Project API Specification

This document contains the documentation for the REST API on the Mjecipe server. It should contain enough information for both implementing and consuming it.

The resources

The most important parts of the resources stored on the server, and their relationships, are summarized in the class diagram in Figure 1 below.

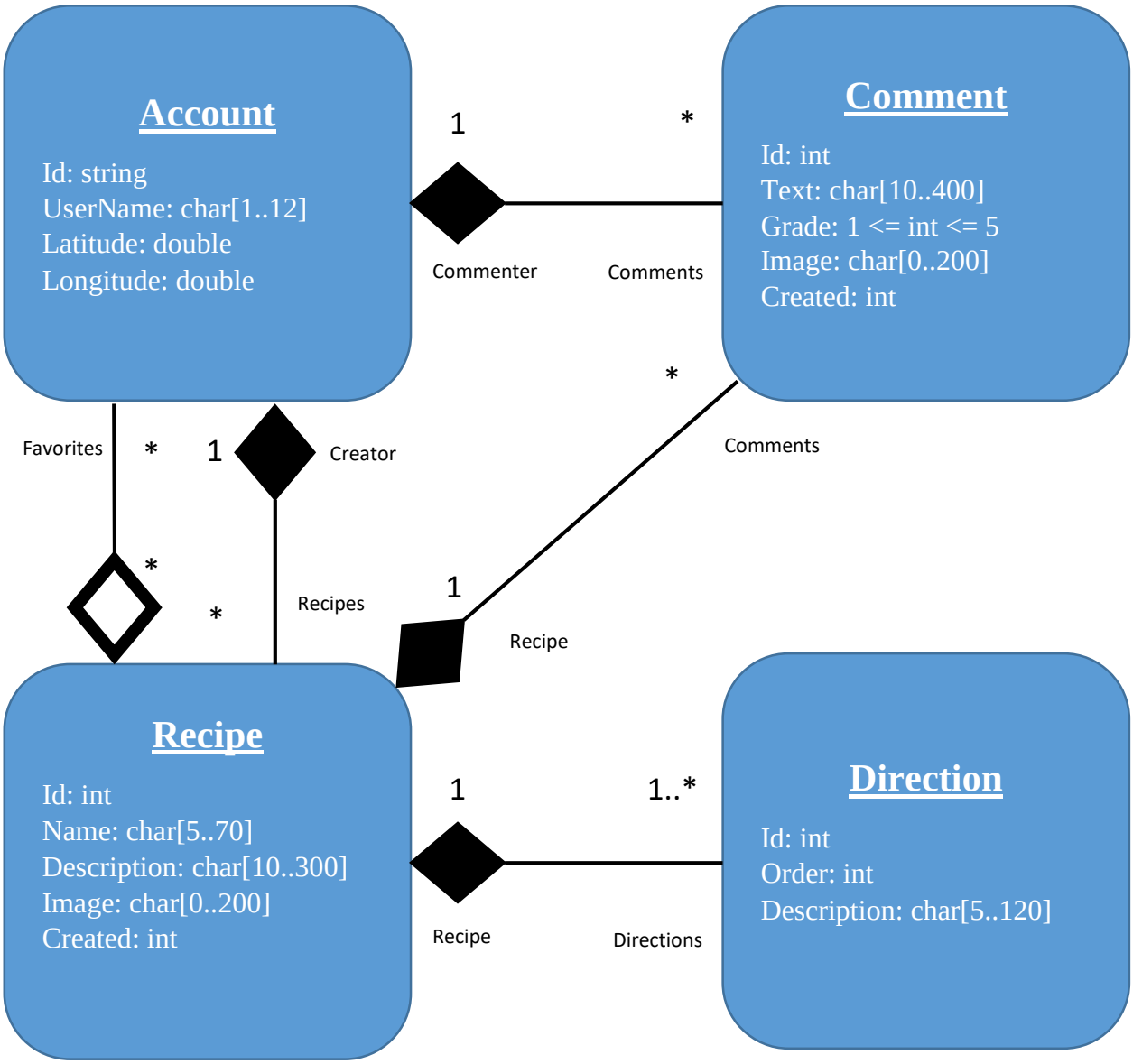

Figure 1, Class diagram of the resources and their relationships. Char[X..Y] simply means a string consisting of X to Y characters.

