|  |
| --- |
| Aqui criar GET e POST, e no postman testo para saber set funciona  POST toda vez que post ele cria um novo produto com preco 12 (201 CREATED) |
| Quanto tento GET piano, retorna 404 (precisa estar em formato dicionário pois vai ser lido como Jason para Restfull API), pois não criei |
| POST - seu eu tento postar um produto existente, ele retornar 201 |
| Aqui a Class ItemsList mostra todos os items que foram POST  Em rosa esta estrutura para colocar cada item dentro da lista |
| Yellow - if the item exist, it will return the item, if not we got an error 404, since we are using next to interect, we need to include None, if not it will break the code, when implement next without an item to interact  Green - it does not allow post a existem item    Yellow - all item, If we send an existem an item, it lamba will filter and returne the item….otherwise it will return null (404)    Green |
| 79  Authentication and logging in—part 1  Jwt - Jason web tooken |
| Create security andu ser file  Yellow - Security.py import the class User from user.py to users list  Blue - Dict comprehension to create user…mapping  Pink - import safe\_str\_cmp to compare string  Def identify use secrect\_key (line 5) to identify the yser    Correction authenticate |
| 72. Authentication and logging in—part 2  Import security module and classes authenticate and identity  Also import flask\_jwt and classes  And include decorator jwt required |
| On postman app  Setup auth route |
| On postman body post the username and password according to security.py  It will generate an access token, you need to copy it, since when we use a method get the decorator will validate it |
| Post a item on /item/<name> |
| Get items |
| Now in order to get the item piano, we need to set up the key as “Authorization” and value as “JWS <access token generate on auth route>” |

|  |
| --- |
| Define the function delete inside Item’s class  Define items list as global inside the function, since item list is on scope global and function is scope local |
|  |

73. DELETE to delete Items

|  |
| --- |
| 74. PUT to create or update Items |
| Define function put request inside items’class  It will allows both post or if an existem item, we will be able to update it |

|  |
| --- |
| 75. Advanced request parsing with Flask-RESTful |
| Import reqparse,  reqparser.RequestParser() create the parser  define the argument, which key we can parse to payload  ex the key is price, only accept the type float, it is required (we cannot omit it) and the message error  data will receiver parser.parse\_args() defined oj add\_argmument |
| If we try to change the name field to pric, we got a error, the same one define on add\_argument |
| If we want to reuse the same parte to post and put methor, move parse create and add\_rgument to scope local of class, and inside of function add parser.parse\_args()  Since it parser belong to class not to an specific resource (that is why it does not have self dot in front o reqparser) , so wee need to add the name of class in from of parser.parse\_args() |

|  |
| --- |
| Line 15 - |
|  |
|  |

80. Logging in and retrieving Users from a database

|  |
| --- |
| Create a database    Find\_by\_username needs to :   * Orange Create a connection * cls is used on @classmethos as conversion, since it is objetct of class, we don’t use self, so it * Yellow \*row is unpack the \_id, username, password |
|  |

|  |
| --- |
| User.py   * Create and end point to Register as Resoucer to register new user * Create a class UserRegister and impor Resoucer, reqparse (pesquisar sobre Resoucer, because I think that all api needs an endpoint, and it needs to receive a resource, since ) * Reqparser é para passa * Connect with database to insert * Since id is auto-incremnte on query it will be NULL, and the other two question mark is to username and password. query = "INSERT INTO users VALUES (NULL, ?, ?)" * Create parser will parse throught the JSON of the request * Parser add\_argument, type, required and help * data variable will receive the aparser   App.py   * Import the resource (UserRegister) * Create a resourser for UserRegister under endpoint /register |
| JSON payload pesquisar |
|  |

81. Signing up and writing Users to a database

|  |
| --- |
| If statement use User.Find\_by\_username to check if username exist |
|  |
|  |

82. Preventing duplicate usernames when signing users up

83. Retrieving our Item resources from a database

|  |
| --- |
| Create items table and add test items |
| Move class Item and Items from app.py to a new file item.py and import dependencies and update it to connect to database |
| Remove imports required for classes moved to item.py  And import item.py and the classes |
| I have issue on postman due to find\_by\_id quantity of arguments, so I remove None argument inside User.find\_by\_id and leave only user\_id |

84. Writing our Item resources to a  
database

|  |
| --- |
| Since we updating the code to use database, we need to update the class Item  Update function post for class Item to self.find\_by\_name(name)  Create find\_by\_name and move the code that connect to databasefrom get function to this function self.find\_by\_name  Since get requires jwt token, but post not, we need to create @classmethod for find\_by\_name  Update the get function |
| Update post to connect to database |
| Update delete to connect to database |
| Update PUT and create @classmethod for update function |
| @classmethod def insert(cls, item) and update post function, since both put and post will insert item to database |
| Update class ItemList(Resource): |