

Stimulation des appels API sans permissions et sans accès officiel : (made by Loïck Devismes and Hajar Benkhadra)

Introduction:

les étapes initiale:

- investigation sur une documentation d'une vrai API
- identifier ce qu'il faut et les standards à suivre
- documenter les endpoints disponible
- utiliser des outils comme: mckoon, postman
- créer de mock endpoint comme GET,POST...
- créer une réponse JSON
- configurer et tester les mock endpoints
- développer un client API local
- implementer API client ([client.js](#))
- implementer product service ([product.js](#))
- tester les validation
- valider le comportement de l'API

api mock guide :

<https://mockoon.com/articles/what-is-api-mocking/>

mockoon documentation :

<https://mockoon.com/tutorials/getting-started/>

mockoon installation link:

<https://mockoon.com/download/>

postman website with guide:

<https://learning.postman.com/docs/introduction/overview/>

postman installation link:

<https://www.postman.com/downloads/>

différence entre Mockoon et Postman:

Mockoon:fake api server	Postman:api testing tool
Créer une API factice avec un faux serveur backend pour simuler le comportement d'une API avec des réponses prédéfinies ou existantes (réponse JSON mentionnée ci-dessus), crée des points de terminaison fictifs sans nécessiter de code backend. Serveur mock local Espace non partagé (Live Share)	Test d'API (réelles, non simulées, avec un serveur backend existant),Interagit avec une API en direct en envoyant des requêtes,Mieux utilisé pour le débogage, la documentation et l'automatisation des tests d'API,Comprend des requêtes HTTP et une authentification,Stocke des variables

	d'environnement pour différentes configurations, Automatise les tests d'API avec des scripts (JS), Prend en charge la surveillance des API, C'est un espace partagé
--	---

Mockoon ne peut pas remplacer Postman, et inversement :

- Mockoon n'envoie pas de requêtes API réelles.
- Postman dispose d'une fonctionnalité de serveur mock, mais elle est moins flexible que celle de Mockoon pour le mock local.
- Postman est plus adapté aux tests d'API qu'au mock.
- Mockoon ne peut pas être intégré dans du code.

1. Etude de l'accès à L'API d'Aliexpress

Sur le site open service Aliexpress ([Getting Started - AliExpress Open Platform](#)) Il est cité qu'après la création d'un compte développeur **Pour avoir accès à l'API**, il est nécessaire de remplir un formulaire d'information. Ce dernier demande des informations que nous n'avons pas encore tel que: un extrait Kbis et le numéro SIREN de l'entreprise. ;

openseservice.aliexpress.com/register?spm=a2o9v.11225312.0.0.394c2e32vdZjCC

AliExpress Overseas

Create your account [Already member? Login here.](#)

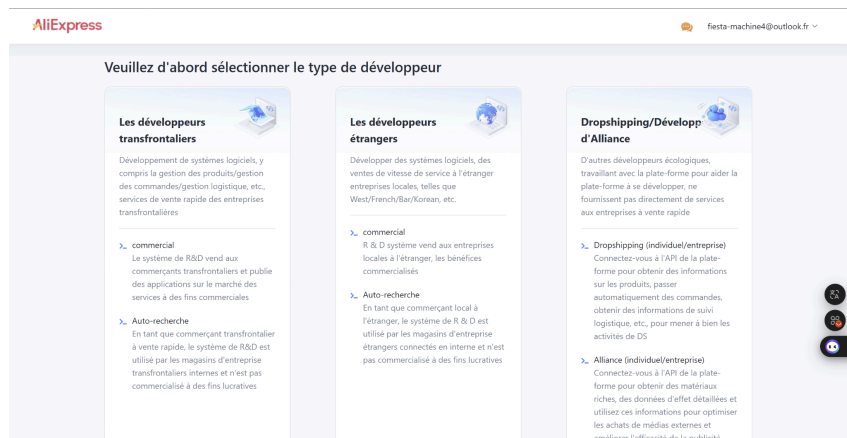
Email Address*
Please enter your email

Full Name*
First Last

Slider to get SMS Code

Password*
8+chars, incl. let., num., &special (excl. <=>space)

☐ Please carefully read [AliExpress.com Free Membership Agreement](#) before using AliExpress Open Platform ("AOP") services. Your full compliance with all provisions hereof is required. By proceeding to use the AOP, you shall be deemed to have fully read and understood the following terms and conditions, you agree and consent to be bound by this Agreement.



