How To Make an App

SmartMonday du 07 avril 2025

None

Cas pratique : le CISTEM

Problématique : Automatisation et simplification de la supply chain & tracking au FOSDEM [Catering]

https://github.com/Cerkinfo/CISTEM_

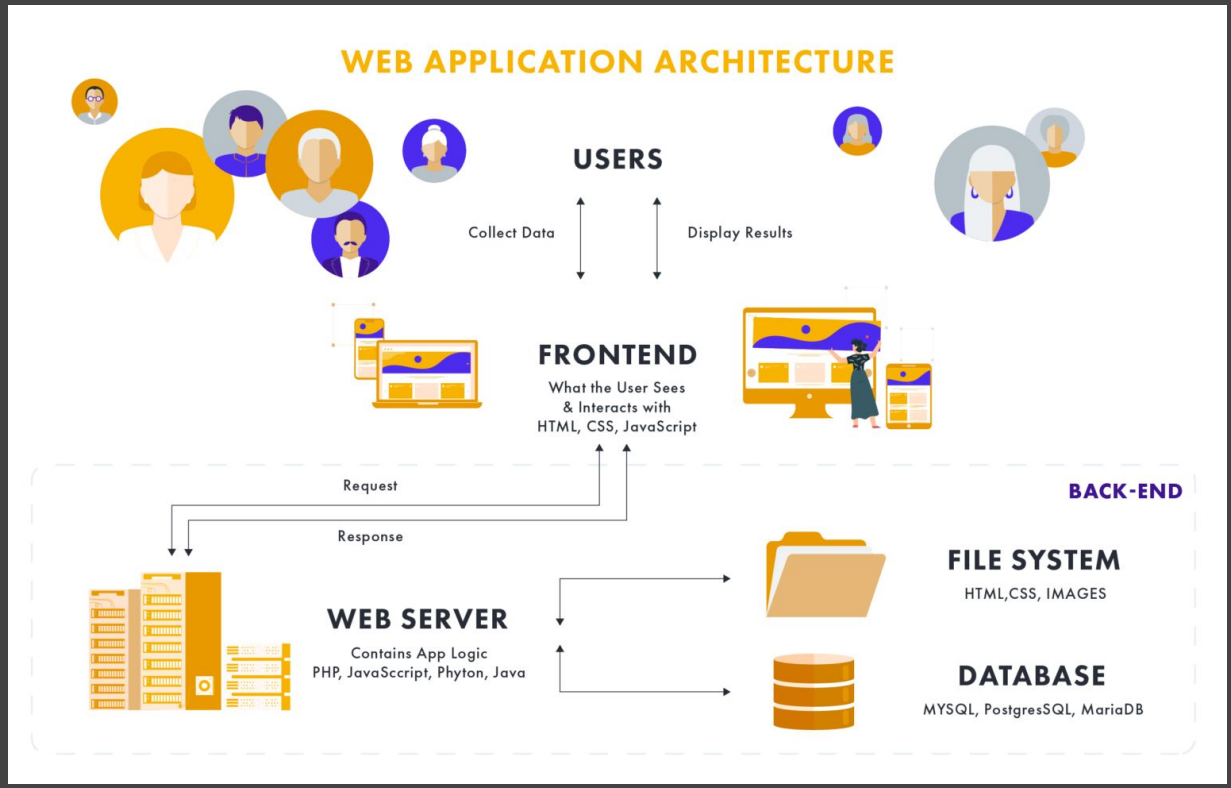
```
$> npm install
```

```
$> cd apps/CISTEM-public && npm run compile && cd ../../
```

```
$> cd apps/FOSDEM-bar && npm run compile && cd ../../
```

```
$> turbo dev
```

Cas pratique : le CISTEM









Choisir son language WebApp

CISTEM : JavaScript.

Avantages :

- **Multiplateforme** : Fonctionne à la fois côté client (navigateur) et côté serveur (grâce à Node.js).
- **Grande communauté** : beaucoup de bibliothèques, frameworks et documentation.
- **Rapidité** : Interprété directement par le navigateur, sans besoin de compilation préalable.
- **Interactivité** : Idéal pour les sites web dynamiques et interactifs.
- **Facilité d'apprentissage** : Accessible même pour les débutants en programmation.

