|  |
| --- |
| 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): |

# SECTION 6

|  |
| --- |
| Create the two folders models and resources and create the \_\_init\_\_.py inside folder, it is only for old version of python 3.5, it will tell python that can look inside this folders. |
| Move the py files to folders, and now we have package. So we need to update the files that import them include the name of package, so security.py was importing user now it has to be resources.user |

93. Creating User and Item models

|  |
| --- |
| Class User is not a Resourcer, beacause the API cannot receive data into this class or send this class as a JSON representation. This Class is a hepler that we use to store some data about the User and also a helper that contains a couple of methods that allow is to easily retreive User object from a database.  Model is our internal representation of an entity.  Resources is our extenral representation of ana entity. So the client interacting with resources, and wehn our APO responds it responds with resources. It is what client sees.  So we need to move the class user to model package. Now we has class User inside models, and Class UserRegister inside resources. |
| Update Class User name to UserModel and Update all instance that use User to UserModel |
| Also update security.py and package name from resources to models, and all instance of class User to UserModels |
| Also update security.py and package name from resources to models, and all instance of class User to UserModels |
| Resource is user to map endpoints  But @classmethod and find\_by\_name , insert and update are not and endpoint, so the client does not interact with these methods directly. They are not called by an API directly, they are only user from within our code, so therefore it doesn't make sense to pollute the resource with these methods because it doesn't help at all. So we need to move them from resource to models.  So the resource is only containing methods that API interact with.  Item model > create Create a JSON method and all this is going to do is it's going to return a JSON representation of the model, basically a dictionary.  Item Resources > Import Item Model and update the all instance self the entirety with ItemMOdel |
| Does not make sense for ItemModel has @classmethod for insert and update, we have the ItemModel, which represents an item, and the insert method is taking in an item that is going to insert into the database, we need to update from cls to self |
| Update @classmethod find\_by\_name to return object instead of disctionary. from {'item': {'name': row[0], 'price': row[1]}} to cls(row[0], row[1])    Or uppacking it |
| Blue - Update ItemModel.find\_by\_name return objects as opposeded to a dictionary  Pink - item = ItemModel.find\_by\_name(name) if it exist will update otherwise create an item to database  Yellow = create variable item to packing class ItemModel to use it to get, post and put |
|  |
|  |

95. Advanced Postman: environments and tests

|  |
| --- |
| CREATE ENVIROMENT TO {{url}} |
| SELECT ENVIROMENT |
| CREATE AN ENVIORMENT VARIABLE FOR THE JWT\_TOKEN |
| TESTS CREATE A VAR IN JS TO CREATE A VARIABLE TO RECEIVE JWT TOKEN  IF WE CLICK ON EYE INCO WE CAN SEE THE VARIABLE |
| CREATE TEST TO GET ITEM TO SHOW STATUS CODE AND AND TIME MESSAGE LESS 200MS |
| TEST RESULTS SHOW ERROR TO CODE 200, SINCE POST METHOD IS 201, NEED TO UPDATE TESTS |

97. Telling SQLAlchemy about our tables and columns

|  |
| --- |
| Create db.py to create db SQLAlchemy  Init app db on app.py  Set app.config['SQLALCHEMY\_TRACK\_NOTIFICATION'] = False. It turns off flask sqlalchemy trackers, but it does not turn off SQLalchemy tracker main library.  Also extend db.Model on User.Model and ItemModel and give \_\_tablename\_\_ and columns  There is small erros on code, I got it after running app.py |

98. Implementing the ItemModel using SQLAlchemy

|  |
| --- |
| INCLUDE Primary key id for items table |
| ItemModel update to implementing SQLAlchemy: left old | right new   * Update function find\_by\_name * Create save\_to\_db and delete\_from\_db * Remove insert and update |
| Update item resource  update delete, since item model will delete\_from\_db  update post and put to use save\_to\_db |
| App.py update to tell sqlalchemy where to find data.db file (app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///data.db')  Also where is sqlite, it can be MySQL, PostSQL and Oracle |

99. Implementing the UserModel using SQLAlchemy

|  |
| --- |
|  |

101. No more creating tables manually—telling SQLAlchem

|  |
| --- |
| Delete creates\_tables script, since sqlalchemy will do it automatically per the decorator below  So app.before\_first\_request will create data.db if we don’t have data.db, it will happen before first request, ex.: if I don’t data.db and go to postman and send a register, it will create the database |

102. Creating a new model: StoreModel

|  |
| --- |
| Create StoreModel - just copy ItemModel to this Model, and ajust per below  Red Line - It is connection with SQLAlchemy  Green - Create a relationship btw the ItemModel and StoreModel / And ForeingKey stores.id is how they are connected. Now in order to create a item we need to link it a store\_id, so the store\_id on \_\_init\_\_ for ItemModel is the one on StoreModel  Yellow - key to connect to both models  So the flow is create a db instance of SQLAlchemy, both models StoreModel and ItemModel creates tables and add values on db instance, and before add values to item tables, we need to have store value on store table, because store\_id value for ItemModel is a ForeignKy to Store Table, it is how relationship connect both tables  Json function on StoreModel to returns a item, we need to create a query builder add parameter lazy=’dynamic on items variable, because if we don’t create querybuilder, the item would be an object and it would store all items from items table. |
| Item Resource - Create add\_argument to store\_id. And unpacking price and store\_id in item inside post and put function |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |