Duval Nicolas
nicolas.duval.54@proton.me

Dataviz FT

30 septembre 2025

VUE D'ENSEMBLE

L'objectif est de développer un site permettant de visualiser les offres d'emploi de développeur informatique sur Nancy (dans un premier temps) dans une séquence temporelle et géographique (choropleth map, saisie des dates ou périodes visées, animation, nom de l'entreprise + stack requise lors du survole des points avec la souris + salaire...).
Ce site proposera d'autres indicateurs interactifs qui seront mis à jour quotidiennement (évolution du nombre d'emplois pour un département spécifique, nombre d'offres présentant le salaire visé, modifications des offres existantes etc...).
La mise à jour des annonces se fera par le biais de consommation de l'api francetravail.io avec mise en bdd, nettoyage...

Architecture technique proposée

Backend (Python)

- **Framework :** FastAPI (léger, async, documentation auto)
- **Modules :**
  - API
  - Nettoyage + enrichissement (stack detection, géocodage)
  - Exposition des données via API REST
  - Tâche planifiée (cron ou APScheduler) pour maj quotidienne
- **Tests :** pytest + coverage + mocks pour les appels externes
- **Qualité :** linting (ruff ou flake8), typage (mypy), CI GitHub Actions

Base de données

- PostgreSQL avec schéma relationnel
- PostGIS pour la cartographie (départements, points)
- Indexation : sur date, département, stack
- Historisation : table des disparitions ou changements

Frontend (React)

- **Librairies :**
    - kepler.gl pour les cartes animées
    - Recharts ou plotly.js pour les indicateurs dynamiques
    - React Query ou SWR pour la gestion des appels API
    - Design : minimal, fonctionnel, responsive

DevOps & Qualité logicielle

- **GitHub Actions :**
    - Lint + test à chaque push
    - Build + déploiement (Docker, Railway, Render, ou VPS)
- **Docker :**
    - Conteneurisation du backend + frontend
    - Volume pour la base PostgreSQL
- **Monitoring :**
    - Logs scraping + erreurs API
    - Alertes en cas d’échec de mise à jour

Compléments à envisager

Sécurité & robustesse

- Validation des entrées (Pydantic)
- Protection contre les injections (ORM ou requêtes préparées)
- Limitation du scraping pour éviter les blocages

Observabilité

- Dashboard admin (nombre d'offres, erreurs scraping, etc.)
- Statistiques internes (temps de réponse, volume de données)

Tests frontend

- Jest + React Testing Library pour les composants
- Tests d'intégration sur les appels API

Structuration du projet

- Monorepo ou multi-repo selon ton confort
- Convention de nommage, README technique, changelog

Schéma d’architecture (beta) :

ORGANISATION :

Plan séquentiel pour le projet de visualisation des offres d’emploi développeur

SPRINT 1 : (10 jours)

Étape 1 — Mise en place de la gestion de projet

- Cahier des charges simplifié
- Rédaction du README
- Schéma de l’architecture
- Mise en place du Jira en mode Scrum

Étape 2 — Création d’un mono-repo GitHub

- Un seul dépôt contenant :

  dossier backend (FastAPI + scraping)

  dossier frontend (React + dataviz)

  dossier tests

  dossier docs (cahier des charges, schémas)

  dossier .github/workflows pour la CI/CD

Étape 3 — Accès à l’API offre d’emploi de francetravail.io

- Création du compte et récupération des infos de connexion
- test de l’api

Livrable attendu :

1. Cahier des charges simplifié
2. Jira fonctionnel complet
3. Schéma architecture
4. Mono-repo GitHub
5. Acces fonctionnel à l'api francetravail.io

SPRINT 2 : (1 à 2 semaines)

Étape 1 — Collecte exploratoire de data

- Créer un module Python dédié à la consommation d'api
- Prévoir un mode manuel (exploration) et un mode automatique (mise à jour quotidienne)
- Récupérer les données brutes en JSON

Étape 2 — Nettoyage / enrichissement des données

- Identifier les champs utiles
- Préparer le corpus pour la détection des stacks

Étape 3 — Déterminer d'autres indicateurs intéressants

Étape 4 — Finaliser l'API

Étape 5 — Mise en place des tests de l'API

Livrable attendu :

1. Fichier JSON de données exploratoires
2. Liste des indicateurs potentiellement intéressant
3. Liste des stacks organisées
4. API fonctionnelle

SPRINT 3 : (1 à 2 semaines)

Étape 1 — Modélisation / schema SQL

Étape 2 — Mise en place de la BDD PostgreSQL/PostGIS

Étape 3 — Mise en place des test d'insertion

Livrable attendu :

1. Schema SQL
2. BDD
3. Test

SPRINT 4 : (3 semaines?)

Étape 1 — Mise en place du squelette React

- Maquette simplifiée du site (?)
- Initialiser le projet React
- Préparation page d'accueil et éléments de base (header, footer, composants de base)

Étape 2 — Exploration des frameworks de dataviz

- Tester Comparer fluidité, personnalisation, facilité d'intégration
- Choisir les outils les plus adaptés pour les indicateurs dynamiques
- Choix de la lib de dataviz (Kepler.gl, Recharts, Plotly.js, ECharts)
- Préparation de la page de dataviz

Étape 3 — Test React

- Composant
- Hook
- Appels API mockés

Livrable attendu :

1. Version Beta du site

SPRINT 5 : (2 semaines?)

Étape 1 — Mise en place d’un Docker Compose

Étape 2 — CI/CD GitHub Action

Étape 3 — Gestion des environnements

Livrable attendu :

1. Docker
2. CI/CD

SPRINT 5 : (2 semaines?)

Étape 1 — Couverture globale des tests

Étape 2 — Test E2E

Étape 3 — Monitoring et alerte

Étape 4 — Test d'intégration

Insérez votre texte ici

Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici.

Insérez votre texte ici

Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici Insérez votre texte ici.