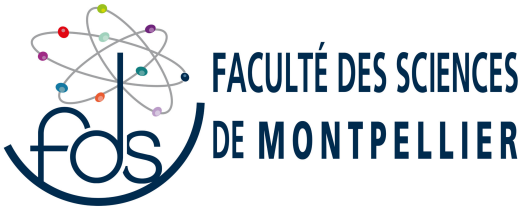



# Documentation sur l'architecture technique

Master 1 physique numérique

Projet réalisé à l'Université de Montpellier
Année scolaire 2022–2023

---

# Architecture technique

---

**Membres du projet :**
Khalis Attou
Pierre Aunay
Antoine Charvin
Tristan Goninet
Lucas Jaspard
Lidia Lappo
Morgane Lendrin

**Encadrants :**
Anne-Muriel Arigon
Brahim Guizal
Hervé Peyre

# 1 Introduction

Ce document a pour but de présenter l'architecture de l'application web que nous allons concevoir. L'architecture respectera le modèle MVC (Modèle-Vue-Contrôleur), nous pourrons ainsi facilement scinder le développement front-end et back-end.

Nous exposerons dans ce document les solutions techniques envisagées ainsi que leurs architectures. Voici un schéma global résumant l'architecture de l'application :

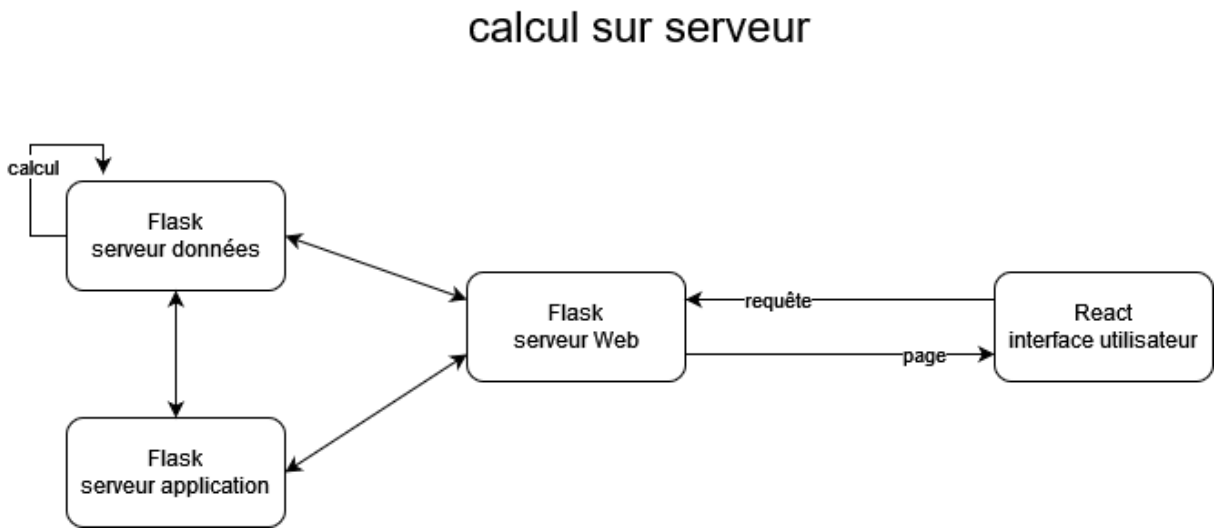


FIGURE 1 – schéma de l'architecture

# 2 Front end

Le front-end, également appelé interface utilisateur, est la partie d'une application web ou d'un site internet que les utilisateurs voient et avec laquelle ils interagissent directement.
C'est la combinaison de la conception, du développement et de la présentation visuelle de l'application, ainsi que de l'implémentation des fonctionnalités qui permettent aux utilisateurs d'effectuer des actions et de recevoir des informations. Le front-end se concentre sur la création d'une expérience utilisateur agréable, intuitive et réactive en utilisant des technologies telles que HTML, CSS et JavaScript.[5]

- HTML est le langage de balisage utilisé pour structurer et organiser le contenu d'une page web. Il définit la structure des éléments tels que les titres, les paragraphes, les images, les liens, etc.[4]
- CSS (Cascading Style Sheets) est un langage utilisé pour définir l'apparence visuelle d'une page web. Il permet de contrôler les couleurs, les polices, les marges, les positions des éléments, et bien plus encore. CSS permet de donner une véritable identité grapgique à un site web.[4]
- JavaScript est un langage de programmation polyvalent qui permet d'ajouter des fonctionnalités interactives et dynamiques à une page web. Il peut être utilisé pour gérer des événements utilisateur, effectuer des calculs, valider des formulaires, communiquer avec des serveurs, créer des animations, etc. JavaScript permet de rendre une page web plus réactive et d'offrir une expérience utilisateur améliorée.[4]

## 2.1 Framework React

On appelle framework front-end tout ensemble de classes, fonctions et utilitaires qui nous facilite la création d'applications web. Ces outils sont compatibles avec tous les navigateurs.

