CLOUD COMPUTING - PROJECT 1

WATCHES

WEBSERVICE

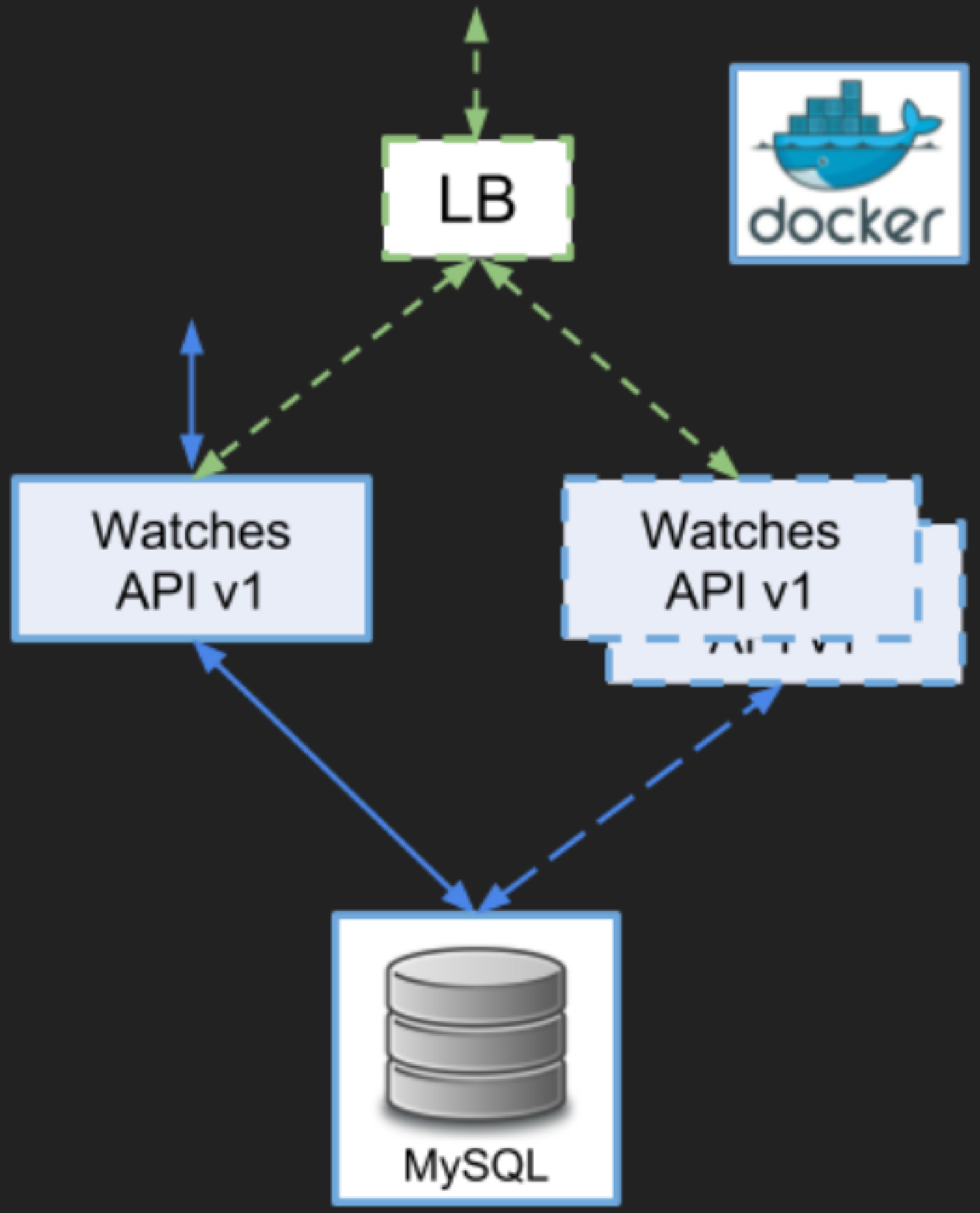
PART I - API

- Develop a RESTful HTTP API from a specification and a DB
- Dockerize and load-balance the implementation
- This document describes Part I

PART II - CLOUD

- Deploy in AWS
- Add load-balancing and scaling components
- Replace components by cloud alternatives
- More about that in a few weeks...

