API - Dokumentation

Gruppe: Bottlenecks
Fach: Praktikum Softwareentwicklung
Hochschule Hof, Sommersemester 2022

Verantwortlich: Johannes Matus, Sebastian Bär

Vorbemerkungen

orange → Parameter muss mit angegeben werden
blau → Parameter ist optional

Bei jeder Anfrage muss im Header der Anfrage Accept : application/json angegeben werden.

1.Authentifizierung und Autorisierung

Die Authentifizierung findet mittels API - Tokens statt. Das Token, das beim Login erhalten wird, muss bei jeder nachfolgenden Anfrage als Bearer-Token in der Anfrage übermittelt werden um den User authentifizieren zu können.
Ohne manuellen Logout sind die Token für maximal 10 Stunden gültig. Nach Ablauf der 10 Stunden muss ein neues Token angefragt werden (erneuter Login).

1.1 Registrierung

Request Typ	POST

<table>
<tr><td>URL</td><td>/api/register</td></tr>
<tr><td>Request Body Beispiel</td><td>

```
{
    "email":"peter.pan@gmail.com",
    "username": "pete2345",
    "first_name": "Peter",
    "last_name": "Pan",
    "password": "password",
    "password_confirmation":"password"
}
```

</td></tr>
<tr><td>Status Codes</td><td>201
422</td></tr>
<tr><td>Beispielantwort</td><td>

```
Statuscode: 201
{
    "success": true,
    "user": {
        "first_name": "Peter",
        "last_name": "Pan",
        "username": "pete2345",
        "email": "peter.pan@gmail.com",
        "id": 6
    }
}

------------------------------------------------------------
--
Statuscode: 422 Beispiel
{
    "message": "The username has already been taken. (and 1 more
error)",
    "errors": {
        "username": [
            "The username has already been taken."
        ],
        "email": [
            "The email field is required."
        ]
    }
}
```

</td></tr>
<tr><td>Parameter</td><td>
<table>
<tr><td>username</td><td></td><td>string</td><td>Einzigartiger Username</td></tr>
<tr><td>first_name</td><td></td><td>string</td><td>Vorname</td></tr>
</table>
</td></tr>
</table>

	<table><tr><td>last_name</td><td></td><td>string</td><td>Nachname</td></tr><tr><td>email</td><td></td><td>string</td><td>E-Mail im richtigen Format</td></tr><tr><td>password</td><td></td><td>string</td><td>Passwort</td></tr><tr><td>password_confirmation</td><td></td><td>string</td><td>Wiederholung des Passworts</td></tr></table>

1.2 Login

Bei erfolgreichem Login werden die Userdaten sowie ein Bearer-Token zurückgeliefert, dieses Token muss in folgenden Anfragen im Header als solchen gekennzeichnet und mitgeschickt werden um die Anfrage zu verifizieren und dem User zuordnen zu können. Ein User kann sich mit seiner E-Mail Adresse und seinem Passwort einloggen.

Request Typ	POST
URL	/api/login

Request Body Beispiel	{
	"email":"ingo@ex.com",
	"password":"password"
	}

Status Codes	200 401 422

Beispielantwort	Statuscode: 200
	{
	"success": true,
	"username": {
	"id": 1,
	"username": "fhartmann",
	"email": "ingo@ex.com",
	"first_name": "Katlyn",
	"last_name": "Gislason"
	},
	"bearer_token": "8|lxvqbrgBU48KbRQMfHddE4zCxqP68IhTbjNQNyR3"
	}
	---------- 	
	--
	Statuscode: 401

<table>
<tr><td></td><td>

```
{
    "message": "Invalid credentials"
}
----------------------------------------------------------------
--
Statuscode: 422
{
    "message": "The email must be a valid email address. (and 1
more error)",
    "errors": {
        "email": [
            "The email must be a valid email address."
        ],
        "password": [
            "The password field is required."
        ]
    }
}
```

</td></tr>
<tr><td>Parameter</td><td>

email		string	E-Mail im richtigen Format
password		string	Passwort

</td></tr>
</table>

1.3 Logout

Beim ausloggen muss das Bearer Token im Header der Anfrage übergeben werden um die Anfrage einem Nutzer zuordnen zu können, andernfalls kann der User nicht ausgeloggt werden und das Token ist nach wie vor gültig. (Status Code: 401, siehe Beispielantwort)

Request Typ	POST
URL	/api/logout
Request Body Beispiel	-
Status Codes	200 401
Beispielantwort	Statuscode: 200 { "message": "Successfully logged out"

	}
	---------- 	
	--
	Statuscode: 401
	{
	"message": "Unauthenticated."
	}

Parameter	-

1.4 Autorisierung

Für alle Aktionen ist das Projekt als Rahmen der Aktionen anzusehen.
Der Ersteller eines Projekts kann dieses vollumfänglich verwalten und hat sozusagen Adminrechte.
Mitgliedern eines Projekts können bestimmte Rechte vom Administrator gewährt werden. Allerdings kann ein Projektmitglied, selbst mit vollen Rechten, nie den selben Rechteumfang erhalten wie der Administrator.

Folgende Aktionen können lediglich vom Ersteller (Administrator) des Projekts ausgeführt werden:

- Das Projekt bearbeiten/abschließen
- Das Projekt löschen
- Mitglieder zum Projekt hinzufügen/entfernen
- Rechte der Projektmitglieder aktualisieren
- Räume erstellen/entfernen/bearbeiten
- Ankündigungen erstellen/löschen
- Tags löschen

Neben den aufgeführten Aktionen kann der Besitzer jede Aktion ausführen die Nutzer ebenfalls ausführen können, wenn sie vollen Rechteumfang erhalten.

Folgende Rechte können für Gruppenmitglieder eines Projektes erteilt werden:

can_create_tasks	can_edit_tasks	can_create_tags
Gibt an ob der Nutzer Tasks erstellen kann. Sollte der Nutzer dies nicht dürfen kann er anderen Tasks lediglich zugewiesen werden und Aktionen	Grundsätzlich kann jeder Nutzer Tasks vollumfänglich bearbeiten die er selbst erstellt hat (und diese auch löschen). Bei Auswahl von	Gibt an ob der Nutzer Projektweit Tags erstellen kann. Die erstellten Tags können dann Tasks zugewiesen werden (von jedem Benutzer

ausführen die im Bearbeitungsprozess notwendig sind (Status ändern, Task abschließen und beim Abschluss kommentieren).	“can_edit_tasks” kann der Nutzer dies allerdings für jede Task innerhalb des Projekts (Löschen ausgeschlossen).	der Tasks erstellen kann).