L'ambition de React est de créer des interfaces utilisateurs, avec un outil rapide et modulaire. L'idée principale derrière React est que vous construisiez votre application à partir de composants. Un composant regroupe à la fois le HTML, le JS et le CSS, créés sur mesure pour vos besoins, et que vous pouvez réutiliser pour construire des interfaces utilisateurs.[6]

Ce principe de composants répond parfaitement à la demande de programmation orientée objet dans notre projet. Une architecture basée sur des composants offre une modularité élevée et une extensibilité aisée.

En somme, l'utilisation de React pour le développement d'un projet informatique de simulation du phénomène de diffraction peut offrir des avantages significatifs en termes de réactivité, de réutilisabilité, de performance et de compatibilité multiplateforme.

### 2.1.1 Communication React / Flask

Dans une interface React, il est possible de générer des fichiers JSON contenant les données de la simulation de diffraction optique et de les envoyer à un serveur de calcul. Cela peut être réalisé en collectant les données de l'interface utilisateur, en les convertissant en format JSON à l'aide de fonctions JavaScript, puis en les envoyant au serveur à l'aide de requêtes HTTP.[1]

Le serveur de calcul peut ensuite interpréter le fichier JSON, effectuer les calculs nécessaires et renvoyer les résultats à l'interface React. Cela permet une communication efficace entre l'interface et le serveur pour exécuter la simulation.

# 3 Back-end

## 3.1 Serveur web

### 3.1.1 Flask

Flask est un micro-framework web en Python qui offre une approche simple pour la création de serveurs web.[1]

Son fonctionnement repose sur le principe d'une architecture à routeur, où les différentes parties de l'application sont gérées par des fonctions appelées "routes". Ces routes sont définies en associant des URL spécifiques à des fonctions Python qui décrivent le comportement attendu lorsque cette URL est requêtée.

Lorsque Flask reçoit une requête HTTP, il examine l'URL demandée et recherche la route correspondante. Une fois que la route est trouvée, Flask appelle la fonction associée à cette route, et cette fonction peut effectuer diverses actions, telles que récupérer des données depuis une base de données, générer une réponse HTML ou JSON, ou encore rediriger l'utilisateur vers une autre page.
Nous avons choisis cette solution afin de rester au plus possible sur un langage que nous maîtrisons tous, notamment au vu du peu de temps restant pour l'implémentation. Une autre solution est présentée dans la partie suivante.

### 3.1.2 Node JS

Il est également possible d'utiliser le framework Node Js pour cette partie serveur web afin de tout écrire dans un seul et même langage : JavaScript.[3]

### 3.1.3 Déploiement

L'application sera déployée en utilisant deux logiciels de serveur web, à savoir Nginx et Apache. Cette approche permet de tirer parti des avantages de chaque serveur. Pour gérer efficacement un grand nombre de connexions simultanées, Nginx sera positionné en tant que première couche, devant Apache.

Nginx joue un rôle de proxy frontal pour Apache, ce qui signifie que lorsque Nginx reçoit une demande de contenu statique, il peut directement fournir les fichiers au client, sans avoir besoin de passer par Apache.

On peut également dire que Nginx agit comme un proxy inverse. Dans ce cas, lorsqu'une demande de contenu dynamique est reçue, Nginx transmet cette demande à Apache. Apache traite alors la demande et renvoie les résultats au client en passant par Nginx.

## 3.2 Modèle

Le modèle sera implémenté en Python. Dans le cas où les calculs se font à distance, nous utiliserons le framework Flask. Si ils se font en local, nous opterons pour le framework React et le modèle sera implémenté en JavaScript.

### 3.2.1 Flask

Le serveur de calcul sera implémenté en Flask. Nous avons choisi ce framework afin de rester sur un langage sur lequel nous sommes tous à l'aise et sécuriser cette fonctionnalité de calcul sur serveur.[1]

Nous avons choisi cette solution pour les mêmes arguments que précédemment. Encore une fois, une autre solution envisageable est présenté dans la partie suivante.

