bus.gal-api

peprolinbot

Nov 14, 2022

CONTENTS

CHAPTER

ONE

BUS.GAL-API

Python API wrapper for bus.gal which uses the associated app's http API to get the inforamtion about buses, your card and your user account. I got the endpoints using mitmproxy.

1.1 Call for help

If you find stops which have different names but are physically the same, please open an issue. Thanks for your contribution.

1.2 Documentation

Documentation can be found here

1.3 Installation

Just run:

```
pip install busGal_api
```

1.4 Quick example

This is just a simple command-line "client"

```
import busGal_api as api
from datetime import datetime

def menu(results):
    for i, result in enumerate(results):
        print(f"{i} -- {result.name}")
```

(continues on next page)

(continued from previous page)

```
    return int(input("Which number you want? >>>"))

origin = input("Where do you want to start your trip? >>>")
results = api.search_stop(origin)
selection = menu(results)
origin = results[selection]

destination = input("Where do you want to go? >>>")
results = api.search_stop(destination)
selection = menu(results)
destination = results[selection]

trip = api.Trip(origin, destination, datetime.now())

if trip.expeditions == None:
    print("No results")
    exit()

print("\nORIGIN  |  DEPARTURE  |  DESTINATION  |  ARRIVAL\n")
for expedition in trip.expeditions:
    print(f"{expedition.origin.name}  |  {expedition.departure.strftime('%H:%M')}  |
↪{expedition.destination.name}  |  {expedition.arrival.strftime('%H:%M')}")
```

1.5 Disclaimer

This project is not endorsed by, directly affiliated with, maintained by, sponsored by or in any way officially related with la Xunta de Galicia, the bus operators or any of the companies involved in the bus.gal website and the app.

1.5.1 busGal_api

busGal_api module

class Trip(*origin*, *destination*, *date*, *operator=None*, *equivalents=True*)

Bases: `object`

Trip class. Used for getting results as Expedition objects

Parameters

- **origin** (*_Stop*) – Origin stop
- **destination** (*_Stop*) – Destination stop
- **date** (*datetime.datetime*) – The date the trip will take place. Just the day matters
- **operator** (*_Operator*) – The operator that you would like to own the buses
- **equivalent** (*bool*) – Wether or not to get equivalent trips. Fixes stops which are the same one in the real world but are different ones in the API.

_get_equivalent_expeditions_from_api(*origin*, *destination*, *date*, *operator*)

Calls _get_expeditions_from_api for every equivalent combination of stops and returns a list with all the expeditions merged. Called on creation if equivalents==true.

Returns
List of avaliable expeditions

Return type
list[*_Expedition*]

_get_expeditions_from_api(*origin*, *destination*, *date*, *operator*)

Obtains all the expeditions from the app API. Called on creation

Returns
List of avaliable expeditions

Return type
list[*_Expedition*]

destination

Destination stop

Type
_Stop

expeditions

List of avaliable expeditions

Type
list[*_Expedition*]

origin

Origin stop

Type
_Stop

class _Expedition(*data*, *date*, *operator=None*)

Bases: `object`

Represents any of the expeditions of a Trip. Get's it's own data from dictionary of the json response of api

Parameters

- **data** (*dict*) – Dictionary corresponding to the expedition in the json response of the API
- **date** (*datetime.datetime*) – Date when the expedition takes place. Just the day matters
- **operator** (*_Operator*) – Operator object to make sure the operator property has the right type. If not given will use "operator" type

arrival

Arrival time

Type
datetime.datetime

code

Code-name of the expedition

Type
str

departure

Deaparture time

Type
datetime.datetime

`destination`

Destination stop object

Type
_Stop

`id`

Id of the expedition

Type
int

`line`

Bus line object

Type
_Line

`on_demand`

Whether th stop works under demand

Type
bool

`operator`

Operator object. Note that if no operator argument was given the type of this will default to "operator"

Type
_Operator

`origin`

Origin stop object

Type
_Stop

`url`

Url on bus.gal for the expedition page

Type
str

`class _Line(id, name)`

Bases: `object`

Class that represents a bus line

`id`

Id of the line

Type
int

`name`

Name of the line

Type
str

class _Operator(*id*, *name*, *type*)

Bases: object

Class that represents an operator enterprise

id

Id of the operator

Type
int

name

Name of the operator

Type
str

type

Type of the operator. Ex: 'group', 'operator'

Type
str

class _Stop(*id*, *name*, *type*, *type_id*, *bus_stop_id=None*)

Bases: object

Class that represents a bus stop

bus_stop_id

Id the API gives just in expedition details. Idk what it is. It's used to obtain _Expedition.url

Type
int

id

Id of the stop

Type
int

name

name of the stop

Type
str

type

Type string of the stop . Ex: 'municipality', 'busstop'

