ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Πρόγραμμα Μεταπτυχιακών Σπουδών
Επιστήμη και Τεχνολογία της Πληροφορικής και των Υπολογιστών

Ειδίκευση Λογισμικού και Πληροφοριακών Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάκτηση και παρουσίαση πληροφορίας από αδόμητο κείμενο στο διαδίκτυο

Extracting and presenting data from unstructured web sources

Αθηνά Τσώνη
Α.Μ. 19043

Εισηγητής: Νικόλαος Ζάχαρης, Καθηγητής

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάκτηση και παρουσίαση πληροφορίας από αδόμητο κείμενο στο διαδίκτυο.

Αθηνά Τσώνη
Α.Μ. 19043

Εισηγητής:

Νικόλαος Ζάχαρης, Καθηγητής

Digitally signed by Nikolaos Zacharis
Date: 2023.07.18 11:51:07 +03'00'

Εξεταστική Επιτροπή:

Αντώνιος Μπόγρης, Καθηγητής

Antonios Bogris
Digitally signed by Antonios Bogris
Date: 2023.07.31 09:08:59 +03'00'

Γεώργιος Πρεζεράκος, Καθηγητής

Georgios Prezerakos
Digitally signed by Georgios Prezerakos
Date: 2023.07.18 12:36:39 +03'00'

Ημερομηνία εξέτασης 18/7/2023

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Η κάτωθι υπογεγραμμένη Αθηνά Τσώνη του Παναγιώτη, με αριθμό μητρώου MCSE19043 φοιτήτρια του Προγράμματος Μεταπτυχιακών Σπουδών Επιστήμη και Τεχνολογία της Πληροφορικής και των Υπολογιστών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:

«Είμαι συγγραφέας αυτής της μεταπτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Η Δηλούσα

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία ασχολείται με την εξαγωγή δεδομένων από το διαδίκτυο ή αλλιώς web scraping. Το διαδίκτυο αποτελεί τη μεγαλύτερη πηγή πληροφοριών στη σημερινή εποχή και περιλαμβάνει ως επί το πλείστον αδόμητα δεδομένα που είναι δύσκολο να συλλεχθούν. Η εξαγωγή των δεδομένων μπορεί να γίνει με αντιγραφή - επικόλληση, αλλά ο τρόπος αυτός είναι εξαιρετικά χρονοβόρος και μη αποδοτικός σε περιπτώσεις μεγάλου όγκου δεδομένων. Για να επιταχύνουμε και να αυτοματοποιήσουμε τη συλλογή πληροφοριών από τον παγκόσμιο ιστό χρησιμοποιούμε τη μέθοδο του web scraping.

Η εργασία αναφέρεται αρχικά στις διάφορες τεχνικές και εφαρμογές του web scraping. Στη συνέχεια παρουσιάζονται τα σημαντικότερα εργαλεία για τη συλλογή δεδομένων μέσω προγραμματισμού ή μέσω έτοιμων εφαρμογών, όπως SaaS εργαλεία, desktop εφαρμογές και add-ons για τα προγράμματα περιήγησης. Η εργασία επικεντρώνεται στη συλλογή δεδομένων με Python και παρουσιάζονται οι κυριότερες βιβλιοθήκες που διαθέτει η γλώσσα για web scraping. Τέλος, περιγράφονται οι καλές πρακτικές που πρέπει να ακολουθούνται κατά την άντληση δεδομένων από το διαδίκτυο, όπως η προσαρμογή των κεφαλίδων HTTP, η διαχείριση των cookies κλπ.

Στο πλαίσιο της εργασίας υλοποιήθηκε μια διαδικτυακή εφαρμογή για τη συλλογή αξιολογήσεων από έναν δημοφιλή ιστότοπο σύγκρισης τιμών. Η εφαρμογή διαθέτει ένα απλό UI με φόρμα αναζήτησης, στην οποία ο χρήστης εισάγει το προϊόν που τον ενδιαφέρει. Οι αξιολογήσεις του προϊόντος εμφανίζονται σε μορφή πίνακα και ο χρήστης μπορεί να κατεβάσει τα αποτελέσματα σε αρχείο CSV. Η υλοποίηση της εφαρμογής έγινε σε γλώσσα Python και χρησιμοποιήθηκαν τα frameworks Scrapy και Flask.

ABSTRACT

The present thesis concerns web scraping: the process of collecting web data in an automated manner. The internet is the largest source of information nowadays and contains mostly unstructured data that is difficult to collect. While manual extraction through copy-pasting is feasible, it is a time-consuming and inefficient process, particularly when dealing with large amounts of data. To expedite and automate the information gathering process from the World Wide Web, web scraping is employed.

This thesis reviews the different web scraping approaches, techniques and areas of application. It presents various tools for extracting data programmatically, and software tools, such as SaaS scrapers, desktop scrapers and browser extensions. The study primarily focuses on data collection using the Python programming language and highlights the main Python libraries and tools used for web scraping. Additionally, it describes best practices for extracting web data, such as adjusting HTTP headers, handling cookies, and more.

In the second part of the thesis, a web application was implemented to collect reviews from a popular price comparison website. The application has a simple UI with a search form where users can type in the product of interest. Product reviews are scraped and displayed in tabular format, and users have the option to download the results in a CSV file. The application was developed in Python using the Scrapy and Flask frameworks.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: εξαγωγή δεδομένων ιστού, συλλογή δεδομένων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: web scraping, web harvesting, web data extraction, data collection, python, scrapy, flask, web application, διαδικτυακή εφαρμογή

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΑΠΟΣΠΑΣΜΑΤΩΝ ΚΩΔΙΚΑ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

CSV	Comma-Separated Values
JSON	JavaScript Object Notation
HTTP	Hyper Text Transfer Protocol
HTML	HyperText Markup Language
XQuery	XML Query Language
HTQL	Hyper-Text Query Language
XML	Extensible Markup Language
DOM	Document Object Model
CSS	Cascading Style Sheets
XPath	XML Path Language
URL	Uniform Resource Locator

API	Application Programming Interface
TCP	Transmission Control Protocol
SaaS	Software as a Service
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
AJAX	Asynchronous JavaScript And XML
DHTML	Dynamic HTML
WSGI	Web Server Gateway Interface
CDN	Content Delivery Network
XHR	XMLHttpRequest

1. ΕΙΣΑΓΩΓΗ

Εξαγωγή δεδομένων ιστού είναι η διαδικασία εξαγωγής αδόμητων δεδομένων από ιστοσελίδες, η μετατροπή τους σε δομημένα δεδομένα και τέλος η αποθήκευση τους για μετέπειτα επεξεργασία και ανάλυση. Ο όρος συναντάται, επίσης, ως web data extraction, web harvesting, screen scraping ή information scraping. Στην ελληνική γλώσσα ονομάζεται εξαγωγή δεδομένων ιστού ή ιστοσυγκομιδή. Τα δεδομένα που εξάγονται μπορούν να αποθηκευτούν σε βάση δεδομένων ή σε αρχείο τύπου excel, csv ή json.

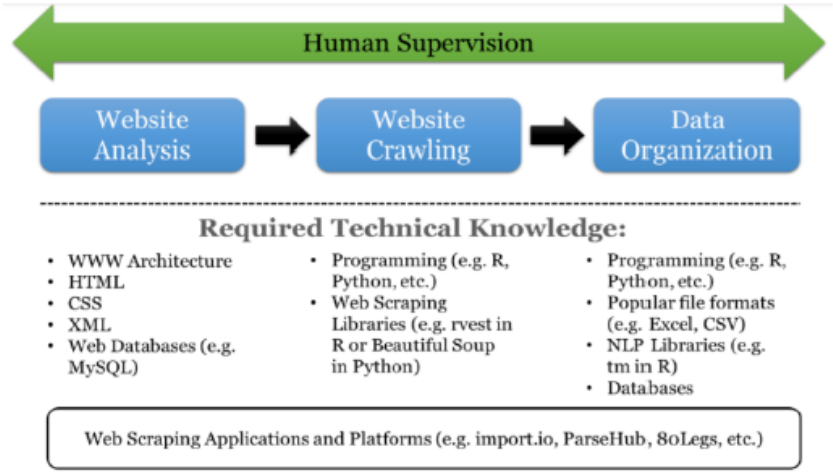
1.1 Εξαγωγή δεδομένων ιστού (web scraping)

Η πλειοψηφία των χρηστών αλληλεπιδρά με το διαδίκτυο μέσω της πρόσβασης σε ιστότοπους μέσα από προγράμματα περιήγησης, όπου οι πληροφορίες και τα πολυμέσα παρουσιάζονται με τρόπο φιλικό προς τον χρήστη. Ωστόσο, αυτή είναι μόνο μια από τη δυνατότητες του διαδικτύου καθώς υπάρχει πληθώρα πληροφοριών που δεν είναι ορατές. Οι διεπαφές προγραμματισμού εφαρμογών (API) μπορούν να χρησιμοποιηθούν για εύκολη πρόσβαση σε αυτές τις πληροφορίες, αλλά συνήθως οι κάτοχοι των ιστότοπων επιλέγουν να μη δώσουν πρόσβαση μέσω API. Από την άλλη πλευρά, η εξαγωγή δεδομένων ιστού είναι μια γρήγορη και αποτελεσματική επιλογή για τη συλλογή και επεξεργασία δεδομένων από χιλιάδες ή εκατομμύρια σελίδες. Η τεχνική αυτή έχει εφαρμογή σε πολλούς τομείς και κυρίως στον τομέα της επιχειρηματικής ευφυΐας (business intelligence). Οι σύγχρονες ανάγκες επιβάλλουν την ύπαρξη διαδικτυακής παρουσίας για τις επιχειρήσεις και την παρακολούθηση του ανταγωνισμού, και η εξαγωγή δεδομένων από το διαδίκτυο είναι ένα σημαντικό εργαλείο προς αυτή την κατεύθυνση [1].

Τα δεδομένα είναι πολύ σημαντικά για τις επιχειρήσεις και τους οργανισμούς, καθώς βοηθούν στη λήψη αποφάσεων, και στη σημερινή εποχή τα περισσότερα δεδομένα βρίσκονται διαθέσιμα στο διαδίκτυο. Η απόκτηση των δεδομένων είναι το πρώτο βήμα σε οποιαδήποτε μελέτη και εφαρμογή στον τομέα της επιστήμης δεδομένων (data science). Στο στάδιο αυτό τα δεδομένα λαμβάνονται είτε από ιδιωτικές πηγές, όπως αρχεία πωλήσεων και οικονομικές εκθέσεις εταιρειών, είτε από δημόσιες πηγές, όπως περιοδικά, ιστότοπους και ανοιχτά δεδομένα είτε με αγορά των δεδομένων. Η ανάλυση ιστοτόπου (website analysis), η ανίχνευση ιστοτόπου (website crawling) και η οργάνωση των δεδομένων είναι οι τρεις κύριες διαδικασίες που είναι συνυφασμένες με την εξαγωγή δεδομένων ιστού (βλ. Εικόνα 1). Η εξαγωγή δεδομένων (web scraping) διαφέρει από την εξόρυξη δεδομένων (data mining), γιατί η εξόρυξη δεδομένων εμπεριέχει και την ανάλυση των δεδομένων που συλλέγονται και απαιτεί τη χρήση εξελιγμένων στατιστικών μεθόδων. Η εξαγωγή δεδομένων ιστού είναι μια σχετικά απλή διαδικασία, καθώς υπάρχει διαθέσιμος ένας μεγάλος αριθμός εργαλείων και βιβλιοθηκών για την υλοποίηση των περισσότερων λειτουργιών που

απαιτούνται. Η δυνατότητα αποστολής αιτημάτων HTTP με προσαρμοσμένες κεφαλίδες (headers) και payload είναι τυπικό χαρακτηριστικό των περισσότερων εργαλείων εξαγωγής δεδομένων ιστού [1].

