Πανεπιστήμιο Πελοποννήσου

Τμήμα Ηλεκτρολόγων Μηχανικών

και Μηχανικών Υπολογιστών

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΤΕΧΝΙΚΕΣ SCRAPING ΓΙΑ ΑΝΑΚΤΗΣΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΕΦΑΡΜΟΓΗ ΤΟΥΣ ΣΕ ΤΟΥΡΙΣΤΙΚΕΣ ΠΛΑΤΦΟΡΜΕΣ

Επιβλέπων: Δρ. Χριστοδούλου Σωτήριος

Φοιτητής: Αρβανιτάκης Παναγιώτης - 2780

Περιεχόμενα

Ορισμοί

Web Scraper Εφαρμογή που εξάγει δεδομένα από ιστοσελίδες

JSON JavaScript Object Notation, μορφοποίηση δεδομένων για ανταλλαγή μέσω εφαρμογών

Node.js πλατφόρμα ανάπτυξης λογισμικού χτισμένη σε περιβάλλον Javascript

API Application Programming Interface, διεπαφή ανάμεσα σε 2 τμήματα λογισμικού

VPN Virtual Private network, εικονικό ιδιωτικό δίκτυο

JOB Διεργασία που εκτελείται στο background ενός server

Cron: Βοηθητικό πρόγραμμα λογισμικού γνωστό και ως cron job

Κεφάλαιο 1 - Εισαγωγή

1.1 Πρόβλημα

Στις μέρες μας το διαδίκτυο περιλαμβάνει αμέτρητες ιστοσελίδες που περιέχουν μεγάλο όγκο πληροφοριών. Ο χρήστης είναι υποχρεωμένος για να ενημερωθεί σωστά να δαπανήσει αρκετό χρόνο, κάτι που ίσως τον απομακρύνει, και να επισκεφτεί πολλές φορές πολλές από αυτές προκειμένου να συλλέξει ολόκληρη την πληροφορία.

Συγκεκριμένα η αύξηση των εταιριών που σχετίζονται με τον τομέα του τουρισμό και συγκεκριμένα πλατφόρμες που περιλαμβάνουν όλες τις ξενοδοχειακές μονάδες του κόσμο δίνουν στον χρήστη πληροφορίες που σχετίζονται με την διαμονή, όπως τιμές καθώς και διάφορες παροχές που περιέχει το συγκεκριμένο ξενοδοχείο. Επίσης υπάρχει η δυνατότητα να ασκήσει κριτική για να γνωρίζουν και οι επόμενοι που θα διαμείνουν, περισσότερες πληροφορίες για αυτό το κατάλυμα και πιθανώς βελτίωση του ξενοδόχου.

Με βάση την αύξηση του τουρισμού οι ξενοδόχοι έχουν τοποθετήσει το ξενοδοχείο τους σε όλα τα κανάλια που έχουν αναπτυχθεί για να αυξήσουν την αναγνωσιμότητα και την ανταγωνιστικότητα τους. Αυτό σημαίνει ότι καθημερινά η τιμή του κάθε δωματίου αλλάζει ανά κανάλι και αυτό οφείλεται στην πολιτική κάθε εταιρείας που προβάλλει το συγκεκριμένο κατάλυμα. Επιπλέον σε κάθε διαφορετική πλατφόρμα υπήρχαν και διαφορετικά σχόλια από ανθρώπους που είχαν διαμείνει σε αυτά.

Δημιουργείται έτσι η ανάγκη εξαγωγής της πληροφορίας με τέτοιο τρόπο έτσι ώστε να είναι απολύτως κατανοητή από τον ξενοδόχο το οποίο αποτελεί το κύριο θέμα της παρούσης εργασίας. Συλλέγοντας όλα τα δεδομένα του δίνουμε την δυνατότητα εκτός της μαζικής ανάγνωσης να μπορεί να κάνει και μαζική ανάλυση αυτών.

Η πληροφορία που αναζητούμε χωρίζεται σε δύο κατηγορίες σε αυτή των σχολίων των χρηστών και σε αυτή των διαθέσιμων δωματίων και τιμών αυτών.

Στην κατηγορία των σχολίων των χρηστών συλλέγονται:

- Η ώρα που έκανε ο χρήστης το σχόλιο
- Το ίδιο το σχόλιο
- Όλες τις πληροφορίες του χρήστη όπως όνομα, τοποθεσία κλπ.
- Την βαθμολογία που έχει δώσει
- Την απάντηση του ξενοδόχου στο συγκεκριμένο σχόλιο
- Την ώρα που απάντησε
- Πόσοι έχουν αντιδράσει στο σχόλιο (like – dislike)
- Τον τίτλο του σχολίου
- Το κείμενο το οποίο αναφέρει κάτι θετικό και κάτι αρνητικό για το κατάλυμα
- Tip

Στην κατηγορία των τιμών συλλέγονται:

- Η τιμή
- Το νόμισμα
- Οι παροχές του δωματίου (wiFi, πισίνα, πρωινό κλπ)
- Τον τύπο του δωματίου (δίκλινο, μονόκλινο κλπ)
- Το όνομα του
- Το πλήθος των ατόμων
- Condition (πίνακας που περιλαμβάνει – Αν υπάρχει έκπτωση στο πρωινό, αν γίνεται ακύρωση με πληρωμή ή όχι κλπ)

1.2 Μεθοδολογία

Η μεθοδολογία της παρούσας πτυχιακής εργασίας βασίστηκε σε μια έρευνα που έγινε πάνω στην εύρεση σχολίων και τιμών ανταγωνιστών από ανθρώπους που διαθέτουν ξενοδοχειακές μονάδες. Είναι πολλοί αυτοί οι οποίοι θέλουν να διαβάζουν κριτικές των πελατών τους καθώς και να γνωρίζουν - ενημερώνονται για τις τιμές των ανταγωνιστών τους.

Η ανάγκη αυτή οδήγησε αρχικά στην καταγραφή των αναγκών που χρειάζεται στο συστημα. Υπήρξε μια μελέτη πάνω σε νέες και γρήγορες τεχνολογίες έτσι ώστε να παρέχει το σύστημα γρήγορα αυτό που θέλει ο πελάτης. Στην συνέχεια εκτιμήθηκε ο χρονος δημιουργίας και μια πρώτη εκτίμηση αρχιτεκτονικής και αναγκών. Για την

δημιουργία του συστήματος επιλέχθηκαν τεχνολογίες ανάμεσα σε πολλές και εφαρμοστηκαν στην υλοποίηση.

Ακολούθησε η αναζήτηση των καναλιών που έχουν τον περισσότερο πληθυσμό σε χρήστες. Σε αυτό το στάδιο χρησιμοποιήθηκαν τα χαρακτηριστικά των πελατών που θα ήθελε να προσεγγίσει αυτό το σύστημα π.χ. ηλικία χώρα κλπ οπότε αν το σύστημα είχε σχεδιαστεί για ηλικίες άνω των 40 δεν επιλέχθηκε κανάλι που είχε μέσο όρο ηλικίας 20 χρονον. Στην συνέχεια εκτιμήθηκε η πληροφορία που χρειαζόταν, από κάθε κανάλι, να γνωρίζει ο χρήστης καθώς και το σύστημα. Εκεί ήταν αναγκαίο να εκτιμηθούν τα υπάρχοντα χαρακτηριστικά και να ξεχωρίσουν μονο τα απαραίτητα.

Κατά την υλοποίηση του συστήματος εκτιμήθηκαν δύο τρόποι από τους οποίους βγήκαν μετρήσεις και στο στάδιο ελέγχου διαλέχτηκε ο ένας. Στην συνέχεια δημιουργήθηκε ένας κεντρικό κόμβος ο οποίος ανάλογα το αίτημα καλεί το ανάλογο κόμβο τιμών ή σχολίων. Ακόμη αναπτύχθηκαν κι άλλοι κόμβοι που κάνουν λειτουργικό το σύστημα όπως είναι:

- createJobs δημιουργεί τα αιτήματα
- scrape_parse παίρνει τα αιτήματα και τα στέλνει στον κομβο του scrap
- error αποθηκεύει τα error
- consumer_database αποθηκεύει τα αποτελέσματα των αιτημάτων στην βάση

Συνεχίζοντας εμφανίστηκαν ανάγκες που δεν είχαν καταγραφεί κατα το αρχικό πλάνο του συστήματος. Τέτοια προβλήματα ήταν το μπλοκάρισμα από κανάλια μετά από των υπερβολικών κλήσεων από το σύστημα. Εκεί βρέθηκε λύση με την χρήση vpn και συγκεκριμένα του protonVPN. Κατα την ολοκλήρωση του συστήματος δοκιμάστικε αρχικά με πραγματικά δεδομένα και στην συνέχεια δόθηκε σε πραγματικό τουριστικό σύστημα για να γίνει μια πλήρη δοκιμή όσον αφορά την λειτουργικότητα καθώς και να φανεί αν καλύπτει όλες τις απαιτήσεις που θέλουν οι ξενοδόχοι.

1.3 Δομή της εργασίας

Στις επόμενες ενότητες θα γίνει ανάλυση του τρόπου ανάπτυξης και υλοποίησης της εργασίας, αφού αρχικά γίνει μια βασική επεξήγηση θεωρητικών όρων και εννοιών που χρησιμοποιήθηκαν κατά τη διάρκεια της υλοποίησης της. Θα

παρουσιαστούν εναλλακτικοί τρόποι υλοποίησης για τους οποιους θα γίνει ανάλυση έτσι ώστε να διαπιστωθεί ο λόγος χρησιμοποίησης ή μη χρησιμοποίησης τους.

Στην ενότητα 2 γίνεται αρχικά μια θεωρητική ανάλυση γύρω από τις μεθόδους που χρησιμοποιούνται. Στην συνέχεια, στην υποενότητα 2.2 διευκρινίζονται οι εφαρμογές καθώς και τα νομικά - ηθικά ζητήματα. Επίσης στην υποενότητα 2.3 και 2.4 γίνεται αναφορά στις τεχνικές scraping και αποφυγής ενεργειών που επηρεάζουν την λειτουργικότητα του συστήματος. Τέλος στην 2.5 παρουσιάζονται τα εργαλεία που χρησιμοποιούνται καθώς και παρόμοια αυτών.

Στην ενότητα 3 παρουσιάζεται ο σχεδιασμός του συστήματος μέσα από σενάρια χρήσης και διαγράμματα σεναρίων χρήσης UML. Γίνεται μια πλήρη καταγραφή της αρχιτεκτονικής ολόκληρου του συστήματος καθώς και της βάσης δεδομένων και προβολή της μέσω ER.

Στην ενότητα 4 αναλύεται κώδικας και τα διάφορα αρχεία που πλαισιώνουν το σύστημα. Επίσης γίνεται εμφανή η προτίμηση συγκεκριμένου τρόπου υλοποίησης.

Τέλος στην ενότητα 5 παρουσιάζονται ιδέες για επέκταση του συστήματος καθώς και για άλλες εφαρμογές. Επιπλέον αναλύονται πρακτικές Anti scraping κυρίως για να δυσκολέψει ή να καθυστερήσει στον scraper.

Κεφάλαιο 2 - Web Scraping

2.1 Γενικά

Ο παγκόσμιος ιστός περιέχει μεγάλη ποσότητα πληροφοριών και αυτό οφείλεται στους αυξανόμενους χρήστες του, έτσι λοιπόν οι ιστοσελίδες έχουν το ρόλο να ταξινομήσουν και να δομούν τις πληροφορίες με τέτοιο τρόπο έτσι ώστε να είναι απολύτως κατανοητές από τους χρήστες. Η δομή αυτή, οι πληροφορίες που υπάρχουν σε πολλές σελίδες και πολλά άλλα δυσκολεύουν πρωτίστως τον αναλυτή να επεξεργαστεί τα δεδομένα και στην συνέχεια τους χρήστες που υπάρχουν στιγμές που τους εξυπηρετεί η σωστή δομημένη αλλά και παράλληλα μαζεμένη πληροφορία. Αυτό το πρόβλημα έρχονται να λύσουν οι τεχνικές και οι διαδικασίες εξαγωγής πληροφορίας από ιστοσελίδες που ονομάζεται Web Page Scraping η οποία εξάγει, δομεί και παρουσιάζει μαζικά την πληροφορία. Η διεργασία αυτή έχει αρκετές δυσκολίες:

- Την εύρεση ενός συγκεκριμένου μονοπατιού που θα χαρακτηρίζει μοναδικά μια πληροφορία στην ιστοσελίδα.
- Σε πολλές περιπτώσεις η φυσική γλώσσα περιέχει χαρακτήρες δυσνόητους ή χαρακτήρες που δεν μας χρειάζονται πλέον, και θα πρέπει να εντοπίζονται και να αφαιρούνται.
- Οι ιστοσελίδες αλλάζουν σχεδίαση τακτικά είτε
    - Για πάντα
    - Όταν αντιληφθούν όταν η συγκεκριμένη είσοδος στην σελίδα τους γίνεται από ένα ρομπότ – πρόγραμμα.
- Λόγω της μεγάλης επισκεψιμότητα συγκεκριμένη ώρα μέσα στην ημέρα μπορεί να υπάρχει καθυστέρηση φόρτωσης του περιεχομένου.

Για τους παραπάνω λόγους θα πρέπει το σύστημα να είναι ευέλικτο και θεωρητικά να αποφεύγει τους παραπάνω κινδύνους ενημερώνοντας τον προγραμματιστή για τυχόν πρόβλημα.

2.2 Εφαρμογές και νομικά - ηθικά ζητήματα

Το web scraping είναι μία διαδικασία που αποσκοπεί στο να συλλέξει χρήσιμα δεδομένα από διαφορετικές ιστοσελίδες έτσι ώστε να βοηθήσει τους χρήστες του διαδικτύου να βλέπουν ακριβώς την πληροφορία που αναζητούν. Άλλωστε τόσο στην περίπτωση ενός bot scraper όσο και στην απλή χρήση προγράμματος περιήγησης (browser), ο "χρήστης" ζητά πρόσβαση σε ανοικτά δεδομένα, συνεπώς δεν υπάρχει διαφορά στη νομική αντιμετώπιση. Ορισμένες ιστοσελίδες θεωρούν αυτές τις πληροφορίες "ιδιοκτησία" τους και την ιστο συγκομιδή ως "κλοπή", ωστόσο η νομική αντιμετώπιση στις ΗΠΑ δεν οδηγεί σε αυτό το συμπέρασμα.

Μία ενδιαφέρουσα παρατήρηση αποτελεί το γεγονός ότι επειδή αποτελεί έναν καινούργιο σχετικά κλάδο εντοπίζεται έλλειψη σε επαρκή νομοθεσία. Ο Upadhyay και η ομάδα του (2017) θεωρούν πως σε εφαρμογές που αποσκοπούν σε έρευνα, διδασκαλία ή καλλιτεχνική έκφραση η χρήση ξένων δεδομένων επιτρέπεται, αλλά απαγορεύεται σε περιπτώσεις εμπορικής εκμετάλλευσης. Επίσης, όταν κάποιος εξάγει δεδομένα, για να διαβεβαιώσει πως δεν ξεπερνάει τα όρια, είναι καλό να εξάγει μόνο τα δημοσίως διαθέσιμα δεδομένα που δεν είναι κρυπτογραφημένα. Κάθε ιστοσελίδα περιγράφει στο αρχείο robot.txt τους όρους χρήσης της. Το robot.txt είναι ένα αρχείο που φιλοξενείται στην ιστοσελίδα και δηλώνει ποιοι πόροι δεν είναι προσβάσιμοι από αυτόματες διαδικασίες, όπως αυτή του web scraping.

Υπάρχουν όμως και περιπτώσεις στις οποίες μπορεί ο χρήστης να καλύπτεται από τις νομικές συνέπειες, αλλά να υποκύπτει σε ηθικές διενέξεις. Αντιπροσωπευτικό παράδειγμα ηθικής συμπεριφοράς είναι ο χρήστης να διαβεβαιώνει ότι τηρείται η συχνότητα με την οποία επιτρέπεται η πρόσβαση στην ιστοσελίδα, καθώς υπάρχει πιθανότητα υπερφόρτωσης της ιστοσελίδας. Σε κάθε περίπτωση οι ιστοσελίδες έχουν το δικαίωμα να μπλοκάρουν τον χρήστη ή ακόμη και ολόκληρα συστήματα αλλαγής της διεύθυνσης ip (vpn). Σε αυτή την περίπτωση υπάρχει πιθανότητα να χάσει πραγματικούς χρήστες.