### 3.2.2 Node JS

Le serveur de calcul peut être implémenter avec Flask pour les mêmes arguments que dans l'une des parties précédentes, à savoir utiliser un seul et même langage pour toute l'application.[3]

### 3.2.3 React JS

Dans l'éventualité où les calculs sont effectués en temps réel sur la machine utilisateur, ce qui réduit la dépendance vis-à-vis du serveur et élimine les temps de latence liés à l'envoi des données et à la réception des résultats, l'algorithme sera implémenté à l'aide du framework React. Une description plus complète de ce framework est disponible en partie 2.1.[6]

### 3.2.4 PyScript

Nous avions pensés utiliser Pyscript, un framework JavaScript qui permet de lancer du code python dans des balises HTML. Cependant nous avons rencontrés quelques problèmes lors de tests menés avec celui-ci comme par exemple la gestion du format JSON. De plus ce framework étant très jeune, il dispose de peu de documentation.[2]

# 4 Tests

## 4.1 Test front-end

Afin de tester si nous pouvons utiliser React pour gérer la partie interface utilisateurs, nous avons réaliser un test d'application web React.

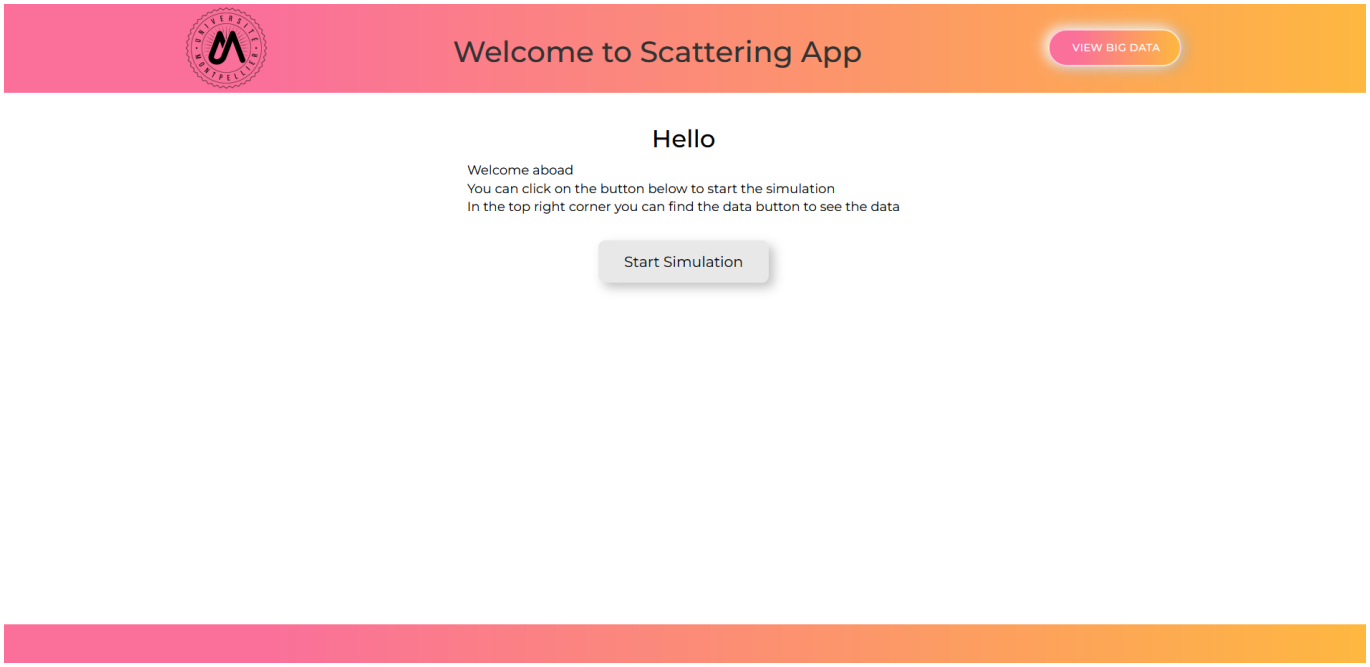


FIGURE 2 – Aperçu de la page d'acceuil avec React

## 4.2 Test back-end

Afin de tester si un serveur Flask convient à notre problèmatique, nous avons réaliser un test. Pour ce test l'utilisateur entre une taille minimal et maximal de matrice à inverser ainsi qu'un pas pour parcourir cette liste de matrice. Ensuite nous inversons chaque matrice dans Flask en mesurant le temps d'inversion à chaque itération. Le résultat est ensuite envoyé à React au format Json afin d'être affiché avec Plotly.js. Chaque matrice est une matrice de nombres complexes aléatoires et l'inversion est réalisée à l'aide de la librairie Numpy.