Räume buchen kann jedes Projektmitglied, sofern der gewünschte Zeitslot frei ist.

2.Projekte

2.1 Projekt erstellen

Beim Erstellen eines Projektes wird das Projekt automatisch dem User der die Anfrage stellt zugewiesen.

Request Typ	POST
URL	/api/project/

Request Body Beispiel	<pre>{
	"title":"Project One",
	"description":"This is the description of the project",
	"due_date": "2022-06-22"
	}</pre>

Status Codes	201 500 422

Beispielantwort	<pre>Statuscode: 201
	{
	"success": true,
	"project": {
	"title": "Project One",
	"description": "This is the description of the project",
	"due_date": "2022-06-22",
	"creator_user_id": 1,
	"updated_at": "2022-05-20T14:50:13.000000Z",
	"created_at": "2022-05-20T14:50:13.000000Z",
	"id": 8
	}
	}</pre>

Parameter			
	title	string	Max-Länge: 255
	description	string	Max-Länge: 3000
	due_date	date	Format: YYYY-MM-DD

2.2 Projekt holen

Die Antwort beinhaltet alle Informationen zum angefragten Projekt (id, Titel, Beschreibung, … siehe Beispielantwort) inklusive Informationen zum Ersteller des Projekts.

Request Typ	GET
URL	/api/project/{id}
Request Body Beispiel	-
Status Codes	200 404 500

Beispielantwort	Statuscode: 200
	{
	"success": true,
	"project": {
	"id": 1,
	"title": "Titel",
	"description": "Quaerat ea fugit numquam officiis.",
	"creator_user_id": 1,
	"due_date": "2022-07-03",
	"completion_date": "2022-05-20",
	"created_at": "2022-05-16T12:46:36.000000Z",
	"updated_at": "2022-05-20T06:36:47.000000Z",
	"creator": {
	"id": 1,
	"username": "addison.will",
	"email": "marietta00@example.net",
	"first_name": "Damien",
	"last_name": "Harber"
	}
	}
	}

Parameter	-

2.3 Projekt löschen

Wird ein Projekt gelöscht werden auch alle damit verbundenen Daten (Mitglieder, Tasks, Tags, Räume, Buchungen) gelöscht.

Request Typ	DELETE
URL	/api/project/{id}
Request Body Beispiel	-
Status Codes	200 404 500

Beispielantwort	Statuscode: 201
	{
	"success": true,
	}
	---------- 	
	--
	Statuscode: 404
	{
	"success": false,
	"message": "Project not found"
	}

Parameter	-

2.4 Projekt aktualisieren

Der Endpoint kann genutzt werden um den Titel, die Beschreibung oder das Datum, an dem das Projekt abgeschlossen sein soll zu aktualisieren. Alle Parameter sind optional und es wird nur das aktualisiert, was im Body der Anfrage angegeben ist.

Request Typ	PUT
URL	/api/project/

<table>
<tr><td>Request Body Beispiel</td><td>

```
{
    "title": "New Title"
}
```

```
{
    "due_date" : "2022-31-12"
}
```

</td></tr>
<tr><td>Status Codes</td><td>200
422
500</td></tr>
<tr><td>Beispielantwort</td><td>

```
Statuscode: 200
{
    "success": true,
    "project": {
        "id": 1,
        "title": "New Title",
        "description": "This is the  description of the
project",
        "creator_user_id": 1,
        "due_date": "2022-06-22",
        "completion_date": null,
        "created_at": "2022-05-20T14:50:13.000000Z",
        "updated_at": "2022-05-20T15:20:35.000000Z"
    }
}
```

```
Statuscode: 422
{
    "errors": {
        "due_date": [
            "The due date is not a valid date."
        ]
    }
}
```

</td></tr>
<tr><td>Parameter</td><td>

title		string	Max-Länge: 255
description		string	Max-Länge: 3000
due_date		date	Format: YYYY-MM-DD

</td></tr>
</table>

2.5 Projekt abschließen

Der Endpoint kann genutzt werden um ein Projekt abzuschließen. Dabei wird das Abschlussdatum auf das Datum des Tages gesetzt, an dem die Request verarbeitet wird.

Request Typ	PUT
URL	/api/project/{id}/complete
Request Body Beispiel	-
Status Codes	200 404

Beispielantwort	Statuscode: 200
	{
	"success": true,
	"project": {
	"id": 1,
	"title": "New Title",
	"description": "This is the description of the project",
	"creator_user_id": 1,
	"due_date": "2022-07-12",
	"completion_date": "2022-05-20",
	"created_at": "2022-05-20T14:50:13.000000Z",
	"updated_at": "2022-05-20T15:29:19.000000Z"
	}
	}

Parameter	-

2.6 Alle Tags eines Projektes erhalten

Request Typ	GET
URL	/api/project/{id}/tags
Request Body Beispiel	-
Status Codes	200 404

	500

Beispielantwort	Statuscode: 200
	{
	"success": true,
	"project_id": 1,
	"tags": []
	}
	---------- 	
	--
	Statuscode: 200
	{
	"success": true,
	"project_id": 1,
	"tags": [
	{
	"id": 9,
	"title": "Design",
	"description": null,
	"project_id": 1,
	"created_at": "2022-05-21T08:16:32.000000Z"
	}
	]
	}

Parameter	-

2.7 Projektübersicht erhalten

Die Projektübersicht liefert eine detaillierte Auflistung der Anzahl der verschiedenen Aufgaben zum jeweiligen Status, sowie die Projektmitglieder samt Rechten, eine Prozentangabe der abgeschlossenen Aufgaben, sowie Titel, Beschreibung Fälligkeitsdatum und Abschlussdatum. Falls das Projekt noch nicht abgeschlossen wurde, ist der Wert des *completed_date* Feldes *NULL*.

Request Typ	GET
URL	/api/project/{id}/overview
Request Body Beispiel	-
Status Codes	200 404 500

Beispielantwort	Statuscode: 200
	{
	"success": **true**,
	"project_overview": {
	"title": "Rerum dolorum minima est eveniet.",
	"description": "Officiis quo corporis autem.",
	"due_date": "2023-04-17",
	"completed_date": **null**,
	"completed_tasks": 0,
	"failed_tasks": 1,
	"paused_tasks": 2,
	"in-progress_tasks": 2,
	"progress_percentage": 0,
	"members": [
	{
	"id": 49,
	"username": "alysa18",
	"email": "iaufderhar@example.com",
	"first_name": "Velva",
	"last_name": "Kessler",
	"pivot": {
	"project_id": 2,
	"user_id": 49,
	"can_create_tasks": 1,
	"can_assign_tasks": 0,
	"can_create_tags": 1
	}
	}
	]
	}
	}
	---------- 	
	--
	Statuscode: 404
	{
	"success": **false**,
	"message": "Project not found"
	}

Parameter	-

3.Projektmitglieder

3.1 Mitglied erstellen

Der Endpoint kann genutzt werden um Projektmitglieder zu erstellen und die Rechte dieser festzulegen. Das Projektmitglied wird über den Usernamen identifiziert und dem Projekt mit der angegebenen id zugewiesen.
Mitglieder können nur vom Besitzer eines Projekts hinzugefügt werden.

