Valérian Lambert
Julie Leroy
Arsene Jerome
Thomas Madrange-Maire
Clément Berard

Compte rendu – Développement de la partie web de l'application

sae 3e année 2025-2026

1. Introduction

Le but de ce rapport c'est d'expliquer les étapes du développement de la composante Web de l'application conçue pour piloter une infrastructure de calcul distribué.

Le principal but du projet est l'exploitation d'un cluster de Raspberry Pi, dont l'objectif principal est de permettre l'exécution et la gestion de programmes de calcul parallèle.

Afin de rendre cette application de calcul accessible, le développement d'une application web est nécessaire. Elle va permettre de faire le lien entre l'utilisateur final quelque soit son niveau et l'infrastructure technique. Cette interface joue un rôle de point de contrôle, centralisant plusieurs fonctionnalités essentielles suivantes :

- **Gestion des accès :** Se connecter et s'inscrire sur la plateforme,
- **Lancement de tâche de calcul :** Lancer un programme de calcul sur le cluster de Raspberry Pi.
- **Restitution des données calculées :** Consulter les résultats des calculs effectués affichés directement sur l'application web.

2. Démarche de développement de la partie web

2.1 Conception visuelle et structuration des données

Le développement de la partie web a été structuré en suivant une approche itérative, allant de la conception visuelle à l'implémentation technique. Cette démarche s’est déroulée en plusieurs grandes étapes.

La première étape a été la mise en place de l’identité visuelle et l’architecture de l’application web. Pour ce faire nous avons créé une maquette web ainsi qu’un schéma de base de données.Pour la maquette web nous avons cherché un logo, la palette et nous avons placé les éléments comme on le souhaitait sur nos pages sur Figma.

