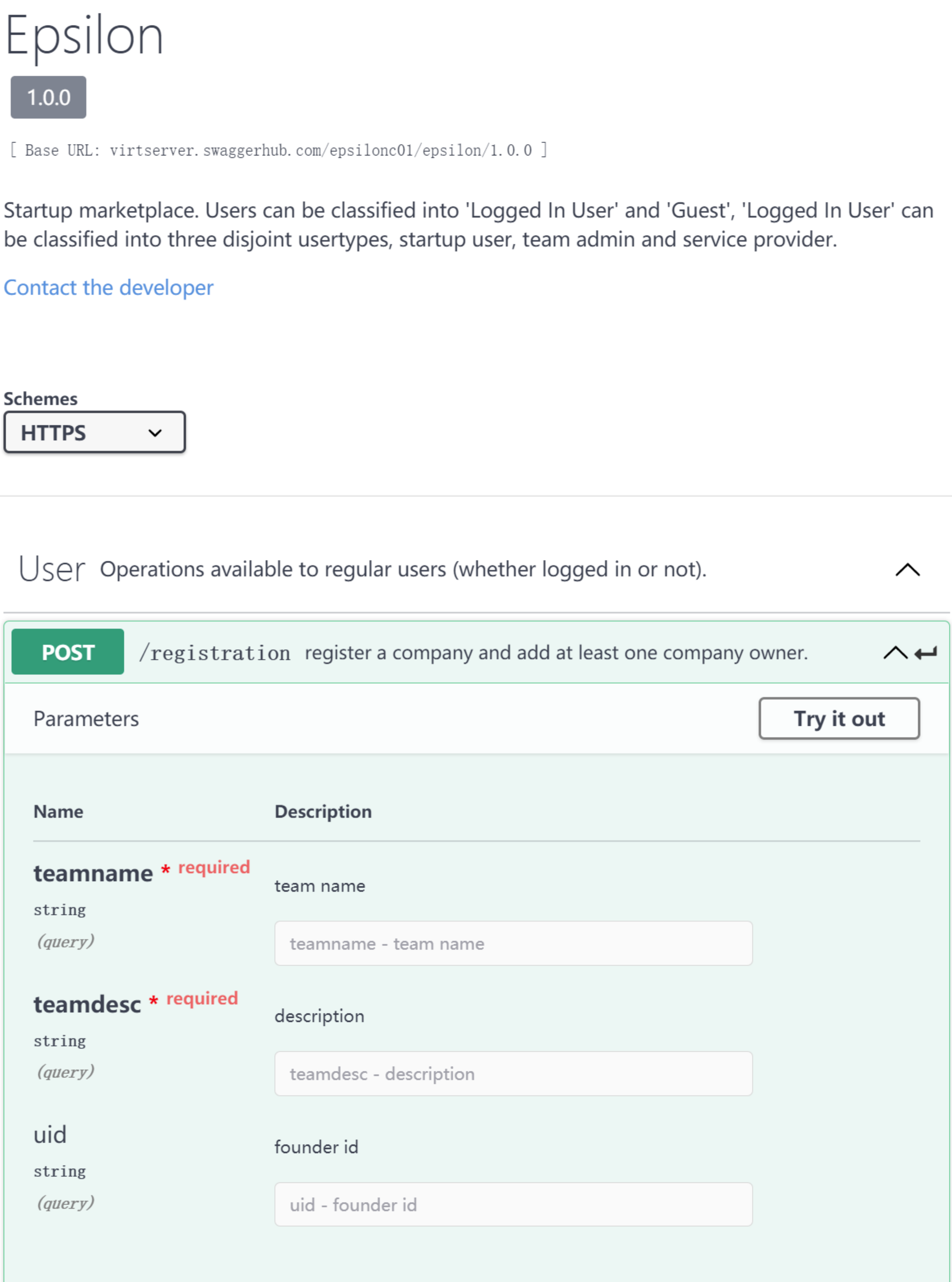


Epsilon

1.0.0

[Base URL: virtserver.swaggerhub.com/epsilonc01/epsilon/1.0.0]

Startup marketplace. Users can be classified into 'Logged In User' and 'Guest', 'Logged In User' can be classified into three disjoint usertypes, startup user, team admin and service provider.

Contact the developer

Schemes

HTTPS

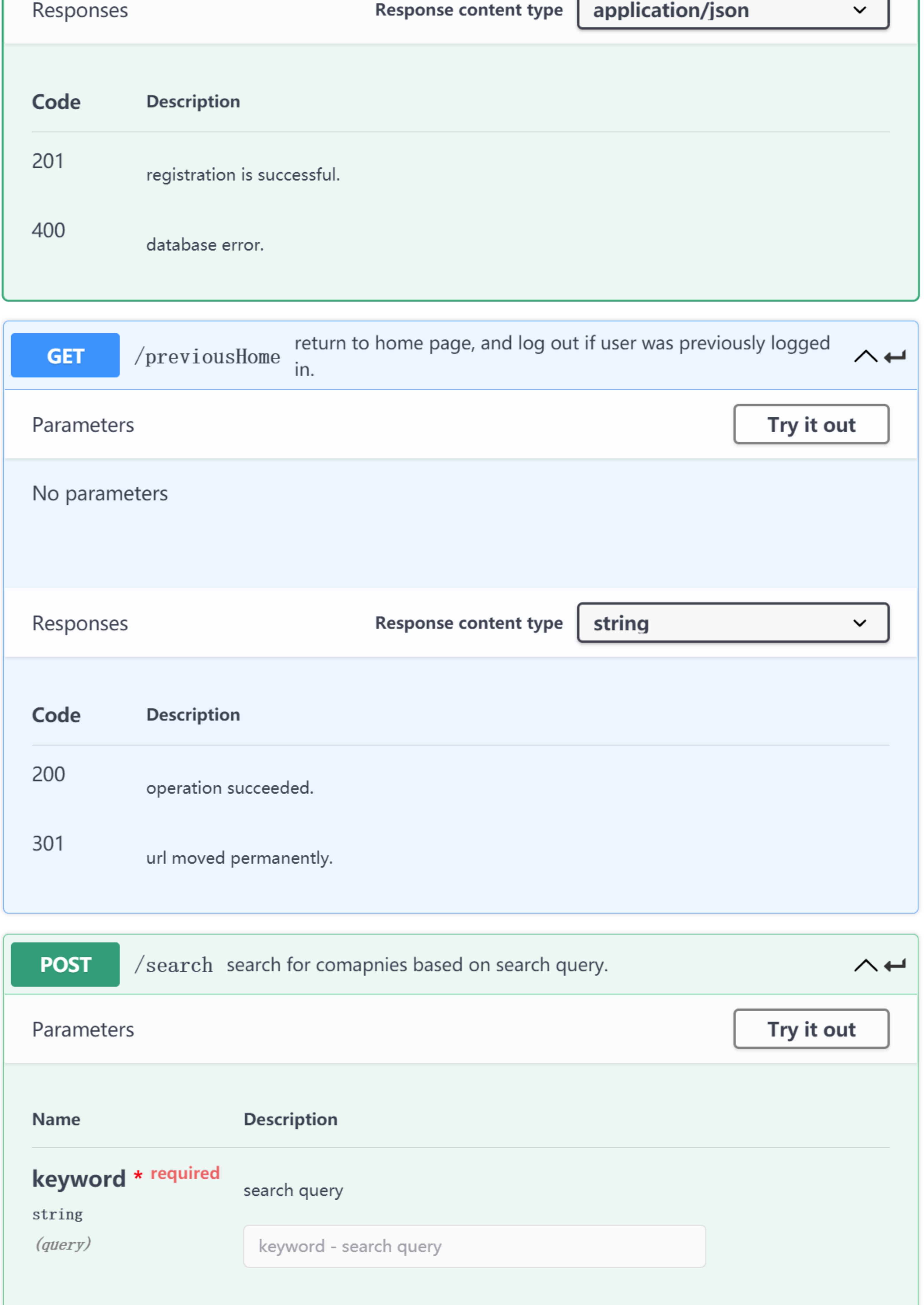
User Operations available to regular users (whether logged in or not).

POST /registration register a company and add at least one company owner.

Parameters

Try it out

Name	Description
teamname * required string *(query)*	team name teamname - team name
teamdesc * required string *(query)*	description teamdesc - description
uid string *(query)*	founder id uid - founder id



Responses

Response content type: application/json

Code	Description
201	registration is successful.
400	database error.

GET /previousHome return to home page, and log out if user was previously logged in.