2.3 Τεχνικές Scraping

Για την δημιουργία ενός scraper δεδομένων ιστου χρειάζονται να χρησιμοποιηθεί μια από τις τεχνικές που υπάρχουν. Οι τεχνικές αυτές είναι τρείς:

1. Οι Βιβλιοθήκες που υπάρχουν για την συγκεκριμένη γλώσσα προγραμματισμού που διαλέξαμε
2. Τα πλαίσια
3. Τα Desktop-based περιβάλλοντα

Βιβλιοθήκες

Ένας από τους πιο συνηθισμενους τρόπους δημιουργίας ενός scraper είναι η υλοποίηση του με βιβλιοθήκες οι οποίες συνιστώνται για την δημιουργία ενός νέου συστήματος χρησιμοποιώντας τη γλώσσα προγραμματισμού που γνωρίζει καλύτερο ο προγραμματιστής. Σε αυτήν την περίπτωση έτοιμες βιβλιοθήκες, μας δίνουν την δυνατότητα να έχουμε πρόσβαση σε μια ιστοσελίδα της οποίας τα περιεχόμενα που συλλέγονται αναλύονται και επεξεργάζονται από λειτουργίες που παρέχει ήδη η συγκεκριμένη βιβλιοθήκη.

Μια από τις πιο δημοφιλείς βιβλιοθήκες πρόσβασης σε ιστότοπους είναι η Curl. Υποστηρίζει τις κύριες δυνατότητες του πρωτοκόλλου HTTP, συμπεριλαμβανομένων πιστοποιητικών SSL, HTTP POST, HTTP PUT, FTP uploading, HTTP-based upload, proxy, cookies HTTP authentication. Επιπλέον, έχει χρήσιμες συνδέσεις σε πολλές γλώσσες προγραμματισμού. Η Perl, που είναι μία από τις γλώσσες προγραμματισμού που χρησιμοποιείται ευρύτερα στην βιβλιοπληροφορική, ενσωματώνει το www (Mechanize Web automation module). Άλλες γλώσσες που μπορούν να χρησιμοποιηθούν είναι οι έτοιμες βιβλιοθήκες ή πακέτα που είναι γραμμένα σε Java, Python, C++ και PHP. Οι επιλογές αυτές δεν ειναι εξαντλητικές, καθώς η υλοποίηση ενός web Scraping μπορεί να επιτευχθεί από πληθώρα γλωσσών προγραμματισμού.

Πλαίσια

Η χρήση μιας Βιβλιοθήκης κώδικα για την υλοποίηση ενός scraper μπορεί να έχει και κάποια μειονεκτήματα. Ένα μειονέκτημα είναι ότι δεν προσφέρουν ολοκληρωμένες λύσεις καθώς μπορεί να χρειάζεται άλλη για την πρόσβαση στον ιστό και άλλες για την ανάλυση και εξαγωγή περιεχομένου από την HTML. Επιπλέον, ένας web scraper είναι γνωστό ότι μπορεί να επηρεαστεί σημαντικά από αλλαγές στον HTML κώδικα και για αυτόν τον λόγο χρειάζονται συνεχή συντήρηση.

Τα πλαίσια προσφέρουν την πιο ολοκληρωμένη λύση καθώς προσφέρουν ένα συνδυασμό από βιβλιοθήκες. Για παράδειγμα το scrapy της python είναι ένα πλαίσιο στο οποίο ορίζεται ένα σύνολο από διευθύνσεις URL και μια λειτουργία τύπου "Ανάλυσης". Οι ιστοσελίδες αναλύονται αυτόματα και τα περιεχόμενα εξάγονται χρησιμοποιώντας εκφράσεις XPath. Παρακάτω παρουσιάζεται ένας πίνακας ο οποίος συνοψίζει και συγκρίνει πολλές από τις πιο δημοφιλείς βιβλιοθήκες και πλαίσια ανοιχτού κώδικα. Στον πίνακα μεταξύ άλλων απεικονίζεται το όνομα κάθε λογισμικού καθώς και η γλώσσα προγραμματισμού που εκπροσωπεί. Τέλος παρουσιάζεται και ο τρόπος με τον οποίο αναλύει την ιστοσελίδα το κάθε λογισμικό.

	Type C: HTTP client P: Parsing F: Framework	Domain-specific language	API/stand alone	Language	Extraction facilities R: Regular expressions H: HTML parsed tree X: XPath C: CSS selectors
UNIX shell (curl/wget, grep, sed, cut, paste, awk)	CP	No	SA	bash	R
Curl/libcurl	C	No	Both	C + bindings	
Web-Harvest	F	Yes	Both	Java	RX
Jsoup	CP	No	API	Java	HC
HttpClient	C	No	API	Java	
jARVEST	F	Yes	Both	JRuby/Java	RXC
WWW::Mechanize	CP	No	API	Perl	RX
Scrapy	F	No	Both	Python	RX
BeautifulSoup	P	No	No	Python	H

We have selected several available Web scraping packages oriented to programmers. There are six libraries implementing an HTTP client (C) and/or HTML parsing/extraction (P) and three frameworks (F). Web-Harvest and jARVEST frameworks present a domain-specific language for defining robots, based on XML and Ruby, respectively. For all the analyzed alternatives, we report their extraction facilities, including regular expressions (R), HTML parsed tree (H), XPath expressions (X) and CSS Selectors (C).

Εικόνα 2.3.1 Βιβλιοθήκες και πλαίσια για Web Scraping

Desktop-based περιβάλλοντα

Είναι πληροφοριακά συστήματα έτοιμα προς χρήση. Αυτό το είδος ανταποκρίνεται στις ανάγκες των απλών προγραμματιστών. Είναι εργαλεία που ενισχύονται από γραφικά περιβάλλοντα σχεδιασμού. τα οποία διευκολύνουν την δημιουργία και τη συντήρηση ενός web scraping. Συνήθως, το λογισμικό περιλαμβάνει ένα ενσωματωμένο πρόγραμμα περιήγησης, όπου ο χρήστης μπορεί να πλοηγηθεί στην ιστοσελίδα και να επιλέξει διαδραστικά τα στοιχεία που θέλει να εξάγει. Επιπλέον, διαθέτουν τις επιλογές εξαγωγής δεδομένων σε μορφή CSV, Excel και XML προκειμένου να γίνουν είσοδος μιας βάσης δεδομένων.

Τα μειονεκτήματα των έτοιμων προς χρήση περιβαλλόντων είναι η εμπορική διανομή, που σημαίνει ότι θα χρειαστεί να χρησιμοποιηθεί επί πληρωμή το εργαλείο, καθώς και η περιορισμένη πρόσβαση API, το οποίο καθιστά αρκετά δύσκολη την

ενσωμάτωση του σε διαφορετικά εργαλεία. Στον παρακάτω πίνακα συγκρίνονται επτά Desktop-based περιβάλλοντα:

	IrobotSoft[a]	Visual Web Ripper[b]	Newbie[c]	Mozenda[d]	Screen-scraper[e]	WebSundew[f]	FMiner[g]
Software type							
License	Freeware	Commercial	Commercial	Commercial	Freeware (Basic edition)	Commercial	Commercial
Open source	No	No	No	No	No	No	No
Platforms	Win	Win	Win	Win	Win Linux Mac	Win	Win
Site access							
Form POST	Yes	Yes	Yes	Yes	Yes	No	Yes
Session	Yes	Yes	Yes	Yes	Yes	No	Yes
Conf. user agent	IE	IE and 2 internal UAs	IE Firefox	IE	No	Firefox and I internal UA	No
Iteration over pages	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Anonymizer Proxies	Yes	Yes	N/A	No	No	N/A	Yes
Formats							
Input formats	.irb	.rip	Webpages URL list	.xml	.sss	.zws	.sep
Output formats	Text CSV XML DB	CSV XML DB Excel	Text Excel DB	CSV TSV XML Excel	Text CSV DB	CSV XML Excel	CSV Excel DB
Robot file format	.irb	.rip	.nbs scripts	.xml	.sss	.zws	.sep
Runtime							
Multi-threading	Yes	Yes	Yes	No	Yes	Yes	Yes
Progressive results	Yes	No	Yes	Yes	Yes	Yes	Yes
Design environment							
GUI-based designer	Yes	Yes	Yes	Yes	Yes	Yes	Yes
API access	No	Yes	Yes	Yes	Yes	Yes	No
Scriptable	Yes	Limited	Yes	No	Yes	N/A	No

Comparison of functionalities of different desktop-based scraping solutions. We have selected several features to evaluate the tools, including software license, supported platforms, site access capabilities, runtime aspects and robot design possibilities. [a]http://www.irobotsoft.com. [b]http://www.visualwebripper.com. [c]http://www.newbielabs.com. [d]http://www.mozenda.com. [e]http://www.screen-scraper.com. [f]http://www.websundew.com. [g]http://www.fminer.com.

Εικόνα 2.3.2 Λύσεις για Desktop-based Web Scraping

2.4 Τεχνικές Αποφυγής

Οι τρόποι αποφυγής είναι πολύ χρήσιμοι για τους προγραμματιστές καθώς μειώνουν τις πιθανότητες διακοπής του προγράμματος τους και αυξάνουν το ποσοστό επιτυχής υλοποίησης του. Οι ποιό βασικοί τρόποι είναι:

1. Πιο αργή ανίχνευση δεδομένων

Τα bots κάθε ιστοσελίδας αρχίζουν να λαμβάνουν δεδομένα. Στην περίπτωση που αυτό γίνεται με γρήγορη συχνότητα τότε είναι εύκολο από έναν ιστότοπο να εντοπίσει τον scraper καθώς ένα μεγάλο κομμάτι των χρηστών δεν μπορούν να πιάσουν τέτοιες ταχύτητες. Για να αντιμετωπιστεί θα πρέπει το πρόγραμμα να μιμείται τις ανθρώπινες κινήσεις. Δηλαδή θα πρέπει εντός του προγράμματος να εκτελούνται εντολές παύσης. Επίσης είναι αναγκαίο να μειωθούν οι ταυτόχρονες

αιτήσεις στο σύστημα. Στην καλύτερη περίπτωση θα μπορούρούσε να δημιουργηθεί πρόγραμμα που θα αυξάνει ή θα μειώνει την ταχύτητα με βάση τον φόρτο εργασίας μιας ιστοσελίδας.

2. Αποφυγή ίδιου pattern αιτημάτων στον κεντρικό server κάθε καναλιού

Συνήθως οι άνθρωποι δεν εκτελούν επαναλαμβανόμενες ενέργειες καθώς περιηγούνται σε έναν ιστότοπο με τυχαίες ενέργειες. Τα Web Scraping Bot συνήθως χρησιμοποιούν συγκεκριμένο μοτίβο ανίχνευσης καθώς έχουν προγραμματιστεί με αυτόν τον τρόπο. Ιστότοποι με έξυπνους μηχανισμούς είναι εύκολο να τους εντοπίσουν.

Ενσωματώνοντας μερικά τυχαία κλικ στη σελίδα, κινήσεις του ποντικιού και τυχαίες ενέργειες π.χ. scroll κ.λ.π. το σύστημα μοιάζει με άνθρωπο.

3. Δημιουργία αιτημάτων μέσω proxy server

Κατά την εισαγωγή ενός χρήστη στο διαδίκτυο κάθε ιστοσελίδα μπορεί να αναγνωρίσει την Ip του επομένως μπορεί να αντλήσει χρήσιμες πληροφορίες όπως μοτιβα πλοηγησης, αν ο χρήστης μπαίνει για πρώτη φορά κλπ.

Πολλά αιτήματα που προέρχονται από μια συγκεκριμένη IP οδηγούν σε αποκλεισμό, για αυτό χρησιμοποιούνται οι τεχνικές που παρέχουν την δυνατότητα πολλών διαφορετικών IP. Έτσι καθίσταται δύσκολος ο εντοπισμός πληροφοριών της αρχικής IP.

Υπάρχει δυνατότητα από διάφορες μεθόδους να χρησιμοποιηθούν ip με τυχαία σειρά διασφαλίζοντας έτσι την μυστικότητα της πλοήγησης. Τέτοιες μέθοδοι είναι:

- TOR
- VPNs (protonvpn, nordvpn κλπ)
- Free Proxies
- Shared Proxies
- Private Proxies
- Data Center Proxies
- Residential Proxies

4. Αλλαγή του user Agent πριν από κάθε request

Ο user Agent είναι ένα εργαλείο που υποδεικνύει στον server ποιο πρόγραμμα περιήγησης ιστού χρησιμοποιείται. Στην περίπτωση που δεν αναφέρεται ο user Agent τότε οι ιστότοποι δεν επιστρέφουν περιεχόμενο. Κάθε αίτημα πού γίνεται περιέχει μια κεφαλίδα στην οποία αναγράφεται και ο user Agent επιτρέποντας στις σελίδες να κατανοούν ότι, μετά από την συνεχή χρήση του ίδιου, το συγκεκριμένο request έγινε από κάποιο bot.

Υπάρχουν ιστοσελίδες αλλά και βιβλιοθήκες που δίνουν στους προγραμματιστές λίστες από όλους τους user Agent που υπάρχουν. Έτσι μετά από κάθε κλίση αλλάζοντας τον αυξάνεται η πιθανότητα να μην καταλάβει ένας ιστότοπος το bot. Σε αντίθετη περίπτωση θα πρέπει να προστεθούν επιπλέον πληροφορίες στην κεφαλίδα. Μερικές από αυτές είναι:

- User-Agent
- Accept
- Accept-Language
- Referer
- DNT
- Updgrade-Insecure-Requests
- Cache-Control

2.4 Εργαλεία

Nodejs

Το **Node.js** είναι μια πλατφόρμα ανοιχτού κώδικα (open source cross-platform) ανάπτυξης λογισμικού η οποία έχει υλοποιηθεί σε runtime περιβάλλον Javascript. Η συγκεκριμένη πλατφόρμα είναι δωρεάν και χρησιμοποιείται από αρκετούς προγραμματιστές.

Το runtime περιβάλλον του **Node.js** χρησιμοποιεί τη μηχανή V8 JavaScript της Google (Google V8 JavaScript Engine) για την ερμηνεία του JavaScript. Με τη συγκεκριμένη πλατφόρμα γίνεται η υλοποίηση μιας ποικιλίας εργαλείων server και εφαρμογών. Κάποια βασικά χαρακτηριστικά του Node.js είναι:

- **Ασύγχρονο και Event Driven**

Όλα τα API της βιβλιοθήκης του Node.js είναι ασύγχρονα, δηλαδή, non-blocking. Αυτό σημαίνει ότι ένας server βασισμένος σε Node.js δεν περιμένει ένα API να για να επιστρέψει δεδομένα. Ο server μεταφέρεται στο επόμενο API μετά την κλήση του και ο μηχανισμός κοινοποίησης των Event του Node.js βοηθά το server να πάρει την απάντηση από την προηγούμενη κλήση που έκανε στο API.

- **Πολύ γρήγορο**

  Λόγω του ότι είναι βασισμένο στη μηχανή V8 JavaScript της Google η βιβλιοθήκη του εκτελεί πολύ γρήγορα τον κώδικα.

- **No Buffering**

  Οι Node.js εφαρμογές ποτέ δεν κάνουν buffer τα δεδομένα.

Puppeteer

Η Puppeteer είναι μια βιβλιοθήκη Node.js ανοιχτού κώδικα που αναπτύχθηκε από την Google με ευρεία υποστήριξη για σχεδόν οποιαδήποτε ενέργεια στο πρόγραμμα περιήγησης Chrome της Google. Η βασική ιδέα είναι ένα API σε υψηλό επίπεδο που επιτρέπει να αυτοματοποιούμε τις ενέργειες σε οποιοδήποτε από τα προγράμματα περιήγησης Google, το Chrome και το Chromium.

Τα περισσότερα πράγματα που μπορούν να πραγματοποιηθούν χειροκίνητα στο πρόγραμμα περιήγησης μπορούν να γίνουν χρησιμοποιώντας την βιβλιοθήκη Puppeteer. Μερικά από αυτά είναι:

- Παραγωγή screenshots και pdf από τις σελίδες
- Αυτόματη συμπλήρωσης φόρμας
- Εισαγωγή κειμένου, πάτημα πάνω σε κουμπί καθώς και επιλογή συγκεκριμένου πεδίου από drop down μενού

Selenium

Τα τελευταία χρόνια το selenium έχει γίνει το πιο διαδεδομένο εργαλείο στην αγορά, λόγω του μοντέλου ανοιχτού κώδικα και της αυξανόμενης ωριμότητας σε σύγκριση με άλλα δωρεάν εργαλεία. Το Selenium έχει πλέον τοποθετηθεί ως η τυπική λύση όταν ασχολείται με λειτουργικό αυτοματισμό, επειδή έχει χαμηλό κόστος

απόκτησης και υψηλό επίπεδο ενοποίησης με .Net, Python, Node.js και φυσικά με την Java.