Image de la palette sélectionné

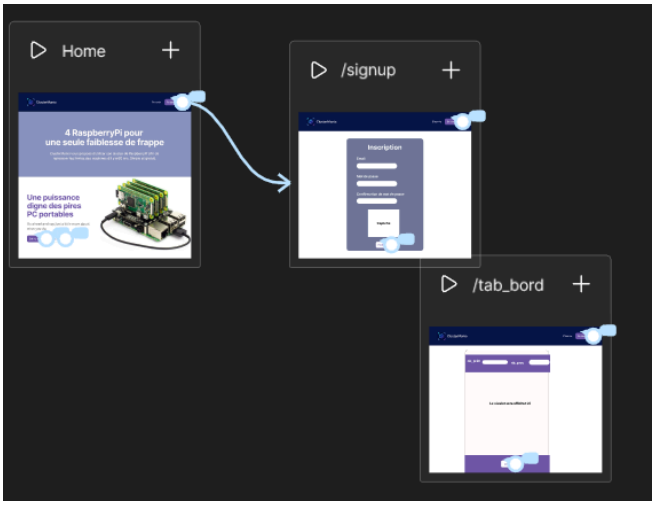


Image de la maquette débuté en partie sur Figma

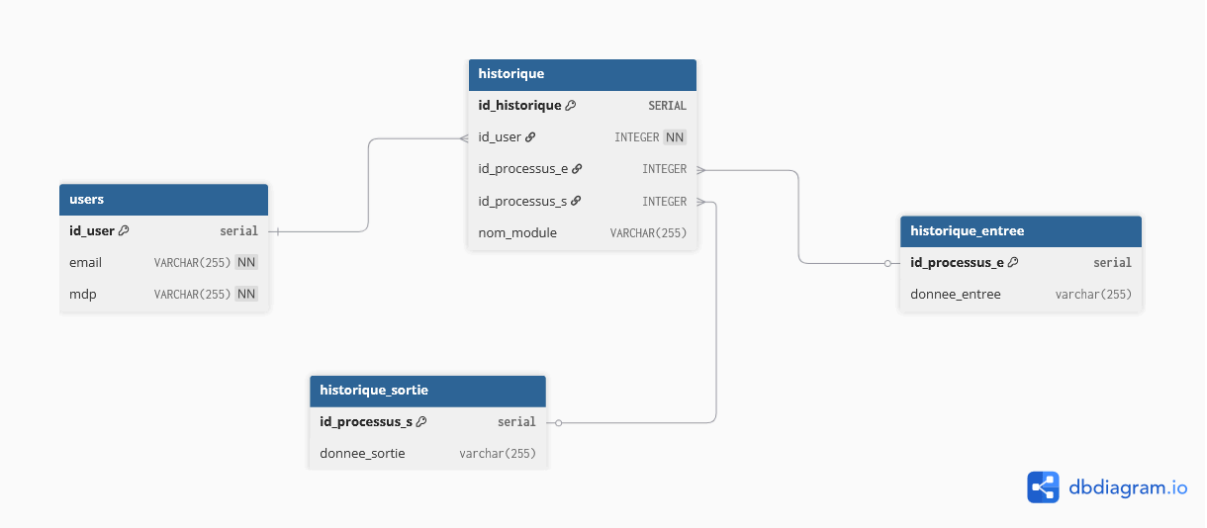


Schéma de la base de données

Cela nous a permis d'avoir un premier aperçu de ce que nous attendions pour le site de manière visuelle et sur le plan stockage de données.

2.2 Architecture de l'applications

Ensuite à partir de la maquette Figma nous avons créé une maquette en html/css.
Et c'est à partir de cette maquette là que nous avons créé l'application Web.

Sur le plan technique, la partie web repose sur :

- **PHP** pour la logique serveur et la gestion des requêtes,
- **HTML, CSS et JavaScript** pour la construction de l'interface utilisateur et les interactions côté client.
- **MySQL** pour la gestion de base de données

L'application web est composés de 7 pages (dans sa version final) :

- **Accueil** : page sur lequel l'utilisateur arrive lorsque qu'il entre sur l'application web, elle contient quelque phrase de présentation
- **Login** : Gestion de la connexion à la plateforme web
- **Logout :** Page indiquant la fin de la session et que l'utilisateur a été déconnecté après qu'il est cliqué sur le bouton déconnexion
- **Signup :** Gestion de la création de compte
- **Tab_bord** : Interface principale pour lancer et voir les résultats de calculs.
- **Profil** : Page de l'utilisateur contenant son nom et un bouton de déconnexion.
- **Redirection** : Page permettant de rediriger en cas de mauvaise utilisation de la plateforme

Le header, le footer et la barre de navigation sont ajoutés à chaque page en include pour éviter la répétition de code.

2.3 Ajout de fonctionnalités

Une fois la structure générale du site mise en place, le développement s'est concentré sur l'implémentation des fonctionnalités de la plateforme. Cette phase a permis de faire passer un niveau au-dessus pour rendre le site dynamique.

L'ajout de fonctionnalité s'est déroulé en plusieurs axes :

2.3.1 Gestion des utilisateurs et sécurité

La première partie des fonctionnalités consiste à gérer tout ce qui concerne les connexions et les accès aux ressources.

- **Système d'authentification** :Nous avons développé des formulaires d'inscription et de connexion utilisant la méthode HTTP POST. Ce choix est crucial pour la sécurité : contrairement à la méthode GET, POST transmet les données (identifiants, mots de passe) de manière invisible dans le corps de la requête, évitant ainsi l'affichage d'informations sensibles dans l'historique ou l'URL du navigateur.

  Le formulaire d'inscription ajoute des utilisateurs tandis que celui de connexion consulte la base de données pour vérifier si l'utilisateur existe et lui donner accès au tableau de bord.

