Projet 3:

Currency converter API

Adresse Github:

https://github.com/julfust/Currency-converter

1 Analyse client:

1.1 Money Value

Money Value est une start-up agissant dans le domaine de la finance spécialisée dans le marché des changes.

1.2 Problématique

Afin de pouvoir réaliser les opérations les plus efficaces, les employés de l'entreprise ont besoin d'un outil leur permettant de suivre de façon organisée les différents taux de conversion entre les devises.

Pour répondre à cette demande nous avons été chargés de créer une application web permettant de créer et de modifier une liste de taux de conversions entre différentes devises.

Cet outil devra permettre d'augmenter les profits de l'entreprise en améliorant l'efficacité du travail de ses employés.

2 Choix technologique

2.1 Outils Back-End

Laravel:
Laravel est un framework back-end php permettant de créer des applications web et des API. Il est entièrement développé en

programmation orientée objet et respecte le design pattern Model-Vue-Controller.

Laravel sanctum:
Laravel sanctum est une librairie qui met à disposition un système d'authentification spécialement adapté à l'architecture web-client et au développement d'API.

2.2 Outils Front-End

Vue.js:
Vue.js est un framework javascript permettant de construire des applications web monopage (SPA). Lui aussi conçu selon le principe Model-Vue-Controller, il met à disposition du développeur une grande variété d'outils permettant d'accélérer le développement d'interface web client.

Tailwind.css:
Tailwind.css est une librairie css permettant d'utiliser un ensemble de classes utilitaires pré-faites, ce qui permet d'accélérer grandement le temps de développement. De plus, la librairie est très flexible et peut s'intégrer facilement avec d'autres outils complémentaires.

Element Plus:
Element Plus est une libraire UI, elle met à disposition un ensemble de composant d'interface utilisateur incluant l'intégration visuel et une

logique fonctionnelle. Là encore, cela permet de réduire grandement le temps de développement Front-End de l'application.

Axios:
Axios est une librairie javascript utilisée pour les appels serveur. Elle fournit un client http basé sur les promesses permettant de réaliser différents types de requête. Elle simplifie ainsi grandement tout le paramétrage de la connexion au serveur sur la partie Front.

3 Evaluation du temps de travail

Poste de développement	Temps de travail
Migrations + modèles + seeders (base de donnée)	1 + 1/2 jour
Système d’authentification	1 jour
CRUD paire de conversion	3 jour
Visualisation requêtes	1/2 jour
Requête API (sans interface)	1 jour

4 Liste fonctionnelle

4.1 Système d’administration (interface inclus)

- Système d’authentification (login)
- Visualisation de la liste des paires de conversion
- Ajout de paire de conversion
- Modification de paire de conversion (devises et taux)
- Suppression de paire de conversion
- Visualisation du nombre de requêtes effectuées pour chaque paire

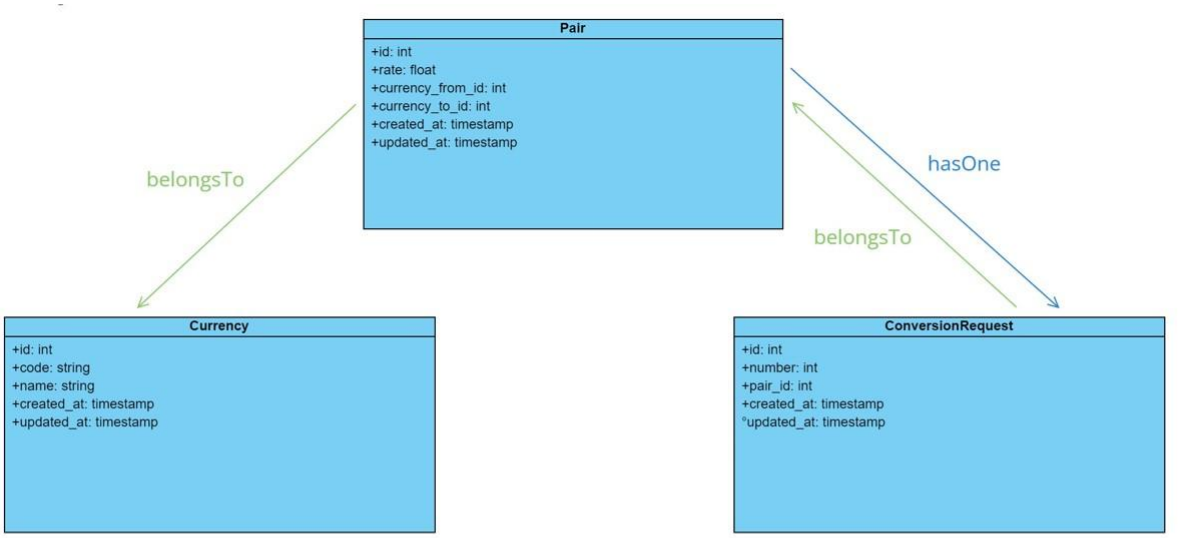
4.2 Requête API (sans interface)