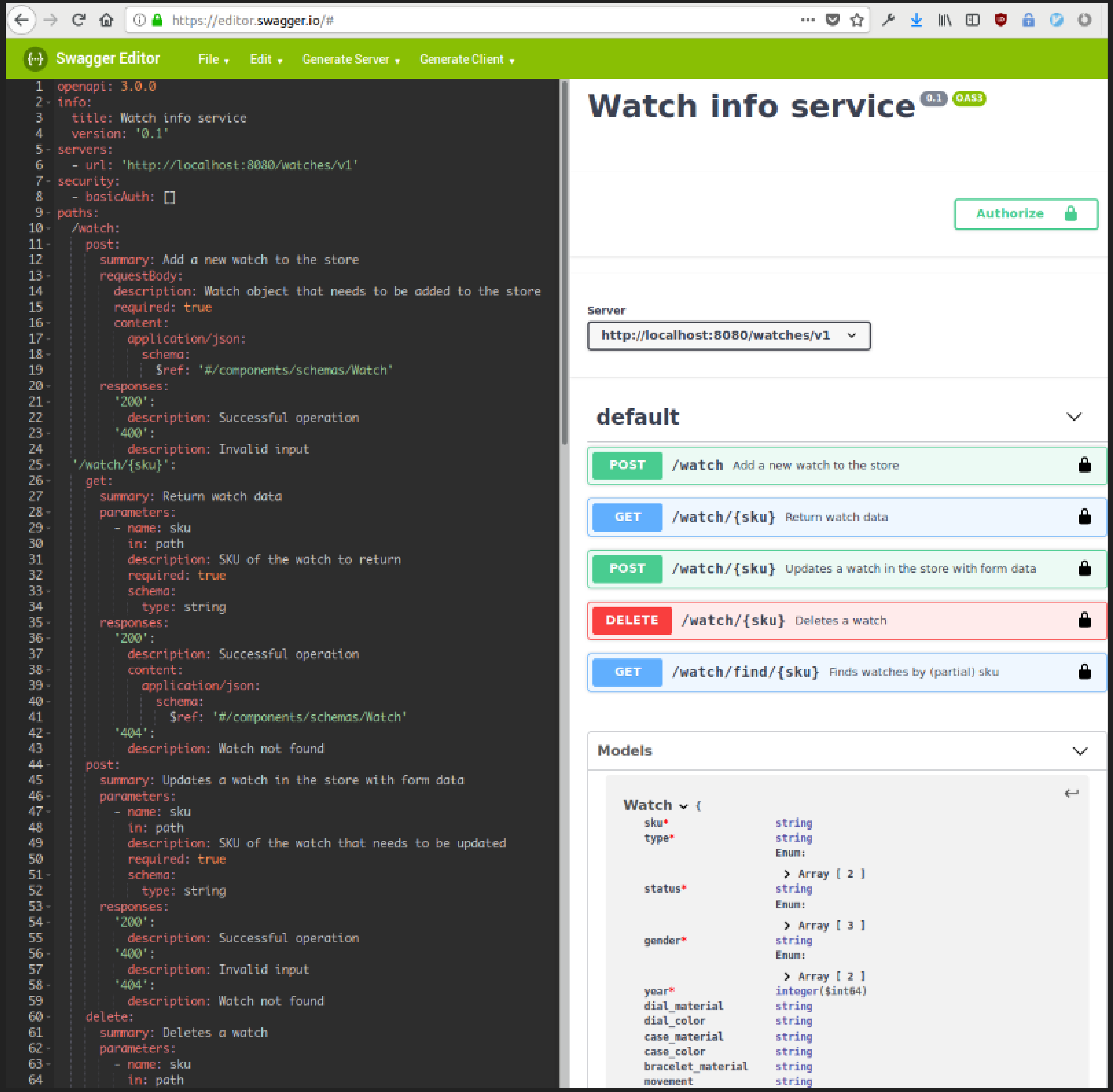
ARCHITECTURE



GITLAB

- /cloudcomputing-2018/project1
- Description of the API
  - openapi.yaml
    - OpenApi v3 (OAS3)
  - swager.yaml
    - Swagger v2
  - ~ equivalent: some tools are not yet compatible with OAS3 version
  - Display/test in https://editor.swagger.io/
- DB
  - watches.sql

OPENAPI - SWAGGER



```yaml
openapi: 3.0.0
info:
  title: Watch info service
  version: '0.1'
servers:
  - url: 'http://localhost:8080/watches/v1'
security:
  - basicAuth: []
paths:
  /watch:
    post:
      summary: Add a new watch to the store
      requestBody:
        description: Watch object that needs to be added to the store
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Watch'
      responses:
        '200':
          description: Successful operation
        '400':
          description: Invalid input
  '/watch/{sku}':
    get:
      summary: Return watch data
      parameters:
        - name: sku
          in: path
          description: SKU of the watch to return
          required: true
          schema:
            type: string
      responses:
        '200':
          description: Successful operation
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Watch'
        '404':
          description: Watch not found
    post:
      summary: Updates a watch in the store with form data
      parameters:
        - name: sku
          in: path
          description: SKU of the watch that needs to be updated
          required: true
          schema:
            type: string
      responses:
        '200':
          description: Successful operation
        '400':
          description: Invalid input
        '404':
          description: Watch not found
    delete:
      summary: Deletes a watch
      parameters:
        - name: sku
          in: path
```

