ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΗΓΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΠΛΗΡΟΦΟΡΙΚΗΣ - ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ ΚΑΙ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ

ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΟΥΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ (JAVA)

ΑΝΑΦΟΡΑ ΤΕΛΙΚΗΣ ΕΡΓΑΣΙΑΣ

ΟΜΑΔΑ ΑΝΑΠΤΥΞΗΣ:

- ΝΙΚΟΛΑΣ ΠΑΤΕΡΑΣ – **ΜΠΣΠ21043**
- ΒΑΣΙΛΕΙΟΣ ΖΑΡΤΗΛΑΣ ΠΑΠΑΧΑΡΑΛΑΜΠΟΥΣ – **ΜΠΣΠ21015**

Πειραιάς, Μάρτιος 2022

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΕΧΟΜΕΝΑ 2
1. ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ 3
2. ΤΕΚΜΗΡΙΩΣΗ ΚΩΔΙΚΑ 4
2.1. ΒΑΣΙΚΑ 4
2.1.1. ΦΑΚΕΛΟΣ RESOURCES 4
2.1.2. ΚΛΑΣΕΙΣ ΜΟΝΤΕΛΩΝ 4
2.1.3. ΚΛΑΣΗ Constants 4
2.1.4. ΚΛΑΣΗ Color 5
2.1.5. ΚΛΑΣΗ Main 5
2.1.6. ΚΛΑΣΗ UserInput 5
2.2. ΦΑΚΕΛΟΣ DATABASE 5
2.2.1. ΚΛΑΣΗ DBManager 6
2.2.2. ΚΛΑΣΗ GetFromAPI 6
2.2.3. ΚΛΑΣΗ GetFromDataset 7
2.2.4. ΚΛΑΣΗ GetFromDatabase 8
2.3. BLOCKCHAIN 9
2.3.1. ΚΛΑΣΗ Block 9
2.3.2. ΚΛΑΣΗ Blockchain 10
2.4. ΚΑΘΑΡΙΣΜΟΣ DATASET 11
3. ΒΙΒΛΙΟΓΡΑΦΙΑ ΚΑΙ ΠΗΓΕΣ 12
4. ΒΙΒΛΙΟΘΗΚΕΣ ΚΑΙ ΕΡΓΑΛΕΙΑ 13

1. ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ

Καταρχάς, για βάση δεδομένων χρησιμοποιήσαμε την SQLite η οποία είναι ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων.

Πλεονεκτήματα τις SQLite:

- Γρήγορη και ελαφριά.
- Τα δεδομένα μετακινούνται εύκολα και είναι προσβάσιμα από πολλές άλλες γλώσσες. Η φορητότητα των δεδομένων της είναι τεράστιο όφελος.

2. ΤΕΚΜΗΡΙΩΣΗ ΚΩΔΙΚΑ

2.1. ΒΑΣΙΚΑ

Αρχικά, χρησιμοποιείται το Maven το οποίο είναι εργαλείο αυτοματισμού κατασκευής που χρησιμοποιείται κυρίως για έργα προγράμματα Java.

Στην συνέχεια, δημιουργήθηκε η κλάση DBManager για την διαχείριση της βάσης δεδομένων και μια κλάση μοντέλου - Plain Old Java Object (POJO) «Statistic» που είναι μια κλάση που αντιπροσωπεύει ένα αντικείμενο δεδομένων που μπορεί να χρησιμοποιηθεί για τη μεταφορά δεδομένων σε εφαρμογή της Java. Ενσωματώνει την άμεση πρόσβαση σε δεδομένα στο αντικείμενο και διασφαλίζει ότι όλα τα δεδομένα στο αντικείμενο έχουν πρόσβαση μέσω των μεθόδων *get*.

2.1.1. ΦΑΚΕΛΟΣ RESOURCES

Στον συγκεκριμένο φάκελο βρίσκεται όλο το υλικό του προγράμματος, αναλυτικότερα, βρίσκεται το δεύτερο dataset που δόθηκε (https://www.ecdc.europa.eu/en/publications-data/download-todays-data-geographic-distribution-covid-19-cases-worldwide) αφού έγινε καθαρισμός πρώτα δια τον λόγο ότι υπήρχαν αχρείαστες στήλες.