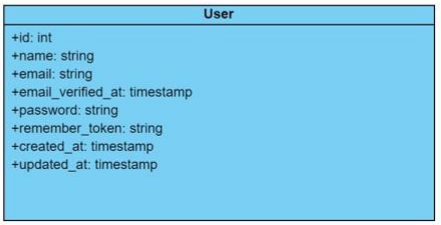
- Vérification du fonctionnement du service
- Récupération de la liste des paires de conversion supportées
- Conversion d’une quantité de devise suivant une paire existante

5 Recettage

Fonctionnalité	Status
Système d'authentification (login)	Incomplète: -> Guard de navigation sur l'espace authentifié non mis en place
Visualisation de la liste des paires de conversion	Opérationnelle
Ajout de paire de conversion	Opérationnelle
Modification de paire de conversion	Opérationnelle
Suppression de paire de conversion	Opérationnelle
Visualisation du nombre de requêtes effectuées pour chaque paire	Opérationnelle
Vérification du fonctionnement du service	Opérationnelle
Récupération de la liste des paires de conversion supportées	Opérationnelle
Conversion d'une quantité de devise suivant une paire existante	Opérationnelle

6 Diagramme base de données





7 Documentation de l’API

Pair: Correspond à une paire de conversion

PairResource:

```php
class PairResource extends JsonResource
{
    /**
     * Transform the resource into an array.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return array|\Illuminate\Contracts\Support\Arrayable|\JsonSerializable
     */
    public function toArray($request)
    {
        return [
            "id" => $this->id,
            "currencyFrom" => [
                "id" => $this->currencyFrom->id,
                "code" => $this->currencyFrom->code,
                "name" => $this->currencyFrom->name,
            ],
            "currencyTo" => [
                "id" => $this->currencyTo->id,
                "code" => $this->currencyTo->code,
                "name" => $this->currencyTo->name,
            ],
            "rate" => $this->rate
        ];
    }
}
```

Structure données PaireRessource

7.1 URL: http://localhost:8000/api/pairs

Méthode: GET

Fonctionnalité: retourne la liste de toutes les paires de conversion

Données retournées: PairResource::collection

Messages d’erreurs:

1) Utilisateur non authentifié

URL: http://localhost:8000/api/pairs

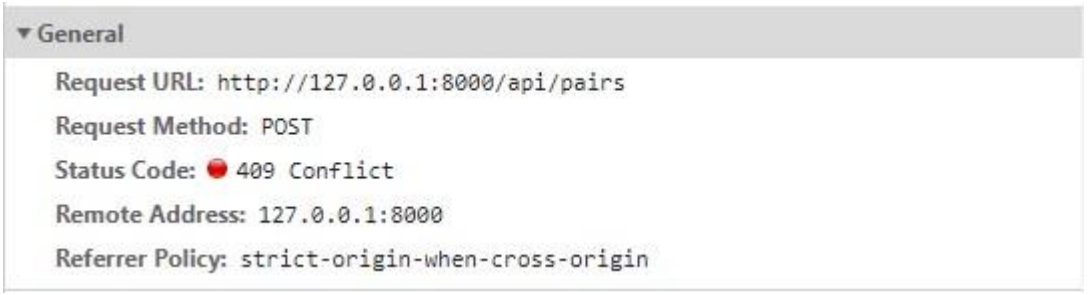
Méthode: POST

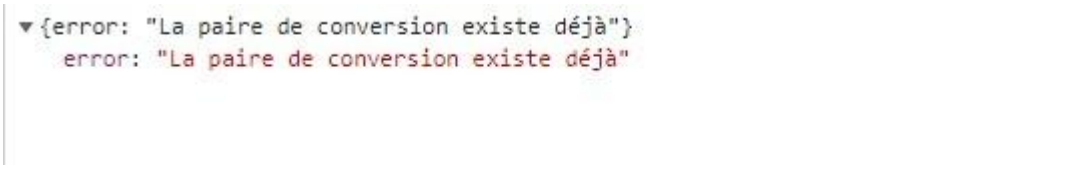
Fonctionnalité: Crée et retourne une nouvelle paire de conversion