Ποια τα ωφέλει της puppeteer σε σχέση με το Selenium;

Τα κύρια πλεονεκτήματα που έχουν επισημανθεί κατά τη σύγκριση των Puppeteer και Selenium είναι:

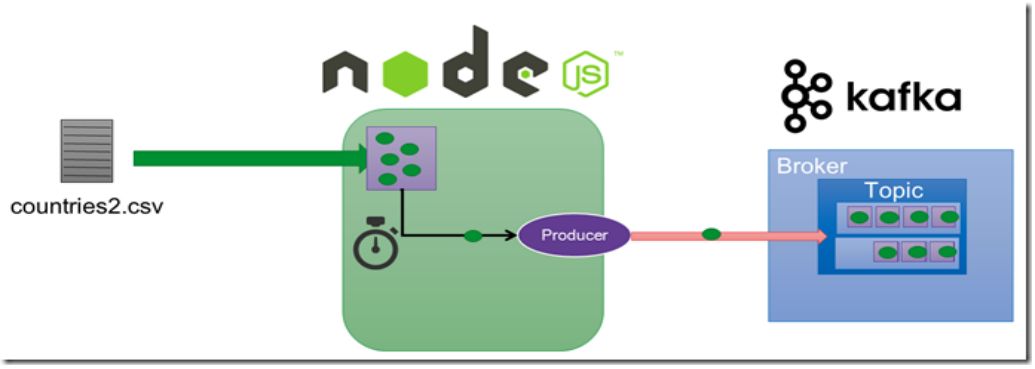
- Η Puppeteer επιτρέπει την μέτρηση των χρόνων φόρτωσης και απόδοσης που παρέχονται από το εργαλείο Ανάλυσης απόδοσης του Chrome.
- Η Puppeteer παρέχει μεγαλύτερο έλεγχο στα προγράμματα περιήγησης του Chrome από ό, τι προσφέρει το Selenium WebDriver.
- Η Puppeteer καταργεί την εξάρτηση από ένα εξωτερικό πρόγραμμα οδήγησης για την εκτέλεση των δοκιμών, αν και αυτό το πρόβλημα στο Selenium θα μπορούσε να μετριαστεί χρησιμοποιώντας την εξάρτηση Web Driver Manager από τον Boni García.
- Η Puppeteer έχει οριστεί σαν προεπιλεγμένη λειτουργία εκτέλεσης ως headless και μπορεί επίσης να αλλάξει για να παρακολουθεί την εκτέλεση ζωντανά σε λειτουργία non-headless.

Apache Kafka

Με την ονομασία Apache Kafka είναι γνωστή πλατφόρμα λογισμικού για επεξεργασία ροών δεδομένων. Είναι γραμμένη στις γλώσσες προγραμματισμού Scala και Java, ενώ πρόκειται για λογισμικό κώδικα ελεύθερου προς ανάπτυξη από όλους. Το έργο αποσκοπεί να παρέχει μια ενιαία πλατφόρμα για χειρισμό ροών δεδομένων σε πραγματικό χρόνο, με χαρακτηριστικά την υψηλή απόδοση και ελάχιστες περιόδους αδράνειας. Στην αρχιτεκτονική του το επίπεδο αποθήκευσης είναι ουσιαστικά μια ουρά δημοσίευσης και κατανάλωσης μηνυμάτων, με τεράστια επιδεκτικότητα διεύρυνσης, σχεδιασμένη σαν ένα κατανεμημένο αρχείο καταγραφής συναλλαγών. Αυτό την καθιστά ιδιαίτερα πολύτιμη για εφαρμογές που απευθύνονται σε επιχειρήσεις και βιομηχανίες με την ανάγκη να επεξεργάζονται ροές δεδομένων.

Το Kafka χρησιμοποιείται συχνά σε αρχιτεκτονικές ροής δεδομένων σε πραγματικό χρόνο για την παροχή αναλυτικών στοιχείων σε πραγματικό χρόνο. Δεδομένου ότι το Kafka είναι ένα γρήγορο, επεκτάσιμο, ανθεκτικό και ανεκτικό σε σφάλματα σύστημα δημοσίευσης-εγγραφής μηνυμάτων, χρησιμοποιείται σε

περιπτώσεις χρήσης όπου JMS, RabbitMQ και AMQP ενδέχεται να μην μπορούν να ανταποκριθούν λόγω όγκου και απόκρισης.

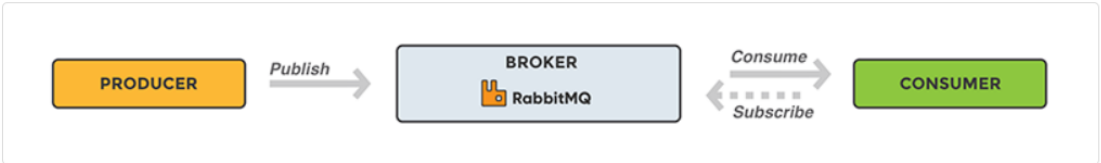


Εικόνα 2.4.1 Λειτουργία Kafka μέσω NodeJS

RabbitMQ

Το RabbitMQ είναι ένα λογισμικό ουράς μηνυμάτων γνωστό επίσης ως broker μηνυμάτων ή διαχειριστής ουρών. Στην ουσία είναι ένα λογισμικό όπου ορίζονται οι ουρές, με τις οποίες συνδέονται εφαρμογές για τη μεταφορά ενός μηνύματος ή μηνυμάτων.

Ένα μήνυμα μπορεί να περιλαμβάνει κάθε είδους πληροφορίες. Θα μπορούσε, για παράδειγμα, να έχει πληροφορίες σχετικά με μια διαδικασία ή μια εργασία που θα πρέπει να ξεκινά από μια άλλη εφαρμογή (η οποία θα μπορούσε ακόμη και να βρίσκεται σε άλλο διακομιστή) ή θα μπορούσε να είναι απλώς ένα απλό μήνυμα κειμένου. Το λογισμικό διαχείρισης ουρών αποθηκεύει τα μηνύματα έως ότου μια εφαρμογή λήψης συνδεθεί και αφαιρέσει ένα μήνυμα από την ουρά. Στη συνέχεια, η εφαρμογή λήψης επεξεργάζεται το μήνυμα.



Εικόνα 2.4.2 Εσωτερική λειτουργία του Kafka

Γιατί kafka και όχι RabbitMQ

Η Kafka προσφέρει πολύ υψηλότερη απόδοση από τους broker μηνυμάτων όπως είναι το RabbitMQ. Χρησιμοποιεί διαδοχικό δίσκο I/O για την ενίσχυση της

απόδοσης, καθιστώντας την κατάλληλη επιλογή για την εφαρμογή ουρών. Μπορεί να πετύχει υψηλή απόδοση (εκατομμύρια μηνύματα ανά δευτερόλεπτο) με περιορισμένους πόρους. Κατάλληλο για μεγάλες περιπτώσεις χρήσης δεδομένων.

Το RabbitMQ μπορεί επίσης να επεξεργαστεί ένα εκατομμύριο μηνύματα ανά δευτερόλεπτο, αλλά απαιτεί περισσότερους πόρους (περίπου 30 κόμβους). Μπορείτε να χρησιμοποιήσετε το RabbitMQ για πολλές από τις ίδιες περιπτώσεις χρήσης με το Kafka, αλλά θα πρέπει να το συνδυάσετε με άλλα εργαλεία όπως το Apache Cassandra.

Grafana and Prometheus

Το έργο Grafana ξεκίνησε από τον Torkel Ödegaard το 2014 και έχει γίνει τα τελευταία δύο χρόνια ένα από τα πιο δημοφιλή έργα ανοιχτού κώδικα στο GitHub. Σας επιτρέπει να κάνετε ερώτημα, οπτικοποίηση και ειδοποίηση σχετικά με μετρήσεις και αρχεία καταγραφής, ανεξάρτητα από το πού βρίσκονται.

Η Grafana διαθέτει μοντέλο με δυνατότητα προσθήκης δεδομένων και συνοδεύεται από πλούσια υποστήριξη για πολλές από τις πιο δημοφιλείς βάσεις δεδομένων χρονοσειρών όπως Graphite, Prometheus, Elasticsearch, OpenTSDB και InfluxDB. Διαθέτει επίσης ενσωματωμένη υποστήριξη για προμηθευτές παρακολούθησης cloud όπως το Google Stackdriver, το Amazon Cloudwatch, το Microsoft Azure και τις βάσεις δεδομένων SQL όπως η MySQL και η Postgres. Η Grafana είναι το μόνο εργαλείο που μπορεί να συνδυάσει δεδομένα από τόσα μέρη σε έναν πίνακα οργάνων.

Η Prometheus είναι ένα σύστημα παρακολούθησης ανοιχτού κώδικα που αναπτύχθηκε από μηχανικούς στο SoundCloud το 2012. Το 2016, η Prometheus ήταν το δεύτερο έργο που έγινε δεκτό στο Ίδρυμα Cloud Native Computing μετά το Kubernetes και επίσης το δεύτερο που αποφοίτησε το 2018.

Το σύστημα παρακολούθησης Prometheus περιλαμβάνει ένα πλούσιο, πολυδιάστατο μοντέλο δεδομένων, μια συνοπτική και ισχυρή γλώσσα ερωτημάτων που ονομάζεται PromQL, μια αποτελεσματική ενσωματωμένη βάση δεδομένων χρονομέτρων και περισσότερες από 150 ενσωματώσεις με συστήματα τρίτων.

Η Grafana Labs είναι περήφανη που υποστηρίζει την ανάπτυξη του έργου Prometheus απασχολώντας συντηρητές Prometheus, οικοδομώντας υποστήριξη

πρώτης κατηγορίας για το Prometheus στο Grafana διασφαλίζοντας έτσι ότι οι πελάτες της Grafana λαμβάνουν υποστήριξη και χαρακτηριστικά γνωρίσματα Prometheus που χρειάζονται.

Protonvpn

Το ProtonVPN είναι ένας πάροχος υπηρεσιών εικονικού ιδιωτικού δικτύου (VPN) που διαχειρίζεται η ελβετική εταιρεία Proton Technologies AG.

Ασφάλεια

Η κυκλοφορία του χρήστη στέλνετε στο Διαδίκτυο μέσω κρυπτογραφημένου vpn Tunnel, έτσι ώστε οι κωδικοί πρόσβασης και τα εμπιστευτικά δεδομένα να παραμένουν ασφαλή.

Ιδιωτικότητα

Κρατάει το ιστορικό ιδιωτικό καθώς δεν καταγράφεται η δραστηριότητα των χρηστών και δεν κοινοποιούνται δεδομένα σε τρίτα σημεία.

Ελευθερία

Το ProtonVPN καταργεί τα εμπόδια της λογοκρισίας στο Διαδίκτυο, επιτρέποντάς σας να έχετε πρόσβαση σε οποιονδήποτε ιστότοπο ή περιεχόμενο.

Postgresql

Η PostgreSQL είναι μια σχεσιακή βάση δεδομένων ανοικτού κώδικα με πολλές δυνατότητες. Η ανάπτυξη έχει ξεκινήσει εδώ και δύο δεκαετίες και βασίζεται σε μια αποδεδειγμένα καλή αρχιτεκτονική η οποία έχει δημιουργήσει μια ισχυρή αντίληψη των χρηστών της γύρω από την αξιοπιστία, την ακεραιότητα δεδομένων και την ορθή λειτουργία. Υπάρχει δυνατότητα χρησιμοποίησης από όλα τα λειτουργικά συστήματα όπως είναι το Linux, το UNIX (AIX, BSD, HP-UX, SGI, IRIX, MAC OS X, Solaris, Tru64) και τα Windows

JWT

Τα JSON Web Tokens αποτελούν ένα standard μιας και η πληροφορία που φέρουν μεταδίδεται μέσω JSON. Λειτουργούν καλά ανάμεσα σε πολλές διαφορετικές γλώσσες όπως .NET, Python, Node.js, Java, PHP, Ruby, Go, JavaScript, και Haskell. Και τα χαρακτηριστικά του είναι:

- Τα JWTs περιέχουν μέσα τους όλη την απαραίτητη πληροφορία. Αυτό

σημαίνει ότι ένα JWT είναι σε θέση να μεταφέρει βασική πληροφορία για αυτό, το φορτίο του (που συνήθως είναι πληροφορίες που σχετίζονται με έναν χρήστη) και μια υπογραφή.

- Τα JWTs μπορούν να μεταφερθούν εύκολα. Από τη στιγμή που είναι αυτόνομα, μπορούν να χρησιμοποιηθούν μέσα σε HTTP headers όταν πιστοποιούν ένα API. Μπορούν επίσης να σταλούν μέσω του URL (δεν συνιστάται).

Μορφή

Είναι εύκολο να αναγνωριστεί ένα JWT. Είναι 3 αλφαριθμητικά χωρισμένα με τελεία. Για παράδειγμα: aaaaaaaaaaa.bbbbbbbbbbb.ccccccccccccccc.

Από τη στιγμή που περιλαμβάνει 3 πεδία, καθένα από αυτά δημιουργείται διαφορετικά. Τα πεδία αυτά είναι:

- Header
- Payload
- Signature

Ανάλυση

Ο **header** περιλαμβάνει 2 μέρη:

- αναφορά του τύπου, που είναι JWT
- ο hashing αλγόριθμος που χρησιμοποιείται

Ένα παράδειγμα φαίνεται παρακάτω:

{ “typ”: “JWT”, “alg”: “HS256” }

Επειδή αυτό το κομμάτι κωδικοποιείται (base64encode), το πρώτο μέρος του JWT είναι της μορφής: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ

Το **payload** μεταφέρει το κυρίως φορτίο του JWT. Εδώ είναι που μπαίνει η πληροφορία που χρειάζεται για να μεταδοθεί καθώς και άλλες πληροφορίες σχετικές με το token. Ένα παράδειγμα του payload φαίνεται παρακάτω:

{ “iss”: “scotch.io”, “exp”: 1300819380, “name”: “Chris Sevilleja”, “admin”: true }

Το οποίο όταν κωδικοποιηθεί επιστρέφει: eyJpc3MiOiJzY290Y2guaW8iLCJleHAiOjEzMDA4MTkzODAsIm5hbWUiOiJDaHJpcyBTZXZpb GxlamEiLCJhZG1pbiI6dHJ1ZX0 . **Αυτό αποτελεί το δεύτερο μέρος του JWT.**

Το τρίτο και τελευταίο μέρος αποτελεί η **υπογραφή**. Αυτό το μέρος αποτελείται από τη hash τιμή των header, payload και ενός μυστικού κλειδιού. Το κλειδί αυτό βρίσκεται στο server. Έτσι ο server είναι σε θέση να πιστοποιήσει υπάρχοντα tokens και να δημιουργήσει νέα. Έτσι προκύπτει και το τελευταίο μέρος του token: 03f329983b86f7d9a9f5fef85305880101d5e302afafa20154d094b229f75773.

<u>Διαδικασία</u>

Η κρυπτογράφηση και η αποκρυπτογράφηση γίνεται αυτόματα χρησιμοποιώντας την αντίστοιχη βιβλιοθήκη για κάθε γλώσσα προγραμματισμού. Παρακάτω υπάρχουν οι βασικές εντολές της βιβλιοθήκης jsonwebtoken στο nodejs:

- jwt.verify(token,privateKey,function(err))
- jwt.sign(infoJSON,privateKey,tokenInfo,function(err))
    - infoJSON: όλες οι πληροφορίες που στέλνονται π.χ. {name:panayiotis,age:22}
    - tokenInfo: μπορεί να είναι ο αλγόριθμος κρυπτογράφησης και “ο χρόνος ζωής του” π.χ. {expiresIn:60*60,algorithm:’RS256’}
    - privateKey: μοναδικός κωδικός που γνωρίζουν μόνο οι εφαρμογές που επικοινωνούν

Κεφάλαιο 3 - Σχεδιασμός συστήματος

3.1 Ανάλυση απαιτήσεων

Το αντικείμενο αυτής της πτυχιακής εργασίας απαιτεί την δημιουργία ενός web Scraper ο οποίος θα μπορεί να επισκεφτεί μια σελίδα, να την επεξεργαστεί και στο τέλος να συλέξει την πληροφορία που χρειάζεται ο χρήστης. Κυρίως έχει δημιουργηθεί για συστήματα τουρισμού και για αυτό μπορεί να εντοπίσει πληροφορίες όπως διαθεσιμότητα δωματίων, τιμές, κριτικές, πληροφορίες χρηστών και βαθμολογία ξενοδοχείου. Ο εντοπισμός γίνεται με την χρήση tag και keyword που έχουν διαμορφωθεί σε μορφή JSON και προστίθενται στον κάθε scraper bot κατα την έναρξη του. Τα δεδομένα που συλλέγονται αποθηκεύονται σε μια βάση δεδομένων και μέσω του API προσφέρονται στον κάθε χρήστη που θα τα ζητήσει.