Εξαγωγή δεδομένων ιστού είναι η συλλογή δεδομένων από έναν ιστότοπο με τη χρήση λογισμικού, το οποίο αντιγράφει τον τρόπο με τον οποίο ο χρήστης περιηγείται στον παγκόσμιο ιστό. Αυτό μπορεί να γίνει με την εκτέλεση ενός προγράμματος περιήγησης ή με την αποστολή αιτημάτων HTTP. Επομένως, αντί για αντιγραφή και επικόλληση των δεδομένων από έναν ιστότοπο και αποθήκευση σε αρχείο τύπου excel, η τεχνική του web scraping προσφέρει την ίδια λειτουργικότητα μέσω εφαρμογών λογισμικού που πραγματοποιούν τις ίδιες εργασίες με μεγαλύτερη ταχύτητα και ακρίβεια [1].



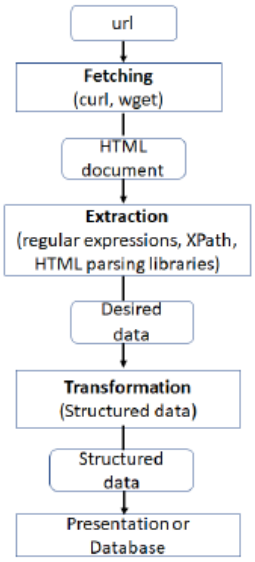
Εικόνα 1. Εξαγωγή δεδομένων ιστού [1]

Η διαδικασία της εξαγωγής δεδομένων ιστού χωρίζεται σε τρία στάδια, όπως φαίνεται στην Εικόνα 2, και είναι τα εξής [1]:

Στάδιο ανάκτησης: Αρχικά θα πρέπει να αποκτήσουμε πρόσβαση στον ιστότοπο που μας ενδιαφέρει. Αυτό επιτυγχάνεται μέσω του πρωτοκόλλου HTTP, το οποίο είναι ένα πρωτόκολλο διαδικτύου για αποστολή και λήψη αιτημάτων από διακομιστές (web servers). Τα προγράμματα περιήγησης ιστού χρησιμοποιούν παρόμοιες μεθόδους για τη λήψη του περιεχομένου των ιστοσελίδων. Σε αυτό τα στάδιο μπορούν να χρησιμοποιηθούν βιβλιοθήκες, όπως οι curl και wget, για την αποστολή GET αιτημάτων στη σελίδα-στόχο και για τη λήψη της σελίδας HTML.

Στάδιο εξαγωγής: Μετά την ανάκτηση της σελίδας HTML ακολουθεί η εξαγωγή των δεδομένων. Στο στάδιο αυτό μπορούν να χρησιμοποιηθούν κανονικές εκφράσεις (regular expressions), βιβλιοθήκες για HTML parsing ή ερωτήματα XPath. Η γλώσσα XPath (XML Path Language) είναι ένα χρήσιμο εργαλείο για την εύρεση πληροφοριών σε έγγραφα.

Στάδιο μετασχηματισμού: Αφού έχουν συγκεντρωθεί τα απαραίτητα δεδομένα, ακολουθεί η μετατροπή τους σε δομημένη μορφή για την παρουσίαση ή την αποθήκευση τους.



Εικόνα 2. Διαδικασία εξαγωγής δεδομένων ιστού [1]

1.2 Τεχνικές web scraping

Αντιγραφή και επικόλληση: Ο απλούστερος τρόπος συλλογής δεδομένων ιστού είναι η αντιγραφή των δεδομένων από μια ιστοσελίδα και η επικόλληση σε ένα αρχείο κειμένου ή σε ένα υπολογιστικό φύλλο. Μερικές φορές ακόμα και η καλύτερη τεχνολογία δεν μπορεί να αντικαταστήσει την εξέταση των πληροφοριών και τη χειροκίνητη αντιγραφή-επικόλληση από τον άνθρωπο, και σε ορισμένες περιπτώσεις είναι η μόνη εφαρμόσιμη λύση όταν οι ιστότοποι θέτουν εμπόδια για να αποτρέψουν την αυτόματη συλλογή των δεδομένων τους.

Χρήση κανονικών εκφράσεων: Μια άλλη μέθοδος για εξαγωγή πληροφοριών από ιστοσελίδες βασίζεται στην εντολή grep του UNIX ή στη χρήση κανονικών εκφράσεων (regular expressions).

HTTP προγραμματισμός: Με τη χρήση socket προγραμματισμού (socket programming) και με αποστολή αιτημάτων HTTP μπορεί να γίνει ανάκτηση πληροφοριών από στατικές και δυναμικές ιστοσελίδες.

Ανάλυση HTML (HTML parsing): Ορισμένες ημι-δομημένες γλώσσες ερωτημάτων, όπως η XQuery και η HTQL, μπορούν να χρησιμοποιηθούν για την ανάλυση μιας σελίδας HTML και τη μετέπειτα ανάκτηση και μετατροπή του περιεχομένου της σελίδας.

Ανάλυση DOM (DOM parsing): Τα προγράμματα εξαγωγής δεδομένων ιστού μπορούν να ανακτούν το δυναμικό περιεχόμενο που δημιουργείται με script στην πλευρά του πελάτη (client),

χρησιμοποιώντας ένα πρόγραμμα περιήγησης. Το πρόγραμμα περιήγησης αναλύει την ιστοσελίδα σε ένα δέντρο DOM, με βάση το οποίο ανακτώνται συγκεκριμένα τμήματα της σελίδας από τα προγράμματα web scraping. Για την ανάλυση του δέντρου DOM χρησιμοποιούνται γλώσσες όπως η XPath.

Κάθετη συγκέντρωση δεδομένων (vertical aggregation): Αρκετές εταιρείες έχουν αναπτύξει ειδικές πλατφόρμες κάθετης συλλογής δεδομένων. Οι πλατφόρμες αυτές δημιουργούν και παρακολουθούν ένα πλήθος bot εξειδικευμένων στη συλλογή δεδομένων συγκεκριμένου τύπου. Στην περίπτωση αυτή δεν απαιτείται άμεση ανθρώπινη παρέμβαση για τον κάθε ιστότοπο-στόχο. Αρχικά δημιουργείται μια γνωσιακή βάση για τη συγκεκριμένη κατηγορία δεδομένων και στη συνέχεια η πλατφόρμα δημιουργεί αυτόματα τα bot. Η επιτυχία της πλατφόρμας αξιολογείται με βάση την ποιότητα των πληροφοριών που ανακτά και την επεκτασιμότητά της, δηλαδή πόσο γρήγορα μπορεί να κλιμακωθεί σε εκατοντάδες ή χιλιάδες ιστότοπους.

Αναγνώριση σημασιολογικού περιεχομένου (semantic annotation recognizing): Οι σελίδες από τις οποίες επιχειρείται η εξαγωγή των δεδομένων ενδέχεται να περιέχουν μεταδεδομένα ή σημασιολογικές σημάνσεις που μπορούν να χρησιμοποιηθούν για τον εντοπισμό δεδομένων. Εάν οι σημάνσεις είναι ενσωματωμένες στις σελίδες, όπως είναι τα Microformats, η συγκεκριμένη τεχνική web scraping μπορεί να θεωρηθεί ως ειδική περίπτωση ανάλυσης του DOM. Εάν οι σημάνσεις είναι οργανωμένες σε ένα σημασιολογικό επίπεδο, η αποθήκευση και η διαχείρισή τους γίνονται ξεχωριστά από τις ιστοσελίδες. Στην περίπτωση αυτή, τα εργαλεία εξαγωγής δεδομένων (scrapers) μπορούν να ανακτούν από αυτό το επίπεδο το σχήμα των δεδομένων και οδηγίες πριν τη συλλογή των δεδομένων [2].

Ανάλυση ιστοσελίδων με μηχανική όραση: Πρόκειται για μια σύγχρονη προσέγγιση του web scaping η οποία λαμβάνει υπόψη την οπτική αναπαράσταση μιας ιστοσελίδας και βασίζεται στη μηχανική όραση (computer vision) και στη μηχανική μάθηση (machine learning). Ορισμένες εφαρμογές τέτοιου είδους είναι γνωστές με τα ακρωνύμια VIPS, ViPER, ViNT, WeRE, VFX. Οι τεχνικές αυτές δεν «κοιτάζουν» την ιστοσελίδα απλά ως εικόνα, αλλά υπολογίζουν οπτικά χαρακτηριστικά, όπως η θέση και οι διαστάσεις για τα διάφορα μπλοκ περιεχομένου, λαμβάνοντας πληροφορίες από το πρόγραμμα περιήγησης. Επομένως, η φόρτωση της ιστοσελίδας στο πρόγραμμα περιήγησης είναι απαραίτητη για την εξαγωγή χαρακτηριστικών, τα οποία αξιοποιούνται στη συνέχεια για τον εντοπισμό ή την ταξινόμηση σχετικών τμημάτων περιεχομένου.

Σύμφωνα με προηγούμενες δημοσιεύσεις για τη χρήση μηχανικής όρασης στην εξαγωγή δεδομένων ιστού, οι τεχνολογίες που εφαρμόζονται είναι η ανάλυση διάταξης εγγράφων (document layout analysis) και η ανίχνευση αντικειμένων (object detection).

Η διαδικασία που ακολουθείται κατά την ανάλυση διάταξης εγγράφων περιλαμβάνει αρχικά την αναγνώριση τμημάτων περιεχόμενου στην ιστοσελίδα με κλασικές μεθόδους μηχανικής όρασης,

όπως η ανίχνευση περιγραμμάτων, και στη συνέχεια ακολουθεί η ταξινόμηση αυτών των τμημάτων.

Μια εναλλακτική και πιο σύγχρονη μέθοδος είναι η ανίχνευση αντικειμένων (object detection), η οποία δεν βασίζεται στην ανίχνευση χαρακτηριστικών ή περιγραμμάτων όπως η προηγούμενη προσέγγιση. Σε αυτή την περίπτωση χρησιμοποιείται ένα συνελικτικό νευρωνικό δίκτυο (convolutional neural network ή CNN), και συγκεκριμένα ένα χωρικό δίκτυο CNN (Regional-CNN ή R-CNN), το οποίο εκπαιδεύεται κατάλληλα ώστε όχι μόνο να ταξινομεί το περιεχόμενο των διαφόρων τμημάτων περιεχομένου που του δίνονται αλλά και να μπορεί να τα εντοπίζει.

Στις μεθόδους που προαναφέρθηκαν εφαρμόζεται μηχανική μάθηση για την ταξινόμηση των μπλοκ περιεχομένου μιας ιστοσελίδας. Αφού εντοπιστούν τα διάφορα τμήματα περιεχομένου, εκπαιδεύεται ένας ταξινομητής για την κατηγοριοποίηση αυτών των τμημάτων σε «περιεχόμενο», «τίτλο», «συντάκτη» κλπ. Στο πεδίο του web scraping οι αλγόριθμοι ταξινόμησης που χρησιμοποιούνται συνήθως είναι οι μηχανές διανυσματικής υποστήριξης (Support Vector Machines ή SVM) [3].