Données retournées: new PairResource()

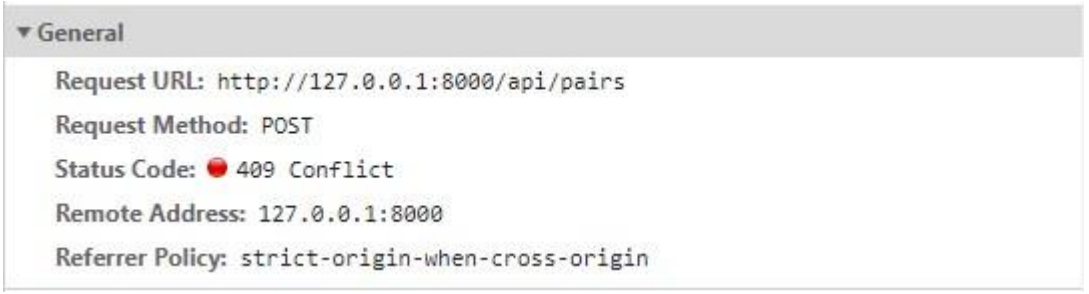
Messages d’erreurs:

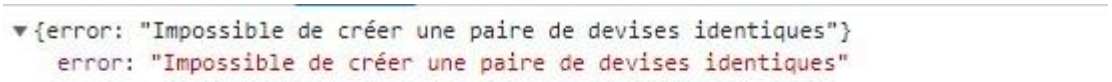
1) Création d’une paire déjà existante:





2) Création d’une paire avec 2 devises identiques:





URL: http://localhost:8000/api/pairs/{pairId}

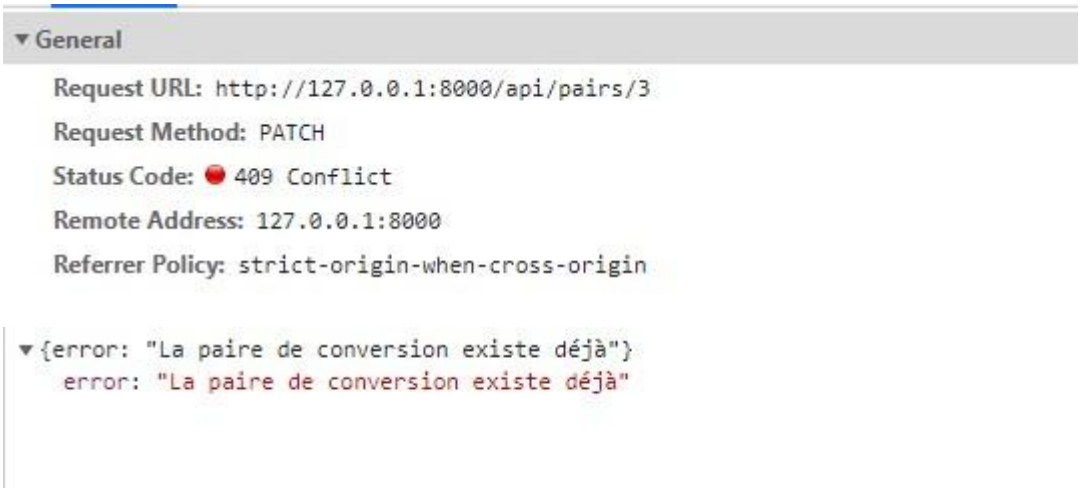
Méthode: PATCH

Fonctionnalité: Modifie une paire de conversion. Retourne les nouvelles données de la paire modifiée

Données retournées: new PairResource()

Messages d’erreurs:

1) Génération d’une paire déjà existante:



2) Sélection de deux devises identiques:

URL: http://localhost:8000/api/pairs/{pairId}

Méthode: DELETE

Fonctionnalité: Supprime une paire de conversion. Renvoie un booléen à true si la paire à bien été supprimée

Messages d’erreurs: Aucun

Currency: Correspond à une devise

CurrencyRessource:

```php
<?php

namespace App\Http\Resources;

use Illuminate\Http\Resources\Json\JsonResource;

class CurrencyResource extends JsonResource
{
    /**
     * Transform the resource into an array.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return array|\Illuminate\Contracts\Support\Arrayable|\JsonSerializable
     */
    public function toArray($request)
    {
        return [
            "id" => $this->id,
            "code" => $this->code,
            "name" => $this->name
        ];
    }
}
```

Structure données CurrencyResource

URL: http://localhost:8000/api/currencies

Méthode: GET

Fonctionnalité: retourne la liste de toutes les devises

Données retournées: CurrencyResource::collection

Messages d’erreurs:

1) Utilisateur non authentifié:

URL: http://localhost:8000/api/currencies

Méthode: POST

Fonctionnalité: Crée et retourne une nouvelle devise

Données retournées: new CurrencyResource()

Messages d’erreurs:

URL: http://localhost:8000/api/currencies/{currencyId}

Méthode: PATCH

Fonctionnalité: Modifie une devise. Retourne les nouvelles données de la devise modifiée

Données retournées: new CurrencyResource()

Messages d’erreurs:

Méthode: DELETE

Fonctionnalité: Supprime une devise. Renvoie un booléen à true si la devise à bien été supprimée

Messages d’erreurs: Aucun

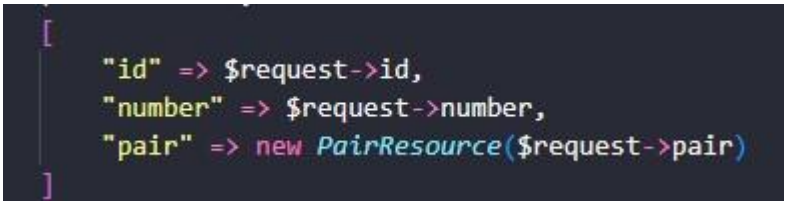
ConversionRequest: Correspond à un nombre de requêtes de conversion de devise effectué sur une paire

URL: http://localhost:8000/api/conversion-request

Méthode: GET

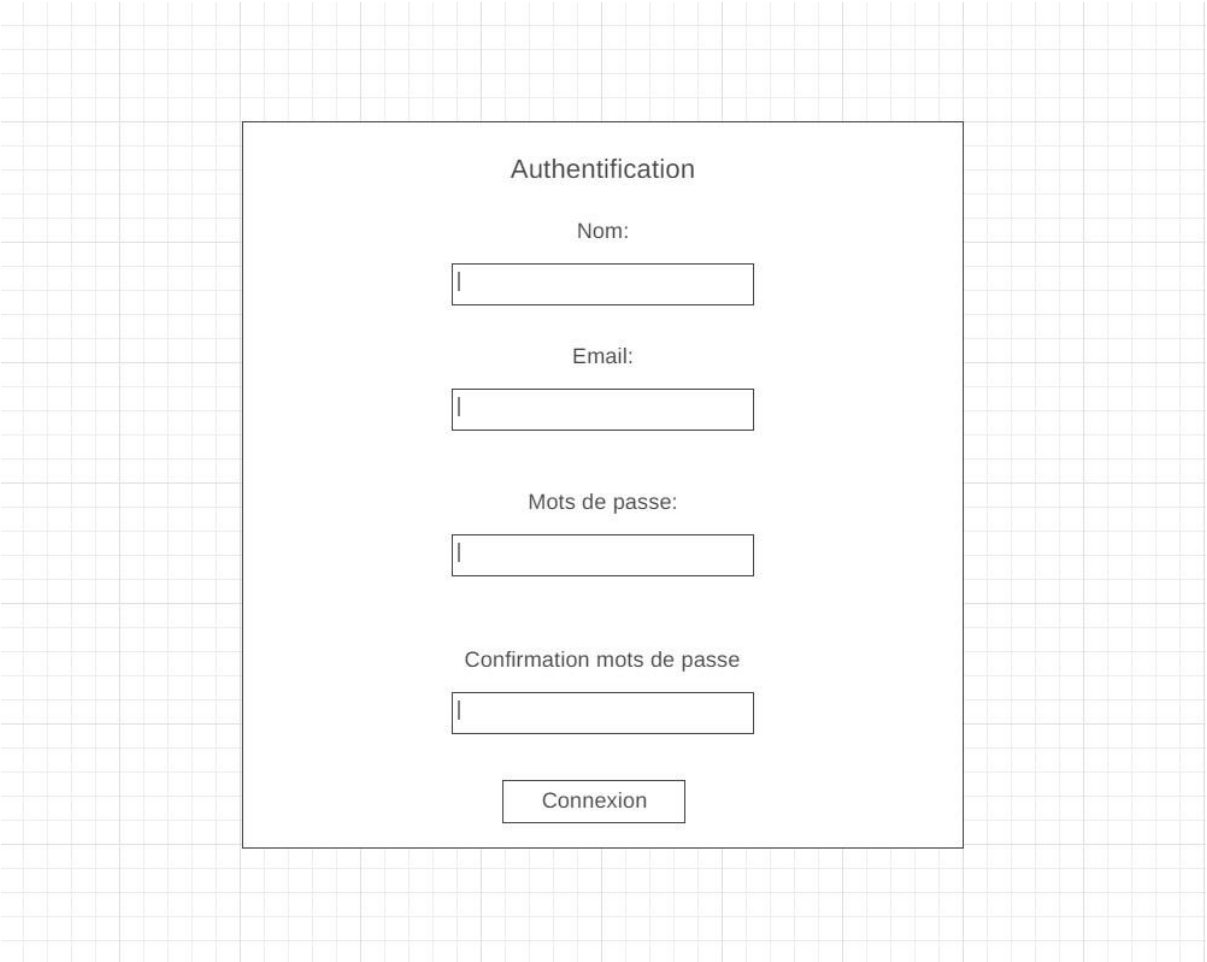
Fonctionnalité: retourne la liste du nombre de requêtes pour chacune des paires