Watch info service 0.1 OAS3

Authorize

Server
http://localhost:8080/watches/v1

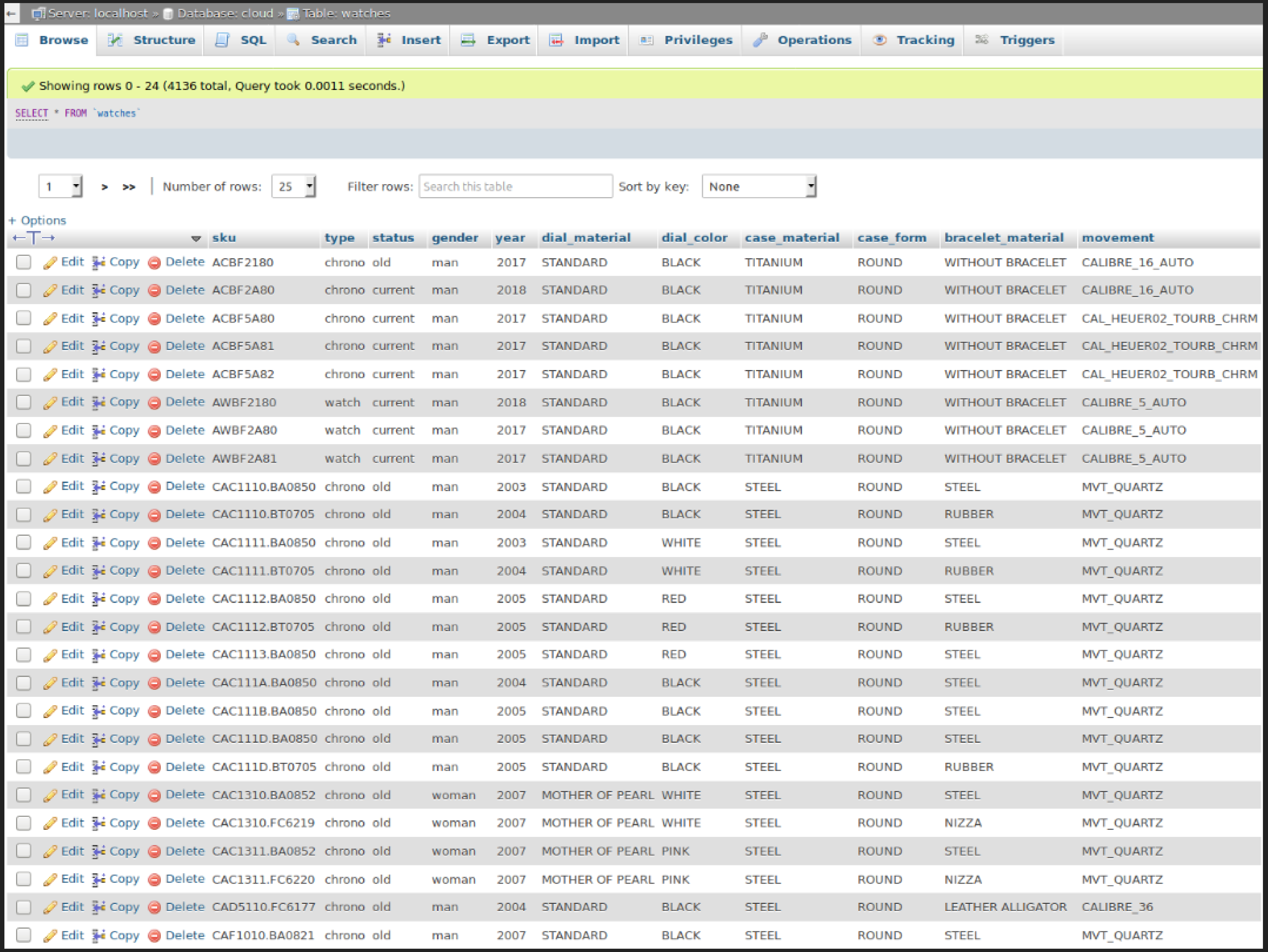
default

POST /watch Add a new watch to the store
GET /watch/{sku} Return watch data
POST /watch/{sku} Updates a watch in the store with form data
DELETE /watch/{sku} Deletes a watch
GET /watch/find/{sku} Finds watches by (partial) sku

Models

```
Watch {
    sku*                 string
    type*                string
                         Enum:
                          > Array [ 2 ]
    status*              string
                         Enum:
                          > Array [ 3 ]
    gender*              string
                         Enum:
                          > Array [ 2 ]
    year*                integer($int64)
    dial_material        string
    dial_color           string
    case_material        string
    case_color           string
    bracelet_material    string
    movement             string
```

MYSQL - DATA



✓ Showing rows 0 - 24 (4136 total, Query took 0.0011 seconds.)