Για την ορθή υλοποίηση του web scraper θα πρέπει ο διαχειριστής να ακολουθεί κάποια συγκεκριμένα βήματα. Αρχικά ο controller του θα πρέπει να ζητάει από το scrap system ένα μοναδικό token για κάθε χρήστη, αυτό επιτυγχάνεται κάνοντας ένα request GET στο endpoint του API /auth. Αφού δεχτεί σαν απάντηση τον κωδικό τότε το στέλνει στην κεντρική σελίδα που περιέχει τα αποτελέσματα του scrape system. Εκει ο διαχειριστής θα πρέπει να έχει δημιουργήσει μια φόρμα, στην οποία θα μπορεί ο κάθε χρήστης να προσθέτει νέο task στο σύστημα. Οι πληροφορίες που θα πρέπει να περιέχει είναι:

- Σχόλια
    - Την διεύθυνση url
- Τιμές
    - Την διεύθυνση url
    - Το βάθος ημερομηνιών
    - Την ημερομηνία έναρξης
    - Τις ηλικίες των παιδιών
    - Το πλήθος των ατόμων άνω των 18 ετών

Εκτός από αυτές τις πληροφορίες ο διαχειριστής θα πρέπει από μόνος του να βάλει περιορισμούς σε κάθε χρήστη. Τέτοιοι περιορισμοί μπορεί να είναι το άνω όριο βάθους, αιτημάτων σχολίων, αιτημάτων τιμών, αν ο χρήστης μπορεί να εισάγει τουλάχιστον ένα αίτημα, συχνότητα κάθε περάσματος. Έτσι σε προκαθορισμένη ώρα

και ημέρα τρέχει για κάθε ξενοδοχείο κάθε αίτημα και συλλέγει τις πληροφορίες με βάση των παραπάνω παραμέτρων.

3.2 Περιπτώσεις χρήσης

Scraper

Στο κομμάτι του Web Scraping ως actor θεωρείται ο μηχανισμός scraper καθώς και ο διαχειριστής του συστήματος

Σκοπός: Scrape Data

Actor: Scraper

Ενδιαφερόμενα μέρη και ενδιαφέροντα: Ο διαχειριστής να έχει θέσει σε κατάσταση λειτουργίας τον Scraper.

Προϋποθέσεις: Έχουν ολοκληρωθεί οι κατάλληλες ρυθμίσεις για κάθε ξενοδοχείο που πρέπει να συλλέξει πληροφορίες ο Scraper

Ιδανικό σενάριο:

- Scraper: λαμβάνει τις πληροφορίες
- Scraper: επεξεργάζεται την σελίδα με βάση τις πληροφορίες (πάτημα κουμπιών κλπ)
- Scraper: τραβάει την πληροφορία που χρειάζεται
- Scraper: αποθηκεύει την πληροφορία στην βάση.

Εναλλακτικά Σενάρια:

Σενάριο που για κάποιο λόγο κατά την διάρκεια συλλογής της πληροφορίας κάτι δεν πήγε καλά:

- Scraper: λαμβάνει τις πληροφορίες
- Scraper: επεξεργάζεται την σελίδα με βάση τις πληροφορίες (πάτημα κουμπιών κλπ)
- Scraper: για κάποιο λόγο δεν καταφέρνει να πάρει την πληροφορία
- Διαχειριστής: επιστρέφει μήνυμα λάθους και συνεχίζει την διαδικασία

Σενάριο που δεν πήγε καλά λόγο Recaptcha:

- Scraper: λαμβάνει τις πληροφορίες
- Scraper: Προσπαθεί να συνδεθεί στην σελίδα αλλά εντοπίζει reCaptcha
- Διαχειριστής: αλλάζει ip
- Διαχειριστής: επιστρέφει μήνυμα λάθους και συνεχίζει την διαδικασία

API

Το API εκτελεί πολλές λειτουργίες οι οποίες φαίνονται παρακάτω:

1ο Σενάριο authentication

Για την επιτυχή επιστροφή token από το σύστημα scrape προς τον διαχειριστή προέκυψαν οι παρακάτω περιπτώσεις χρήσης:

Σκοπός: fetch token

Actor: API

Ενδιαφερόμενα μέρη και τι ζητούν:

- Διαχειριστής συστήματος: κατάσταση εκτέλεσης /auth με παραμέτρους το hotelId και username και password χρηστών. Αν υπάρχει ήδη token τοποθετείται στον header.

Προϋποθέσεις:

- Έχει γίνει εισαγωγή των χρηστών στην βάση δεδομένων και υπάρχει κάποιο hotelId έτσι ώστε να χαρακτηρίζεται ο χρήστης μοναδικός.

Ιδανικό σενάριο:

- controller: στέλνει τα δεδομένα μαζί με τον header
- API: διασταυρώνει τα στοιχεία
- API: δημιουργεί το token
- API: στέλνει το token

Εναλλακτικά σενάρια:

- Ο controller έχει ήδη token και είναι σωστό και δεν έχει λήξει
    - controller: στέλνει τα δεδομένα μαζί με τον header
    - API: διασταυρώνει τα στοιχεία
    - API: βλέπει το token και κοιτάει αν ειναι εγκυρο
    - API: αλλάζει την ημερομηνία λήξης του token
    - API: στέλνει το token
- Ο controller έχει ήδη token αλλά είναι ψεύτικο
    - Controller: στέλνει τα δεδομένα μαζί με τον header
    - API: διασταυρώνει τα στοιχεία
    - API: βλέπει το token οτι δεν ειναι εγκυρο
    - API: επιστρέφει μήνυμα λάθους
- Ο controller έχει ήδη token είναι σωστό και έχει λήξει.
    - Controller: στέλνει τα δεδομένα μαζί με τον header

- API: διασταυρώνει τα στοιχεία
- API: βλέπει ότι token είναι έγκυρο αλλά έχει λήξει
- API: δημιουργεί νέο token
- API: επιστρέφει νέο token

2ο Σενάριο επιστροφή όλων των σχολίων με συγκεκριμένο HotelID

Πεδίο εφαρμογής: API

Κύριοι Παράγοντες: Controller, API

Ενδιαφερόμενα μέρη και τι ζητούν: Ανατρέχει την βάση και εντοπίζει σχόλια με συγκεκριμένο hotelId

Προϋποθέσεις: σε κάθε request ο controller να προσθέτει το token και θα πρέπει να έχουν περαστεί οι ρυθμίσεις από το διαχειριστικό του controller

Ιδανικό σενάριο:

- Controller: στέλνει το hotelid
- API: κοιτάει στην βάση
- API: εντοπίζει τα review
- API: επιστρέφει όλα τα αποτελέσματα

Σενάριο αποτυχίας:

- Controller: στέλνει το hotelid
- API: κοιτάει στην βάση
- API: δεν εντοπίζει κάποια δεδομένα
- API: επιστρέφει κενό πίνακα.

3ο Σενάριο επιστροφή όλων των τιμών με συγκεκριμένο HotelID

Πεδίο εφαρμογής: API

Κύριοι Παράγοντες: Controller, API

Ενδιαφερόμενα μέρη και τι ζητούν: Ανατρέχει την βάση και εντοπίζει τιμές με συγκεκριμένο hotelId

Προϋποθέσεις: σε κάθε request ο controller να προσθέτει το token και θα πρέπει να έχουν περαστεί οι ρυθμίσεις από το διαχειριστικό του controller

Ιδανικό σενάριο:

- Controller: στέλνει το hotelid
- API: κοιτάει στην βάση
- API: εντοπίζει τις τιμές

- API: επιστρέφει όλα τα αποτελέσματα

Σενάριο αποτυχίας:

- Controller: στέλνει το hotelid
- API: κοιτάει στην βάση
- API: δεν εντοπίζει κάποια δεδομένα
- API: επιστρέφει κενό πίνακα.

4ο Σενάριο επιστροφή όλων των νέων σχολίων

Πεδίο εφαρμογής: API

Κύριοι Παράγοντες: Controller, API

Ενδιαφερόμενα μέρη και τι ζητούν: Ανατρέχει την βάση και εντοπίζει τα τελευταία σχόλια με συγκεκριμένο hotelId

Προϋποθέσεις: σε κάθε request ο controller να προσθέτει το token και θα πρέπει να έχουν περαστεί οι ρυθμίσεις από το διαχειριστικό του controller

Ιδανικό σενάριο:

- Controller: στέλνει το hotelid
- API: κοιτάει στην βάση
- API: εντοπίζει τα σχόλια
- API: επιστρέφει όλα τα αποτελέσματα

5ο Σενάριο επιστροφή ρυθμίσεων που έχουν γίνει από το διαχειριστικό του controller για κάθε χρήστη

Πεδίο εφαρμογής: API

Κύριοι Παράγοντες: Controller, API

Ενδιαφερόμενα μέρη και τι ζητούν: Ανατρέχει την βάση και εντοπίζει τιμές με συγκεκριμένο hotelId

Προϋποθέσεις: σε κάθε request ο controller να προσθέτει το token

Ιδανικό σενάριο:

- Controller: στέλνει το hotelid
- API: κοιτάει στην βάση
- API: εντοπίζει τις ρυθμίσεις
- API: επιστρέφει όλα τα αποτελέσματα

Σενάριο αποτυχίας:

- Controller: στέλνει το hotelid

- API: κοιτάει στην βάση
- API: δεν εντοπίζει κάποια δεδομένα
- API: επιστρέφει κενό πίνακα.

6ο Σενάριο Νέο αίτημα

Πεδίο εφαρμογής: API

Κύριοι Παράγοντες: Controller, API

Ενδιαφερόμενα μέρη και τι ζητούν: -

Προϋποθέσεις: σε κάθε request ο controller να προσθέτει το token και θα πρέπει να έχουν περαστεί οι ρυθμίσεις από το διαχειριστικό του controller

Ιδανικό σενάριο:

- Controller: στέλνει το JSON με το τις παραμέτρους
- API: γίνεται έλεγχος παραμέτρων
- API: δημιουργεί νέα εγγραφή στην βάση
- API: επιστρέφει μήνυμα επιτυχούς εκτέλεσης

Σενάριο αποτυχίας:

- Controller: στέλνει το JSON με το τις παραμέτρους
- API: γίνεται έλεγχος παραμέτρων
- API: εντοπίζει λανθασμένα στοιχεία
- API: επιστρέφει μήνυμα αποτυχίας

7ο Σενάριο Ενημέρωση αιτήματος

Πεδίο εφαρμογής: API

Κύριοι Παράγοντες: Controller, API

Ενδιαφερόμενα μέρη και τι ζητούν: -

Προϋποθέσεις: σε κάθε request ο controller να προσθέτει το token και θα πρέπει να έχουν περαστεί οι ρυθμίσεις από το διαχειριστικό του controller

Ιδανικό σενάριο:

- Controller: στέλνει το JSON με το τις παραμέτρους
- API: γίνεται έλεγχος παραμέτρων
- API: δημιουργεί νέα εγγραφή στην βάση
- API: επιστρέφει μήνυμα επιτυχούς εκτέλεσης

Σενάριο αποτυχίας:

- Controller: στέλνει το JSON με το τις παραμέτρους

- API: γίνεται έλεγχος παραμέτρων
- API: εντοπίζει λανθασμένα στοιχεία
- API: επιστρέφει μήνυμα αποτυχίας

8ο Σενάριο Διαγραφή αιτήματος

Πεδίο εφαρμογής: API

Κύριοι Παράγοντες: Controller, API

Ενδιαφερόμενα μέρη και τι ζητούν: -

Προϋποθέσεις: σε κάθε request ο controller να προσθέτει το token Ιδανικό σενάριο:

- Controller: στέλνει το μοναδικό id και το hotelID
- API: γίνεται έλεγχος παραμέτρων
- API: διαγράφει τις εγγραφές
- API: επιστρέφει μήνυμα επιτυχούς εκτέλεσης

Σενάριο αποτυχίας:

- Controller: στέλνει το μοναδικό id και το hotelID
- API: γίνεται έλεγχος παραμέτρων
- API: εντοπίζει λανθασμένα στοιχεία
- API: επιστρέφει μήνυμα αποτυχίας

9ο Σενάριο Δημιουργια ρυθμίσεων

Πεδίο εφαρμογής: API

Κύριοι Παράγοντες: Controller, API

Ενδιαφερόμενα μέρη και τι ζητούν: -

Προϋποθέσεις: σε κάθε request ο controller να προσθέτει το token

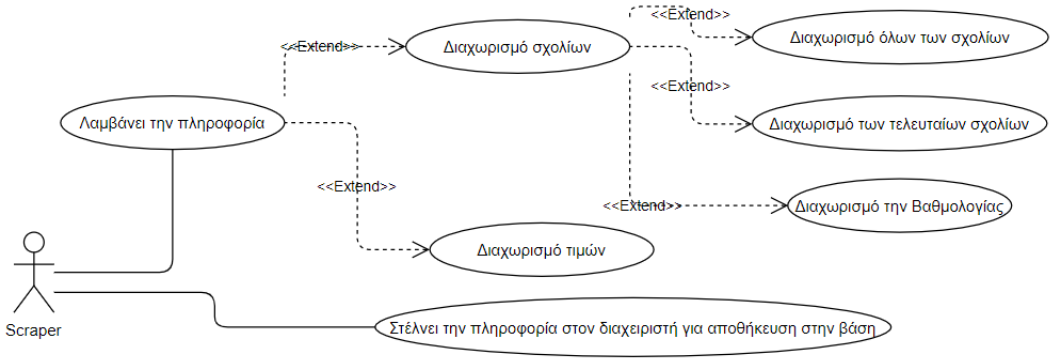
Ιδανικό σενάριο:

- Controller: στέλνει το μοναδικό id και το hotelID
- API: γίνεται έλεγχος παραμέτρων
- API: γίνεται η εγγραφή στην βάση
- API: επιστρέφει μήνυμα επιτυχούς εκτέλεσης

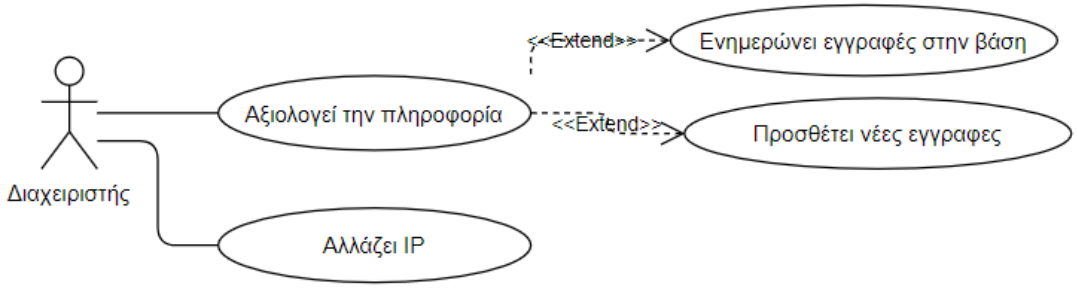
Σενάριο αποτυχίας:

- Controller: στέλνει το μοναδικό id και το hotelID
- API: γίνεται έλεγχος παραμέτρων
- API: εντοπίζει λανθασμένα στοιχεία
- API: επιστρέφει μήνυμα αποτυχίας

3.3 Διαγράμματα περιπτώσεων χρήσης

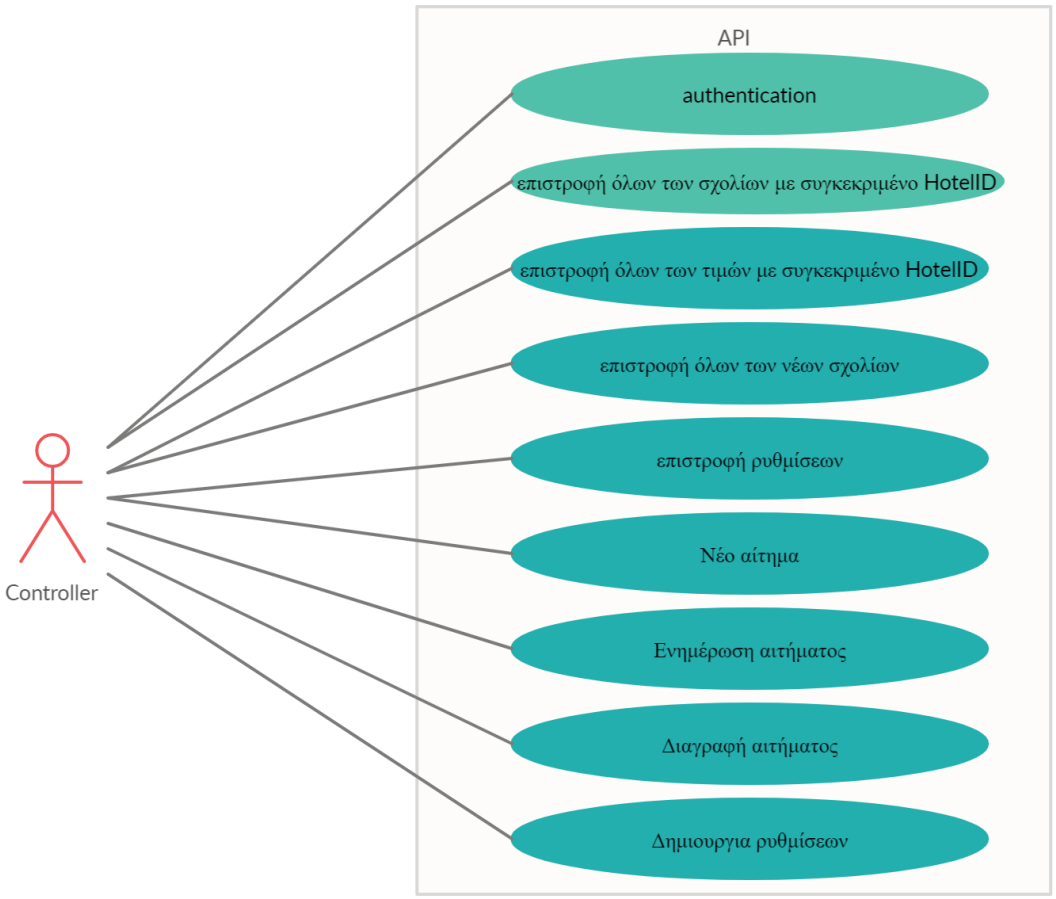


Εικόνα 3.3.1 Διάγραμμα περιπτώσεων χρήσης Scraper



Εικόνα 3.3.2 Διάγραμμα περιπτώσεων χρήσης Διαχειριστή

API



Εικόνα 3.3.2 Διάγραμμα περιπτώσεων χρήσης για το API

3.4 Αρχιτεκτονική

Φυσική Αρχιτεκτονική

Για την δημιουργία και την έναρξη του συστήματος απαιτείται η χρήση δύο server εκ των οποίων:

ο πρώτος θα περιλαμβάνει:

- το API
- το grafana API

ο δεύτερος θα περιλαμβάνει:

- το κάφκα
- protonVpn

- Postgresql

Το vpn που χρησιμοποιείται και συγκεκριμένα το protonVpn το οποίο υπάρχει στον δεύτερο server αλλάζει συνεχώς την ip του μηχανήματος και αυτό προκαλεί πρόβλημα στα API που πρέπει να τρέχουν. Για αυτόν τον λόγο χρησιμοποιείται κι άλλος server προκειμένου να τρέχουν μόνο εργασίες που χρειάζονται σταθερή ip. Ακόμη για την αποθήκευση των δεδομένων έχει στηθεί postgresql στον δεύτερο server και αυτό γιατί ο συγκεκριμένος έχει μεγαλύτερη μνήμη και χωρητικότητα.

