Compte-rendu SAE 2.04 : serveur web

1. Contexte

Nous voulons afficher sur une page web les résultats de la génération d'une image de figure mathématique. Nous partons d'un site qui comprend une figure, le flocon de von Koch, et nous devons rajouter le carré de Nees.

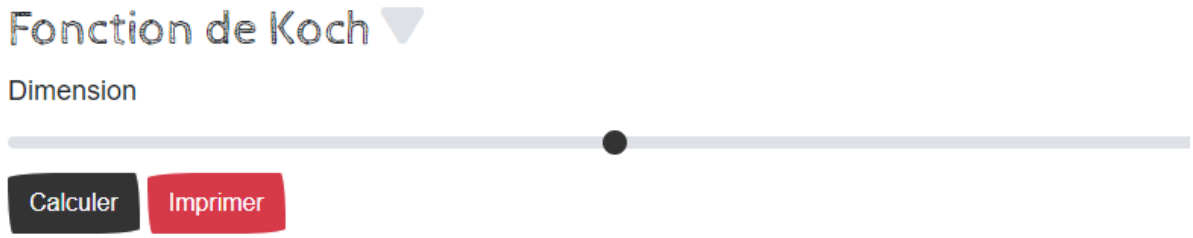
Le site est basé sur Symfony. Un formulaire permet de saisir la dimension souhaitée des segments (entier entre 0 et 6). Sur validation du formulaire, l'entrée est lue par le contrôleur PHP, et envoyée au programme. On peut afficher la figure ou l'imprimer.

Nous avons installé le site sur une machine virtuelle Debian 10.

Formulaire :

```
<form id="form_koch" method="post" action="/calculerKoch">
        <div>
            <label for="dimension" class="form-label">Dimension</label>
                <input type="range" class="form-range" min="0" max="6"
step="1" id="dimension" name="dimension">
        </div>
        <div>
            <button type="submit" class="btn btn-primary" name="calculer"
value="1">Calculer</button>
            <button type="submit" class="btn btn-danger" name="imprimer"
value="1">Imprimer</button>
        </div>
    </form>
```

Le champ de type « input » permet de choisir une valeur avec un curseur :



Compte-rendu SAE 2.04 : serveur web

Contrôleur PHP :

```php
/**
     * @Route("/calculerKoch", name="calculerKoch")
     */
    public function calculerKoch(Request $request): Response
    {
        // Récupère les paramètres issus du formulaire (on indique le champ
name)
        $dimension = $request -> request -> get("dimension") ;
        // Pour les boutons : si appui contenu champ value sinon NULL
        $calculer  = $request -> request -> get("calculer");
        $imprimer  = $request -> request -> get("imprimer");

        // Oui : Appelle le script Python koch.py qui se trouve dans le
répertoire /public
        $process = new Process(['python3','koch.py', $dimension]);
        $process -> run();
        // Récupère la valeur de retour renvoyé par le script python
        $fichier=$process->getOutput();

        // Retourne un message si l'exécution c'est mal passée
        if (!$process->isSuccessful())
            return new Response ("Erreur lors de l'exécution du script Python
:<br>".$process->getErrorOutput());

        // A-t-on appuyé sur Calculer ?
        if ($calculer!=NULL)
            return $this->render('artmath/koch.html.twig', [
                'fichier' => $fichier,
            ]);
        else {
            // On a appuyé sur Imprimer
            return $this->render('artmath/imprimer.html.twig', [
                'fichier' => $fichier,
            ]);
        }
    }
```

La fonction `request` permet de récupérer le contenu de la variable `$_POST` venant d'un formulaire. Le programme est appelé par l'objet `Process()` qui permet d'exécuter une commande dans l'environnement du système. Cet objet possède des propriétés qui permettent de remonter ses erreurs et d'utiliser sa sortie.