1.3 Εφαρμογές

Η εξαγωγή δεδομένων ιστού χρησιμοποιείται ευρέως για διάφορους σκοπούς, όπως σύγκριση τιμών στο διαδίκτυο, παρατήρηση αλλαγών στα δεδομένα καιρού, ανίχνευση αλλαγών σε ιστότοπους, έρευνα, ενσωμάτωση δεδομένων από πολλαπλές πηγές, ενημέρωση για προσφορές και εκπτώσεις, συλλογή πληροφοριών για θέσεις εργασίας, παρακολούθηση ενός brand και ανάλυση αγοράς.

Χρησιμοποιείται, επίσης, ως ένα γρήγορο και αποτελεσματικό μέσο συλλογής δεδομένων. Η εξαγωγή δεδομένων από το διαδίκτυο έχει πολυάριθμες εφαρμογές σε διάφορους τομείς και αποτελεί απαραίτητη προϋπόθεση για την ανάλυση μεγάλων δεδομένων. Παρακάτω αναφέρονται μερικοί από τους τομείς στους οποίους βρίσκει εφαρμογή [44].

Υγεία

Η υγειονομική περίθαλψη δεν είναι πλέον ένας τομέας που βασίζεται εξ ολοκλήρου στη φυσική επαφή με τον ασθενή, αλλά έχει γίνει με έναν τρόπο ψηφιακός. Η εφαρμογή του web scraping στον τομέα της υγείας μπορεί να σώσει ζωές, καθώς συμβάλλει στη λήψη σημαντικών αποφάσεων.

Οι εργαζόμενοι στον τομέα της υγείας αντιμετωπίζουν τη συλλογή δεδομένων ασθενών ως μια κουραστική και επίπονη διαδικασία. Παρόλο που η ανάγκη για κλινικά δεδομένα είναι τεράστια, ο μεγάλος αριθμός των ασθενών καθιστά τη συλλογή των δεδομένων σχεδόν αδύνατη. Για το σκοπό αυτό οι ερευνητές Melchor et al. (2020) προτείνουν ένα σύστημα που συλλέγει αυτόματα κλινικά

δεδομένα από ασθενείς με SARS-CoV2 που επισκέπτονται το νοσοκομείο, προκειμένου να χρησιμοποιηθούν σε μελλοντική έρευνα [45].

Μια άλλη εφαρμογή του web scraping στον τομέα της υγειονομικής περίθαλψης είναι η έρευνα που διεξήχθη από τους Dascalu et al. (2019) στην οποία συλλέγονται πληροφορίες από φυλλάδια φαρμάκων [46].

Μέσα κοινωνικής δικτύωσης

Η εξαγωγή δεδομένων από τα μέσα κοινωνικής δικτύωσης μπορεί να βοηθήσει τις εταιρείες να βελτιώσουν τις διαφημιστικές τους καμπάνιες, να αναλύσουν τη γνώμη των πελατών για τα προϊόντα τους, να βελτιώσουν τις δημόσιες σχέσεις και την αφοσίωση του κοινού.

Ένα τέτοιο παράδειγμα είναι η εφαρμογή που αναπτύχθηκε από τους Himawan, Priadana και Murdiyanto (2020), η οποία συγκεντρώνει δεδομένα από λογαριασμούς στο Instagram χρησιμοποιώντας τεχνολογία web scraping. Οι ερευνητές επέλεξαν τη μέθοδο της εξαγωγής δεδομένων ιστού αντί της χρήσης του API που διαθέτει το Instagram, γιατί το API έχει αρκετούς περιορισμούς όσον αφορά την πρόσβαση και την ανάκτηση δεδομένων από την πλατφόρμα. Η εφαρμογή δοκιμάστηκε σε 15 λογαριασμούς με συνολικό αριθμό δημοσιεύσεων που κυμαίνονταν από 100 έως 11,000. Σύμφωνα με τα αποτελέσματα της έρευνας, καταγράφηκαν με επιτυχία δεδομένα από 2,412 λογαριασμούς. Η συγκεκριμένη εφαρμογή μπορεί να χρησιμοποιηθεί για την αποθήκευση δεδομένων από το Instagram σε ένα σύστημα διαχείρισης βάσεων δεδομένων ή την εξαγωγή τους σε αρχείο excel, json ή csv [47].

Χρηματοοικονομικά

Σε μια άλλη έρευνα οι Axenbeck και Breithaupt (2021) χρησιμοποίησαν δεδομένα 4,487 επιχειρήσεων από μια γερμανική έρευνα καινοτομίας, για να εξετάσουν ποια είναι τα χαρακτηριστικά των ιστοτόπων που λειτουργούν ως προγνωστικά στοιχεία για την καινοτόμα δραστηριότητα μιας εταιρείας. Η συλλογή των δεδομένων από τους ιστότοπους των εταιρειών έγινε με τον scraper ARGUS. Τα χαρακτηριστικά μελετήθηκαν με διάφορες μεθόδους εξόρυξης δεδομένων και με διάφορα μοντέλα ταξινόμησης Random Forest. Τα αποτελέσματα έδειξαν ότι τα σημαντικότερα χαρακτηριστικά των ιστότοπων που μπορούν να αποτελέσουν δείκτες καινοτομίας είναι το περιεχόμενο κειμένου, η χρήση της αγγλικής γλώσσας, ο αριθμός των σελίδων και το πλήθος των χαρακτήρων [48].

Στην εργασία των Balendran et al. (2018) γίνεται τεχνική ανάλυση ειδησεογραφικών άρθρων που έχουν συλλεχθεί από το διαδίκτυο και τα οποία μπορεί να επηρεάσουν την κίνηση των μετοχών στο χρηματιστήριο. Τα άρθρα εξάγονται από αξιόπιστο ιστότοπο και τα περιεχόμενα του ιστότοπου συνοψίζονται προκειμένου να γίνει ανάλυση και μοντελοποίηση συμβάντων (event modeling) για να προβλεφθούν αλλαγές στην χρηματιστηριακή αγορά [49].

Μάρκετινγκ

Η εργασία των Boegershausen et al. (2022) αναφέρεται στον τεράστιο όγκο δεδομένων πελατών που είναι διαθέσιμος στο διαδίκτυο και ο οποίος μπορεί να αξιοποιηθεί για την ανάλυση της συμπεριφοράς των πελατών και τη διεξαγωγή έρευνας [50]. Οι Saranya et al. (2020) χρησιμοποιούν μοντέλα μηχανικής μάθησης για να προβλέψουν την πρόθεση αγοράς με βάση τη συμπεριφορά του πελάτη κατά τη διάρκεια διαδικτυακών αγορών. Τα δεδομένα συλλέγονται με τη μέθοδο του web scraping και στη συνέχεια αναλύονται για να προβλεφθεί η πρόθεση αγοράς [51].

Οι Nguyen, Sukunesan και Huynh (2021) χρησιμοποιούν web scraping για να αναλύσουν την αλληλεπίδραση του κοινού στα μέσα κοινωνικής δικτύωσης αυστραλιανών μικρομεσαίων επιχειρήσεων. Συλλέγουν δεδομένα από το Instagram μέσω του API της πλατφόρμας, εξετάζουν τα δεδομένα και συμπεραίνουν ότι η προσθήκη ετικετών (tags) αντί για hashtags οδηγεί σε μεγαλύτερη αλληλεπίδραση του κοινού [52].

Άλλοι τομείς

Ο Deng (2020) χρησιμοποίησε στην εργασία του τεχνικές web scraping για να συλλέξει πληροφορίες σχετικά με τη βιομηχανία εξόρυξης στην Κίνα [53]. Οι Kotouza et al. (2020) εκμεταλλεύτηκαν τις τεχνικές εξαγωγής δεδομένων ιστού για να σχεδιάσουν ένα σύστημα που θα παρέχει πληροφορίες σχετικά τις νεότερες τάσεις της μόδας σε εργαζόμενους του συγκεκριμένου κλάδου [54]. Σε άλλη δημοσίευση, οι συγγραφείς Wang και Song (2019) συγκέντρωσαν δεδομένα από το διαδίκτυο για να εξάγουν χρήσιμες πληροφορίες για τον τομέα της δασοκομίας στην Κίνα [55]. Οι Seliverstov et al. (2020) μελέτησαν την ασφάλεια του οδικού δικτύου στο Βορειοδυτικό Ομοσπονδιακό Διαμέρισμα της Ρωσίας, συλλέγοντας κριτικές από το διαδίκτυο με τη βιβλιοθήκη Scrapy της Python [56].

Έρευνα

Οι Suganya και Vijayarani (2021) χρησιμοποίησαν τη μέθοδο του web scraping για ανάλυση παραπομπών (citations) από το διαδίκτυο, προκειμένου να δημιουργήσουν ένα εργαλείο που θα βοηθά τους ερευνητές να βρίσκουν παρόμοιες δημοσιεύσεις. Μελέτησαν και σύγκριναν τρεις μεθόδους: Particle Swarm Optimization, Hidden Markov Model και Firefly σε συνδυασμό με web scraping για να συλλέξουν πληροφορίες για παραπομπές στο διαδίκτυο με βάση ένα συγκεκριμένο ερώτημα (query). Από την έρευνα τους προέκυψε ότι η εξαγωγή δεδομένων ιστού με βάση τον αλγόριθμο Firefly είχε τα καλύτερα αποτελέσματα [57].

Ομοίως, οι συγγραφείς Rahmatulloh και Gunawan (2020) χρησιμοποίησαν εξαγωγή δεδομένων ιστού με βάση το HTML DOM για να παραγάγουν συγκεντρωτικά δεδομένα για επιστημονικά άρθρα από το Google Scholar. Δημιούργησαν μια PHP εφαρμογή με MySQL server για την παρουσίαση των αποτελεσμάτων, η οποία δίνει τη δυνατότητα εξαγωγής των δεδομένων σε αρχείο pdf ή excel [58].

Ο Li (2020) προτείνει τη χρήση εξαγωγής δεδομένων ιστού και επεξεργασίας φυσικής γλώσσας για να μειωθεί ο χρόνος που απαιτείται για τον εντοπισμό θεματικών περιοχών με ερευνητικό κενό. Η προσέγγιση αυτή δοκιμάστηκε σε τρεις τομείς με δημοσιεύσεις μεταξύ 1988 και 2019. Αρχικά συλλέχθηκαν και αναλύθηκαν οι τίτλοι των δημοσιεύσεων από το Google Scholar και στη συνέχεια έγινε ταξινόμηση των συμφράσεων με φθίνουσα συχνότητα. Με τον τρόπο αυτό εντοπίστηκε το σύνολο των λέξεων-κλειδιών που δεν είχαν χρησιμοποιηθεί στους τίτλους των άρθρων και προσδιορίστηκε το ερευνητικό κενό [59].

Η δημοσίευση των Santos et al. (2020) περιγράφει την εξαγωγή ενός συνόλου δεδομένων σχετικά με την επιστημονική έρευνα για τον COVID-19, προκειμένου να εντοπιστούν οι χώρες, οι επιστήμονες και οι ερευνητικές ομάδες που ήταν περισσότερο ενεργές στη μελέτη του κορονοϊού. Εξήχθη ένα σύνολο δεδομένων από διάφορες πηγές (Scopus, PubMed, arXiv και bioRxiv), το οποίο περιείχε 40,212 εγγραφές με μεταδεδομένα άρθρων από τον Ιανουάριο του 2019 μέχρι τον Ιούλιο του 2020. Τα δεδομένα συλλέχθηκαν με τεχνικές web scraping στη γλώσσα Python και υπέστησαν προ-επεξεργασία (data wrangling) με τη βιβλιοθήκη Pandas. Η εξαγωγή των δεδομένων από τις βάσεις PubMed και Scopus έγινε μέσω API, ενώ για τις βάσεις arXiv και bioRxiv χρησιμοποιήθηκε η βιβλιοθήκη Scrapy [60].