Συγκεκριμένα οι δύο server παρέχουν τα παρακάτω χαρακτηριστικά

<u>ο πρώτος</u>:

- RAM: 819M
- SWP: 512M
- CPU: 1 core

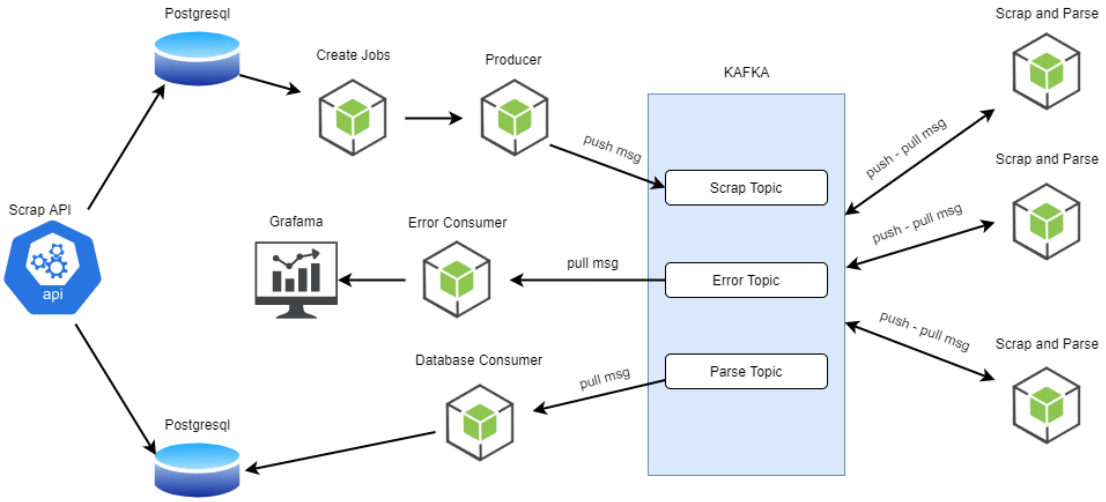
<u>ο δεύτερος</u>:

- RAM: 8GB
- SWP: 512M
- CPU: 4 cores

Λογική αρχιτεκτονική

Το σύστημα αρχίζει από το API το οποίο θα πρέπει να έχει κάνει εγγραφές στην βάση. Οι εγγραφές αυτές θα πρέπει να περιλαμβάνουν όλες τις χρήσιμες πληροφορίες. Στην συνέχεια και παράλληλα ο κόμβος createJobs ελέγχει την βάση και δημιουργεί νέα job στην βάση. Ο Producer τα τοποθετεί στο scrap topic. Κάθε scrap_parse κομβος αρχικά ελέγχει το scrap topic μετα αποθηκεύει στο parse και error topic. Ακολουθεί ο Error consumer ο οποίος δέχεται τα error απο το error topic και τα στέλνει στο grafana API το οποίο με την σειρά του τα τοποθετεί με τον σωστό τρόπο προκειμένου να δημιουργούνται τα κατάλληλα διαγράμματα. Ακόμη ο dataBase Consumer λαμβάνει τα αποτελέσματα και τα αποθηκεύει στην βάση. Τέλος όλα τα δεδομένα μπορούν να προσπελαστούν μέσω του scrape API.

<u>Σχήμα</u>



Εικόνα 3.4.1 Σχήμα προσομοίωσης λογικής αρχιτεκτονικής

3.5 Σχεδιαγραμμα ER βάσης δεδομένων

Στο συγκεκριμένο σύστημα έχει αναπτυχθεί η βάση δεδομένων για την εξυπηρέτηση πολλών και διαφορετικών συστημάτων τα οποία θα μπορούν να επικοινωνούν με το σύστημα μέσω αυτής.

Έτσι λοιπόν έχουν δημιουργηθεί για την κάλυψη αυτής της ανάγκης 5 πίνακες εκ των οποίων οι τέσσεροι έχουν κάποια συσχέτιση μεταξύ τους. Οι πίνακες αυτοί είναι:

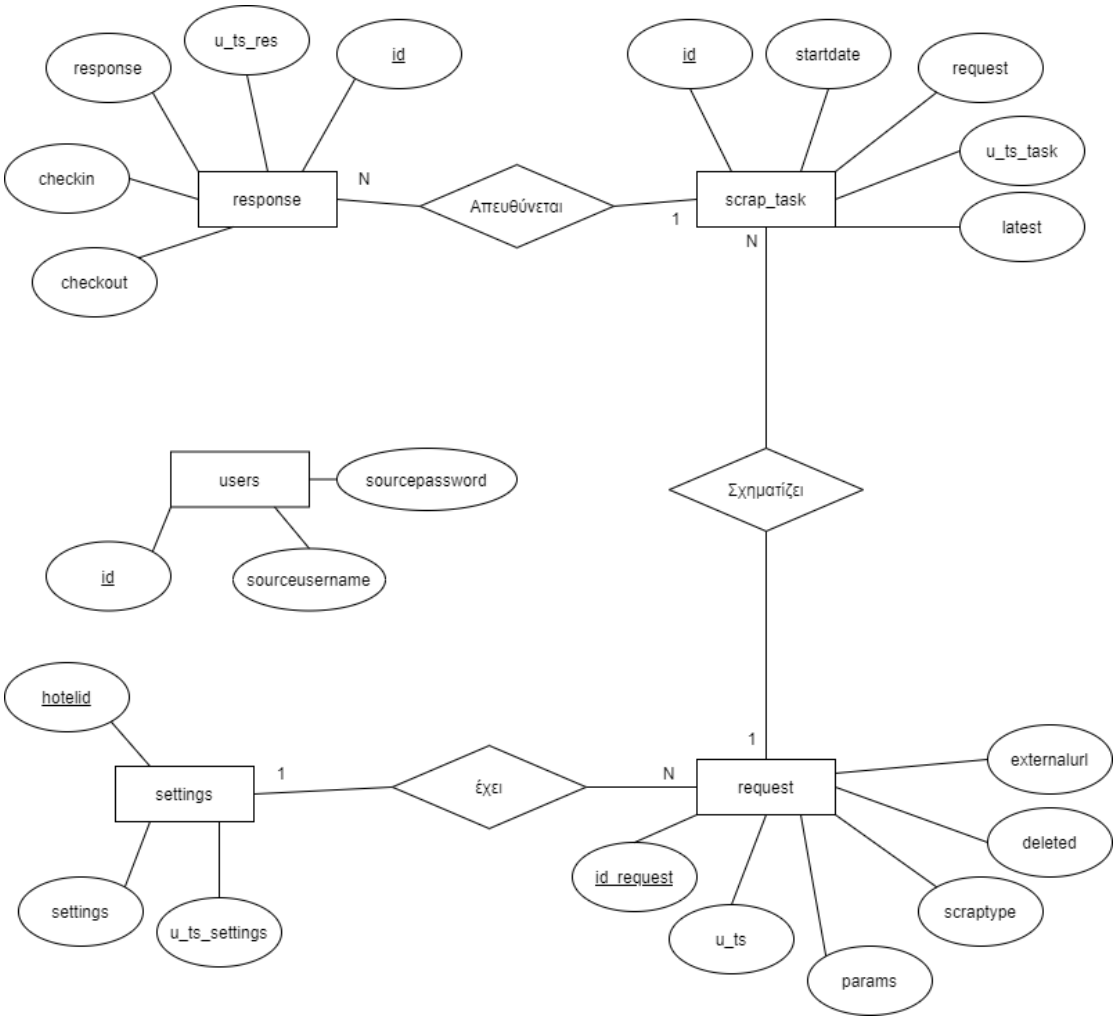
- request: σε αυτόν τον πίνακα αποθηκεύονται τα αιτήματα του χρήστη όπως ακριβώς τα έχει δηλώσει.
- response: περιέχει τα αποτελέσματα μετά από την scraping διαδικασία.
- settings: σε αυτόν τον πίνακα υπάρχουν γενικές ρυθμίσεις για κάθε ξενοδοχείο.
- scrap_task: αποθηκεύονται οι δουλειές που πρέπει να κάνει το σύστημα μας που συνδέονται πληρως με αυτές που ζήτησε ο πελάτης.
- users: αναφέρονται οι χρήστες οι οποίοι μπορεί να είναι admin, user ακόμα και ένα όνομα ενός εξωτερικού συστήματος.

Ανάμεσα στους πίνακες υπάρχουν κάποιες συσχετίσεις προκειμένου να μοιράζεται οι πληροφορία μεταξύ τους. Αυτές είναι:

- 1 πρός 1 ανάμεσα σε response και scrap_task και αυτό γιατί κάθε scrap_task απευθύνεται μόνο σε ένα response.

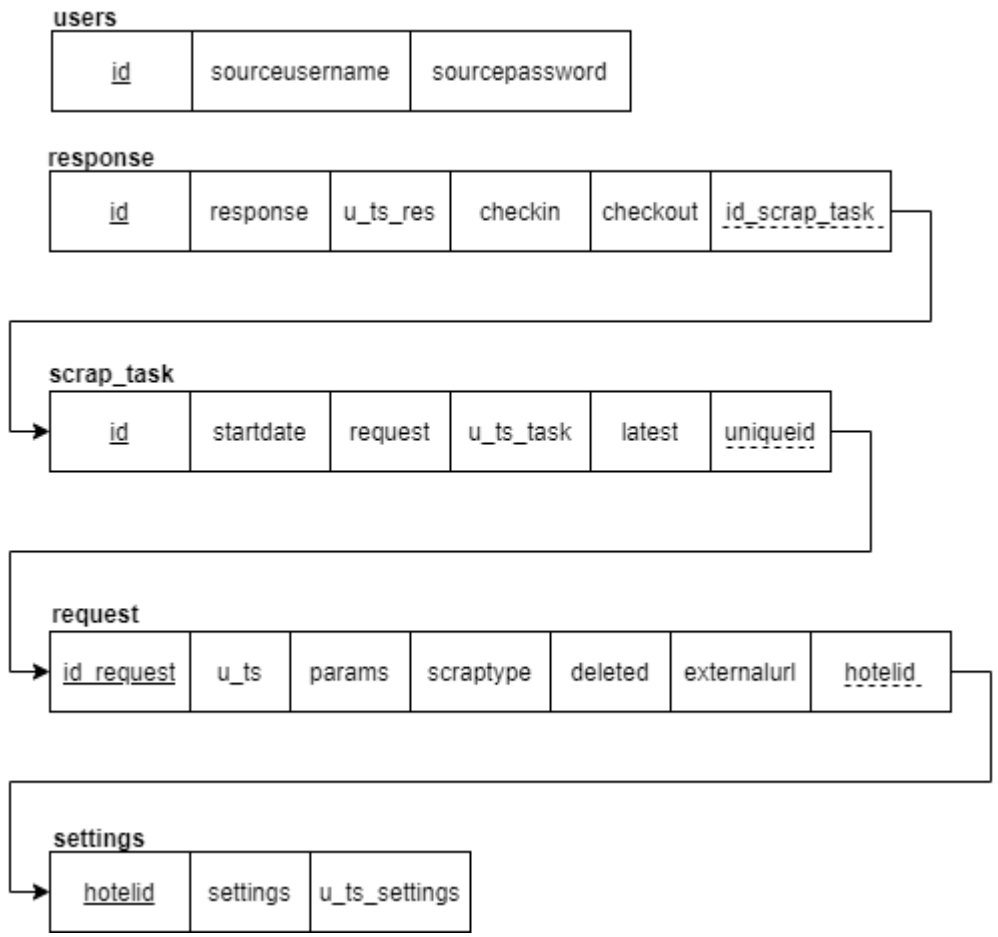
- N-1 ανάμεσα σε scrap_task και request καθώς ένα request σχηματίζει πολλά scrap_task
- N-1 ανάμεσα σε request και settings αφού πολλά request μπορούν να έχουν τις ίδιες ρυθμίσεις

Διάγραμμα Οντοτήτων Συσχετίσεων



Εικόνα 3.5.1 Διάγραμμα Οντοτήτων Συσχετίσεων

Σχεσιακό Μοντέλο



Εικόνα 3.5.2 Σχεσιακό Μοντέλο

Κεφάλαιο 4 - Υλοποίηση συστήματος

4.1 Διαφορετικοί τρόποι υλοποίησης

Για την δημιουργία του συγκεκριμένου συστήματος μελετήθηκαν δύο τρόποι υλοποίησης:

1. Χρησιμοποιεί ένα αρχείο για την κάθε ενέργεια. Οι ενέργειες που μπορούν να γίνουν είναι είτε το scraping είτε ο διαχωρισμός των δεδομένων. Η διαδικασία που ακολουθεί είναι αρχικά ένα αρχείο αναγνωρίζει το κανάλι του αιτήματος (Scrape Task) και ανάλογα καλεί την συνάρτηση scraping. Όταν ολοκληρωθεί καλείται, με είσοδο όλον τον κώδικα html, η αντίστοιχη συνάρτηση η οποία διαχωρίζει τα δεδομένα και τα επιστρέφει στην μορφή JSON. Συνολικά χρειάζονται δύο αρχεία για κάθε κανάλι.
2. Στον συγκεκριμένο τρόπο χρησιμοποιείται ένα αρχείο για όλα τα κανάλια και για όλες τις ενέργειες. Έχει τρία βασικά αρχεία:
    - Το αρχείο των ταγκ περιέχει σε μορφή JSON όλα τα ταγκ που περιέχουν την πληροφορία που θέλουμε. Το συγκεκριμένο λειτουργεί σαν config αρχείο.
    - Το αρχείο που εκτελεί το scraping και τον διαχωρισμό των δεδομένων και επιστροφή τους σε μορφή JSON
    - Και αυτό που είναι υπεύθυνο για τον διαχωρισμο των Task.

Μετα από μελέτη και στους δύο τρόπους υλοποίησης παρατηρήθηκε ότι ο δεύτερος είναι ο καλύτερος καθώς:

- Ευκολότερη προσθήκη νέου καναλιού διότι το μόνο που χρειάζεται είναι να συμπληρωθεί το json των ταγκ. Ενώ στον άλλον ήταν απαραίτητη η δημιουργία δύο αρχείων.
- Γίνεται απευθείας εντοπισμός αλλαγής του markup της σελίδας αφού το tag αφορά συγκεκριμένη πληροφορία και όχι ολόκληρη την html. Στον πρώτο τρόπο το error του Markup εντοπιζόταν στην διαλογή δεν δεδομένων. Έτσι γλυτώνουμε ένα βήμα με τον δεύτερο.
- Ο κώδικας είναι πιο κατανοητος σε νέους προγραμματιστές που ενδεχομένως να γίνουν ενσωμάτωση στο συνολικό project.

