Errata – Epreuve Intégrée – Boisdenghien Jordan

1. PaypalController.php

```php
public function success(Request $request)
{
    //PayPal Infos
    $provider = new PayPalClient;
    $provider->setApiCredentials(config('paypal'));
    $paypalToken = $provider->getAccessToken();

    $response = $provider->capturePaymentOrder($request->token);
    //dd($response); //uncomment to check response

    if(isset($response['status']) && $response['status'] == 'COMPLETED')
    {
        $cartItems = session()->get('cart', []);

        if(!$cartItems) //Verification but shoudln't trigger with our components but better to safeguard
        {
            return redirect()->route('paypal.checkout')->with('error', 'Votre panier est vide, impossible de payer !');
        }

        //Order and pivot table Figure_Order + Mailing :
        $order = $this->createOrder($cartItems, $response);
        if(!$order instanceof Order)
        {
            return redirect()->route('paypal.checkout')->with('error', 'Un problème est survenu lors de la commande !');
        }
        //Paypal_Transaction :
        $this->createPTransaction($order->id, $response);
        session()->forget('cart');
        return redirect()->route('order.success', ['orderId' => $order->id]);
    }
    else
    {
        return redirect()->route('paypal_cancel');
    }
    return redirect()->route('paypal_cancel'); //If we don't go out from the if, problem so cancel
}
```

La fonction "success" permet de "recevoir" la réponse de PayPal envoyée à la suite du paiement (avec la fonction "payment" dans le contrôleur). Elle crée, si la réponse de PayPal est bien reçue et qu'elle indique "COMPLETE", la commande dans la base de données et met à jour le stock puis elle va créer l'entrée dans la table "paypaltransactions" via des fonctions privées. Ces fonctions sont décrites dans la suite de ce document.

```php
public function cancel()
{
    return redirect()->route('paypal.checkout')->with('error', 'Le paiement PayPal a été annulé !');
}
```

La fonction cancel est utilisée en cas de problème lors du paiement, elle renvoie vers la page du paiement PayPal avec le message d'erreur qui indique qu'un problème est survenu lors du paiement.

```php
private function createOrder($cartItems, $paypalResponse)
{
    $totalPriceHT = 0; //Price without tax
    $totalPrice = 0;
    $selectedShippingId = Session::get('selectedshipping_id');
    $selectedShippingInfo = Shipping::findOrFail($selectedShippingId);
    $shippingCost = $selectedShippingInfo->price;
    $user = Auth::user(); //Checking if user is correctly logged in, technically it's impossible to go on this page not logged but "safeguard"
    if(!$user)
    {
        session()->flash('error', 'Vous devez être connecté pour pouvoir réaliser cet achat !');
        return redirect()->route('login');
    }

    foreach($cartItems as $figureId => $item) //Calculate HT price for order
    {
        $figure = Figure::findOrFail($figureId);
        $figure_price = $figure->price;
        $tva_rate = Tva::findOrFail($figure->tva_id)->purcent;
        $figure_priceHT = $figure_price / (1.0 + $tva_rate);
        $totalPriceHT += $figure_priceHT * $item['quantity'];
        $totalPrice += $figure_price * $item['quantity'];
    }

    $order = Order::create([ //Adding order in table orders
        'user_id' => $user->id,
        'status' => 'Payement Paypal validé',
        'price' => number_format($totalPriceHT, 2), //Without vat
        'pricettc' => number_format($totalPrice, 2), //With vat
        'shipping_at_date' => number_format($shippingCost, 2), //Price of shipping at order date
        'paypal_status' => $paypalResponse['status'],
        'shipping_id' => $selectedShippingId,
        'tracker' => null,
        'isVisible' => 1,
    ]);
```

```php
    //Adding to table FiguresOrders for each figures in the order
    foreach($cartItems as $figureId => $item)
    {
        $figure = Figure::findOrFail($figureId);

        //Updating quantity in stock
        if($figure && $figure->stock_qty >= $item['quantity'])
        {
            $figure->decrement('stock_qty', $item['quantity']);
        }
        else //In case of not enought quantity but shouldn't be possible
        {
            session()->flash('message', "La figurine {$item['name']} n'a pas assez de stock : {$figure->stock_qty}.");
            return redirect()->route('cart');
        }

        $priceTtcAtDate = $item['price'];
        $tva = Tva::FindOrFail($figure->tva_id)->purcent;
        $priceAtDate = number_format($item['price'] / (1.0+$tva), 2);

        $order->figures()->attach($figureId, [
            'price_at_date' => $priceAtDate,
            'pricettc_at_date' => $priceTtcAtDate,
            'quantity' => $item['quantity'],
        ]);
    }
```

```php
    //Mailing (we retake informations for each figure from pivot table)
    $orderDetails = FiguresOrders::where('order_id', $order->id)
    ->join('figures', 'figures.id', '=', 'figuresorders.figure_id')
    ->select('figures.name', 'figuresorders.price_at_date', 'figuresorders.pricettc_at_date', 'figuresorders.quantity')
    ->get();
    Mail::to(auth()->user()->email)->send(new OrderDetailsMail($order, $orderDetails));

    return $order;
}
```