Compte-rendu SAE 2.04 : serveur web

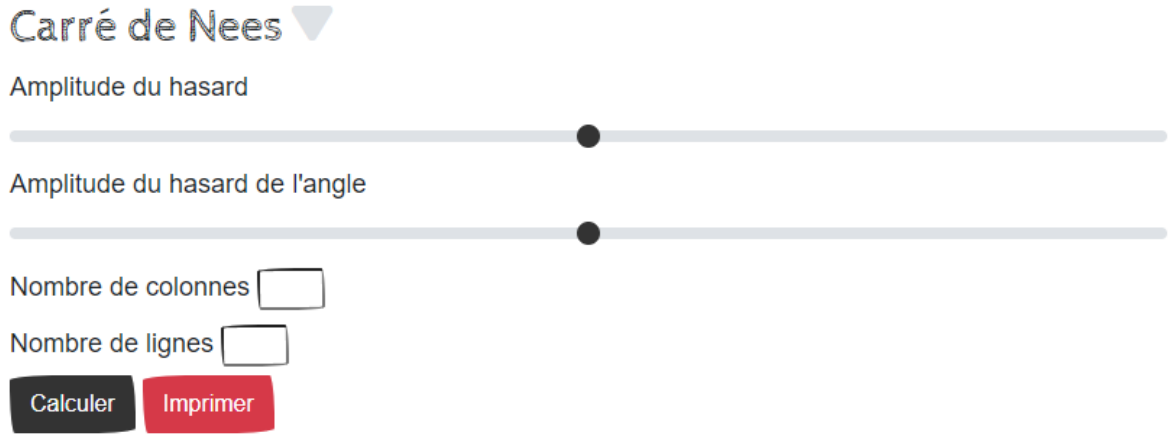
2. Carré de Nees

Pour générer le carré de Nees, nous disposons déjà du programme Python. Cependant, à la différence du flocon, il prend 4 arguments. Nous devons donc les récupérer à l'aide d'un formulaire. Voici le formulaire créé à cet effet.

```
<div class="nees example-wrapper">
    <h2>Carré de Nees <i class="fa-solid fa-play"></i></h2>
    <!-- C'est grâce à l'attribut name que l'on peut récupérer la valeur
associée dans symfony -->
    <form id="form_nees" method="post" action="/calculerNees">
        <div>
            <label for="amplitude" class="form-label">Amplitude du
hasard</label>
                <input type="range" class="form-range" min="0" max="1"
step="0.05" id="amplitude" name="amplitude">
        </div>
        <div>
            <label for="angle" class="form-label">Amplitude du hasard de
l'angle</label>
                <input type="range" class="form-range" min="0" max="1"
step="0.05" id="angle" name="angle">
        </div>
        <div>
            <label for="colonnes" class="form-label">Nombre de
colonnes</label>
                <input type="number" class="form-text" min="0" max="50"
step="1" id="colonnes" name="colonnes">
        </div>
        <div>
            <label for="lignes" class="form-label">Nombre de lignes</label>
                <input type="number" class="form-text" min="0" max="50"
step="1" id="lignes" name="lignes">
        </div>
        <div>
            <button type="submit" class="btn btn-primary" name="calculer"
value="1">Calculer</button>
            <button type="submit" class="btn btn-danger" name="imprimer"
value="1">Imprimer</button>
        </div>
    </form>
</div>
```

Compte-rendu SAE 2.04 : serveur web

On prend en compte deux valeurs qui doivent être comprises entre 0 et 1, un nombre de colonnes et un nombre de lignes. Comme ces valeurs n'ont pas besoin d'être contraintes, nous utilisons des champs de texte classiques. Cependant, ils sont de type « number » pour que le navigateur puisse effectuer des contrôles de saisie. Nous utilisons également l'attribut `required` pour afficher un message en cas de case vide.



Dans le contrôleur, nous avons ajouté une route vers le calcul de cette figure.

```
/**
     * @Route("/calculerNees", name="calculerNees")
     */
    public function calculerNees(Request $request): Response
    {
        // Récupère les paramètres issus du formulaire (on indique le champ
name)
        $amplitude = $request -> request -> get("amplitude");
        $angle = $request -> request -> get("angle");
        $colonnes = $request -> request -> get("colonnes");
        $lignes = $request -> request -> get("lignes");
        // Pour les boutons : si appui contenu champ value sinon NULL
        $calculer  = $request -> request -> get("calculer");
        $imprimer  = $request -> request -> get("imprimer");

        if(!isset($colonnes) OR !isset($lignes)) {
            $erreur = "Les champs doivent être remplis";
            return $this->render('artmath/index.html.twig', [
                'erreur' => $erreur
            ]);
        } // Vérification de champs à faire fonctionner

        // Oui : Appelle le script Python koch.py qui se trouve dans le
répertoire /public
        $process = new Process(['python3','nees_carre.py', $amplitude, $angle,
$colonnes, $lignes]);
        $process -> run();
```

```
        // Récupère la valeur de retour renvoyé par le script python
        $fichier='reponse.png';

        // Retourne un message si l'exécution c'est mal passée
        // if (!$process->isSuccessful())
        //     return new Response ("Erreur lors de l'exécution du script
Python :<br>".$process->getErrorOutput());

        // A-t-on appuyé sur Calculer ?
        if ($calculer!=NULL)
            return $this->render('artmath/nees.html.twig', [
                'fichier' => $fichier,
            ]);
        else {
            // On a appuyé sur Imprimer
            return $this->render('artmath/imprimer.html.twig', [
                'fichier' => $fichier,
            ]);
        }
    }
```

Nous traitons les informations de manière similaire à la première figure. La différence étant leur pluralité, ce qui n'empêche pas l'exécution : quand on a plusieurs paramètres, on les place à la suite dans l'ordre dans la commande d'exécution.