Parameters

No parameters

Responses

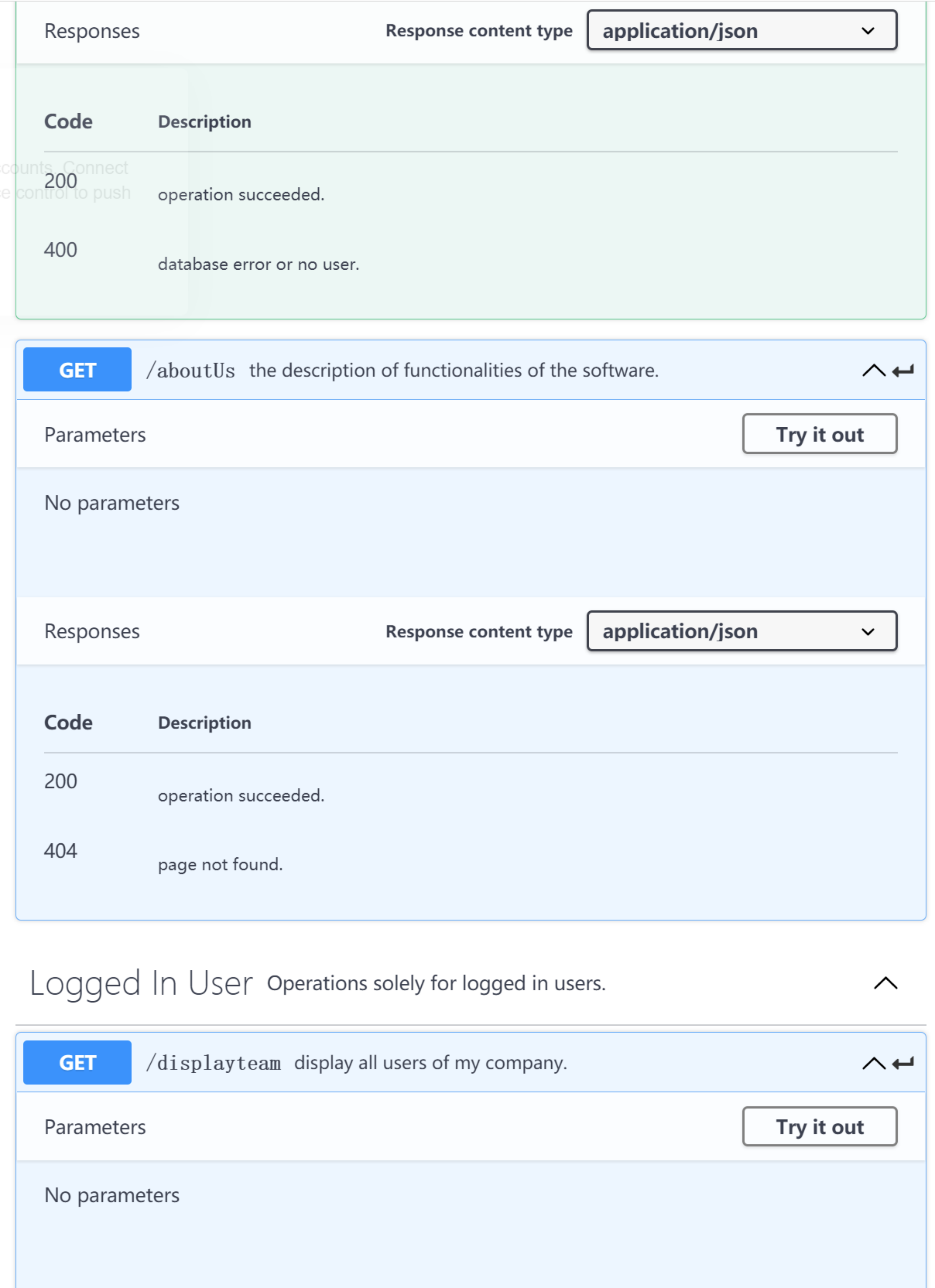
Response content type: string

Code	Description
200	operation succeeded.
301	url moved permanently.

POST /search search for comapnies based on search query.

Parameters

Name	Description
keyword * required string *(query)*	search query keyword - search query



Responses

Response content type: application/json

Code	Description
200	operation succeeded.
400	database error or no user.

GET /aboutUs the description of functionalities of the software.

Parameters

No parameters

Responses

Response content type: application/json

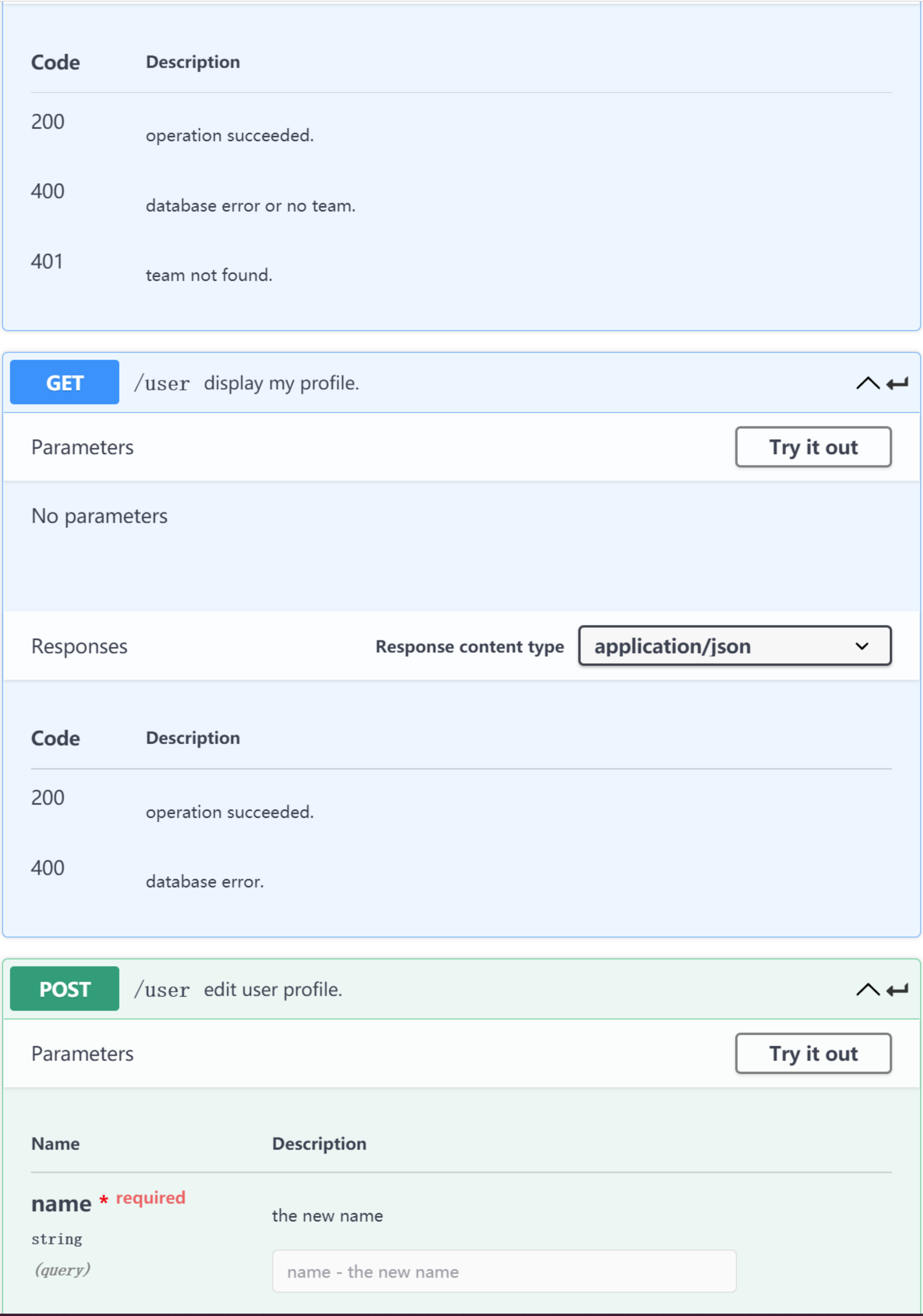
Code	Description
200	operation succeeded.
404	page not found.

Logged In User Operations solely for logged in users.

GET /displayteam display all users of my company.

Parameters

No parameters



Code	Description
200	operation succeeded.
400	database error or no team.
401	team not found.

/user display my profile.