- Οι χρόνοι είναι μικρότεροι συγκριτικά μεταξύ των δύο. Ο χρόνος είναι πολυτιμος στο συγκεκριμένο project καθώς ιδανικά προσπαθούμε να δώσουμε την εντύπωση της ζωντανής μετάδοσης των δεδομένων.
- Είναι πιο πιθανό να χρησιμοποιηθεί το συγκεκριμένο σύστημα σε διαφορετικά έργα που αφορούν την συλλογή σχολίων και τιμών. Αυτά μπορεί να είναι:
    - Κριτικές πτήσεων
    - Σχόλια σε social Media (instagram, facebook κλπ)
    - Σχόλια εκδηλώσεων
    - Σχόλια και τιμές εστιατορίων

4.2 Διαχείριση σφαλμάτων και αποφυγή αυτών

Ένα από τα σημαντικότερα κομμάτια κάθε συστήματος είναι αυτό της διαχείρισης σφαλμάτων. Παίζει καθοριστικό ρόλο στην επιτυχή λειτουργία και διακρίνεται κυρίως σε δύο κατηγορίες:

- Το λογικό σφάλμα: αφορά μια ατέλεια που επιτρέπει την εκτέλεση ενός προγράμματος, αλλά δεν του επιτρέπει να εκτελεστεί σωστά. Παραδείγματα λογικών σφαλμάτων είναι ο πολλαπλασιασμός δύο τιμών ενώ θέλαμε να τις διαιρέσουμε ή η έξοδος αποτελεσμάτων πριν αποκτηθεί η κατάλληλη είσοδος
- Το συντακτικό σφάλμα που αφορά την λάθος γραφή ή τεκμηρίωση μιας συγκεκριμένης γλώσσας προγραμματισμού. Παραδείγμα αυτών είναι όταν χρησιμοποιούμε εντολές άλλων γλωσσών σε μια γλώσσα που δεν τις περιλαμβάνει.

Στην τεχνική του scraping υπάρχουν επιπλέον σφάλματα που αφορούν κυρίως την πλοήγηση του bot σε κάποια ιστοσελίδα. Αυτά είναι:

- Το markup error: αντικειμενικά είναι το πιο δύσκολο στον εντοπισμό σφαλμάτων καθώς είναι κάτι που δεν γνωρίζετε από την αρχή και εμφανίζεται μελλοντικά. Επιβάλλει στον προγραμματιστής να μαντέψει πιθανές αλλαγές του κώδικα μιας ιστοσελίδας χτίζοντας έξυπνα τον κώδικα του προβλέποντας όλες τις πιθανές αλλαγές
- Τα status error: εκτός του ότι μας δηλώνουν το είδος τους σφάλματος παράλληλα μας προετοιμάζουν για κάποιο που πρόκειται να εμφανιστεί. Ένα

τέτοιο status code είναι το 429 το οποίο μας δηλώνει ότι έχουμε εκτελέσει αρκετές κλήσεις και σύντομα θα μας εμφανιστεί reCaptcha (too many request)

- Τα proxy error: αφορούν κυρίως αυτά που προέρχονται από την ip και αυτό γιατί ενδεχομένως να είναι υπερφορτωμένη ή να μην υπάρχει πλέον. Ακόμη περιέχει και όλα αυτά που προέρχονται από τον browser όπως είναι μια αποτυχημένη εκκίνηση του.

4.3 Αρχεία

Ολόκληρο το σύστημα αποτελείται από τέσσερις κεντρικούς φακέλους και συνολικά από Ν αρχεία. Ο φάκελος lib περιλαμβάνει τους υποφακέλους browser και utils.

Τα αρχεία που αφορούν το kafka είναι αυτά που διαχειρίζονται ολόκληρο το σύστημα όσον αφορά την εκκίνηση, τον διαμερισμό των task και την αποθήκευση των δεδομένων και των σφαλμάτων. Πιο συγκεκριμένα υπάρχουν στα παρακάτω:

createJobs.js

Το συγκεκριμένο αρχείο είναι η αρχή του συστήματος. Αρχικά δημιουργεί όλα τα topic που χρειάζονται εκτελώντας την συνάρτηση createtopics. Οι υπόλοιπες λειτουργίες του αφορούν την διαχείριση των job. Στην αρχή γίνεται ένα αίτημα στην βάση τύπου select έτσι ώστε να ενημερωθεί για νέα task και στην συνέχεια ανάλογα το task ενημερώνει, διαγράφει ή δημιουργεί κάποιο job. Το νέο job αποθηκεύεται στο scrap topic.

scrap_parse.js

Αρχικά ανοίγει έναν browser έτσι ώστε όλοι οι client να έχουν κοινό προκειμένου να μειωθεί η κατανάλωση της μνήμης. Στην συνέχεια καλείται να διαβάσει όλα τα task από το scrap topic έτσι ώστε να τα στείλει για εκτέλεση. Έπειτα περιμένει να δεχτεί δύο περιπτώσεις είτε ένα Json αρχείο που περιέχει όλα τα αποτελέσματα μετά από την διαδικασία scraping είτε κάποιο είδος σφάλματος. Στην περίπτωση του σφάλματος ελέγχει το είδος και αποφασίζει ανάμεσα σε επιλογές. Αν το error οφείλεται στην ip τότε πρέπει να κλείσει τον browser να αλλάξει ip μέσω του vpn που χρησιμοποιεί το συστημα και μετά να τον ξανα ανοίξει. Σε περίπτωση που είναι άγνωστο το παραβλέπει, το εμφανίζει και συνεχίζει κανονικά την λειτουργία του.

databaseConsumer.js

Εκτελεί ότι είναι απαραίτητο έτσι ώστε να αποθηκευτούν στην βάση όλα τα αποτελέσματα. Με βάση το τύπο reviews ή prices εκτελεί insert αν δεν υπάρχει κάποια εγγραφεί στην βάση με αυτό το id ή update αν υπάρχει. Η συγκεκριμένη λειτουργία πραγματοποιείται με CONFLICT, παράδειγμα τέτοιου query είναι το παρακάτω:

```
client.query(`INSERT INTO response (response, u_ts_res, checkin,
checkout,id_scrap_task)
        values ($1, $2, $3, $4,$5)
        ON CONFLICT (id_scrap_task,checkin,checkout)
          DO UPDATE SET (response, u_ts_res) = ($1, $2)`,
  [response, new Date(), "1", "1", Number(request.id_scrap_task)]
);
```

Ακόμη υπάρχουν και πιο περίπλοκα όπως είναι το παρακάτω το οποίο προσθέτει σε όλα τα review τα τελευταία που εντόπισε:

```
await latestToAllClient.query(`UPDATE response
                    SET response = jsonb_set(response, '{reviews}', response ->
'reviews' || $2)
                    where id_scrap_task = $1
                     and $3 not in (select value -> 'review' ->> 'id' as id
                              from json_array_elements(
                                       (select response -> 'reviews'
                                        from response as b
                                        where id_scrap_task = $1)::json) as reviews)`,
  [Number(request.id_scrap_task), response.reviews[i], response.reviews[i].review.id]
)
```

errorConsumer.js

Περιμένει να δημιουργηθεί κάποιο σφάλμα προκειμένου να ξεκινήσει να λειτουργεί. Μόλις του έρθει κάποιο είναι υπεύθυνο να καλέσει το grafana API προκειμένου να κρατιούνται στατιστικά στοιχεία καθώς επίσης το εμφανίζει στα logs έτσι ώστε να γνωρίζει ο προγραμματιστής ολόκληρη την περιγραφή του error.

Υπάρχουν και τα αρχεία που αφορούν μόνο την την σωστή λειτουργία και την ολοκλήρωση της scraping διαδικασίας.

CreateTopics.js

Περιλαμβάνει συνάρτηση που δημιουργεί τα topic που ζητουνται και είναι απαραίτητα προκειμένου να ξεκινήσει το σύστημα. Η συγκεκριμένη συνάρτηση δέχεται σαν είσοδο έναν πίνακα με τα ονόματα των topic ([error,scrap,parse]) καθώς και έναν πίνακα που θα πρέπει να έχει μέγεθος ίσο με αυτό των topic και θα περιέχει το πλήθος των Partition έτσι ώστε κάθε τιμή να αντιστοιχίζεται με μία του πίνακα topics. Σε περίπτωση που οι είσοδοι είναι [error,scrap,parse] και [1,2,3] τότε το topic error θα έχει 1, το topic scrap θα έχει 2 και το topic parce 3 partition.

```
const createTopics = async (nameTopics, numPartitions) => {
    let allTopics = [];
    await admin.connect();
    for (let i = 0 ; i < nameTopics.length; i++) {
        allTopics[i] = {topic: nameTopics[i], numPartitions:
numPartitions[i]}
    }
    await admin.createTopices({
        topics: allTopics
    });
    await admin.disconnect();
}
```

doScrap.js

Το συγκεκριμένο αρχείο περιλαμβάνει δύο κομμάτια αυτό των τιμών και αυτό των σχολίων. Η κλήση γίνεται γράφοντας “doScrap[‘type’](target,clientId,browser) όπου:

- το type μπορεί να δεχτεί συνολικά δύο τιμές reviews ή prices,
- το target όλα τα στοιχεία του scrapTask
    - Για τις τιμές (prices)

      {

      "urlUniqueId":Integer, //id του κεντρικού συστήματος που το χρησιμοποιεί

      "externalUrl":String, //Συγκεκριμένο url

"bocHotelId":String, //Ποιο ξενοδοχείο
"type":String, //Ο τύπος (reviews ή prices)
"parameters":{
"depth":integer, //Για ποσο βάθος
"startDate":String, //Από ποιά ημερομηνία θα ξεκινήσει
"childrenAge":array, //Πίνακας που περιέχει μόνο τις ηλικίες των παιδιών
"adults":integer, //Το πλήθος των ενήλικων
"nights":integer //Ανα πόσες μέρες θα κοιτάει.
},
"active":Boolean, //αν είναι ακόμα ενεργό το συγκεκριμένο task
"frequency":String, //Η συχνότητα που αφορά τις τιμές
"frequencyRare": String //null για τις τιμές
}

- Για τα σχόλια (review)

{
"urlUniqueId":Integer //id του κεντρικού συστήματος που το χρησιμοποιεί
"externalUrl":String, //Συγκεκριμένο url
"bocHotelId":String, //Ποιο ξενοδοχείο
"type":String, //Ο τύπος (reviews ή prices)
"parameters":Json, //json με όλες τις παραμέτρους. Για τα review καμία
"active":boolean, //αν είναι ακόμα ενεργό το συγκεκριμένο task
"frequency":String, //Η συχνότητα που αφορά όλα τα review
"frequencyRare": String //Η συχνότητα που αφορά τα τελευταία σχόλια (latest review).
}

- το clientId είναι το μοναδικό id κάθε client και ο browser που είναι ο ίδιος ο browser.

Η λειτουργία του αρχείου είναι κυρίως ο διαχωρισμός των task σε prices και review καθώς και τον σωστό εντοπισμό των error αφού περιλαμβάνει το κεντρικό try catch. Επίσης στο σώμα του σχηματίζει το συνολικό αποτέλεσμα (response) το οποίο αποτελείται από το αίτημα (request) και τα αποτελέσματα του scraping. Όλο το response είναι σε μορφή json και έχει την μορφή:

review response

```
{
        request:{
        },globalScore:{
        },reviews:[
        ]
}
```

prices response

```
{
        request:{
        },prices:[
        ]
}
```

scrapReview.js

Το αρχείο που υλοποιεί το scraping για τα review. Δέχεται σαν είσοδο:

- τα tag των review
- την σελίδα (page) που έχει δημιουργηθεί στο αρχείο doScrap.js
- το url που πρέπει να γίνει το scraping
- το channel όπου είναι το κανάλι που πρέπει να επισκεφτεί το bot
- την startDate που χρησιμεύει στην περιπτωση που το σύστημα θέλει να εντοπίσει τα τελευταία σχόλια (latest). Έτσι το bot κοιτάει από την συγκεκριμένη ημερομηνία και μετά να διαβάζει. Υπάρχει και η περίπτωση όλων των review όπου η τιμή είναι Null.
- hotelId το οποίο είναι το μοναδικό id κάθε ξενοδοχείου
- την latest που δηλώνει αν το συγκεκριμένο task αφορά scraping των τελευταίων σχολίων ή όχι.

Η λειτουργία του ξεκιναει με την φόρτωση της συγκεκριμένης διεύθυνσης (url). Στην συνέχεια ελέγχει μέσω του status code αν υπάρχει κάποιο πρόβλημα με

την σελίδα. Έπειτα ετοιμάζει την σελίδα για να ξεκινήσει η συλλογή πληροφοριών. Συγκεκριμένα κάνει scroll, πατάει κουμπιά και εκχωρεί τιμές όπου είναι απαραίτητο. Αφού γίνουν όλα τα παραπάνω αρχίζει η κεντρική λειτουργία και με βάση τα tag ελέγχεται αν υπάρχει recaptcha και γίνεται η συλλογή δεδομένων. Στο τέλος δημιουργεί ένα μοναδικό κρυπτογραφημένο id συνδυάζοντας στην περίπτωση που έχει βρεθεί externalId τα hotelId,externalId,# και channel ενώ στην περίπτωση που έχει δεν έχει βρεθεί συνδυάζει τα hotelId,#,όνομα και τοποθεσία του χρήστη, την ημερομηνία που έχει γίνει το σχόλιο, το αρνητικό σχόλιο , το θετικό σχόλιο, # και το όνομα του καναλιού και μετά ότι έχει εντοπιστεί επιστρέφεται στο αρχείο doScrap.js.

scrapPrices.js

Το αρχείο αφορά το scraping των τιμών και δέχεται σαν είσοδο τα παρακάτω:

- τα tag των τιμών
- την σελίδα που έχει ανοίξει για αυτό το tag, το αρχείο doscrap.js
- το url που πρέπει να φορτωθεί έτσι ώστε να ξεκινήσει η διαδικασία.
- το diffdays που είναι τύπου integer και περιέχει την διαφορά ανάμεσα σε checkin και checkout
- το persons: το πλήθος των ατόμων συμπεριλαμβάνοντας και τα παιδιά.
- το nights: το πλήθος των ημερών που θέλουν να διαμείνουν.
- το checkin: την ημερομηνία έναρξης διαμονής
- το checkout: την ημερομηνία λήξης διαμονής
- το channel: το κανάλι που προέρχεται το Url
- το hotelId: το μοναδικό id του ξενοδοχείου

Η λειτουργία είναι ίδια με αυτή του αρχείου scrapReview.js η διαφορά τους είναι η πληροφορία και ο τρόπος διαλογής της.

tagsPrices.json

Το αρχείο που περιλαμβάνει τα tag για τις τιμές. Συγκεκριμένα:

- loadType: χαρακτηρίζει τον τύπο φόρτωσης της σελίδας και μπορεί να δεχτεί 3 τιμές , null - 1 - 2
    - Η τιμή null αφορά το default load. Έχει σαν όριο 30 δευτερόλεπτα πριν εμφανίσει κάποιο σφάλμα.
    - Η τιμή 1 αφορά το load που αφιερώνει απεριόριστα δευτερόλεπτα για την μια σελίδα.

