# Πρότυπο αναφοράς άσκησης

Συστήματα Διαχείρισης Δεδομένων Μεγάλου Όγκου

Εργαστηριακή Άσκηση 2023/24

|  |  |  |
| --- | --- | --- |
| **Όνομα** | **Επώνυμο** | **ΑΜ** |
| Πανταζή | Ηλιάνα | 1072642 |
| Πουργουρίδης | Ορφέας | 1069664 |

Βεβαιώνω ότι είμαι συγγραφέας της παρούσας εργασίας και ότι έχω αναφέρει ή παραπέμψει σε αυτήν, ρητά και συγκεκριμένα, όλες τις πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, προτάσεων ή λέξεων, είτε αυτές μεταφέρονται επακριβώς (στο πρωτότυπο ή μεταφρασμένες) είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για το συγκεκριμένο μάθημα/σεμινάριο/πρόγραμμα σπουδών.

Έχω ενημερωθεί ότι σύμφωνα με τον εσωτερικό κανονισμό λειτουργίας του Πανεπιστημίου Πατρών άρθρο 50§6, τυχόν προσπάθεια αντιγραφής ή εν γένει φαλκίδευσης της εξεταστικής και εκπαιδευτικής διαδικασίας από οιονδήποτε εξεταζόμενο, πέραν του μηδενισμού, συνιστά βαρύ πειθαρχικό παράπτωμα.

|  |  |
| --- | --- |
| Υπογραφή  Πανταζή Ηλιάνα  20 / 09 / 2024 | Υπογραφή  Πουργουρίδης Ορφέας  20 / 09 / 2024 |

# Συνημμένα αρχεία κώδικα

Μαζί με την παρούσα αναφορά υποβάλλουμε τα παρακάτω αρχεία κώδικα

|  |  |  |
| --- | --- | --- |
| Αρχείο | Αφορά το ερώτημα | Περιγραφή/Σχόλιο |
| 1. **producre.py** 2. **consumer.py** | **1** | * **producer: αφορά το υποερώτημα 4 (αποστολή δεοδμένων)** * **consumer: αφορά το υποερώτημα 5 (παραλαβή δεδομένων)** |
| 1. **producer.py** 2. **sparkjob.py** | **2** | * **producer: από εκεί λαμβάνει η διεργασία μας τα δεοδμένα** * **sparkjob: υλοποίηση διεργασίας για παραλαβή και επεξεργασία των raw data** |
| 1. **sparkjob.py** 2. **mongoqueries.py** | **3** | * **sparkjob: περιλαμβάνει και την δημιουργία κατάλληλων συλλογών για τα δεδομένα** * **mongoqueries: υλοποίηση queries** |

**\*\*\* Επισυνάπτεται ένα επιπλέον αρχείο το οποίο αφορά το τι εντολή πρέπει να τρέξουμε στην terminal έτσι ώστε να κυλήσουν όλα ομαλά: commands\_for\_servers.txt**

# Τεχνικά χαρακτηριστικά περιβάλλοντος λειτουργίας

*[Τεχνικά χαρακτηριστικά φυσικού Η/Υ που χρησιμοποιήθηκε για την εργασία, αν χρησιμοποιήθηκε hosted υπηρεσία μπορείτε απλά να αναφέρετε αυτό αντί για τον πίνακα]*

|  |  |
| --- | --- |
| **Χαρακτηριστικό** | **Τιμή** |
| CPU model | ***Intel® Core™ i7-8550U*** |
| CPU clock speed | ***1.8GHz*** |
| Physical CPU cores | ***4*** |
| Logical CPU cores | ***8*** |
| RAM | ***16*** |
| Secondary Storage Type | ***SSD*** |

# Ερώτημα 1: Παραγωγή δεδομένων

1. *Ο εξομοιωτής λειτουργεί κανονικά στο περιβάλλον μας.*
2. *Υλοποιήθηκε Kafka Broker ακολουθώντας τις οδηγίες.*
3. *Δημιουργήθηκε topic ”vehicle\_positions” στο οποίο θα στέλνονται τα δεδομένα το εξομοιωτή.*
4. *Από το αρχείο producer.py:*