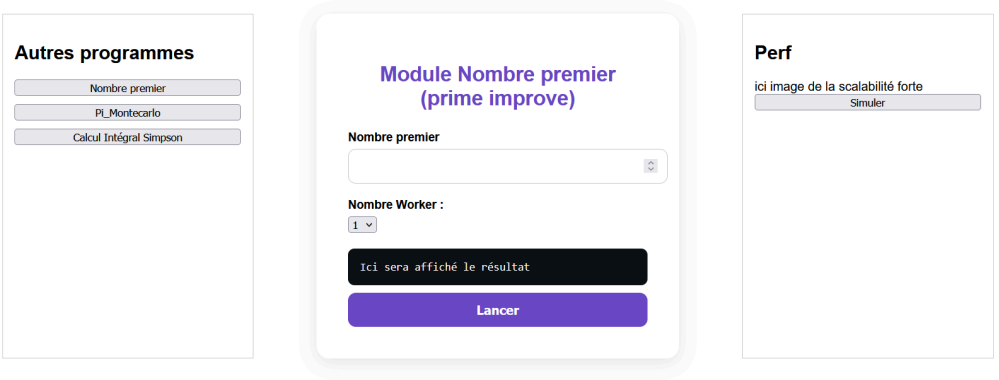
  Pour garantir la confidentialité de ces informations, nous avons choisi d'utiliser la fonction password_hash() de PHP plutôt que l'ancien standard MD5. Nous avons fait ce choix en raison du fait que MD5 est assez vulnérable aux attaques par brute de force et que contrairement à password_hash() il n'intègre pas d'algorithme de hachage robuste (BCrypt) qui génère automatiquement un "sel" unique pour chaque mot de passe. C'est-à-dire que même si deux utilisateurs ont le même mot de passe leurs empreintes stockées seront différentes.

- **Protection contre les bots** : Utilisation d'un captcha sous forme d'image de sorte à prévenir la création automatique de compte par des bots dans le formulaire d'inscription. L'image permet de faire en sorte que le robot ne puisse pas copier coller ce contient l'image pour répondre, cela bloque les bots simples qui ne sont pas capable d'interpréter les images.

- **Gestion des sessions** : Utilisation des variables de sessions PHP pour restreindre l'accès au tableau de bord aux seuls utilisateurs connectés.

2.3.2 Tableau de bord

Le tableau de bord contient le cœur du projet, c'est-à-dire la partie lancement de scripts et affichage des résultats. Son interface a été pensé en trois partie de sorte à faciliter l'utilisation pour l'utilisateur:



- **Zone de Sélection (Bloc de droit)** : Cette partie permet de choisir le programme de calcul à exécuter. Pour fluidifier l'expérience, nous avons utilisé un script JavaScript qui bascule l'affichage du formulaire central sans recharger la page. De plus, un mécanisme de persistance permet de conserver le dernier formulaire utilisé visible après l'exécution d'un calcul, évitant ainsi à l'utilisateur de perdre sa sélection.

- **Zone de paramétrage (Bloc central) :** C'est ici que l'utilisateur saisit les variables nécessaires aux scripts de calcul distribué. Ce formulaire dynamique fait le lien direct entre les entrées utilisateur et les arguments qui seront transmis au cluster de Raspberry Pi.

- **Zone de vision des performances (Bloc de gauche) :** À gauche pour mesurer les performances et qui est censée afficher une courbe de scalabilité forte ou faible avec un certain nombre de lancer .

Durant le processus de développement, le code a dû être vérifié plusieurs fois pour être sûr qu'il soit de bonne qualité. Nous l'avons notamment fait lorsque nous avons commencé à ajouter des nouveaux modules qui nécessitait un formulaire avec des entrées différentes à l'exception des workers. De ce fait plutôt que d'avoir des répétition de code, nous avons implémenté deux éléments afin d'avoir une génération dynamique:

- **Fichier de paramétrage :** Nous avons centralisé les caractéristiques de chaque programme (information sur les input et leur label, paramètres requis, nom du script associé et autre) dans un fichier de configuration.

```php
$modules = [
    'prime_improve' => [
        'id'     => 'div_prog1',
        'titre'  => 'Module Nombre premier (prime improve)',
        'script' => 'prime_improve.py',
        'min'    => 1,
        'max'    => 5,
        'inputs' => [
            ['name' => 'nb_prim', 'label' => 'Nombre premier', 'type' => 'number']
        ],
        'langage' => "python"
    ],
```

- **Boucle de génération :** Le code PHP parcourt ce fichier et génère automatiquement le formulaire correspondant.