Parameters

No parameters

Responses

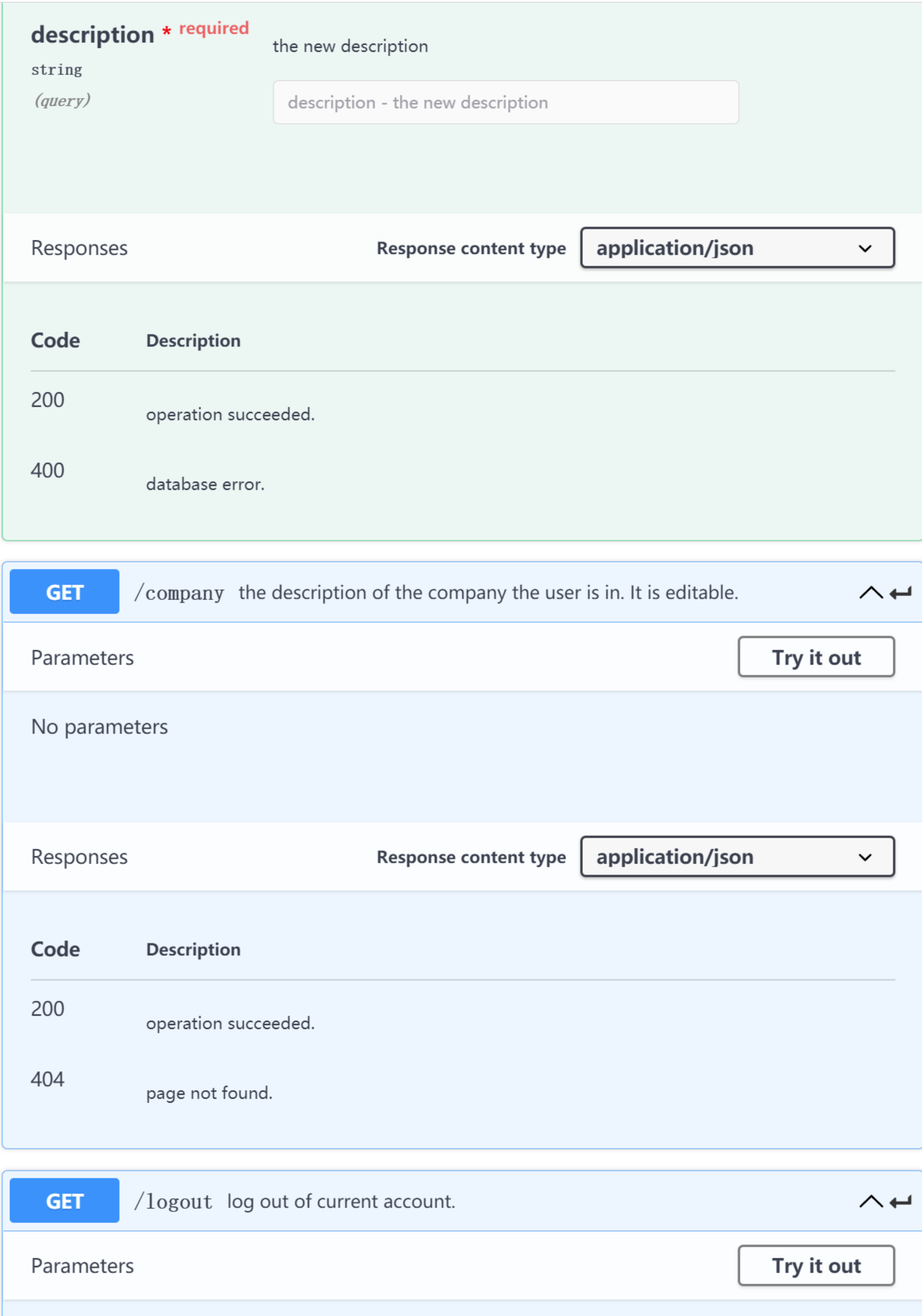
Response content type: application/json

Code	Description
200	operation succeeded.
400	database error.

/user edit user profile.

Parameters

Name	Description
name * required string *(query)*	the new name name - the new name



description * required
string
(query)

the new description

description - the new description

Responses

Response content type application/json

Code	Description
200	operation succeeded.
400	database error.

GET `/company` the description of the company the user is in. It is editable.

Parameters

No parameters

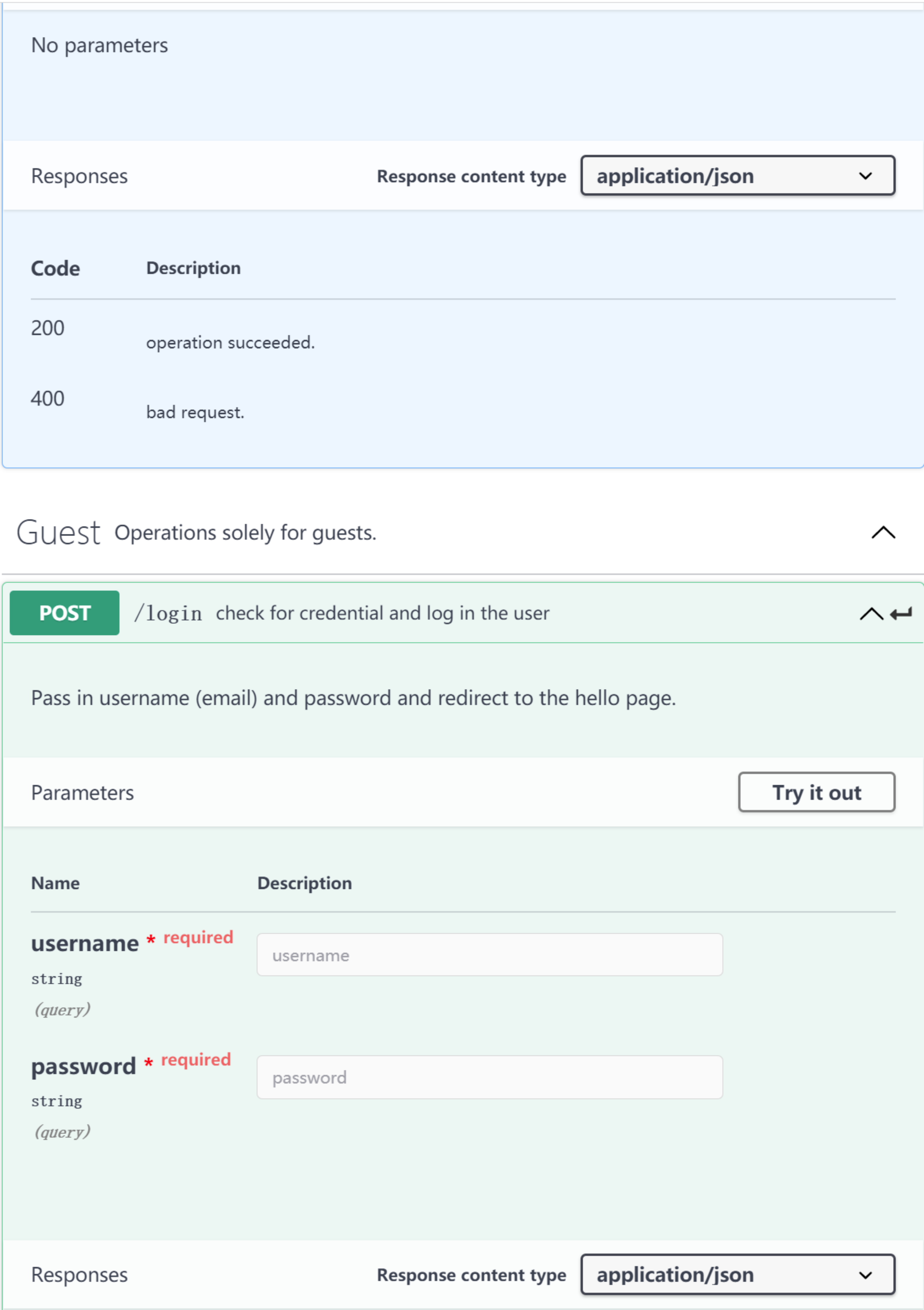
Responses

Response content type application/json

Code	Description
200	operation succeeded.
404	page not found.

GET `/logout` log out of current account.

Parameters



No parameters

Responses

Code	Description
200	operation succeeded.
400	bad request.

Guest Operations solely for guests.

