine Veranstaltung mit dem Namen gibt.
Ergebnis (Body als JSON-Objekt) Das Ergebnis ist ein JSON-Objekt mit folgendem Aufbau.	
`id`	Interne ID der Veranstaltung. Achtung: Diese IDs sind die einzigen im Intranet, die negativ sein können. Das ist ein Relikt aus alten Zeiten.
`longName`	Name der Veranstaltung.
`semester`	Nummer des Studiensemesters, in dem die Veranstaltung angeboten wird.
`internalName`	Interner Kurzname der Veranstaltung laut Studien- und Prüfungsordnung, entspricht dem Aufrufparameter `int-name`.
`contactHours`	Anzahl an Semesterwochenstunden, die die Veranstaltung belegt.
`creditpoints`	ECTS-Punkte, die für das Modul vergeben werden.
`english`	`true`: Die Veranstaltung wird auf englisch gehalten. `false`: Die Veranstaltung findet auf deutsch statt.
`comment`	Kommentar zur Veranstaltung. Der Eintrag kann neben textuellen Angaben die XHTML-Tags (ohne Attribute) und Steuerzeichen enthalten, die im Attribut `goals` der Module (siehe 3.8.2) aufgezählt wurden.
`contents`	Inhaltliche Ziele der Veranstaltung. Der Eintrag kann neben textuellen Angaben die XHTML-Tags (ohne Attribute) und Steuerzeichen enthalten, die im Attribut `goals` der Module (siehe 3.8.2) aufgezählt wurden.
`workload`	Arbeitsaufwand aus studentischer Sicht, rein textuelle Angabe ohne Formatierungen.
`material`	Lehrmaterialien. Der Eintrag kann neben textuellen Angaben die XHTML-Tags (ohne Attribute) und Steuerzeichen enthalten, die im Attribut `goals` der Module (siehe 3.8.2) aufgezählt wurden.

`lecturers`	JSON-Array mit den akademischen Graden und Namen aller Dozenten, die an der Veranstaltung beteiligt sind. Deren IDs werden zur Zeit nicht berücksichtigt.
`moduleName`	Name des Moduls, zu dem die Veranstaltung gehört.
`lectureType`	Art der Veranstaltung. Alle möglichen Werte sind in Abschnitt 3.8.7 vorgestellt.
`elective`	`true`: Es handelt sich um ein Wahlpflichtfach. `false`: Es handelt sich nicht um ein Wahlpflichtfach.
`reading`	`true`: Die Veranstaltung wird von einem Dozenten „gelesen“ (z.B. Vorlesung). `false`: Die Veranstaltung wird nicht „gelesen“ (z.B. Übung, E-Learning-Kurs).
`weighting`	Gewichtung der Prüfungsnote der Veranstaltung im Zeugnis.
`exam`	Beschreibung der Art der Prüfung der Veranstaltung, wenn es sich um eine Einzelprüfung handelt. Werden alle Veranstaltungen des übergeordneten Moduls gemeinsam geprüft („Modulprüfung“), dann ist dieser Eintrag `null`. Der Aufbau dieses Objektes ist bereits im Abschnitt 3.8.2 vorgestellt worden.

3.8.4 Veranstaltungen

Diese Schnittstelle ermittelt die Beschreibungen aller Veranstaltungen eines Moduls.

Aufruf (GET)	
`/mhb/lectures/<stg>/<int-name>`	
Aufruf-Parameter in der URL	
`<stg>`	ID eines Studiengangs (siehe Abschnitt 3.3.5).
`<int-name>`	Interner Name des Moduls laut Prüfungsordnung.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`NOT FOUND` (404)	Wenn es in dem angegebenen Studiengang kein Modul mit dem Namen gibt.
Ergebnis (Body als Array mit JSON-Objekten)	
Das Ergebnis ist ein Array mit JSON-Objekten, deren Aufbau bereits als Ergebnis des Aufrufs in Abschnitt 3.8.3 vorgestellt wurde.	

3.8.5 Prüfungsarten

Diese Schnittstelle ermittelt eine Beschreibung aller möglichen Prüfungsarten.

Aufruf (GET)	
`/mhb/examtypes`	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Immer.
Ergebnis (Body als JSON-Hash-Tabelle	
Das Ergebnis ist eine JSON-Hash-Tabelle, in der der Schlüssel die Art der Prüfung darstellt und der zugehörige Wert deren lokalisierte Repräsentation.	