Inconvénients :

- **Failles de sécurité** : Étant exécuté côté client, il peut être manipulé ou exploité
- **Dynamique et parfois imprévisible** : Les types faibles peuvent mener à des erreurs difficiles à détecter.
- **Fragmentation** : Les comportements peuvent légèrement varier entre les navigateurs.
- **Performance limitée** : Bien que performant pour les tâches légères, il peut être moins adapté pour des applications lourdes en calcul.

Alternatives : TypeScript, Python, Ruby, Dart (Google)









Choisir un Package Manager

CISTEM : NPM https://npmjs.com/

Avantages :

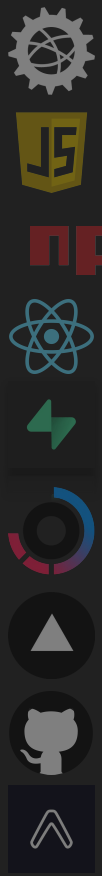
- **Large écosystème** : npm propose des millions de paquets JavaScript disponibles, couvrant pratiquement tous les besoins de développement.
- **Facilité d'utilisation** : Installer ou mettre à jour des paquets se fait avec des commandes simples
- **Gestion des dépendances** : npm facilite le suivi et la résolution des dépendances des projets.
- **Automatisation** : Avec des scripts npm, tu peux automatiser des tâches comme la compilation, les tests, etc.
- **Intégration étroite avec Node.js** : npm est livré par défaut avec Node.js, ce qui rend son adoption facile.

Inconvénients :

- **Problèmes de sécurité** : Les paquets malveillants ou vulnérables dans l'écosystème peuvent être une menace.
- **Conflits de versions** : Les multiples versions des dépendances dans des projets complexes peuvent entraîner des conflits.
- **Trop de choix** : L'abondance de paquets peut rendre difficile de choisir le meilleur ou le plus fiable.
- **Performances** : Les anciennes versions de npm (avant la v5) avaient des problèmes de lenteur, bien que cela se soit grandement amélioré.

Alternatives : Yarn, pnpm, Bun

Choisir un Package Manager



```
$> npm install
```

```
added 2160 packages, and audited 2167 packages in 2m

398 packages are looking for funding
  run `npm fund` for details

27 vulnerabilities (3 low, 5 moderate, 16 high, 3 critical)
```








Choisir un Framework

CISTEM : React (Facebook) https://react.dev/

Avantages :

- **Composants réutilisables** : La logique des composants permet de construire une application en morceaux modulaires, ce qui améliore la maintenance et la réutilisation du code.
- **Performances élevées** : Grâce au Virtual DOM (Document Object Model), React optimise le rendu en ne mettant à jour que les parties nécessaires.
- **Large communauté et écosystème** : Une documentation complète, des milliers de bibliothèques tierces, et un grand nombre de développeurs travaillant sur React garantissent un bon support.
- **Flexibilité** : Contrairement à certains frameworks, React se concentre uniquement sur la vue (le "V" dans MVC), offrant une grande liberté pour structurer les applications.
- **Apprentissage populaire** : Très demandé par les entreprises, avec des ressources nombreuses pour les débutants.

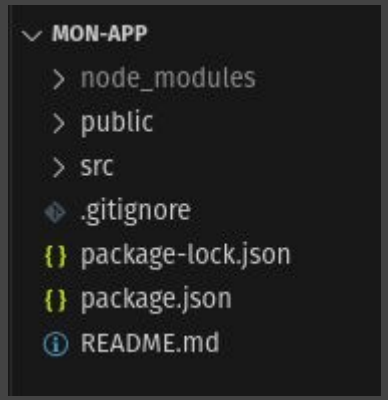
Inconvénients :

