**Endpoints**

**Project**

Retrieve: [const url = "https://projects.sbw.media/Project"](https://projects.sbw.media/Project)

Update: const url = "https://projects.sbw.media/Project/{$projectId}"

Note: views are not updatable

key: is auto increment: ja

Fields:

project: {

ID: 0, // type: int

ParentID: 0, // type: int

Number: "", // type: string

Name: "", // type: string

Description: "", // type: string

TypeID: 0, // type: int

CustomerID: 0, // type: int

Coach: "", // type: string

Status: 0, // type: int

}

**Student**

Retrieve: [const url = "https://projects.sbw.media/Student"](https://projects.sbw.media/Student)

Update: const url = "https://projects.sbw.media/Student/{$studentId}"

Note: views are not updatable

key: is auto increment: ja

Fields:

student: {

ID: 0, // type: int

Name: "", // type: string

Firstname: "", // type: string

Year: 0, // type: int

Fullname: "", // type: string

}

**Teacher**

Retrieve: [const url = "https://projects.sbw.media/Teacher"](https://projects.sbw.media/Teacher)

Update: const url = "https://projects.sbw.media/Teacher/{$teacherId}"

Note: views are not updatable

key: is auto increment: nein

Fields:

teacher: {

Abbr: "", // type: string

Name: "", // type: string

Firstname: "", // type: string

FullName: "", // type: string

}

**Timesheet**

Retrieve: [const url = "https://projects.sbw.media/Timesheet"](https://projects.sbw.media/Timesheet)

Update: const url = "https://projects.sbw.media/Timesheet/{$timesheetId}"

Note: views are not updatable

key: is auto increment: ja

Fields:

timesheet: {

ID: 0, // type: int

ProjectID: 0, // type: int

StudentID: 0, // type: int

Date: "", // type: Date

Minutes: 0, // type: int

Approved: 0, // type: int

}

**TimesheetView**

Retrieve: [const url = "https://projects.sbw.media/TimesheetView"](https://projects.sbw.media/TimesheetView)

Update: const url = "https://projects.sbw.media/TimesheetView/{$timesheetViewId}"

Note: views are not updatable

key: is auto increment: nein

Fields:

timesheetView: {

ID: 0, // type: int

StudentID: 0, // type: int

Fullname: "", // type: string

ProjectName: "", // type: string

Number: "", // type: string

Date: "", // type: Date

Minutes: 0, // type: int

Approved: 0, // type: int

}

**Suchen**

**By ID**

In alle endpoints kann gesucht werden und zwar nach ID: Beispiel "https://projects.sbw.media$/34" sucht nach endpoint mit id gleich '34'. Resultat bei gefunden 200, bei nicht gefunden 404. Bei Erfolg ist das Resultat in JSON Objekt

**By Example**

Auf alle andere Attribute kann gesucht werden mit query-string. Beispiel "https://projects.sbw.media$/?title=hello". Gross-/Kleinschreiben bei Key (title) beachten, beim Wert (helloe) spielt is keine Rolle. Auch Jokerzeichen sind möglich "https://projects.sbw.media$/?title=h\*" sucht alle endpoints mit Feld title beginnend mit 'h'. Bei Erfolg (200) ist das Resultat immer ein JSON Array von Objekte. Resultat bei Key nicht gefunden (missspelled): 400 Bad Request, bei nicht gefunden 204 no content

**Methoden**

Die folgenden Methoden sind zulässig:

**GET**

Daten für Endpunkt abrufen

async function get() {

const url = 'https://projects.sbw.media/endpoint';

const response = await fetch(url);

if(!response.ok) {

throw new Error('HTTP error! status: ' + response.status');

}

const data = await response.json();

return data.resources;

}

**POST**

Erstellen Sie eine neue Resource. Als Payload ein Object mit Properties.

async function post(payload) {

const url = 'https://projects.sbw.media/endpoint';

const response = await fetch(url, {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(payload)

});

if(!response.ok) {

throw new Error('HTTP error! status: ' + response.status');

}

const data = await response.json();

return data;

}

**PUT**

Vorhandenen Eintrag für Endpunkt aktualisieren. Der mit ID angegebene Resource wird aktualisiert. Als Payload ein Object mit Properties.

async function put(id, payload) {

const url = 'https://projects.sbw.media/endpoint/' + id;

const response = await fetch(url, {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(payload)

});

if(!response.ok) {

throw new Error('HTTP error! status: ' + response.status');

}

const data = await response.json();

return data;

}

**LÖSCHEN**

Vorhandenen Eintrag für Endpunkt entfernen. Das mit ID angegebene Objekt wird entfernt.

async function delete(id) {

const url = 'https://projects.sbw.media/endpoint/' + id;

const response = await fetch(url,

{

method: 'DELETE'

});

if(!response.ok) {

throw new Error('HTTP error! status: ' + response.status');

}

const data = await response.json();

return data;

}

**Statuscodes**

Der Dienst folgt REST-Standards mit den folgenden HTTP\_Statuscodes:

**200**

Erfolg

**204**

Erfolg, aber keine Ergebnisse.

**400**

Ungültige Anfrage. Der Text enthält Details.

**404**

Endpunkt nicht gefunden. Inhalt leer

**500**

Interner Serverfehler. Der Text enthält Details.

CC TheKing2