	Die folgenden Schlüssel/Werte-Paare werden verwendet: ▪ `exam_type_bachelor_thesis`: Bachelor-Thesis ▪ `exam_type_concept`: Entwurf ▪ `exam_type_exercise`: Übung ▪ `exam_type_hands_on_work`: Praktische Arbeit ▪ `exam_type_homework`: Hausarbeit ▪ `exam_type_laboratory_work`: Laborarbeit ▪ `exam_type_master_thesis`: Master-Thesis ▪ `exam_type_online_test`: Online-Prüfung ▪ `exam_type_presentation`: Referat ▪ `exam_type_student_research_project`: Studienarbeit ▪ `exam_type_tutorial`: Tutorium ▪ `exam_type_verbal_exam`: Mündliche Prüfung ▪ `exam_type_written_exam`: Klausur ▪ `exam_type_written_or_verbal_exam`: Klausur/mündl. Prüfung

3.8.6 Prüfungslängeneinheiten

Diese Schnittstelle ermittelt eine Beschreibung aller möglichen Längeneinheiten von Prüfungen.

Aufruf (GET) `/mhb/examlengthunits`	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Immer.
Ergebnis (Body als JSON-Hash-Tabelle Das Ergebnis ist eine JSON-Hash-Tabelle, in der der Schlüssel die Einheit der Länge der Prüfung darstellt und der zugehörige Wert deren lokalisierte Repräsentation.	
	Die folgenden Schlüssel/Werte-Paare werden verwendet: ▪ `exam_unit_days`: Tage ▪ `exam_unit_hours`: Stunden ▪ `exam_unit_min`: Min. ▪ `exam_unit_month`: Monat ▪ `exam_unit_months`: Monate ▪ `exam_unit_quantity`: Stück ▪ `exam_unit_semester`: Semester ▪ `exam_unit_servey_percentage`: % der Online-Testumfragen ▪ `exam_unit_week`: Woche

3.8.7 Veranstaltungsarten

Diese Schnittstellen ermittelt eine Beschreibung aller möglichen Veranstaltungsarten.

Aufruf (GET) `/mhb/lecturetypes`	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Immer.
Ergebnis (Body als JSON-Hash-Tabelle	

Das Ergebnis ist eine JSON-Hash-Tabelle, in der der Schlüssel den Typ der Veranstaltung darstellt und der zugehörige Wert deren lokalisierte Repräsentation.	
	Die folgenden Schlüssel/Werte-Paare werden verwendet: - `lecture_type_laboratory`: Labor - `lecture_type_lecture`: Vorlesung - `lecture_type_colloquium`: Kolloquium - `lecture_type_exercise`: Übung - `lecture_type_on_the_job`: Praktische Arbeit - `lecture_type_thesis`: Abschlussarbeit - `lecture_type_seminar`: Seminar - `lecture_type_project_lecture`: Projektvorlesung - `lecture_type_hand_on`: Praktische Arbeit

4

REST-Schnittstelle (IWII, experimentell)

Die Schnittstelle ist experimentell und dient zur Zeit dazu, Vorlesungsfeedback zu sammeln und wiederzugeben.

4.1 Vorlesungsfeedbacks

Die Aufrufe dienen dazu, während der Vorlesung Feedbackinformationen zu sammeln. Alle Angaben zu Zeitpunkten müssen im Format `yyyy-MM-dd HH:mm:ss.SSS` aufgeführt werden bzw. werden in dem Format zurückgegeben. Dabei darf die Zeitangabe auch fehlen. Damit die recht langsamen ADS-Anmeldungen bei sehr häufigen Zugriffen auf den Server nicht unnötig bremsen, werden für die Feedback-Anwendung Hash-Codes (SHA-256) zur Identifikation von Benutzern analog zu Session-IDs verwendet. Feedbacks von Studenten liegen auf dem Server anonymisiert vor, so dass eine Zuordnung zu einzelnen Personen nicht möglich ist.

4.1.1 Vorbereitungsphase

Durch diese Phase wird das Feedback ermöglicht.