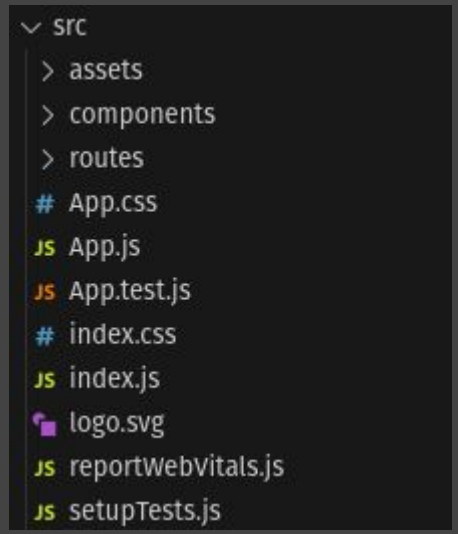
- **Courbe d'apprentissage** : Bien que simple en apparence, des concepts comme JSX, le State et les hooks (useState, useEffect, etc.) peuvent être intimidants pour les débutants.
- **Configuration nécessaire** : Il faut souvent compléter React avec d'autres outils (comme Redux, React Router ou Webpack) pour créer une application complète.
- **Évolutions rapides** : L'écosystème React évolue rapidement, ce qui peut entraîner une obsolescence rapide de certains outils ou pratiques.
- **Limité à la vue** : Nativement, React ne gère pas le backend ou d'autres aspects d'une application, ce qui peut nécessiter des frameworks supplémentaires pour un projet complexe.

Alternatives : Vue.js, Angular, Solid.js

Choisir un Framework

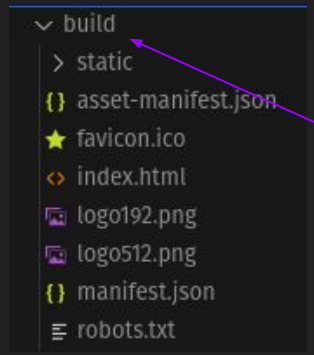
```
$> npx create-react-app mon-app
```



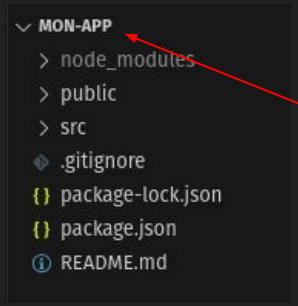


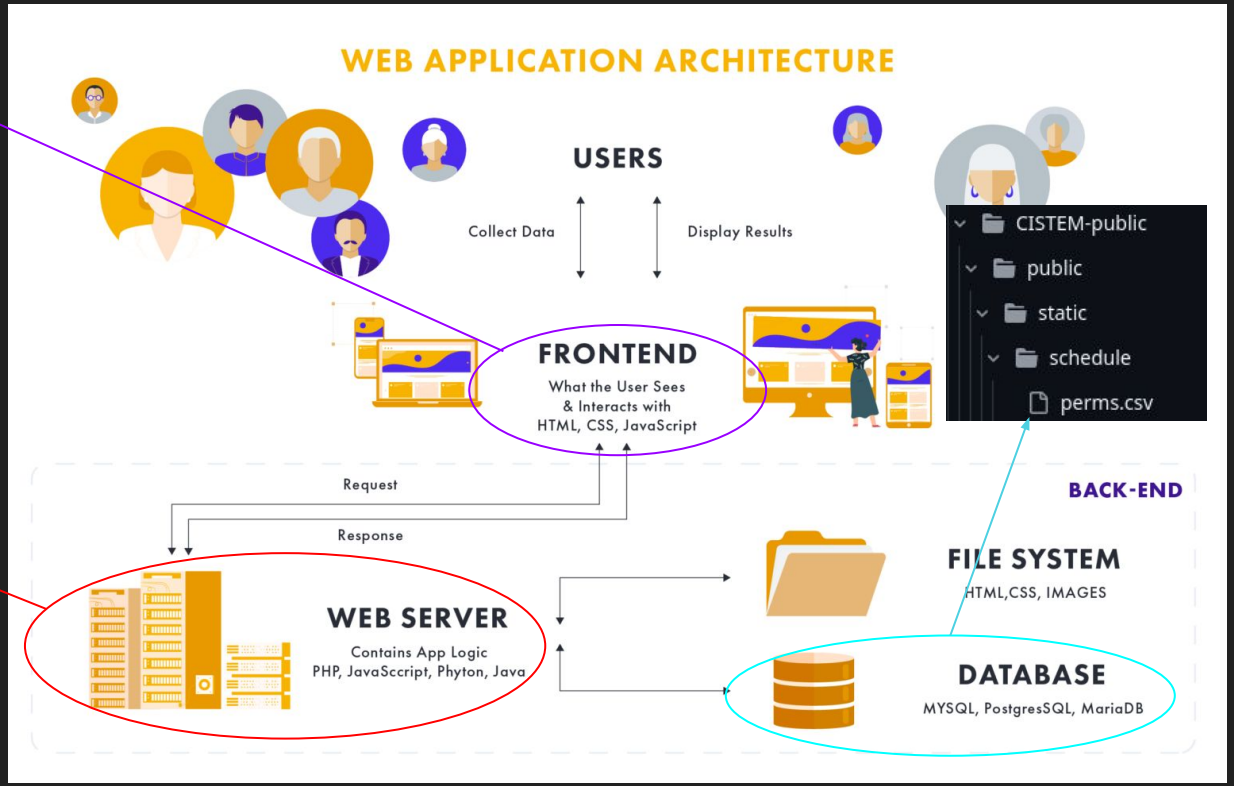
On a notre mini-WebApp Statique !