Request Typ	POST
URL	/api/project/add-new-user

Request Body Beispiel	{
	"username":"marquise17",
	"project_id": 1,
	"can_create_tasks": 1,
	"can_edit_tasks": 0,
	"can_create_tags":1
	}

Status Codes	200 422 403 500

Beispielantwort	Statuscode: 200
	{
	"success": true,
	"project-member": {
	"project_id": 1,
	"can_create_tasks": 1,
	"can_edit_tasks": 0,
	"can_create_tags": 1,
	"user_id": 2,
	"updated_at": "2022-05-20T15:50:07.000000Z",
	"created_at": "2022-05-20T15:50:07.000000Z",
	"id": 6
	}
	}
	---------- 	
	--
	{
	"errors": {
	"username": [
	"The selected username is invalid."

<table>
<tr><td></td><td>

```
        ]
    }
}
→ Username existiert nicht
```

</td></tr>
<tr><td>Parameter</td><td>

username		string	vorhandener Username
project_id		id	Id eines Projekts
can_create_tasks		bool/int	
can_assign_tasks		bool/int	
can_edit_tasks		bool/int	
can_create_tags		bool/int	

</td></tr>
</table>

3.2 Rechte ändern

Der Endpoint kann genutzt werden um Rechte von Projektmitgliedern anzupassen.
Hinweis: Die id in der URL bezieht sich auf die id des Mitglieds (project_user Tabelle) und ist NICHT dieselbe id, wie die des Users.
Die Rechte eines Mitglieds können nur durch den Besitzer des Projekts geändert werden.

<table>
<tr><td>Request Typ</td><td>POST</td></tr>
<tr><td>URL</td><td>/api/project-member/{id}</td></tr>
<tr><td>Request Body Beispiel</td><td>

```
{
    "can_create_tasks": 1
}
```

</td></tr>
<tr><td>Status Codes</td><td>200
422
403
500</td></tr>
<tr><td>Beispielantwort</td><td>

```
Statuscode: 200
{
    "success": true,
    "member": {
        "id": 1,
        "created_at": null,
        "updated_at": "2022-05-20T15:59:00.000000Z",
```

</td></tr>
</table>

<table>
<tr><td></td><td><pre>
        "user_id": 1,
        "project_id": 2,
        "can_create_tasks": 0,
        "can_edit_tasks": 0,
        "can_assign_tasks": 1,
        "can_create_tags": 0
    }
}
</pre></td></tr>
<tr><td>Parameter</td><td>
<table>
<tr><td>can_create_tasks</td><td></td><td>bool/int</td><td></td></tr>
<tr><td>can_edit_tasks</td><td></td><td>bool/int</td><td></td></tr>
<tr><td>can_create_tags</td><td></td><td>bool/int</td><td></td></tr>
</table>
</td></tr>
</table>

3.3 Mitglied entfernen

Der Besitzer eines Projektes kann Mitglieder über diesen Endpoint manuell entfernen. Sollte der Anfragende Nutzer nicht der Besitzer des Projektes, von dem das Mitglied (id) ein Teil ist, sein, wird die Anfrage nicht autorisiert und verworfen (Status - 403).

Request Typ	**DELETE**
URL	/api/project-member/{id}
Request Body Beispiel	-
Status Codes	200 404 403 500

Beispielantwort	Statuscode: 200
	{
	"success": true,
	"message": "Deleted."
	}

Parameter	-

3.4 Alle Mitglieder eines Projekts erhalten

Gibt alle Mitglieder des Projekts inklusive deren Berechtigungen und Nutzerinformationen zurück. Sollte die Anfrage von einem User stammen, der weder Projektmitglied noch Besitzer des Projektes ist, wird die Anfrage nicht autorisiert (Status - 403).

Request Typ	**GET**
URL	/api/project/{id}/members
Request Body Beispiel	-
Status Codes	200 403 404 500

Beispielantwort	Statuscode: 200
	{
	"success": true,
	"project_id": 1,
	"project_creator": {
	"id": 34,
	"username": "isaac14",
	"email": "kirlin.zachary@example.com",
	"first_name": "Freeda",
	"last_name": "Von"
	},
	"members": [
	{
	"id": 2,
	"username": "marquise17",
	"email": "bernier.darrick@example.com",
	"first_name": "Delphia",
	"last_name": "Kemmer",
	"pivot": {
	"project_id": 1,
	"user_id": 2,
	"can_create_tasks": 1,
	"can_edit_tasks": 1,
	"can_create_tags": 1
	}
	}
	]
	}

Parameter	-

4.Tasks

Eine Aufgabe gehört zu einem Projekt und referenziert einen Besitzer/Ersteller und eine zugewiesenen Person (die auch der Besitzer sein kann). Solange das Projekt, in welchem die Task erstellt wurde noch existiert, der Account des Erstellers noch existiert oder sie nicht manuell gelöscht wurde, existiert auch die Aufgabe selbst noch. Sollte die zugewiesene Person ihren Account löschen, so wird der Wert der *assignee_user_id* auf *NULL* gesetzt, die Aufgabe besteht aber weiterhin.
Sollte der zugewiesene Tag der Aufgabe gelöscht werden, wird das Feld *tag_id* auf null gesetzt.

