

 Home(/)

0x01. Basic authentication

 My Projects(/projects/me)

Weight: 1

Projects(/projects/current)



Project over - took place from Aug 5, 2024 6:00 AM to Aug 7, 2024 6:00 AM



An auto QA review will automatically make corrections(/to_review)



Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)

In a nutshell...

- **Auto QA review:** 166.0/169 mandatory & 22.8/27 optional

- **Altogether: 181.16%**



Curriculums(/dashboards/my_curriculums)

◦ Mandatory: 98.22%

◦ Optional: 84.44%

◦ Calculation: $98.22\% + (98.22\% * 84.44\%) == 181.16\%$ 

Concepts(/concepts)



Conference rooms(/dashboards/video_rooms)



Background Context

Servers(/servers)

In this project, you will learn what the authentication process means and implement a **Basic Authentication** on a simple API.



Sandboxes(/user_containers/current)

In the industry, you should **not** implement your own Basic authentication system and use a module or framework that doing it for you (like in Python-Flask: Flask-HTTPAuth



or JWT token/Tools (/tools/Tools_For_Other_Python_Tools)). Here, for the learning purpose, we will walk through each step of this mechanism to understand it by doing.



Video on demand(/dashboards/videos)



Peers(/users/peers)

Discord(<https://discord.com/app>)

My Profile(/users/my_profile)



Resources

Curriculum([dashboards/my_curriculums](#))

Read or watch:



- REST API Authentication Mechanisms ([/rltoken/ssg5umgsMk5jKM8WRHk2Ug](#))
- Base64 in Python ([/rltoken/RpaPRyKx1rdHgRSUyuPfeg](#))
- HTTP header Authorization ([/rltoken/WIARq8tQPUGQq5VphLKM4w](#))
- Conference rooms([dashboards/video_rooms](#))
- Flask ([/rltoken/Zes_6jYFlaadm1IG470tfg](#))
- Base64 - concept ([/rltoken/br6Rp4iMaOce6EAC-JQnOw](#))



Servers([/servers](#))

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone ([/rltoken/swilZazfz7mspY1vjuy_Zg](#)), **without the help of Google**:

General

Tools([dashboards/my_tools](#))



- What authentication means
- What Base64 is
- How to encode a string in Base64
- What Basic authentication means
- How to send the Authorization header



Peers([/users/peers](#))

Requirements



Discord(<https://discord.com/app>)

Python Scripts

- All your files will be interpreted/compiled on Ubuntu 18.04 LTS using python3 (version 3.7)
- All your files should end with a new line
- The first line of all your files should be exactly `#!/usr/bin/env python3`
- A README file at the root of the folder of the project, is mandatory
- Your code should use the `pycodestyle` style (version 2.5)





- All your files must be executable
- The length of your files will be tested using `wc`
- All your modules should have a documentation (`python3 -c`

```
'print(__import__("my_module").__doc__)' )
```



- All your classes should have a documentation (`python3 -c`
`Home()`
`'print(__import__("my_module").MyClass.__doc__)')`



- All your functions (inside and outside a class) should have a documentation (`python3 -c`
`myPlanning(myPlanning)`
`'print(__import__("my_module").my_function.__doc__)'` and `python3 -c`
`'print(__import__("my_module").MyClass.my_function.__doc__)')`



- A documentation is not a simple word, it's a real sentence explaining what's the purpose of the module, class or method (the length of it will be verified)



QA Reviews I can make(/corrections/to_review)

Tasks

Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)

0. Simple-basic-API

mandatory



Curriculums(/dashboards/my_curriculums)

Score: 100.0% (Checks completed: 100.0%)



Download and start your project from this archive.zip (/rltoken/2o4gAozNufil_KjoxKI5bA)

Concepts(/concepts)

In this archive, you will find a simple API with one model: `User` . Storage of these users is done via a serialization/deserialization in files.



