

Cahier des charges projet MLOps

**| Membres du projet** : Zineb Slitine, Emmanuel Elias

## Contexte et Objectifs

## **a)** **Cadre du projet**

## La loi de transition énergétique pour la croissance verte fixe aux collectivités locales des objectifs d’efficacité énergétique. Cela passe par la réduction des émissions de gaz à effet de serre et la consommation énergétique des bâtiments.

Dans cette optique, les collectivités doivent adapter leur consommation en fonction des conditions météorologiques qui ont, sans surprise, un impact direct sur la consommation d’énergie.

L’objectif de ce projet est de mettre en place le MVP d’une application permettant de prédire la consommation énergétique en fonction des données météorologiques prévisionnelles.

* Nous sommes une agence nationale de labellisation et de notation des territoires. Nous mettons à disposition des Régions/territoires une interface “Conso Predict” qui prédit la consommation électrique du territoire en prenant en compte les prévisions météorologiques locales (température, humidité etc...).
* Les utilisateurs de l’application sont donc les territoires, qui sont les clients de l’agence. Ils ont accès à une interface leur permettant d’interroger l’API “Conso Predict”
* L’agence est chargée de l’exploitation de l’application, notamment le service IT de l’agence se charge de l’administration de l’application.
* Dans une version ultérieure de l’application, il pourrait être pertinent d’implémenter une note éco2mix (ou étiquette énergie) qui dépendrait de l’électricité consommée sur le territoire. Cette note contribuerait avec d’autres indicateurs au calcul du bilan carbone. Elle serait calculée à partir de l’intensité carbone des différents composants du mix électrique (nucléaire, hydraulique, photovoltaïque, éolien terrestre).

## **b) La data**

Pour mettre en place et déployer notre modèle de Machine learning, nous avons utilisé trois sources de données. Pour le monitoring deux sources de données sont nécessaires :

| Type de données | Source | Historique disponible | Besoin |
| --- | --- | --- | --- |
| 1. Historique de consommation électrique (MW) | Odré  (RTE) | ┑De Janvier 2012 à Mai 2022  ┑Zone : Territoire national | Développement |
| 1. Données météorologiques historiques | Météo Net  (Météo France) | ┑De 2016 à 2018 ┑Zone : Nord-Ouest, Sud-Est | Développement |
| Type de données | Source | Temps réel | Besoin |
| 1. Données météorologiques prévisionnelles | Météo concept | ┑Prévision journalière sur 14 jours maximum  ┑Zone : **Bretagne** | Déploiement |
| 1. Données en temps réel de consommation électrique (MW) | Odré | ┑Données au 1/4h | Monitoring |
| Type de données | Source | Historique | Besoin |
| 1. Données météorologiques historiques | Opendatasoft | ┑Données à l’heure | Retrain |

* La consommation électrique (MW) est la variable cible du modèle. L’historique de cette donnée s’étend sur 124 mois, nous avons travaillé sur 3 années de données. Ainsi, nous écartons les années ayant connu des montées en charge de données ainsi que les 2 années de Covid (2020 et 2021).
* Les données météorologiques historiques sont les variables explicatives du modèle.
* Les données météorologiques prévisionnelles alimenteront I’API en vue du calcul de la consommation prévisionnelle. D’autre part, elles serviront à l’équipe IT de l’agence à évaluer le modèle en vue d’un potentiel remaniement.
* Dans le cadre du **MVP**, nous avons fait le choix de travailler sur une seule zone géographique pour la phase de construction du modèle : **La Bretagne**.
* Une généralisation du modèle aux autres régions devra prendre en compte les spécificités de consommation électrique et météorologique. Dans ce sens, une étude d’opportunité est nécessaire.
* La fréquence de prédiction souhaitée par le client est mensuelle. En effet, le territoire doit être en mesure de produire un reporting d’efficacité énergétique tout au long de l’année. Le MVP doit donc produire des prévisions a minima 1 fois/mois.

## Le modèle

* Nous avons utilisé le modèle à convolution dit TCN, qui a notamment fait ses preuves[[1]](#footnote-0) sur les données financières. Parmi ses avantages, la faible profondeur d’historique nécessaire pour que celui-ci soit capable d’apprendre des liens existants à l’intérieur d’une seule série de données ou même entre plusieurs séries.
* C’est un modèle à convolution dont nous avons trouvé une mise en application par Philippe Remy[[2]](#footnote-1). L’objectif de ce réseau de neurones est de prédire quotidiennement la consommation électrique à horizon de 14 jours.
* **La principale métrique d’évaluation** du modèle est la distance euclidienne entre les prédictions et les observations (le MAE en complément).
* Les métriques standards (moyenne, écart-type) sont également calculées lors de l’évaluation de la performance du modèle, puis du drift.