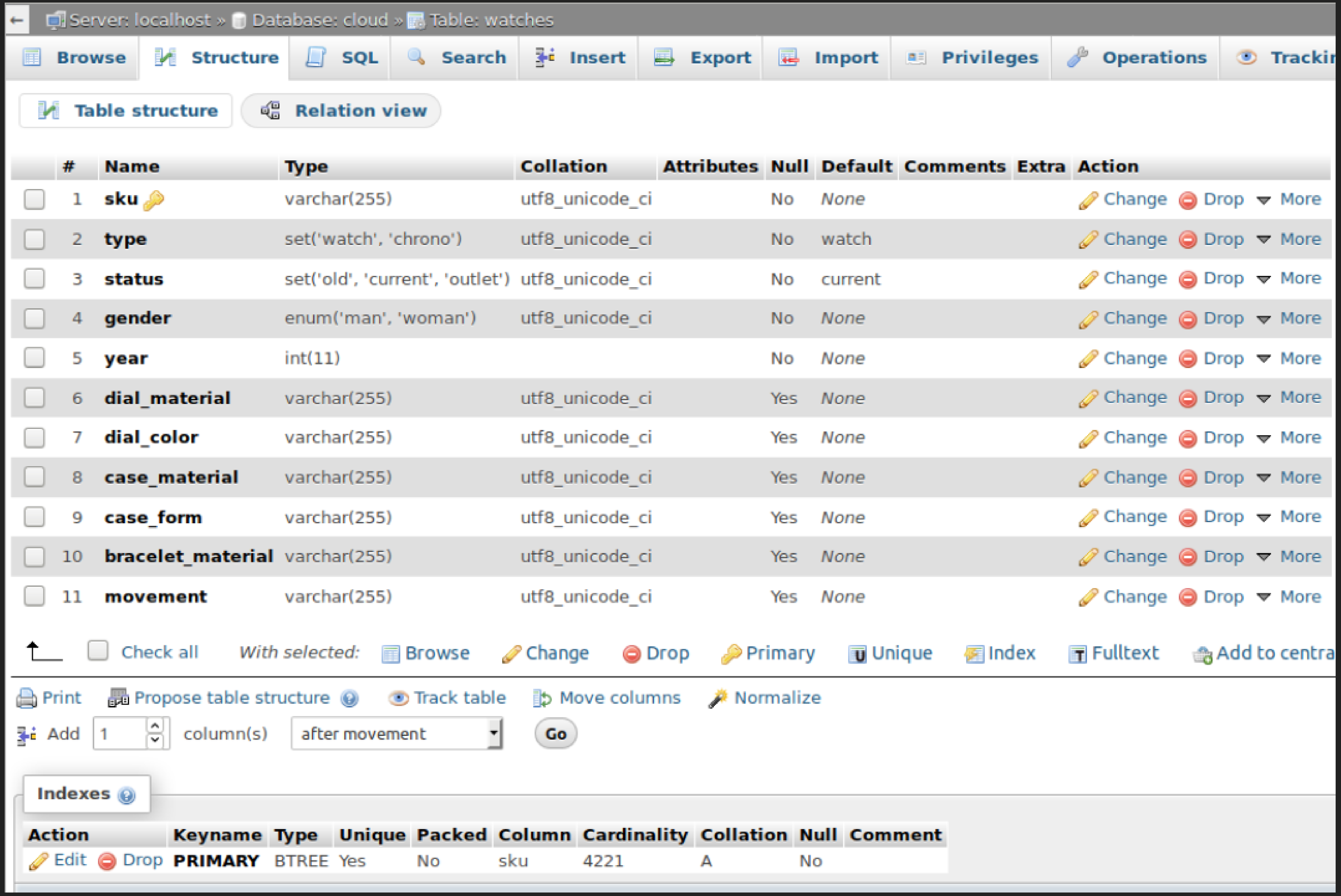
```
SELECT * FROM `watches`
```

1 > >> | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

	sku	type	status	gender	year	dial_material	dial_color	case_material	case_form	bracelet_material	movement
☐ Edit Copy Delete	ACBF2180	chrono	old	man	2017	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CALIBRE_16_AUTO
☐ Edit Copy Delete	ACBF2A80	chrono	current	man	2018	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CALIBRE_16_AUTO
☐ Edit Copy Delete	ACBF5A80	chrono	current	man	2017	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CAL_HEUER02_TOURB_CHRM
☐ Edit Copy Delete	ACBF5A81	chrono	current	man	2017	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CAL_HEUER02_TOURB_CHRM
☐ Edit Copy Delete	ACBF5A82	chrono	current	man	2017	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CAL_HEUER02_TOURB_CHRM
☐ Edit Copy Delete	AWBF2180	watch	current	man	2018	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CALIBRE_5_AUTO
☐ Edit Copy Delete	AWBF2A80	watch	current	man	2017	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CALIBRE_5_AUTO
☐ Edit Copy Delete	AWBF2A81	watch	current	man	2017	STANDARD	BLACK	TITANIUM	ROUND	WITHOUT BRACELET	CALIBRE_5_AUTO
☐ Edit Copy Delete	CAC1110.BA0850	chrono	old	man	2003	STANDARD	BLACK	STEEL	ROUND	STEEL	MVT_QUARTZ