4.1.1.1 Vorlesung für Feedbacks vorbereiten

Dieser Schritt wird von einem Dozenten durchgeführt, um eine eindeutige 4-stellige PIN für eine Vorlesungsabstimmung zu erhalten. Die PIN wird dem Studenten mitgeteilt. Auf dem Server wird die PIN zusammen mit einem eindeutigen Hash-Code des Dozenten abgelegt. Durch den Hash-Code authentifiziert sich der Dozent später für administrative Aufgaben gegenüber dem Server.

Aufruf (POST)	
`/feedback/lecture/<veranst>`	
Aufruf-Parameter in der URL	
`<veranst>`	Interner Name der Veranstaltung, zu der Feedback gesammelt werden soll (z.B. `IB 211`).
Authentifizierung	

Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`UNAUTHORIZED` (401)	Es wurden keine oder falsche Anmeldeinformationen übergeben.
`NOT FOUND` (404)	Es gibt keine Veranstaltung mit dem angegebenen Namen.
`GATEWAY TIMEOUT` (504)	Es konnte keine Verbindung mit dem Knowledge-Server hergestellt werden. Die Operation wurde nicht durchgeführt.
Ergebnis (Body JSON-Objekt) Das Ergebnis ist ein JSON-Objekt des folgenden Aufbaus.	
`pin`	PIN, die den Studenten mitgeteilt wird, um die Vorlesung zu identifizieren.
`authHash`	Hash-Code als Authentifizierungsmittel des Dozenten zur späteren Verwaltung der Veranstaltung. Zur Berechnung des Hash-Codes werden ein interner Schlüssel, der ADS-Name des Dozenten sowie die PIN herangezogen.

4.1.1.2 Anmeldung von Studenten zu Feedbacks

Durch diesen Aufruf melden sich Studenten mithilfe der veröffentlichten PIN zu der jeweiligen Veranstaltung an. Durch den Server wird ein Hash-Wert des jeweiligen Studenten ermittelt und zurückgereicht. Dieser Hash-Wert ist über mehrere Sessions und mehrere Veranstaltungen hinweg eindeutig dem Studenten zugewiesen. Der Hash-Wert wird für die Session (Vorlesung) durch den Server, zusammen mit der Vorlesungs-PIN, abgespeichert. Diese Information wird zur Authentifzierung bei der Abgabe von Feedback-Nachrichten genutzt.

Aufruf (POST) `/feedback/lecture/student/<pin>`	
Aufruf-Parameter in der URL	
`<pin>`	Vom Dozenten veröffentlichte 4-stellige PIN.
Authentifizierung Der Aufruf verlangt eine Authentifizierung mittels „HTTP Basic Authentication“. Da diese Daten vom Browser nur mit Base64 kodiert werden, sollte der Aufruf der URL nur über HTTPS erfolgen, um die Anmeldedaten zu schützen.	
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`UNAUTHORIZED` (401)	Es wurden keine oder falsche Anmeldeinformationen übergeben.
`NOT FOUND` (404)	Die PIN ist unbekannt. Nach fünf Fehlversuchen wird der Aufrufer bis zum Ende des Tages gesperrt.
Ergebnis (Body als Zeichenkette) Das Ergebnis ist JSON-Objekt mit folgendem Aufbau.	
`authHash`	Authentifizierungscode des Studenten für eine Veranstaltung. Der Code wird für die folgenden Feedback-Aufrufe benötigt.
`internalLectureName`	Interner Name der Veranstaltung, für die der `authCode` verwendet werden kann.

4.1.1.3 Status einer Vorlesung auslesen

Dieser Aufruf ermittelt zu einem Authorisierungs-Hash-Wert eines Dozenten oder zu einem Evaluierungs-Hash-Wert eines Studenten den Status der zugeordneten Veranstaltung.

Aufruf (GET)	
`/feedback/lecture/status/<hashcode>`	
Aufruf-Parameter in der URL	
`<hashcode>`	Authorisierungs-Hash-Wert des Dozenten aus dem Aufruf 4.1.1.1 oder Hash-Wert eines Studenten aus dem Aufruf 4.1.1.2.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Immer.
Ergebnis (Body als Zahl)	
Das Ergebnis ist eine Zahl mit dem Status der Veranstaltung. Dabei werden die folgenden Status-Codes verwendet:	
`0`	Es gibt keine Veranstaltung (mehr), der der im Aufruf übergebene Hash-Wert zugeordnet ist.
`1`	Die Veranstaltung befindet sich in der Live-Feedback-Phase.
`2`	Die Veranstaltung befindet sich in der Post-Feedback-Phase.