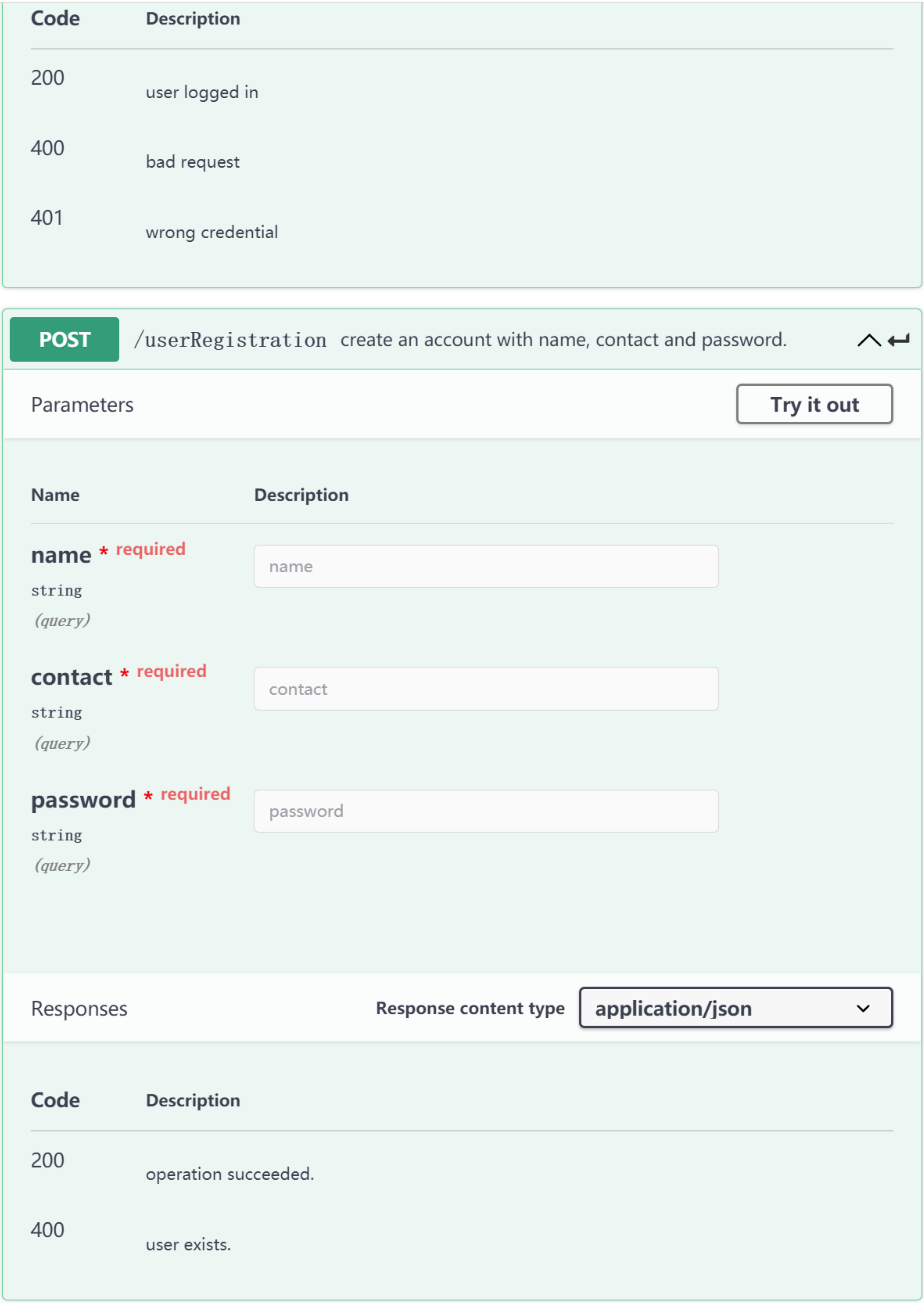
POST /login check for credential and log in the user

Pass in username (email) and password and redirect to the hello page.

Parameters

Name	Description
username * required string *(query)*	username
password * required string *(query)*	password

Responses



Code	Description
200	user logged in
400	bad request
401	wrong credential

POST /userRegistration create an account with name, contact and password.

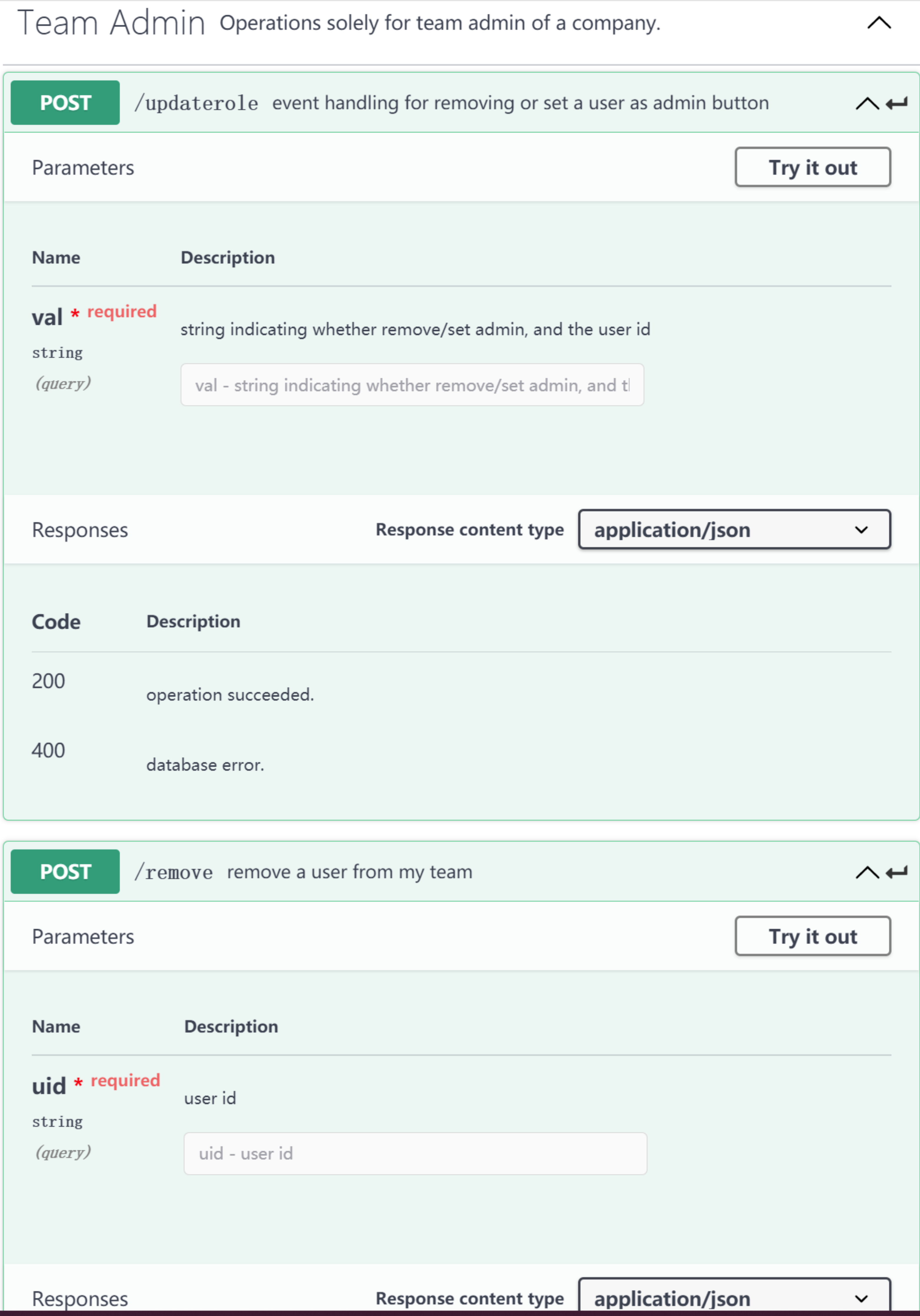
Parameters

Name	Description
name * required string *(query)*	name
contact * required string *(query)*	contact
password * required string *(query)*	password

Responses

Response content type: application/json

Code	Description
200	operation succeeded.
400	user exists.



Team Admin Operations solely for team admin of a company.

POST /updaterole event handling for removing or set a user as admin button

Parameters

Name	Description
val * required string *(query)*	string indicating whether remove/set admin, and the user id val - string indicating whether remove/set admin, and t

Responses

Response content type: application/json

Code	Description
200	operation succeeded.
400	database error.