*Λογική: μέσα στο αντικείμενο κλάσης KafkaProducer εισαγάγαμε το κομμάτι του εξομοιωτή το οποίο παράγει τα δεδομένα ου χρειαζόμαστε (περιλαμβάνουμε και την εκτύπωσή τους για επιβεβαίωση) στην συνέχεια δημιουργήσαμε την δομή με την οποία θέλουμε ο καταναλωτής να λαμβάνει τα δεδομένα μέσω dataframe. Επιπλέον αποθηκεύουμε τα ληφθέντα δεδομένα σε ένα .json αρχείο έτσι ώστε να βεβαιωθούμε πως η επεξργασία έγινε σωστά. Στην συνέχεια στέλνουμε τα δεδομένα στον καταναλωτή.*

1. *Από το αρχείο consumer.py:*

*Μέσω του τελευταίας εντολής print βεβαιωνόμαστε πως τα δεδομένα έχουν ληφθεί σωστά.*

***Screenshots:***

***Λειτουργία Kafka:***

* *Zookeeper:*

*A screenshot of a computer screen

Description automatically generated*

* *Kafka:*

*A screenshot of a computer

Description automatically generated*

***Python Scripts:***

* *Producer Script:*

*A screenshot of a computer

Description automatically generated*

* *Consumer Script:*

*A screenshot of a computer

Description automatically generated*

# Ερώτημα 2: Κατανάλωση και επεξεργασία με Spark

Όπως και την περίπτωση του Kafka με στόχο να παραλάβουμε τα δεδομένα μας με την σωστή δομή δημιουργήσαμε ένα “template”. Στην αρχή το template αφορά τα δεδομένα όπως στέλνονται απευθείας από το Kafka ενώ στην συνέχεια με τα κατάλληλα processing και aggregations καταλήξαμε στο τελικό αποτέλεσμα που φαίνεται παρακάτω:

**\*\*\*Σχόλιο: Για την σωστή ροή της υλοποίησης πρώτα τρέχουμε το script του producer (producer.py) και στη συνέχεια τρέχουμε το script της διεργασίας (sparkjob.py) ως εξής:**

***Screenshots:***

***A black screen with white numbers

Description automatically generated***

# Ερώτημα 3: Αποθήκευση σε MongoDB

1. *Εγκαταστάθηκε η MongoDB.*
2. *Ενσωματώθηκε ο driver.*
3. *Στο ίδιο αρχείο με την υλοποίηση του προηγούμενου ερωτήματος υλοποιήθηκε και η δημιουργία της Mongo Database με τα κατάλληλα collections για τα raw και τα processes data αντίστοιχα (collection: data = raw, collection: processed = processed)*

*Παρακάτω παρουσιάζονται Screenshots από το MongoDb Compass, το οποίο αποτελεί interface για ευκολότερη διαχείριση της βάσης μας, μετά την κλήση της sparkjob.py διεργασίας:*

***Screenshots:***

* ***Collection data:***

*A screenshot of a computer

Description automatically generated*

* ***Collection processed:***

*A screenshot of a computer code

Description automatically generated*

* ***Παραδείγματα υλοποίησης των Queries:***

***A black screen with white text

Description automatically generated***

***A black background with white text

Description automatically generated***

***A black background with white text

Description automatically generated***

**\*\*\*Σημείωση: Όλη η εργασία υλοποιήθηκε σε περιβάλλον Linux Ubuntu μέσω WSL.**

# Σχολιασμός αποτελεσμάτων

# Βιβλιογραφία

*Πέραν των link που μας προτάθηκαν από την εκφώνηση για τυχόν errors που προκύπταν κατά την διαδικασία υλοποίησης οποιουδήποτε task βασιστήκαμε κυρίως στο Stack Overflow (https://stackoverflow.com) και στα official documentations των αντίστοιχων εργαλείων που χειριζόμασταν.*