Livrable Projet Festival

Réalisé par

BOUGEROLLES Jean

CORIN Gaëtan

DOUSSIET Lou

MARSILLE Antoine

VIDOR Fabien

[Introduction 2](#_2upj0kc12l5q)

[**Le diagramme global annoté. 3**](#_100aktfv8vxb)

[**Une présentation synthétique des technologies comparées. 4**](#_m08aul8a7gpa)

[**Un argumentaire structuré sur les choix technologiques. 5**](#_w2ejh030di8g)

[Python : 5](#_my18g4bunxnz)

[Apache SPARK : 5](#_g1wp7wybly2m)

[Hadoop HDFS: 5](#_oxhxjeo7nzee)

[Neo4J : 5](#_skn7c2c8idfx)

[Spark Structured Streaming: 6](#_1r27rziqs520)

[Kotlin : 6](#_mcdozfeqzpva)

[**Une évaluation critique des limites et axes d’évolution possibles. 7**](#_eaqj82i8aydb)

[**Code sources 7**](#_7hq70wl0xf6b)

[Phase de Proof Of Concept 7](#_s3mur0fuhm4m)

[Données utilisés 7](#_unbvesyk46ze)

[Étapes d’Extraction et de Transformation 8](#_eu3q7xh8cl24)

Conclusion

#### 

#### **Introduction**

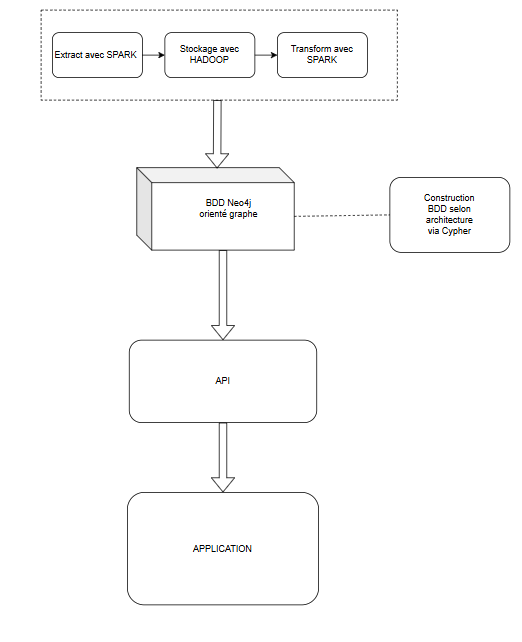
Le contexte du projet est d’allier Open Data et l’architecture distribuée pour créer une application mobile afin d’aider les participants du Festival d’Avignon tout au long des évènements prévus.

L’objectif est de récupérer les données sur le programme du festival, les emplacements des parkings et tout autres données qui pourraient ajouter des fonctionnalités pratiques pour les utilisateurs de l’application.

Dans le cadre du cours d’architecture distribuée, il nous est demandé d’imaginer une solution qui soit adaptée à la volumétrie des données et leur hétérogénéité et en temps réel.

## 

## Le diagramme global annoté.



## Une présentation synthétique des technologies comparées.

Dans le cadre de notre projet pour le festival, nous avons étudié et comparé plusieurs technologies distribuées afin de répondre aux exigences de traitement de données massives, de temps réel et de mobilité.

Pour la base de données, MongoDB, Cassandra et Neo4J ont été envisagées. MongoDB est très adapté aux données semi-structurées grâce à sa souplesse de modélisation en JSON et sa facilité de mise en œuvre. Cassandra offre une haute scalabilité et disponibilité, ce qui le rend idéal pour les applications nécessitant une gestion massive de données en temps réel. Neo4J repose sur un modèle graphe qui permet de représenter facilement les connexions entre les entités du système, ce qui peut s’avérer pertinent pour des optimisations de parcours. Son langage de requête cypher permet d’exprimer efficacement des parcours et des contraintes relationnelles.

Concernant le traitement des données, nous avons comparé Spark et Hadoop MapReduce. MapReduce est robuste et éprouvé pour le traitement de très gros volume de données mais est relativement lent car il est basé sur des lectures/écritures disque. Spark offre une rapidité accrue grâce à son traitement en mémoire, et supporte divers modules pour le traitement de flux, le SQL, ou encore le machine learning.

L’orchestration des traitements est également un point central dans une architecture distribuée. Apache Airflow est souvent utilisé pour sa capacité à gérer des pipelines complexes de manière déclarative en définissant les dépendances entre les tâches. Spark Structured Streaming se présente comme une solution hybride, capable d’assurer à la fois le traitement et l’orchestration des flux de données en continu. Il permet de définir des transformations de données comme des requêtes déclaratives, exécutées de façon incrémentale au fil des arrivées des données. Cela en fait un outil particulièrement efficace pour orchestrer des pipelines de streaming complexes sans avoir recours à un orchestrateur externe.

Le développement natif, avec Kotlin pour Android et Swift pour iOS, permet de tirer pleinement parti des capacités de chaque système d’exploitation. Ces langages donnent accès aux API les plus avancées, garantissent les meilleures performances, et assurent une intégration fluide avec l’environnement natif (notifications, capteurs, navigation, etc.). Toutefois, cette approche implique de maintenir deux bases de code distinctes, doublant ainsi les efforts de développement et de maintenance.