Il y aurait aussi une autre API mais moins complète pour les personnes affiliés à Aliexpress mais le processus est encore plus complexe et ce n'est pas ce que nous cherchons.

Et pour finir on retrouve l'API interne à Aliexpress qu'on peut scraper sur le site directement. Voici un exemple d'une réponse API d'un produit scrappé:

Cependant, cette réponse n'est très certainement pas ce qu'on recevra de l'API développeur, car elle sont largement différentes et que l'API interne à Aliexpress est privée.

Donc n'ayant pas de piste, notre seul espoir est de regarder sur internet. et donc j'ai demandé à ChatGPT et il m'a répondu qu'effectivement,

```

},
"separator": ";",
},
"SHIPPING": {
  "bgColor": "#FFCE9",
  "deliveryExt": "[{\\"d0\\":\\"9.30\\",\\"d1\\":\\"CNY\\",\\"itemSc",
  "deliveryLayoutInfo": [
    {
      "additionLayout": [
        {
          "blockGroup": "others-0",
          "blockType": "others",
          "componentId": "109530",
          "content": "Suivi disponible",
          "displayCondition": "tracking=visible",
          "distance": 1,
          "medusaKey": "tracking_available@others",
          "order": 4,
          "type": "textWithMark",
          "useInDeliveryOptionPanel": true,
          "useInDetailFirstScreen": false,
          "useInSkuSwitch": false
        }
      ]
    }
  ],
  "bizData": {
    "patternId": 2566001,
    "logisticsComposeThreshold": "0,00E",
    "discount": 0,
    "tracking": "visible",
    "workDayOneFourBeforeCutTime": "no",
    "delayCompensationTime": "0",
    "displayTomorrow": "no",
    "itemScene": "newGlobal",
    "shipToCode": "FR",
    "thresholdOverZero": "no",
    "delayCompensationOriginalCurrency": "EUR",
    "delayCompensationOriginalAmount": "1",

```

Les API d'Aliexpress sont bien privées.
J'ai aussi demandé à quoi pourrait
ressembler une réponse vers l'API dev et
il m'a proposé ceci:

```
{
  "item_id": "1005004854830874",
  "title": "Montre connectée intelligente Bluetooth",
  "main_image": "https://ae01.alicdn.com/kf/HTB1.jpg",
  "price": {
    "original": "49.99",
    "discounted": "29.99",
    "currency": "USD"
  },
  "shop": {
    "name": "SmartGadget Official Store",
    "seller_id": "123456789"
  },
  "description": "<p>Montre avec écran AMOLED...</p>",
  "images": [
    "https://ae01.alicdn.com/kf/HTB1.jpg",
    "https://ae01.alicdn.com/kf/HTB2.jpg"
  ],
  "attributes": [
    {
      "name": "Couleur",
      "value": "Noir"
    },
    {
      "name": "Étanche",
      "value": "Oui"
    }
  ],
  "shipping": {
    "countries_available": ["FR", "US", "ES"],
    "estimated_delivery": "7-15 jours",
    "cost": "0.00"
  },
  "reviews": {
    "average_rating": 4.6,
    "review_count": 234
  }
}
```

Ce qui est le plus probable et fiable qu'on peut trouver. (A considérer que beaucoup d'éléments manquent et que la réponse ressemblera plus ou moins à ça.)

Ce qui veut dire que nous ne pouvons que **spéculer sur la méthode d'interprétation de l'API**. Car nous n'avons aucun indice sur les appels à produire ainsi que sur les résultats donnés.

On peut noter qu'il n'y a aucune information par rapport à l'ajout au panier et aux ordres d'achats.

2. Identifications des fonctionnalités

Page d'accueil:

- Catégories simples avec 3-4 exemples de produits en vogue dans cette dernière catégories. éléments comme: petite photo, prix, réduc, nom.
- produit d'exemples (certainement venant d'un algorithme de recommandations) même éléments que l'appel de catégories.