Conference rooms(/dashboards/video_rooms)

Setup and start server



```
bob@dylan:~/servers$ pip2 install -r requirements.txt
```

```
...
```

```
bob@dylan:~$
```



```
bob@dylan:~$ API_HOST=0.0.0.0/ API_PORT=5000 python3 -m api.v1.app
```

```
* Serving Flask app "app" (lazy loading)
```

```
...
```



```
bob@dylan:~$
```

Tools(/dashboards/my_tools)

Use the API (*in another tab or in your browser*)



Video on demand(/dashboards/videos)



Peers(/users/peers)



Discord(<https://discord.com/app>)



My Profile(/users/my_profile)

```

bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/status" -vvv
* Trying 0.0.0.0...
* TCP_NODELAY set
* Connected to 0.0.0.0 (127.0.0.1) port 5000 (#0)
> GET /api/v1/status HTTP/1.1
> Host: 0.0.0.0:5000
> User-Agent: curl/7.54.0
> Accept: */*
> My Planning(/planning/me)
* HTTP 1.0, assume close after body
< HTTP/1.0 200 OK
< Content-Type: application/json
< Content-Length: 16
< Access-Control-Allow-Origin: *
< Server: Werkzeug/0.10.1 Python/3.7.5
< Date: Mon, 18 May 2020 20:29:21 GMT
<
{"status": "OK"}
* Closing connection 0
bob@dylan:~$

```

 **Curriculums(/dashboards/my_curriculums)**

Repo:



- GitHub repository: alx-backend-user-data
- Directory: 0x01-Basic_authentication



Conference rooms(/dashboards/video_rooms)

Check submission

Get a sandbox

View results



Error handler: Unauthorized

mandatory

 **Score: 100.0% (Checks completed: 100.0%)**

Sandboxes(user_containers/current)

What the HTTP status code for a request unauthorized? 401 of course!



Tools(/dashboards/my_tools)

Edit api/v1/app.py :



- Add a new error handler for this status code, the response must be:
 - a JSON: {"error": "Unauthorized"}
 - status code 401
 - you must use jsonify from Flask



Peers(/users/peers)

For testing this new error handler, add a new endpoint in api/v1/views/index.py :

- Route: GET /api/v1/unauthorized
- This endpoint must raise a 401 error by using abort - Custom Error Pages (/rltoken/RH0gY_XQuSB75Q-Jbl-fdg)

By calling abort(401) , the error handler for 401 will be executed.

in the first terminal:

My Profile(/users/my_profile)





```
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 python3 -m api.v1.app
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
..(I).
```



In a second terminal:
Home(/)

```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/authorized"
```



```
{
  "error": "Unauthorized"
}
```



```
bob@dylan:~$
```



```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/authorized" -vvv
```

```
* Trying 0.0.0.0...
* TCP_NODELAY set
* Connected to 0.0.0.0 (127.0.0.1) port 5000 (#0)
> GET /api/v1/authorized HTTP/1.1
> Host: 0.0.0.0:5000
? > User-Agent: curl/7.7.1.54.0
> Accept: */*
>
```



```
* HTTP 1.0, assume close after body
< HTTP/1.0 401 UNAUTHORIZED
  Content-Type: application/json
  Content-Length: 30
< Server: Werkzeug/0.12.1 Python/3.4.3
  Date: Sun, 24 Sep 2017 22:50:40 GMT
<
```



```
{
  "error": "Unauthorized"
}
```



```
* Closing connection 0
bob@dylan:~$
```



Sandboxes(/user_containers/current)

Repo:



- GitHub repository: alx-backend-user-data
- Directory: 0x01-Basic_authentication
- File: api/v1/app.py, api/v1/views/index.py



Video on demand(/dashboards/videos)

Check submission

> Get a sandbox

View results



Error handler: Forbidden

mandatory



Score: 100.0% (Checks completed: 100.0%)
Discord(<https://discord.com/app>)

What the HTTP status code for a request where the user is authenticate but not allowed to access to resource? 403 of course!



Edit api/v1/app.py :

- My Profile(/users/my_profile)
Add a new error handler for this status code, the response must be:

- a JSON: {"error": "Forbidden"}
- status code 403
- (/) ◦ you must use jsonify from Flask

For testing this new error handler, add a new endpoint in `api/v1/views/index.py` :



Home(/)

- Route: GET `/api/v1/forbidden`
- This endpoint must raise a 403 error by using `abort` - Custom Error Pages



(/token/RH0gY-XQwSB75G-Jbl-fdg)

By calling `abort(403)`, the error handler for 403 will be executed.



