Bilemo- API

Documentation

Version : 1.0

Table des matières

1 - Introduction :

Bilemo est une API de vente de téléphones en B2B. Il s'agit d'une API REST qui répond à niveau 3 du modèle de maturité de Richardson.

Elle permet d'obtenir la liste des produits disponibles à la vente (ou qui le seront bientôt), ainsi que de gérer une liste de clients et leurs addresses liées.

2 - Généralités :

L'API nécessite de s'authentifier pour pouvoir être utilisée. Le protocole Oauth2 à, pour cela été retenu (Facebook est le fournisseur du service d'authentification retenu). Pour utiliser l'API, vous devrez donc transmettre à chaque requête votre clef d'API Facebook, pour authentification, dans le header de la requête, de la manière suivante :

Authorization : VOTRE_CLEF_D_API_FACEBOOK

Un autre header optionnel cette fois, peut être transmis. Il permet de choisir le format des données reçues, ainsi que la version de l'API utilisée. Pour l'instant, l'API ne peut transmettre des données qu'en JSON, et ne dispose que d'une seule version, mais cela sera porté à évoluer dans le temps. Voici la manière de faire :

Accept : application/json;version=1.0

3 – Les points d'accès :

3.1 – Articles :

3.2.1 GET /articles

Ce point d'accès permet d'obtenir la liste de tous les articles disponibles à la vente, ou bientôt disponibles. Des paramètres peuvent être passés en « Query String » afin de gérer la pagination, l'ordre et le tri par marques.

Filtres

brand:

- Requirement: \w+
- Description: The brand to search for

order:

- Requirement: asc|desc
- Description: Sort order (asc or desc)
- Default: asc

limit:

- Requirement: \d+

•Description: Max number of products per page.

•Default: 10

page:

•Requirement: \d+

•Description: The pagination offset

•Default: 1

isAvailable:

•Requirement: TRUE|FALSE|true|false|1|0

•Description: Availability for the product

•Default: TRUE

availabilityDate:

•Requirement: ^(0[1-9]|[1-2][0-9]|3[0-1])-(0[1-9]|1[0-2])-[0-9]{4}$

•Description: Availability date of the product (JJ-MM-AAAA)

•Default: The current date

Status code :

- 200 : Tout c'est bien passé.
- 404 : N'a pas trouvé d'articles correspondant à la recherche.

3.2.2 GET /articles/details/{id}

Il permet d'obtenir le détail d'un seul article, identifié par son identifiant. Le paramètre id situé dans l'url appelé doit obligatoirement être une variable de type « integer ».

Status code :

- 200 : Tout c'est bien passé
- 404 : L'article demandé n'a pas été trouvé.

3.2.3 GET /articles/{id}

Permet la consultation d'une famille de produits en particulier, identifié par son identifiant. Le paramètre id situé dans l'url appelé doit obligatoirement être une variable de type « integer ».

Status code :

- 200 : Tout c'est bien passé
- 404 : L'article demandé n'a pas été trouvé.

3.2 - Customers :

3.2.4 GET /customers

Ce point d'entrée permet de consulter la liste des utilisateurs de votre site. Des paramètres peuvent être passés en « Query String » afin de gérer les recherches par mail, voir les utilisateurs ayant un compte actif ou non, l'ordre d'affichage ainsi que la pagination.

Filtres

mail

- Requirement

  ^[^\W][a-zA-Z0-9_]+(\.[a-zA-Z0-9_]+)*\@[a-zA-Z0-9_]+(\.[a-zA-Z0-9_]+)*\.[a-zA-Z]{2,4}$
- Description The mail to search for

order

- Requirement asc|desc
- Description Sort order (asc or desc)
- Default asc

limit

- Requirement \d+
- Description Max number of products per page.
- Default 10

page

- Requirement \d+
- Description The pagination offset
- Default 1

isAvailable

- Requirement TRUE|FALSE|true|false|0|1
- Description Availability of the customer (TRUE or FALSE)
- Default TRUE

Status code :

- 200 : Tout c'est bien passé.
- 404 : N'a pas trouvé d'utilisateur correspondant à la recherche.

3.2.5 POST /customers

Permet l'ajout d'un utilisateur en base de données.