4.1.1.4 Anzahl aktiver Studierender auslesen

Dieser Aufruf ermittelt zu einem Authorisierungs-Hash-Wert die Anzahl an Studierenden, die sich zu dieser Veranstaltung angemeldet haben.

Aufruf (GET)	
`/feedback/studentcount/<hashcode>`	
Aufruf-Parameter in der URL	
`<hashcode>`	Authorisierungs-Hash-Wert des Dozenten aus dem Aufruf 4.1.1.1.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`NOT FOUND` (404)	Es gibt keine Veranstaltung zu dem angegebenen Hash-Wert.
Ergebnis (Body als Zahl)	
Das Ergebnis ist eine Zahl, die die Anzahl an Studierenden wiedergibt, die sich zu der Veranstaltung durch Aufruf 4.1.1.2 angemeldet haben.	

4.1.2 Phase der Live-Feedbacks

Mit dem folgenden Aufrufen werden Live-Rückmeldungen zu einer Veranstaltung verarbeitet.

4.1.2.1 Feedback-Namen auslesen

Der Aufruf liest alle erlaubten Namen mit deren IDs, Texten und Wertebereichen aus.

Aufruf (GET)
`/feedback/livenames`

Ergebnis (HTTP-Status-Code)	
`OK` (200)	Immer.
Ergebnis (Body als Array mit JSON-Objekten) Das Ergebnis ist ein Array mit JSON-Objekten des folgenden Aufbaus, wobei jeder Eintrag einen Namen beschreibt.	
`id`	ID des Namens.
`name`	Textuelle Beschreibung des Namens.
`minValue`	Kleinster erlaubter Wert als `double`-Zahl.
`maxValue`	Größter erlaubter Wert als `double`-Zahl.

4.1.2.2 Feedback schreiben

Der Aufruf schreibt eine Rückmeldung zu einer Veranstaltung mit Hilfe des Knowledge-Servers in die Datenbank. Der Hash-Wert (SHA-256) des Users, der den Eintrag geschrieben hat, wird automatisch aus den Anmeldedaten im Aufruf ermittelt.

Aufruf (POST) `/feedback/feedbackmsg/live/<studentPinHash>`	
Aufruf-Parameter in der URL	
`<studentPinHash>`	Hash-Wert aus dem Aufruf 4.1.1.2.
Aufruf-Parameter als JSON-Objekt (Body)	
`timestamp`	Zeitstempel der Rückmeldung.
`optionID`	ID der Option der Rückmeldung (Art der Rückmeldung).
`value`	Numerischer Wert der Rückmeldung als `double`-Zahl.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`BAD REQUEST` (400)	Wenn die Parameter im Aufruf-Objekt fehlerhaft aufgebaut oder unvollständig sind.
`UNAUTHORIZED` (401)	Es wurden keiner oder ein falscher Hash-Wert übergeben bzw. die Live-Phase ist beendet.
`NOT FOUND` (404)	Es gibt keine Veranstaltung zu dem Hash-Wert.
`INTERNAL SERVER ERROR` (500)	Wenn bei der Bearbeitung auf dem Server ein Fehler auftrat.
`GATEWAY TIMEOUT` (504)	Es konnte keine Verbindung mit dem Knowledge-Server hergestellt werden. Die Operation wurde nicht durchgeführt.
Ergebnis (Body als JSON-Objekt) Das Ergebnis ist eine Zeichenkette mit zusätzlichen Informationen zum Fehler.	
`statusCode`	Fehlercode: `0` Der Zeitstempel `timestamp` fehlt. `2` Es existiert keine Veranstaltung zu dem Hash-Wert. `5` Die ID der Option `optionID` ist außerhalb des Gültigkeitsbereichs oder fehlt. `6` Der Wert `value` fehlt. `7` Der Wert `value` liegt außerhalb des gültigen Bereichs.

4.1.2.3 Feedback schreiben (Handlungsanweisung erstellen)

Diese URL kann nur vom Knowledge-Server aufgerufen werden. Durch sie wird eine Handlungsanweisung erstellt, die in der Datenbank persistiert wird.