in the first terminal: `Projects(/projects/current)`



```
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 python3 -m api.v1.app
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
....
```



in a second terminal: `Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)`



`Curriculum(s)(dashboards/my_curriculums)`



```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/forbidden" -vvv
* Trying 0.0.0.0...
```



```
* TCP_NODELAY set
* Connected to 0.0.0.0 (127.0.0.1) port 5000 (#0)
> GET /api/v1/forbidden HTTP/1.1
```



```
> Host: 0.0.0.0:5000
> User-Agent: curl/7.54.0
> Accept: */*
```



```
> * HTTP 1.0, assume close after body
< HTTP/1.0 403 FORBIDDEN
< Content-Type: application/json
< Content-Length: 27
< Server: Werkzeug/0.12.1 Python/3.4.3
< Date: Sun, 24 Sep 2017 22:54:22 GMT
```



```
< Video on demand(/dashboards/videos)
{
  "error": "Forbidden"
}
```



```
* Closing connection 0
bob@dylan:~$
Peers(/users/peers)
```



Repo: Discord(<https://discord.com/app>)

- GitHub repository: `alx-backend-user-data`
- Directory: `0x01-Basic_authentication`
- File: `api/v1/app.py`, `api/v1/views/index.py`

My Profile(/users/my_profile)



[Check submission](#)[Get a sandbox](#)[View results](#)[\(/\)](#)

3. Auth class

mandatory[Home\(/\)](#)

Score: 100.0% (Checks completed: 100.0%)

[My Planning\(/planning/me\)](#)

Now you will create a class to manage the API authentication.

- Create a folder `api/v1/auth`

- Create an empty file `api/v1/auth/__init__.py`

- Create the class `Auth`:

- in the file `api/v1/auth/auth.py`

- import `request` from `Flask`

- class name `Auth`

- public method `def require_auth(self, path: str, excluded_paths: List[str]) -> bool`: that returns `False` - `path` and `excluded_paths` will be used later, now, you don't need to take care of them

- public method `def authorization_header(self, request=None) -> str`: that returns

- `None` - `request` will be the `Flask` request object

- public method `def current_user(self, request=None) -> TypeVar('User')`: that returns `None` - `request` will be the `Flask` request object

[Concepts\(/concepts\)](#)

This class is the template for all authentication system you will implement.



```
bob@dylan:~$ cat main_0.py
```

```
#!/usr/bin/env python3
```

```
""" Main 0
```

```
"""
```

```
Servers(/servers)
```

```
from api.v1.auth.auth import Auth
```

```
a = Auth()
```

```
Sandboxes(/user_containers/current)
```

```
print(a.require_auth("/api/v1/status/", ["/api/v1/status/"]))
```

```
print(a.authorization_header())
```

```
print(a.current_user())
```



```
bob@dylan:~$
```

```
Video on demand(/dashboards/videos)
```

```
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 ./main_0.py
```

```
False
```

```
None
```

```
None
```



```
bob@dylan:~$
```

```
Peers(/users/peers)
```

[Repo: Discord\(https://discord.com/app\)](#)

- GitHub repository: `alx-backend-user-data`

- Directory: `0x01-Basic_authentication`

- File: `api/v1/auth`, `api/v1/auth/__init__.py`, `api/v1/auth/auth.py`

[My Profile\(/users/my_profile\)](#)

[Check submission](#)[Get a sandbox](#)[View results](#)[\(\)](#)

4. Define which routes don't need authentication

mandatory[Home\(/\)](#)

Score: 100.0% (Checks completed: 100.0%)

[Update the planning \(/planning/me\)](#)

update the planning (/planning/me) requires_auth(self, path: str, excluded_paths: List[str]) -> bool:
in Auth that returns True if the path is not in the list of strings excluded_paths:



- [Projects \(/projects/curriculum\)](#)

- Returns True if excluded_paths is None or empty

- Returns False if path is in excluded_paths



- [QA Reviews I can make \(/corrections/to:review\)](#)

- You can assume excluded_paths contains string path always ending by a /

- This method must be slash tolerant: path=/api/v1/status and path=/api/v1/status/ must



- be returned False if excluded_paths contains /api/v1/status/

- [Evaluation quizzes \(/dashboards/my_current_evaluation_quizzes\)](#)

```
bob@dylan:~$ cat main_1.py
```

```
#!/usr/bin/env python3
```

```
""" Main 1
```



```
""" Curriculum (/dashboards/my_curriculums)
```

```
from api.v1.auth.auth import Auth
```



```
a = Auth() Concepts (/concepts)
```

```
print(a.require_auth(None, None))
```



```
print(a.require_auth(None, [])) Conference rooms (/dashboards/video_rooms)
```

```
print(a.require_auth("/api/v1/status/", []))
```

```
print(a.require_auth("/api/v1/status/", ["/api/v1/status/"]))
```



```
print(a.require_auth("/api/v1/status", ["/api/v1/status/"])) Servers (/servers)
```

```
print(a.require_auth("/api/v1/users", ["/api/v1/status/"]))
```

```
print(a.require_auth("/api/v1/users", ["/api/v1/status/", "/api/v1/stats/"]))
```



```
Sandboxes (/user_containers/current)
```

```
bob@dylan:~$
```

```
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 ./main_1.py
```



```
True Tools (/dashboards/my_tools)
```

```
True
```

```
True
```



```
False Video on demand (/dashboards/videos)
```

```
False
```

```
True
```

```
True
```

```
bob@dylan:~$
```



```
Peers (/users/peers)
```



Repo: [Discord\(https://discord.com/app\)](https://discord.com/app)

- GitHub repository: alx-backend-user-data

- Directory: 0x01-Basic_authentication

- File: api/v1/auth/auth.py

[My Profile\(/users/my_profile\)](#)

[Check submission](#)[Get a sandbox](#)[View results](#)[\(/\)](#)

5. Request validation!

mandatory[Home\(/\)](#)

Score: 100.0% (Checks completed: 100.0%)



Now you will validate all requests to secure the API:

[api/v1/projects/projects/current\)](#)

- If request is None , returns None
- If request doesn't contain the header key Authorization , returns None
- Otherwise, return the value of the header request Authorization

Update the file `api/v1/app.py` :[Evaluation quizzes\(/dashboards/my_current_evaluation_quizzes\)](#)

- Create a variable `auth` initialized to `None` after the `CORS` definition
- Based on the environment variable `AUTH_TYPE` , load and assign the right instance of authentication to `auth`

[Curriculums\(/dashboards/my_curriculums\)](#)

- `import Auth from api.v1.auth.auth`

[Concepts\(/concepts\)](#)

- create an instance of `Auth` and assign it to the variable `auth`

Now the biggest piece is the filtering of each request. For that you will use the Flask method

[before_request \(/rltoken/kzBrJT9aaokbD6aWYyQzXg\)](#)[Conference rooms\(/dashboards/video_rooms\)](#)

- Add a method in `api/v1/app.py` to handler `before_request`
 - if `auth` is `None` , do nothing
- If request path is not part of this list `['/api/v1/status/', '/api/v1/unauthorized/', '/api/v1/forbidden/']` , do nothing - you must use the method `require_auth` from the `auth` instance
 - if `auth.authorization_header(request)` returns `None` , raise the error `401` - you must use `abort`

[Servers\(/servers\)](#)

- If request path is not part of this list `['/api/v1/status/', '/api/v1/unauthorized/', '/api/v1/forbidden/']` , do nothing - you must use the

[method require_auth from the auth instance](#)

- if `auth.authorization_header(request)` returns `None` , raise the error `401` - you must use `abort`

[Tools\(/dashboards/my_tools\)](#)

- If `auth.authorization_header(request)` returns `None` , raise the error `403` - you must use `abort`



the first terminal:

[Video on demand\(/dashboards/videos\)](#)

```
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 AUTH_TYPE=auth python3 -m api.v1.app
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

[Peers\(/users/peers\)](#)



In a second terminal:

[Discord\(https://discord.com/app\)](https://discord.com/app)[My Profile\(/users/my_profile\)](#)