1.4 Νομιμότητα και ηθική του web scraping

Νομιμότητα του web scraping

Ενώ υπάρχουν διαθέσιμα πολυάριθμα εργαλεία και τεχνολογίες για την εξαγωγή δεδομένων από το διαδίκτυο, η νομιμότητα του web scraping αποτελεί ακόμα “γκρίζα ζώνη” στο νομικό πεδίο (Snell & Menaldo, 2016). Νομιμότητα εννοούμε στην περίπτωση αυτή τη συμμόρφωση με τους ισχύοντες νόμους και τις νομικές θεωρίες. Μέχρι τώρα δεν υπάρχει συγκεκριμένη νομοθεσία αναφορικά με το web scraping αλλά υπόκειται σε ένα σύνολο νομικών θεωριών και νόμων, όπως η “παραβίαση πνευματικών δικαιωμάτων”, η “παραβίαση σύμβασης”, ο νόμος περί απάτης και κατάχρησης υπολογιστών (Computer Fraud and Abuse Act-CFAA) που ισχύει στις ΗΠΑ και η “παραβίαση περιουσίας” (“trespass to chattels”) (Dreyer & Stockton 2013, Snell & Menaldo 2016). Παρακάτω θα αναφέρουμε πως εφαρμόζονται αυτές οι θεμελιώδεις νομικές θεωρίες στην εξαγωγή δεδομένων ιστού.

Όροι χρήσης

Έχει υποστηριχθεί συχνά στο νομικό πεδίο ότι ο κάτοχος ενός ιστότοπου μπορεί να απαγορεύσει την αυτόματη πρόσβαση στον ιστότοπο δηλώνοντάς το ρητά στην πολιτική “όρων χρήσης”, η οποία δημοσιεύεται στον ιστότοπο. Η μη συμμόρφωση με τους όρους μπορεί να οδηγήσει σε “παραβίαση της σύμβασης” από την πλευρά του χρήστη του ιστότοπου (Dryer & Stockton, 2013).

Προκειμένου να διωχθεί κάποιος χρήστης για παραβίαση των όρων, θα πρέπει να έχει συνάψει μια ρητή συμφωνία με τον κάτοχο του ιστότοπου για συμμόρφωση με τους όρους χρήσης (με την επιλογή ενός checkbox, για παράδειγμα). Επομένως η απλή απαγόρευση της ανίχνευσης (crawling) και της εξόρυξης (scraping) στον ιστότοπο δεν μπορεί να αποκλείσει κάποιον από νομική άποψη.

Υλικό που προστατεύεται από πνευματικά δικαιώματα

Η εξαγωγή και αναδημοσίευση δεδομένων ή πληροφοριών που ανήκουν στον κάτοχο του ιστότοπου και προστατεύονται ρητά από πνευματικά δικαιώματα μπορεί να οδηγήσει σε υπόθεση "παραβίασης πνευματικών δικαιωμάτων" (Dryer & Stockton, 2013). Ωστόσο, ένας ιστότοπος δεν κατέχει απαραίτητα τα δεδομένα που δημιουργούνται από τους χρήστες του. Για παράδειγμα, ένας ιστότοπος που δημοσιεύει κριτικές προϊόντων δεν έχει στην ιδιοκτησία του τις κριτικές που γράφουν οι χρήστες. Επιπλέον, οι ιδέες δεν μπορούν να προστατεύονται από πνευματικά δικαιώματα – μόνο συγκεκριμένη μορφή ή αναπαράσταση αυτών των ιδεών. Επομένως, μπορεί κανείς να χρησιμοποιήσει δεδομένα που προστατεύονται από πνευματικά δικαιώματα για να δημιουργήσει περιλήψεις από αυτά τα δεδομένα. Επίσης, μπορεί να χρησιμοποιηθεί υλικό που προστατεύεται από πνευματικά δικαιώματα, αλλά σε περιορισμένη κλίμακα και σύμφωνα με την αρχή της "δίκαιης χρήσης".

Σκοπός της εξαγωγής των δεδομένων

Οποιαδήποτε παράνομη ή δόλια χρήση δεδομένων που λαμβάνονται μέσω web scraping απαγορεύεται από τη νομοθεσία. Για παράδειγμα, ένα άτομο που έχει πρόσβαση σε δεδομένα ιστού, τα οποία είναι γνωστό ότι είναι εμπιστευτικά και προστατεύονται, μπορεί να διωχθεί σύμφωνα με τον νόμο περί απάτης και κατάχρησης υπολογιστών (Computer Fraud and Abuse Act - CFAA) εάν προκύψει ζημιά μεγαλύτερη από 5,000$ (Dryer & Stockton, 2013). Στο διαδίκτυο αυτό συμβαίνει συχνά όταν κάποιος αποκτά εν γνώσει του πρόσβαση σε "premium περιεχόμενο" και στη συνέχεια το μεταπωλεί ή συνεχίζει να έχει πρόσβαση στο περιεχόμενο μέσω μη εξουσιοδοτημένου καναλιού και αφού έχει λάβει ειδοποίηση "παύσης και παραίτησης" ("cease and desist") από τον ιδιοκτήτη του ιστότοπου (Snell & Menaldo, 2016).

Βλάβη στον ιστότοπο

Εάν η εξαγωγή δεδομένων προκαλέσει υπερφόρτωση ή καταστροφή του ιστότοπου ή του διακομιστή, τότε το άτομο που ευθύνεται για τη ζημιά μπορεί να διωχθεί σύμφωνα με το νόμο "παραβίασης περιουσίας" ("trespass to chattels") (Dryer & Stockton, 2013). Ωστόσο, η ζημιά θα πρέπει να είναι υλική και να αποδεικνύεται εύκολα στο δικαστήριο, προκειμένου ο κάτοχος του διακομιστή να δικαιούται οικονομική αποζημίωση [4].

Ηθική του web scraping

Ενώ οι υπάρχοντες νόμοι και οι νομικές θεωρίες έχουν εφαρμοστεί στην εξαγωγή δεδομένων ιστού τόσο στα δικαστήρια όσο και στη νομική βιβλιογραφία, η ηθική του web scraping δεν συναντάται στη βιβλιογραφία. Υπάρχουν διάφορες θεωρήσεις σχετικά με την ηθική, αλλά στη συγκεκριμένη περίπτωση θεωρούμε την ηθική ως "ένα σύνολο εννοιών και αρχών που μας καθοδηγούν στον προσδιορισμό της συμπεριφοράς που βοηθά ή βλάπτει τα αισθανόμενα όντα" (Paul & Elder, 2006). Εκτός από την παραβίαση της υπάρχουσας νομοθεσίας, η εξαγωγή δεδομένων από το διαδίκτυο μπορεί να έχει ως αποτέλεσμα ακούσια βλάβη στα πρόσωπα που σχετίζονται με έναν συγκεκριμένο ιστότοπο, όπως ο κάτοχος ή οι χρήστες του. Αυτές οι επιβλαβείς συνέπειες είναι δύσκολο να προβλεφθούν (Light & McGrath, 2010), ωστόσο ορισμένες από αυτές συζητούνται παρακάτω.

Απόρρητο προσώπων

Ένα ερευνητικό έργο που βασίζεται σε δεδομένα που συλλέγονται από έναν ιστότοπο ενδέχεται να θέσει ακούσια σε κίνδυνο το απόρρητο των ατόμων που αλληλεπιδρούν με τον ιστότοπο (Mason, 1986). Για παράδειγμα, αντιστοιχίζοντας τα δεδομένα που συλλέγονται από έναν ιστότοπο με άλλες πηγές εντός και εκτός διαδίκτυου, ένας ερευνητής μπορεί ακούσια να αποκαλύψει την ταυτότητα αυτών που δημιούργησαν τα δεδομένα (Ives & Krotov, 2006). Ακόμη και αν δεν παραβιάζεται το απόρρητο, το ζήτημα είναι ότι οι χρήστες του ιστότοπου ενδέχεται να μην έχουν συναινέσει σε χρήση των δεδομένων τους από τρίτους. Επομένως, η χρήση αυτών των δεδομένων χωρίς συναίνεση αποτελεί παραβίαση των δικαιωμάτων των υποκειμένων της έρευνας (Buchanan, 2017). Αυτές οι παραβιάσεις μπορούν να οδηγήσουν σε σοβαρές συνέπειες για τον κάτοχο του ιστότοπου, δεδομένης της έντονης ανησυχίας για το απόρρητο στο διαδίκτυο και έπειτα από τα σκάνδαλα που έχουν έρθει κατά καιρούς στη δημοσιότητα, όπως αυτά των εταιρειών Facebook και Cambridge Analytica.

Απόρρητο οργανισμών και εμπορικά μυστικά

Ακριβώς όπως τα άτομα έχουν δικαίωμα στην ιδιωτική ζωή, ομοίως και οι οργανισμοί έχουν το δικαίωμα να διατηρούν εμπιστευτικές ορισμένες πτυχές της λειτουργίας τους (Mason, 1986). Η αυτόματη συλλογή δεδομένων από το διαδίκτυο μπορεί να αποκαλύψει ακούσια εμπορικά μυστικά ή εμπιστευτικές πληροφορίες σχετικά με τον οργανισμό που κατέχει τον ιστότοπο. Για παράδειγμα, συλλέγοντας αυτόματα τις αγγελίες εργασίας από έναν ιστότοπο αγγελιών μπορεί κανείς να εκτιμήσει το μερίδιο αγοράς και τα έσοδα του ιστότοπου. Μπορεί επίσης να αποκαλύψει ορισμένες λεπτομέρειες και πιθανώς ορισμένες αδυναμίες στον τρόπο αποθήκευσης των δεδομένων στον ιστότοπο (Ives & Krotov, 2006). Όλα αυτά μπορούν να βλάψουν τη φήμη της εταιρείας που κατέχει τον ιστότοπο και να οδηγήσουν σε οικονομικές απώλειες.

Απώλεια εσόδων

Όταν κάποιος αποκτά πρόσβαση σε έναν ιστότοπο χωρίς να χρησιμοποιεί τη διεπαφή ιστού που έχει κατασκευαστεί για τους χρήστες, δεν εκτίθεται στις διαφημίσεις που προβάλλει ο ιστότοπος για την δημιουργία εσόδων. Επίσης, από ένα έργο web scraping μπορεί να προκύψει ένα προϊόν (π.χ. μια αναφορά) που να αποτρέπει τον πελάτη να αγοράσει από τον αρχικό κάτοχο των δεδομένων. Με άλλα λόγια, το προϊόν που δημιουργείται με τη βοήθεια του web scraping μπορεί άμεσα ή έμμεσα να ανταγωνίζεται την επιχείρηση που κατέχει τον ιστότοπο (Hirschey, 2014). Όλα τα παραπάνω μπορεί να οδηγήσουν σε οικονομικές απώλειες για τον ιδιοκτήτη του ιστότοπου ή σε άδικη κατανομή αξίας από τον κάτοχο των δεδομένων (Mason, 1986) [4].

2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

Στο συγκεκριμένο κεφάλαιο παρουσιάζονται ορισμένες χρήσιμες έννοιες που αφορούν την εξαγωγή δεδομένων από το διαδίκτυο. Αρχικά, γίνεται αναφορά στο μοντέλο αντικειμένων εγγράφου (DOM) και στη συνέχεια περιγράφονται οι διαθέσιμοι τρόποι για την επιλογή δεδομένων από έγγραφα HTML και XML.