2.1.2. ΚΛΑΣΕΙΣ ΜΟΝΤΕΛΩΝ

Οι κλάσεις αυτές βρίσκονται στο φάκελο «models» και όπως αναφέρεται πιο πάνω αντιπροσωπεύουν ένα αντικείμενο δεδομένων περιλαμβάνοντας getters/setters για να γρήγορη πρόσβαση τους αλλά και αρχικοποίησης μέσω των constructor τους.

2.1.3. ΚΛΑΣΗ Constants

Κλάση στην οποία αποθηκεύονται όλες οι σταθερές του προγράμματος. Αποτελεί παράδειγμα σωστού κώδικα αφού αν χρειαστεί να γίνει αλλαγή, θα αλλάξει σε όλα τα σημεία του κώδικα που θα καλείται.

2.1.4. ΚΛΑΣΗ Color

Η συγκεκριμένη κλάση είναι τύπου «enum». Το enum είναι μια ειδική «κλάση» που αντιπροσωπεύει μια ομάδα σταθερών (αμετάβλητες μεταβλητές, όπως οι τελικές μεταβλητές). Η κλάση αυτή χρησιμοποιείται για να δηλώσουμε χρωματικές αναπαραστάσεις που θα χρησιμοποιούνται για την προβολή κειμένου με χρώματα σε διάφορες περιπτώσεις όπως μηνύματα επιτυχίας ή λάθους.

2.1.5. ΚΛΑΣΗ Main

Η main() μέθοδος είναι το σημείο εισόδου οποιουδήποτε προγράμματος Java. Η σύνταξή του είναι πάντα public static void main(String[] args).

2.1.6. ΚΛΑΣΗ UserInput

Η συγκεκριμένη κλάση μπορεί να χαρακτηριστεί κι ως βοηθητική κλάση εφόσον χρησιμοποιείται για τις εισόδους του χρήστη. Τρεις σε άθροισμα μέθοδοι βρίσκονται στην κλάση. Η μέθοδος country() χρησιμοποιείται κάθε φορά που ο χρήστης πρέπει να δώσει μία χώρα προς αναζήτηση, η μέθοδος months() χρησιμοποιείται για την είσοδο των μηνών που επιθυμεί ο χρήστης να του επιστραφούν αποτελέσματα και τέλος η μέθοδος choices() χρησιμοποιείται απλά για να μπορεί ο χρήστης να «κινείται» στο πρόγραμμα και να επιλέγει ανάμεσα στις επιλογές που του δίνουνται.

2.2. ΦΑΚΕΛΟΣ DATABASE

Επειδή το API αντιμετώπιζε κάποια προβλήματα τα δεδομένα για τα μηνιαία στατιστικά τα πήραμε από ένα dataset όπου αφορούσε δεδομένα του 2020 μόνο. Έτσι τα χωρίσαμε σε δύο διαφορετικές κλάσεις για να είναι πιο καθαρός ο κώδικας μας.

2.2.1. ΚΛΑΣΗ DBManager

Στην παρούσα κλάση γίνεται όλο το backend για την διαχείριση της βάσης. Πρώτα από όλα, η κλάση χρησιμοποιεί το πρότυπο σχεδίασης του «Singleton» για να μην γίνεται άσκοπη χρήση της μνήμης για την διαχείριση της πρόσβασης σε έναν πόρο που είναι κοινόχρηστος από ολόκληρη την εφαρμογή και θα ήταν καταστροφικό να υπάρχουν δυνητικά πολλαπλές παρουσίες της ίδιας κλάσης. Η διασφάλιση της ασφαλούς πρόσβασης σε κοινόχρηστους πόρους.

Υπάρχει η συνάρτηση connect() η οποία επιστρέφει την κατάσταση της σύνδεσης στην βάση έτσι ώστε να μην χρειάζεται να την καλούμε συνέχεια κάθε φορά που θέλουμε να κάνουμε κάτι στην βάση.

Στην main καλούμε την μέθοδο initializeTables() η οποία αρχικοποιεί τους πίνακες αν δεν υπάρχουν ήδη. Να σημειωθεί ότι όλα τα insertions/updates γίνονται με prepared statements τα οποία είναι πιο ασφαλής αφού προστατεύουν την βάση μας από SQL Injection.