4.1 Alle Tasks eines Projekts erhalten

Request Typ	GET
URL	/api/project/{id}/tasks
Request Body Beispiel	-
Status Codes	200 404
Beispielantwort	Statuscode: 200 { "success": true, "project_id": 6, "tasks": [] } (-> keine Tasks für dieses Projekt vorhanden) ———————————————————————————————— -- Statuscode: 200 "success": true, "project_id": 2, "tasks": [{ "id": 8, "title": "Quo nemo modi voluptatem animi.", "description": "Totam minima distinctio inventore et architecto. Sit est accusantium dicta est. Dignissimos aut quisquam sunt illum totam rerum omnis nam.", "creator_user_id": 2,

<table>
<tr><td></td><td><pre>
"assignee_user_id": 2,
"due_date": "2022-04-03",
"completed_date": null,
"due_time": "08:23:24",
"completed_time": null,
"completion_comment": null,
"project_id": 2,
"status_id": 2,
"priority_id": 2,
"tag_id": null,
"created_at": "2022-05-16T12:46:36.000000Z",
"updated_at": "2022-05-16T12:46:36.000000Z",
"assignee": {
"id": 2,
"username": "marquise17",
"email": "bernier.darrick@example.com",
"first_name": "Delphia",
"last_name": "Kemmer"
},
"creator": {
"id": 2,
"username": "marquise17",
"email": "bernier.darrick@example.com",
"first_name": "Delphia",
"last_name": "Kemmer"
},
"status": {
"id": 2,
"title": "In Arbeit",
"slug": "in-progress"
},
"tag": null,
"priority": {
"id": 2,
"title": "Mittel",
"slug": "medium"
}
}, … usw
{...}, {...}]}
</pre></td></tr>
<tr><td>Parameter</td><td>-</td></tr>
</table>

4.2 Task erstellen

Beim Erstellen einer Task wird das Projekt automatisch dem User der die Anfrage stellt zugewiesen. Aufgaben erstellen können lediglich der Besitzer des Projekts und Mitglieder die das Recht dazu besitzen (can_create_tasks).

<table>
<tr><td>Request Typ</td><td>POST</td></tr>
<tr><td>URL</td><td>/api/task</td></tr>
<tr><td>Request Body Beispiel</td><td><pre>{
    "assignee_user_id":2,
    "project_id":1,
    "title":"Eat more Lasagne",
    "description":"You need to eat more lasagne.",
    "due_date": "2022-06-09",
    "due_time" : "12:15",
    "priority_id": 1,
    "tag_id": 4234,
    "status_id" : 2
}</pre></td></tr>
<tr><td>Status Codes</td><td>201
422
403
500</td></tr>
<tr><td>Beispielantwort</td><td><pre>Statuscode: 201
{
    "success": true,
    "task": {
        "creator_user_id": 1,
        "assignee_user_id": 2,
        "project_id": 1,
        "title": "Eat more Lasagne",
        "description": "You need to eat more lasagne.",
        "due_date": "2022-06-09",
        "due_time": "12:15",
        "updated_at": "2022-05-21T07:15:33.000000Z",
        "created_at": "2022-05-21T07:15:33.000000Z",
        "id": 18
    }
}</pre></td></tr>
<tr><td>Parameter</td><td><table>
<tr><td>assignee_user_id</td><td>int</td><td>Existierende user_id</td></tr>
</table></td></tr>
</table>

project_id		int	Existierende project_id
status_id		int	Existierende status_id
priority_id		int	Existierende priority_id
tag_id		int	Existierende tag_id
title		string	Max-Länge: 255
description		string	Max-Länge: 1000
due_date		date	Format: YYYY-MM-DD
due_time		date	Format: H:i

4.3 Task bearbeiten

Tasks können von Nutzern bearbeitet werden, die die Task entweder erstellt haben, oder das Recht besitzen alle Tasks zu bearbeiten (can_edit_tasks). Besitzer eines Projekts können jede Task bearbeiten.

Request Typ	PUT
URL	/api/task/{id}

Request Body Beispiel	{
	"title": "Make and eat more Lasagne",
	"due_date": "2022-06-07"
	}

Status Codes	201 404 403 422

Beispielantwort	{
	"success": true,
	"task": {
	"id": 18,
	"title": "Make and eat more Lasagne",
	"description": "You need to eat more lasagne.",
	"creator_user_id": 1,
	"assignee_user_id": 2,
	"due_date": "2022-06-07",

<table>
<tr><td></td><td><pre>
        "completed_date": null,
        "due_time": "12:15:00",
        "completed_time": null,
        "completion_comment": null,
        "project_id": 1,
        "status_id": null,
        "priority_id": null,
        "tag_id": null,
        "created_at": "2022-05-21T07:15:33.000000Z",
        "updated_at": "2022-05-21T07:37:24.000000Z"
    }
}
</pre></td></tr>
<tr><td>Parameter</td><td>
<table>
<tr><td>assignee_user_id</td><td></td><td>int</td><td>Existierende user_id</td></tr>
<tr><td>status_id</td><td></td><td>int</td><td>Existierende status_id</td></tr>
<tr><td>priority_id</td><td></td><td>int</td><td>Existierende priority_id</td></tr>
<tr><td>tag_id</td><td></td><td>int</td><td>Existierende tag_id</td></tr>
<tr><td>title</td><td></td><td>string</td><td>Max-Länge: 255</td></tr>
<tr><td>description</td><td></td><td>string</td><td>Max-Länge: 1000</td></tr>
<tr><td>due_date</td><td></td><td>date</td><td>Format: YYYY-MM-DD</td></tr>
<tr><td>due_time</td><td></td><td>date</td><td>Format: H:i</td></tr>
<tr><td>completion_comment</td><td></td><td>string</td><td>Max-Länge: 255</td></tr>
</table>
</td></tr>
</table>

4.4 Task löschen

Tasks können lediglich vom Besitzer des Projekts oder dem Ersteller der Aufgabe gelöscht werden.

Request Typ	DELETE
URL	/api/task/{id}
Request Body Beispiel	-
Status Codes	200 404

	403 500

Beispielantwort	Statuscode 200:
	{
	"success": true,
	"message": "Task deleted."
	}
	---------- 	
	--
	Statuscode 404:
	{
	"success": false,
	"message": "Task not found"
	}

Parameter	-

4.5 Task abschließen

Beim Abschließen einer Aufgabe muss die id in der URL übergeben werden. Der Abschlusskommentar ist optional. Das Datum und die Uhrzeit werden automatisch beim Verarbeiten der Anfrage gesetzt. Die status_id wird ebenfalls automatisch auf 1 (“Erledigt”) gesetzt.
Eine Task kann nur von deren Ersteller, der zugewiesenen Person oder dem Besitzer der Projekts abgeschlossen werden.

Request Typ	PUT
URL	/api/task/{id}/complete

Request Body Beispiel	{
	"completion_comment": "Was fun!"
	}

Error Codes	201 404 403 422

Beispielantwort	{
	"success": true,
	"data": {
	"completion_comment": "Was fun!",
	"completed_date": "2022-05-21",

	<pre>"completed_time": "07:44:54",
	"status_id": 1
	}
	}</pre>

Parameter	completion_comment | string | Max-Länge: 255

5.Tags

5.1 Tag erstellen

Tags können entweder vom Ersteller des Projekts oder von Mitgliedern des Projekts erstellt werden. Um Tags erstellen zu können muss das Projektmitglied jedoch das Recht dazu vom Ersteller des Projekts zugeteilt bekommen (can_create_tags).