Aufruf (POST)	
`/feedback/instruction/post/<veranst>`	
Aufruf-Parameter in der URL	
`<veranst>`	Interner Name der Veranstaltung, zu der Feedback gesammelt werden soll (z.B. `IB 211`).
Aufruf-Parameter als JSON-Objekt (Body)	
`timestamp`	Zeitstempel der Rückmeldung.
`optionID`	ID der Option der Rückmeldung (Art der Rückmeldung).
`truthValue`	Numerischer Wert der Rückmeldung als `double`-Zahl.
`value`	Numerischer Wert der Rückmeldung als `double`-Zahl.
`gradient`	Numerischer Wert der Rückmeldung als `double`-Zahl.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`BAD REQUEST` (400)	Wenn die Parameter im Aufruf-Objekt fehlerhaft aufgebaut oder unvollständig sind.
`UNAUTHORIZED` (401)	Es wurden keiner oder ein falscher Hash-Wert übergeben bzw. die Live-Phase ist beendet.
`NOT FOUND` (404)	Es gibt keine Veranstaltung zu dem Hash-Wert.
`INTERNAL SERVER ERROR` (500)	Wenn bei der Bearbeitung auf dem Server ein Fehler auftrat.
Ergebnis (Body als JSON-Objekt)	
Das Ergebnis ist eine Zeichenkette mit zusätzlichen Informationen zum Fehler.	
`statusCode`	Fehlercode: `0` Der Zeitstempel `timestamp` fehlt. `2` Es existiert keine Veranstaltung zu dem Hash-Wert. `5` Die ID der Option `optionID` ist außerhalb des Gültigkeitsbereichs oder fehlt. `6` Der Wert `value` fehlt. `7` Der Wert `value` liegt außerhalb des gültigen Bereichs.

4.1.2.4 Feedbacks auslesen (Handlungsanweisungen abholen)

Der Aufruf liest die Rückmeldungen zu einer Veranstaltung aus.

Aufruf (GET)	
`/feedback/instruction/<pin>`	
Aufruf-Parameter in der URL	
`<pin>`	PIN, mit der die Veranstaltung evaluiert wird, aus dem Aufruf 4.1.1.1.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`BAD REQUEST` (400)	Wenn die URL-Parameter fehlerhaft aufgebaut sind.

NOT FOUND (404)	Es gibt keine Veranstaltung zu dem Hash-Wert.
Ergebnis (Body als Array mit JSON-Objekten) Das Ergebnis ist ein Array mit JSON-Objekten des folgenden Aufbaus:	
timestamp	Zeitstempel der Handlungsanweisung.
lectureInternalName	Interner Name der Veranstaltung, zu der diese Rückmeldung gehört.
optionID	ID der Feedbackoption (Art der Rückmeldung).
truthValue	Fuzzy-Relevanzwert der jeweiligen Feedbackoption im Wertebereich [0,1] als double-Zahl.
value	Numerischer Wert der Rückmeldung als double-Zahl.
gradient	Numerischer Wert der Rückmeldung als double-Zahl.

4.1.3 Phase der Live-Feedbacks beenden

Der beendet die Live-Feedback-Phase und erlaubt Post-Feedbacks.

Aufruf (DELETE) /feedback/lecture/live/<authHash>	
Aufruf-Parameter in der URL	
<authHash>	Hash-Wert des Dozenten aus dem Aufruf 4.1.1.1.
Ergebnis (HTTP-Status-Code)	
OK (200)	Immmer (auch wenn der Hash-Wert zu keiner Veranstaltung gehört).
GATEWAY TIMEOUT (504)	Es konnte keine Verbindung mit dem Knowledge-Server hergestellt werden. Die Operation wurde nicht durchgeführt.
Ergebnis Das Ergebnis ist ein leerer Body.	

4.1.4 Phase der Post-Feedbacks

Mit dem folgenden Aufrufen werden Post-Rückmeldungen zu einer Veranstaltung verarbeitet.

4.1.4.1 Erlaubte Feedback-Titel auslesen

Der Aufruf liest alle erlaubten Titel mit deren IDs, Texten und Wertebereichen aus.

Aufruf (GET) /feedback/posttitles	
Ergebnis (HTTP-Status-Code)	
OK (200)	Immer.
Ergebnis (Body als Array mit JSON-Objekten) Das Ergebnis ist ein Array mit JSON-Objekten des folgenden Aufbaus, wobei jeder Eintrag einen Titel beschreibt.	
id	ID des Titels.
title	Textuelle Beschreibung des Titels.
minValue	Kleinster erlaubter Wert.