☐ Edit Copy Delete	CAC1110.BT0705	chrono	old	man	2004	STANDARD	BLACK	STEEL	ROUND	RUBBER	MVT_QUARTZ
☐ Edit Copy Delete	CAC1111.BA0850	chrono	old	man	2003	STANDARD	WHITE	STEEL	ROUND	STEEL	MVT_QUARTZ
☐ Edit Copy Delete	CAC1111.BT0705	chrono	old	man	2004	STANDARD	WHITE	STEEL	ROUND	RUBBER	MVT_QUARTZ
☐ Edit Copy Delete	CAC1112.BA0850	chrono	old	man	2005	STANDARD	RED	STEEL	ROUND	STEEL	MVT_QUARTZ
☐ Edit Copy Delete	CAC1112.BT0705	chrono	old	man	2005	STANDARD	RED	STEEL	ROUND	RUBBER	MVT_QUARTZ
☐ Edit Copy Delete	CAC1113.BA0850	chrono	old	man	2005	STANDARD	RED	STEEL	ROUND	STEEL	MVT_QUARTZ
☐ Edit Copy Delete	CAC111A.BA0850	chrono	old	man	2004	STANDARD	BLACK	STEEL	ROUND	STEEL	MVT_QUARTZ
☐ Edit Copy Delete	CAC111B.BA0850	chrono	old	man	2005	STANDARD	BLACK	STEEL	ROUND	STEEL	MVT_QUARTZ
☐ Edit Copy Delete	CAC111D.BA0850	chrono	old	man	2005	STANDARD	BLACK	STEEL	ROUND	STEEL	MVT_QUARTZ
☐ Edit Copy Delete	CAC111D.BT0705	chrono	old	man	2005	STANDARD	BLACK	STEEL	ROUND	RUBBER	MVT_QUARTZ
☐ Edit Copy Delete	CAC1310.BA0852	chrono	old	woman	2007	MOTHER OF PEARL	WHITE	STEEL	ROUND	STEEL	MVT_QUARTZ
☐ Edit Copy Delete	CAC1310.FC6219	chrono	old	woman	2007	MOTHER OF PEARL	WHITE	STEEL	ROUND	NIZZA	MVT_QUARTZ
☐ Edit Copy Delete	CAC1311.BA0852	chrono	old	woman	2007	MOTHER OF PEARL	PINK	STEEL	ROUND	STEEL	MVT_QUARTZ