```

bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/status"
{
  (/)status": "OK"
}
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/status/"
{
  "status": "OK"
}
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users"
{
  Projects(/projects/current)
  "error": "Unauthorized"
}
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users" -H "Authorization: Test"
{
  "error": "Forbidden"
}
bob@dylan:~$

```


 Curriculums(/dashboards/my_curriculums)
Repo:



- GitHub repository: alx-backend-user-data
- Directory: 0x01-Basic_authentication
- File: api/v1/app.py, api/v1/auth/auth.py


 Conference rooms(/dashboards/video_rooms)

Check submission

> Get a sandbox

View results



 Servers(/servers)

6. Basic auth

mandatory



 Sandboxes(/user_containers/current)

Score: 100.0% (Checks completed: 100.0%)


 Tools(/dashboards/my_tools)
 Create a class `BasicAuth` that inherits from `Auth`. For the moment this class will be empty.

Update `api/v1/app.py` for using `BasicAuth` class instead of `Auth` depending of the value of the environment variable `AUTH_TYPE`, if `AUTH_TYPE` is equal to `basic_auth`:

- import `BasicAuth` from `api.v1.auth.basic_auth`
- create an instance of `BasicAuth` and assign it to the variable `auth`


 Peers(/users/peers)
 otherwise, keep the previous mechanism with `auth` an instance of `Auth`.

In the first terminal:

```


Discord(https://discord.com/app)
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 AUTH_TYPE=basic_auth python3 -m api.v
1.app
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
....

```


In a second terminal:


My Profile(/users/my_profile)

```
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/status"
{
  (/)status": "OK"
}
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/status/"
{
  "status": "OK"
}
My Planning(/planning/me)
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users"
{
  Projects(/projects/current)
  "error": "Unauthorized"
}
bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users" -H "Authorization: Test"
{
  "error": "Forbidden"
}
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
bob@dylan:~$
```


 Curriculumms(/dashboards/my_curriculumms)

Repo:

-  GitHub repository: alx-backend-user-data
- Concepts(/concepts)
- Directory: 0x01-Basic_authentication
- File: api/v1/app.py, api/v1/auth/basic_auth.py

 Conference rooms(/dashboards/video_rooms)

Check submission

 Get a sandbox

View results


 Servers(/servers)

7. Basic - Base64 part

mandatory

 Sandboxes(/user_containers/current)

Score: 100.0% (Checks completed: 100.0%)

 Tools(/dashboards/my_tools)

Add the method `def extract_base64_authorization_header(self, authorization_header: str)`
-> `str`: in the class `BasicAuth` that returns the Base64 part of the `Authorization` header for a

 Basic Authentication:
Video on demand(/dashboards/videos)

- Return `None` if `authorization_header` is `None`
- Return `None` if `authorization_header` is not a string
- Return `None` if `authorization_header` doesn't start by `Basic` (with a space at the end)
- Otherwise, return the value after `Basic` (after the space)
- You can assume `authorization_header` contains only one `Basic`

 Discord(<https://discord.com/app>)



My Profile(/users/my_profile)

- You can assume `decoded_base64_authorization_header` will contain only one :
My Profile(/users/my_profile)

```

bob@dylan:~$ cat main_4.py
#!/usr/bin/env python3
"""Main 4
"""

from api.v1.auth.basic_auth import BasicAuth
Home()

a = BasicAuth()

print(MyPlanning(/planning/me))
print(a.extract_user_credentials(89))
print(a.extract_user_credentials("Holberton School"))
print(Projects(/projects/current))
print(a.extract_user_credentials("Holberton:School"))
print(a.extract_user_credentials("bob@gmail.com:toto1234"))

bob@dylan:~$ cat main_4.py
#!/usr/bin/env python3
"""Main 4
"""

from api.v1.auth.basic_auth import BasicAuth
Home()

a = BasicAuth()

print(MyPlanning(/planning/me))
print(a.extract_user_credentials(89))
print(a.extract_user_credentials("Holberton School"))
print(Projects(/projects/current))
print(a.extract_user_credentials("Holberton:School"))
print(a.extract_user_credentials("bob@gmail.com:toto1234"))

bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 ./main_4.py
(None, None)
(None, None)
('Holberton', 'School')
('bob@gmail.com', 'toto1234')
bob@dylan:~$
Curriculums(/dashboards/my_curriculums)


```

 **epo:** Concepts(/concepts)

- GitHub repository: alx-backend-user-data
- Directory: 0x01-Basic authentication
- File: api/v1/auth/basic_auth.py

 Servers(/servers)

Check submission


 Get a sandbox

View results

>40. Basic User object

mandatory

 Score: 78.57% (Show details)

 Add the method `def user_object_from_credentials(self, user_email: str, user_pwd: str)` - Video on demand(/dashboards/videos) - TypeVar("User"): In the class `BasicAuth` that returns the `User` instance based on his email and password.

- Return `None` if `user_email` is `None` or not a string
- Return `None` if `user_pwd` is `None` or not a string
- Return `None` if your database (file) doesn't contain any `User` instance with email equal to `user_email` - you should use the class method `search` of the `User` to lookup the list of users based on their email. Don't forget to test all cases: "what if there is no user in DB?", etc.
- Return `None` if `user_pwd` is not the password of the `User` instance found - you must use the method `is_valid_password` of `User`
- Otherwise, return the `User` instance

My Profile(/users/my_profile)

```

bob@dylan:~$ cat main_5.py
#!/usr/bin/env python3
"""(?) Main 5
"""

import uuid
from app.v1.auth.basic_auth import BasicAuth
from models.user import User

""" create user via planning/me
user_email = str(uuid.uuid4())
user_clear_pwd = str(uuid.uuid4())
user = Projects/projects/current
user.email = user_email
user.first_name = "Bob"
user.password = user_clear_pwd
user.display_name = "Bob Dylan"
user.curriculum = "QA Reviews + corrections/to_review"
user.password = user_clear_pwd
print("New user: {}".format(user.display_name()))
user_evaluation_quizzes(/dashboards/my_current_evaluation_quizzes)

""" Retreive this user via the class BasicAuth """

a = BasicAuth()
Curriculum(/dashboards/my_curriculum)
u = a.user_object_from_credentials(None, None)
print(u.display_name() if u is not None else "None")
Concepts(/concepts)
u = a.user_object_from_credentials(89, 98)
print(u.display_name() if u is not None else "None")
Conference rooms(/dashboards/video_rooms)
u = a.user_object_from_credentials("email@notfound.com", "pwd")
print(u.display_name() if u is not None else "None")
Servers(/servers)
u = a.user_object_from_credentials(user_email, "pwd")
print(u.display_name() if u is not None else "None")
Sandboxes(/user_containers/current)
u = a.user_object_from_credentials(user_email, user_clear_pwd)
print(u.display_name() if u is not None else "None")
Tools(/dashboards/my_tools)
bob@dylan:~$
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 ./main_5.py
New user: Bob Dylan
None
None
None
None
Peers(/users/peers)
Bob Dylan
bob@dylan:~$

```

Discord(<https://discord.com/app>)

Repo:

- GitHub repository: alx-backend-user-data
- Directory: 0x01-Basic_authentication
- File: app/v1/auth/basic_auth.py



(/)

11. Basic - Overload current_user - and BOOM!

mandatory

Home(/)

Score: 100.0% (Checks completed: 100.0%)

Now, you have a complete Basic authentication.

Add the method `def current_user(self, request=None) -> TypeVar('User')` in the class `BasicAuth` that overloads `Auth` and retrieves the `User` instance for a request:

-
- You must use `authorization_header`
 - You must use `extract_base64_authorization_header`
 - You must use `decode_base64_authorization_header`
 - You must use `extract_user_credentials`
 - You must use `user_object_from_credentials`

With this update, now your API is fully protected by a Basic Authentication. Enjoy!

In the first terminal:

Curriculums(/dashboards/my_curriculums)

bob@dylan:~\$ cat main_6.py

#!/usr/bin/env python3

""" Main 6
"""
import base64
from api.v1.auth.basic_auth import BasicAuth
from models.user import User

""" Create a user test """
user_email = "bob@hbtn.io"
user_clear_pwd = "H0lbertonSchool98!"
user = User()
user.email = user_email
user.password = user_clear_pwd
print("New user: {} / {}".format(user.id, user.display_name()))
user.save()

basic_clear = "{} {}".format(user_email, user_clear_pwd)
print("Basic Base64: {}".format(base64.b64encode(basic_clear.encode('utf-8')).decode("utf-8")))

bob@dylan:~\$

bob@dylan:~/API/003\$ API_HOST=0.0.0.0 API_PORT=5000 ./main_6.py

New user: 9375973a-68c7-46aa-b135-29f79e837495 / bob@hbtn.io

Basic Base64: Ym9iQGhidG4uaw86SDBsYmVydG9uU2Nob29sOTgh

bob@dylan:~\$ https://discord.com/app

bob@dylan:~\$ API_HOST=0.0.0.0 API_PORT=5000 AUTH_TYPE=basic_auth python3 -m api.v1.app

* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)

....

In a second terminal:

```

bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/status"
{
  ("status": "OK"
}

bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users"
{
  "error": "Unauthorized"
}

bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users" -H "Authorization: Test"
{
  "error": "Forbidden"
}

bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users" -H "Authorization: Basic tes
t"
{
  "error": "Forbidden"
}

bob@dylan:~$
bob@dylan:~$ curl "http://0.0.0.0:5000/api/v1/users" -H "Authorization: Basic Ym9
iQGhidd4uaw86SDBSfMvYdG9u02Nob29501gh"
[
  {
    "created_at": "2017-09-25 01:55:17",
    "email": "bob@hbtn.io",
    "first_name": null,
    "id": "9375973a-68c7-40aa-b135-2d1f9c837495",
    "last_name": null,
    "updated_at": "2017-09-25 01:55:17"
  }
]

bob@dylan:~$

```

Repo: Tools/(dashboards/my_tools)

- GitHub repository: alx-backend-user-data
- Directory: 0x01-Basic_authentication
- Video on demand/(dashboards/videos)
- File: api/v1/auth/basic_auth.py

Check submission

Peers(/users/peers)

Get a sandbox

View results

12. Basic - Allow password with ":"

#advanced

 Discord(<https://discord.com/app>)

Score: 100.0% (*Checks completed: 100.0%*)



Improve the method `def extract_user_credentials(self, decoded_base64_authorization_header)` to allow password with `:`.

In the first terminal:

```

bob@dylan:~$ cat main_100.py
#!/usr/bin/env python3
"""(?) Main 100
....

import base64
Home(/)
from api.v1.auth.basic_auth import BasicAuth
from models.user import User

My Planings(/planning/me)
create user
user_email = "bob100@hbtn.io"
user_clear_pwd = "H0lberton:School:98!"
Projects(/projects/current)
user = User()
user.email = user_email
QA Reviews I can make/corrections/to_review
print("New user: {}".format(user.id))
user.save()
? Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
basic_clear = "{}:{}".format(user_email, user_clear_pwd)
print("Basic Base64: {}".format(base64.b64encode(basic_clear.encode('utf-8')).decode("utf-8")))

Curriculums(/dashboards/my_curriculums)
bob@dylan:~$
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 ./main_100.py
New user: 5891469b-d2d5-4d33-b05d-02617d665368
Concepts(/concepts)
Basic Base64: H0lberton:School:98!
bob@dylan:~$
bob@dylan:~$ API_HOST=0.0.0.0 API_PORT=5000 AUTH_TYPE=basic_auth python3 -m api.v
1.app Conference rooms(/dashboards/video_rooms)
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
....
Servers(/servers)

```

In a second terminal:

```

>_ Sandboxes(/user_containers/current)

Tools(/dashboards/my_tools)

Video on demand(/dashboards/videos)

```

Peers(/users/peers)

Discord(<https://discord.com/app>)

My Profile(/users/my_profile)



13. Require auth with stars

#advanced



(/)

Score: 73.75% (Checks completed: 100.0%)



Home(/)

Improve the require_auth(self, path, excluded_paths) by allowing * at the end of excluded paths.



My Planning(/planning/me)

Example of excluded_paths = ["/api/v1/stats*"]:



Project(/project/current)

- /api/v1/users will return True
- /api/v1/project/current will return False
- /api/v1/stats will return False



QA Reviews I can make(/corrections/to_review)

Repo:



- GitHub repository: alx-backend-user-data
- Directory: 0x01-Basic_authentication
- File: api/v1/auth/auth.py



Curriculums(/dashboards/my_curriculums)

Check submission

> Get a sandbox

View results



Concepts(/concepts)



Conference rooms(/dashboards/video_rooms)

Copyright © 2024 ALX, All rights reserved.



Servers(/servers)



Sandboxes(/user_containers/current)



Tools(/dashboards/my_tools)



Video on demand(/dashboards/videos)



Peers(/users/peers)



Discord(https://discord.com/app)



My Profile(/users/my_profile)