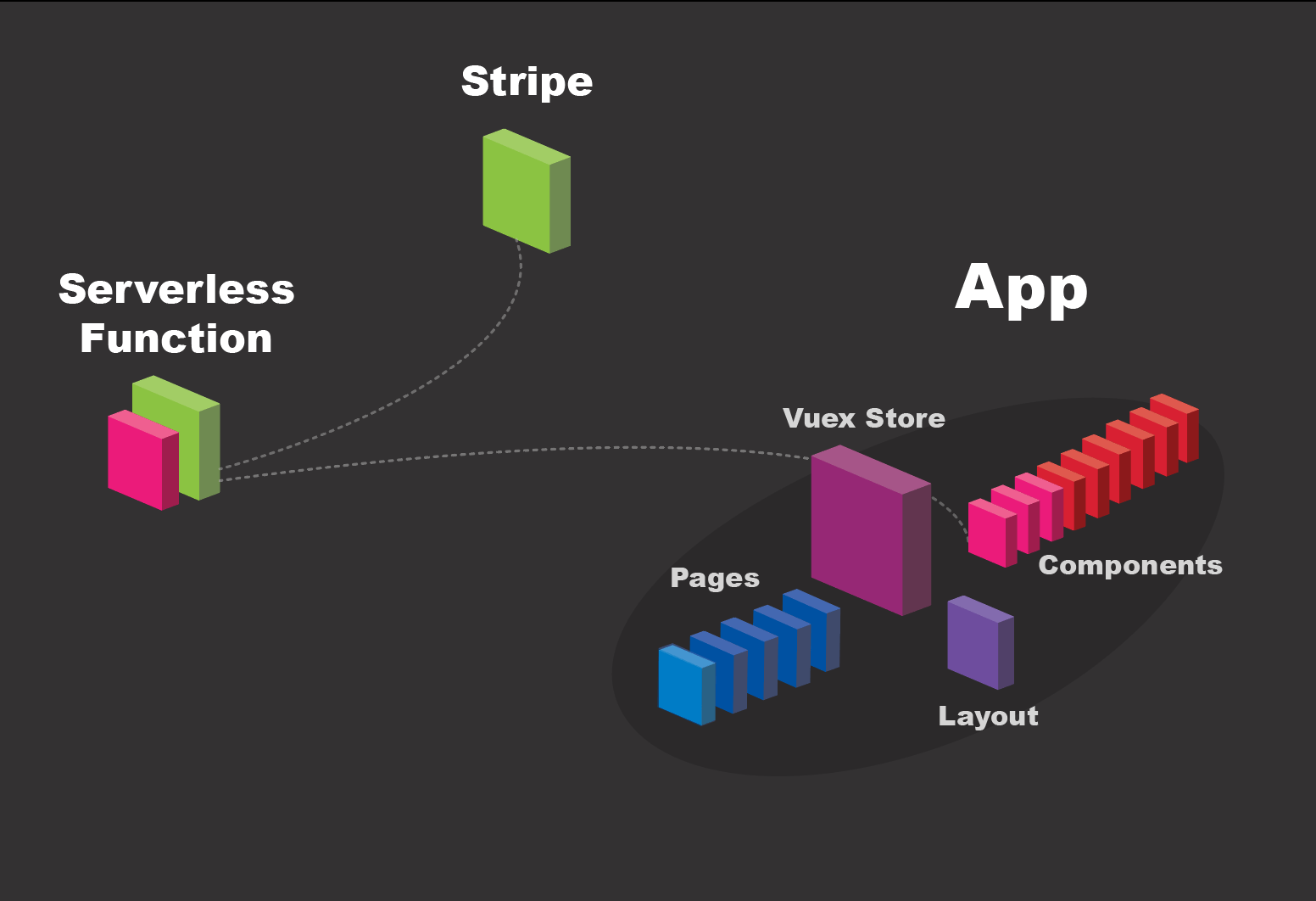
Atelier 9 : Shopping cart avec Vuex et payement Stripe



Nous allons construire un store avec Vuex. Dans ce store mis en place avec Vuex, nous détenons un manifeste de toutes nos données produits utilisées pour remplir les pages avec des articles. Nous utiliserons également ces informations pour remplir un objet de panier (actuellement vide) où des articles peuvent être ajoutés pour l'achat. Nous utiliserons ces données sur une page appelée « Cart.vue » dans le répertoire des pages.