☐ Edit Copy Delete	CAC1311.FC6220	chrono	old	woman	2007	MOTHER OF PEARL	PINK	STEEL	ROUND	NIZZA	MVT_QUARTZ
☐ Edit Copy Delete	CAD5110.FC6177	chrono	old	man	2004	STANDARD	BLACK	STEEL	ROUND	LEATHER ALLIGATOR	CALIBRE_36
☐ Edit Copy Delete	CAF1010.BA0821	chrono	old	man	2007	STANDARD	BLACK	STEEL	ROUND	STEEL	MVT_QUARTZ

MYSQL - SCHEMA



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	**sku**	varchar(255)	utf8_unicode_ci		No	*None*			Change Drop More
2	**type**	set('watch', 'chrono')	utf8_unicode_ci		No	watch			Change Drop More
3	**status**	set('old', 'current', 'outlet')	utf8_unicode_ci		No	current			Change Drop More
4	**gender**	enum('man', 'woman')	utf8_unicode_ci		No	*None*			Change Drop More
5	**year**	int(11)			No	*None*			Change Drop More
6	**dial_material**	varchar(255)	utf8_unicode_ci		Yes	*None*			Change Drop More
7	**dial_color**	varchar(255)	utf8_unicode_ci		Yes	*None*			Change Drop More
8	**case_material**	varchar(255)	utf8_unicode_ci		Yes	*None*			Change Drop More
9	**case_form**	varchar(255)	utf8_unicode_ci		Yes	*None*			Change Drop More
10	**bracelet_material**	varchar(255)	utf8_unicode_ci		Yes	*None*			Change Drop More
11	**movement**	varchar(255)	utf8_unicode_ci		Yes	*None*			Change Drop More