## Les données du modèle

Le modèle de ML nécessite des données statiques pour sa construction et des données dynamiques pour l’alimenter avec de nouvelles observations.

1. **Les données pour la construction du modèle**

* La zone géographique/région
* Mois (1 à 12)
* Jour dans le mois (1 à 31)
* Jour de la semaine (1 à 6)
* La consommation quotidienne électrique (MW) totale
* Le mix = consommation quotidienne électrique (MW) par filière de production : thermique, éolien, solaire, nucléaire, hydraulique
* Température quotidienne régionale
* Taux d’humidité
* Précipitations
* La pression
* La force du vent
* La direction du vent
* La température de rosée
* Consommation électrique :

<https://odre.opendatasoft.com/explore/dataset/eco2mix-regional-tr/table/?disjunctive.libelle_region&disjunctive.nature&sort=-date_heure>

* Température / humidité/ précipitations/ pression/ force du vent :

<https://meteonet.umr-cnrm.fr/>

<https://odre.opendatasoft.com/explore/dataset/temperature-quotidienne-regionale/table/?disjunctive.region&sort=date>

1. **Les données d’alimentation de “Conso Predict”**

Les données source de “Conso Predict” proviendront des données de l’API “Météo Concept”, qui à leur tour sont collectées dans différents types de stations : les stations officielles en libre accès de Météo France (réseau SYNOP), les stations de Météo Concept et les stations des partenaires et clients, qui acceptent de distribuer leurs données en Open Data.

Pour chaque station (commune), Météo Concept permet de récupérer les indicateurs prévisionnels journaliers sur les 14 prochains jours.

## API

Dans le cadre du MVP, nous utiliserons l’interface native de l’API (swagger) avec les endpoints suivants :

* Création d’un utilisateur
* Authentification et génération d’un token
* Requêtage (sur la base de données de l’application) selon les critères suivants :
  + Date de début et date de fin des prédictions souhaitées, qui devront être entre J et J+14
  + Localité
  + Version du modèle

## Batch

Dans le module de l’application que nous avons appelé “batch”, nous gérons à la fois :

* les extractions/prédictions récurrentes permettant d’alimenter la base de données sur laquelle se source l’API
  + Une série de scripts Python permet d’extraire les données météo, de les préprocesser, d’effectuer les prédictions et de les stocker en base de données
* le monitoring
  + Une autre série de scripts Python permet d’effectuer les tâches décrites dans le paragraphe suivant

## La base de données

Nous utilisons une base de données mySQL, accessible avec l’interface PHPMyadmin pour les besoins d’administration, de consultation des logs de connexion à l’API …

## La conteneurisation

L’application est totalement “conteneurisée” avec Docker, ce qui permet notamment une installation ultra-simple, même sur une VM vierge.

Sur le conteneur “batch”, sont montés des volumes “Data” et “Model” permettant de persister sur la VM les datas générées et les nouvelles versions de modèle.

Sur le conteneur “db” est monté un volume “mysql-data” permettant de persister la base de données sur la VM.

Tous les conteneurs accèdent à la base de données pour des besoins d’écriture ou de lecture.

## Le CI/CD

Un processus d’intégration continue est mis en place à travers une Github Action, qui exécute une série de tests unitaires avec Pytest puis reconstruit les images “api” et “batch” et les pousse sur Dockerhub.

N’ayant pas véritablement d’environnement de Production, nous n’avons pas mis en place de processus de déploiement continu. Mais nous avons régulièrement vérifié qu’une installation “from scratch” (sur une VM vierge) fonctionnait correctement.

## Testing & Monitoring

**Le testing :**

Un script de tests unitaires est utilisé pour vérifier tous les endpoints de l’API “Conso Predict” avec toutes les possibilités de cas passants et non passants.

Nous faisons le choix de lancer ce script manuellement lorsque l’application tourne.

Il aurait été possible de jouer ces tests automatiquement dans une Github action, mais nous avons jugé que cela aurait été trop lourd, en effet, cela aurait nécessité de déployer toute l’application (API+BDD) dans l’environnement Github action et d’insérer des jeux de données de test dans la BDD puisque le déploiement aurait créé la BBD à vide.