`maxValue`	Größter erlaubter Wert.

4.1.4.2 Feedback schreiben

Der Aufruf schreibt eine Rückmeldung zu einer Veranstaltung in die Datenbank. Der Hash-Wert (SHA-256) des Users, der den Eintrag geschrieben hat, wird automatisch aus den Anmeldedaten im Aufruf ermittelt.

Aufruf (POST) `/feedback/feedbackmsg/post/<studentPinHash>`	
Aufruf-Parameter in der URL	
`<studentPinHash>`	Hash-Wert des Studenten aus dem Aufruf 4.1.1.2.
Aufruf-Parameter als JSON-Objekt (Body)	
`timestamp`	Zeitstempel der Rückmeldung.
`optionID`	ID der Option der Rückmeldung (Art der Rückmeldung).
`value`	Numerischer Wert der Rückmeldung als `double`-Zahl.
`comment`	Kommentar zur Rückmeldung, die Länge wird auf 256 Zeichen gekürzt. Der Kommentar darf fehlen.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`BAD REQUEST` (400)	Wenn die Parameter im Aufruf-Objekt fehlerhaft aufgebaut oder unvollständig sind.
`UNAUTHORIZED` (401)	Wenn der Hash-Wert nicht zu einem Studenten gehört.
`NOT FOUND` (404)	Es gibt keine Veranstaltung zu dem Hash-Wert.
`INTERNAL SERVER ERROR` (500)	Wenn bei der Bearbeitung auf dem Server ein Fehler auftrat.

GATEWAY TIMEOUT (504)	Es konnte keine Verbindung mit dem Knowledge-Server hergestellt werden. Die Operation wurde nicht durchgeführt.
Ergebnis (Body als JSON-Objekt) Das Ergebnis ist eine Zeichenkette mit zusätzlichen Informationen zum Fehler.	
statusCode	Fehlercode: 0 Der Zeitstempel timestamp fehlt. 2 Es existiert keine Veranstaltung zu dem Hash-Wert. 5 Die ID der Option optionID ist außerhalb des Gültigkeitsbereichs oder fehlt. 6 Der Wert value fehlt. 7 Der Wert value liegt außerhalb des gültigen Bereichs.
description	Kurze textuelle Beschreibung des Fehlers.

4.1.4.3 Feedbacks auslesen

Der Aufruf liest Rückmeldungen zu einer Veranstaltung in einem Zeitraum aus.

Aufruf (GET) /feedback/feedbackmsg/post/<start>/<authHash>	
Aufruf-Parameter in der URL	
<start>	Startzeitpunkt (inkl.), ab dem Feedbacks berücksichtigt werden sollen.
<authHash>	Hash-Wert des Dozenten aus dem Aufruf 4.1.1.1.
Ergebnis (HTTP-Status-Code)	
OK (200)	Im Erfolgsfall.
BAD REQUEST (400)	Wenn die URL-Parameter fehlerhaft aufgebaut sind.
UNAUTHORIZED (401)	Der Hash-Wert gehört zu keinem Dozenten.
NOT FOUND (404)	Es gibt keine Veranstaltung zu dem Hash-Wert.
Ergebnis (Body als Array mit JSON-Objekten) Das Ergebnis ist ein Array mit JSON-Objekten des folgenden Aufbaus, wobei jeder Eintrag ein Feedback beschreibt.	
timestamp	Zeitstempel, den die Rückmeldung beim Ablegen auf dem Server hatte.
lectureInternalName	Interner Name der Veranstaltung, zu der diese Rückmeldung gehört.
optionID	ID der Option der Rückmeldung (Art der Rückmeldung).
option	Vollständiger Name der Option der Rückmeldung.
truthValue	Numerischer Wert der Rückmeldung als double-Zahl.
comment	Zusätzlicher Kommentar zur Rückmeldung.

4.1.5 Aufräumphase

In dieser Phase können Feedbacks wieder gelöscht werden.

4.1.5.1 Live-Feedbacks löschen

Der Aufruf löscht alle Rückmeldungen zu einer Veranstaltung in einem Zeitraum aus der Datenbank.