ℹ️ CISTEM-public & FOSDEM-bar s'arrêtent ici



$> npm run build





La DataBase : les .csv ça va 2 secondes

CISTEM : Supabase https://supabase.com/

Avantages :

- **Open-source** : Supabase est une alternative open-source à Firebase, ce qui permet une personnalisation et une transparence accrues.
- **Base de données PostgreSQL** : Chaque projet inclut une base de données PostgreSQL robuste, offrant des fonctionnalités avancées et une compatibilité SQL complète.
- **API instantanées** : Supabase génère automatiquement des API RESTful et GraphQL basées sur le schéma de la base de données.
- **Authentification intégrée** : Gestion des utilisateurs avec des politiques de contrôle d'accès basées sur PostgreSQL.
- **Fonctionnalités en temps réel** : Permet des mises à jour instantanées via des websockets, idéal pour les applications collaboratives.
- **Facilité de configuration** : Une expérience développeur fluide et rapide.

Inconvénients :

- **Fonctionnalités limitées** : Certaines fonctionnalités sont encore en phase alpha ou bêta, ce qui peut limiter leur utilisation.
- **Support limité** : Le support pour les tokens API à long terme et certaines intégrations peut être insuffisant.
- **Pas de correspondance parfaite avec Firebase** : Bien qu'il soit une alternative, il ne couvre pas toutes les fonctionnalités de Firebase.
- **Dépendance à Internet** : En cas de panne réseau, l'accès à l'application SaaS peut être compromis.

Alternatives : Firebase (Google), PocketBase, MongoDB, Appwrite

La DataBase : les .csv ça va 2 secondes

Versionalisation, Production et Sécurité

À ce stade il devient important de se poser des questions importantes sur le projet, son ampleur et son avenir.

Supabase vous incitera d'ailleurs fortement à réfléchir à la sécurité avec sa auth RLS à configurer sur chaque table.

Dans les gros projets, c'est lors de cette étape qu'on commence à mettre en place une version de preview et de production pour l'application, afin de ne pas tout casser lors du développement d'une nouvelle fonctionnalité.

Enfin, il deviendra également utile de versionalisé son projet et déployer par strate les modifications plutôt que de déployer chaque nouveau commit. Ainsi, cela permettra de pouvoir faire un historique des versions et déployer une version précédente fonctionnelle si un problème survient avec votre update.

La DataBase : les .csv ça va 2 secondes

```
import { createClient } from "@supabase/supabase-js";

export const supabaseUrl = process.env.EXPO_PUBLIC_SUPABASE_URL ||
  "https://XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.supabase.co";
export const supabaseAnonKey = process.env.EXPO_PUBLIC_SUPABASE_ANON_KEY ||
  "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6ImthZ3
console.log("🚀 ~ supabaseUrl:", supabaseUrl);
export const supabase = createClient(supabaseUrl, supabaseAnonKey);
```

```
import { supabase } from "./supabase";

export const fetch_orders = async () => {
  try {
    const { data, error } = await supabase
      .from('orders')
      .select(`
        id, created_at, location(name), name, order, status
      `)

    if (error) throw new Error(`Erreur lors de la récupération des commandes : ${error.message}`);

    return { success: true, data: data };
  } catch (error) {
    console.error(error);
    return { success: false, message: error.message };
  }
};
```









Monorepo & Optimisation

CISTEM : Turborepo https://turbo.build

Problématique : j'ai plusieurs webapp qui partagent des design, assets, services,… Et j'aimerais également les développer en même-temps.