Cette approche permet d'ajouter un nouveau module de calcul en quelques secondes simplement en modifiant le fichier de paramétrage, sans avoir à toucher au code HTML de la page.

2.3.3 Les modules de calculs ajoutés

Les modules de calculs ajoutés sont le cœur du site, et sont aussi le déclencheur de l'exécution des programmes de calcul. Tous les paramètres récoltés par le site auprès de l'utilisateur sont transmis à un script shell, qui se charge de générer les commandes d'exécution des programmes.
L'utilisateur linux d'apache, www-data, ne doit avoir aucun droit sur les fichiers internes au RPi si l'on veut conserver une sécurité accrue face aux tentatives de piratage par le site web. Le tableau de bord PHP passe donc par un autre utilisateur, mpiuser, pour pouvoir conserver les restrictions de www-data tout en faisant en sorte que le script shell exécuté sur le RPi ait accès aux programmes.

Module Prime

Le premier modèle développé permet de lancer un calcul distribué visant à déterminer des nombres premiers.

Depuis l'interface web, l'utilisateur peut :

- Saisir la valeur numérique à analyser,
- Définir le nombre de workers (nœuds du cluster Raspberry) à utiliser pour le calcul.

Une fois transmis au programme Python, le calcul est réparti sur plusieurs Raspberry Pi (au minimum un), ce qui permet d'exploiter le parallélisme.

Les résultats affichés sur l'interface web correspondent directement aux sorties générées par les instructions print du programme Python.

Module Pi MonteCarlo

Un second module a été ajouté afin de calculer une approximation du nombre Pi à l'aide d'un programme de calcul distribué.

L'utilisateur peut paramétrer le calcul en renseignant :

- Le nombre de points utilisés pour l'approximation,
- Le nombre de workers participant au calcul.

Une fois le calcul lancé depuis l'interface web, un script Python est exécuté et réparti sur plusieurs Raspberry Pi.

Module de calcul de l'intégral de Simpson pour la loi normale

Le troisième module développé concerne le calcul de l'intégrale de Simpson appliquée à la loi normale.

Depuis l'interface web, l'utilisateur peut renseigner :

- Les paramètres habituels de la fonction (bornes, nb de segment,μ,σ),
- Le nombre de workers utilisés pour le calcul distribué.

Le backend PHP déclenche ensuite l'exécution du script java correspondant. Le calcul est réparti sur plusieurs Raspberry Pi.

Comme pour les autres modules, les résultats affichés sur l'interface web correspondent aux informations produites par les print du programme.

3. Difficultés rencontrées

Le développement de ce projet a été marqué par plusieurs défis qui ont nécessité une analyse approfondie des interactions entre le serveur web et le système d'exploitation du cluster.

Durant le développement plusieurs difficultés ont pu être rencontré

3.1. Problèmes de sécurité

La minimisation du risque de l'exécution de scripts malveillants directement sur notre cluster de RPi a été l'un de nos plus grands défis.

- La difficulté : Les programmes utilisés étant très puissants et les possibilités de piratage étant multipliés avec autant de personnalisation des commandes de la part de l'utilisateur, nous nous sommes rendus compte de la dangerosité de laisser l'utilisateur taper ses propres commandes à exécuter sur le cluster.
- La solution : Faire un système de formulaire envoyant ses informations, vérifiées par le site web, à un shell script qui se charge de générer la commande. Elle est ensuite

exécutée par un autre utilisateur linux que celui accessible par l'utilisateur web pour éviter que ce dernier ait accès à nos programmes.

3.2. Problème de permission

Une difficulté rencontrée lors du premier transfert de la plateforme web vers le Raspberry Pi a été un problème de droit d'exécution ainsi qu'un droit de modification.