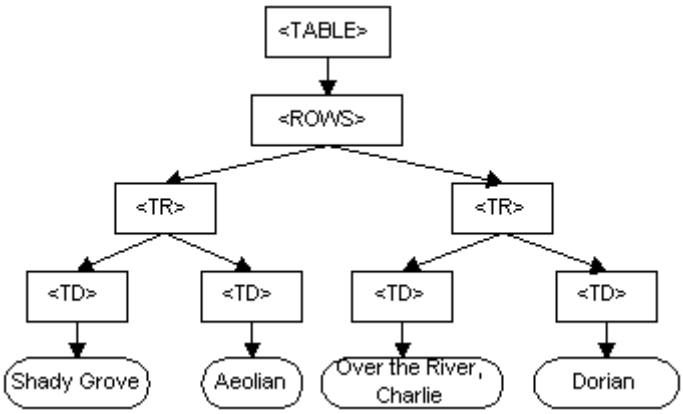
2.1 Μοντέλο Αντικειμένων Εγγράφου (DOM)

Το Μοντέλο Αντικειμένων Εγγράφου (Document Object Model, DOM) είναι μια διεπαφή προγραμματισμού εφαρμογών (API) για έγγραφα HTML και XML. Καθορίζει τη λογική δομή των εγγράφων και τον τρόπο πρόσβασης και χειρισμού ενός εγγράφου. Στην προδιαγραφή DOM ο όρος “έγγραφο” χρησιμοποιείται με την ευρεία έννοια. Η XML χρησιμοποιείται ως τρόπος αναπαράστασης πολλών διαφορετικών ειδών πληροφορίας που μπορούν να αποθηκευτούν σε διαφορετικά συστήματα, και μεγάλο μέρος αυτών των πληροφοριών παραδοσιακά θεωρούνται δεδομένα και όχι έγγραφα. Ωστόσο, η XML παρουσιάζει αυτά τα δεδομένα ως έγγραφα και το DOM μπορεί να χρησιμοποιηθεί για τη διαχείριση αυτών των δεδομένων [5].

Με το DOM οι προγραμματιστές μπορούν να δημιουργούν έγγραφα, να περιηγούνται στη δομή τους και να προσθέτουν, να μετατρέπουν ή να διαγράφουν στοιχεία και περιεχόμενο. Μέσω του DOM οτιδήποτε βρίσκεται σε ένα έγγραφο HTML ή XML μπορεί να προσπελαστεί, να αλλάξει ή να διαγραφεί με ελάχιστες εξαιρέσεις [5].

Το DOM μοιάζει πολύ με τη δομή των εγγράφων που μοντελοποιεί. Στην Εικόνα 3 φαίνεται ένας πίνακας HTML και η αναπαράστασή του σε δέντρο DOM.

```
<TABLE>
  <ROWS>
     <TR>
        <TD>Shady Grove</TD>
        <TD>Aeolian</TD>
     </TR>
     <TR>
        <TD>Over the River,
Charlie</TD>
        <TD>Dorian</TD>
     </TR>
  </ROWS>
</TABLE>
```



Εικόνα 3. Παράδειγμα δέντρου DOM [5]

Ένα δέντρο DOM είναι η ιεραρχική αναπαράσταση ενός εγγράφου HTML ή XML. Αποτελείται από έναν ριζικό κόμβο (root node), που είναι το ίδιο το έγγραφο, και μια σειρά θυγατρικών κόμβων (child nodes) που αντιπροσωπεύουν τα στοιχεία (elements), τα χαρακτηριστικά (attributes) και το περιεχόμενο κειμένου του εγγράφου. Κάθε κόμβος στο δέντρο (εκτός από τον ριζικό κόμβο) έχει μόνο έναν γονικό κόμβο (parent node), ενώ μπορεί να έχει πολλούς θυγατρικούς κόμβους.

Ο τρόπος αναπαράστασης ενός εγγράφου στο δέντρο DOM έχει ως εξής:

- Τα στοιχεία ενός εγγράφου HTML ή XML παριστάνονται ως κόμβοι στο δέντρο DOM. Κάθε κόμβος στοιχείου (element node) έχει μια ετικέτα (tag), χαρακτηριστικά και μπορεί να περιέχει άλλους κόμβους στοιχείων ή κόμβους κειμένου ως θυγατρικούς.
- Το κείμενο που περιέχεται σε ένα στοιχείο παριστάνεται ως κόμβος κειμένου (text node) στο δέντρο DOM. Οι κόμβοι κειμένου δεν έχουν χαρακτηριστικά ή θυγατρικούς κόμβους και είναι πάντα κόμβοι φύλλων (leaf nodes) στο δέντρο.
- Τα χαρακτηριστικά ενός στοιχείου παριστάνονται ως ιδιότητες του κόμβου στοιχείου στο δέντρο DOM.

Ο έλεγχος του δέντρου DOM μπορεί να γίνει με τη χρήση JavaScript ή άλλων γλωσσών προγραμματισμού. Οι πιο συνηθισμένες λειτουργίες περιλαμβάνουν πλοήγηση στο δέντρο, προσθήκη, αφαίρεση και τροποποίηση των κόμβων, καθώς και λήψη ή ρύθμιση των ιδιοτήτων των κόμβων. Το DOM API παρέχει ένα σύνολο μεθόδων και ιδιοτήτων για την εκτέλεση αυτών των λειτουργιών, όπως είναι οι getElementById, createElement, appendChild και innerHTML [6].

Κάθε φορά που ανοίγει μια ιστοσελίδα, το πρόγραμμα περιήγησης ανακτά τον κώδικα HTML της σελίδας. Ωστόσο, ο κώδικας HTML είναι απλά μια αναπαράσταση της ιεραρχίας των αντικειμένων του εγγράφου και ο browser χρειάζεται πρώτα να τον αναλύσει σε μια εσωτερική δομή δέντρου, την οποία θα χρησιμοποιήσει στη συνέχεια για τις επόμενες ενέργειες. Η δομή αυτή είναι το DOM.

Για να δει κανείς το DOM μιας ιστοσελίδας που έχει φορτώσει ο browser, μπορεί να ανοίξει το παράθυρο Developer Tools και να δει τη δενδρική αναπαράσταση του εγγράφου στην καρτέλα "Elements" (στον Chrome) ή στην καρτέλα "Inspector" (στον Firefox). Ο χρήστης βλέπει το δέντρο DOM που χρησιμοποιεί ο browser εκείνη τη στιγμή για να προβάλει τη σελίδα. Αν ο ιστότοπος χρησιμοποιεί JavaScript για να τροποποιήσει το περιεχόμενό του, ο κώδικας JavaScript μπορεί να αλλάξει το δέντρο DOM και τελικά να είναι διαφορετικό από τον αρχικά απεσταλμένο κώδικα HTML, που φαίνεται με την επιλογή "View Page Source".

2.2 Επιλογή στοιχείων σε έγγραφα HTML

Ένα σημαντικό βήμα κατά την εξαγωγή δεδομένων ιστού είναι να περιγράψει ο χρήστης στον υπολογιστή το περιεχόμενο που αναζητά. Υπάρχουν διάφορα εργαλεία για την περιγραφή μοτίβων στη δομή ενός εγγράφου HTML ούτως ώστε να επιλεγεί και να εξαχθεί το κατάλληλο περιεχόμενο. Τα σημαντικότερα από αυτά τα εργαλεία είναι τα εξής:

- **Εκφράσεις XPath**. Οι εκφράσεις XPath περιγράφουν τμήματα ενός εγγράφου με δομή δέντρου, το οποίο μπορεί να είναι XML ή HTML. Η γλώσσα XPath μπορεί να περιγράψει με μεγάλη ακρίβεια ποιοι κόμβοι θα συμπεριληφθούν ή θα εξαιρεθούν από την αναζήτηση.
- **Επιλογείς CSS** (CSS selectors). Οι επιλογείς CSS λειτουργούν όπως και οι εκφράσεις XPath για την επιλογή τμημάτων ενός εγγράφου HTML. Ωστόσο, σχεδιάστηκαν για web development (για την μορφοποίηση των στοιχείων ενός εγγράφου, π.χ. για τον ορισμό του χρώματος) και οι δυνατότητες που προσφέρουν είναι περιορισμένες συγκριτικά με τις εκφράσεις XPath. Κάθε επιλογέας CSS μπορεί να μετατραπεί σε μια ισοδύναμη έκφραση XPath.
- **Κανονικές εκφράσεις** (regular expressions). Οι κανονικές εκφράσεις περιγράφουν ένα μοτίβο χαρακτήρων. Μπορούν να χρησιμοποιηθούν για τον εντοπισμό δεδομένων, όπως μια ημερομηνία (με τη μορφή yyyy-mm-dd, d/m/yyyy κλπ) ή μια διεύθυνση email ή για την επιλογή των URL από τα οποία ο χρήστης ενδιαφέρεται να συλλέξει δεδομένα.

2.2.1 Εκφράσεις XPath

Η XPath (XML Path Language) χρησιμοποιεί εκφράσεις ή παραστάσεις διαδρομής (path expressions). Κάθε έκφραση διαδρομής ορίζει τον τρόπο με τον οποίο μπορούμε να προσδιορίσουμε έναν κόμβο ή ένα σύνολο κόμβων στη δενδρική αναπαράσταση ενός εγγράφου XML. Υπάρχουν δύο τύποι εκφράσεων διαδρομής [7]:

- Απόλυτες (absolute), οι οποίες ξεκινούν με κάθετο (/) που αναφέρεται στη ρίζα του εγγράφου.
- Σχετικές (relative), οι οποίες ξεκινούν χωρίς την κάθετο (/), με ένα στοιχείο. Δηλαδή αναφέρονται σε κάποιο κόμβο αναφοράς.

Στην XPath υπάρχουν επτά είδη κόμβων: element (στοιχείου), attribute (χαρακτηριστικού), text (κειμένου), namespace (χώρου ονομάτων), processing-instruction (εντολής επεξεργασίας), comment (σχολίων) και document (root) (εγγράφου ή ρίζας). Οι σχέσεις μεταξύ των κόμβων είναι: parent (γονέας), child (παιδί), sibling (αμφιθαλής), ancestor (πρόγονος), descendant (απόγονος) [7]. Για την επιλογή-προσπέλαση των κόμβων χρησιμοποιούνται τα παρακάτω αναγνωριστικά:

- nodename για επιλογή όλων των κόμβων με το όνομα “nodename”
- / για επιλογή από τον κόμβο root
- // για επιλογή των κόμβων του εγγράφου (από τον τρέχοντα κόμβο) που ταιριάζουν με την επιλογή ανεξαρτήτως που βρίσκονται
- . για επιλογή του τρέχοντα κόμβου
- .. για επιλογή του γονέα του τρέχοντα κόμβου
- @ για επιλογή χαρακτηριστικών (attributes)

Ο Πίνακας 1 περιέχει ορισμένες εκφράσεις XPath για την προσπέλαση των κόμβων του εγγράφου XML που φαίνεται στον Κώδικα 1 [7].

```
<?xml version="1.0" encoding="UTF-16"?>
<!DOCTYPE library PUBLIC "library.dtd">
<library location="Bremen">
   <author name="Henry Wise">
      <book title="AI"/>
      <book title="Modern Web"/>
   </author>
   <author name="William Smart">
       <book title="AI"/>
   </author>
   <author name="Cynthia Singleton">
       <book title="Semantic Web"/>
   </author>
   <publisher>
       <name>Wiley</name>
       <year>1920</year>
   </publisher>
</library>
```

Κώδικας 1. Παράδειγμα εγγράφου XML

Πίνακας 1. Εκφράσεις XPath: επιλογή - προσπέλαση κόμβων [7]

Εκφράσεις διαδρομής	**Ερμηνεία**
library	όλα τα παιδιά του στοιχείου
/library	το στοιχείο-ρίζα library
/library/author	όλα τα στοιχεία author που είναι παιδιά του library
/library/@location	όλα τα χαρακτηριστικά location που είναι σε παιδιά του library
//author	όλα τα στοιχεία author όπου κι αν βρίσκονται
/library//author	όλα τα στοιχεία author που είναι απόγονοι του library, όπου κι αν είναι
//@name	όλα τα χαρακτηριστικά name
//book/@title="AI"	όλοι οι κόμβοι title σε οποιοδήποτε κόμβο book με τιμή "AI"
//book[@title="AI"]	όλοι οι κόμβοι book που έχουν title με τιμή "AI"

Τα κατηγορήματα χρησιμοποιούνται για την εύρεση συγκεκριμένων κόμβων ή κόμβων που περιέχουν συγκεκριμένες τιμές. Οι τιμές αυτές μπαίνουν σε τετράγωνες παρενθέσεις. Στον Πίνακα 2 φαίνονται ορισμένα παραδείγματα για το ίδιο έγγραφο XML.