Avantages :

- **Mise en cache intelligente** : Turborepo utilise un système de cache local et distant pour éviter de répéter les tâches déjà effectuées, ce qui accélère considérablement les builds.
- **Exécution parallèle** : Les tâches sont exécutées en parallèle tout en respectant les dépendances, ce qui optimise les performances.
- **Flexibilité** : Compatible avec npm, Yarn et pnpm, il s'intègre facilement dans des projets existants sans imposer de structure rigide.
- **Support natif TypeScript et JavaScript** : Idéal pour les projets modernes utilisant ces technologies.
- **Optimisation CI/CD** : Réduit les temps d'exécution des pipelines grâce à son cache partagé.

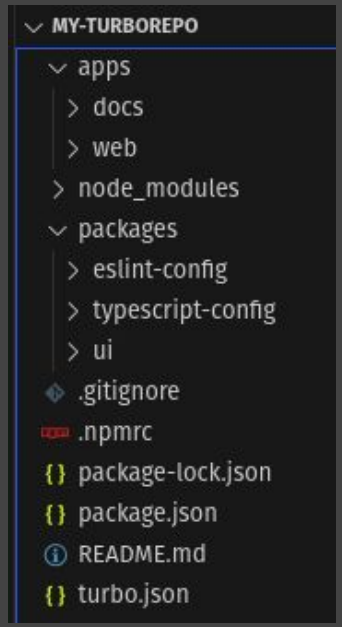
Inconvénients :

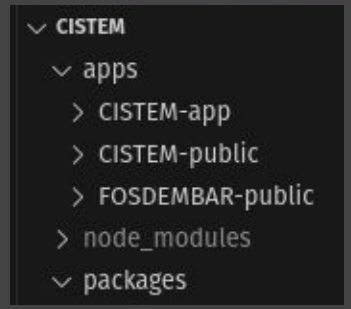
- **Courbe d'apprentissage** : Les concepts comme la mise en cache et la configuration peuvent être complexes pour les débutants.
- **Dépendance à l'écosystème JavaScript** : Moins adapté pour des projets utilisant d'autres langages ou frameworks.
- **Fonctionnalités avancées limitées** : Comparé à certains outils comme Bazel, il peut manquer de fonctionnalités pour des projets très complexes.
- **Communauté encore en croissance** : Bien qu'il gagne en popularité, il a une communauté plus petite que des outils comme Nx.

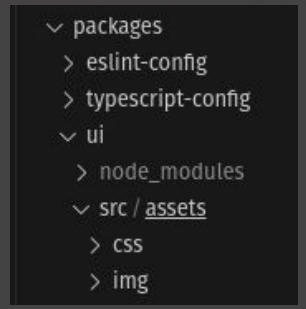
Alternatives : Nx, Lerna, Rush, Bazel, Pants

Monorepo & Optimisation

```
$> npx create-turbo@latest my-turborepo
```







Déploiement : Dynamique

CISTEM : Vercel https://vercel.com/

Avantages:

- **Déploiement simplifié** : Vercel offre une intégration fluide avec GitHub, GitLab et Bitbucket, permettant des déploiements automatiques à chaque commit.
- **Performances élevées** : Grâce à son réseau CDN mondial et à son architecture serverless, les sites et applications déployés sont rapides et fiables.
- **Support des frameworks modernes** : Conçu pour des frameworks comme Next.js, React, et d'autres outils modernes.
- **Aperçus en direct** : Permet de partager des aperçus de déploiement avec les équipes ou les clients pour des retours rapides.
- **Facilité d'utilisation** : Interface utilisateur intuitive, idéale pour les développeurs front-end.

Inconvénients :

- **Coût** : Les fonctionnalités avancées peuvent devenir coûteuses pour des projets à grande échelle.
- **Limité aux applications front-end** : Moins adapté pour des projets nécessitant un backend complexe.
- **Dépendance à l'écosystème Vercel** : Les projets peuvent devenir fortement liés à leur infrastructure, rendant la migration plus difficile.
- **Fonctionnalités serverless limitées** : Comparé à des solutions comme AWS Lambda, les options serverless sont moins flexibles.