Request Typ	POST
URL	/api/tag

Request Body Beispiel	<pre>{
	"project_id": 1,
	"title": "Design"
	}</pre>

Status Codes	201 422 500

Beispielantwort	<pre>Statuscode: 201
	{
	"success": true,
	"tag": {
	"project_id": 1,
	"title": "Design",
	"created_at": "2022-05-21T08:04:42.000000Z",
	"id": 8
	}
	}</pre>

Parameter	project_id | int | Existierende project_id

	title		string	Max-Länge: 255
	description		string	Max-Länge: 255

5.2 Tag löschen

Tags können ausschließlich vom Besitzer des Projekts gelöscht werden.

Request Typ	DELETE
URL	/api/tag/{id}
Request Body Beispiel	-
Status Codes	200 403 404 500

Beispielantwort	Statuscode 200:
	{
	"success": true,
	"message": "Tag deleted."
	}
	---------- 	
	--
	Statuscode 404:
	{
	"success": false,
	"message": "Tag could not be found."
	}

Parameter	-

6. User

6.1 Alle Projekte eines Nutzers erhalten

Das Ergebnis liefert alle Projekte des Nutzers, unterteilt in Projekte die er selbst erstellt hat (projects_created) und Projekte in denen er Mitglied ist (project-member_of) samt Rechten und Informationen zum jeweiligen Ersteller des Projekts.

Request Typ	GET
URL	/api/user/{id}/projects
Request Body Beispiel	-
Error Codes	201 401 404

Beispielantwort	Statuscode 200
	{
	"success": true,
	"projects_created": [
	{
	"id": 16,
	"title": "Sed recusandae qui officiis aut esse vel",
	"description": "Fugiat tempora incidunt nihil et qui”,
	"creator_user_id": 49,
	"due_date": "2022-12-06",
	"completion_date": null,
	"created_at": "2021-10-20T13:18:36.000000Z",
	"updated_at": "2022-05-29T13:10:18.000000Z",
	"progress_percentage": 33
	}
	],
	"project-member_of": [
	{
	"id": 2,
	"title": "Rerum dolorum minima est eveniet.",
	"description": "Officiis quo corporis autem.",
	"creator_user_id": 1,
	"due_date": "2023-04-17",
	"completion_date": null,
	"created_at": "2023-03-01T23:44:25.000000Z",
	"updated_at": "2022-05-29T13:10:18.000000Z",
	"progress_percentage": 0,
	"user_project_rights": {
	"user_id": 49,
	"project_id": 2,
	"can_edit_tasks": 0,
	"can_create_tasks": 1,
	"can_assign_tasks": 0,

	"can_create_tags": 1
	},
	"creator": {
	"id": 1,
	"username": "robyn77",
	"email": "qkutch@example.org",
	"first_name": "Blaise",
	"last_name": "Hermiston"
	}
	}
	]
	}
	---------- 	
	--
	Statuscode 404
	{
	"success": false,
	"message": "User not found."
	}
	---------- 	
	--
	Statuscode 404
	{
	"success": true,
	"message": "No projects found for this user."
	}

Parameter	-

6.2 Tasks eines Users mit bestimmten Status erhalten

Liefert alle Tasks mit dem angegebenen Status-Slug des momentan angemeldeten Benutzers (identifiziert über das Bearer-Token).
Folgende Status-Slugs stehen zur Verfügung:

- completed
- in-progress
- paused
- failed

Request Typ	GET
URL	/api/user/tasks/{slug}

Request Body Beispiel	-
Error Codes	201 401 404

Beispielantwort	Statuscode 200
	{
	"success": true,
	"tasks": [
	{
	"id": 50,
	"title": "Optio odit consequatur doloribus",
	"description": "Accusamus quo odio aliquid nihil",
	"creator_user_id": 31,
	"assignee_user_id": 1,
	"due_date": "2022-03-31",
	"completed_date": "2022-06-01",
	"due_time": "01:06:28",
	"completed_time": "08:41:39",
	"completion_comment": "I did it motherfucker.",
	"project_id": 18,
	"status_id": 1,
	"priority_id": 2,
	"tag_id": 35,
	"created_at": "2022-05-29T13:10:20.000000Z",
	"updated_at": "2022-06-01T08:41:39.000000Z",
	"creator": {
	"id": 31,
	"username": "wkshlerin"
	},
	"project": {
	"id": 18,
	"title": "Consectetur voluptate consequatur"
	},
	"priority": {
	"id": 2,
	"title": "Mittel",
	"slug": "medium"
	},
	"tag": {
	"id": 35,
	"title": "voluptas",
	"description": "Repellat porro dolore consequa",
	"project_id": 29,

	<pre>"created_at": "2022-05-29T13:10:19.000000Z"
	},
	"status": {
	"id": 1,
	"title": "Erledigt",
	"slug": "completed"
	}
	}
	]
	}
	---------- 	
	--
	Statuscode 404
	{
	"success": false,
	"message": "Status not found"
	}
	(bei falschem Slug)
	---------- 	
	--
	Statuscode 404
	{
	"success": true,
	"message": "No projects found for this user."
	}</pre>

Parameter	-

6.3 Ein Projekt verlassen

Durch das in der Request mitgelieferte Bearer-Token kann der User identifiziert werden und das Projekt (id-Parameter in der URL) verlassen. Der Nutzer kann das Projekt nicht verlassen, wenn für die angegebene id kein Projekt existiert, er der Besitzer des Projektes ist oder er kein Mitglied des angegebenen Projektes ist.

Request Typ	**DELETE**
URL	/api/project/{id}/leave
Request Body Beispiel	-
Status Codes	200 404

	500

Beispielantwort	Statuscode 200
	{
	"success": true,
	"message": "Successfully left the project."
	}
	---------- 	
	--
	Statuscode 404
	{
	"success": false,
	"message": "You are not a member of this project."
	}

Parameter	-

6.4 Benutzerdaten anpassen

Durch das in der Request mitgelieferte Bearer-Token kann der User identifiziert werden.
Die Daten des Nutzers werden entsprechend des Request Bodys aktualisiert.