Πίνακας 2. Εκφράσεις XPath: κατηγορήματα [7]

Εκφράσεις διαδρομής	**Ερμηνεία**
/library/author[1]	το πρώτο στοιχείο author που είναι παιδί του στοιχείου library
/library/author[last()]	το τελευταίο στοιχείο author που είναι παιδί του στοιχείου library
/library/author[last()-1]	το προτελευταίο στοιχείο author που είναι παιδί του στοιχείου library
/library/author[position()<3]	τα δύο πρώτα στοιχεία author που είναι παιδιά του στοιχείου library
//book[@title]	όλα τα βιβλία που έχουν χαρακτηριστικό title
//book[@title="AI"]	όλοι οι κόμβοι book που έχουν title με τιμή "AI"
/library/publisher[year>1900]	όλα τα στοιχεία publisher που έχουν στοιχείο year με περιεχόμενο/τιμή μεγαλύτερο/η του 1900

2.2.2 Επιλογείς CSS

Η γλώσσα CSS (Cascading Style Sheets) χρησιμοποιείται για τη μορφοποίηση ενός εγγράφου HTML και επιτρέπει το διαχωρισμό των οπτικών χαρακτηριστικών (π.χ. χρώματα, γραμματοσειρές, θέση στοιχείων) από τη δομή ενός εγγράφου. Οι επιλογείς CSS προσφέρουν έναν εύκολο τρόπο για τον εντοπισμό στοιχείων σε ένα έγγραφο HTML, γι' αυτό χρησιμοποιούνται κατά κόρον στο frontend development με JavaScript και σε κάθε JavaScript framework. Οι περισσότερες γλώσσες προγραμματισμού υποστηρίζουν τους επιλογείς CSS είτε εγγενώς είτε μέσω third-party βιβλιοθηκών. Οι επιλογείς CSS είναι χρήσιμοι και για την εξαγωγή δεδομένων ιστού. Για τη συλλογή των δεδομένων από έναν ιστότοπο είναι απαραίτητος αρχικά ο εντοπισμός τους, και οι επιλογείς CSS αποτελούν χρήσιμο εργαλείο σε αυτή την περίπτωση. Ο Πίνακας 3 περιέχει ορισμένους από τους επιλογείς CSS που χρησιμοποιούνται συχνότερα.

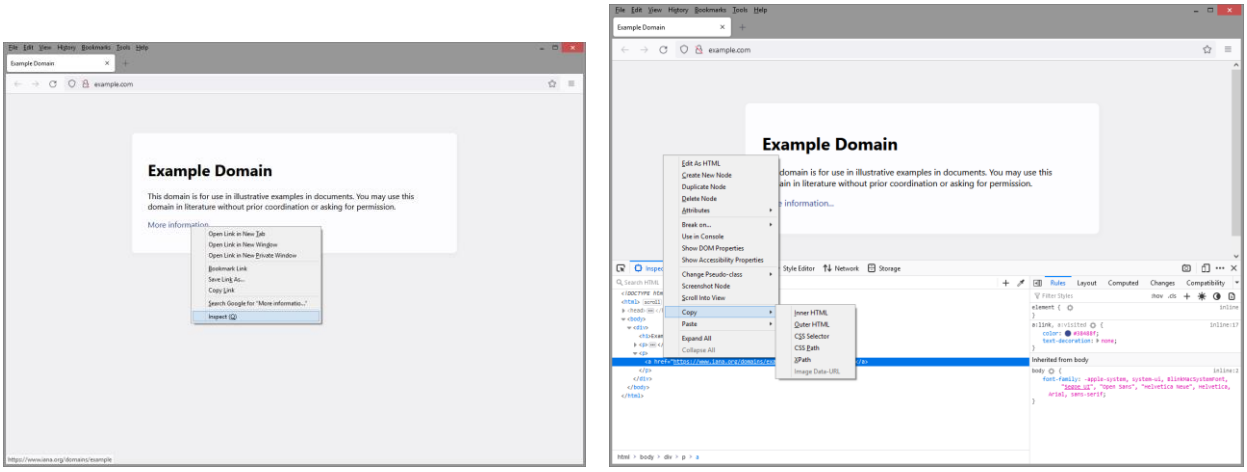
Μπορούμε να χωρίσουμε τους επιλογείς CSS σε πέντε κατηγορίες [8]:

- Απλοί επιλογείς. Η επιλογή των στοιχείων γίνεται βάσει ονόματος, αναγνωριστικού (id) ή κλάσης (class).
- Επιλογείς συνδυασμού (combinator selectors). Η επιλογή των στοιχείων γίνεται με βάση μια συγκεκριμένη σχέση μεταξύ τους. Υπάρχουν 4 διαφορετικοί τύποι: επιλογέας απογόνων (χαρακτήρας space), επιλογέας θυγατρικών στοιχείων (>), επιλογέας γειτονικών θυγατρικών στοιχείων (+), επιλογέας γενικών θυγατρικών στοιχείων (~).
- Επιλογείς ψευδο-κλάσης (pseudo-class selectors). Η επιλογή των στοιχείων γίνεται με βάση μια συγκεκριμένη κατάσταση.
- Επιλογείς ψευδο-στοιχείων (pseudo-elements selectors). Επιλέγεται ένα τμήμα του στοιχείου.
- Επιλογείς χαρακτηριστικών (attribute selectors). Τα στοιχεία επιλέγονται με βάση ένα χαρακτηριστικό τους ή την τιμή ενός χαρακτηριστικού.

Πίνακας 3. Επιλογείς CSS [8]

Επιλογέας	**Παράδειγμα**	**Ερμηνεία**
element	p	όλα τα στοιχεία <p>
.class	.intro	όλα τα στοιχεία με την κλάση class="intro"
#id	#firstname	όλα τα στοιχεία με το αναγνωριστικό id="firstname"
*	*	όλα τα στοιχεία του εγγράφου
element.class	p.intro	όλα τα στοιχεία <p> με κλάση class="intro"
.class1.class2	.name1.name2	όλα τα στοιχεία που περιέχουν τις τιμές name1 και name2 στο χαρακτηριστικό class
element element	div p	όλα τα στοιχεία <p> που βρίσκονται μέσα σε στοιχεία
element>element	div > p	όλα τα στοιχεία <p> που έχουν γονικό στοιχείο
element+element	div + p	το πρώτο στοιχείο <p> που βρίσκεται αμέσως μετά από ένα στοιχείο
element1~element2	p ~ ul	κάθε στοιχείο του οποίου προηγείται ένα στοιχείο <p>
[attribute]	[target]	όλα τα στοιχεία με το χαρακτηριστικό target
[attribute=value]	[target=_blank]	όλα τα στοιχεία με το χαρακτηριστικό target ="_blank"
[attribute*=value]	a[href*="w3schools"]	όλα τα στοιχεία <a> με το χαρακτηριστικό target η τιμή του οποίου περιέχει το substring "w3schools"
:first-child	p:first-child	κάθε στοιχείο <p> που είναι το πρώτο θυγατρικό στοιχείο του γονικού του
:first-of-type	p:first-of-type	κάθε στοιχείο <p> που είναι το πρώτο στοιχείο <p> του γονικού του
:nth-child(n)	p:nth-child(2)	κάθε στοιχείο <p> που είναι το δεύτερο στοιχείο <p> του γονικού του
:not(selector)	:not(p)	κάθε στοιχείο που δεν είναι στοιχείο <p>
::first-letter	p::first-letter	το πρώτο γράμμα κάθε στοιχείου <p>
::first-line	p::first-line	η πρώτη γραμμή κάθε στοιχείου <p>

Για να βρούμε τον επιλογέα CSS ενός στοιχείου στον browser, κάνουμε δεξί κλικ πάνω στο στοιχείο και επιλέγουμε "Inspect". Στην καρτέλα "Elements/Inspector" εμφανίζεται το στοιχείο μαρκαρισμένο και μπορούμε να πάρουμε τον επιλογέα CSS κάνοντας δεξί κλικ και επιλέγοντας "Copy - CSS Selector" (βλ. Εικόνα 4).



Εικόνα 4. Εύρεση CSS selector ενός στοιχείου στον browser

Ορισμένοι ιστότοποι, όπως για παράδειγμα το Twitter, προσθέτουν τυχαίες τιμές στα χαρακτηριστικά (attributes) των στοιχείων της σελίδας. Στην περίπτωση αυτή, ο επιλογέας CSS αλλάζει κάθε φορά που στέλνουμε ένα νέο request, γι' αυτό θα πρέπει να επιλέξουμε χαρακτηριστικά τα οποία παραμένουν αμετάβλητα. Η αντιγραφή του επιλογέα CSS από τον browser μπορεί να λειτουργήσει μόνο όταν ο ιστότοπος έχει καθαρή δομή με καλά ορισμένα αναγνωριστικά (id) και κλάσεις (class). Διαφορετικά ο χρήστης θα πρέπει να αναζητήσει άλλο τρόπο για την επιλογή των στοιχειών (βλ. Εικόνα 5).

Εικόνα 5. Παράδειγμα ιστότοπου με μεταβλητά attributes

2.2.3 Κανονικές εκφράσεις (regular expressions)

Κανονική έκφραση είναι μια συμβολοσειρά που ορίζει ένα μοτίβο αναζήτησης στο κείμενο. Γιατί όμως να επιλέξουμε τη συγκεκριμένη μέθοδο, ενώ υπάρχουν διαθέσιμες άλλες τεχνολογίες για την εξαγωγή δεδομένων από κώδικα HTML, όπως οι επιλογείς CSS και οι εκφράσεις XPath; Τα δεδομένα είναι εύκολο να εντοπιστούν και να εξαχθούν όταν περικλείονται σε κατάλληλα στοιχεία HTML και φέρουν χαρακτηριστικά (attributes) ανάλογα με τη σημασία τους. Υπάρχουν όμως περιπτώσεις στις οποίες μεγάλα τμήματα κειμένου βρίσκονται μέσα σε ένα στοιχείο <p>, για παράδειγμα. Τότε οι κανονικές εκφράσεις μπορούν να φανούν χρήσιμες για την εξαγωγή δεδομένων όπως τιμές, ημερομηνίες, τηλεφωνικοί αριθμοί κλπ.

Στους Πίνακες 4, 5 και 6 φαίνονται οι ειδικοί χαρακτήρες, οι ακολουθίες χαρακτήρων και τα σύνολα χαρακτήρων αντίστοιχα που έχουμε στη διάθεσή μας όταν χρησιμοποιούμε κανονικές εκφράσεις.