Τέλος, υπάρχουν η συναρτήσεις addNewEntry()/6 και dataForStats() που προσθέτουν δεδομένα στην βάση και εμφανίζουν δεδομένα αντίστοιχα.

2.2.2. ΚΛΑΣΗ GetFromAPI

Αρχικά βλέπουμε ότι υπάρχει το instance της κλάσης ConsoleProgress για να μπορέσουμε να εμφανίσουμε μήνυμα φόρτωσης στον χρήστη καθώς από πίσω τρέχει ο κώδικας για την αναζήτηση των αποτελεσμάτων που ζήτησε.

Μετέπειτα στον κύριο κώδικα παίρνουμε την παράμετρο της μεθόδου, στην μέθοδο ζητάμε να δοθεί μία μεταβλητή τύπου String, όπου είναι η χωρά η οποία θέλει ο χρήστης να του επιστρέψουμε αποτελέσματα και δημιουργούμε το URL μες το οποίο θα κάνουμε σύνδεση με το API.

Μετατρέπου τα δεδομένα που είναι σε JSON μορφή και τα αποθηκεύουμε σε ένα Map<>, να σημειώσουμε ότι δημιουργούμε και ένα δεύτερο Map για να πάρουμε μόνο το κλειδί «data» από το προηγούμενο Map, εφόσον είναι εκεί τα δεδομένα που μας ενδιαφέρουν.

Ελέγχουμε αν το κλειδί «location» είναι κενό. Εάν δεν είναι τότε η χώρα που αναζήτησε ο χρήστης υπάρχει. Μετέπειτα παίρνουμε τα δεδομένα από το «data» και τα μετατρέπου και παράλληλα τα αποθηκεύουμε σε μεταβλητές τύπου long εφόσον μιλούμε για πολύ μεγάλους αριθμού (8 ψηφίων και άνω σε κάποιες περιπτώσεις).

Τέλος, παρουσιάζουμε τα δεδομένα στην κονσόλα και τα αποθηκεύουμε στο Blockchain μέσω της DBManager -> addNewEntry.

2.2.3. ΚΛΑΣΗ GetFromDataset

Αρχικά να σημειώσουμε ότι το dataset είναι στο φάκελο dataset με όνομα covidData.json και χρησιμοποιήθηκε για την εύρεση μηνιαίων στατιστικών.

Βλέπουμε ότι γίνεται ξανά η χρήση της κλάσης ConsoleProgress για να μπορέσουμε να εμφανίσουμε και πάλι μήνυμα φόρτωσης στον χρήστη καθώς από πίσω τρέχει ο κώδικας για την αναζήτηση των αποτελεσμάτων που ζήτησε.

Στην συνέχει μετατρέπουμε το JSON αρχείο και το αποθηκεύουμε σε ένα πίνακα.

Μετά βλέπουμε ότι με κατά κάποιο τρόπο ξεδιπλώνουμε τους μήνες που έδωσε ο χρήστης για να μπορέσουμε να αναζητήσουμε τα αποτελέσματα. Ακόμη διορθώνουμε το String (την χώρα δηλαδή) πού έδωσε ο χρήστης έτσι ώστε και πάλι να μπορέσουμε να κάνουμε την αναζήτηση. Τελειώνουμε με τις εισόδους του χρήστη εφόσον μέσω ενός stream() επιστρέφουμε σε μία μεταβλητή Boolean αν υπάρχει η χώρα που ζήτησε μέσα στο dataset.

Εφόσον υπάρχει τότε με ένα for περνάμε όλο το dataset και επιλέγουμε τις εγγραφές που αντιστοιχούν στους μήνες και στην χώρα που ζήτηση ο χρήστης και υπολογίζουμε θανάτους και κρούσματα (cases).

Τέλος, παρουσιάζουμε τα δεδομένα στην κονσόλα και τα αποθηκεύουμε στο Blockchain μέσω της DBManager -> addNewEntry.

2.2.4. ΚΛΑΣΗ GetFromDatabase

Στην κλάση αυτή παίρνουμε τα δεδομένα που έχουν αποθηκευτεί στην βάση μας και μόνο έτσι ώστε να υπολογίσουμε στατιστικά και να την εμφανίσουμε στην κονσόλα.