Request Typ	PUT
URL	/api/user

Request Body Beispiel	{
	"first_name":"Tim",
	"last_name":"Apple",
	"email":"tim.apple@gmail.com",
	"password":"newpassword",
	"password_confirmation":"newpassword"
	}

Status Codes	200 401 404

Beispielantwort	Statuscode 200:
	{
	"success": true,
	"password_changed": true,
	"user": {

<table>
<tr><td></td><td><pre>
        "id": 1,
        "username": "ingo",
        "email": "tim.apple@gmail.com",
        "first_name": "Tim",
        "last_name": "Apple"
    }
}
</pre></td></tr>
<tr><td>Parameter</td><td>
<table>
<tr><td>first_name</td><td></td><td>String, Maximale Länge: 30</td></tr>
<tr><td>last_name</td><td></td><td>String, Maximale Länge: 30</td></tr>
<tr><td>username</td><td></td><td>String, Maximale Länge: 20, noch nicht vergeben</td></tr>
<tr><td>email</td><td></td><td>E-Mail Format, Maximale L.: 30, noch nicht vergeben</td></tr>
<tr><td>password</td><td></td><td>String, Mindestens 8 Zeichen</td></tr>
<tr><td>password_confirmation</td><td></td><td>password</td></tr>
</table>
</td></tr>
</table>

6.5 Einen Benutzer löschen

Durch das in der Request mitgelieferte Bearer-Token kann der User identifiziert und das zugehörige User Konto gelöscht werden. Projekte, die der Nutzer erstellt hat werden gelöscht. Alle Buchungen in denen der User referenziert wird werden ebenfalls gelöscht. Bei Tasks wird die id, die auf den User verweist auf NULL gesetzt (assignee oder creator id).

Request Typ	**DELETE**
URL	/api/user
Request Body Beispiel	-
Status Codes	200 404 500

Beispielantwort	Statuscode 200
	{
	"deleted": true
	}

	---------- 	
	--
	Statuscode 404
	{
	"deleted": false,
	"message": "User not found."
	}
	---------- 	
	--
	Statuscode 500
	{
	"deleted": false,
	"message": "User could not be deleted."
	}

Parameter	-

7. Prioritäten

Die verfügbaren Prioritäten sind Anwendungsweit vorgegeben (Hoch, Mittel, Gering) und können weder bearbeitet noch ergänzt werden. Um beim Erstellen von Tasks eine Auswahl geben zu können und die richtige id in die Anfrage einzufügen, können alle Prioritäten samt id, Titel und Slug angefragt werden.
Für diese Anfrage muss kein Authentifizierung stattfinden.

7.1 Alle Prioritäten erhalten

Request Typ	GET
URL	/api/priorities/all
Request Body Beispiel	-
Error Codes	200 404
Beispielantwort	[[{ "id": 1, "title": "Hoch", "slug": "high" }, {

	"id": 2,
	"title": "Mittel",
	"slug": "medium"
	},
	{
	"id": 3,
	"title": "Gering",
	"slug": "low"
	}
	]
	]

Parameter	-

8. Status

Die verfügbaren Status sind Anwendungsweit vorgegeben (Erledigt, In Arbeit, Pausiert, Fehlgeschlagen) und können weder bearbeitet noch ergänzt werden. Um beim Erstellen/Bearbeiten von Tasks eine Auswahl geben zu können und die richtige id in die Anfrage einzufügen, können alle Status samt id, Titel und Slug angefragt werden.
Für diese Anfrage muss kein Authentifizierung stattfinden.

7.1 Alle Prioritäten erhalten

Request Typ	GET
URL	/api/priorities/all
Request Body Beispiel	-
Error Codes	200 404
Beispielantwort	[[{ "id": 1, "title": "Hoch", "slug": "high" }, { "id": 2, "title": "Mittel", "slug": "medium"

	<pre>},
	{
	"id": 3,
	"title": "Gering",
	"slug": "low"
	}
	]
	]</pre>

Parameter	-

9. Räume

9.1 Raum erstellen

Request Typ	POST
URL	/api/room

Request Body Beispiel	<pre>{
	"title": "Testraum",
	"description": "Ein Raum zum testen",
	"room_number": "A111",
	"capacity": "100",
	"equipment_info": "Whiteboard",
	"address_info": "1234 Straße",
	"opening_time": "12:00:00",
	"closing_time": "13:00:00",
	"opened_on_weekends": "0",
	}</pre>

Status Codes	201 500

Beispielantwort	<pre>Statuscode: 201
	"success": true,
	"room": {
	"title": "Testraum",
	"description": "Ein Raum zum testen",
	"room_number": "A111",
	"capacity": "100",
	"equipment_info": "Whiteboard",
	"address_info": "1234 Straße",
	"opening_time": "12:00:00",</pre>

<table>
<tr><td></td><td>

<pre>
        "closing_time": "13:00:00",
        "opened_on_weekends": "0",
        "updated_at": "2022-05-29T18:20:30.000000Z",
        "created_at": "2022-05-29T18:20:30.000000Z",
        "id": 24
    }
</pre>

</td></tr>
<tr><td>Parameter</td><td>

<table>
<tr><td>title</td><td>string</td><td>Max-Länge: 255</td></tr>
<tr><td>description</td><td>string</td><td>Max-Länge: 255</td></tr>
<tr><td>room_number</td><td>string</td><td>Max-Länge: 12</td></tr>
<tr><td>capacity</td><td>int</td><td>Anzahl Stellen zw. 1-11</td></tr>
<tr><td>equipment_info</td><td>string</td><td>Max-Länge: 500</td></tr>
<tr><td>address_info</td><td>string</td><td>Max-Länge: 500</td></tr>
<tr><td>openening_time</td><td>time</td><td>Format: H:i:s</td></tr>
<tr><td>closing_time</td><td>time</td><td>Format: H:i:s</td></tr>
<tr><td>opened_on_weekends</td><td>tinyint</td><td></td></tr>
</table>

</td></tr>
</table>

9.2 Alle Räume eines Projektes erhalten

Request Typ	GET
URL	/api/project/{id}/rooms
Request Body Beispiel	-
Error Codes	200 404
Beispielantwort	Statuscode: 200 { "success": true, "project_id": 1, "rooms": [] } (-> keine Räume für dieses Projekt vorhanden)

	---------- 	
	--
	Statuscode: 200
	"success": true,
	"project_id": 2,
	"rooms": [
	{
	"id": 2,
	"created_at": "2022-05-29T18:06:24.000000Z",
	"updated_at": "2022-05-29T18:06:24.000000Z",
	"title": "maxime",
	"description": "Vel vitae fugiat impedit.",
	"room_number": "G292",
	"capacity": 19,
	"equipment_info": "Ullam et accusamus eum.",
	"address_info": "97301 Neal Burg
	66014-7002,Quitzonshire",
	"opening_time": "05:05:48",
	"closing_time": "15:54:33",
	"opened_on_weekends": 1,
	"project_id": 2
	}
	]