## 

## 

## 

## Un argumentaire structuré sur les choix technologiques.

Voici les choix technologiques que nous avons décidés d’utiliser ainsi que leurs raisons argumentés:

#### **Python :**

Le choix de Python s’est fait automatiquement, par souci d'efficacité et de rapidité, en effet chacun des collaborateurs du groupe sait utiliser Python et sait où aller chercher des informations et de la documentation.

Le rôle de Python dans ce POC est de tester comment récupérer les données, de les visualiser, de les transformer.

#### **Apache SPARK** :

Pour traiter et manipuler les données dans l’objectif de passer d’un POC a un projet industrialisable, nous avons fait le choix d’utiliser Spark.

Son rôle va être d’extraire les données, d’appliquer des transformations, de faire des jointures…

Le choix de Spark a été privilégié par rapport à d’autres frameworks de traitement distribué car en comparaison à des frameworks semblable à Hadoop MapReduce qui réalise des lectures/écritures dans le disque dur, Spark réalise ces traitements directement en mémoire (in mémory computing), ce qui permet de réaliser des traitements beaucoup plus rapidement. Il permet aussi de conserver les données en cache, ce qui permet une relecture rapide.

En ayant une vision d'expansion de l’application, la rapidité et l’orchestration de différentes étapes de traitement avec mise en cache sont des critères essentiels qui nous ont guidé dans notre choix..

#### **Hadoop HDFS:**

Pour stocker les données brutes, on utilise Hadoop HDFS, c’est un système de fichier distribué qui permet de gérer de grandes quantités de données et de fonctionner sur du matériel standard.

L'intérêt de stocker les données via HDFS est de sécuriser la données : grâce au réplica, si un nœud tombe, les données restent disponibles, sans intervention manuelle.

#### **Neo4J** :

Au vu des différentes entités de données que nous allons créer, et des besoins métiers qui sont demandés, nous allons avoir de nombreuses relations au sein de notre bdd.

Neo4J est spécialement conçu pour faire des recherches sur de nombreuses données grâce à son principe de création de table par Index pour une recherche rapide. De plus, il performe sur des query avec liaisons complexes, ce qui nous pousse à penser que c’est le logiciel idéal pour une potentielle expansion du nombre d’entités.  
Il s’agit aussi d’une base en cluster avec des leaders et followers, ce qui permet d’obtenir une scalabilité déjà implémenter pour un potentiel besoin futur.

En terme de CAP de système distribué, neo4J est considéré CP.

C- Cohérence: Tous les nœuds primaires (leaders et followers) suivent un ordre strict d’écriture, ce qui crée une cohérence accrue à travers les nœuds.

P - Tolérance au partitionnement: Lors d’une détection de partition, neo4j bloque les écritures sur les nœuds hors quorum(les nœuds isolés). Ces nœuds isolés, une fois qu’ils peuvent de nouveaux communiquer avec le cluster principal, vont ensuite se re-synchroniser avec le reste du cluster.

En revanche, la A- Accessibilité n’est pas constante, car durant un partitionnement, certaines écritures peuvent être bloquées.

#### **Spark Structured Streaming:**

Spark Structured Streaming est la brique temps réel de Spark : un moteur distribué qui ingère des flux continus, les transforme avec la même API DataFrame/SQL que le batch et diffuse les résultats quasi instantanément, tout en assurant scalabilité et « exactly-once ».

C’est un moteur de traitement de flux évolutif et tolérant aux pannes basé sur le moteur Spark SQL.

Nous l’avons choisi parce qu'il fait partie des services natifs de Sprak, il garde la trace de ce qu’il a déjà traité ; si un nœud tombe, il redémarre sans perdre ni dupliquer les messages.

Spark Structured Streaming est pratique pour du temps réel « quelques secondes » sans multiplier les technologies et utiliser un seul cluster pour tout faire.

#### **Kotlin** :

Kotlin est le langage choisi pour réaliser l’application car c’est LE langage recommandé par Android. En prenant en compte que 72% de la population utilise un téléphone fonctionnant sur android, cela assure une maintenabilité et une mise à jour récurrente pour fonctionner sur le téléphone des utilisateurs.  
De plus, Kotlin est un langage moins verbeux que Java, ce qui simplifie le développement. Sa popularité lui permet d’être facilement maintenable.  
Il s’agit d’un langage rapide, et qui contient moins de bug natif que des langages comme React Natif par exemple.

## Une évaluation critique des limites et axes d’évolution possibles.

1. La collecte des données pourrait être améliorée. En contactant l’organisation du festival pour avoir l’url du programme directement, nous pourrions garantir la disponibilité et la fiabilité des données.
2. Il n’y a pas de système de supervision, si un élément casse nous ne sommes pas alertés. La mise en place d’un outil de supervision tel que Grafana peut pallier ce problème.
3. Les traitements de données effectués ne sont pas entièrement vérifiés. Il peut être judicieux de mettre en place un système de tests unitaires.