- **La difficulté** : Lorsque l'on envoyait les paramètres depuis le formulaire, le site plantait directement. Il pouvait avoir différentes explications à cela comme par exemple les chemins ou bien une fonctionnement différent de la commande donner dans shell_exec ou encore un problème de droit sur l'utilisateur.

- **La solution :** Nous avons dû ajuster les droits d'accès et les permissions du système. Les programmes de calcul ne sont accessibles qu'aux utilisateurs de confiance en exécution comme en écriture (pas www-data), et le script shell de génération des commandes est accessible uniquement en exécution.

3.3. Stabilité de l'affichage des performances

- **La difficulté :** Une section dédiée à la visualisation des performances en fonction d'un problème donné avait été initialement prévue dans le tableau de bord. Cependant, nous avons rencontré des problèmes d'instabilité logicielle rendant l'affichage incohérent ou trop lent par rapport à l'exécution réelle sur le cluster.

- **La décision :** Par souci de **fiabilité** et pour garantir une expérience utilisateur fluide, nous avons pris la décision de retirer cette fonctionnalité de la version finale. Cela nous a permis de nous concentrer sur la robustesse du cœur de l'application : le lancement et la récupération des calculs.

3.4. Arbitrages sur les fonctionnalités

Le cas de l'historique

Une fonctionnalité d'historique des calculs passés était initialement prévue pour permettre aux utilisateurs de consulter leurs anciennes requêtes. Cependant, ce module a été écarté pour deux raisons :

- **Manque de temps :** La priorité a été donnée à la mise en place des modules sur la plateforme web qui a été plus longue que prévu

- **Pertinence limitée :** Après analyse, il s'est avéré que pour cette version du projet, l'affichage immédiat du résultat était suffisant. Stocker chaque itération de calcul aurait complexifié inutilement la base de données sans apporter une réelle valeur ajoutée à l'utilisateur final à ce stade.

4.Conclusion

En suivant une approche organisée, depuis la création graphique jusqu'à la mise en œuvre technique, la plateforme web créée répond aux objectifs initiaux : faciliter l'accès au calcul parallèle, regrouper le démarrage des opérations et présenter les résultats de façon précise et instantanée. Les choix technologiques (PHP, HTML/CSS, JavaScript, MySQL) se sont révélés adaptés au contexte du projet et ont permis d'assurer une bonne séparation entre l'interface utilisateur, la logique applicative et l'exécution des calculs distribués.

Nous avons particulièrement mis l'accent sur la sûreté et la qualité du code, notamment en gérant efficacement les sessions, en utilisant des méthodes de hachage performantes pour les mots de passe et en organisant le code afin d'optimiser sa maintenance et son évolution. d'une performance, d'un historique des opérations ou encore d'une amélioration de l'interface utilisateur.

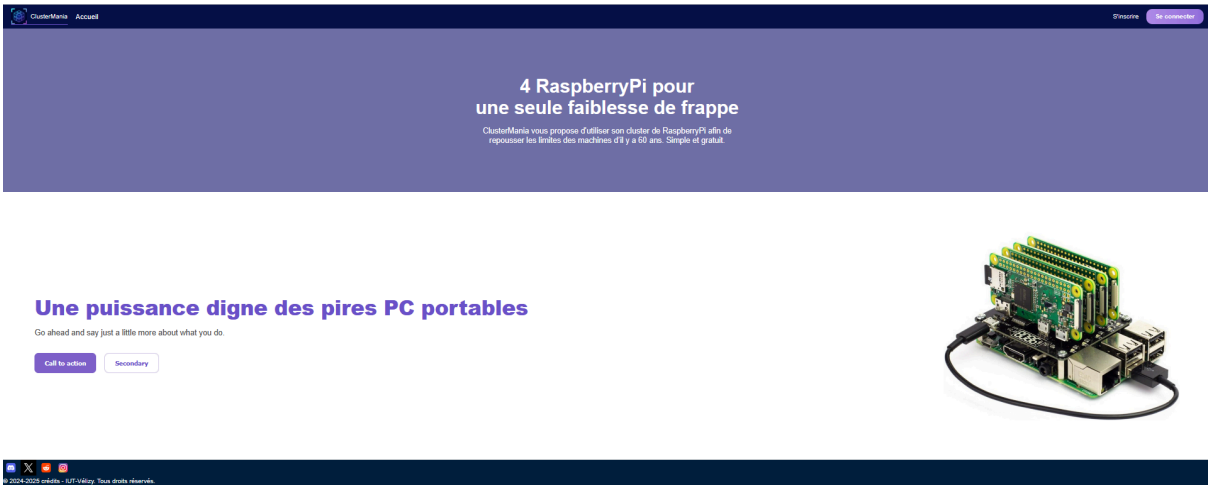
Les problèmes rencontrés, particulièrement en ce qui concerne les autorisations système et l'affichage des performances, ont abouti à des décisions techniques minutieusement élaborées pour assurer la stabilité et la fiabilité de l'application. En définitive, ce projet représente une fondation robuste et opérationnelle qui pourra être améliorée par la suite, en particulier grâce à l'intégration d'instruments d'analyse.