Parameter	-

9.3 Raum holen

Request Typ	GET
URL	/api/room/{id}
Request Body Beispiel	-
Error Codes	200 404
Beispielantwort	{ "id": 1, "created_at": "2022-05-29T18:06:24.000000Z", "updated_at": "2022-05-29T18:06:24.000000Z", "title": "rerum",

	"description": "Consequatur nesciunt autem facilis veritatis illo nulla eum eos.", "room_number": "B156", "capacity": 30, "equipment_info": "Sit ullam quis molestias a.", "address_info": "48041 Schowalter Curve Apt. 657 09401-4323,Port Jeanetteport", "opening_time": "20:10:11", "closing_time": "08:22:58", "opened_on_weekends": 1, "project_id": 25 }
Parameter	-

9.4 Raum bearbeiten

Request Typ	PUT
URL	/api/room/{id}

Request Body Beispiel	{
	"capacity": "100",
	"opened_on_weekends": "0"
	}

Status Codes	201 404

Beispielantwort	{
	"success": true,
	"room": {
	"id": 1,
	"created_at": "2022-05-29T18:06:24.000000Z",
	"updated_at": "2022-05-29T19:40:55.000000Z",
	"title": "rerum",
	"description": "Consequatur nesciunt autem facilis veritatis illo nulla eum eos.",
	"room_number": "B156",
	"capacity": "100",
	"equipment_info": "Sit ullam quis molestias a.",
	"address_info": "48041 Schowalter Curve Apt. 657 09401-4323,Port Jeanetteport",
	"opening_time": "20:10:11",

<table>
<tr><td></td><td>

```
        "closing_time": "08:22:58",
        "opened_on_weekends": "0",
        "project_id": 25
    }
`
```

</td></tr>
<tr><td>Parameter</td><td>

title		string	Max-Länge: 255
description		string	Max-Länge: 255
room_number		string	Max-Länge: 12
capacity		int	Anzahl Stellen zw. 1-11
equipment_info		string	Max-Länge: 500
address_info		string	Max-Länge: 500
openening_time		time	Format: H:i:s
closing_time		time	Format: H:i:s
opened_on_weekends		tinyint	

</td></tr>
</table>

9.5 Raum löschen

Request Typ	DELETE
URL	/api/room/{id}
Request Body Beispiel	-
Status Codes	200 404 500

Beispielantwort	Statuscode 200:
	{
	"success": true,
	"message": "Room deleted."
	}
	---------- 	
	--

	Statuscode 404:
	{
	"success": false,
	"message": "Room not found"
	}

Parameter	-

10. Raumbuchungen

10.1 Alle Buchung eines Raums holen

Request Typ	GET
URL	/api/room/{id}/bookings
Request Body Beispiel	-
Error Codes	200 404

Beispielantwort	Statuscode: 200
	{
	"success": true,
	"room_id": 10,
	"bookings": []
	} (-> keine Buchungen für diesen Raum vorhanden)
	---------- 	
	--
	Statuscode: 200
	"success": true,
	"room_id": 11,
	"bookings": [
	{
	"id": 1,
	"created_at": "2022-05-29T18:06:24.000000Z",
	"updated_at": "2022-05-29T18:06:24.000000Z",
	"room_id": 11,
	"user_id": 13,
	"reservation_date": "2022-05-28",
	"start_time": "00:00:00",
	"end_time": "00:00:00",
	"room": {

	"id": 11,
	"created_at": "2022-05-29T18:06:24.000000Z",
	"updated_at": "2022-05-29T18:06:24.000000Z",
	"title": "in",
	"description": "Minus nemo velit doloribus suscipit culpa qui sint.",
	"room_number": "B183",
	"capacity": 14,
	"equipment_info": "Laboriosam et eligendi est a quam repudiandae iure vel.",
	"address_info": "95303 Garett Forges Apt. 586 52507-5034,Lake Kenyaburgh",
	"opening_time": "18:57:30",
	"closing_time": "18:41:27",
	"opened_on_weekends": 1,
	"project_id": 26
	},
	"user": {
	"id": 13,
	"username": "jonathon.stokes",
	"email": "nkshlerin@example.org",
	"first_name": "Wilmer",
	"last_name": "Little"
	}
	}
	]

Parameter	-

10.2 Alle Buchung eines Users holen

Request Typ	GET
URL	/api/user/{id}/bookings
Request Body Beispiel	-
Error Codes	200 404
Beispielantwort	Statuscode: 200 { "success": true, "room_id": 1,

```
   "bookings": []
} (-> keine BUchungen für diesen User vorhanden)
-----------------------------------------------------------------
--
Statuscode: 200
  "success": true,
   "room_id": 13,
   "bookings": [
       {
           "id": 1,
           "created_at": "2022-05-29T18:06:24.000000Z",
           "updated_at": "2022-05-29T18:06:24.000000Z",
           "room_id": 11,
           "user_id": 13,
           "reservation_date": "2022-05-28",
           "start_time": "00:00:00",
           "end_time": "00:00:00",
           "room": {
               "id": 11,
               "created_at": "2022-05-29T18:06:24.000000Z",
               "updated_at": "2022-05-29T18:06:24.000000Z",
               "title": "in",
               "description": "Minus nemo velit doloribus
suscipit culpa qui sint.",
               "room_number": "B183",
               "capacity": 14,
               "equipment_info": "Laboriosam et eligendi est a
quam repudiandae iure vel.",
               "address_info": "95303 Garett Forges Apt. 586
52507-5034,Lake Kenyaburgh",
               "opening_time": "18:57:30",
               "closing_time": "18:41:27",
               "opened_on_weekends": 1,
               "project_id": 26
           },
           "user": {
               "id": 13,
               "username": "jonathon.stokes",
               "email": "nkshlerin@example.org",
               "first_name": "Wilmer",
               "last_name": "Little"
           }
       }
   ]
```

	]
Parameter	-

10.3 Buchung erstellen

Request Typ	POST
URL	/api/bookings

Request Body Beispiel	{
	"room_id": "2",
	"user_id": "1",
	"reservation_date": "2022-06-25",
	"start_time": "12:00:00",
	"end_time": "13:00:00",
	}