Page de produit:

- Produit détaillés: prix nom, desc, prix de livraison, mini desc, avis, nombre d'avis, note, nmbr vendu, couleur, taille (si vêtement ou accessoire), images
- Articles similaires: liés à un algo de recommandation.

Page de Catégories:

- Catégories complexe: multiples produits d'une catégorie, certainement suivant aussi un algo de recommandation

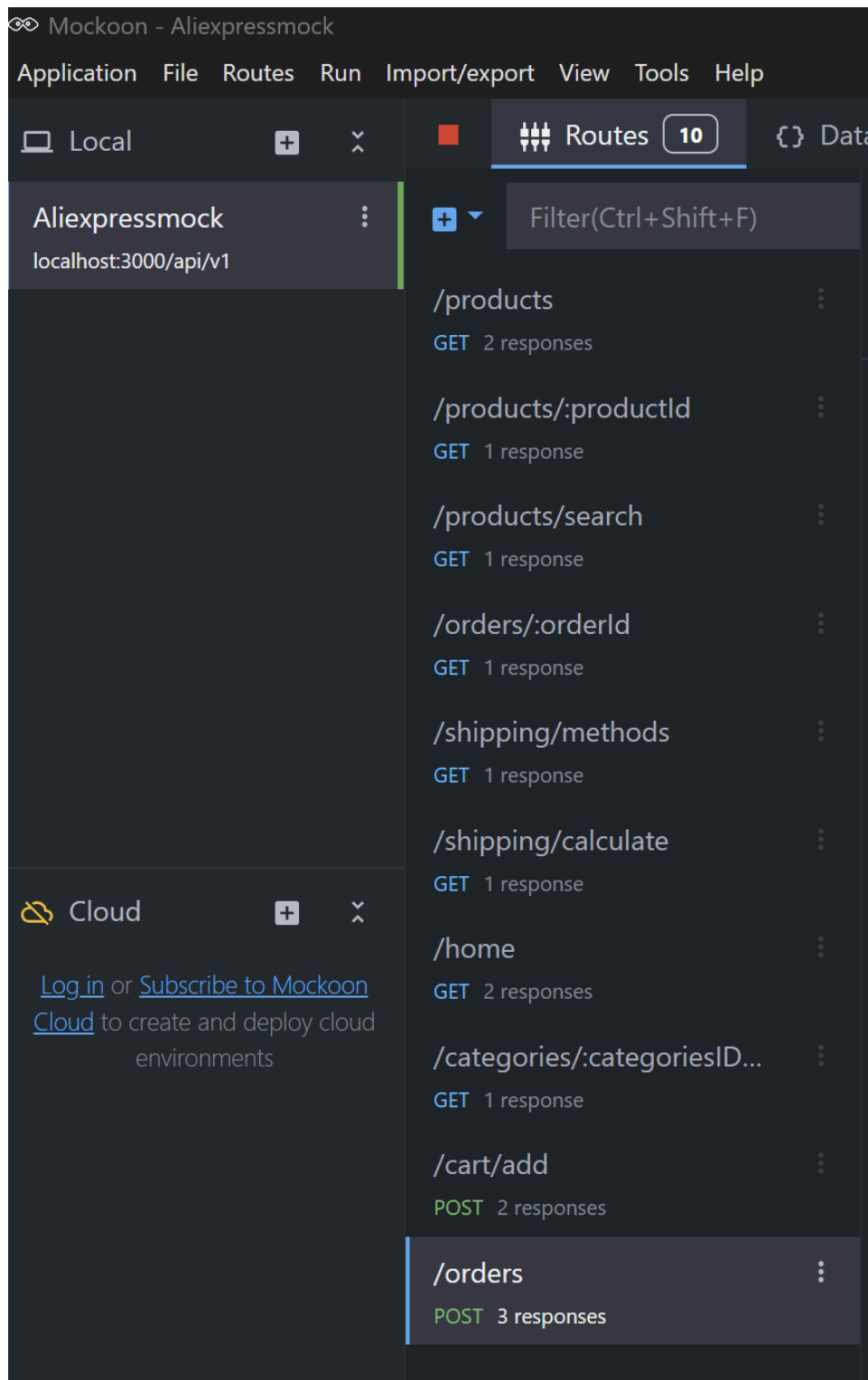
On peut conclure, qu'il y a 3 types de recherches différentes:

- Les produits: simples pour les deals du jour ou autre, et complexe pour les pages de produit
- Les catégories: simples pour une présentation de la catégories, et complexe pour les pages associés
- Les recommandations: Basés sur un algorithme extrêmement complexe.