Compte-rendu SAE 2.04 : serveur web

Voici le résultat du flocon de von Koch :

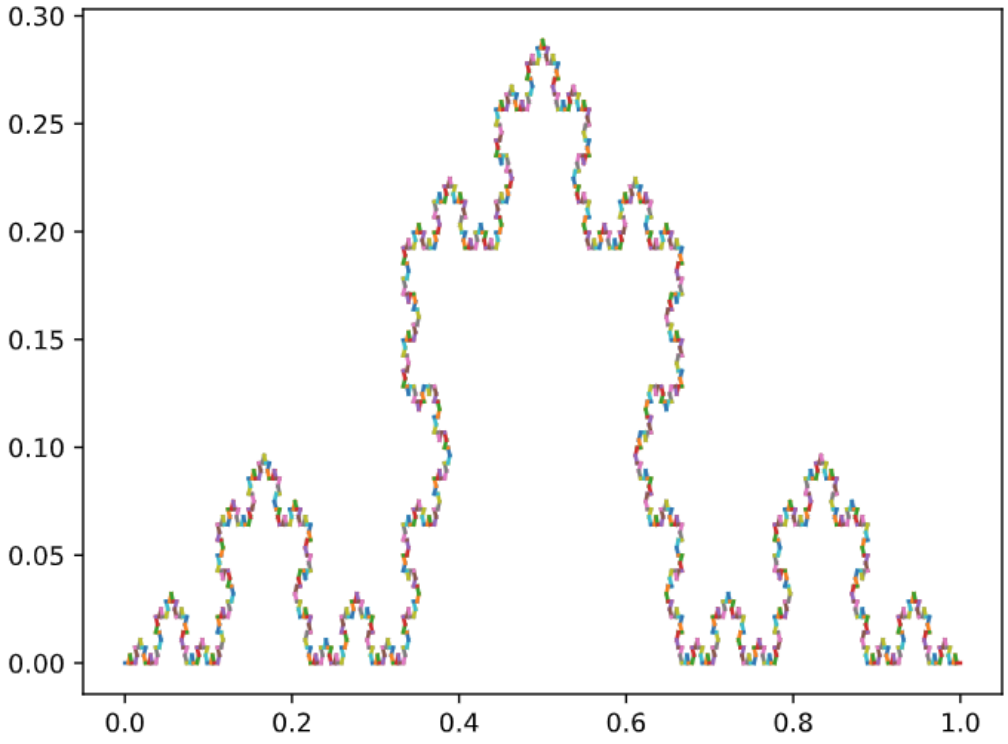
Art mathématique

Fonction de Koch :