4.2. Etat final du site

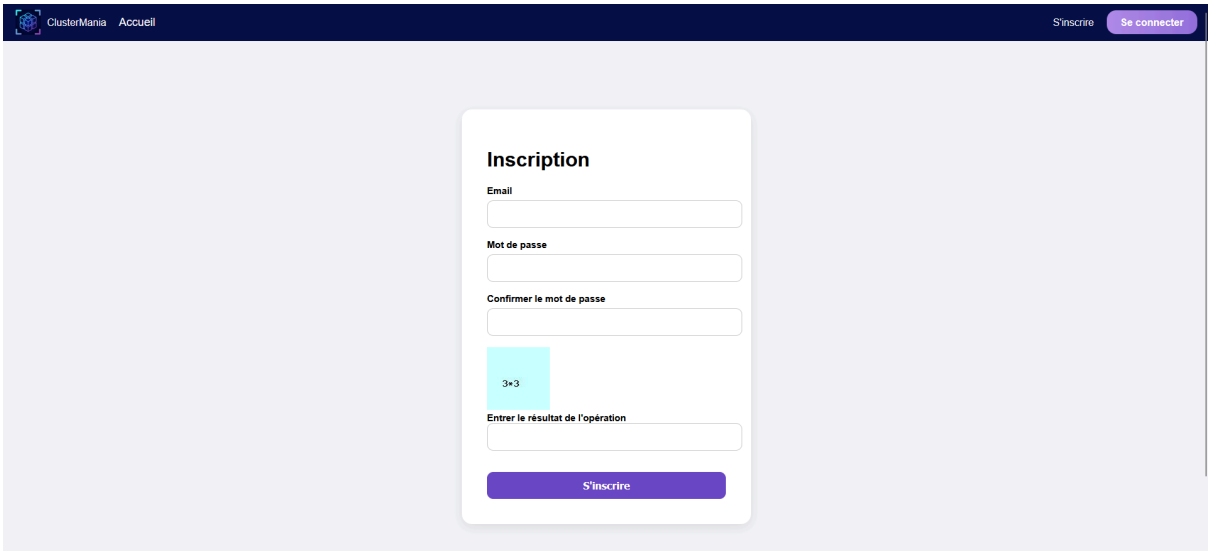
Barre de navigation quand l'utilisateur n'est pas connecté :

Barre de navigation quand l'utilisateur est connecté :