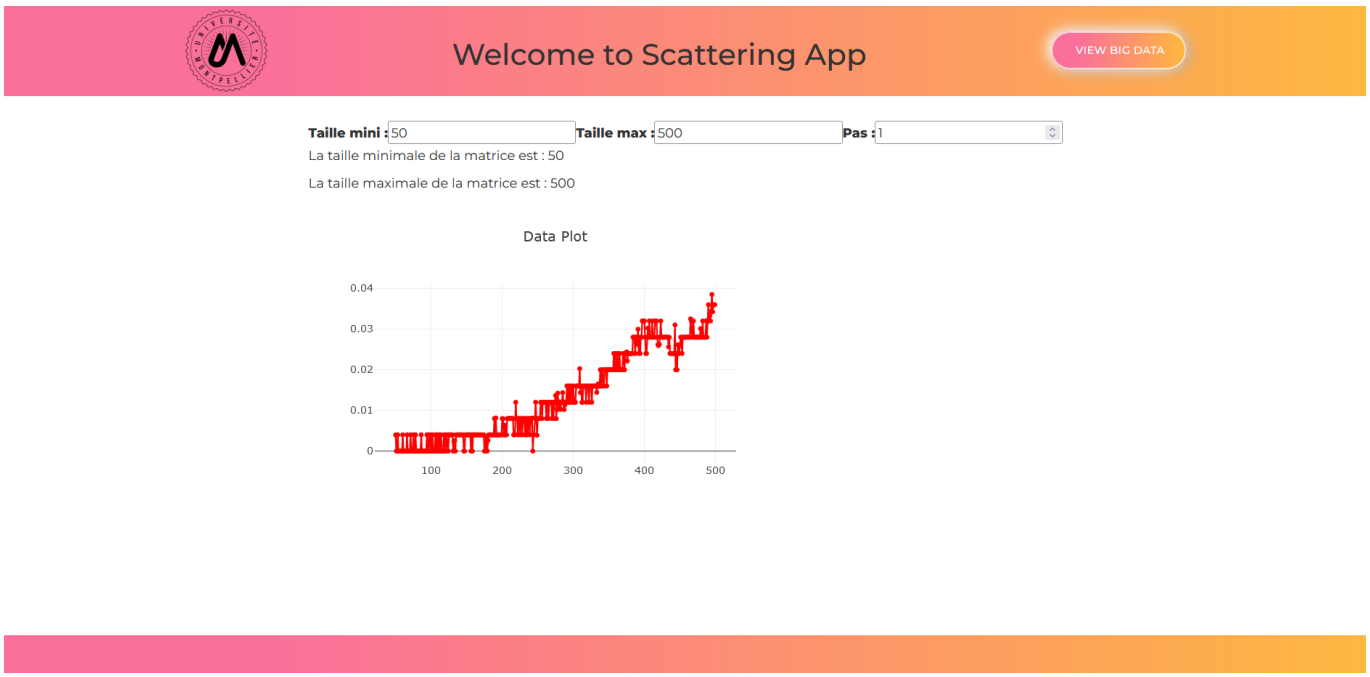


FIGURE 3 – Apperçu de la réponse renvoyé par Flask

Ce test nous aura permis de faire fonctionner ensemble une interface React avec un serveur Flask avec un cas concret d'inversion de matrices, le tout dans un temps d'exécution raisonnable. Ces tests ont été fait sur Mozilla Firfox, Chrome ainsi que Safari.

### 4.3 Choix retenu

Le choix que nous avons retenu est d'uiliser React pour l'interface utilisateur ainsi que Flask pour le serveur. Le modèle sera implémenté en Python et en Javascript pour respectivement un lancement du calcul sur un serveur dédié ou sur le client. La pertinence de ce choix a été validé après consultation par P. POMPIDOR.
@

## Références

[1] FLASK. *Welcome to Flask — Flask Documentation (2.3.x)*. URL : `https://flask.palletsprojects.com/en/2.3.x/` (visité le 29/05/2023).

[2] Anaconda INC. *Pyscript.net*. URL : `https://pyscript.net/` (visité le 29/05/2023).

[3] NODE.JS. *Documentation*. Node.js. URL : `https://nodejs.org/en/docs` (visité le 29/05/2023).

[4] OPENCLASSROOM. *Créez votre site web avec HTML5 et CSS3*. OpenClassrooms. URL : `https://openclassrooms.com/fr/courses/1603881-creez-votre-site-web-avec-html5-et-css3` (visité le 29/05/2023).

[5] OPENCLASSROOM. *Formations en ligne et cours en accès libre*. URL : `https://openclassrooms.com/fr/` (visité le 29/05/2023).

[6] REACT. *React*. URL : `https://react.dev/` (visité le 29/05/2023).