Account
Represents a user. Authentication is carried out through passwords or Facebook Login. Users can create new recipes and post comments on recipes. They can also mark recipes as one of their favorites (optional for grade 3).

Recipe
Represents a recipe. Consists of a sequence of directions. The creator of the recipe can also upload a jpeg image of the cooked recipe. The `Created` field stores a Unix timestamp from when the recipe was created. The `Image` field stores the path to the image.

Direction
A description of one of the steps in the process of cooking the recipe. The `Order` field keeps track of in which order one should execute the instructions (ascending)

Comment
Users feedback on a recipe. Consists of text and a grade (1, 2, 3, 4 or 5), but it is also possible to upload a jpeg image. The `Created` field stores a Unix timestamp from when the comment was created. The `Image` field stores the path to the image.

The REST API
The base path to the api should be `/api/v1`. A summary of the REST API can be found in Table 1 below. Using Facebook for authentication is optional (an individual assignment for higher grade).

Method	URI	Explanation
POST	`/accounts/password`	Creates a new account with password authentication.
POST	`/accounts/facebook`	Creates a new account using a token from Facebook.
GET	`/accounts/{id}`	Returns information about the account with the given `{id}`.
PATCH	`/accounts/{id}`	Updates the account with the given `{id}`.
DELETE	`/accounts/{id}`	Deletes the accounts with the given `{id}`.
GET	`/accounts/{id}/recipes`	Returns all recipes the account with the given `{id}` has created.
GET	`/accounts/{id}/comments`	Returns all comments made by the account with the given `{id}`.
PUT	`/accounts/{id}/favorites`	Updates the recipes marked as favorites for the account with the given `{id}`.
GET	`/accounts/{id}/favorites`	Returns the recipes the account with the given `{id}` has marked as favorites.
POST	`/tokens/password`	Creates a new token from a username and password.
POST	`/tokens/facebook`	Creates a new token from a Facebook token.
POST	`/recipes`	Creates a new recipe.
GET	`/recipes?page={page}`	Returns information about the most recently added recipes on page {page} (10 recipes per page).
GET	`/recipes/{id}`	Returns information about the recipe with the given `{id}`.
DELETE	`/recipes/{id}`	Deletes the recipe with the given `{id}`.

PATCH	`/recipes/{id}`	Updates the recipe with the given `{id}`.
PUT	`/recipes/{id}/image`	Uploads an image to the recipe.
POST	`/recipes/{id}/comments`	Creates a comment for the recipe with the given `{id}`.
GET	`/recipes/{id}/comments`	Returns all comments for the recipe with the given `{id}`.
GET	`/recipes/search?term=X`	Returns recipes containing the given search `{term}`.
PATCH	`/comments/{id}`	Updates the comment with the given `{id}`.
DELETE	`/comments/{id}`	Deletes the comment with the given `{id}`.
PUT	`/comments/{id}/image`	Uploads an image to the comment with the given `{id}`.

Table 1, Summary of the REST API. The rows with gray background are optional for grade 3.

A REST API built on top of HTTP should make use of the different available HTTP status codes. However, the more status codes we are using, the harder it will be to implement the API, and the API will also be harder to learn and use (the documentation becomes more complex). As a result, one should try to use as few different status codes as possible. For example, Facebook's Graph API does take this to the extreme and always return the status code 200, even if the request couldn't be fulfilled. This can be compared with Twitter's API, which includes 15 different status codes.

Our REST API will only be used internally by ourselves, so we have the freedom to design it however we please. For simplicity we will only use the six different status codes as explained in Table 2 below.

Status Code	Description
200 OK	The request was successfully fulfilled. The response contains an object.
201 Created	The request was successfully fulfilled. The created resource can be found at the location stored in the `Location` header in the response.
204 No Content	The request was successfully fulfilled, but there is no content to return.
401 Unauthorized	The request has not been fulfilled since the client is not authorized to request it.
404 Not Found	The request has not been fulfilled since the requested resource doesn't exist.
400 Bad Request	The request has not been fulfilled. Error codes can be found in body of the response (this is a "catch all other errors" response to be used when 401 and 404 are inappropriate).

Table 2, The status codes the REST API is using.

For authorization we will use the `Authorization` header in the requests. The token to be passed here can be retrieved by posting the user's username and password to `/tokens/password`.