Πίνακας 4. Ειδικοί χαρακτήρες (metacharacters) [9]

Χαρακτήρας	**Περιγραφή**	**Παράδειγμα**
[]	Σύνολο χαρακτήρων	"[a-m]"
\	Αρχή ειδικής ακολουθίας ή διαφυγή ειδικών χαρακτήρων	"\d"
.	Οποιοσδήποτε χαρακτήρας εκτός από τον χαρακτήρα νέας γραμμής	"he..o"
^	Ξεκινάει με	"^hello"
$	Τελειώνει με	"planet$"
*	0 ή περισσότερες εμφανίσεις της έκφρασης που προηγείται	"he.*o"
+	1 ή περισσότερες εμφανίσεις της έκφρασης που προηγείται	"he.+o"
?	0 ή 1 εμφάνιση της έκφρασης που προηγείται	"he.?o"
{}	Η έκφραση που προηγείται εμφανίζεται τόσες φορές όσο και ο αριθμός που ορίζεται μέσα στο άγκιστρο	"he.{2}o"
|	Είτε το ένα είτε το άλλο	"falls|stays"

Πίνακας 5. Ειδικές ακολουθίες χαρακτήρων (special sequences) [9]

Χαρακτήρας	**Περιγραφή**	**Παράδειγμα**
\A	Μια συμβολοσειρά ξεκινά με τους συγκεκριμένους χαρακτήρες	"\AThe"
\b	Μια λέξη ξεκινά ή τελειώνει με τους συγκεκριμένους χαρακτήρες	r"\bain" r"ain\b"
\B	Μια λέξη περιέχει τους συγκεκριμένους χαρακτήρες αλλά όχι στην αρχή ή στο τέλος της	r"\Bain" r"ain\B"
\d	Ψηφίο (0-9)	"\d"
\D	Οποιοσδήποτε χαρακτήρας εκτός από ψηφίο	"\D"
\s	Χαρακτήρας κενού	"\s"
\S	Οποιοσδήποτε χαρακτήρας εκτός από κενό	"\S"

\w	Γράμμα μικρό ή κεφαλαίο (a-Z), ψηφίο (0-9) ή underscore (_)	"\w"
\W	Οποιοσδήποτε χαρακτήρας εκτός από γράμματα, ψηφία και underscore	"\W"
\Z	Μια συμβολοσειρά τελειώνει με τους συγκεκριμένους χαρακτήρες	"Spain\Z"

Πίνακας 6. Σύνολα χαρακτήρων (sets) [9]

Σύνολο	**Περιγραφή**
[arn]	Οποιοσδήποτε από τους χαρακτήρες a, r ή n
[a-n]	Οποιοδήποτε μικρό γράμμα μεταξύ a και n
[^arn]	Οποιοσδήποτε χαρακτήρας εκτός από a, r και n
[0123]	Οποιοδήποτε από τα ψηφία 0, 1, 2 ή 3
[0-9]	Οποιοδήποτε ψηφίο από 0 έως 9
[0-5][0-9]	Οποιοσδήποτε διψήφιος αριθμός μεταξύ 00 και 59
[a-zA-Z]	Οποιοδήποτε γράμμα μεταξύ a και z, είτε μικρό είτε κεφαλαίο
[+]	Στα σύνολα οι χαρακτήρες +, *, ., |, (), $,{} δεν έχουν κάποια ξεχωριστή σημασία, επομένως η έκφραση [+] αναζητά τον χαρακτήρα + στη συμβολοσειρά

3. ΕΞΑΓΩΓΗ ΔΕΔΟΜΕΝΩΝ ΙΣΤΟΥ ΚΑΙ ΓΛΩΣΣΑ PYTHON

Στο κεφάλαιο αυτό γίνεται σύγκριση της Python με άλλες γλώσσες προγραμματισμού και αναφέρονται οι λόγοι για τους οποίους αποτελεί τη δημοφιλέστερη γλώσσα για web scraping. Στη συνέχεια παρουσιάζονται οι κυριότερες βιβλιοθήκες που διαθέτει για web scraping: built-in βιβλιοθήκες, third-party βιβλιοθήκες, βιβλιοθήκες για ασύγχρονα αιτήματα και εργαλεία για αυτοματοποίηση προγραμμάτων περιήγησης.

3.1 Python: η δημοφιλέστερη γλώσσα για web scraping

Η Python χρησιμοποιείται σε πολλούς τομείς, όπως η ανάπτυξη εφαρμογών διαδικτύου, η μηχανική μάθηση (machine learning), η επιστήμη δεδομένων (data science), η ανάπτυξη βιντεοπαιχνιδιών και βεβαίως η εξαγωγή δεδομένων ιστού.

Τα χαρακτηριστικά της Python που την καθιστούν ιδανική για web scraping είναι τα εξής:

- Διαθέτει πολυάριθμες βιβλιοθήκες. Η Python διαθέτει μια πληθώρα βιβλιοθηκών για εξαγωγή δεδομένων ιστού, όπως οι BeautifulSoup, Selenium και lxml.
- Είναι εύχρηστη. Ο προγραμματισμός με Python είναι σχετικά εύκολος. Ασφαλώς χρειάζονται γνώσεις προγραμματισμού προκειμένου να γράψει κανείς κώδικα για web scraping, αλλά σε σύγκριση με άλλες γλώσσες η Python είναι ευκολότερη και η σύνταξή της είναι σαφής και ευανάγνωστη.
- Προσφέρει εξοικονόμηση χρόνου. Η εξαγωγή δεδομένων ιστού χρησιμοποιείται για να απλοποιήσει χρονοβόρες διαδικασίες, όπως η χειροκίνητη συλλογή τεράστιων ποσοτήτων δεδομένων. Ομοίως και η Python μπορεί με λίγες γραμμές κώδικα να ολοκληρώσει μια περίπλοκη εργασία. Είναι μια γλώσσα που επιλέγεται συχνά από τους προγραμματιστές για εξοικονόμηση χρόνου.
- Διαθέτει κοινότητα υποστήριξης. Η Python είναι μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού και διαθέτει ενεργή κοινότητα. Οι προγραμματιστές μοιράζονται τις γνώσεις τους και απαντούν σε ερωτήσεις, επομένως οι χρήστες που δυσκολεύονται κατά τη σύνταξη του κώδικα μπορούν εύκολα να αναζητήσουν βοήθεια.

Η Python είναι η πιο δημοφιλής γλώσσα για εξαγωγή δεδομένων ιστού, διότι μπορεί να χειριστεί σχεδόν όλες τις διαδικασίες που σχετίζονται με τη συλλογή των δεδομένων. Ωστόσο, υπάρχουν και άλλες γλώσσες που χρησιμοποιούνται σε αυτόν τον τομέα όπως είναι οι Ruby, C++, PHP και Java. Οι γλώσσες αυτές έχουν πλεονεκτήματα και μειονεκτήματα σε σύγκριση με την Python, ορισμένα από τα οποία αναφέρονται παρακάτω [10].

Python και Ruby

Η Ruby είναι μια διερμηνευόμενη, υψηλού επιπέδου και γενικής χρήσης γλώσσα προγραμματισμού. Η σύνταξη της είναι απλή και ευανάγνωστη σε σύγκριση με άλλες γλώσσες. Χρησιμοποιείται κυρίως για τη δημιουργία web εφαρμογών, αλλά αυτή δεν είναι η μόνη περίπτωση χρήσης, καθώς η Ruby είναι κατάλληλη και για εξαγωγή δεδομένων ιστού. Διαθέτει αρκετές βιβλιοθήκες για web scraping, όπως οι NokoGiri και HTTParty. Με τη χρήση αυτών των βιβλιοθηκών οι προγραμματιστές μπορούν να δημιουργήσουν πλήρως λειτουργικούς web scrapers. Η βιβλιοθήκη NokoGiri περιλαμβάνει αναλυτές (parsers) XML και HTML, και υποστηρίζει επιλογείς XPath ή CSS. Η βιβλιοθήκη HTTParty χρησιμοποιείται για την αποστολή HTTP αιτημάτων.

Τα μειονεκτήματα της Ruby σε σύγκριση με την Python είναι τα εξής:

1. Η Ruby είναι πιο αργή σε σχέση με την Python. Και οι δύο είναι διερμηνευόμενες (interpreted) γλώσσες προγραμματισμού. Αυτό σημαίνει ότι είναι πιο αργές από τις μεταγλωττιζόμενες (compiled) γλώσσες όπως η C++. Ωστόσο, η Python είναι ταχύτερη από τη Ruby και η ταχύτητα αποτελεί σημαντικό παράγοντα στην εξαγωγή δεδομένων ιστού.
2. Η Ruby δεν χρησιμοποιείται τόσο ευρέως όσο η Python. Οι προγραμματιστές που είναι αρχάριοι στον τομέα του web scraping θα δυσκολευτούν να βρουν τεκμηρίωση. Επίσης, είναι δύσκολο να βρουν βοήθεια όταν αντιμετωπίζουν προβλήματα στον κώδικά τους [10].

Python και C++

Η C++ είναι γλώσσα προγραμματισμού γενικού σκοπού. Χρησιμοποιείται ευρέως για την ανάπτυξη λειτουργικών συστημάτων, βιντεοπαιχνιδιών, προγραμμάτων περιήγησης και άλλων πολύπλοκων συστημάτων όπου απαιτείται κωδικοποίηση σε επίπεδο υλικού. Είναι μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού παγκοσμίως.

Ωστόσο, εάν πρέπει να επιλέξει κανείς μια γλώσσα προγραμματισμού για web scraping, η Python είναι καλύτερη από την C++ ειδικά για αρχάριους.

1. Η C++ δεν είναι η καλύτερη επιλογή για προγραμματισμό που σχετίζεται με το web, καθώς είναι στατική γλώσσα προγραμματισμού. Η συγγραφή κώδικα για web scraping είναι ευκολότερη με μια δυναμική γλώσσα προγραμματισμού, όπως η Python.
2. Η C++ είναι πιο δύσκολη στην εκμάθηση σε σύγκριση με την Python,. Αν θέλει κάποιος να συλλέξει δεδομένα από το διαδίκτυο, είναι προτιμότερο να εξοικονομήσει χρόνο και να μάθει Python, καθώς το κύριο χαρακτηριστικό της Python είναι ότι επιτρέπει στους προγραμματιστές να υλοποιούν μεγάλα task σε λιγότερες γραμμές κώδικα. Ωστόσο, η C++ είναι μια εξαιρετική γλώσσα προγραμματισμού. Αξίζει κανείς να τη μάθει για την συγγραφή

κώδικα σε έργα με αυξημένες απαιτήσεις, που οι προγραμματιστές δεν μπορούν να υλοποιήσουν χωρίς τη C++ [10].

Παρόλο που η C++ δεν συνιστάται για τη δημιουργία προγραμμάτων ανίχνευσης, υπάρχει διαθέσιμη η βιβλιοθήκη libcurl που μπορεί να χρησιμοποιηθεί για την ανάκτηση διευθύνσεων URL. Το πλεονέκτημα της C++ είναι ότι προσφέρει υψηλή απόδοση και ταχύτητα, καθώς είναι μεταγλωττιζόμενη γλώσσα προγραμματισμού [10].