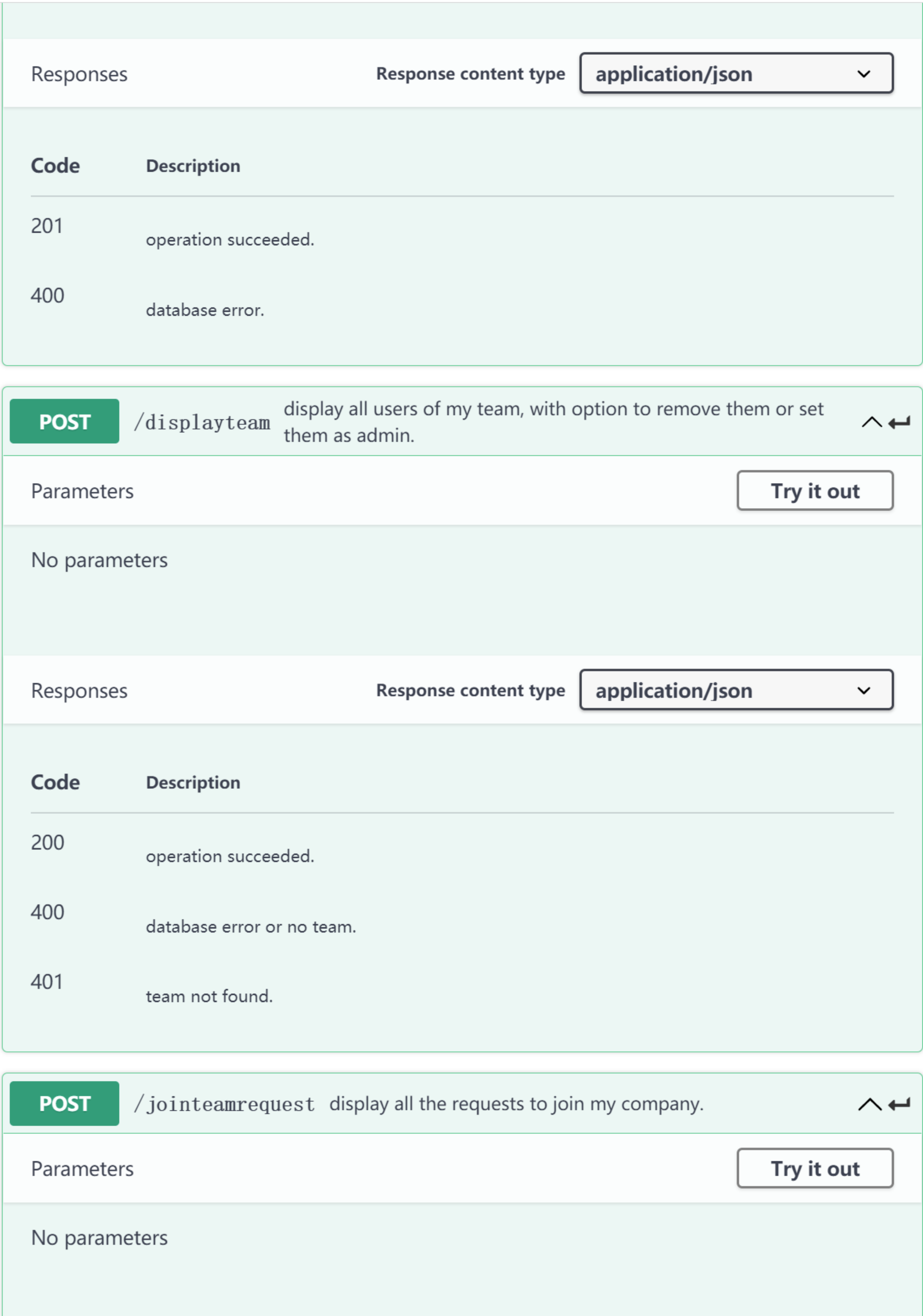
POST /remove remove a user from my team

Parameters

Name	Description
uid * required string *(query)*	user id uid - user id

Responses

Response content type: application/json



Code	Description
201	operation succeeded.
400	database error.

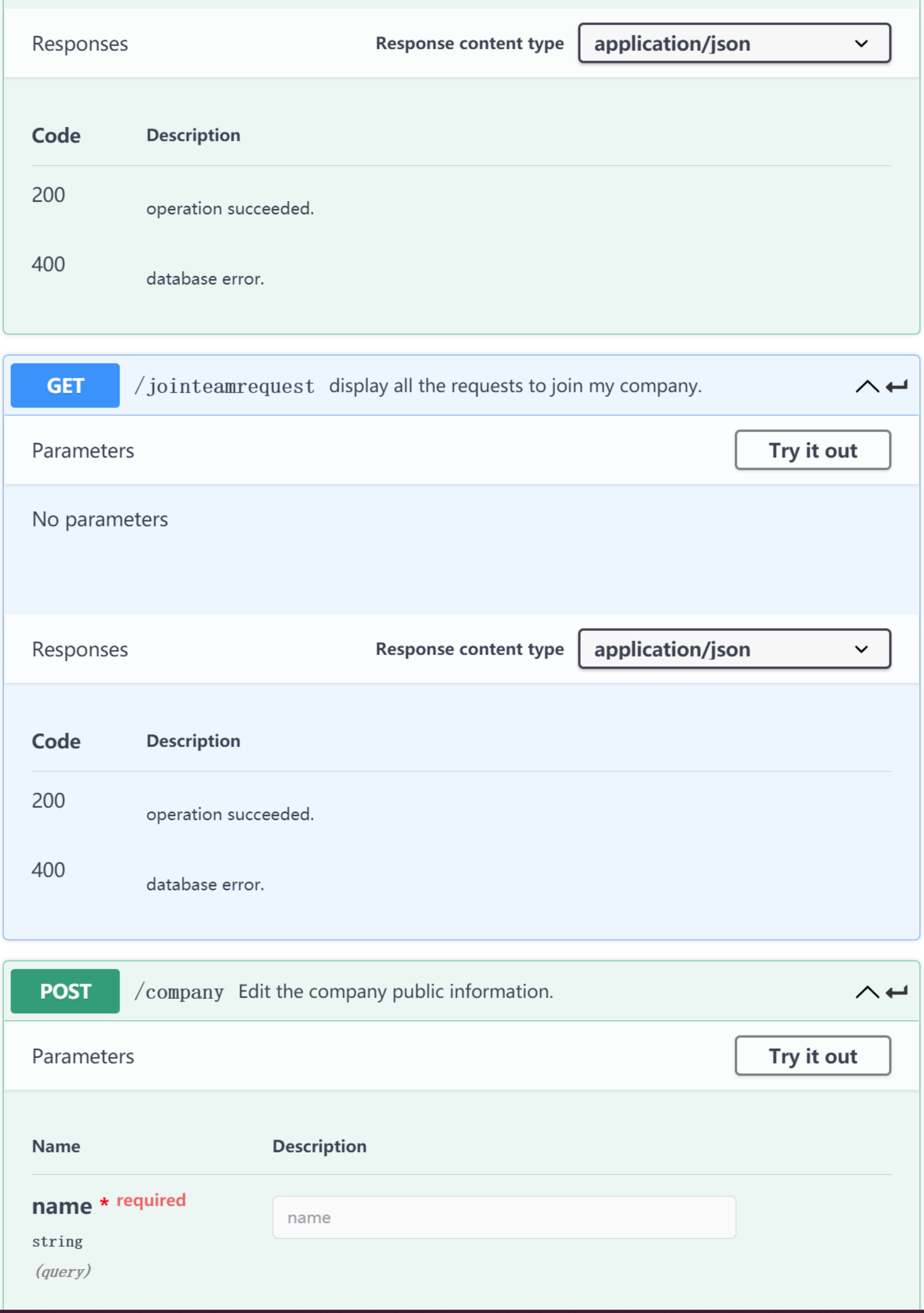
/displayteam display all users of my team, with option to remove them or set them as admin.

No parameters

Code	Description
200	operation succeeded.
400	database error or no team.
401	team not found.

/jointeamrequest display all the requests to join my company.

No parameters



Responses

Code	Description
200	operation succeeded.
400	database error.

GET /jointeamrequest display all the requests to join my company.