## Code source

#### Phase de Proof Of Concept

Lors du développement, la première partie a été concentrée sur la partie ETL avec l’Extract, le Transform et le Load. Une version POC est réalisée en notebook Python. Il s’agit de la version de développement avant la phase d’industrialisation, afin de prouver la faisabilité et viabilité du projet et des données sources

Il y a plusieurs sites d’extractions sur lesquels extraire les données concernés, qui sont tous disponible en Opendata

#### Données utilisés

- Le premier site d’extraction est le site internet **Datasud**. Il s’agit d’une plateforme qui permet d’explorer les données ouvertes en Provence-Alpes-Côte d’Azur. Elle contient les données **des parkings, les stationnements de vélo, et les stationnement de personnes à mobilité réduite** d'Avignon. L’objectif étant de récupérer ces données en réalisant des extractions sous format xls.

Les parkings : <https://www.datasud.fr/explorer/fr/jeux-de-donnees/avignon-equipements-parkings/donnees>

Stationnement vélo : <https://www.datasud.fr/explorer/fr/jeux-de-donnees/avignon-stationnement-pour-les-velos-arceaux/info>

Stationnement pers mob réduite : <https://www.datasud.fr/explorer/fr/jeux-de-donnees/avignon-stationnement-pour-personnes-a-mobilite-reduite-pmr/info>

- Le deuxième site est le **PDF du festival 2025 d’Avignon**. Ce PDF contient l’ensemble des spectacles du festival, leur lieux ainsi que les dates de représentation sous forme d’un tableau.

L’objectif étant de récupérer les informations du PDF, puis d’en faire un CSV propre et utilisable.

spectacles du festival: <https://festival-avignon.com/fr/telechargements/programme_FDA_79>

- Le troisième site d’extraction est le **site officiel du festival** [**festival-avignon.com**](http://festival-avignon.com)Le site officiel permet de récupérer les différents **lieux de festivité d’Avignon et leurs emplacements**, en récupérant les coordonnées de chacun de ces sites

carte des lieux du festival: <https://festival-avignon.com/fr/edition-2025/programmation/par-lieu>

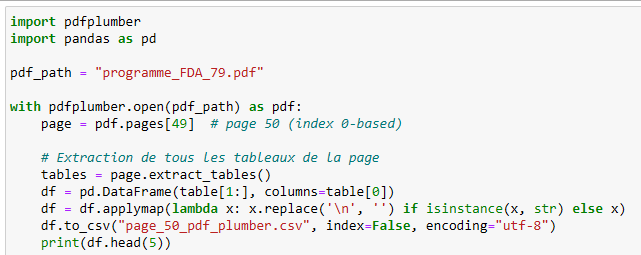
#### Étapes d’Extraction et de Transformation

Voici les étapes d'Extraction et de Transformation en version POC afin de tester la faisabilité et viabilité de l’ETL futur industrialisé.

Pour commencer, on récupère les **parkings, les stationnements de vélo, et les stationnement de personnes à mobilité réduite** d'Avignon:



Le chargement des parkings et stationnements de vélo se fait simplement par extraction python.  
Les PMR, en revanche, sont issus d’une carte ou les données sont générées dynamiquement en javascript, puis chargées par le navigateur en csv. Il faudra donc réaliser un bot de scrapping qui va cliquer sur le bouton pour réaliser le téléchargement des données.

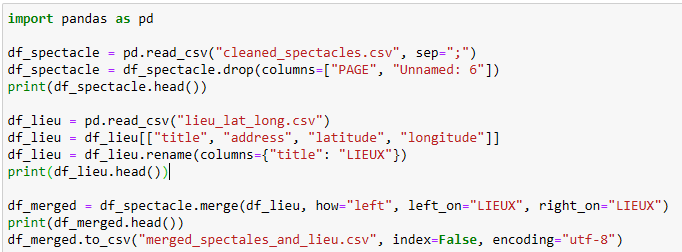
Ensuite, on récupère **l’ensemble des spectacles du festival** en utilisant le **PDF du festival**:  


Une fois la récupération des données en csv, nous sommes obligés de retravailler légèrement les différents titres de spectacles qui malheureusement ont tendance à s'effacer pour certains d’entre eux, causé par la manière de comment le PDF a été créé.

Enfin, on récupère les différents **lieux de festivité d’Avignon** et leurs emplacements en utilisant les données directement dans le html du site



Il nous reste désormais a **merger les noms des lieux** du PDF extrait en csv avec les noms des lieux de la carte et de leurs emplacements



Nous avons désormais l’ensemble des données préparées contenant pour chacun une longitude et une latitude.  
  
L’étape suivante consiste à réaliser le Chargement de l’ETL en récupérant les csv et xls formaté précédemment, puis en les chargeant sur Neo4J.

Conclusion :

Sur ce projet, la gestion du temps était essentielle, nous avons donc choisi certaines solutions pour arriver à un résultat au plus vite.

Cependant certaines évolutions du projet ont été étudiées comme passer la partie ETL en architecture distribuées en utilisant Spark et Hadoop. Intégrer des données temps réel comme les temps d’attente sur chaque spectacle ou la gestion des places restantes sur les parkings grâce à Spark Structured Streaming.