Digital Career Institute

API Basics

Digital Career Institute

API Basics - REST Verbs and Error Codes

Digital Career Institute

Goal of the Module

The goal of this submodule is to give learners an overview API Basics.

By the end of this submodule, the learners should be able to understand:

- the REST paradigm

Digital Career Institute

Topics

- REST verbs and error codes

Digital Career Institute

REST Verbs and Error Codes

Manipulation of resources using HTTP methods CRUD

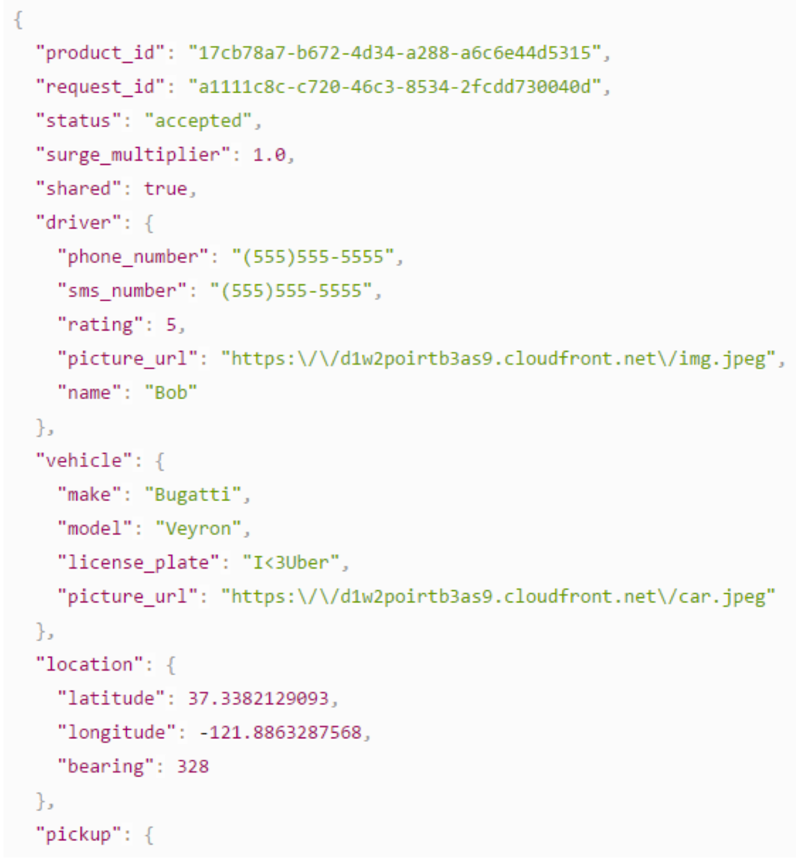
- **POST** – create new resource
- **GET** – returns a resource
- **PUT** – updates the resource (overwrite)
- **DELETE** – deletes the resource
- **PATCH –** updates the resource (partially)

REST Verbs/Error Codes - GET

Uber Example - GET

API endpoint:
api.uber.com

GET
/requests/{request_id}



```
{
  "product_id": "17cb78a7-b672-4d34-a288-a6c6e44d5315",
  "request_id": "a1111c8c-c720-46c3-8534-2fcdd730040d",
  "status": "accepted",
  "surge_multiplier": 1.0,
  "shared": true,
  "driver": {
    "phone_number": "(555)555-5555",
    "sms_number": "(555)555-5555",
    "rating": 5,
    "picture_url": "https:\/\/d1w2poirtb3as9.cloudfront.net\/img.jpeg",
    "name": "Bob"
  },
  "vehicle": {
    "make": "Bugatti",
    "model": "Veyron",
    "license_plate": "I<3Uber",
    "picture_url": "https:\/\/d1w2poirtb3as9.cloudfront.net\/car.jpeg"
  },
  "location": {
    "latitude": 37.3382129093,
    "longitude": -121.8863287568,
    "bearing": 328
  },
  "pickup": {
```

REST Verbs/Error Codes - GET

Design Pattern:

- **Get list of resources:**

  URI Template:

  GET /{namespace}/{resource}

  Example:

  GET /{namespace}/articles?color=red

- **Get resource:**

  URI Template

  GET /{namespace}/{resource}/{resource-id}

  Example:

  GET /user_management/users/BE14A7269802498F992813885546D058

REST Verbs/Error Codes - GET

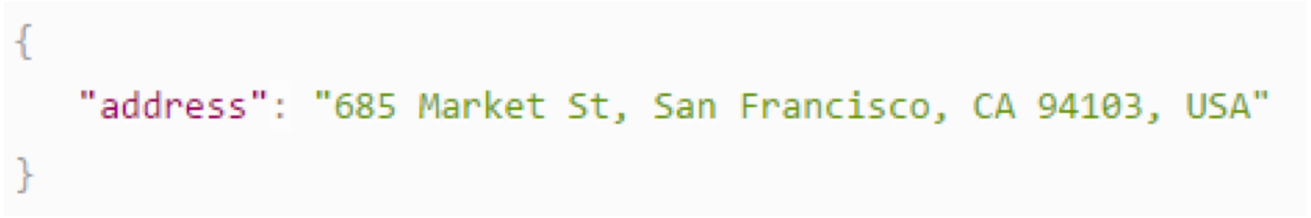
Relevant HTTP Status Codes:

- Success : 200 OK
- Failure : 404 NOT FOUND

➔ (following the HTTP status code guidelines, we could theoretically also face a 403 FORBIDDEN, or 301 MOVED PERMANENTLY etc.. but these codes are not specific or characteristic to REST verb error codes)

REST Verbs/Error Codes - PUT

Uber Example: PUT



```
{
    "address": "685 Market St, San Francisco, CA 94103, USA"
}
```

API endpoint:
api.uber.com

PUT
/places/{place_id}

place_id is either home/work

REST Verbs/Error Codes - PUT

Typical use case: Update single resource

- Shape of the PUT request should maintain parity with the GET response for the selected resource
- Fields to be updated are in the request body (and can be optional or ignored during deserialization, such as "create_time" or other system-calculated values)

URI Template:

```
PUT /{namespace}/{resource}/{resource-id}
```

Example request:

```
PUT /user_management/users/BE14A7269802498F992813885546D058
```

REST Verbs/Error Codes - PUT

Relevant HTTP Status Codes:

- Any failed request validation responds with **400 Bad Request** HTTP status
- If clients attempt to modify read-only fields, this should also be a **400 Bad Request**
- After successful update, PUT operations should respond with **200 OK** or **204 No Content** status, with no response body
- If the resource to be updated is not found, the response code is **404 NOT FOUND**

PUT vs. PATCH

The PATCH verb is exactly like PUT in every way, except that it **only updates the parts** of the resource indicated in the request body.

If we want to change one property of an object using PUT, we need to get the entire object first, change the property we want and then add the entire object to the body of the request.

If we want to change one property of an object using PATCH, we only need to know the id of the object and the name and value of the property to change and send them in the body of the request.

REST Verbs/Error Codes - POST

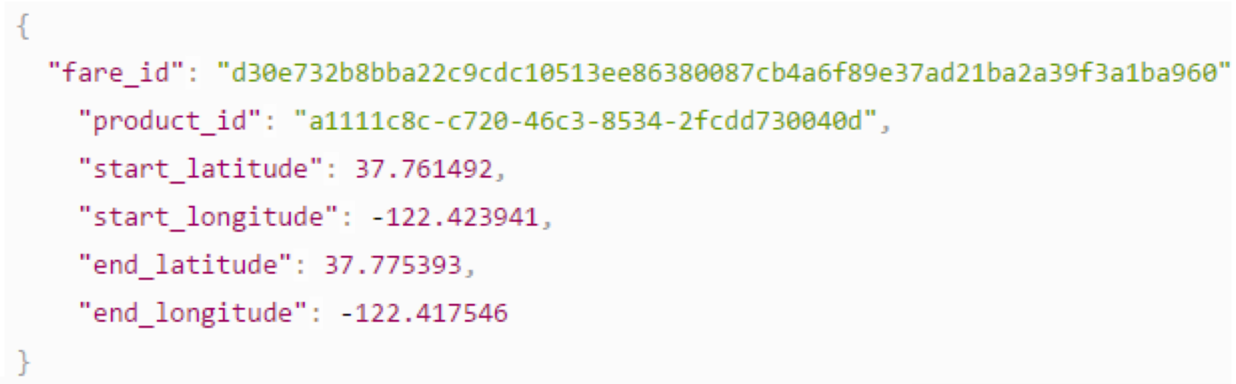
Uber Example: POST

API endpoint:

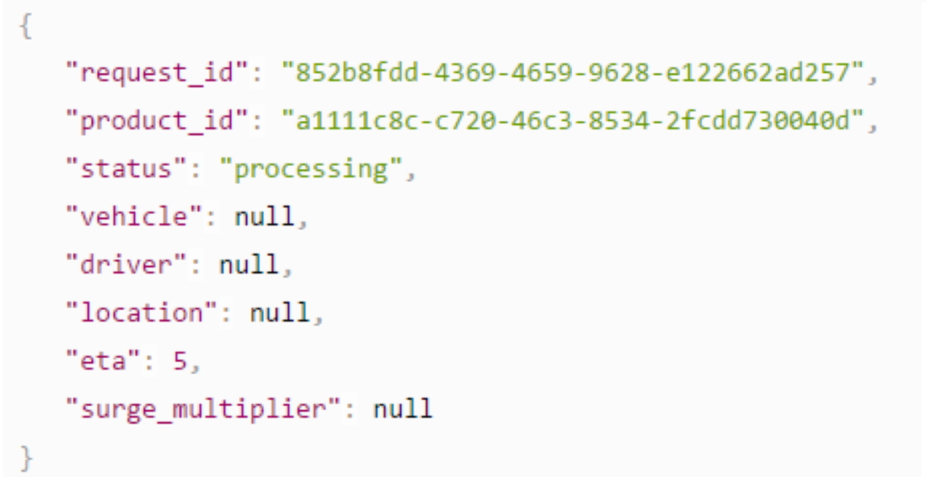
api.uber.com

POST

/requests



```
{
  "fare_id": "d30e732b8bba22c9cdc10513ee86380087cb4a6f89e37ad21ba2a39f3a1ba960"
    "product_id": "a1111c8c-c720-46c3-8534-2fcdd730040d",
    "start_latitude": 37.761492,
    "start_longitude": -122.423941,
    "end_latitude": 37.775393,
    "end_longitude": -122.417546
}
```



```
{
   "request_id": "852b8fdd-4369-4659-9628-e122662ad257",
   "product_id": "a1111c8c-c720-46c3-8534-2fcdd730040d",
   "status": "processing",
   "vehicle": null,
   "driver": null,
   "location": null,
   "eta": 5,
   "surge_multiplier": null
}
```

REST Verbs/Error Codes - POST

Typical use case: Create single resource

- When talking strictly in terms of REST, POST methods are used to create a new resource into the collection of resources.

- Ideally, if a resource has been created on the origin server, the response SHOULD be HTTP response code **201** (Created) and contain an entity which describes the status of the request and refers to the new resource, and a Location header.

REST Verbs/Error Codes - POST

- Sometimes, the action performed by the POST method might not result in a resource that can be identified by a URI. In this case, either HTTP response code 200 (OK) or 204 (No Content) is the appropriate response status.

- Responses to this method are **not cacheable**, unless the response includes appropriate Cache-Control or Expires header fields.

Example request URIs

HTTP POST http://www.appdomain.com/users

HTTP POST http://www.appdomain.com/users/123/accounts

REST Verbs/Error Codes - POST

Safe methods:

- As per HTTP specification, the **GET and HEAD methods should be used only for retrieval of resource representations** – and they do not update/delete the resource on the server. Both methods are said to be considered “**safe**“.
- This allows user agents to represent other methods, such as **POST, PUT and DELETE**, in a unique way so that the user is made aware of the fact that a possibly **unsafe** action is being requested – and they can **update/delete the resource on server** and so should be used carefully

Idempotent methods:

- The term idempotent is used more comprehensively to describe an **operation that will produce the same results if executed once or multiple times**

REST Verbs/Error Codes - POST

- Idempotence is a handy property in many situations, as it means that an operation can be repeated or retried as often as necessary without causing unintended effects
- In case of non-idempotent operations: the algorithm may have to keep track of whether the operation was already performed or not.
- In HTTP specification, The methods **GET, HEAD, PUT and DELETE are declared idempotent methods**. Other methods OPTIONS and TRACE SHOULD NOT have side effects, so both are also inherently idempotent.

Please note that POST is **neither safe nor idempotent**, and invoking two identical POST requests will result in two different resources containing the same information (except resource ids)

REST Verbs/Error Codes - DELETE

Digital Career Institute

API endpoint: **api.uber.com**

DELETE

/requests/{request_id}

Typical use case: delete single resource

Status-Codes:

Success: 200 (OK)

Failure: 404 (Not Found), if ID not found or invalid.

REST Verbs/Error Codes - Summary Table

Digital Career Institute

HTTP Method	CRUD	Entire Collection (e.g. /users)	Specific Item (e.g. /users/123)
POST	Create	201 (Created), 'Location' header with link to /users/{id} containing new ID.	Avoid using POST on single resource
GET	Read	200 (OK), list of users. Use pagination, sorting and filtering to navigate big lists.	200 (OK), single user. 404 (Not Found), if ID not found or invalid.
PUT	Update/Replace	405 (Method not allowed), unless you want to update every resource in the entire collection of resource.	200 (OK) or 204 (No Content). Use 404 (Not Found), if ID not found or invalid.
PATCH	Partial Update/Modify	405 (Method not allowed), unless you want to modify the collection itself.	200 (OK) or 204 (No Content). Use 404 (Not Found), if ID not found or invalid.
DELETE	Delete	405 (Method not allowed), unless you want to delete the whole collection — use with caution.	200 (OK). 404 (Not Found), if ID not found or invalid.

At the core of the lesson

Lessons learned:

- We know how REST APIs are generally built
- We know how to create, read, update and delete REST resources
- We learned about the REST Verbs and Error Codes

Digital Career Institute

Documentation

Digital Career Institute

Documentation

1. OPENAPI Spec
2. JSON Schema
3. WSDL and XML
4. Zalando API guidelines
5. RestfulAPI.net
6. RPC

THANK YOU

Contact Details
DCI Digital Career Institute gGmbH

Digital Career Institute
DCI