Dimension

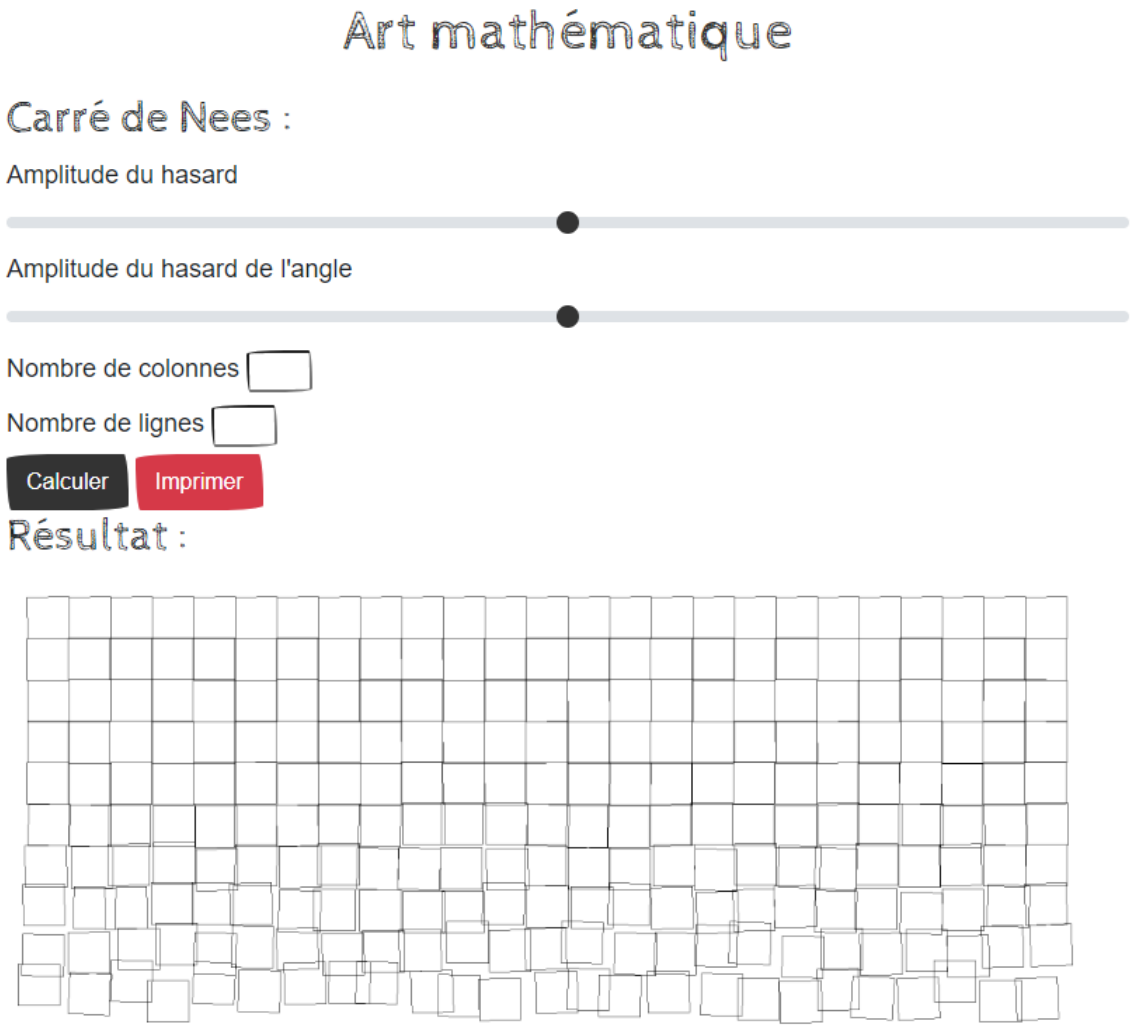
Calculer Imprimer

Résultat :



Compte-rendu SAE 2.04 : serveur web

Carré de Nees :



Le formulaire de chaque figure reste disponible pour en regénérer une avec de nouvelles valeurs.

Compte-rendu SAE 2.04 : serveur web

3. Améliorations

Nous avons séparé le code, comprenant initialement formulaire et réponse sur une page, en plusieurs :

- Page d'accueil avec formulaire
- Page de résultat Koch
- Page de résultat Nees

Avec un titre h1 cliquable pour revenir à l'accueil.

Nous avons également, pour des raisons de commodité visuelle, réduit les menus non-utilisés. Un clic les déplie et fait pivoter la flèche grâce à du jQuery :

```
$('.koch h2').click(function(){
    $('#form_koch').slideToggle(500);
    $('.koch .fa-play').toggleClass('open');
});
$('.nees h2').click(function(){
    $('#form_nees').slideToggle(500);
    $('.nees .fa-play').toggleClass('open');
});
```

Même si ce n'est pas essentiel pour un tel projet, la gestion des formulaires peut être améliorée. Actuellement, codé en dur dans le HTML, cela nous permet de récupérer la valeur et de vérifier l'entrée utilisateur à l'aide du navigateur. Mais la modification du code source à l'aide du navigateur anéantirait ces vérifications et ferait planter le programme qui recevrait des valeurs erronées :

Erreur lors de l'exécution du script Python :
:241: RuntimeWarning: Your system is avx2 capable but pygame was not built with
nbcolonnes=int(sys.argv[3]) ValueError: invalid literal for int() with base 10: ''

La méthode la plus sûre reste une vérification côté serveur. Pour ce faire, Symfony comporte une fonction de génération dynamique de formulaire, qui se charge à la fois de :

- Supprimer les injections SQL
- Vérifier si l'entrée est du bon type (chiffre, texte, e-mail…)
- Vérifier si le champ n'est pas vide

Pour rendre cela possible, nous devons d'abord créer les objets correspondants.

Par exemple pour le carré de Nees : il possédera 4 propriétés, celles dont on aura besoin pour générer la figure.

```
PS C:\Users\Paul\Documents\Symfony\SAE-2.04> symfony console make:entity

Class name of the entity to create or update (e.g. VictoriousPizza):
> Nees

created: src/Entity/Nees.php
created: src/Repository/NeesRepository.php

Entity generated! Now let's add some fields!
You can always add more fields later manually or by re-running this command.

New property name (press <return> to stop adding fields):
> amplitude

Field type (enter ? to see all types) [string]:
> float

Can this field be null in the database (nullable) (yes/no) [no]:
>

updated: src/Entity/Nees.php

Add another property? Enter the property name (or press <return> to stop
adding fields):
> angle

Field type (enter ? to see all types) [string]:
> float

Can this field be null in the database (nullable) (yes/no) [no]:
>

updated: src/Entity/Nees.php

Add another property? Enter the property name (or press <return> to stop
adding fields):
> colonnes

Field type (enter ? to see all types) [string]:
> integer

Can this field be null in the database (nullable) (yes/no) [no]:
>
```

Compte-rendu SAE 2.04 : serveur web

```
updated: src/Entity/Nees.php

Add another property? Enter the property name (or press <return> to stop
adding fields):
> lignes

Field type (enter ? to see all types) [string]:
> integer

Can this field be null in the database (nullable) (yes/no) [no]:
>

updated: src/Entity/Nees.php

Add another property? Enter the property name (or press <return> to stop
adding fields):
>



 Success!


Next: When you're ready, create a migration with php bin/console
make:migration
```