Aufruf (DELETE)	
`/feedback/instruction/<authHash>`	
Aufruf-Parameter in der URL	
`<authHash>`	Hash-Wert des Dozenten aus dem Aufruf 4.1.1.1.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`BAD REQUEST` (400)	Wenn die URL-Parameter fehlerhaft aufgebaut sind.
`NOT FOUND` (404)	Es gibt keine Veranstaltung zu dem Hash-Wert.
Ergebnis	
Das Ergebnis ist ein leerer Body.	

4.1.5.2 Post-Feedbacks löschen

Der Aufruf löscht Post-Rückmeldungen zu einer Veranstaltung in einem Zeitraum aus der Datenbank.

Aufruf (DELETE)	
`/feedback/feedbackmsg/post/<authHash>/`	
Aufruf-Parameter in der URL	
`<authHash>`	Hash-Wert des Dozenten aus dem Aufruf 4.1.1.1.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`BAD REQUEST` (400)	Wenn die URL-Parameter fehlerhaft aufgebaut sind.
`NOT FOUND` (404)	Es gibt keine Veranstaltung zu dem Hash-Wert.
Ergebnis	
Das Ergebnis ist ein leerer Body.	

4.1.5.3 Veranstaltung löschen

Der Aufruf löscht alle Rückmeldungen (Live und Post) zu einer Veranstaltung sowie die Veranstaltung selbst.

Aufruf (DELETE)	
`/feedback/lecture/<authHash>/`	
Aufruf-Parameter in der URL	
`<authHash>`	Hash-Wert des Dozenten aus dem Aufruf 4.1.1.1.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Immer.
Ergebnis	
Das Ergebnis ist ein leerer Body.	

4.1.6 Ad-Hoc-Fragen

Diese Aufrufe erweitern die Smart-Classroom Funktionalität um die Möglichkeit für den Dozenten, während der Vorlesung Ad-Hoc Fragen zu stellen, auf die durch Eingabe eines einzelnen Wortes seitens der Studierenden geantwortet wird. Nachdem der Dozent die Fragerunde geschlossen hat, werden die Antworten auf dem Knowledgeserver (durch Text-Mining-Methoden) geclustert und können anschließend auf Dozentenseite visualisiert werden. Ad-hoc-Fragen sind zwischen den Zeitpunkten der Vorbereitungs- 4.1.1 und Aufräumphase 4.1.5.3 erlaubt.

4.1.6.1 Frage erstellen

Durch diesen Aufruf wird angegeben, dass der Dozent eine Ad-Hoc Frage gestellt hat und nun Antworten durch Studenten abgegeben werden können. Die Authentifizierung erfolgt hier durch den Dozenten-Hash aus dem Aufruf 4.1.1.1.

Aufruf (POST)	
`/feedback/ahq/question/<authHash>/`	
Aufruf-Parameter in der URL	
`<authHash>`	Hash-Wert des Dozenten aus dem Aufruf 4.1.1.1.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`NOT FOUND` (404)	Es gibt keine Veranstaltung zu dem Hash-Wert.
Ergebnis	
Das Ergebnis ist ein leerer Body.	

4.1.6.2 Frage abholen

Dieser Aufruf dient als Workaround und soll langfristig durch eine PUSH-Lösung ersetzt werden. Die Studentenapplikation überprüft hiermit sequentiell, ob für die aktuelle Vorlesung eine Ad-Hoc Frage gestellt wurde, um gegebenenfalls eine Antwort vom Studenten einzufordern. Wird die URL dagegen mit einem Dozenten-Hash aufgerufen, dann wird geprüft, ob der Dozent für diese Veranstaltung eine Frage zur Beantwortung freigegeben hat.

Aufruf (GET)	
`/feedback/ahq/question/<hash>/`	
Aufruf-Parameter in der URL	
`<hash>`	Hash-Wert des Studenten aus dem Aufruf 4.1.1.2 oder Dozenten-Hash aus dem Aufruf 4.1.1.1
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`NOT FOUND` (404)	Es gibt keine Veranstaltung zu dem Hash-Wert.
Ergebnis	
Das Ergebnis ist ein Text-Body mit den folgenden beiden Antwortmöglichkeiten als Text.	
`true`	Der Student darf eine Antwort schreiben (bei Aufruf mit Studenten-Hash) bzw. der Dozent hat eine Frage zur Beantwortung freigegeben (bei Aufruf mit Dozenten-Hash).