Nous allons également utiliser Stripe. Stripe propose un produit appelé Elements. Elements vous permet d'intégrer le système de paiement de Stripe dans votre propre formulaire de paiement et de le personnaliser comme votre propre site pour une expérience cohérente.

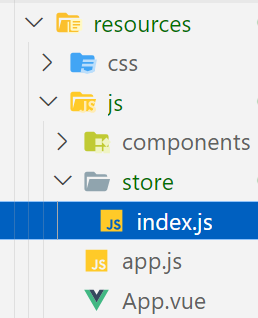
L’architecture globale est la suivante :



# Installation du package Vuex

npm install vuex@next

# Création du store





import { createStore } from 'vuex'

export default createStore({

  state: {

    cart:[],

    cartTotal:0,

  },

  mutations: {

   async initialiseStore(state) {

         if(localStorage.getItem('cart')){

          state.cart = JSON.parse(localStorage.getItem('cart'))

         }

         if(localStorage.getItem('cartTotal')){

         state.cartTotal = parseFloat(localStorage.getItem('cartTotal'))

         }

         return true;

    },

    async clearCart(state){

      if(localStorage.getItem('cart')){

      localStorage.removeItem('cart');

      state.cart =[];

    }

    if(localStorage.getItem('cartTotal')){

      localStorage.removeItem('cartTotal');

      state.cartTotal =0;

    }

    return true;

    },

    addCart(state,payload){

      const productData = payload;

      const productInCartIndex = state.cart.findIndex(

        (ci) => ci.product.id === productData.product.id

      );

      if (productInCartIndex < 0) {

      state.cart.push(payload)

      //calculating the total

      state.cartTotal = state.cart.reduce((accumulator, object) => {

        return parseFloat(accumulator) + parseFloat(object.product.prix\*object.qty);

      }, 0);

       //saving in web storage

      localStorage.setItem('cartTotal',JSON.stringify(state.cartTotal));

      localStorage.setItem('cart',JSON.stringify(state.cart));

    }

    },

    minusCart(state,payload){

     //update quantity

     const productData = payload;

     const productInCartIndex = state.cart.findIndex(

       (ci) => ci.product.id === productData.product.id

     );

     if (productInCartIndex >= 0) {

       state.cart[productInCartIndex].qty--;

     }

      //Recalculating the total

       state.cartTotal = state.cart.reduce((accumulator, object) => {

         return parseFloat(accumulator) + parseFloat(object.product.prix\*object.qty);

       }, 0);

     //saving in web storage

       localStorage.setItem('cartTotal',JSON.stringify(state.cartTotal));

       localStorage.setItem('cart',JSON.stringify(state.cart));

    },

    removeFromCart(state,payload){

      state.cart = state.cart.filter(function( obj ) {

        return obj.product.id !== payload.product.id;

      });

      //calculating the total

      state.cartTotal = state.cart.reduce((accumulator, object) => {

        return parseFloat(accumulator) + parseFloat(object.product.prix\*object.qty);

      }, 0);

       //saving in web storage

      localStorage.setItem('cartTotal',JSON.stringify(state.cartTotal));

      localStorage.setItem('cart',JSON.stringify(state.cart));

    },

    plusCart(state,payload){

      //update quantity

      const productData = payload;

      const productInCartIndex = state.cart.findIndex(

        (ci) => ci.product.id === productData.product.id

      );

      if (productInCartIndex >= 0) {

        state.cart[productInCartIndex].qty++;

      }

       //Recalculating the total

        state.cartTotal = state.cart.reduce((accumulator, object) => {

          return parseFloat(accumulator) + parseFloat(object.product.prix\*object.qty);

        }, 0);

      //saving in web storage

        localStorage.setItem('cartTotal',JSON.stringify(state.cartTotal));

        localStorage.setItem('cart',JSON.stringify(state.cart));

      },

  },

  actions: {},

  modules: {},

})



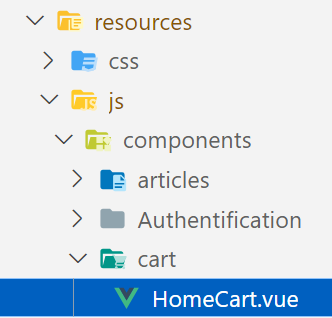
import store from "./store"

…

app.use(router).use(store);