Compte-rendu SAE 2.04 : serveur web

Exemple avec le carré de Nees :

```
class NeesType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options):
void
    {
        $builder
            ->add('amplitude', RangeType::class, [
                'attr' => [
                    'class' => 'form-range',
                    'min' => 0,
                    'max' => 1,
                    'step'=> 0.05
                ],
            ])
            ->add('angle', RangeType::class, [
                'attr' => [
                    'class' => 'form-range',
                    'min' => 0,
                    'max' => 1,
                    'step'=> 0.05
                ],
            ])
            ->add('colonnes', NumberType::class, ['attr' => [
                'class' => 'form-text',
                'placeholder' => '10'
            ],
            'constraints' => [
                new NotBlank([
                    "message" => "Veuillez entrer un nombre"
                ]),
            ]])
            ->add('lignes', NumberType::class, ['attr' => [
                'class' => 'form-text',
                'placeholder' => '10'
            ],
            'constraints' => [
                new NotBlank([
                    "message" => "Veuillez entrer un nombre"
                ]),
            ]])
            ->add('calculer', SubmitType::class, [
                'attr' => [
                    'class' => 'btn btn-primary'
                ]
            ])
            ->add('imprimer', SubmitType::class, [
                'attr' => [
```

```
                    'class' => 'btn btn-danger'
                ]
            ])
        ;
    }
    public function configureOptions(OptionsResolver $resolver): void
    {
        $resolver->setDefaults([
            'data_class' => Nees::class,
        ]);
    }
}
```

Ceci est la classe qui sert de modèle de formulaire pour la figure de Nees, stocké dans `src\Form\Type`. On génère les champs de formulaire de type `range` et `number` avec les classes existantes à cet effet dans Symfony. Symfony permet de modifier leurs propriétés par défaut comme nous le faisons en HTML. Cela nous permet d'ajouter les classes CSS nécessaires au style, et les propriétés de pas et d'extrema sur les champs qui le nécessitent.

L'objet `NotBlank` permet de vérifier côté serveur que le champ est bien rempli : on en a besoin pour les champs de type `number`. L'option sera aussi très utile sur un formulaire sur un champ d'adresse électronique, un nom d'utilisateur ou un mot de passe.

La fonction `setDefaults` permet de spécifier quelle classe (ici Nees, une des entités créées précédemment) utilisera par défaut le formulaire si aucune n'est spécifiée dans le contrôleur (ce que nous avons fait quoi qu'il en soit).

On génère ensuite le formulaire de cette manière :

1. On crée l'objet, auquel on peut éventuellement donner une valeur par défaut,

```
$koch = new VonKoch();
$koch->setDimension(3);
```

2. On génère le formulaire d'après la classe-modèle, et des paramètres éventuels,

```
$formKoch = $this->createForm(VonKochType::class, $koch, [
            'attr' => ['id' => 'form_koch'],
            'action' => $this->generateUrl('calculerKoch')
        ]);
```

3. On instancie le formulaire dans la fonction de génération de la page du contrôleur,

```
return $this->renderForm('artmath/index.html.twig', [
            'formKoch' => $formKoch,
            'formNees' => $formNees
        ]);
```

4. On instancie le formulaire dans le code Twig de la page web.

```
{% block body %}
<div class="koch example-wrapper">
    <h2>Fonction de Koch <i class="fa-solid fa-play"></i></h2>
    <!-- C'est grâce à l'attribut name que l'on peut récupérer la valeur
associée dans symfony -->
    {{form(formKoch)}}
</div>
<div class="nees example-wrapper">
    <h2>Carré de Nees <i class="fa-solid fa-play"></i></h2>
    <!-- C'est grâce à l'attribut name que l'on peut récupérer la valeur
associée dans symfony -->
    {{form(formNees)}}
</div>
{% endblock %}
```

Compte-rendu SAE 2.04 : serveur web

Voici la fonction de la page d'accueil pour les deux formulaires.

```php
    public function index(Request $request): Response
    {
        $koch = new VonKoch();
        $koch->setDimension(3);

        $formKoch = $this->createForm(VonKochType::class, $koch, [
            'attr' => ['id' => 'form_koch'],
            'action' => $this->generateUrl('calculerKoch')
        ]);

        $nees = new Nees();
        $nees->setAmplitude(0.5);
        $nees->setAngle(0.5);

        $formNees = $this->createForm(NeesType::class, $nees, [
            'attr' => ['id' => 'form_nees'],
            'action' => $this->generateUrl('calculerNees')
        ]);

        return $this->renderForm('artmath/index.html.twig', [
            'formKoch' => $formKoch,
            'formNees' => $formNees
        ]);
    }
```

Quelques ajustements ont été nécessaires dans les fonctions de calcul. On n'a plus besoin de récupérer les valeurs des deux boutons, un appui sur Calculer renvoyant automatiquement vers le calcul. Symfony renvoie une erreur si l'on tente de lire la valeur du bouton qui n'a pas été utilisé. On ne lit donc la valeur du bouton Imprimer que plus tard au moment de décider sur quelle page on redirige.