Status Codes	201 422 500

Beispielantwort	Statuscode: 422
	{
	"success": false,
	"message": "This timeslot is not available"
	}
	--- 	
	Statuscode: 422
	{
	"success": false,
	"message": "The timeslot of the booking is not within the time range of the opening hours"
	}
	--- 	
	Statuscode: 201
	"success": true,
	"room": {
	"room_id": "2",
	"user_id": "1",
	"reservation_date": "2022-06-25",
	"start_time": "12:00:00",
	"end_time": "13:00:00",
	"updated_at": "2022-05-30T08:04:45.000000Z",

<table>
<tr><td></td><td><pre>
    "created_at": "2022-05-30T08:04:45.000000Z",
    "id": 51
}

    }
</pre></td></tr>
<tr><td>Parameter</td><td>
<table>
<tr><td>room_id</td><td></td><td>id</td><td>Id eines existierenden Raums</td></tr>
<tr><td>user_id</td><td></td><td>id</td><td>Id eines existierenden Users</td></tr>
<tr><td>reservation_date</td><td></td><td>date</td><td>Format: Y-m-d</td></tr>
<tr><td>start_time</td><td></td><td>date</td><td>Format: H:i:s</td></tr>
<tr><td>end_time</td><td></td><td>date</td><td>Format: H:i:s</td></tr>
</table>
</td></tr>
</table>

10.4 Buchung löschen

Request Typ	DELETE
URL	/api/bookings/{id}
Request Body Beispiel	-
Status Codes	200 404

Beispielantwort	Statuscode 200:
	{
	"success": true,
	"message": "Booking deleted"
	}
	---------- 	
	--
	Statuscode 404:
	{
	"success": false,
	"message": "Booking not found"
	}

Parameter	-

11. Ankündigungen

11.1 Alle Ankündigungen eines Users holen

Request Typ	GET
URL	/api/user/{id}/announcements
Request Body Beispiel	-
Error Codes	200 404

Beispielantwort	Statuscode: 200
	{
	"success": true,
	"user_id": 7,
	"announcements": []
	} (-> keine Announcements für diesen User vorhanden)
	---------- 	
	--
	Statuscode: 200
	"success": true,
	"user_id": 1,
	"announcements": [
	{
	"id": 1,
	"created_at": "2022-05-29T18:06:22.000000Z",
	"updated_at": "2022-05-29T18:06:22.000000Z",
	"subject": "ut",
	"message": "Ut sed hic quaerat sed quisquam nobis neque.",
	"user_id": 1,
	"project_id": 19
	}
	]

Parameter	-

11.2 Alle Ankündigungen eines Projektes holen

Request Typ	GET
URL	/api/project/{id}/announcements
Request Body Beispiel	-
Error Codes	200 404

Beispielantwort	Statuscode: 200
	{
	"success": true,
	"project_id": 7,
	"announcements": []
	} (-> keine Announcements für diesen Project vorhanden)
	---------- 	
	--
	Statuscode: 200
	"success": true,
	"project_id": 5,
	"announcements": [
	{
	"id": 16,
	"created_at": "2022-05-29T18:06:22.000000Z",
	"updated_at": "2022-05-29T18:06:22.000000Z",
	"subject": "voluptas",
	"message": "Distinctio porro eos ratione sed.",
	"user_id": 5,
	"project_id": 5
	}
	]

Parameter	-

11.3 Alle Ankündigungen eines Projektes eines Users

Request Typ	GET
URL	/api/project/{id}/user/{id}/announcements
Request Body Beispiel	-
Error Codes	200 404

Beispielantwort	Statuscode: 404
	{
	"success": false,
	"message": "No announcement found"
	}
	---------- 	
	--
	Statuscode: 200
	"success": true,
	"project_id": "5",
	"user_id": "5",
	"announcements": [
	{
	"id": 16,
	"created_at": "2022-05-29T18:06:22.000000Z",
	"updated_at": "2022-05-29T18:06:22.000000Z",
	"subject": "voluptas",
	"message": "Distinctio porro eos ratione sed.",
	"user_id": 5,
	"project_id": 5
	}
	]

Parameter	-

11.4 Ankündigung erstellen

<table>
<tr><td>Request Typ</td><td>POST</td></tr>
<tr><td>URL</td><td>/api/announcements</td></tr>
<tr><td>Request Body Beispiel</td><td><pre>{
        "subject": "test",
        "message": "test",
        "user_id": "21",
        "project_id": "13"
}</pre></td></tr>
<tr><td>Status Codes</td><td>201
422
500</td></tr>
<tr><td>Beispielantwort</td><td><pre>Statuscode: 422
{
   "success": false,
   "message": "User not part of project"
}

Statuscode: 201
"success": true,
   "Announcement": {
        "subject": "test",
        "message": "test",
        "user_id": "21",
        "project_id": "13",
        "updated_at": "2022-05-30T13:38:10.000000Z",
        "created_at": "2022-05-30T13:38:10.000000Z",
        "id": 51
   }</pre></td></tr>
<tr><td>Parameter</td><td>
<table>
<tr><td>subject</td><td></td><td>string</td><td>Max-Länge: 255</td></tr>
<tr><td>message</td><td></td><td>string</td><td>Max-Länge: 10000</td></tr>
<tr><td>user_id</td><td></td><td>id</td><td>Existierender User</td></tr>
<tr><td>project_id</td><td></td><td>id</td><td>Existierendes Projekt</td></tr>
</table>
</td></tr>
</table>

11.5 Ankündigung bearbeiten

Request Typ	PUT
URL	/api/announcements/{id}

Request Body Beispiel	<pre>{
	  "subject": "New Subject",
	  "message": "New Message"
	}</pre>

Status Codes	201 404

Beispielantwort	<pre>"success": true,
	  "user": {
	  "id": 1,
	  "created_at": "2022-05-30T09:11:35.000000Z",
	  "updated_at": "2022-05-30T13:54:10.000000Z",
	  "subject": "new subject",
	  "message": "new message",
	  "user_id": 39,
	  "project_id": 4
	  }</pre>

Parameter	<table><tr><td>subject</td><td></td><td>string</td><td>Max-Länge: 255</td></tr><tr><td>message</td><td></td><td>string</td><td>Max-Länge: 10000</td></tr></table>

11.6 Ankündigung löschen

Request Typ	DELETE
URL	/api/announcements/{id}
Request Body Beispiel	-
Status Codes	200 404

Beispielantwort	<pre>Statuscode 200:
	{
	  "success": true,</pre>

	"message": "Announcement deleted"
	}
	---------- 	
	--
	Statuscode 404:
	{
	"success": **false**,
	"message": "Announcement not found"
	}

Parameter	-