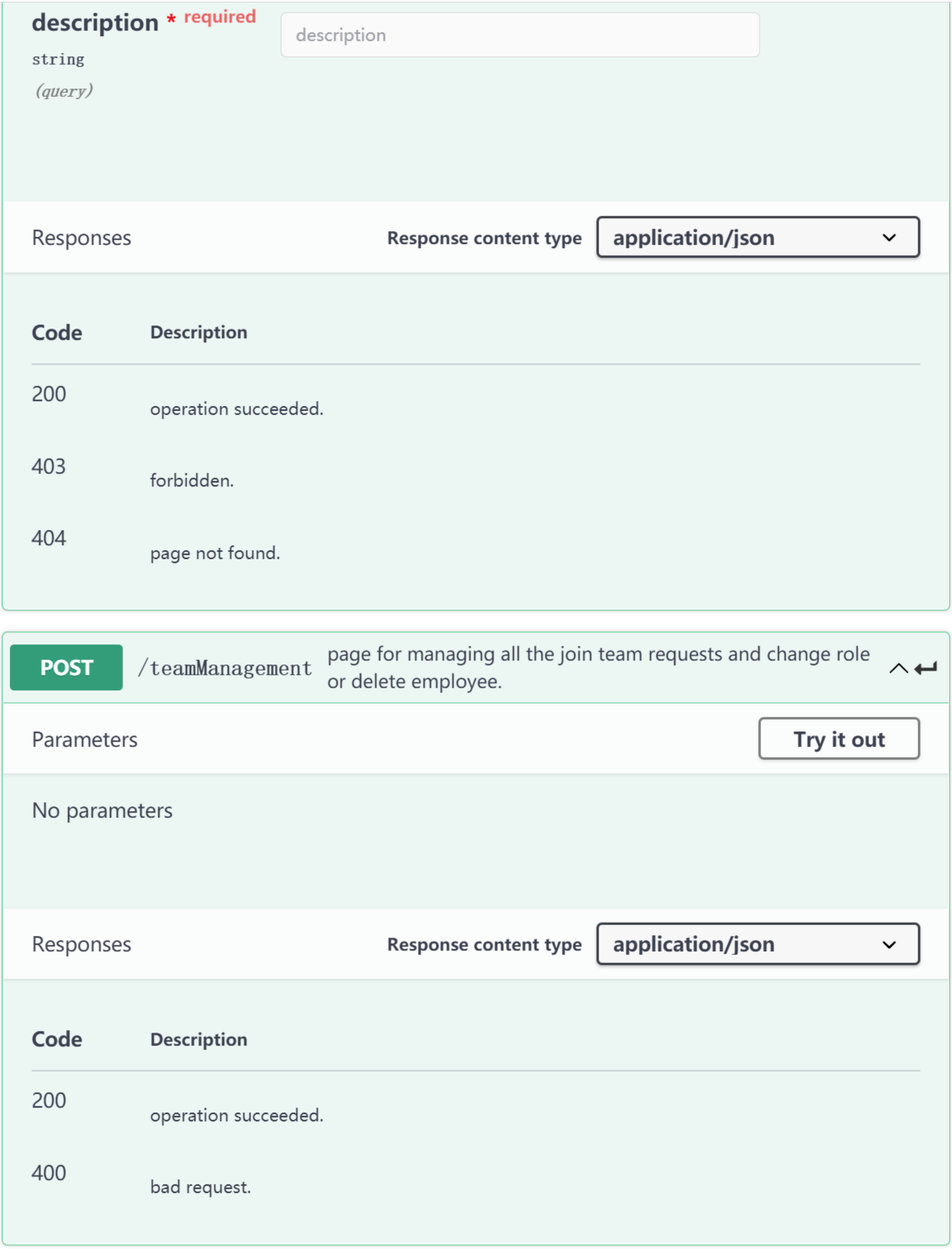
No parameters

Responses

Code	Description
200	operation succeeded.
400	database error.

POST /company Edit the company public information.

Name	Description
name * required string *(query)*	name



description * required
string
(query)

Responses

Code	Description
200	operation succeeded.
403	forbidden.
404	page not found.

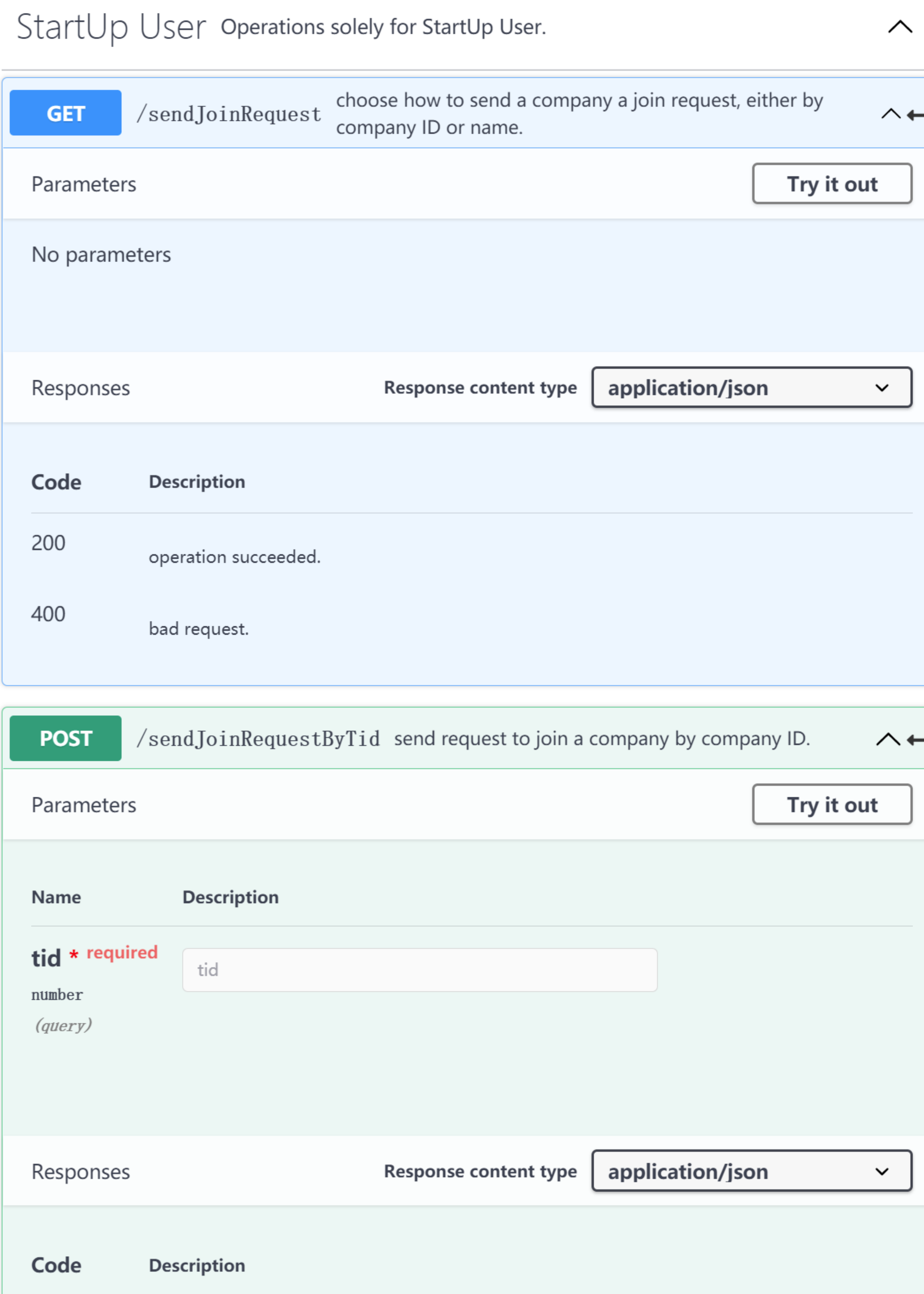
POST /teamManagement page for managing all the join team requests and change role or delete employee.

Parameters

No parameters

Responses

Code	Description
200	operation succeeded.
400	bad request.



StartUp User Operations solely for StartUp User.

GET `/sendJoinRequest` choose how to send a company a join request, either by company ID or name.

Parameters

No parameters

Responses

Response content type: application/json

Code	Description
200	operation succeeded.
400	bad request.

POST `/sendJoinRequestByTid` send request to join a company by company ID.