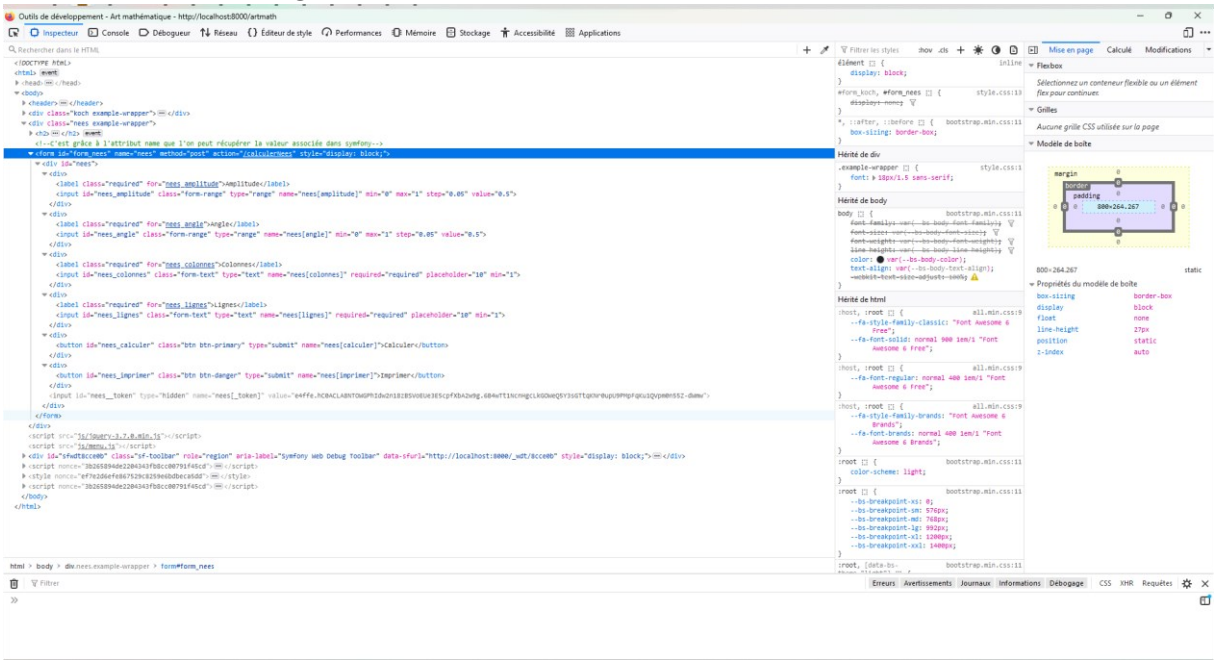
```php
public function calculerNees(Request $request): Response
    {
        // Récupère les paramètres issus du formulaire (on indique le champ
name)
        $amplitude = $request -> request -> all()['nees']['amplitude'];
        $angle = $request -> request -> all()['nees']['angle'];
        $colonnes = $request -> request -> all()['nees']['colonnes'];
        $lignes = $request -> request -> all()['nees']['lignes'];

        if(!isset($colonnes) OR !isset($lignes)) {
            $erreur = "Les champs doivent être remplis";
            return $this->render('artmath/index.html.twig', [
                'erreur' => $erreur
            ]);
        } // Vérification de champs à faire fonctionner
```

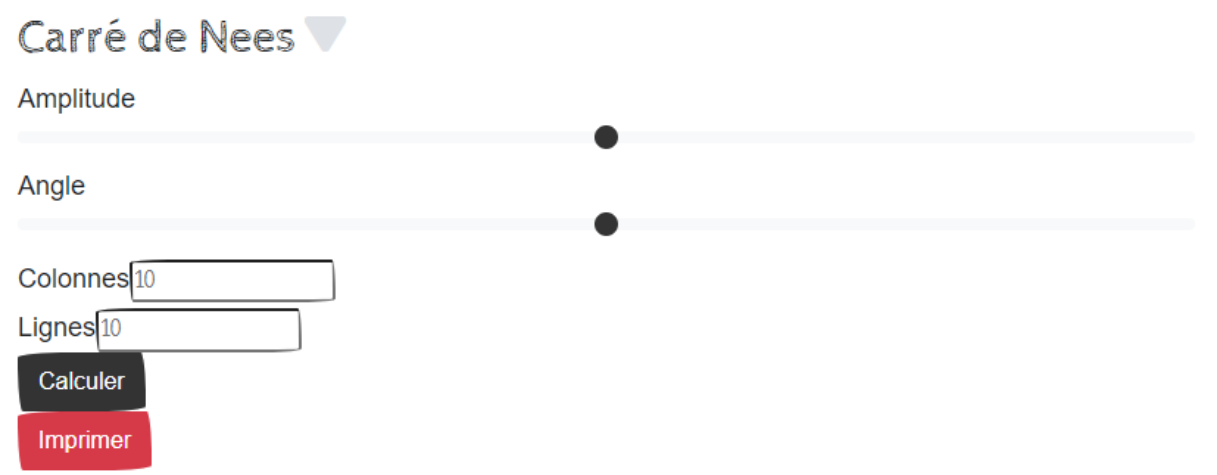
Compte-rendu SAE 2.04 : serveur web

```
        // Oui : Appelle le script Python koch.py qui se trouve dans le
répertoire /public
        $process = new Process(['python3','nees_carre.py', $amplitude, $angle,
$colonnes, $lignes]);
        $process -> run();
        // Récupère la valeur de retour renvoyé par le script python
        $fichier='reponse.png';

        // Retourne un message si l'exécution c'est mal passée
        if (!$process->isSuccessful())
            return new Response ("Erreur lors de l'exécution du script Python
:<br>".$process->getErrorOutput());

        // A-t-on appuyé sur Calculer ?
        if (!isset($request -> request -> all()['nees']['imprimer']))
            return $this->render('artmath/nees.html.twig', [
                'fichier' => $fichier,
            ]);
        else {
            // On a appuyé sur Imprimer
            return $this->render('artmath/imprimer.html.twig', [
                'fichier' => $fichier,
            ]);
        }
    }
```