Create a new account (using username & password)

Creates a new account with the given username and password.

Request template

```
POST /api/v1/accounts HTTP/1.1
Host: 11.12.21.22
Accept: application/json
Content-Type: application/json
Content-Length: XXX

{"userName": "MemberA", "password": "memberA1!",
 "longitude": 0, "latitude": 0}
```

Possible response templates

201 Created

```
HTTP/1.1 201 Created
Location: /api/accounts/abc...def
Content-Length: XXX

{"id": "abc...def", "userName": "MemberA",
 "longitude": 0, "latitude": 0}
```

400 Bad Request

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Content-Length: XXX

{"errors": ["ErrorCode1", ...]}
```

Possible error codes:

- `UserNameMissing`
- `InvalidUserName`
- `DuplicateUserName`
- `PasswordMissing`
- `PasswordTooShort`
- `PasswordRequiresNonAlphanumeric`
- `PasswordRequiresDigit`
- `PasswordRequiresLower`
- `PasswordRequiresUpper`
- `LongitudeMissing`
- `LatitudeMissing`

Create a new account (using a Facebook token)

Creates a new account with the given username connected to the Facebook account with the given token.

Request template

```
POST /api/v1/accounts HTTP/1.1
Host: 11.12.21.22
Accept: application/json
Content-Type: application/json
Content-Length: XXX

{"userName": "MemberA", "token": "af...jk", "longitude": 0, "latitude": 0}
```

Possible response templates

201 Created

```
HTTP/1.1 201 Created
Location: /api/accounts/abc...def
```

400 Bad Request

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Content-Length: XXX

{"errors": ["ErrorCode1", ...]}
```

Possible error codes:

- `UserNameMissing`
- `InvalidUserName`
- `DuplicateUserName`
- `TokenMissing`
- `TokenInvalid`
- `LongitudeMissing`
- `LatitudeMissing`

Get information about an account

Returns information about the account with the given `{id}` (from the URI).

Request template

```
GET /api/v1/accounts/{id} HTTP/1.1
Host: 11.12.21.22
Accept: application/json
```

Possible response templates

200 OK

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: XXX

{"id": "abc...def", "userName": "MemberA", "longitude": 0, "latitude": 0}
```

404 Not Found

```
HTTP/1.1 404 Not Found
```

Update information about an account

Updates the information for the account with the given `{id}` (from the URI). It should only be possible to update your own account. It should only be possible to update the fields `Longitude` and `Latitude`.

Request template

```
PATCH /api/v1/accounts/{id} HTTP/1.1
Host: 11.12.21.22
Authorization: Bearer 2YotnFZ.FEjr1zC.sicMWpAA
Content-Type: application/json
Content-Length: XXX

{"longitude": 14.1618, "latitude": 57.7826}
```

Possible response templates

204 No Content

```
HTTP/1.1 204 No Content
```

401 Unauthorized

```
HTTP/1.1 401 Unauthorized
```

404 Not Found

```
HTTP/1.1 404 Not Found
```

Delete an account

Deletes the account with the given `{id}` (from the URI). This will also delete:

- All comments created by the account.
- All recipes created by the account and
    - All comments on those recipes (including those posted from other accounts).

It should only be possible to delete your own account.

Request template

```
DELETE /api/v1/accounts/{id} HTTP/1.1
Host: 11.12.21.22
Authorization: Bearer 2YotnFZ.FEjr1zC.sicMWpAA
```

Possible response templates

204 No Content

```
HTTP/1.1 204 No Content
```

401 Unauthorized

```
HTTP/1.1 401 Unauthorized
```

404 Not Found

```
HTTP/1.1 404 Not Found
```

Get a user's recipes

Returns all recipes created by the user with the given `{id}` (from the URI).

Request template

```
GET /api/v1/accounts/{id}/recipes HTTP/1.1
Host: 11.12.21.22
Accept: application/json
```

Possible response templates

200 OK

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: XXX

[{
  "id": 1,
  "name": "Recipe A",
  "image": "the-path/filename.jpeg",
  "created": 1471962685
}, ...]
```

404 Not Found

```
HTTP/1.1 404 Not Found
```

Get a user's comments

Returns all comments created by the account with the given `{id}` (from the URI).