Υπολογίζονται και εμφανίζονται τα εξής:

1. Χώρες που αναζητήθηκαν με αλφαβητική σειρά.
2. Θάνατοι και Υποθέσεις ανά χώρα.
3. Συνολικοί θάνατοι και υποθέσεις για όλες τις χώρες μαζί.
4. Χώρα με τους περισσότερους και λιγότερους θανάτους.
5. Χώρα με τις περισσότερες και λιγότερες υποθέσεις.
6. Μέσος όρος θανάτων και υποθέσεων.

Σε αυτήν την κλάση γίνεται εκτεταμένη χρήση του stream(). Στην πραγματικότητα χρησιμοποιήθηκε για κάθε υπολογισμό. Μετά από κάθε υπολογισμό εμφανίζουμε στον χρήστη στα αποτελέσματα.

Να σημειώσουμε ότι η χρήση της μεθόδου containsCaseInsensitive()/1 γίνεται για να μας βοηθήσει να αποφεύγουμε την διπλοτυπία στους υπολογισμού εφόσον αν ο χρήστης αναζητήσει προηγούμενος δύο ή περισσότερες φορές για μια χώρα τότε θα αποθηκευτεί κάθε αναζήτηση σου. Στην ουσία αποθηκεύουμε σε μία λίστα κάθε φόρα την χώρα για την οποία έχουμε καταγράψει τα στατιστικά της και με την βοήθεια αυτής της μεθόδου ελέγχουμε αν υπάρχει στην λίστα ή όχι και πράττουμε ανάλογα.

2.3. BLOCKCHAIN

2.3.1. ΚΛΑΣΗ Block

Οι παρακάτω λειτουργίες θα χρησιμοποιηθούν στην υλοποίηση του Block.

A. Δημιουργία μπλοκ: Για την δημιουργία ενός μπλοκ, υλοποιείται μια κλάση POJO «Block». Στο μπλοκ της τάξης:

1. Το «*hash*» θα περιέχει τον κατακερματισμό του μπλοκ και,
2. Το «*previousHash*» θα περιέχει τον κατακερματισμό του προηγούμενου μπλοκ.
3. Τα «data» χρησιμοποιούνται για την αποθήκευση των δεδομένων του μπλοκ και,
4. Το «*timeStamp*» που θα είναι τύπου Long χρησιμοποιείται για την αποθήκευση της χρονικής σήμανσης του μπλοκ. Εδώ ο τύπος δεδομένων μεγάλου μήκους χρησιμοποιείται για την αποθήκευση του αριθμού των χιλιοστών του δευτερολέπτου.
5. Η συνάρτηση «calculateHash()» για τη δημιουργία του κατακερματισμού.
6. Η συνάρτηση «mineBlock()/1» που παίρνει σαν όρισμα το *prefix* που θέλουμε να βρούμε και κάνει εξόρυξη το μπλοκ.

B. Υπολογισμός του hash ενός μπλοκ με τον αλγόριθμο SHA-256:

1. Αρχικά, συνενώνουμε διαφορετικά μέρη του μπλοκ για να δημιουργήσουμε έναν κατακερματισμό.
2. Στη συνέχεια, λαμβάνουμε μια παρουσία της συνάρτησης κατακερματισμού SHA-256 από το *MessageDigest*.
3. Επομένως, δημιουργούμε την τιμή κατακερματισμού των δεδομένων εισόδου μας, η οποία είναι ένας πίνακας byte.
4. Τέλος, μετατρέπουμε τον πίνακα byte σε δεκαεξαδική συμβολοσειρά, ένας κατακερματισμός τυπικά αντιπροσωπεύεται ως 32-ψήφιος δεκαεξαδικός αριθμός.

C. Η εξόρυξη ενός μπλοκ σημαίνει την επίλυση μιας υπολογιστικά πολύπλοκης εργασίας για το μπλοκ, (ξεκινάμε με την προεπιλεγμένη τιμή του *nonce* και την αυξάνουμε κατά ένα):

1. Ξεκινάμε ορίζοντας το *prefix* που θέλουμε να βρούμε.
2. Στη συνέχεια, ελέγχουμε αν βρήκαμε τη λύση.
3. Αν όχι, αυξάνουμε το *nonce* και υπολογίζουμε τον κατακερματισμό σε βρόχο.
4. Ο βρόχος συνεχίζεται μέχρι να πετύχουμε το τζάκποτ.