# La page Home du cart

Créer le dossier cart ayant le fichier HomaeCart.vue



<template>

      <div class="row">

          <div class="col-md-12" style="display:flex;flex-wrap:wrap;justify-content:center">

            <div class="card" style="width: 18rem;margin:12px" v-for="prod in Articles" :key="prod.id">

              <img class="card-img-top" :src="prod.imageart" :alt=prod.designation  width="70">

              <div class="card-body">

                  <h5 class="card-title">{{ prod.designation }}</h5>

                  <p class="card-text">{{ prod.marque }}</p>

              </div>

              <div style="text-align:center">

                  <ul class="list-group list-group-flush">

                      <li class="list-group-item">Prix : {{ prod.prix }} TND</li>

                  </ul>

                  <button class="btn btn-warning" @click="addToCart(prod)" v-if="prod.qtestock>1">Add to Cart</button>

                  <button class="btn btn-default" v-if="prod.qtestock<=1">OUT OF SOLD</button>

              </div>

            </div>

          </div>

      </div>

  </template>

  <script setup>

    import axios from "../config/axios.js";

    import store from '../../store'

import { ref, onMounted } from 'vue';

const Articles = ref([]);

 onMounted(() => {

            getArticles();

            }

);

const getArticles=()=>{

                  axios

                .get('/api/articles/')

                .then(response => {

                                 Articles.value = response.data;

                });

            }

const    addToCart=(prod)=> {

              store.commit("addCart", {product:prod,qty:1});

               }

  </script>



import HomeCart from './components/cart/HomeCart.vue';

…

{

        path:'/shopping',

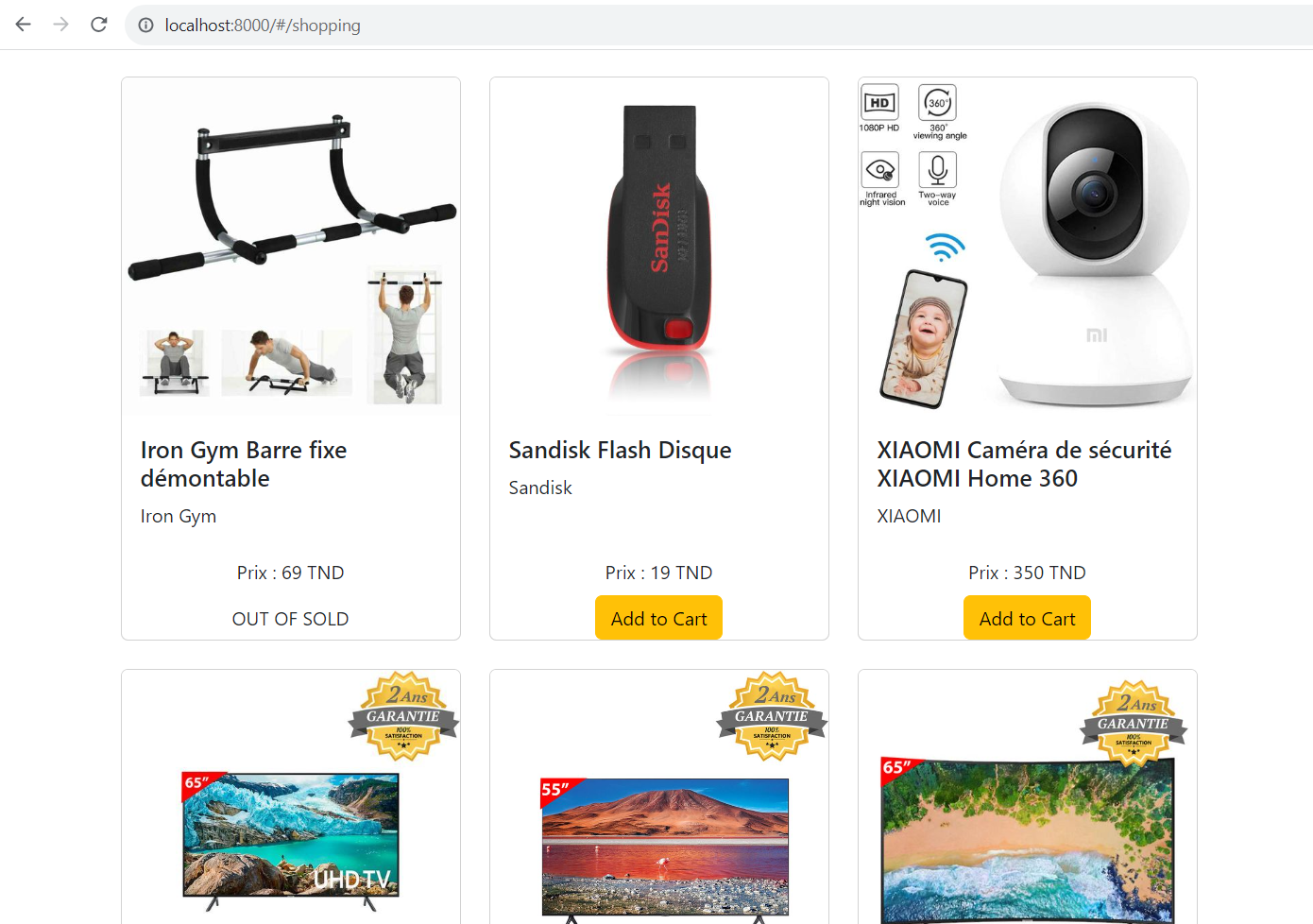
        name:'HomeCart',

        component:HomeCart

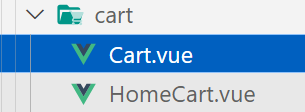
    },



 <router-link to="/shopping" class="nav-item nav-link"> Shopping Cart </router-link>



# Le panier



<template>

    <div v-if="$store.state.cart.length > 0" style="display:flex">

        <div class="col-10">

      <table>

       <tr v-for="(c) of $store.state.cart" :key="c.product.id">

         <td><img :src="c.product.imageart" :alt="c.product.designation" width="50"></td>

         <td>{{ c.product.designation }}</td>

         <td>- Prix : {{ c.product.prix }} TND </td>

         <td>- Quantité : <strong> {{ c.qty }} </strong></td>

         <td><button class="btn btn-dark" @click="plus(c)"> + </button></td>

         <td><button class="btn btn-warning" @click="minus(c)" v-if="c.qty > 1"> - </button>

             <button class="btn btn-danger" @click="removeFromCart(c)"> X </button>

         </td>

       </tr>

      </table>

       </div>

        <div class="col-10">

              <button class="btn btn-info"> CHECKOUT </button>

            <div>Total items : {{$store.state.cart.length }}</div>

            <div>Total price  : {{$store.state.cartTotal}}</div>

           <router-link :to="{name: 'HomeCart'}" > <button class="btn btn-success"> MORE </button> </router-link>

           <button class="btn btn-secondary" @click="clearCart()"> CLEAR </button>

       </div>

     </div>

     <div v-else>

               <div class="error-template container">

                 <h1>Oops!</h1>

                 <h2>No Products Found</h2>

                 <div class="error-details">Your cart is empty</div>

                 <div class="error-actions">

                   <a class="btn text-white">

                     <span class="glyphicon glyphicon-envelope"></span>

                     <router-link to="/shopping"

                       >Take me to Products Page</router-link

                     >

                   </a>

                 </div>

               </div>

   </div>

   </template>

   <script setup>

   import axios from "../config/axios.js";

   import store from '../../store'

import { ref, onMounted } from 'vue';

const Produits = ref([]);

onMounted(() => {

       store.commit('initialiseStore')

       console.log(store.state.cart)

       getProduits();

           }

);

 const   clearCart=()=> {

        store.commit("clearCart");

       }

 const  removeFromCart=(item)=> {

        store.commit("removeFromCart", item);

       }

 const  plus=(item)=> {

         Produits.value.map((pro)=>{

         if(pro.id==item.product.id){

           if(item.qty < pro.qtestock )

           {

            store.commit("plusCart", item);

           }

           else

                 alert("Quantité stock indisponible")

         }

         });

       }

 const minus=(item)=> {

            if(item.qty < 1 )

           {

            alert("Quantité stock non valable")

           }

           else

            store.commit("minusCart", item);

         }

const  getProduits=()=>{

               axios.get('/api/articles/').then(res => {

                   Produits.value = res.data;

               }).catch(error => {

                   console.log(error)

               });

          }

   </script>

   <style>

   .error-template {

     padding: 40px 15px;

     text-align: center;

   }

   .error-actions {

     margin-top: 15px;

     margin-bottom: 15px;