**Le monitoring :**

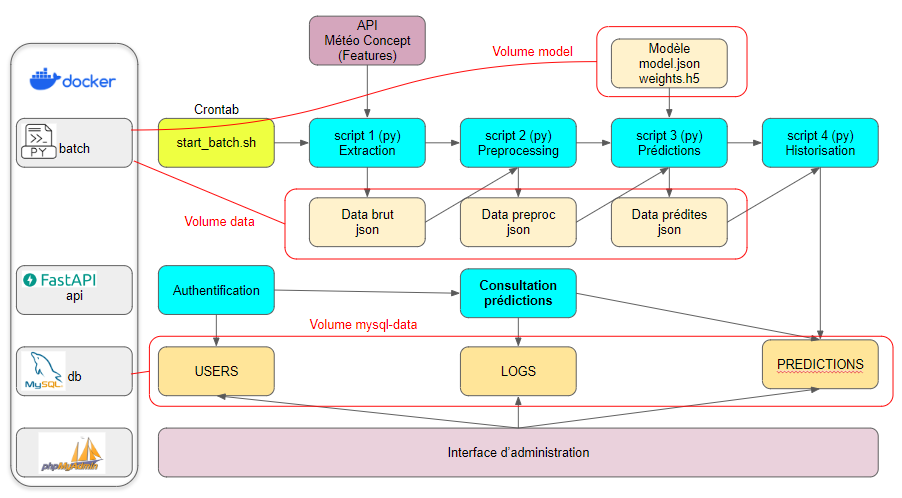
Les données de consommation électrique sont mises à jour à deux occasions. Elles sont vérifiées et complétées en milieu de M+1 : ce sont les données **consolidées**. Elles deviennent **définitives** au deuxième trimestre de A+1 lorsque tous les partenaires ont transmis et vérifié l'ensemble des comptages.

La stratégie de monitoring choisie consiste à :

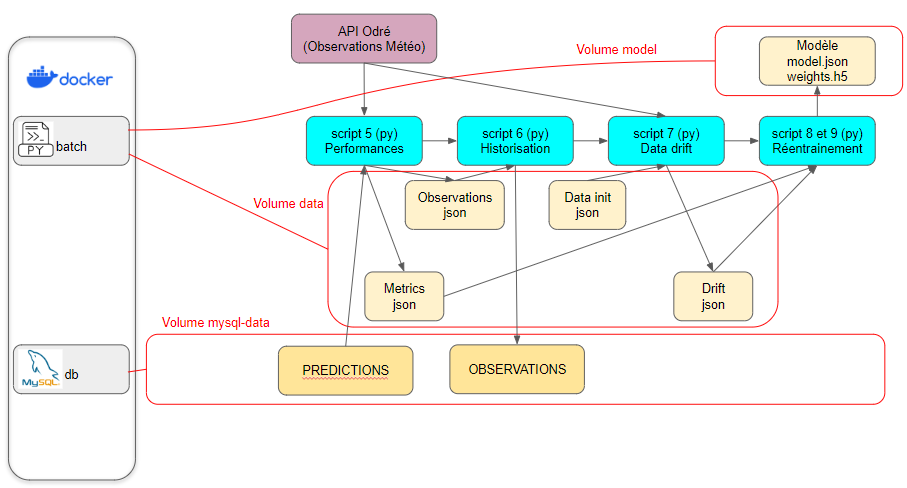
* Evaluer la performance du modèle et éventuellement le ré-entraîner :
  + 1ère évaluation :15 jours après la dernière prédiction, les données observées en temps réel sont comparées aux données prédites, évaluées, puis stockées.
  + 2ème évaluation : 40 jours après la première prédiction nous aurons constitué 2 à 3 cohortes de prédiction. Nous lançons une vérification du « modèle drift » entre les données consolidées et les données de développement du modèle (moyenne, écart-type, distance euclidienne (%)).
  + 3ème évaluation : 90 jours après la première prédiction nous aurons constitué 6 cohortes environ. Les indicateurs de performance et de drift sont calculés sur les 6 dernières cohortes. Si le seuil d’alerte est dépassé alors le ré-entraînement est lancé, aucune action n’est menée sinon.
  + 4ème évaluation : Tous les ans lorsque les données consolidées définitives sont disponibles, le datascientist doit intervenir pour vérifier la qualité des nouvelles données d’une part, et la qualité des prédictions d’autre part.
* Mener des actions à chaque évaluation :
  + 1ère évaluation : un mail de notification est envoyé à l’administrateur lorsque les données attendues sont disponibles.
  + 2ème évaluation : un mail de notification est envoyé à l’administrateur lorsque les données attendues sont disponibles. Si la performance du modèle lors de la 1ère évaluation est inférieure à 60% ou que la performance est inférieure à 70% lors de la 3ème évaluation, alors un ré-entraînement est déclenché puis stocké.
  + 3ème évaluation : L’application est bloquée et un mail d’alerte est envoyé à l’administrateur. Il sera alors chargé de réétudier le modèle ML avec l’équipe data pour trouver d’autres solutions.