Alternatives : Netlify, Heroku, AWS Amplify (Amazon), Render, Firebase (Google)

Déploiement : Statique

CISTEM : Github Pages (Microsoft) https://pages.github.com/

Avantages :

- **Gratuit** : Hébergement gratuit pour les sites statiques, ce qui est idéal pour les projets personnels ou open-source.
- **Facilité d'utilisation** : Intégration directe avec les dépôts GitHub, permettant un déploiement rapide en quelques clics.
- **CDN intégré** : Les sites sont servis via un réseau de distribution de contenu (CDN), garantissant des temps de chargement rapides.
- **Personnalisation de domaine** : Possibilité d'utiliser un domaine personnalisé pour ton site.

Inconvénients :

- **Limité aux sites statiques** : Ne prend pas en charge les fonctionnalités dynamiques comme les bases de données ou les scripts côté serveur.
- **Dépendance à GitHub** : Nécessite un dépôt GitHub pour fonctionner.
- **Personnalisation limitée** : Certaines fonctionnalités avancées ne sont pas disponibles.
- **Pas idéal pour les grandes entreprises** : Plus adapté aux projets personnels ou aux petites équipes.

Alternatives : Vercel, Cloudflare Pages, Surge

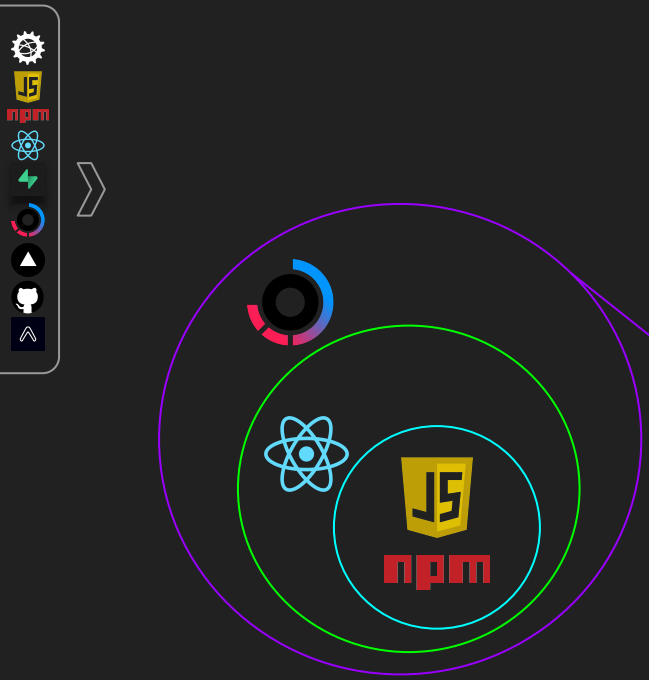
Déploiement : Statique

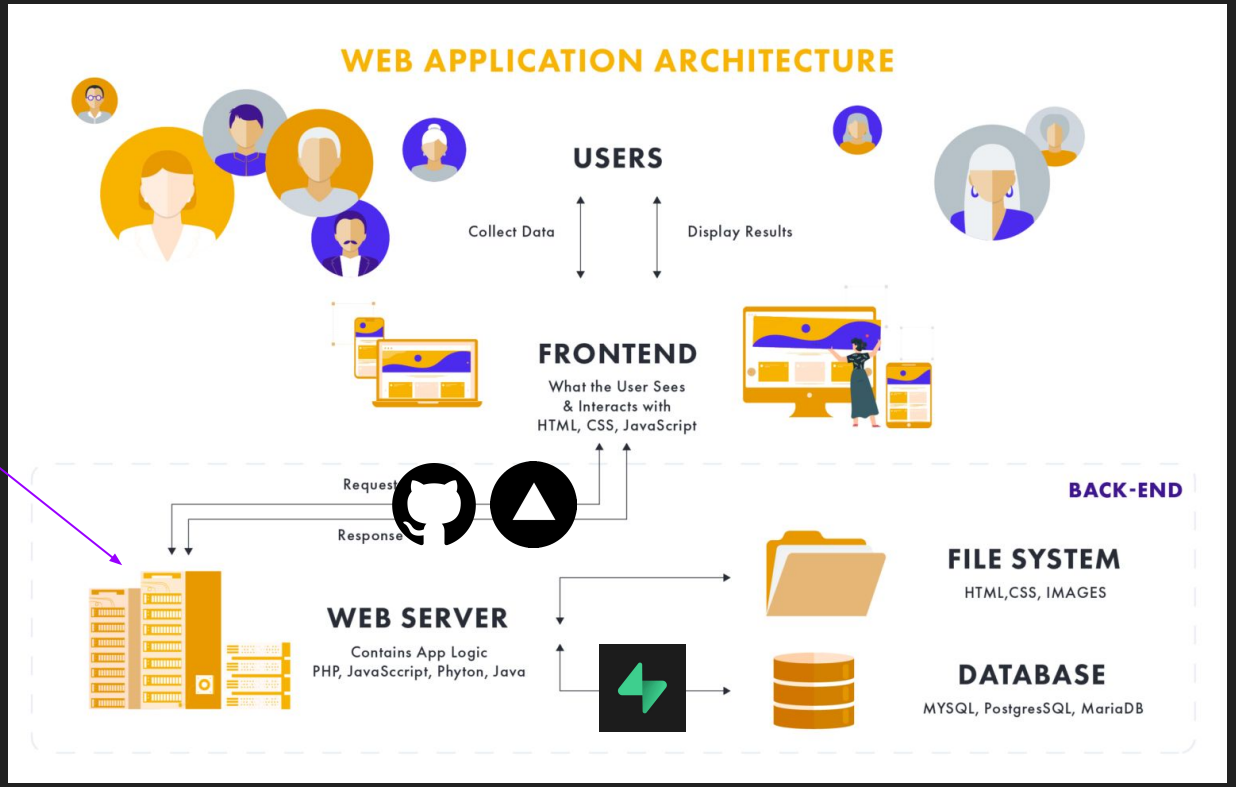
Multi-déploiement & Moindre coûts

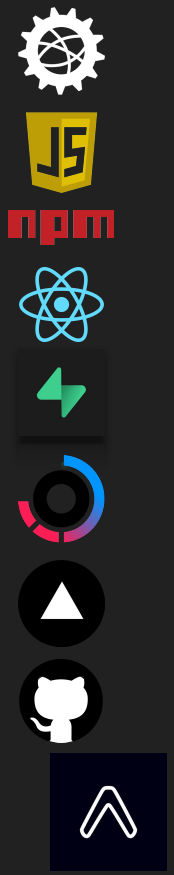
Lors du déploiement c'est également le bon moment pour passer en revue la modularité et le bon découpage des apps. Si cela est bien fait et que c'est de circonstance, il devient intéressant de déployer uniquement les pages dynamiques sur Vercel (ou en local) et de déployer le reste sur Github Pages. On aboutit ainsi à un projet bien structuré autant sur le développement que financièrement ou structurellement (si on fait un hébergement local pour le dynamique)