   }

   .error-actions .btn {

     margin-right: 10px;

   }

   </style>



import Cart from './components/cart/Cart.vue';

…

 {

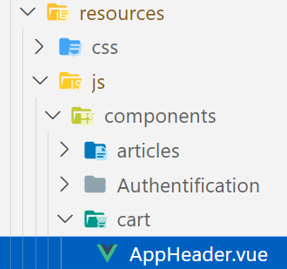
        path:'/cart',

        name:'Cart',

        component:Cart

    },

Puis Créer le fichier cart/AppHeader.vue :



<template>

<ul class="navbar-nav ms-auto mb-2 mb-lg-0">

    <li class="nav-item me-3">

      <router-link class="nav-link" :class="$route.name == 'Cart'? 'active':''" aria-current="page" :to="{ name: 'Cart' }">

        <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-cart4" viewBox="0 0 16 16">

        <path d="M0 2.5A.5.5 0 0 1 .5 2H2a.5.5 0 0 1 .485.379L2.89 4H14.5a.5.5 0 0 1 .485.621l-1.5 6A.5.5 0 0 1 13 11H4a.5.5 0 0 1-.485-.379L1.61 3H.5a.5.5 0 0 1-.5-.5zM3.14 5l.5 2H5V5H3.14zM6 5v2h2V5H6zm3 0v2h2V5H9zm3 0v2h1.36l.5-2H12zm1.11 3H12v2h.61l.5-2zM11 8H9v2h2V8zM8 8H6v2h2V8zM5 8H3.89l.5 2H5V8zm0 5a1 1 0 1 0 0 2 1 1 0 0 0 0-2zm-2 1a2 2 0 1 1 4 0 2 2 0 0 1-4 0zm9-1a1 1 0 1 0 0 2 1 1 0 0 0 0-2zm-2 1a2 2 0 1 1 4 0 2 2 0 0 1-4 0z"/>

        </svg>

        <span v-if="$store.state.cart.length > 0" class="align-items-center justify-content-center translate-middle badge rounded-pill bg-secondary">

          {{ $store.state.cart.length }}

        </span>

      </router-link>

    </li>

</ul>

</template>

<script setup>

    import { onMounted } from 'vue';

    import store from "../../store";

    onMounted(() => {

         store.commit('initialiseStore')

            }

);

</script>



                    <router-link to="/shopping" class="nav-item nav-link"> Shopping Cart </router-link>

                    <AppHeader />

…

<script setup>

    import AppHeader from './components/cart/AppHeader.vue';

    import { onMounted } from 'vue';

    import store from "./store"

    onMounted(() => {

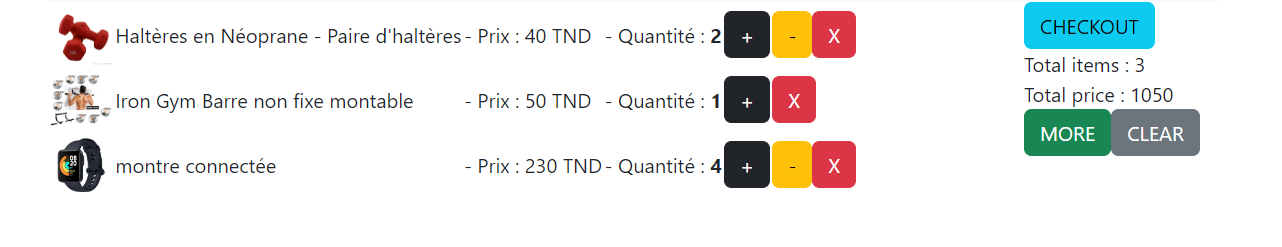
         store.commit('initialiseStore')

            }

);

</script>





# Payement Stripe

Un flux de paiement avec Stripe se divise en deux étapes :

1. Côté client, dans votre code frontend, vous collectez et tokenisez les informations de paiement du client à l'aide de Checkout ou Elements, puis envoyez le jeton résultant à votre serveur backend.

2. Côté serveur, dans votre code backend, vous utilisez le jeton dans une requête API, par ex. pour créer une charge ou un client.