`false`	Der Student kann keine Antwort schreiben (bei Aufruf mit Studenten-Hash) bzw. der Dozent hat keine Frage zur Beantwortung freigegeben (bei Aufruf mit Dozenten-Hash).

4.1.6.3 Antwort erstellen

Durch diesen Aufruf können Antworten für die derzeit offene Frage gepostet werden. Antworten werden per REST Aufruf an den Knowledge-Server weitergeleitet und anschließend verworfen. Pro Student kann nur eine Antwort gepostet werden. Eine Antwort besteht aus einem Wort mit einer maximalen Länge von 100 Zeichen.

Aufruf (POST) `/feedback/ahq/answer/<studentPinHash>/`	
Aufruf-Parameter in der URL	
`<studentPinHash>`	Hash-Wert des Studenten aus dem Aufruf 4.1.1.2.
Aufruf-Parameter als Text (Body)	
	Text mit der Antwort
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`NOT FOUND` (404)	Es gibt keine Veranstaltung zu dem Hash-Wert.
`GATEWAY TIMEOUT` (504)	Es konnte keine Verbindung mit dem Knowledge-Server hergestellt werden. Die Operation wurde nicht durchgeführt.
Ergebnis Das Ergebnis ist ein leerer Body.	

4.1.6.4 Antworten anfordern

Durch diesen Aufruf fordert der Dozent die (vorverarbeiteten) Antworten an.

Aufruf (GET) `/feedback/ahq/answerlist/<authHash>/`	
Aufruf-Parameter in der URL	
`<authHash>`	Hash-Wert des Dozenten aus dem Aufruf 4.1.1.1.
Ergebnis (HTTP-Status-Code)	
`OK` (200)	Im Erfolgsfall.
`NOT FOUND` (404)	Es gibt keine Veranstaltung zu dem Hash-Wert.
`INTERNAL_SERVER_ERROR` (500)	Bei einem sonstigen internen Fehler.
`GATEWAY TIMEOUT` (504)	Es konnte keine Verbindung mit dem Knowledge-Server hergestellt werden. Die Operation wurde nicht durchgeführt.
Ergebnis Das Ergebnis ist ein Array mit JSON-Objekten des folgenden Aufbaus, wobei jeder Eintrag eine Antwort beschreibt.	
`text`	Textantwort
`count`	Wie oft wurde diese Antwort gegeben?

clusterID	ID des Wortclusters, in den die Antwort eingeordnet wurde.

4.1.6.5 Anzahl an Antworten auslesen

Durch diesen Aufruf wird ausgelesen, wieviele Antworten auf die aktuelle Frage einer Veranstaltung gegeben wurden.

Aufruf (GET)	
/feedback/ahq/answercount/<hash>/	
Aufruf-Parameter in der URL	
<hash>	Hash-Wert des Dozenten aus dem Aufruf 4.1.1.1 oder Studenten-PIN-Hash-Wert aus dem Aufruf 4.1.1.2.
Ergebnis (HTTP-Status-Code)	
OK (200)	Im Erfolgsfall.
NOT FOUND (404)	Es gibt keine Veranstaltung zu dem Hash-Wert.
Ergebnis	
Das Ergebnis ist eine Zahl, die die Anzahl an Antworten auf die aktuelle Vorlesungsfrage wiedergibt.	

4.1.6.6 Frage löschen

Durch diesen Aufruf sperrt der Dozent eine Frage, so dass keine weiteren Antworten gepostet werden können. Der Aufruf darf mehrfach erfolgen. Ist eine Frage bereits gesperrt, so wird das nicht als Fehler behandelt.

Aufruf (GET)	
/feedback/ahq/question/<authHash>/	
Aufruf-Parameter in der URL	
<authHash>	Hash-Wert des Dozenten aus dem Aufruf 4.1.1.1.
Ergebnis (HTTP-Status-Code)	
OK (200)	Im Erfolgsfall.
NOT FOUND (404)	Es gibt keine Veranstaltung zu dem Hash-Wert.
INTERNAL_SERVER_ERROR (500)	Bei einem sonstigen internen Fehler.
Ergebnis	
Außer dem Status-Code wird keine weitere Information zurückgegeben.	