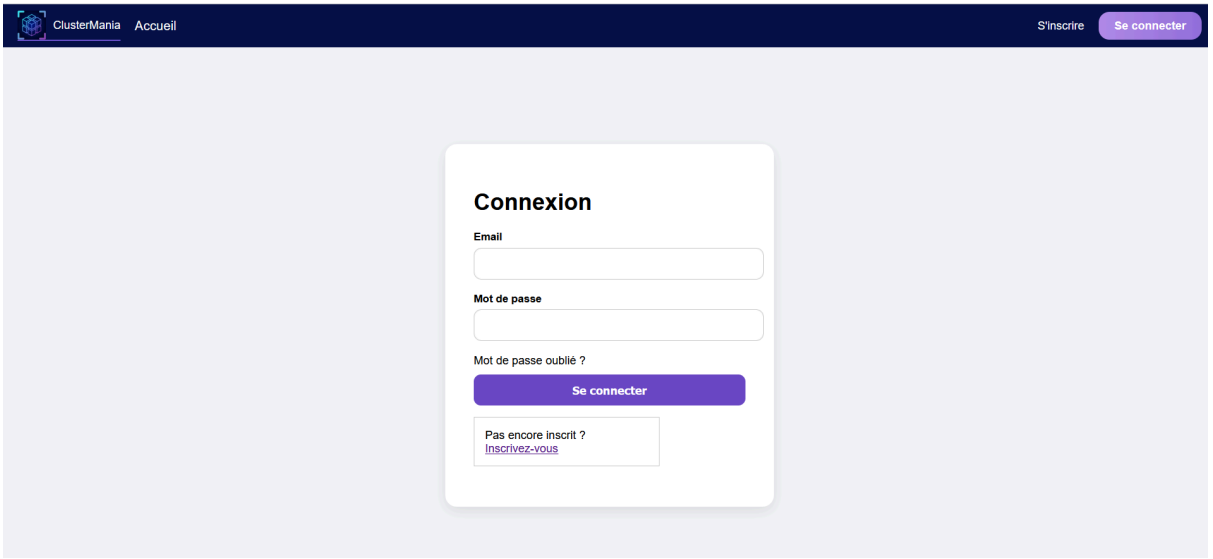
Page d'accueil :



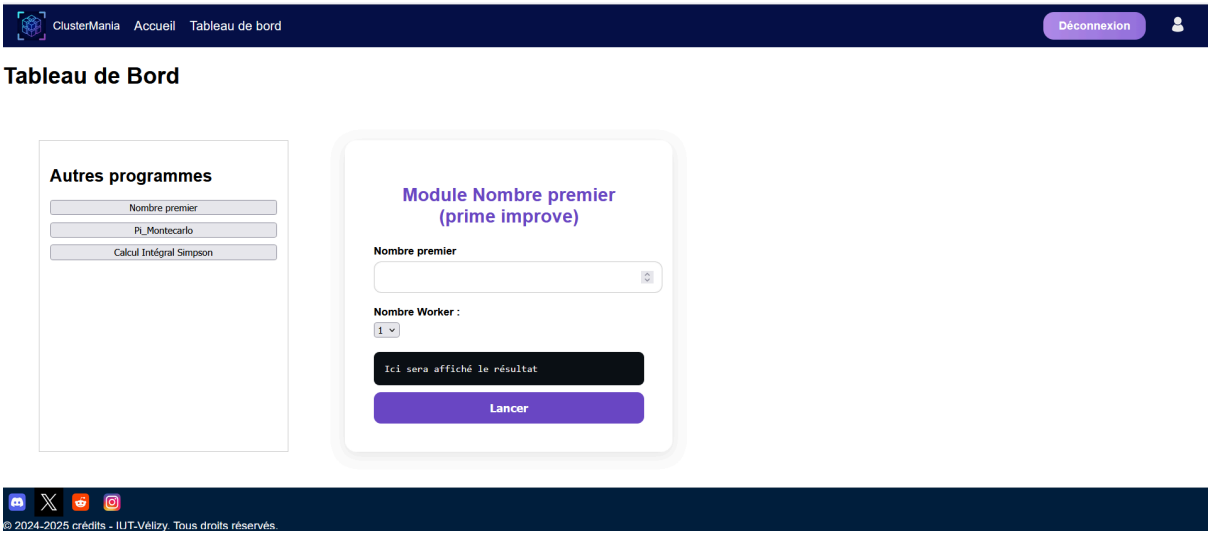
Page d’inscription :