Check all *With selected:* Browse Change Drop Primary Unique Index Fulltext Add to centra
Print Propose table structure Track table Move columns Normalize
Add 1 column(s) after movement Go

Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Drop	**PRIMARY**	BTREE	Yes	No	sku	4221	A	No	

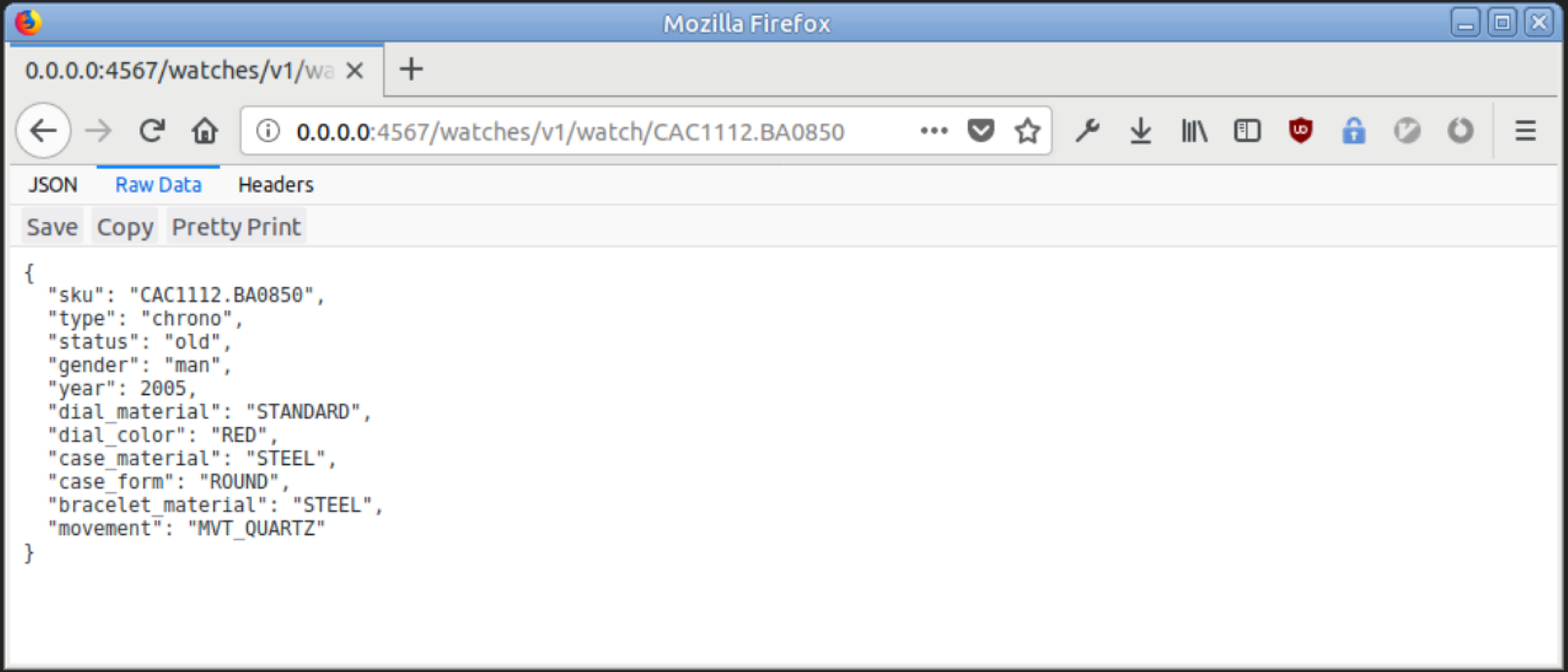
PLAN 1/3

- Install Docker MySQL
  - https://hub.docker.com/_/mysql/
    - Read the page, there are many examples
  - Set a **permanent volume**
  - Load the data received
  - Add appropriate **indexes** to search efficiently
  - You can also install (locally or in another docker) PHPMyAdmin to visualize your data

```
# Use "docker run" first time only, then start/stop

$ docker run --name my-mysql -p3306:3306 -e MYSQL_ROOT_PASSWORD=secret -v /home/db:/var/lib/mysql mysql
$ docker stop my-mysql
$ docker start my-mysql

$ mysql -h 0.0.0.0 -u root -p
CREATE DATABASE cloud;
CREATE USER cloud IDENTIFIED BY 'cloud';
GRANT ALL PRIVILEGES ON cloud.* TO cloud;
FLUSH PRIVILEGES;
exit

$ mysql -h 0.0.0.0 -u cloud -p cloud < watches.sql

$ mysql -h 0.0.0.0 -u cloud -p cloud
SELECT * FROM watches;
exit

$ docker run --name myadmin -d --link my-mysql -p 1080:80 phpmyadmin/phpmyadmin
$ docker stop myadmin
$ docker start myadmin

# Browse to http://0.0.0.0:1080
```

PLAN 2/3

- WebService
  - In the language / REST framework of your choice
  - From a **Swagger/OpenAPI** specification
    - Set proper **HTTP success/error codes** (from spec)
    - **Authentication** (HTTP basic auth): cloud / computing
  - Set expiration **headers** so all data read (GET) is valid for 1 hour
  - Use a (high level) DB library
    - Connect to the DB (MySQL port is 3306 by default)
    - Use the library to query and extract data (or objects)
  - Use a JSON library to convert the results
- Bonus
  - Embed a HAProxy in front of your WS (http://www.haproxy.org/)
    - Same Docker image as WS