Request template

```
GET /api/v1/accounts/{id}/comments HTTP/1.1
Host: 11.12.21.22
Accept: application/json
```

Possible response templates

200 OK

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: XXX

[{
  "id": 1,
  "recipeId": 1,
  "text": "The comment",
  "grade": 3,
  "image": "the-path/filename.jpeg",
  "created": 1471962687
}, ...]
```

404 Not Found

```
HTTP/1.1 404 Not Found
```

Update a user's favorite recipes

Changes the favorite recipes associated with the account with the given `{id}` (from the URI). It should only be possible to update the favorite recipes in your own account. The `id`s sent to the server are the `id`s of the recipes.

Request template

```
PUT /api/v1/accounts/{id}/favorites HTTP/1.1
Host: 11.12.21.22
Authorization: Bearer 2YotnFZ.FEjr1zC.sicMWpAA
Accept: application/json
Content-Type: application/json
Content-Length: XXX

[{
  "id": 1
}, ...]
```

Possible response templates

204 No Content

```
HTTP/1.1 204 No Content
```

401 Unauthorized

```
HTTP/1.1 402 Unauthorized
```

404 Not Found

```
HTTP/1.1 404 Not Found
```

400 Bad Request

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Content-Length: XXX

{"errors": ["ErrorCode1", ...]}
```

Possible error codes:

- `RecipeIdDoesNotExist`

Get a user's favorite recipes

Returns the favorite recipes associated with the account with the given `{id}` (from the URI). It should only be possible to see your own favorites.

Request template

```
GET /api/v1/accounts/{id}/favorites HTTP/1.1
Host: 11.12.21.22
Authorization: Bearer 2YotnFZ.FEjr1zC.sicMWpAA
Accept: application/json
```

Possible response templates

200 OK

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: XXX

[{
  "id": 1,
  "name": "Recipe A",
  "image": "the-path/filename.jpeg",
  "created": 1471962694
}, ...]
```

401 Unauthorized

```
HTTP/1.1 402 Unauthorized
```

404 Not Found

```
HTTP/1.1 404 Not Found
```

Create a new token (using username and password)

Creates a new token that represents the user with the provided username and password (as described in the OAuth 2.0 specification: https://tools.ietf.org/html/rfc6749#section-4.3).

Request template

```
POST /api/v1/tokens/password HTTP/1.1
Host: 11.12.21.22
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Content-Length: XXX

grant_type=password&username=MemberA&password=memberA1!
```

Possible response templates

200 Ok

```
HTTP/1.1 200 Ok
Content-Type: application/json
Content-Length: XXX

{"access_token": "2YotnFZFEjr1zCsicMWpAA", "expires_in": 3600}
```

400 Bad Request

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Content-Length: XXX

{"error": "ErrorCode"}
```

Possible error codes:

- `invalid_request`
- `unsupported_grant_type`
- `invalid_client`

Create a new token (using Facebook token)

Creates a new token that represents the user of the account connected to the Facebook user with the provided token (as described in the OAuth 2.0 specification: https://tools.ietf.org/html/rfc6749#section-4.5).

Request template

```
POST /api/v1/tokens/facebook HTTP/1.1
Host: 11.12.21.22
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Content-Length: XXX

grant_type=mjecipes.com/grants/facebook&token= af...jk
```

Possible response templates

200 Ok

```
HTTP/1.1 200 Ok
Content-Type: application/json
Content-Length: XXX

{"access_token": "2YotnFZFEjr1zCsicMWpAA", "expires_in": 3600}
```

400 Bad Request

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Content-Length: XXX

{"error": "ErrorCode"}
```

Possible error codes:

- `invalid_request`
- `unsupported_grant_type`
- `invalid_client`

Create new recipe

Creates a new recipe. It should only be possible to create recipes for your own account.

Request template

```
POST /api/v1/recipes HTTP/1.1
Host: 11.12.21.22
Authorization: Bearer 2YotnFZ.FEjr1zC.sicMWpAA
Accept: application/json
Content-Type: application/json
Content-Length: XXX

{
  "name": "The name",
  "description": "The description.",
  "creatorId": " abc...def",
  "directions": [{
    "order": 1,
    "description": "Direction description 1"
  }, ...]
}
```

Possible response templates

201 Created

```
HTTP/1.1 201 Created
Location: ...
```

401 Unauthorized

```
HTTP/1.1 401 Unauthorized
```

400 Bad Request

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Content-Length: XXX

{"errors": ["ErrorCode1", ...]}
```

Possible error codes:

- `NameMissing`
- `NameWrongLength`
- `DescriptionMissing`
- `DescriptionWrongLength`

- `DirectionsMissing`
- `DirectionOrderMissing`
- `DirectionDescriptionMissing`
- `DirectionDescriptionWrongLength`

Get the most recent added recipes

Returns general information about the most recent added recipes. The result is spread out on different pages, pass the page number in the query string parameter `page` (starts on 1).