## Schéma d’implémentation

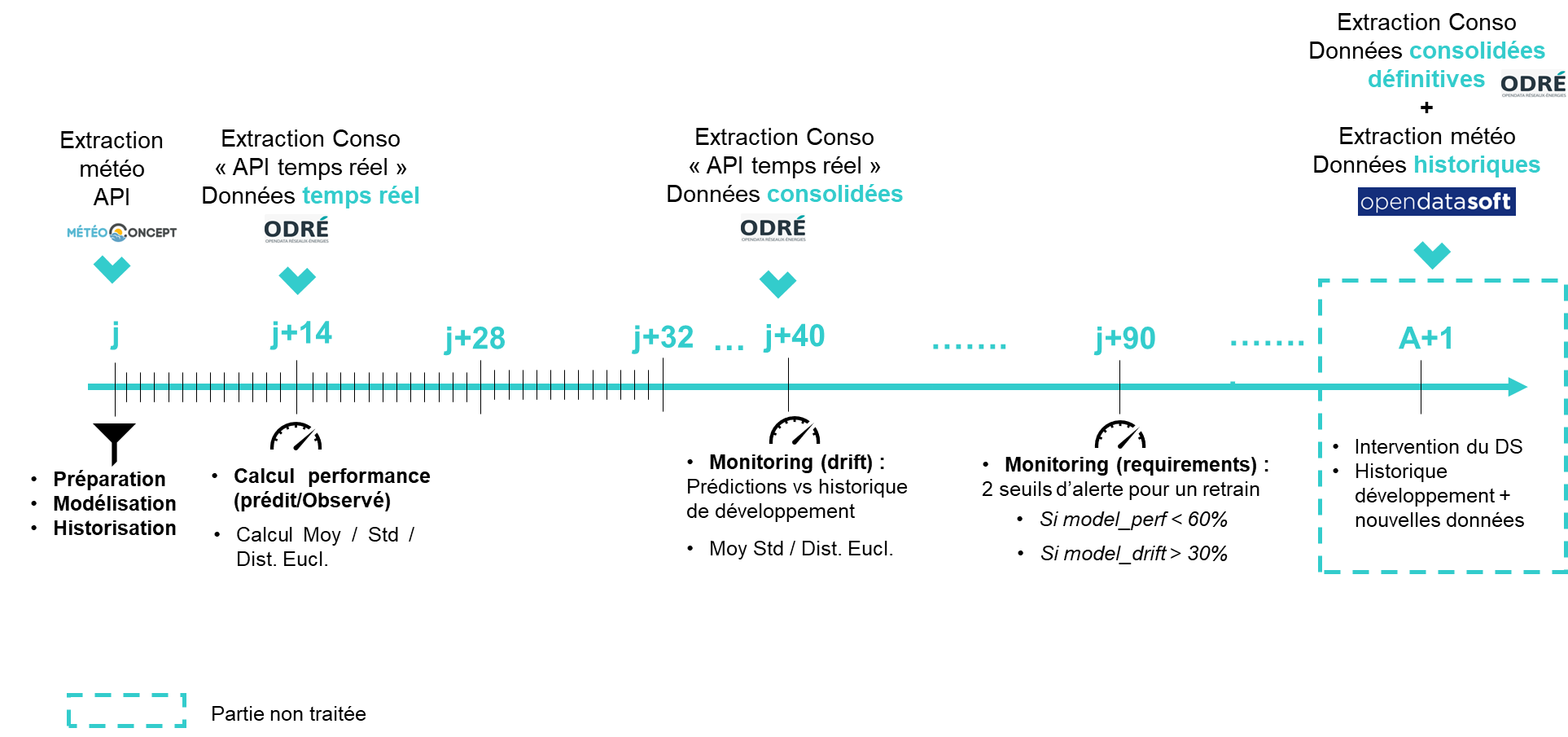
**Cas d’usage : consultation/administration**



**Monitoring**



**Cheminement des données (développement + monitoring)**



1. https://www.mdpi.com/2076-3417/10/7/2322 [↑](#footnote-ref-0)
2. https://github.com/philipperemy/keras-tcn/tree/master [↑](#footnote-ref-1)