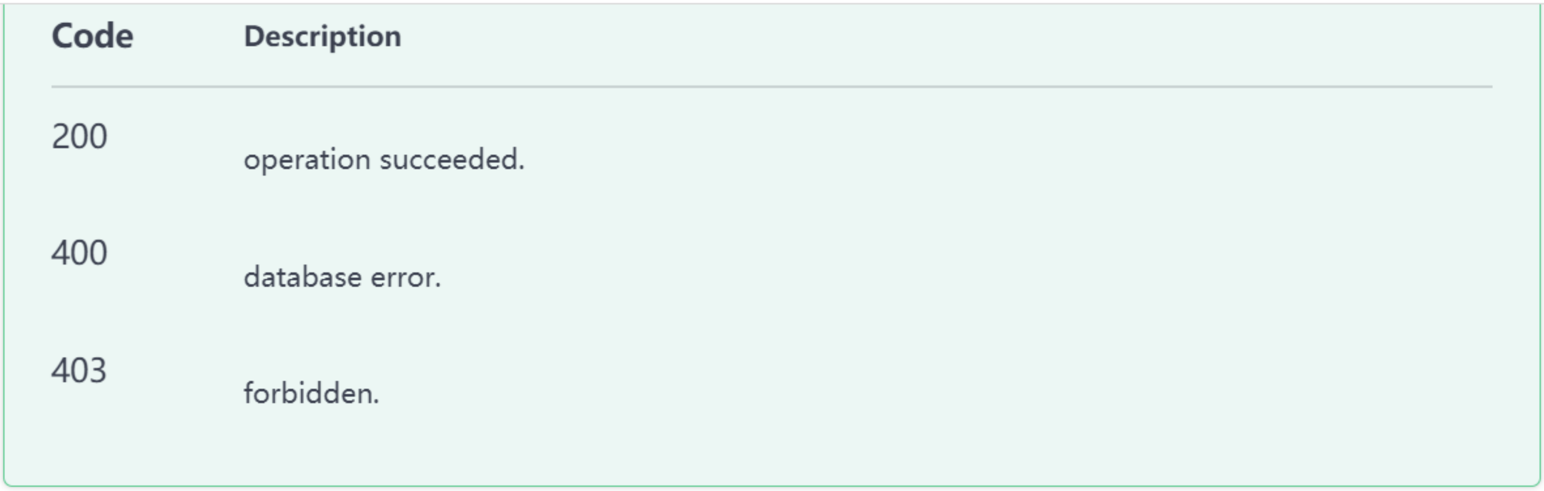
Parameters

Name	Description
tid * required number *(query)*	tid

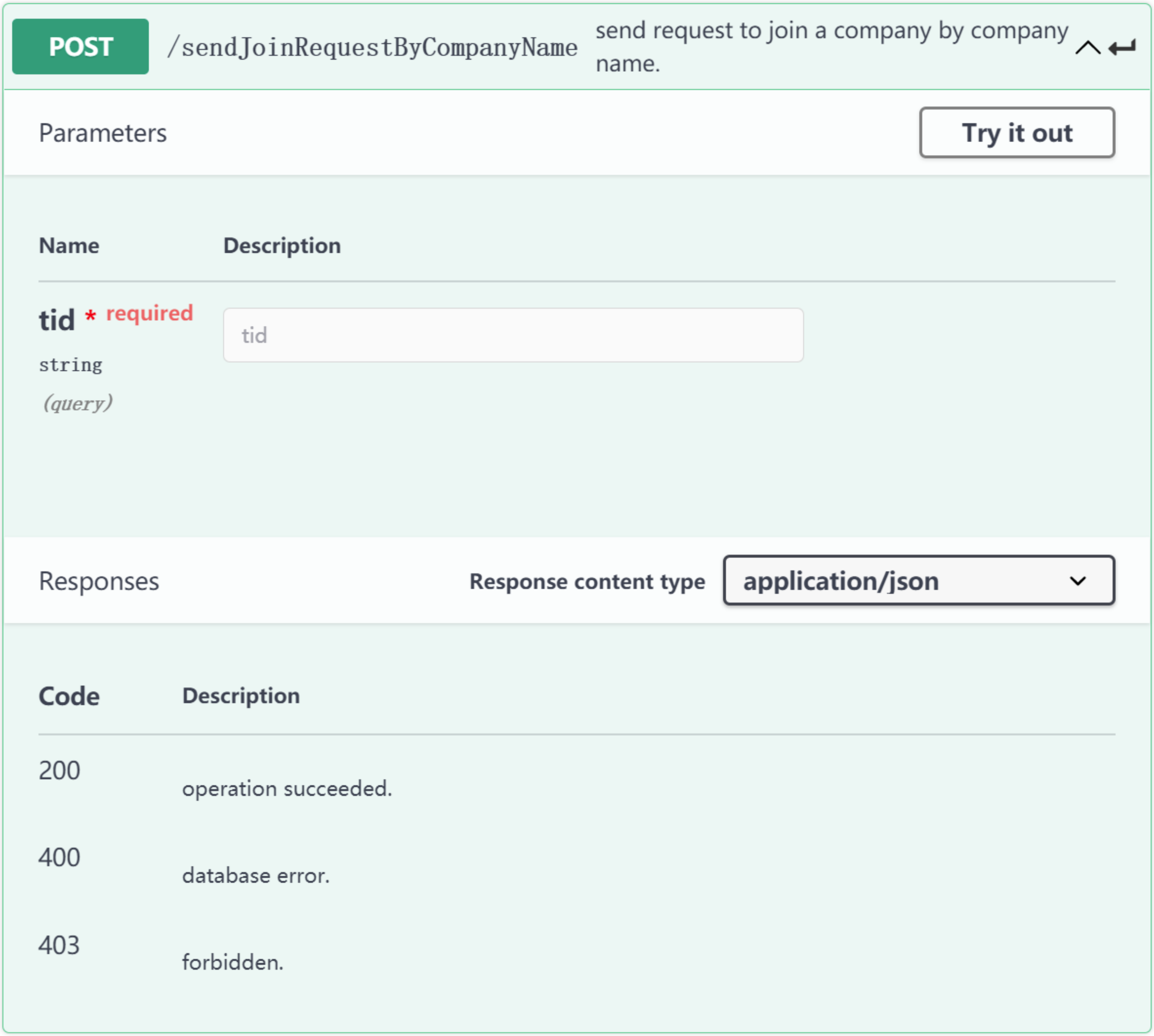
Responses

Response content type: application/json

Code	Description



Code	Description
200	operation succeeded.
400	database error.
403	forbidden.



/sendJoinRequestByCompanyName send request to join a company by company name.

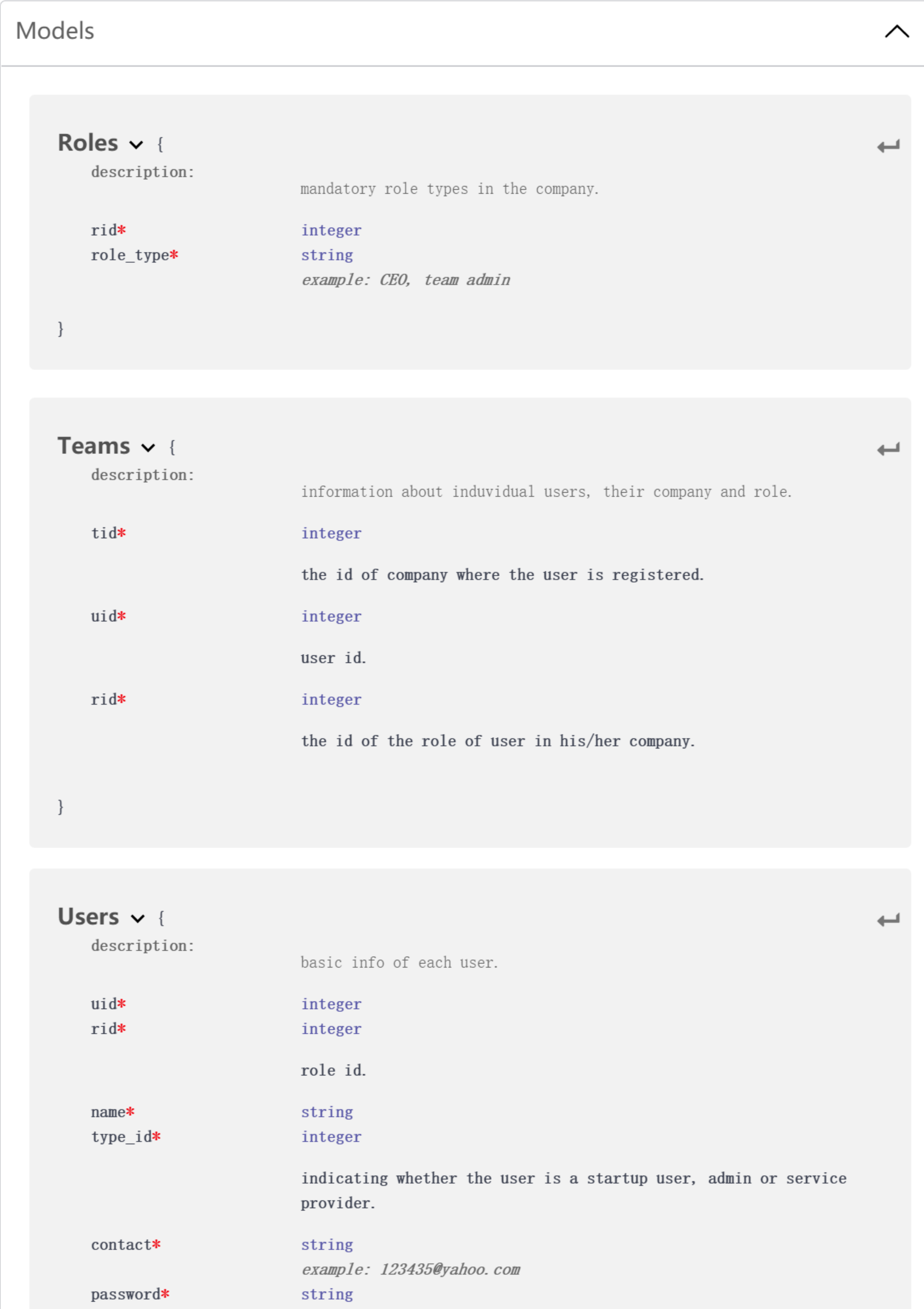
Parameters

Name	Description
tid * required string *(query)*	tid

Responses

Response content type application/json

Code	Description
200	operation succeeded.
400	database error.
403	forbidden.



Models

```
Roles ⌄ {
    description:
                         mandatory role types in the company.

    rid*                 integer
    role_type*           string
                         example: CEO, team admin

}
```

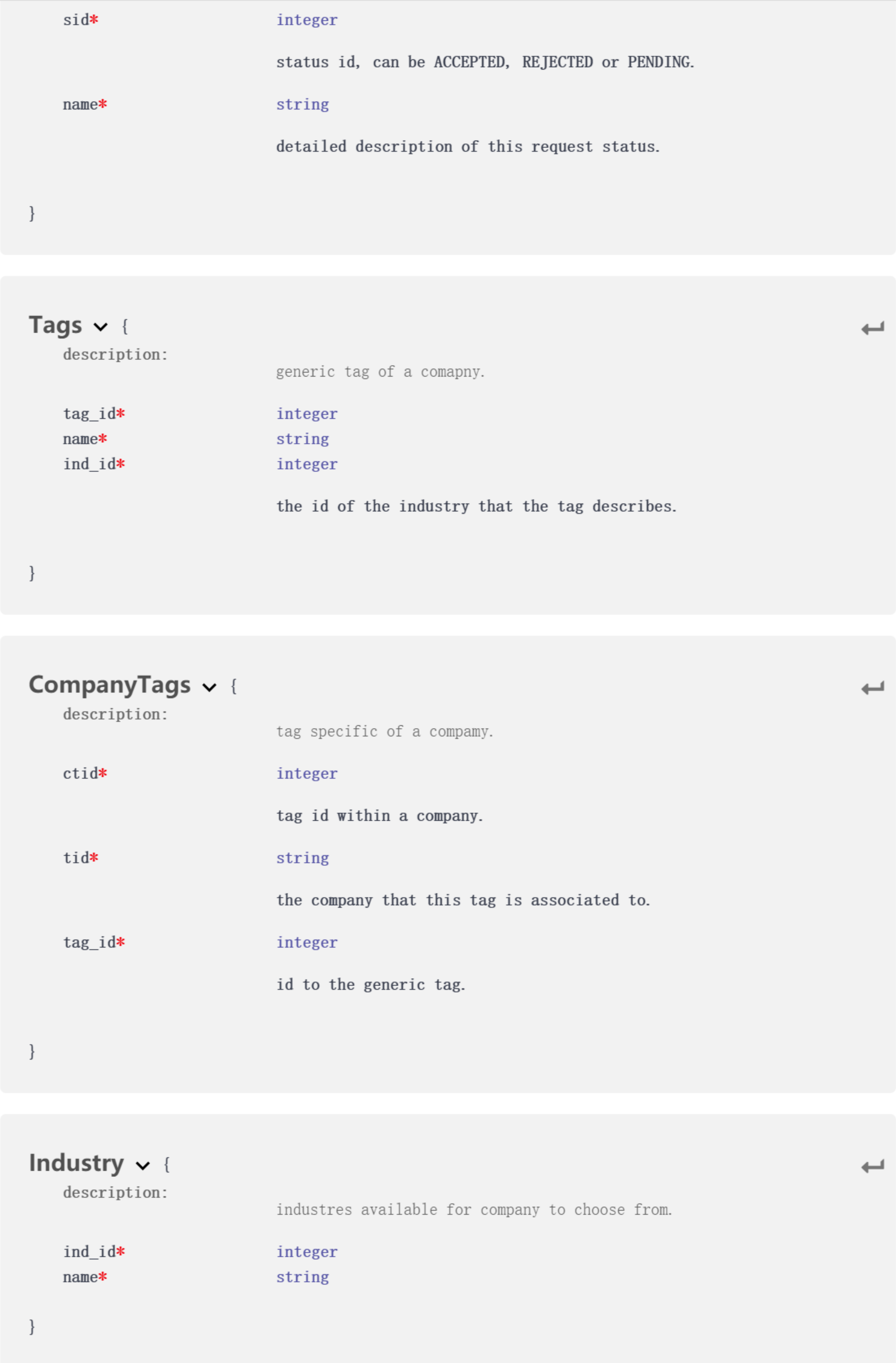
```
Teams ⌄ {
    description:
                         information about induvidual users, their company and role.

    tid*                 integer

                         the id of company where the user is registered.

    uid*                 integer

                         user id.

    rid*                 integer

                         the id of the role of user in his/her company.


}
```

```
Users ⌄ {
    description:
                         basic info of each user.

    uid*                 integer
    rid*                 integer

                         role id.

    name*                string
    type_id*             integer

                         indicating whether the user is a startup user, admin or service
                         provider.

    contact*             string
                         example: 123435@yahoo.com
    password*            string
    password*               string
    description*            string

}
```

```
Company ∨ {
    description:
                            basic info of each company.

    tid*                    integer

                            company id.

    name*                   string
    description*            string
    ind_id*                 integer

                            industry id.

    create_date*            number($date-time)

}
```

```
Request ∨ {
    description:
                            complete info of each request sent by users to a company.

    req_id*                 integer
    uid*                    integer

                            id of sender user.

    tid*                    integer

                            id of the receiver company.

    sid*                    integer

                            status id, can be ACCEPTED, REJECTED or PENDING.

    create_date*            number($date-time)
    last_update*            number($date-time)
    seen*                   boolean

}
```

```
Rstatus ∨ {
    description:
                            request status.

    sid*                    integer
```

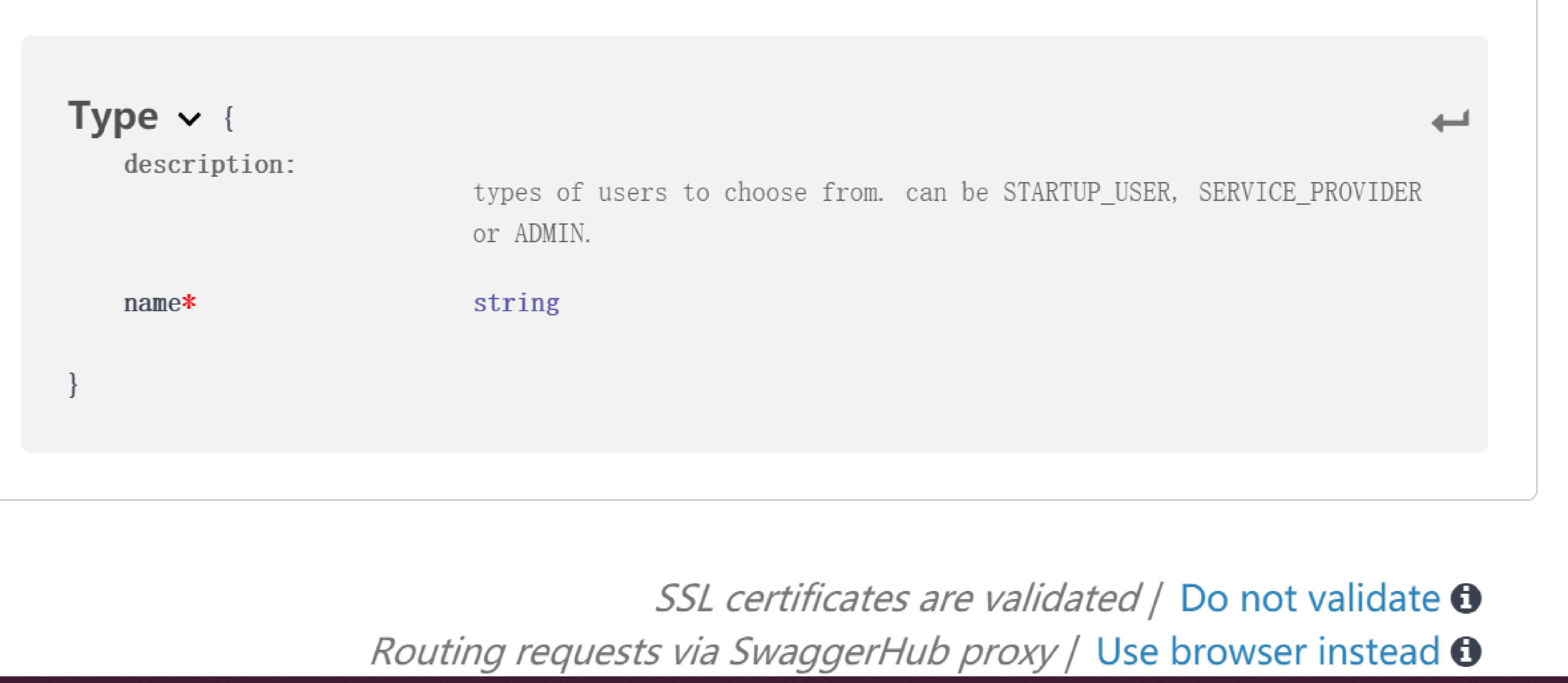


```
    sid*                    integer

                            status id, can be ACCEPTED, REJECTED or PENDING.

    name*                   string

                            detailed description of this request status.

}
```

```
Tags {
    description:
                            generic tag of a comapny.

    tag_id*                 integer
    name*                   string
    ind_id*                 integer

                            the id of the industry that the tag describes.

}
```

```
CompanyTags {
    description:
                            tag specific of a compamy.

    ctid*                   integer

                            tag id within a company.

    tid*                    string

                            the company that this tag is associated to.

    tag_id*                 integer

                            id to the generic tag.

}
```

```
Industry {
    description:
                            industres available for company to choose from.

    ind_id*                 integer
    name*                   string

}
```



```
Type ⌄ {
    description:
                        types of users to choose from． can be STARTUP_USER， SERVICE_PROVIDER
                        or ADMIN．

    name*               string

}
```

SSL certificates are validated / Do not validate ℹ

Routing requests via SwaggerHub proxy / Use browser instead ℹ