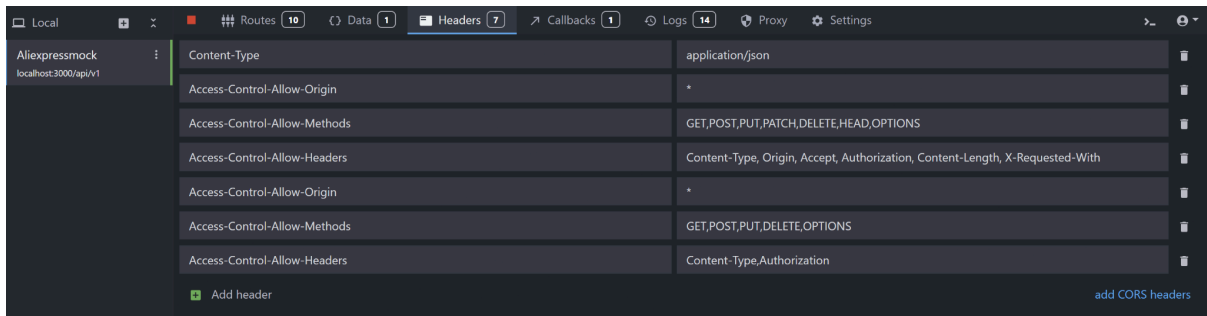
On peut aussi noter des calls à l'API pour mettre un objet dans le panier & placer un ordre d'achat.

3. Création d'une MockAPI sur Mockoon:

premièrement on met un environnement pour notre API et on configure son URL (localhost:3000/api/v1) et puis des routes HTTP(GET,POST...):



puis CORS headers qui sera appliquer pour tout les routes :



apres on lance le server et on récupère les url si ça marche sinon on recevoir des erreurs (techniquement ça à marché)

TO BE TESTED :

in the frontend we try to add this for a direct api call :

```
javascript Copy Download

// In your React/Vue/Angular component
const API_BASE = "http://localhost:3000/v1"; // Mockoon default

// Example: Fetch homepage data
async function loadHomeData() {
  try {
    const response = await fetch(`${API_BASE}/home`, {
      headers: {
        'X-API-Key': 'your-mock-key' // If authentication is enabled
      }
    });
  }

  if (!response.ok) throw new Error('API error');
  return await response.json();
} catch (error) {
  console.error("API Error:", error);
}
```

or through axios:

```

javascript
Copy Download

// api.js
import axios from 'axios';

const mockAPI = axios.create({
  baseURL: 'http://localhost:3000/v1',
  headers: {
    'X-API-Key': 'your-mock-key',
    'Content-Type': 'application/json'
  }
});

// Product Service
export const ProductService = {
  getProduct: (id) => mockAPI.get(`/products/${id}`),
  getCategoryProducts: (categoryId) => mockAPI.get(`/categories/${categoryId}/products`)
};

```

Connect to backend with :

```

javascript
Copy Download

// server.js
const axios = require('axios');

const MOCK_API = 'http://localhost:3000/v1';

// Proxy route example
app.get('/api/products/:id', async (req, res) => {
  try {
    const response = await axios.get(`${MOCK_API}/products/${req.params.id}`);
    res.json(response.data);
  } catch (error) {
    res.status(error.response?.status || 500)
      .json(error.response?.data || { error: "API failure" });
  }
});

```

environment configuration:

```

javascript
Copy Download

// config.js
export const API_CONFIG = {
  baseURL: import.meta.env.VITE_API_URL || 'http://localhost:3000/v1',
  isMock: true // Toggle between mock/real API
};

// Usage:
const api = axios.create({
  baseURL: API_CONFIG.baseURL
});

```