    - Protect your service by limiting the number of simultaneous connections


Mozilla Firefox
0.0.0.0:4567/watches/v1/wa
0.0.0.0:4567/watches/v1/watch/CAC1112.BA0850
JSON
Raw Data
Headers
Save
Copy
Pretty Print

```
{
  "sku": "CAC1112.BA0850",
  "type": "chrono",
  "status": "old",
  "gender": "man",
  "year": 2005,
  "dial_material": "STANDARD",
  "dial_color": "RED",
  "case_material": "STEEL",
  "case_form": "ROUND",
  "bracelet_material": "STEEL",
  "movement": "MVT_QUARTZ"
}
```

PLAN 3/3

- Dockerize the Webservice
  - Use existing dockers images with the runtime you need as a starting point https://hub.docker.com/
  - Connect to DB with command line:
    - `$ docker run --link my-mysql -d your-webservice`
  - Use ENV vars to pass
    - host/credentials to connect to the DB
    - Credential to access your service (HTTP basic auth)
- Bind everything together
  - Write docker-compose.yml
    - Set env vars
  - `$ docker-compose up`
    - ⇨ Everything starts and works
  - ⇨ More infos about docker compose next week

DELIVERABLES

- WS
  - **Source code**
  - **Dockerfile**
  - **Final SQL schema** (without data)
  - (if bonus) haproxy.conf
- Other
  - **docker-compose.yml**
  - **README**
    - Briefly explain the technical choices of your app or any complementary indication
    - Add installation notes if any
- Push in your Gitlab assignment repository (/project1)
  - (if team) Create a team repository for the project

GRADING / DELAY

- Grading:
  - 50% the service (blindly) work as expected
    - Follow exactly the specification
    - 1h caching headers
  - 50% other
    - Code / Dockerfile / README / git usage / ...
  - Bonus
    - +0.5 !
- Delay: 3 weeks (Part I)
  - 2018-10-24T23:59:59+02:00
- Gitlab
  - Show your progress by commiting regularly
- Individually or by team (up to 3 people)
  - By team:
    - Create a new team repo for the project (share with us)
    - All team members use it