ATTENTION : Bien faire attention aux routes et redirections ainsi qu'aux informations partagées entre les apps.

On a notre app optimisée, structurée et accessible !







[Bonus] Déploiement Natif

Cas d’exemple : Expo https://expo.dev/

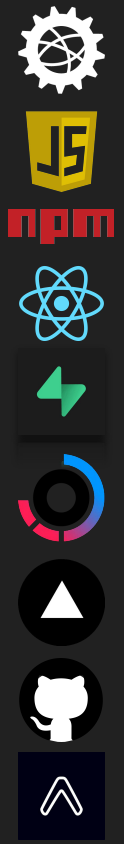
Avantages :

- **Facilité d'utilisation** : Expo permet de démarrer rapidement un projet sans avoir à configurer Xcode ou Android Studio.
- **Mises à jour OTA (Over The Air)** : Les applications peuvent être mises à jour sans passer par les stores.
- **Tests simplifiés** : Grâce à Expo Go, il est possible de tester une application en développement en scannant un QR code.
- **Gestion des librairies natives** : Expo embarque de nombreuses librairies utiles et les met à jour automatiquement.
- **Publication facilitée** : Expo simplifie le déploiement sur les stores Apple et Google.

Inconvénients :

- **Limitations sur les modules natifs** : Certaines fonctionnalités nécessitent de "détacher" Expo (Eject), ce qui peut complexifier le projet.
- **Poids des applications** : Expo inclut des librairies que vous n’utilisez peut-être pas, ce qui alourdit l’application.
- **Dépendance à Expo** : Le processus de build et de publication est lié à Expo, ce qui limite la flexibilité.

Alternatives : Flutter (Google), Swift



Merci à tous !

Des questions ?