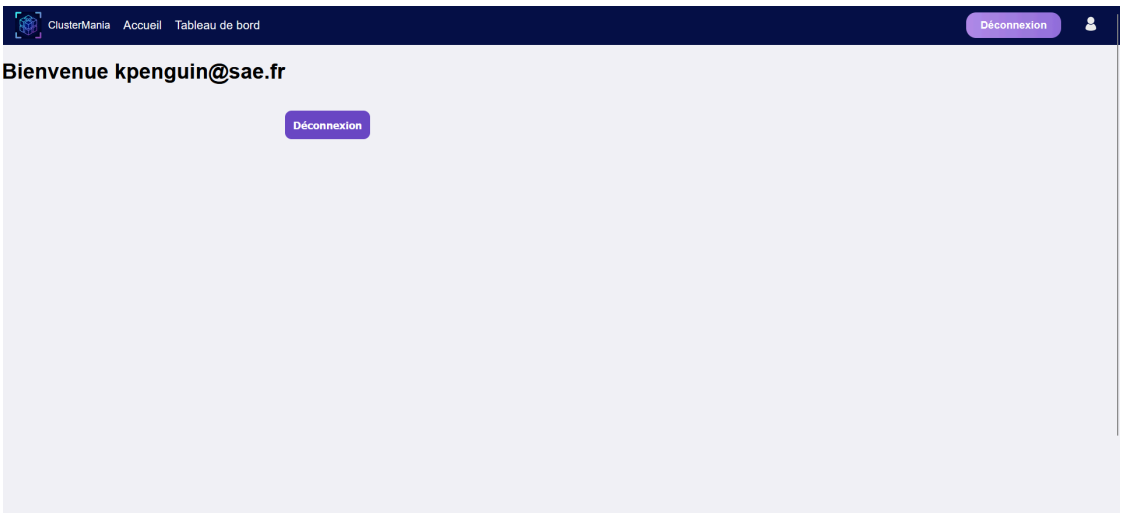
Page de connexion :



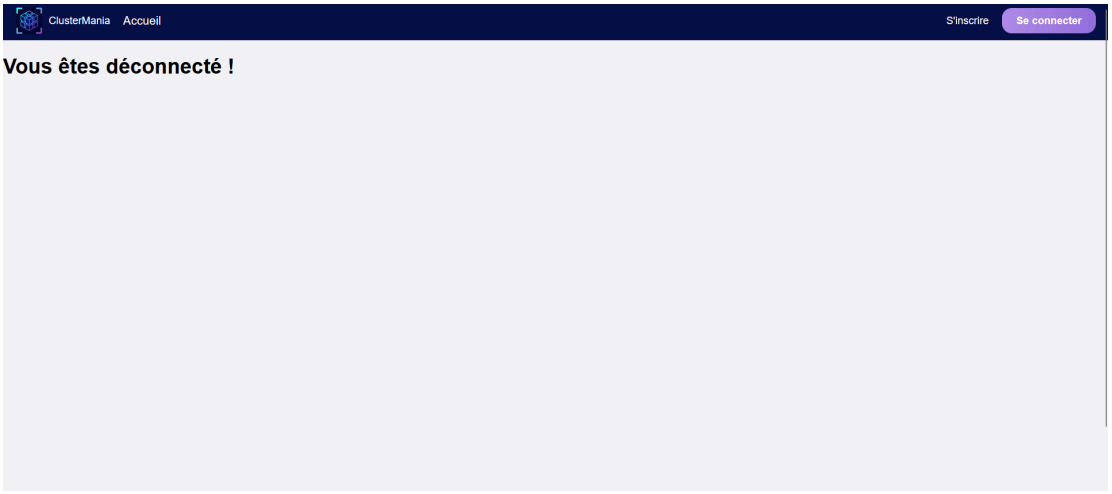
Page du tableau de bord :



Page du profile :



Page de déconnexion :



Page de redirection :