Request template

```
GET /api/v1/recipes?page=1 HTTP/1.1
Host: 11.12.21.22
Accept: application/json
```

Possible response templates

200 OK

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: XXX

[{
  "id": 1,
  "name": "Recipe A",
  "image": "the-path/filename.jpeg",
  "created": 1472022409
}, ...]
```

404 Not Found

Used for invalid page numbers.

```
HTTP/1.1 404 Not Found
```

Get information about a specific recipe

Returns information about the recipe with the given `{id}` (from the URI).

Request template

```
GET /api/v1/recipes/{id} HTTP/1.1
Host: 11.12.21.22
Accept: application/json
```

Possible response templates

200 OK

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: XXX

{
  "id": 1,
  "name": "Recipe A",
  "description": " The description.",
  "creator": {
    "id": "abc...def",
    "userName": "MemberA"
  }
  "image": "the-path/filename.jpeg",
  "created": 1472022409,
  "directions": [{
    "order": 1,
    "description": "The description."
  }, ...]
}
```

404 Not Found

```
HTTP/1.1 404 Not Found
```

Delete a recipe

Deletes the recipe with the given `{id}` (from the URI). One should only be able to delete your own recipes. The comments on this recipe should be deleted as well.

Request template

```
DELETE /api/v1/recipes/{id} HTTP/1.1
Host: 11.12.21.22
Authorization: Bearer 2YotnFZ.FEjr1zC.sicMWpAA
```

Possible response templates

204 No Content

```
HTTP/1.1 204 No Content
```

401 Unauthorized

```
HTTP/1.1 401 Unauthorized
```

404 Not Found

```
HTTP/1.1 404 Not Found
```

Update a recipe

Updates the recipe with the given `{id}` (from the URI). One should only be able to update your own recipes.

Request template

```
PATCH /api/v1/recipes/{id} HTTP/1.1
Host: 11.12.21.22
Authorization: Bearer 2YotnFZ.FEjr1zC.sicMWpAA
Accept: application/json
Content-Type: application/json
Content-Length: XXX

{
  "name": "The new name",
  "description": "The new description...",
  "directions": [{
    "order": 1,
    "description": "The new description..."
  }, ...]
}
```

Possible response templates

204 No Content

```
HTTP/1.1 204 No Content
```

401 Unauthorized

```
HTTP/1.1 401 Unauthorized
```

404 Not Found

```
HTTP/1.1 404 Not Found
```

400 Bad Request

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Content-Length: XXX

{"errors": ["ErrorCode1", ...]}
```

Possible error codes:

- `NameWrongLength`

- `DescriptionWrongLength`
- `DirectionOrderMissing`
- `DirectionDescriptionMissing`
- `DirectionDescriptionWrongLength`

Upload image to a recipe

Uploads an image to recipe with the given `{id}` (from the URI). One should only be able to upload images to your own recipes.

Request template

```
PUT /api/v1/recipes/{id}/image HTTP/1.1
Host: 11.12.21.22
Authorization: Bearer 2YotnFZ.FEjr1zC.sicMWpAA
Content-Type: multipart/form-data;boundary=SOME_BOUNDARY

--SOME_BOUNDARY
Content-Disposition: form-data;name="image";filename="the-filename.jpeg"
Content-Type: image/jpeg

The image's binary content here...
--SOME_BOUNDARY--
```

Possible response templates

204 No Content

```
HTTP/1.1 204 No Content
```

401 Unauthorized

```
HTTP/1.1 401 Unauthorized
```

404 Not Found

```
HTTP/1.1 404 Not Found
```

Create a comment on a recipe

Creates a new comment on the recipe with the given `{id}` (from the URI). One user should not be able to create multiple comments on one and the same recipe.

Request template