Compte-rendu SAE 2.04 : serveur web

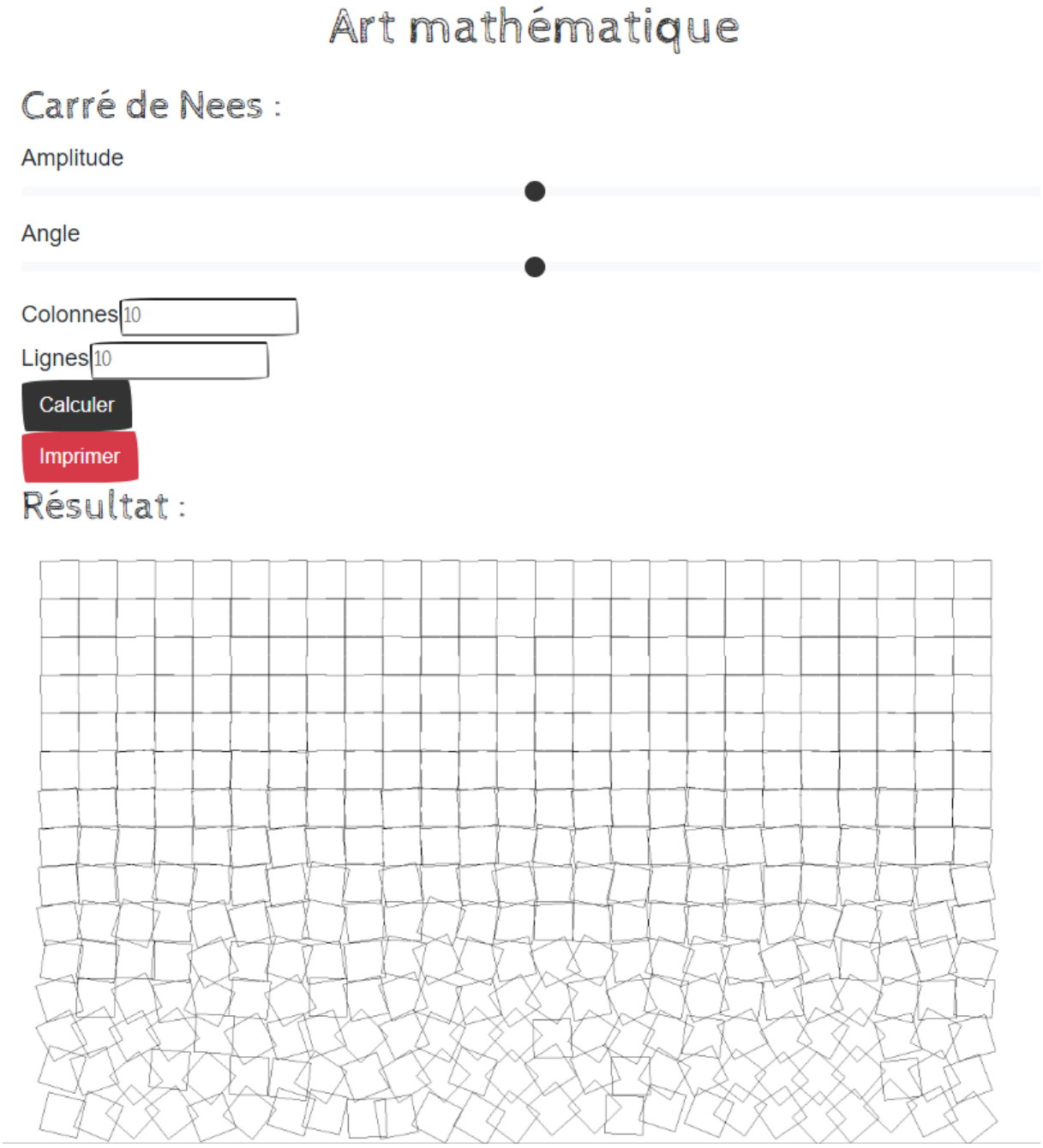
Voici le formulaire généré par Symfony :