## Site Stripe

Créer un compte stripe :

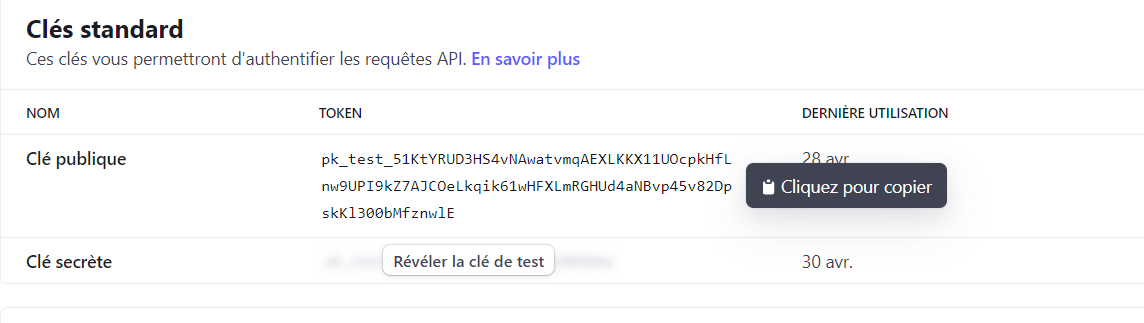
<https://stripe.com/fr-us>

Après Inscription, cliquer sur Développeurs (coin en haut à droite)



Puis cliquer sur Clés API





## Dans le projet

Ensuite, vous devez ajouter le package Stripe composer à votre projet Laravel, ce que vous pouvez faire à l'aide de la commande ci-dessous :

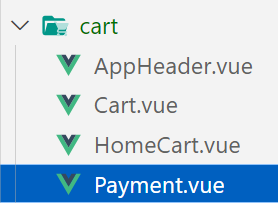
composer require stripe/stripe-php

## Vuejs

Installer le package

npm install @stripe/stripe-js

Sous le dossier cart créer le fichier Payment.vue



<template>

  <form @submit.prevent="handleSubmit">

  <div class="card" style="width: 50rem;">

   <div class="card-body">

    <h5 class="card-title">Montant à payer : {{amount}}</h5>

    <p class="card-text">

    </p>

    <div id="card-element"></div>

    <p class="card-text">

    </p>

    <button type="submit" class="btn btn-success">Procéder au payement</button>

  </div>

 </div>

 </form>

</template>

<script setup>

import { onMounted, ref } from 'vue';

import { loadStripe } from '@stripe/stripe-js';

import store from '../../store'

const stripePromise = loadStripe('pk\_test\_51KtYRUD3HS4vNAwatvmqAEXLKKX11UOcpkHfLnw9UPI9kZ7AJCOeLkqik61wHFXLmRGHUd4aNBvp45v82DpskKl300bMfznwlE');

const stripe = ref(null);

let cardElement; // Déclarer cardElement comme variable globale

let amount=ref(0);

onMounted(async () => {

  amount.value=store.state.cartTotal;

  stripe.value = await stripePromise;

  const elements = stripe.value.elements();

  cardElement = elements.create('card'); // Affecter cardElement

  cardElement.mount('#card-element');

});

const handleSubmit = async () => {

  const { token, error } = await stripe.value.createToken(cardElement);

  if (error) {

    console.error(error);

  } else {

    // Envoie le token au serveur pour traiter le paiement

    console.log(token);

  }

};

</script>

Créer la route



import Payment from './components/cart/Payment.vue';

…

 {

        path:'/payment',

        name:'Payment',

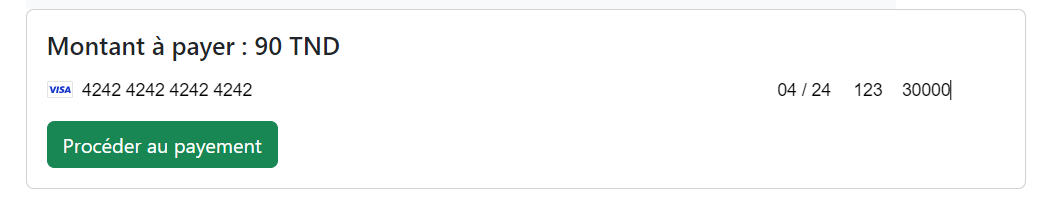
        component:Payment

    },

Faire l’appel dans le bouton checkout



<router-link :to="{name: 'Payment'}" >   <button class="btn btn-info"> CHECKOUT </button> </router-link>





## Controller Laravel

php artisan make:controller StripeController



<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use Stripe\Stripe;

use Stripe\Charge;

use Exception;

class StripeController extends Controller

{

    public function processpayment(Request $request)