Données retournées:

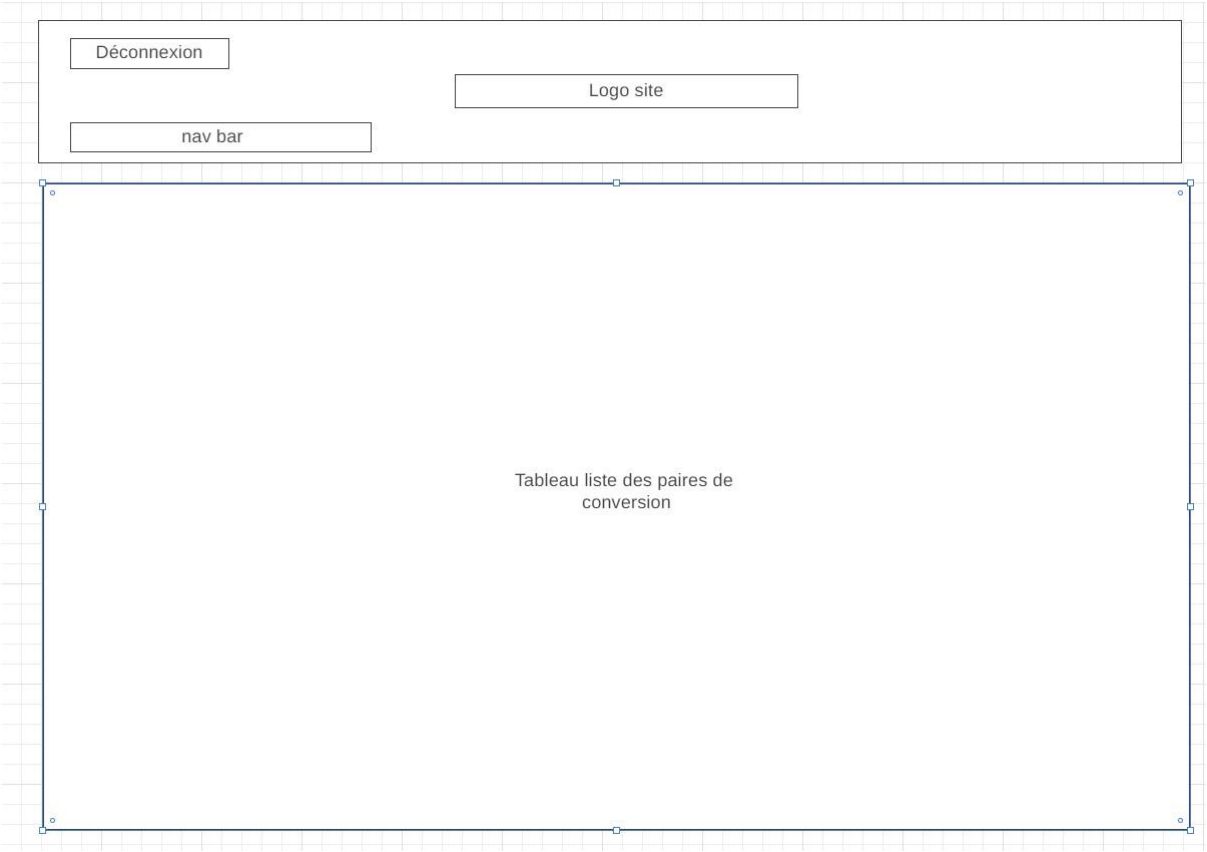


7 Wireframe interface admin

7.1 Formulaire de connexion:



7.2 Interface gestion paires de conversion



7.3 Interface visualisation nombre de requêtes

7.4 Interface gestion devises