API review 1.0.0

25/12/2018

Vitor Carmo Vannuchi <[vitor.vannuchi@gmail.com](mailto:vitor.vannuchi@gmail.com)>



Auth & Customer Search API

Customer Search, Score List and Auth API correspond to a set of features accessible by HTTPS GET or POST calls on the “PicPay” infrastructure related to customer query, score system and a small authentication API.

# Summary

[Authentication](#_8yyb9r4n44n8)

[a. Register](#_z7e14dyhu061)

[Body (example):](#_n7ar73tet9tq)

[Response (success):](#_lb7zzohsn6vt)

[Errors List:](#_4y4fi0y956rh)

[Description:](#_wzwyztqmjhyg)

[b. Login](#_e7padtwaqqsz)

[Headers (example):](#_1etk9ehaxn8a)

[Body (example):](#_eq2el1qv0z5d)

[Response (success):](#_pu5ky7p6t5du)

[Errors List:](#_iirn9tserg9z)

[Description:](#_22ylwe7euy6v)

[Customer Search](#_di8kp7rayajn)

[a. Customer](#_hmdpm5xxjsud)

[Headers (attention on bearer):](#_uzr897lgtwoj)

[Body (example):](#_z4pfp09c5gy2)

[Response (success):](#_4xcul5c8hrbj)

[Error Code List:](#_5aqfeia92xry)

[Description:](#_cwezyjie6uua)

# Authentication

The authentication API is necessary for authorizing your access to the system, for this project no security layer was made, in order to make easier and quicker to provide access to the API system.

It contains the following methods:

* Register
* Login

It also contains logout, but only used for frontend purposes.

## a. Register

The register functions takes as parameter, name, email and password and then generate an API token that must be used on every transaction with the backend other than Auth requests. Body and response are in JSON object.

**Method**: POST

**URL**: /api/auth/signup

**Headers**:

X-Requested-With : XMLHttpRequest

Content-type : application/json

### **Body** (example):

{

"name" : "Vitor Vannuchi",

"email": "vitor.vannuchi@picpay.com",

"password": "123456",

"password\_confirmation": "123456"

}

### **Response** (success):

{

"message":"Successfully created user!",

"token":"5zk8yAAMzg1deYyeNAGzMOQSIkNuZOIMlhUMhQrROxy6gI4m8o6QKSUTRmJW"

}

**Response Errors** (example)**:**

{

"message": "The given data was invalid.",

"errors": {

"email": [

"The email has already been taken."

],

"password": [

"The password field is required."

]

}

}

### Errors List:

"name": ["The name field is required."],

"email": ["The email has already been taken."],

"email": ["The email must be a valid email address."],

"email": ["The email field is required."],

"password": ["The password field is required."]

"password": ["The password confirmation does not match."]

### **Description:**

***body:***

**Name: String -** name of the user/customer

**Email: String -** email of the user/customer, used for access purpose

**Password: String -** Alphanumeric value, used for access purposes

***response:***

**Message: String -** Message with brief description of response

**Errors: Array -** List of errors when response is not successful

## b. Login

The login function is used just for frontend purposes and also to discover the bearer associated with your account, where its enables the backend customer to access the frontend and use the functionalities before expiration. Body and response are in JSON object.

**Method**: POST

**URL**: /api/auth/login

### **Headers** (example):

Authorization: Bearer aOHCQKZdeD8F1uSTCBgpf48QVwZOnXFe3cRh5s6j

Content-type : application/json

### **Body** (example):

{

"email": "vitor.vannuchi@picpay.com",

"password": "123456",

"remember\_me": true

}

### **Response** (success):

{

"access\_token":"1ARlT7YQpMEo3CXRZIimadTcBHVcesm6fg7xrZQL5pyofwDBxr3aVQ5cTyZE",

"token\_type": "Bearer",

"expires\_at": "2019-01-01 19:22:26"

}

**Response Errors** (example)**:**

{

"message": "Unauthorized"

}

### Errors List:

**Unauthorized:** when some of the parameters provided is not correct

### Description:

***Headers:***

**Authorization :** it’s the bearer token provided by passport plugin, it can be found at the database table: oauth\_clients under the name of Laravel Personal Access Client

***Response:***

**Access\_token:** this token will be used for authorizing all requests, frontend and backend.

# Customer Search

The customer search is the main function for searching a customer based on 3 (three) parameters: token (id), name and username. For the customer search you can decide the parameters to be used and their contents used in the search, the API uses a Score system which show the results by relevance, in total a user can have a score of 0 (zero) to 3 (three), being 3 (three) the most relevant and 0 (zero) the least.

It contains the following method:

* Customer

## 

## a. Customer

The customer function gets as parameter any of the 3 (three) parameters or all: token, name, username together with page and show per page integers that will limit your search result. Body and response are in JSON object.

**Method**: POST

**URL**: /api/rest/customer

### **Headers (attention on bearer)**:

Authorization:Bearer 1ARlT7YQpMEo3CXRZIimadTcBHVcesm6fg7xrZQL5pyofwDBxr3aVQ5cTyZE

Accept : application/json

Content-type : application/json

### **Body** (example):

{

"parameters" : {

"id": "",

"name": "vitor",

"username": ""

},

"show\_per\_page": 15,

"page" : 1

}

### **Response** (success):

{

"show\_per\_page":15,

"offset":0,

"current\_page":1,

"total\_rows":5260,

"total\_pages":351,

"Data":[{

"id":930216,

"token":"061c4f03-7dfe-44e6-860c-72aa5440aea4",

"name":"Jackeline Mauri Gioia",

"username":"jackeline.mauri.gioia",

"score":1

},

{

"id":92508,

"token":"baa068c4-88da-4cb0-bd7b-1950b0731115",

"name":"Maurilio Fava",

"username":"maurilio.fava",

"score":null

},

"success":true,

"error\_code":200,

"message":"success",

"errors":[]}

**Response Errors** (example)**:**

{

"success":false,

"error\_code":422,

"message":"Wrong input in one of the body parameters",

"Errors":

{

"name":["The name must be at least 4 characters."]

}

}

### Error Code List:

**200:** Success, everything works

**204:** Success, but no customer was found

**400:** Error, Your JSON is not correct

**422:** Error, One of the body parameters does not respect the validation rules, min 4 characters in the parameters and required/integers in the page and show\_per\_page

### **Description:**

***Header:***

**Authentication:** bearer token provided in the moment of the login.

***Body:***

**parameters:** JSON obj with all 3 possible parameters, is possible to remove and leave just one, we do not recommend query with empty strings.

**id:** is the field to insert any similar to the token/id searched.

**name:** is the field to insert any similar to the name of the user searched.

**username:** is the field to insert any similar to the name of the user searched.

**page:** current page you’re looking for, starting on 1.

**show\_per\_page:** how many items of the search is wanted to be show per page

***Response:***

**show\_per\_page: Integer -** how many items are being show per page

**offset: Integer -** displaying from the offset number of the total search

**current\_page: Integer -** which is the current page

**total\_rows: Integer -** total number of rows to be returned

**total\_pages: Integer -** total number of pages to be returned

**data: Array -** array of rows (users) matching the search, ordered by relevance

**success: Boolean -** boolean relative to success or failure

**error\_code: Integer -** code referent to the status of the request

**message: String -** Message with brief description of response

**errors: Array -** List of errors when response is not successful