```
POST /api/v1/recipes/{id}/comments HTTP/1.1
Host: 11.12.21.22
Authorization: Bearer 2YotnFZ.FEjr1zC.sicMWpAA
Accept: application/json
Content-Type: application/json
Content-Length: XXX

{"text": "The comment.", "grade": 3, "commenterId": "abc...def"}
```

Possible response templates

201 Created

```
HTTP/1.1 201 Created
Location: ...
```

401 Unauthorized

```
HTTP/1.1 401 Unauthorized
```

404 Not Found

```
HTTP/1.1 404 Not Found
```

400 Bad Request

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Content-Length: XXX

{"errors": ["ErrorCode1", ...]}
```

Possible error codes:

- `TextMissing`
- `TextWrongLength`
- `GradeMissing`
- `GradeWrongValue`
- `CommenterIdMissing`

- `CommenterAlreadyComment`

Get comments on a recipe

Returns all the comments made on the recipe with the given `{id}` (from the URI).

Request template

```
GET /api/v1/recipes/{id}/comments HTTP/1.1
Host: 11.12.21.22
Accept: application/json
```

Possible response templates

200 OK

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: XXX

[{
  "id": 1,
  "text": "The comment.",
  "grade": 3,
  "commenter": {
    "id": "abc...def",
    "userName": "MemberA"
  },
  "image": "the-path/filename.jpeg",
  "created": 1472022409
}, ...]
```

404 Not Found

```
HTTP/1.1 404 Not Found
```

Search for recipes

Returns the recipes containing the search `{term}` (from the query string in the URI) in their name or description.

Request template

```
GET /api/v1/recipes/search?term={term} HTTP/1.1
Host: 11.12.21.22
Accept: application/json
```

Possible response templates

200 OK

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: XXX

[{
  "id": 1
  "name": "The name",
  "description": "The description...",
  "image": "the-path/filename.jpeg",
  "created": 1472022409
}, ...]
```

400 Bad Request

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Content-Length: XXX

{"errors": ["ErrorCode1", ...]}
```

Possible error codes:

- `TermMissing`

Update a comment

Updates the comment with the given `{id}` (from the URI). One should only be able to update your own comments.

Request template

```
PATCH /api/v1/comments/{id} HTTP/1.1
Host: 11.12.21.22
Authorization: Bearer 2YotnFZ.FEjr1zC.sicMWpAA
Accept: application/json
Content-Type: application/json
Content-Length: XXX

{"text": "The comment.", "grade": 3}
```

Possible response templates

204 No Content

```
HTTP/1.1 204 No Content
```

401 Unauthorized

```
HTTP/1.1 401 Unauthorized
```

404 Not Found

```
HTTP/1.1 404 Not Found
```

400 Bad Request

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Content-Length: XXX

{"errros": ["ErrorCode1", ...]}
```

Possible error codes:

- `TextWrongLength`
- `GradeWrongValue`

Delete a comment

Deletes the comment with the given `{id}` (from the URI). One should only be able to delete your own comments.

Request template

```
DELETE /api/v1/comments/{id} HTTP/1.1
Host: 11.12.21.22
Authorization: Bearer 2YotnFZ.FEjr1zC.sicMWpAA
```

Possible response templates

204 No Content

```
HTTP/1.1 204 No Content
```

401 Unauthorized

```
HTTP/1.1 401 Unauthorized
```

404 Not Found

```
HTTP/1.1 404 Not Found
```

Upload image to a comment

Uploads an image to the comment with the given `{id}` (from the URI). One should only be able to upload images to your own comments.

Request template

```
PUT /api/v1/comments/{id}/image HTTP/1.1
Host: 11.12.21.22
Authorization: Bearer 2YotnFZ.FEjr1zC.sicMWpAA
Content-Type: multipart/form-data;boundary=SOME_BOUNDARY

--SOME_BOUNDARY
Content-Disposition: form-data;name="image";filename="the-filename.jpeg"
Content-Type: image/jpeg

The image's binary content here...
--SOME_BOUNDARY--
```

Possible response templates

204 No Content

```
HTTP/1.1 204 No Content
```

401 Unauthorized

```
HTTP/1.1 401 Unauthorized
```

404 Not Found

```
HTTP/1.1 404 Not Found
```