    - Η τιμή 2 για το load που περιμένει να φορτώσουν όλα τα αρχεία. Και σε αυτή την περίπτωση το όριο είναι default στα 30 δευτερόλεπτα
- centerTag: είναι το κεντρικό tag που περιέχει όλες τις πληροφορίες που χρειαζόμαστε.
- tagPerRoom: το tag που αφορά την κάθε προσφορά δωματίου ξεχωριστά.
- boardType: το tag που πρέπει να δείχνει στον τίτλο της κάθε προσφοράς.
- roomType: αυτό που αφορά τον τίτλο του δωματίου.
- checkin: αφορά
- flag: βοηθάει στον διαχωρισμό δύο περιπτώσεων εντοπισμού των ανέσεων (features) και του τύπου δωματίου (roomType)
- bedType: περιέχει έναν πίνακα με δύο tag. Στην θέση μηδέν υπάρχει αυτό που αναφέρει όλο το κείμενο και στην θέση ένα προσδιορίζεται ο κάθε τύπος κρεβατιού ξεχωριστά.
- price: tag που αναφέρεται στην τιμή κάθε δωματίου.
- features: tag για τον εντοπισμό των ανέσεων.
- conditions: tag για την κατάσταση του δωματίου. Συνήθως η τιμή που επιστρέφεται είναι αν το ξενοδοχείο έχει δωρεάν ακύρωση.
- reCaptcha: αφορά τον εντοπισμό ενός reCaptcha και περιλαμβάνει 2 επιμέρους tag:
    - tag: προσδιορίζει το κείμενο που περιέχει
    - content: είναι το κείμενο που περιμένουμε να μας εμφανιστεί.
- clickButtons: περιέχει δύο tag:
    - flag: αν θέλουμε να πατηθεί κάποιο κουμπί
    - tagButton: το tag του κουμπιού
- minStay: αφορά τον εντοπισμό του κάτου ορίου διαμονής και περιλαμβάνει δύο tag:
    - tag: περιέχει το tag του κειμένου που γράφει για το κάτω όριο
    - findSomething: πίνακας που περιέχει τα αλφαριθμητικά που περικλειουν τον αριθμό που θέλουμε να αποθηκεύσουμε

tagsReviews.js

Αρχείο που περιλαμβάνει τα tag για των εντοπισμό των πληροφοριών των σχολίων. Πιο αναλυτικά περιλαμβάνει:

- loadType: έχει την ίδια λειτουργία με αυτή του αρχείου tagsPrices

- hotel_title: είναι το tag που περιέχει τον τίτλο κάθε ξενοδοχείου
- flagReviewUrl: Δηλώνει ποια κανάλια μπορούν να αλλάξουν καρτέλα από το url
- strReviewUrl: Πινακάς που περιέχει τα καμμάτια του url που πρέπει να αλλάξουν καθώς επίσης και αυτά που θα τα αντικαταστήσουν.
- scrollFlag: Δηλώνει ποια σε ποια κανάλια πρέπει να γίνει scroll για να εμφανιστούν όλα τα σχόλια.
- reCaptcha: αποτελείται από το tag και αλφαριθμητικό που βρίσκεται μέσα του
- firstButton: περιέχει το tag του πρώτου κουμπιού που είναι απαραίτητο για να εμφανιστούν όλα τα σχόλια.
- sort: δηλώνει με ένα flag αν υπάρχει κουμπί το οποίο ταξινομεί στο σαιτ όλα τα δεδομένα που θέλουμε να "τραβήξουμε" . Επίσης περιέχει το tag και την επιλογή (value) που θέλουμε.
- moreSecond: έχει την τιμή true εάν σε κάποιο κανάλι πρέπει να περιμένει το bot περισσότερα δευτερόλεπτα.
- nextButton: περιέχει το tag του κουμπιού επόμενο σε περίπτωση που περιέχει σελιδοποίηση (pagination).
- loader: το tag του loader εφόσον υπάρχει.
- btnMoreReview: κουμπί που πρέπει να πατηθεί για να εμφανιστούν κι άλλα δεδομένα.
- btnAllLanguages: κουμπί που πρέπει να πατηθεί για να μην γίνεται μετάφταση του κειμένου.
- btnReadMore: κουμπί που πρέπει να πατηθεί για να εμφανιστή ολόκληρη η πληροφορία ενός δεδομένου
- globalScore: Json που περιέχει ότι χρειάζεται για να έχουμε όλα τα δεδομένα της συνολικής βαθμολογίας ενός ξενοδοχείου. Συγκεκριμένα εχει το tag του συνολικού βαθμού, το κεντρικό ταγκ και το αλφαριθμητικό των επιμέρους βαθμολογιών . Επίσης περιέχει και την επιλογή που ο βαθμός βρίσκεται σε κάποιο attribute.
- review: Json που περιέχει όλα τα tag για μια κριτική. Πιο αναλυτικά:
    - externalId: στην περίπτωση που το κανάλι περιέχει μοναδικό id
    - centerTag: το tag που περιέχει ολόκληρο το σχόλιο

- source: Json Που περιέχει ένα flag το οποίο δηλώνει αν υπάρχουν στο κανάλι σχόλια από άλλο. Το tag που περιέχει το εξωτερικό κανάλι και το αλφαριθμητικό που θέλουμε να αφαιρέσει
- title: το tag του τίτλου κάθε σχολίου
- createReviewTimestamp: JSON περιέχει όλα τα απαραίτητα για να βρεθεί η ημερομηνία που δημιουργήθηκε το σχόλιο.
- reviewScore: για τον εντοπισμό της ξεχωριστής βαθμολογίας κάθε χρήστη.
- flagposneg: αν υπάρχουν θετικά και αρνητικά σχόλια
- pos: σε περίπτωση που το flagposneg ειναι true τότε περιέχει το tag του θετικού σχολίου και αντίστοιχα το neg περιέχει το tag του αρνητικού σχολίου
- posNeg: σε περίπτωση που το flagposneg είναι false αφορά το κείμενο που περιέχει και το θετικό και το αρνητικό σχόλιο. Έτσι δηλώνουμε το κείμενο που πρέπει να αφαιρεθεί από το θετικό σχόλιο στο Liked και για το αρνητικό στο disliked
- tip: αφορά την συμβουλή που έχει δώσει ο χρήστης
- tripReason: το tag που περιέχει τον λόγο που ταξίδεψε
- reviewContext: το tag που αναφέρει πόσες μέρες έμεινε ο πελάτης
- reviewLanguageCode: αφορά το tag και το attribute που περιέχουν το συμβολισμό της γλώσσας. π.χ. el, eng
- review: tag που περιέχει το κύριο σχόλιο της κρητικής.
- thumbsUp: περιέχει το πλήθος των θετικών αντιδράσεων.
- stars: το πλήθος των αστεριών που έχει βαθμολογήσει το ξενοδοχείο
- reviewer: συλλέγονται και πληροφορίες που αφορούν τον χρήστη - πελάτη όπως είναι η διεύθυνση του προφίλ του, το όνομα του, η τοποθεσία του, πόσα σχόλια έχει κάνει, πόσες φορές έχουν αντιδράσει θετικά σε αυτόν, πόσους followers έχει, πόσους ακολουθεί ο ιδιος, από πότε είναι χρήστης του καναλίου και το κωδικό της χώρας του.

timestampTags.json

Αρχείο που αφορά αποκλειστικά το κανάλι της google και πως θα μπορέσουμε να φέρουμε την ημερομηνία που παρουσιάζει σε μορφή YYYY-MM-DD. Το κανάλι της google εμφανίζει προσεγγιστικά την ημερομηνία σε

μορφή νούμερο ή α κενό hour/seconds/day/month/year/ κενο ago. Έτσι προκύπτει το παρακάτω json:

```
{
 year:"FullYear",
 years:"FullYear",
 month:"Month",
 months:"Month",
 day:"Date",
 days:"Date",
 hour:"Hours",
 hours:"Hours",
 minute:"Minutes",
 minutes:"Minutes",
 second:"Seconds",
 seconds:"Seconds"
}
```

4.4 Συναρτήσεις

buildUrlPrices.js και buildUrlReviews.js

Είναι αρχεία που έχουν καθοριστικό ρόλο στην λειτουργία του συστήματος καθώς διαμορφώνουν σωστα την διευθυνση. Στο αρχείο των τιμών γίνεται μεγαλύτερη διαμόρφωση γιατί οι περισσότεροι παράμετροι περνιούνται στην διεύθυνση. Τέτοιοι παράμετροι είναι το check in, το checkout κλπ. Στο αρχείο των σχολίων αφαιρειται η ασήμαντη πληροφορία έτσι ώστε να αυξηθεί η ταχύτητα φόρτωσης της σελίδας. Και τα δύο αρχείου αποτελούνται από συναρτήσεις σε πλήθος όσο αυτό των καναλιών. Η κλήση του γίνεται buildUrlPrices[channel] και buildUrlReviews[channel] αντίστοιχα. Έτσι ανάλογα το κανάλι εκτελείται η ανάλογη συνάρτηση.

findSomething()

Συναρτηση που ξεχωρίζει ένα αλφαριθμητικό από ένα αρκετά μεγάλο αλφαριθμητικό. Δέχεται σαν είσοδο τέσσερις παραμέτρους:

- startStr: Ο τύπος του είναι String και περιέχει την αρχή του κειμένου που θέλουμε να εντοπίσουμε.
- finishStr: Ο τύπους του είναι string και περιέχει το τέλος του κειμένου που θέλουμε να εντοπίσουμε.
- json: περιλαμβάνει ολόκληρο το String Που θέλουμε να εξετάσουμε
- x: Είναι τύπου integer και ο ρόλος του είναι σε περίπτωση που δεν γνωρίζουμε το String το αρχικό αλλά γνωρίζουμε από πόσα γράμματα αποτελείται τότε βάζουμε σε αυτή την επιλογή το πλήθος των γραμμάτων.

Η λειτουργία του ξεκινάει αρχικά κοιτώντας αν υπάρχει το startStr σε ολόκληρο το string. Στην συνέχεια βρίσκει το πλήθος των γραμμάτων μέχρι να ξεκινήσει το κείμενο που θέλουμε και σπάει το αρχικό string σε δύο. Και τέλος από αυτό που εντοπίστηκε αφαιρείται και το finishStr. Δείγμα κώδικα φαίνεται παρακάτω:

```
returnStr = json.substring(start, start + first.indexOf(finish));
```

getRandomInt()

Συνάρτηση που επιστρέφει κάποιο τυχαίο αριθμό από το 0 μέχρι κάποιο άνω όριο. Δέχεται σαν είσοδο το άνω όριο του διανύσματος τύπου Integer.

isValidUrl()

Έχει σαν είσοδο το url και ελέγχει για το αν το pattern του url είναι σωστό και αν αποτελεί διεύθυνση κάποιου διαθέσιμου καναλιού. Αρχικά κοιτάει αν στο String αυτό περιέχει τις τιμές google, hotels., expedia, tripadvisor, airbnb, booking και agoda και στην συνέχεια κάνει match το url με το (/(http(s)?:\/\/.)?(www\.)[-a-z]{2,256}\.[a-z]{2,6}/g) που σημαίνει ότι το url θα πρέπει να ξεκινάει με http και ίσως s στην συνέχεια θα πρέπει να έχει δύο σλας (/) τρία (w) μια τελεία (.) και το όνομα καναλιού που μπορεί να είναι από 2 έως 256 γράμματα. Επιπλέον μετά από το κανάλι θα πρέπει να περιέχει τελεία (.) και 2 έως 6 γράμματα κυρίως για (.com .gr .gov.gr). Στο τέλος η τιμή που θα επιστραφεί είναι true ή false και αποτελεί συνδυασμό (AND) των δύο παραπάνω ελέγχων.

findIndexArr()

Συνάρτηση που χρησιμοποιείται στο αρχείο createJobs.js. Δέχεται σαν είσοδο έναν πίνακα και ένα νούμερα. Έτσι εντοπίζει σε σχέση με τον πίνακα το νούμερο μέσα στον πίνακα και στην συνέχεια επιστρέφει την θέση (Index). Σε περίπτωση που δεν εντοπίσει κάτι επιστρέφει το νούμερο -1.

changeIp()

Η συγκεκριμένη συνάρτηση καλείται μόνο μια φορά σε ολόκληρο το σύστημα. Παρόλα αυτά είναι μια από τις πιο βασικές καθώς ολοκληρώνει την αλλαγή ip όπου είναι αυτό απαραίτητο. Χρησιμοποιεί την βιβλιοθήκει exec έτσι ώστε να μπορεί να γράψει σε γραμμή εντολών. Αρχικά ξεκινάει κάνοντας αποσύνδεση στην προηγούμενη και αν δεν υπάρξει κάποιο πρόβλημα συνεχίζει χωρίς να εμφανίσει κάτι ενώ στην αντίθετη περίπτωση εμφανίζεται το μήνυμα “Disconnect Error”. Συνεχίζει προσπαθώντας να κάνει συνδεση σε μια άλλη ip σε περίπτωση που πάνε όλα καλα τότε επιστρέφει true. Στην αντίθετη περίπτωση κοιτάει αν το σφάλμα είναι αυτό που αναφέρεται στο ότι υπάρχει ήδη μια Ip που τρέχει και αν ναι τότε πηγαίνει και σβήνει ότι έχει το αρχείο resolv.conf και γράφει search members.linode.com και nameserver 8.8.8.8 έτσι όταν γίνει νέο connect σε νέα ip να μην υπάρχει σίγουρα άλλη. Το protonVPN πριν από κάθε connect ελέγχει τον nameserver αν είναι αυτός που επιθυμεί να συνδεθεί. Αν δεν είναι τον αλλάζει αυτόματα. Ολη αυτή η διαδικασία μπορεί να γίνει μέχρι 5 φορές καθώς μετα από τις 5 φορές θεωρούμε ότι υπάρχει πρόβλημα είτε αυτό αφορά το σύστημα είτε το protonVPN. Ο κώδικας της συνάρτησης φαίνεται παρακάτω:

```
const changeIP = async () => {
    let str;
    let countOfErr = 0;
    try {
        do {
            countOfErr++;
            try {
                await exec('protonvpn d', function (response) {
                    callback(response);
                });
            } catch (e) {
                console.log(e);
```

```
                console.log("Disconnect Error");
            }
            try {
                str = await exec('protonvpn c -r', function
(response) {
                    callback(response);
                });
                if (str.stdout) {
                    if (str.stdout.indexOf("Could not terminate
OpenVPN process") !== -1) {
                        throw {"stdout": "There is already a VPN
connection running."};
                    }
                }
            } catch (e) {
                if (e.stdout || String(e).indexOf('spawn ENOMEM')
!== -1) {
                    if (String(e).indexOf('spawn ENOMEM') !== -1 ||
e.stdout.indexOf("error connecting to the ProtonVPN API") !== -1 ||
e.stdout.indexOf("There is already a VPN connection running.") !==
-1 || e.stdout.indexOf("Could not terminate OpenVPN process") !==
-1) {
                        str = {"stdout": "Connected!"}
                        try {
                            await exec("swapoff -a && swapon -a");
                        } catch (err) {
                        }
                        try {
                            await exec('echo search
members.linode.com > /etc/resolv.conf', function (response) {
                                callback(response);
                            });
                            await exec('echo nameserver 8.8.8.8 >>
/etc/resolv.conf', function (response) {
                                callback(response);
```

```
                        });
                    }catch (e){
                        console.error(e);
                        console.error('eimai kai edw mesa');
                    }
                    try {
                        str = await exec('protonvpn c -r',
function (response) {
                            callback(response);
                        });
                    } catch (e) {
                        str = {"stdout": "error"};
                        console.log(e);
                    }
                } else {
                    str = {"stdout": "error"};
                }
            } else {
                str = {"stdout": "error"};
            }
        }
        if (countOfErr === 5) {
            break;
        }
    } while (str.stdout.indexOf('Connected!') === -1);
} catch (e) {
    console.log(e);
}
};
```

scrollToEnd()

Η συνάρτηση αυτή αφορά αποκλειστικά το scroll της google. Το πρόβλημα ήταν μεγάλο καθώς το συγκεκριμένο κανάλι παρακολουθεί την κίνηση του scroll κάθε χρήστη. Έτσι δημιουργήθηκε ένα scroll με συγκεκριμένο βήμα, με βάση το βήμα κανονικού χρήστη. Αρχικα δέχεται την σελίδα (page) έτσι ώστε να μπορεί να χρησιμοποιήσει συγκεκριμένες λειτουργίες και το centerTag κάθε σχολίου. Έτσι κάθε

φορά η ταχύτητα του scrolling εξαρτάται από το πλήθος των σχολίων και όταν φτάσει η μπάρα στο τέρμα επιστρέφει αυτόματα μετά το τέλος της φόρτισης σε κάποιο default σημείο. Εάν η μπάρα δεν επιστραφεί 1000 φορές τότε σημαίνει ότι δεν υπάρχουν άλλα review.

```
const scrollToEnd = async (page, tag) => {
	let returnValue = false;
	try{
		returnValue = await page.evaluate(async (tag) =>{
			let distance  = 10;
			let count = 0;
			let reviewscount = 0
			let previousReview = 0;
			while (true) {
				previousReview = reviewcount;
				reviewscount = Array.from(await
document.querySelectorAll (tag)).length;
				distance = 10 * (String(reviewscount).length -
1 );

				if (previousReview === reviewscount){
					count++;
				}else{
					count = 0;
				}
				await window.scrollBy(0,distance);
				await new Promise(function (resolve) {
					setTimeout (resolve,1);
				});
				if (count === 1000){
					break;
				}
			}
			return;
		},tag);
		const load = Date.now() - start;
	} catch (e) {
	}
};
```

createResponseJson()

Διαμορφώνει το τελικό json σε μορφή String και έχει σαν παραμέτρους το αρχικό αίτημα (request) και το τελικό αποτέλεσμα με τα δεδομένα που συλλέχθηκαν. Έτσι η τελική κατάσταση είναι:

```
JSON.stringify({
      request,
      response
});
```

reverseStr()

Η λειτουργία του είναι απλή αλλά παράλληλα τόσο χρήσιμη στην διαμόρφωση του url (αρχεία BuildUrl). Δέχεται σαν είσοδο ένα αλφαριθμητικό και το επιστρέφει αντεστραμμένο. Ένα παράδειγμα είναι: είσοδος “hi” έξοδος “ih”.

```
const reverseStr = function (str) {
    let splitString = str.split("");
    let reverseArray = splitString.reverse();
    str = reverseArray.join("");
    return String(str);
};
```

clickButtons()

Ακόμα μια πολύ σημαντική συνάρτηση για το scraping κυρίως των σχολίων. Η λειτουργία της είναι να πατάει όλα τα κουμπιά που έχουν το tag που έχει δεχτεί σαν είσοδο. Στην περίπτωση των review όταν ένα σχόλιο είναι μεγάλο κάθε κανάλι το σπάει και προσθέτει κουμπί “read more” έτσι η συγκεκριμένη καλείται και τα πατάει όλα.

createCheckInOut()

Διαμορφώνει την ημερομηνία με βάση το pattern YYYY-MM-DD και δέχεται σαν είσοδο το αίτημα και την απόκλιση που πρέπει να έχει το check in με βάση το την ημερομηνία έναρξης. Αρχικά κάνει τύπου date το check in και το checkout έτσι ώστε να μπορεί να κάλέται συναρτήσεις του συγκεκριμένου τύπου. Έτσι καλεί τις

getDate(), getMonth(), getFullYear() και με βάση αυτά τα 3 χτίζει μια νέα ημερομηνία με το pattern που έχει επιλεχθεί. Στην συνέχεια υπολογίζει την διαφορά μεταξύ των δύο ημερομηνιών και προσθέτει το 0 σε όσα από τα νούμερα είναι μικρότερα του 10. Τέλος επιστρέφει έναν πίνακα όπου περιέχει τις δύο ημερομηνίες και την διαφορά τους.