    {

        $token = $request->input('token'); // Récupérer le token envoyé depuis le client

        Stripe::setApiKey('sk\_test\_51KtYRUD3HS4vNAwa7ANL32HQqRTywhV3JHWIp3BxAIHv04bWoz22aKlRs9Q1L6znSX2i5fu5i3Xkl9i2Goz7jAkC00LL0T3lTL');

        try {

            $charge = Charge::create([

                'amount' => $request->input('amount'), // Le montant à payer en cents (100 = 1 dollar)

                'currency' => 'usd', // La devise

                'source' => $token, // Le token Stripe de la carte de crédit

                'description' => 'Paiement via Stripe', // Description facultative

                'receipt\_email'=>$request->input('email') // L'adresse e-mail à laquelle envoyer un reçu de paiement.

            ]);

            // Le paiement a réussi

            return response()->json(['message' => 'Paiement réussi']);

        } catch (\Exception $e) {

            return response()->json(['error' => $e->getMessage()], 400);

        }

    }

}

## La route api

Ajouter les routes :



Route::group([

    'middleware' => 'api',

    'prefix' => 'payment'

], function ($router) {

Route::post('/processpayment', [StripeController::class, 'processpayment']);

});

## Modifier le code vueJS



<template>

  <form @submit.prevent="handleSubmit">

  <div class="card" style="width: 50rem;">

   <div class="card-body">

    <h5 class="card-title">Montant à payer : {{amount}} TND</h5>

    <br/>

    <div class="card-text">

      <div class="form-group">

       <input type="email" class="form-control" placeholder="email" v-model="email" />

     </div>

     <br/>

    </div>

    <div id="card-element"></div>

    <br/>

    <button type="submit" class="btn btn-success">Procéder au payement</button>

  </div>

 </div>

 </form>

</template>

<script setup>

import { onMounted, ref } from 'vue';

import { loadStripe } from '@stripe/stripe-js';

import store from '../../store'

import axios from "../config/axios.js";

const stripePromise = loadStripe('pk\_test\_51KtYRUD3HS4vNAwatvmqAEXLKKX11UOcpkHfLnw9UPI9kZ7AJCOeLkqik61wHFXLmRGHUd4aNBvp45v82DpskKl300bMfznwlE');

const stripe = ref(null);

let cardElement; // Déclarer cardElement comme variable globale

let amount=ref(0);

let email=ref("");

onMounted(async () => {

  amount.value=store.state.cartTotal;

  stripe.value = await stripePromise;

  const elements = stripe.value.elements();

  cardElement = elements.create('card'); // Affecter cardElement

  cardElement.mount('#card-element');

});

const handleSubmit = async () => {

  const { token, error } = await stripe.value.createToken(cardElement);

  if (error) {

    console.error(error);

  } else {

    // Envoie le token au serveur pour traiter le paiement

    const response = await axios.post('/api/payment/processpayment', {

      token: token.id,

      amount: amount.value \* 100, // Convertir le montant en cents

      email:email.value

    });

    if (response.data.message) {

      console.log(response.data.message);

      // Le paiement a réussi

      alert(response.data.message);

    } else {

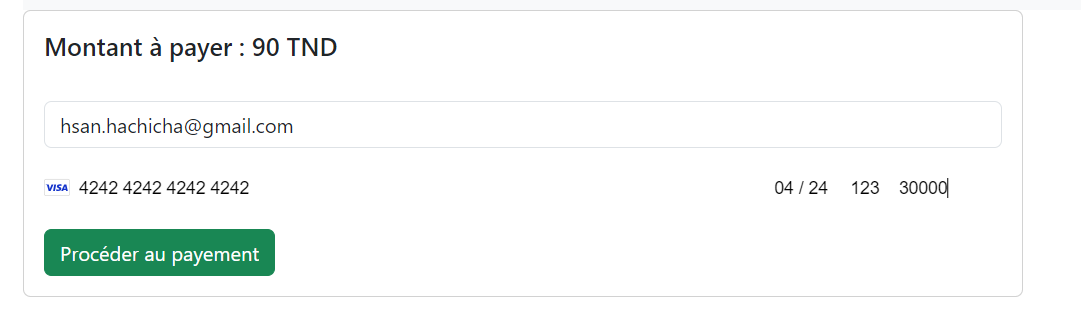
      console.error(response.data.error);