2.3.2. ΚΛΑΣΗ Blockchain

Στην παρούσα κλάση, υπάρχει η συνάρτηση addBlock()/1 η οποία παίρνει ένα String σε Json και ελέγχει ένα υπάρχουν στοιχεία στον πίνακα blocklist ο οποίος είναι τύπου «Block» και στην συνέχεια εάν δεν υπάρχουν, θα δημιουργήσει ένα πρώτο στοιχείο όπου θα καθορίζει μια τυχαία «ψηφιακή υπογραφή», αλλιώς, θα παίρνει το hash του προτελευταίου στοιχείου του πίνακα.

2.4. ΚΑΘΑΡΙΣΜΟΣ DATASET

Έχοντας κατεβάσει το αρχείο JSON από την ιστοσελίδα που μας δόθηκε παρατηρήσαμε ότι έχει αρκετή αχρείαστη πληροφορία, έτσι γράψε ένα πολύ μικρό πρόγραμμα σε Python για να το καθαρίσουμε. Πρακτικά αφαιρέσαμε τα εξής κλειδιά από το JSON αρχείο:

continentExp, popData2019,countryterritoryCode,geoId,Cumulative_number_for_14_days_of_COVID-19_cases_per_100000.

Κώδικας (όπου `json_string το json αρχείο`):

```
1.  def remove_info(d):
2.  if not isinstance(d, (dict, list)):
3.  return d
4.  if isinstance(d, list):
5.  return [remove_info(v) for v in d]
6.  return {k: remove_info(v) for k, v in d.items()
7.  if k not in {'continentExp',
    'popData2019','countryterritoryCode','geoId','Cumulative_number_for_14_days_of_COVID-
    19_cases_per_100000'}}
8.
9.  data = json.loads(json_string)
10. data = remove_info(data)
11. json_string = json.dumps(data)
12. print(json.dumps(data, sort_keys=True, indent=4))
13. open("updated-covidData.json", "w").write(
14. json.dumps(data, sort_keys=True, indent=4, separators=(',', ': '))
15. )
```

3. ΒΙΒΛΙΟΓΡΑΦΙΑ ΚΑΙ ΠΗΓΕΣ

1. Παραδείγματα κώδικα που δόθηκαν από το GuNet2.
2. Δεύτερο dataset: https://www.ecdc.europa.eu/en/publications-data/download-todays-data-geographic-distribution-covid-19-cases-worldwide
3. https://www.geeksforgeeks.org/implementation-of-blockchain-in-java/
4. https://zetcode.com/java/console/
5. https://www.baeldung.com/java-apache-derby
6. https://www.javatpoint.com/design-patterns-in-java
7. https://www.baeldung.com/java-blockchain
8. https://github.com/GP2Engine/Blockchain-Java-Application
9. https://stackoverflow.com/questions/14197162/what-is-the-equivalent-sql-server-type-for-the-c-sharp-long-type
10. https://www.javatpoint.com/java-get-current-date
11. https://stackoverflow.com/questions/31715310/convert-timestamp-string-to-long-in-java
12. https://programmer.ink/think/create-the-first-block-chain-using-java.html
13. https://www.journaldev.com/2394/java-dependency-injection-design-pattern-example-tutorial

4. ΒΙΒΛΙΟΘΗΚΕΣ ΚΑΙ ΕΡΓΑΛΕΙΑ

Όνομα	*Έκδοση*	*Τύπος*
IntelliJ IDEA Ultimate	2021.3.2	Εργαλείο
Maven	4.0.0	Εργαλείο
JDK	17	Εργαλείο
org.xerial.sqlite-jdbc	3.36.0.2	Βιβλιοθήκη
com.google.code.gson	2.9.0	Βιβλιοθήκη

Πιο συγκεκριμένα οι βιβλιοθήκες που χρειάζονται βρίσκονται στο αρχείο pom.xml που είναι η θεμελιώδης μονάδα εργασίας στο Maven. Είναι ένα αρχείο XML που περιέχει πληροφορίες σχετικά με το έργο και τις λεπτομέρειες διαμόρφωσης που χρησιμοποιήθηκαν από τον Maven για την κατασκευή του έργου.