Υπολογισμός διαφοράς ημερομηνιών

```
const difftime = Math.abs(date2 - date1);
const diffDays = Math.ceil (diffTime / (1000 * 60 * 60 * 24));
```

authentication()

Συνάρτηση που ελέγχει την εγκυρότητα κάθε χρήστη με βάση το hotelId και το μοναδικό token που του έχει δημιουργηθεί κατά την πρώτη του εισαγωγή. Πρώτα από όλα κοιτάει αν το authcode είναι null ή undefined και επιστρέφει "Empty Token" αλλιώς προσπαθεί να κάνει αποκρυπτογράφηση με το private key. Το private key δημιουργείται από κάθε οργανισμό και στο τέλος προστίθεται το “#hotelid” έτσι ώστε για κάθε χρήστη να υπάρχει μοναδικός κωδικός. Στην περίπτωση που βγάλει κάποιο error τότε είτε κάποιος έβαλε ψεύτικο token είτε το token έχει λήξει. Στην περίπτωση που γίνει σωστά η αποκρυπτογράφηση τότε κοιτάει αν υπάρχει ο χρήστης αυτός στο redis αν δεν υπάρχει τότε γυρίζει "unknown token" ενώ αν είναι ελέγχει την ημερομηνία λήξης και στη συνεχεια συγκρίνει το token που υπάρχει στον header με αυτό που υπήρχε στο redis αν είναι ίδια ανανεώνει την ημερομηνία και επιστρέφει "Correct Token" αλλιώς επιστρέφει 'Wrong Token'.Στη περίπτωση που έχει λήξει επιστρέφει "token is expired".

Σε όλα τα μηνύματα εκτός από το correct token το frontEnd κάνει refresh για να φέρει ο controller κάθε συστήματος που το χρησιμοποιεί νέο authentication code.

έλεγχος στο redis αν υπάρχει αυτός ο χρήστης

```
await clientRedis.get(`${bocHotelId}#${info.username}`);
```

Αποθήκευση του authentication code στο redis

```
await clientRedis.set(`${bocHotelId}#${info.username}`,
JSON.stringify({authCode,exp: dateNow + (60 * 60)}));
```

Αποκρυπτογράφηση του authentication code

```
jwt.verify(authCode, privateKey);
```

Κεφάλαιο 5 - Συμπεράσματα

5.1 Ιδέες για επέκταση

Παρατηρώντας αρκετούς χρήστες κατά την χρήση του διαδικτύου θα μπορούσε να καταλάβει κανείς ότι το μεγαλύτερο ποσοστό αναζητά την ευκολη πληροφόρηση, δηλαδή συγκεκριμένο σύστημα που θα του δώσει χωρίς "κόπο" ακριβώς αυτό που αναζητά. Ο χρήστης θα επιθυμούσε να μην αναζητούσε καμία πληροφόρηση και το σύστημα να τον ενημερώνει κατευθείαν. Έτσι λοιπόν υπάρχει η δυνατότητα εξέλιξης του συστηματος δημιουργώντας τα παρακάτω υποστήματα.

Σύστημα καιρού

Θα μπορούσε να αναπτυχθεί ένα υπο σύστημα το οποία θα έκανε συλλογή πληροφοριών όσον αφορά τις καιρικές συνθήκες στην τοποθεσία του ξενοδοχείου. Έτσι λοιπόν ο χρήστης θα είχε μια πλήρη ενημέρωση από μια σειρά διάσημων εταιρειών που ο ιδιος ο προγραμματιστής έχει αναζητήσει και εχει βρει. Αυτό θα βοηθούσε τον ξενοδόχο να προετοιμαστεί ως προς το πλήθος ατόμων που μπορούν να τον επισκεφτούν. Ακόμα θα μπορούσε σε περίπτωση που ο καιρός δεν είναι καλός να χαμηλώσει αρκετά τις τιμές του προκειμένου όσοι επισκεφτούν την συγκεκριμένη τοποθεσία να διαλέξουν το συγκεκριμένο ξενοδοχείο.

Συστημα Εκδηλώσεων

Είναι πολλές οι περιοχές ανά τον κόσμο που διοργανώνουν εκδηλώσεις και προσελκύουν χιλιάδες κόσμο. Υπάρχουν περιπτώσεις που οι ξενοδόχοι δεν γνωρίζουν κάποιο συμβάν και έτσι δεν είναι προετοιμασμένοι κατάλληλα για να υποδεχτούν τους πελάτες του.

Έτσι θα ήταν αρκετά χρήσιμο να δημιουργηθεί ένα σύστημα το οποίο θα ενημερώνει τα ξενοδοχεία για κάποιες εκδηλώσεις που πρόκειται να γίνουν. Το σύστημα θα επισκέπτεται διάφορες μεγάλες σελίδες που περιλαμβάνουν την πληροφορία και με τεχνικές scraping θα αποθηκεύουν την χρήσιμη. Ο χρήστης θα ειδοποιείται για το μέγεθος του event και έτσι θα μπορεί να προσαρμοστεί π.χ σε μια πολύ μεγάλη εκδήλωση που είναι δεδομένο ότι τα ξενοδοχεία θα έχουν πληρότητα ο

ξενοδόχος δεχοντας ειδοποίηση από το σύστημα μπορεί να αλλάξει την τιμή προς τα πάνω προκειμένου να έχει περισσότερα κέρδη.

Σύστημα μεγαλύτερης ανάλυσης πληροφοριών

Η τεχνική εξόρυξης δεδομένων μαζεύει χρήσιμες πληροφορίες. Παρόλα αυτά η πληροφορία μπορεί να γίνει ακόμη πιο συγκεκριμένη με χρήση τεχνητής νοημοσύνης καθώς έχουμε την δυνατότητα να επεξεργαστούμε την πληροφορία και στο τέλος να δώσουμε στον χρήστη τα κυριότερα σημεία. Συνήθως τέτοιες πληροφορίες είναι είτε αν το ξενοδοχείο είναι αγαπητό, αν τους έχει προσφέρει ότι επιθυμούσαν κ.λ.π. Αυτό είναι κάτι που θέλει να το ξέρει και είναι αυτό που ξεχωρίζει τα ξενοδοχεία μεταξύ τους.

5.2 Ιδέες για άλλες εφαρμογές

Εφαρμογή πληροφοριών για κάθε προϊόν

Στην τωρινή εποχή έχει παρατηρηθεί ότι όλο και περισσότεροι χρήστες θέλουν να αγοράζουν από το διαδίκτυο. Αυτό είναι κάτι που υπήρχε στο παρελθόν όμως πλέον παρατηρείτε κάτι διαφορετικό. Ο χρήστης εκτός του ότι επιλέγει να αγοράζει από το διαδίκτυο παράλληλα προτιμάει προϊόντα που προέρχονται από χώρα διαφορετική από αυτή που ζει. Επομένως ο πελάτης μπαίνει σε μια διαδικασία αναζήτησης στην οποία πρέπει να εντοπίσει τα καταλληλα μαγαζιά, την διαθεσιμότητα του προϊόντος, την καλύτερη τιμή για αυτόν καθώς και τον χρόνο παράδοσεις. Η εφαρμογή πληροφοριών για κάθε προϊόν μπορεί να δημιουργηθεί για να βοηθήση αυτή την ενέργεια.

Ο προγραμματιστής καλείται να εντοπίσει τα κορυφαία εμπορικά μαγαζιά ανά τον κόσμο. Στην συνέχεια να δημιουργήσει μια εφαρμογή η οποία θα συλλέγει πληροφορίες για κάποιο προϊόν που θα ήθελε ο πελάτης. Έτσι κατα την είσοδο του στο σύστημα θα πρέπει να επιλέξει το προϊόν που ενδιαφέρεται να αγοράσει και στην συνέχεια να προσδιορίσει πληροφορίες για αυτό προκειμένου να βοηθηθεί η εφαρμογή και να του εμφανίσει ακόμα πιο στοχευμένες πληροφορίες. Μετά το βήμα αυτό θα μπορεί να ταξινομήσει με βάση την τιμή, τον χρόνο παράδοσης και θα έχει την δυνατότητα να εμφανίσει συγκεκριμένες χώρες. Τέλος στο κάτω μέρος θα μπορεί

να διαβάσει αρνητικά αλλά και θετικά σχόλια που έχουν εκπληρώσει άλλοι πελάτες που έχουν αγοράσει το συγκεκριμένο προϊόν.

Εφαρμογή εντοπισμού πτήσεων

Σήμερα οι υπάρχουσες εταιρείες έχουν εξελιχθεί και έχουν αναπτυχθεί σε περισσότερες από μια χώρες. Το γεγονός αυτό έχει κάνει τους εργαζόμενους να αναζητούν συνεχώς πτήσεις από διάφορες εταιρίες σε συγκεκριμένες μερες προκειμένου να επισκεφτούν μια άλλη χώρα για κάποιο προγραμματισμένο ραντεβού. Αρκετές φορές γνωρίζουν αρκετά νωρίτερα το ραντεβού προσπαθώντας να βρουν διαθέσιμη πτήση. Λόγω της πολύωρης δουλειάς δεν καταφέρνουν να αφιερώνουν χρόνο καθημερινά έχοντας ώς αποτέλεσμα να χάνονται κάποιες και στο τέλος κάποιες φορές να μην βρίσκεται.

Το πρόβλημα αυτό έρχεται να λύσει η εφαρμογή εντοπισμού πτήσεων. Η εφαρμογή θα έχει login για κάθε εταιρία και στην συνέχεια ένα ημερολόγιο που θα πηγαίνει κάθε εργαζόμενος και θα τοποθετεί το ραντεβού του. Οι πληροφορίες που θα πρέπει να σημειώνει ο εργαζόμενος είναι το εύρος τιμών, την προτίμησή του σε κάποιες αεροπορικές εταιρείες και φυσικά τα στοιχεία του (όνομα, επώνυμο, ταυτότητα, τόπος - χώρα και ότι άλλο χρειάζεται για να κλείσεις ένα εισιτήριο). Στην συνέχεια ένα υποσύστημα θα δεχεται τα δεδομένα και θα ξεκινά παροδικα ανά 2 λεπτά να ελέγχει όλες τις σελίδες αεροπορικών εταιρειών προκειμένου να εντοπίσει κάποια διαθέσιμη. Στην περίπτωση που βρεθεί η εφαρμογή κάνει κράτηση και ενημερώνει με email και sms τον χρήστη για την ολοκλήρωση. Ακόμα του δίνει την δυνατότητα να ενημερωθεί για την πτήση μέσα από το ημερολόγιο.

Εφαρμογή εντοπισμού χαρακτηριστικών Social Media

Στις μέρες μας οι χρήστες καταναλώνουν τον περισσότερο χρόνο τους στα προσωπικά τους προφίλ βλέποντας πως περνούν οι φίλοι τους από διάφορες φωτογραφίες ή δημοσιεύσεις που ανεβάζουν. Η πολύωρη χρήση έχει οδηγήσει πολλούς χρήστες το επάγγελμα τους να είναι τα δικά τους social media. Αυτό γίνεται είτε προσφέροντας διαφημιστικό περιεχόμενο, βγάζοντας ψυχαγωγικά βίντεο είτε από φωτογραφίσεις. Έτσι τα άτομα αυτά χρειάζεται να ξέρουν πληροφορίες και σχόλια ανταγωνιστών.

Η εφαρμογή εντοπισμού χαρακτηριστικών Social Media έχει ως σκοπο την συλλογή διαφόρων πληροφοριών σχετικά με το προφίλ ενός ανταγωνιστή. Σε αυτό το σύστημα χρειάζεται ο χρήστης να δηλώσει και τον ανταγωνιστή του και στην συνέχεια να επιλέξει από μια λίστα τα δεδομένα που θα ήθελε να βλέπει καθημερινά. Το σύστημα επίσης προσφέρει στατιστικά στοιχεία που δηλώνουν την δυναμικότητα του χρήστη σε σχέση με τους ανταγωνιστές του. Τέλος δίνεται επίσης η δυνατότητα αλλαγές στο προφίλ του σύμφωνα με κάποιες καλές τεχνικές που χρησιμοποιεί ο αντίπαλος. Σε αυτό το σημείο το σύστημα προτείνει ποιο θα ήταν το κατάλληλο format κειμένου που θα πρέπει να εχουν οι δημοσιευσεις του.

5.3 Αντι Scraping

Αρκετές ιστοσελίδες διαλέγουν να προστατευτούν από τα αυτοματοποιημένα προγράμματα συλλογής δεδομένων. Για να πετύχουν την κατάλληλη προστασία πρέπει να ακολουθήσουν συγκεκριμένες τεχνικές:

1. IP detection
2. IP rate limiting
3. Browser detection
4. Tracking user behavior

IP detection

Η ιστοσελίδα καθορίζει συγκεκριμένο εύρος κίνησης. Με αυτό τον τρόπο συγκεκριμένη πληροφορία την διαθέτη για συγκεκριμένες χώρες.

IP rate limiting

Η συγκεκριμένη τεχνική είναι από τις πιο διαδεδομένες. Αυτό γιατί μπορεί να καταλάβει αν μια ip κάνει πολλά request σε πολύ γρήγορο χρόνο κάτι που μπορεί να πετύχει μόνο ένας υπολογιστής.

Το μπλοκάρισμα της ip μπορεί να γίνει είτε από έναν άνθρωπο είτε από πρόγραμμα και στις δύο περιπτώσεις τίθεται ένα όριο και αν υπερβει τότε γίνεται η συγκεκριμένη ενέργεια.

Browser detection

Η τεχνική αυτή παρακολουθεί τον user Agent κάθε αιτήματος. Όπως και στην προηγούμενη τεχνική αν υπερβεί το όριο τότε ο αρμόδιος πράττει μια συγκεκριμένη ενέργεια.

Tracking user behavior

Η τελευταία και πιο δύσκολη καθώς συγκεκριμένο πρόγραμμα θα πρέπει να ελέγχει τις κινήσεις των χρηστών. Αναμφίβολα γίνεται σε βάθος χρόνου έτσι ώστε να γίνει σωστα η ανάλυση κινήσεων του bot αρχείου.

Εκτός από τις παραπάνω τεχνικές υπάρχουν και διάφορες μέθοδοι που μπορουν να μπερδέψουν τον scraper. Αυτά είναι:

- Τυχαίο id σε tags με σημαντική πληροφορία, φυσικά αυτό προϋποθέτει ότι τα id δεν χρησιμοποιούνται. Με αυτόν τον τρόπο γίνεται το έργο του scraper πιο δύσκολο καθώς για να τραβήξουν μια πληροφορία χρησιμοποιούν το id.
- Να μην χρησιμοποιούνται τα ιδια tags κοντά στην χρήσιμη πληροφορία. Με αυτό τον τρόπο δεν είναι εφικτό να χρησιμοποιήσει την εντολή ":nth-child(number)" και να πάρει κατευθείαν την πληροφορία που χρειάζεται. Στο παρακάτω παράδειγμα αν υπήρχαν 3 p tags τότε την χρήσιμη πληροφορία θα την λάμβανε με ":nth-child(3)" π.χ.

```
<div>
    <strong></strong>
    <p></p>
    <p>χρήσιμη πληροφορία</p>
</div>
```

Βιβλιογραφία

1) https://www.flood.io/blog/selenium-vs-puppeteer-for-test-automation-is-a-new-leader-emerging
2) https://pptr.dev/
3) https://www.innoview.gr/el/blog/general/258-gnoriste-to-node-js
4) https://el.wikipedia.org/wiki/
5) Apache_Kafka
6) https://medium.com/better-programming/rabbitmq-vs-kafka-1ef22a041793
7) https://www.upsolver.com/blog/kafka-versus-rabbitmq-architecture-performance-use-case
8) https://www.cloudamqp.com/blog/2015-05-18-part1-rabbitmq-for-beginners-what-is-rabbitmq.html
9) https://technology.amis.nl/2017/02/09/nodejs-publish-messages-to-apache-kafka-topic-with-random-delays-to-generate-sample-events-based-on-records-in-csv-file/
10) https://grafana.com/oss/prometheus/
11) https://grafana.com/oss/grafana/
12) http://ir.kdu.ac.lk/bitstream/handle/345/1051/com-059.pdf?sequence=1&isAllowed=y
13) https://protonvpn.com/
14) https://en.wikipedia.org/wiki/ProtonVPN
15) http://datajournalism.okfn.gr/getting_data_3.html
16) https://el.wikipedia.org/wiki/PostgreSQL