Type
str

type_id

Type id of the stop

Type
int

get_operators()

Gets all the existing operators

Returns
List of all operators

Return type
list[*_Operator*]

`get_stops()`

Gets all the existing stops

Returns
List of all stops

Return type
list[*_Stop*]

`search_operator(name)`

Searchs for operators with the specified name, using the app's search api

Parameters
name (*str*) – Search query

Returns
List of operator results

Return type
list[*_Operator*]

`search_stop(name)`

Searchs for stops with the specified name, using the app's search api

Parameters
name (*str*) – Search query

Returns
List of stop results

Return type
list[*_Stop*]

`class Account(email=None, password=None, token=None, user_id=None)`

Bases: `object`

Class that represents a user account. Either email and password or token and user_id (not really necessary) must be specified

Parameters

- **email** (*str*) – Email address to login with
- **password** (*str*) – Password to log in with
- **token** (*str*) – Auth token to use for the requests. It will be auto-obtained if email and password are specified
- **user_id** (*int*) – The user account's id. Not really needed for anything unless you are doing weird shit with the JWTs

`_make_get_request(url)`

Calls make_get_request using the object's token, which is updated after every request. Not intended to be used by clients

Parameters
url (*str*) – Full url to make the request to

Returns
Dictionary made from the request's json

Return type
dict

`_make_post_request`(*url*, *data*)

Calls make_post_request using the object's token, which is updated after every request. Not intended to be used by clients

Parameters

- **url** (*str*) – Full url to make the request to
- **data** (*dict*) – Data to send, it will be sent as application/json

Returns
Dictionary made from the request's json

Return type
dict

`add_card`(*number*, *alias*)

Add a card to the user with the specified number and alias

Parameters

- **number** (*str*) – Number of the card
- **alias** (*str*) – Alias for the card

`delete_card`(*number*)

Delete the card with the specified number

Parameters
number (*str*) – Number of the card

`email`

Email of the user

Type
str

`get_card`(*number*)

Get the object for the user's card with the specified number

Parameters
number (*str*) – Number of the card

Returns
Card object

Return type
_Card

`get_cards`()

Get all the user cards

Returns
List with all the obtained cards' objects

Return type
list(*_Card*)

`identity_number`

Identity number of the user e.g. DNI

Type
str

`identity_type`

Identity type. It can be "DNI" or "other"

Type
str

`last_name`

Last name of the user

Type
str

`login`(*email*, *password*)

Logins with the given email and password and returns a token, which is also set to self.token

Parameters

- **email** (*str*) – Email address to login with
- **password** (*str*) – Password to login with

Returns
User token

Return type
str

`name`

First name of the user

Type
str

`phone_number`

The user's phone number

Type
str

`refresh_data`()

Obtains the user's details and sets them to the object

`rename_card`(*number*, *alias*)

Change the alias of a card with the specified number

Parameters

- **number** (*str*) – Number of the card
- **alias** (*str*) – New alias for the card

`token`

The user token. It is a JWT.

Type
str

user_id

The user id. You won't probably need this

Type
int

class _Card(*account*, *number*)

Bases: object

Class that represents a card. If an object of this class is deleted, so will be the card, it's in the destructor (__del__)

account

The user account the card belongs to

Type
Account

alias

The alias of the card

Type
str

cashed

Amount of money cashed from the Xunta (cantos cartos lle roubaches ó pirolas)

Type
float

expired

Amount of money expired from the Xunta (cantos cartos quedouse o pirolas)

Type
float

is_xente_nova

Whether the card is Xente Nova or no

Type
bool

number

Number of the card

Type
int

pending

Amount of money pending of return from the Xunta (cantos cartos débeche o pirolas)

Type
float

refresh_data()

Obtains the card's details and sets them to the object

rename(*alias*)

Change the alias of the card

Parameters
alias (*str*) – New alias for the card

recover_password(*email*)

Recover an account's password (sends an email with a new temporal one)

Parameters

email (*str*) – Email address

register_account(*email*, *password*, *name*, *last_name*, *identity_type*, *identity_number*, *phone_number*)

Register an user account

Parameters

- **email** (*str*) – Email address
- **password** – Password (Must be at least 6 character length, including a digit and an upper case letter)
- **name** (*str*) – First name
- **last_name** (*str*) – Last name
- **identity_type** (*str*) – Identity type: "DNI" or "other"
- **identity_number** (*str*) – Identity number e.g. your DNI
- **phone_number** (*str*) – Phone number

CHAPTER

TWO

INDICES AND TABLES

- genindex

PYTHON MODULE INDEX

b

INDEX

Symbols

A

B

C

D

E

G

I

L

M

N

O

P