Compte-rendu SAE 2.04 : serveur web

Une petite démonstration :

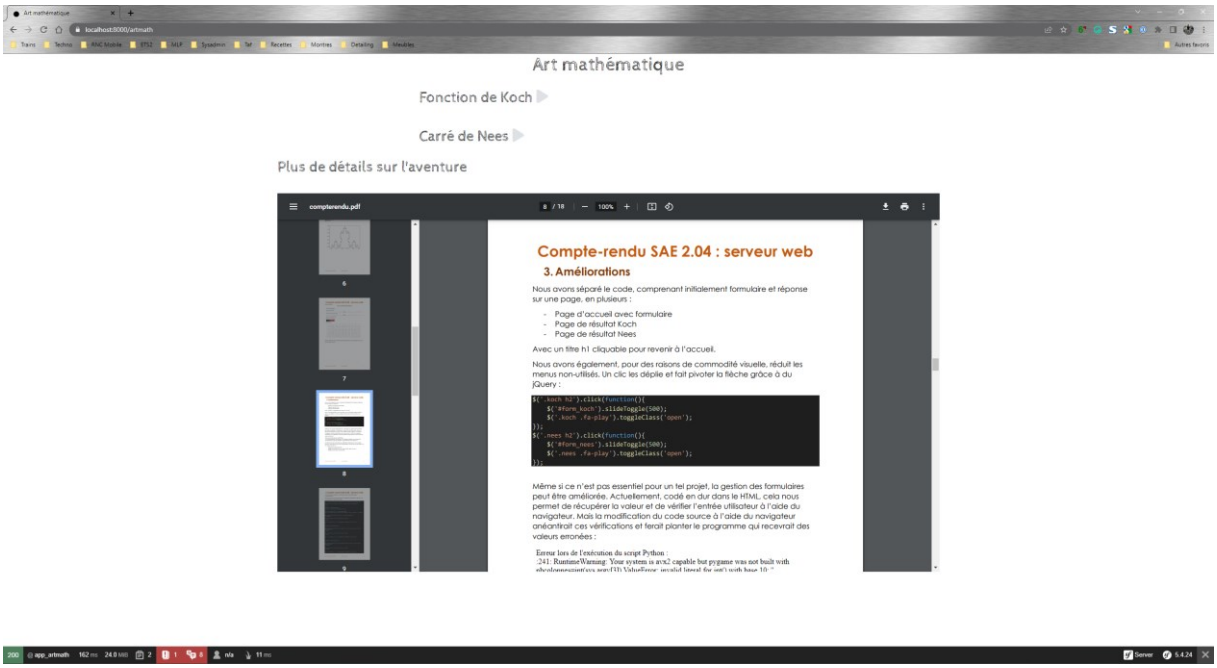


Compte-rendu SAE 2.04 : serveur web

Nous avons intégré le compte-rendu au site, il permettra d'en comprendre le but et le fonctionnement plus en détail. La tâche est rendue simple avec les navigateurs actuels : grâce à la balise `embed`, on peut intégrer facilement un document PDF. Le navigateur se chargera de générer le lecteur contenant le document.

```
<embed class="compterendu" src="compterendu.pdf" type="application/pdf">
```

Comme tous les autres fichiers ajoutés par l'utilisateur, il est à placer dans le dossier `public`. Voici le résultat :



Le serveur bénéficie d'un niveau de sécurité supplémentaire : nous avons remplacé l'authentification SSH par mot de passe par une clé SSH. Cette méthode est quasiment inviolable avec les chiffrements les plus récents et bien plus dûre qu'un mot de passe.

Pour ce faire, on décommente et modifie cette ligne du fichier `/etc/ssh/sshd_config` :

```
PasswordAuthentication no
```

Bien que l'authentification par mot de passe reste possible simultanément, il vaut mieux la désactiver. On n'en a plus besoin avec une clé, et cela met fin aux attaques par dictionnaire ou force brute. Le support des clés, quant à lui, est activé par défaut, donc nous n'avons rien à modifier de plus. Il faut générer les clés et ajouter la clé publique au fichier `/home/paul/.ssh/authorized_keys`, à créer :

```
ssh-keygen -t ed25519
cd ~/.ssh
cat id_ed25519.pub > authorized_keys
```

La clé privée appelée par défaut `id_ed25519` et stockée dans le même dossier est à utiliser dans un client SSH pour se connecter à la machine. Le chiffrement Ed25519 utilise les courbes elliptiques pour générer les clés, et est aujourd'hui l'un des plus prisés. Alors qu'il fournit un niveau de sécurité comparable à une clé RSA 4096 bits, il demande bien moins de puissance de calcul pour le chiffrement et le déchiffrement.