L’envoi des données ce fait par l'envoi d'un objet JSON dans le body de la requête. L'objet devra avoir la structure suivante :


```
{
   "password": "my_encoded_password",
   "salt": "my_encoded_salt",
   "is_checked": true,
   "delivery_addresses": [
      {
         "address1": "50 rue de la place",
         "address2": null,
         "address3": null,
         "city": {
            "id": 1,
            "name": "Limoux",
            "zip_code": "11300",
            "country": {
               "name": "France"
            }
         },
         "is_available": true,
         "is_default": true
      },
      {
         "address1": "40",
         "address2": "rue de la place",
         "address3": null,
         "city": {
            "id": 1,
```

```
            "name": "Limoux",
            "zip_code": "11300",
            "country": {
                "name": "France"
            }
        },
        "is_available": true,
        "is_default": false
    }
  ],
  "name": "Doe",
  "surname": "John",
  "billing_address": {
    "address1": "30 rue de la Mairie",
    "address2": null,
    "address3": null,
    "city": {
        "name": "Limoux",
        "zip_code": "11300",
        "country": {
            "name": "France"
        }
    },
    "is_available": true,
    "is_default": true
  },
  "phone": "0468000050",
  "cell_phone": null,
  "mail": "john.doe@monmail.com",
  "is_available": true
}
```


Le mail devrat être unique.

Status code :

- 200 : Tout c'est bien passé.
- 400 : Le JSON envoyé n'est pas valide.

3.2.6 DELETE /customers/{id}

Permet la suppression d'un utilisateur. En fait, on désactive le compte de l'utilisateur, en mettant son attribut isAvailable à False. Ce qui permet de le retrouver au besoin (réactivation du compte, suivi de l'historique des commandes etc...).

Le paramètre id permet d'identifier l'utilisateur sur lequel agir. Le paramètre appelé doit obligatoirement être une variable de type « integer ».

Status code :

- 204 : Tout c'est bien passé.
- 404 : N'a pas trouvé d'utilisateur correspondant à la recherche.

3.2.7 GET /customers/{id}

Permet de récupérer les données d'un seul utilisateur, identifié par son identifiant, passé dans l'url, en paramètre id.

Status code :

- 200 : Tout c'est bien passé
- 404 : L'utilisateur demandé n'a pas été trouvé.

3.2.8 PUT /customers/{id}

Permet la modification d'un utilisateur existant. Tous les paramètres (sauf l'id) peuvent être modifiés.

L'utilisateur sur lequel agir est identifié par le paramètre id situé dans l'url. Ce paramètre doit être de type « integer ».

Les données seront envoyées dans le body de la requête, sous la forme d'un objet JSON de type utilisateur. La structure du JSON à envoyer est la même que pour un ajout (méthode POST) vu plus haut.

Status code :

- 200 : Tout c'est bien passé.
- 400 : Le JSON envoyé n'est pas valide.
- 404 : l'utilisateur demandé n'a pas été trouvé.

3.3 - Address :

3.2.9 DELETE /address/{address}

Ce point d'entrée permet la suppression (soft delete comme pour les utilisateurs), d'une addresse de livraison. Les adresses de facturation ne peuvent pas être supprimées, elles ne peuvent être que modifiées.

L'adresse à supprimer est identifiée par le paramètre address situé dans l'url. Il représente l'identifiant de l'adresse, et doit être une variable de type « integer ».

Status code :

- 204 : Tout c'est bien passé.
- 404 : l'adresse demandée n'a pas été trouvée.

3.2.10 POST /address/{customer}

Permet l'ajout d'une adresse pour un utilisateur. L'utilisateur sur lequel ajouter l'adresse est identifié par le paramètre customer situé dans l'url. Il doit être de type « integer ».

Les données correspondant à l'adresse doivent être envoyées dans le body de la requête, et représentent un objet JSON de type adresse, respectant le format suivant :

{
"address1": "60 rue de la place",
"address2": null,
"address3": null,
"city": {
"id": 1,
"name": "Limoux",
"zip_code": "11300",
"country": {

```
            "name": "France"
        }
    },
    "is_available": true,
    "is_default": false
}
```


Status code :

- 201 : Tout c'est bien passé.
- 400 : Le JSON envoyé n'est pas valide.
- 404 : L'utilisateur demandé, est introuvable.

3.2.11 PUT /address/{id}

Ce point d'entrée permet la modification d'une adresse. L'adresse à modifier est identifiée par son id passé dans l'url. Cet id doit être de type « integer ».

Les données doivent être transmises dans le body de la requête, sous forme d'un objet JSON de type adresse, du même format que pour l'ajout (requête POST).

Status code :

- 201 : Tout c'est bien passé.
- 400 : Le JSON envoyé n'est pas valide.
- 404 : L'adresse demandée, est introuvable.