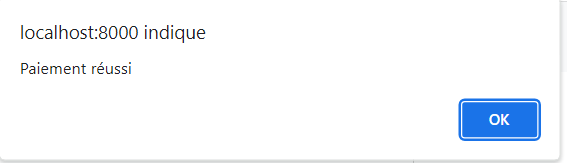
    }

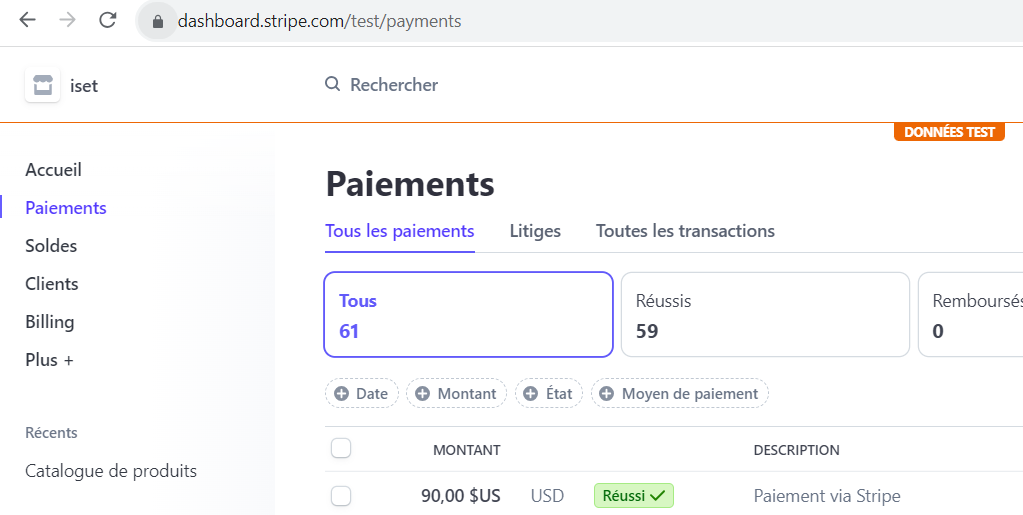
  }

};

</script>







## Vider le cart après le payement

Modifier le code



<template>

  <form @submit.prevent="handleSubmit">

  <div class="card" style="width: 50rem;">

   <div class="card-body">

    <h5 class="card-title">Montant à payer : {{amount}} TND</h5>

    <br/>

    <div class="card-text">

      <div class="form-group">

       <input type="email" class="form-control" placeholder="email" v-model="email" />

     </div>

     <br/>

    </div>

    <div id="card-element"></div>

    <br/>

    <button type="submit" class="btn btn-success">Procéder au payement</button>

  </div>

 </div>

 </form>

</template>

<script setup>

import { onMounted, ref } from 'vue';

import { loadStripe } from '@stripe/stripe-js';

import store from '../../store'

import axios from "../config/axios.js";

import { useRouter } from 'vue-router';

const router = useRouter();

const stripePromise = loadStripe('pk\_test\_51KtYRUD3HS4vNAwatvmqAEXLKKX11UOcpkHfLnw9UPI9kZ7AJCOeLkqik61wHFXLmRGHUd4aNBvp45v82DpskKl300bMfznwlE');

const stripe = ref(null);

let cardElement; // Déclarer cardElement comme variable globale

let amount=ref(0);

let email=ref("");

onMounted(async () => {

  amount.value=store.state.cartTotal;

  stripe.value = await stripePromise;

  const elements = stripe.value.elements();

  cardElement = elements.create('card'); // Affecter cardElement

  cardElement.mount('#card-element');

});

const handleSubmit = async () => {

  const { token, error } = await stripe.value.createToken(cardElement);

  if (error) {

    console.error(error);

  } else {

    // Envoie le token au serveur pour traiter le paiement

    const response = await axios.post('/api/payment/processpayment', {

      token: token.id,

      amount: amount.value \* 100, // Convertir le montant en cents

      email:email.value

    });

    if (response.data.message) {

      console.log(response.data.message);

      // Le paiement a réussi

      alert(response.data.message);

      //Vider le cart

      store.commit('clearCart')

      //Redirection

      router.push('/shopping')

    } else {

      console.error(response.data.error);

    }

  }

};

</script>