La fonction "createOrder($cartItems, $paypalResponse)" va recevoir le panier de l'utilisateur ainsi que la réponse de PayPal. Elle va, du panier, créer l'entrée dans la table "orders" ainsi que

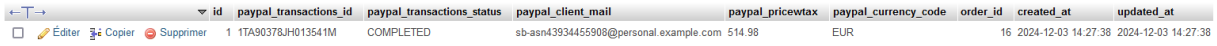
toute les entrées nécessaires dans la table pivot "figuresorders". Elle va ensuite créer le mail avec les détails de la commande et faire en sorte qu'il soit envoyé.

	id	user_id	status	price	pricettc	shipping_at_date	paypal_status	shipping_id	tracker	isVisible	created_at	updated_at
Éditer Copier Supprimer	16	5	Payement Paypal validé	413.21	499.98		15 COMPLETED	3	NULL	1	2024-12-03 14:27:36	2024-12-03 14:27:36

L'entrée est stockée telle que ci-dessus dans la base de données, table "orders".

```php
    private function createPTransaction($orderId, $paypalResponse)
    {
        PaypalTransaction::create(
            [
                'paypal_transactions_id' => $paypalResponse['id'],
                'paypal_transactions_status' => $paypalResponse['status'],
                'paypal_client_mail' => $paypalResponse['payer']['email_address'],
                'paypal_pricewtax' => $paypalResponse['purchase_units'][0]['payments']['captures'][0]['amount']['value'],
                'paypal_currency_code' => $paypalResponse['purchase_units'][0]['payments']['captures'][0]['amount']['currency_code'],
                'order_id' => $orderId,
            ]
        );
    }
}
```

La fonction "createPTransaction" va créer l'entrée dans la table "paypaltransactions" concernant le paiement associé à la commande. Elle prend de la réponse PayPal les différentes informations nous intéressant à conserver.



	id	paypal_transactions_id	paypal_transactions_status	paypal_client_mail	paypal_pricewtax	paypal_currency_code	order_id	created_at	updated_at
Éditer Copier Supprimer	1	1TA90378JH013541M	COMPLETED	sb-asn43934455908@personal.example.com	514.98	EUR	16	2024-12-03 14:27:38	2024-12-03 14:27:38

L'entrée est stockée telle que ci-dessus dans la base de données, table "paypaltrasanctions".

2. Edition d'une figurine

```blade
<div class="mb-4">
    <label for="image_path" class="block text-gray-700">Image :</label>
    <input type="file" id="image_path" accept="image/*" wire:model.defer="image_path" class="w-full mt-2 p-2 border rounder">
    @error('image_path') <span class="text-red-500"> {{ $message }}</span> @enderror
    @if($image_path)
        <p class="mt-2">Prévisualisation de l'image :</p>
        <img src="{{ $image_path->temporaryUrl() }}" alt="Preview" class="max-w-sm">
    @elseif($existing_path)
        <p class="mt-2">Image actuelle :</p>
        <img src="{{ Storage::url('' . $existing_path .'') }}" alt="Actuelle" class="max-w-sm">
    @endif
</div>
```

On ajoute un "elseif" avec un "$existing_path" (qui remplace $image_path dans le "mount" du composant Livewire associée à la vue, ce composant permet notamment de précharger les informations du formulaire), s'il y a déjà une image, on utilise son URL et non pas un URL temporaire (ce qui crée une erreur) et l'URL temporaire est utilisée si l'utilisateur remplace l'image déjà associée (et l'ancienne est supprimée dans le composant, voir ci-dessous).

```php
if($this->image_path)
{
    if($this->existing_path && Storage::disk('local')->exists($this->existing_path)) Storage::disk('local')->delete($this->existing_path);
    $img = $this->image_path->store('figures', 'public');
}
else
{
    $img = $this->existing_path;
}
```

Différence avec la version en ligne pour l'affichage :

```
<img src="{{ Storage::url('app/public/'. $existing_path . '') }}" alt="Actuelle" class="max-w-sm">
```

La version en ligne "trafique" l'URL afin de gérer le non-fonctionnement du symlink Artisan sinon les images ne sont pas affichées.

3. Affichage des transactions PayPal dans le "Dashboard Gérant"

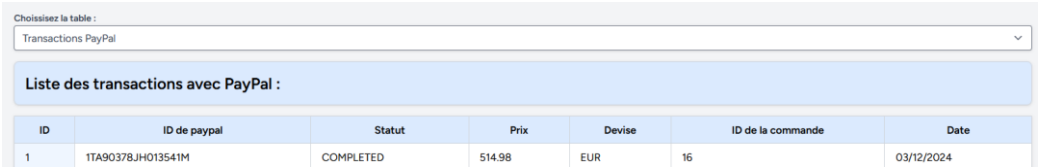
```
@elseif($selectedCrud == 'paypaltransactions')
    <h1 class="text-2xl font-bold mb-4 bg-blue-100 border border-gray-300 rounded-lg p-4 shadow">Liste des transactions avec PayPal :</h1>
    <table class="min-w-full bg-white shadow">
        <thead>
            <tr class="bg-blue-100">
                <th class="border border-gray-300 px-4 py-2">ID</th>
                <th class="border border-gray-300 px-4 py-2">ID de paypal</th>
                <th class="border border-gray-300 px-4 py-2">Statut</th>
                <th class="border border-gray-300 px-4 py-2">Prix</th>
                <th class="border border-gray-300 px-4 py-2">Devise</th>
                <th class="border border-gray-300 px-4 py-2">ID de la commande</th>
                <th class="border border-gray-300 px-4 py-2">Date</th>
            </tr>
        </thead>
        <tbody>
            @foreach ($paypaltransactions as $paypaltransaction)
                <tr>
                    <td class="border border-gray-300 px-4 py-2 bg-blue-50">{{ $paypaltransaction->id }}</td>
                    <td class="border border-gray-300 px-4 py-2">{{ $paypaltransaction->paypal_transactions_id }}</td>
                    <td class="border border-gray-300 px-4 py-2">{{ $paypaltransaction->paypal_transactions_status }}</td>
                    <td class="border border-gray-300 px-4 py-2">{{ $paypaltransaction->paypal_pricewtax }}</td>
                    <td class="border border-gray-300 px-4 py-2">{{ $paypaltransaction->paypal_currency_code }} </td>
                    <td class="border border-gray-300 px-4 py-2">{{ $paypaltransaction->order_id }}</td>
                    <td class="border border-gray-300 px-4 py-2">{{ $paypaltransaction->created_at->format('d/m/Y') }}</td>
                </tr>
            @endforeach
        </tbody>
    </table>
    <div class="mt-4">
        {{ $paypaltransactions->links() }}
    </div>
```

On affiche les transactions PayPal de la table "paypaltransactions" dans la vue du Dashboard si l'utilisateur choisit cette option dans le sélecteur.

```
else if($this->selectedCrud == 'paypaltransactions')
{
    $this->paypaltransactions = PaypalTransaction::all()->sortBy('id');
}
```

```
elseif($this->selectedCrud == 'paypaltransactions')
{
    $this->loadData();
    $paypaltransactions = $this->paginateCollection($this->paypaltransactions ,10);
    return view('livewire.gerant-dashboard', compact('paypaltransactions'));
}
```

On prend la collection des données des transactions PayPal via la fonction "loadData" (qui utilise le code de la première photo). La collection est ensuite triée par l'ID des transactions et on met en page 10 par 10 dans la fonction "render" (seconde image). On affiche la vue spécifiquement dédiée aux transactions PayPal (premier screen de ce point).



La vue dans le Dashboard est telle que ci-dessus. Notez que le prix inclus la livraison.

4. Différence du ".env" pour la version en ligne

```
APP_URL=https://rokuban.com/
```

L'URL est différente en ligne, il ne s'agit plus ici du localhost.

```
# PayPal API Mode

# Values: sandbox or live (Default: live)
PAYPAL_MODE=sandbox

# PayPal Setting & API Credentials - sandbox
PAYPAL_SANDBOX_CLIENT_ID=AbYkPb23ruUL19c5Qiz95VBVzZ1k3EG1L8rvUQ9haeN1B8aKEKata-cO0UQKQLFPrtvojvwpZEkCSe4w
PAYPAL_SANDBOX_CLIENT_SECRET=EOxtmeKhr18qmFDas6wH1GcCh2Fa7YzTTf7MTb5ko-6oQjvrAUeU1gxT_hxYqkqO_I8JGPFZZnEMjOY-

# PayPal Setting & API Credentials - live
# PAYPAL_LIVE_CLIENT_ID=
# PAYPAL_LIVE_CLIENT_SECRET=
# PAYPAL_LIVE_APP_ID=

# Payment Action. Can only be 'Sale', 'Authorization' or 'Order'
PAYPAL_PAYMENT_ACTION=Sale


# Currency. Default is USD. If you need to update it, then set the value through the PAYPAL_CURRENCY environment variable.
PAYPAL_CURRENCY=EUR

# Validate SSL when creating api client. By default, the value is great. To disable validation set to false.
# PAYPAL_VALIDATE_SSL=false
```

PayPal est configuré, ici en sandbox, pour fonctionner avec la version en ligne. Un ID et un mot de passe est donné à la configuration ainsi que d'autres instructions.

```
MAIL_MAILER=smtp
MAIL_HOST=smtp.hostinger.com
MAIL_PORT=465
MAIL_USERNAME=admin@rokuban.com
MAIL_PASSWORD=72/ABg3b!D9;tg2XW${m
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS="noreply@rokuban.com"
MAIL_FROM_NAME="${APP_NAME}"
```

Le mailer est celui de mon hébergeur, configurer selon ses instructions. Le compte utilisée ("MAIL_USERNAME") utilise un alias "noreply@rokuban.com" (géré par l'hébergeur) pour envoyer les mails qui sont générés par l'application web.