Python και PHP

Η PHP είναι μια ανοιχτού κώδικα, γενικού σκοπού γλώσσα προγραμματισμού, η οποία χρησιμοποιείται ευρέως για τη δημιουργία εφαρμογών διαδικτύου. Διαθέτει βιβλιοθήκες για web scraping, όπως είναι οι Goutte, cURL, HTTPful και άλλες. Η βιβλιοθήκη Goutte παρέχει διάφορα API για την ανίχνευση ιστοτόπων και την εξαγωγή δεδομένων από έγγραφα HTML και XML. Η cURL είναι μια από τις πιο δημοφιλείς βιβλιοθήκες για την υποβολή αιτημάτων HTTP με PHP. Η HTTPful είναι μια βιβλιοθήκη PHP που χρησιμοποιείται για αιτήματα HTTP, κάνοντας τα πιο ευανάγνωστα για τους προγραμματιστές.

Παρ' όλα αυτά οι προγραμματιστές σπάνια επιλέγουν την PHP για web scraping, καθώς δεν είναι εύκολη η συγγραφή προγραμμάτων ανίχνευσης ιστού με τη συγκεκριμένη γλώσσα. Για προγραμματιστές αρχάριους στην εξαγωγή δεδομένων ιστού προτείνεται η χρήση της Python που είναι λιγότερο περίπλοκη και πιο εύκολη στην εκμάθηση από την PHP [10].

Python και Java

Η Java είναι από τις πιο διαδεδομένες γλώσσες προγραμματισμού στον κόσμο, γι’ αυτό χρησιμοποιείται συχνά και για web scraping. Διαθέτει μια πληθώρα εργαλείων, βιβλιοθηκών και εξωτερικών API που μπορούν να χρησιμοποιηθούν για τη δημιουργία ενός web scraper, όπως είναι τα JSoup, HTMLUnit και Jaunt. Η JSoup είναι μια απλή βιβλιοθήκη για την εξαγωγή και τον χειρισμό των δεδομένων μέσω της περιήγησης του DOM και υποστηρίζει επιλογή στοιχείων με CSS selectors. Το HTMLUnit είναι ένα framework που επιτρέπει την προσομοίωση διάφορων συμβάντων σε ιστοσελίδες, όπως κλικ και υποβολή φόρμας. Η Jaunt είναι μια βιβλιοθήκη για αυτοματισμό του προγράμματος περιήγησης και άντληση δεδομένων. Είναι χρήσιμη για την εξαγωγή δεδομένων από σελίδες HTML και από JSON payload.

Η Java δεν συνιστά την καλύτερη επιλογή για απαιτητικά έργα web scraping. Ωστόσο, μπορεί να χρησιμοποιηθεί για την κατασκευή ισχυρών web scraper για διάφορους σκοπούς. Η γλώσσα Java χρησιμοποιείται από μεγάλο αριθμό επιχειρήσεων παγκοσμίως και, ως εκ τούτου, διαθέτει μεγάλη κοινότητα, η οποία μπορεί να βοηθήσει νέους ή άπειρους προγραμματιστές στην αντιμετώπιση προβλημάτων [11].

Η Python είναι ιδανική για την εξαγωγή δεδομένων από το διαδίκτυο, καθώς διαθέτει ένα μεγάλο αριθμό βιβλιοθηκών και μια ενεργή κοινότητα για αναζήτηση βοήθειας. Άλλοι λόγοι για τη χρήση της Python σε έργα web scraping είναι η εύκολη εκμάθηση, ο ευανάγνωστος κώδικας και η απλή σύνταξη της γλώσσας.

3.2 Βιβλιοθήκες Python για web scraping

3.2.1 socket

Στην αρχιτεκτονική client - server οι υποδοχές (sockets) επιτρέπουν την επικοινωνία μεταξύ δύο διαφορετικών διεργασιών στον ίδιο ή σε διαφορετικούς υπολογιστές. Η Python διαθέτει το ενσωματωμένο (built-in) module socket, με το οποίο μπορούμε να υποβάλλουμε ένα αίτημα HTTP ανοίγοντας ένα TCP socket. Η συνάρτηση `socket` επιστρέφει ένα αντικείμενο socket, το οποίο διαθέτει κατάλληλες μεθόδους για τη σύνδεση με τον εξυπηρετητή, την αποστολή του αιτήματος και τη λήψη της απόκρισης.

Όπως φαίνεται στον Κώδικα 2, αρχικά γίνεται import το module socket και στη συνέχεια δημιουργείται το αντικείμενο `socket` με παραμέτρους AF_INET και SOCK_STREAM. Η πρώτη παράμετρος (AF_INET) αναφέρεται στο address family IPv4 και η δεύτερη παράμετρος (SOCK_STREAM) δηλώνει connection-oriented επικοινωνία με πρωτόκολλο TCP. Η σύνδεση πραγματοποιείται με τη μέθοδο `connect`, αφού προσδιοριστούν η διεύθυνση URL (ή η διεύθυνση IP) και η θύρα (port) του εξυπηρετητή.

```
import socket

# create an INET, STREAMing socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# now connect to the web server on port 80 - the normal http port
s.connect(("www.python.org", 80))

request_header = b'GET / HTTP/1.0\r\nHost: www. python.org \r\n\r\n'
s.sendall(request_header)

response = ''
while True:
    recv = s.recv(1024)
    if not recv:
        break
    response += str(recv)

print(response)
s.close()
```

Κώδικας 2. Παράδειγμα χρήσης της βιβλιοθήκης socket

Όταν ολοκληρωθεί η σύνδεση με τον εξυπηρετητή, το socket s μπορεί να χρησιμοποιηθεί για την αποστολή του αιτήματος. Το ίδιο socket θα διαβάσει την απάντηση και στη συνέχεια θα καταστραφεί. Αυτό συμβαίνει διότι τα client sockets χρησιμοποιούνται συνήθως για μια ανταλλαγή μηνυμάτων (ή ένα μικρό αριθμό διαδοχικών ανταλλαγών).

Το HTTP αίτημα αποστέλλεται με τη μέθοδο `send` ή `sendall`. Η μέθοδος `send` δεν διασφαλίζει την αποστολή όλων των δεδομένων. Οι εφαρμογές είναι υπεύθυνες για τον έλεγχο της αποστολής και αν διαπιστωθεί ότι η μεταφορά δεν ολοκληρώθηκε, επιχειρείται η αποστολή και των υπόλοιπων δεδομένων. Αντίθετα, η μέθοδος `sendall` αποστέλλει τα δεδομένα μέχρι τέλους ή μέχρι να παρουσιαστεί σφάλμα. Άλλες μέθοδοι για την αποστολή δεδομένων είναι οι `sendto`, `sendmsg`, `sendfile` κλπ, αλλά δε θα επεκταθούμε περισσότερο.

Για τη λήψη των δεδομένων που αποστέλλει ο εξυπηρετητής χρησιμοποιείται η μέθοδος `recv`, η οποία επιστρέφει ένα αντικείμενο bytes. Η `recv` παίρνει σαν παράμετρο το μέγεθος του buffer, στον οποίο θα τοποθετηθούν τα δεδομένα. Η τιμή αυτή είναι συνήθως μια δύναμη του 2 [12].

3.2.2 re

Η Python διαθέτει την ενσωματωμένη βιβλιοθήκη re για τη διαχείριση κανονικών εκφράσεων (regular expressions). Οι πιο συνηθισμένες μέθοδοι της βιβλιοθήκης είναι οι [13]:

- `re.search(pattern, string, flags=0)`: αναζητά την κανονική έκφραση pattern οπουδήποτε μέσα στη συμβολοσειρά string και επιστρέφει ένα αντικείμενο Match εάν βρεθεί.
- `re.match(pattern, string, flags=0)`: αναζητά την κανονική έκφραση pattern στην αρχή της συμβολοσειράς string και επιστρέφει ένα αντικείμενο Match εάν βρεθεί.
- `re.fullmatch(pattern, string, flags=0)`: επιστρέφει ένα αντικείμενο Match όταν ολόκληρη η συμβολοσειρά string ταιριάζει με την κανονική έκφραση pattern.
- `re.split(pattern, string, maxsplit=0, flags=0)`: διασπά τη συμβολοσειρά string στα σημεία που γίνεται αντιστοίχιση με την κανονική έκφραση pattern και επιστρέφει μια λίστα με τα επιμέρους τμήματα του string.
- `re.findall(pattern, string, flags=0)`: επιστρέφει μια λίστα που περιέχει όλες τις αντιστοιχίσεις του pattern στο string. Η συμβολοσειρά string ελέγχεται από τα αριστερά προς τα δεξιά και οι αντιστοιχίσεις επιστρέφονται με την ίδια σειρά που εντοπίστηκαν.

Στις κανονικές εκφράσεις, ο χαρακτήρας backslash ('\') χρησιμοποιείται είτε ως χαρακτήρας διαφυγής είτε για να δηλώσει την αρχή μιας ειδικής ακολουθίας. Αυτό έρχεται σε σύγκρουση με τη χρήση του backslash στις συμβολοσειρές της Python. Για παράδειγμα, αν θέλουμε να

αναζητήσουμε τον χαρακτήρα backslash σε μια συμβολοσειρά, το μοτίβο (pattern) θα πρέπει να είναι '\\\\' γιατί η κανονική έκφραση είναι \\ και κάθε backslash εκφράζεται στην Python ως \\. Η λύση είναι να χρησιμοποιηθεί το πρόθεμα 'r' πριν την έκφραση, ώστε οι χαρακτήρες διαφυγής να αντιμετωπίζονται ως απλοί χαρακτήρες. Για παράδειγμα, το r"\n" είναι μια συμβολοσειρά δύο χαρακτήρων που περιέχει τα '\' και 'n', ενώ το "\n" είναι μια συμβολοσειρά που περιέχει μόνο το χαρακτήρα αλλαγής γραμμής.

3.2.3 urllib3

Η βιβλιοθήκη urllib3 είναι ένας εύχρηστος HTTP client για την Python που δεν ανήκει στα ενσωματωμένα πακέτα της γλώσσας. Η urllib3 προσφέρει δυνατότητες που λείπουν από τις standard βιβλιοθήκες όπως:

- Ασφάλεια νήματος (thread safety)
- Caching συνδέσεων (connection pooling)
- Επαλήθευση TLS/SSL στην πλευρά του client
- Μεταφόρτωση αρχείων με multipart αιτήματα
- Βοηθητικά εργαλεία για την επανάληψη HTTP αιτημάτων και για τη διαχείριση ανακατευθύνσεων
- Υποστήριξη για κωδικοποίηση gzip, deflate και brotli
- Υποστήριξη διακομιστή μεσολάβησης (proxy) για HTTP και SOCKS
- 100% κάλυψη testing (test coverage).

Για να γίνει ένα αίτημα HTTP με τη βιβλιοθήκη urllib3, γίνεται import το module urllib3 και δημιουργείται ένα αντικείμενο `PoolManager`, το οποίο διαχειρίζεται όλες τις λεπτομέρειες σχετικά με το σύνολο των συνδέσεων (connection pooling) και την ασφάλεια νήματος (thread safety). Για την αποστολή του αιτήματος χρησιμοποιείται η μέθοδος `request()`, η οποία επιστρέφει ένα αντικείμενο `HTTPResponse`. Η `request()` είναι κατάλληλη για την αποστολή HTTP αιτημάτων με όλες τις μεθόδους (GET, POST, PUT, DELETE, HEAD).

```
>>> import urllib3
>>> http = urllib3.PoolManager()
>>> r = http.request('GET', 'http://httpbin.org/ip')
>>> r.status
200
>>> r.data
b'{\n "origin": "104.232.115.37"\n}\n'
>>> r.headers
HTTPHeaderDict({'Content-Length': '33', ...})
```

Κώδικας 3. Αποστολή αιτήματος HTTP με τη βιβλιοθήκη urllib3

Το αντικείμενο `HTTPResponse` έχει τα χαρακτηριστικά `status`, `data` και `headers`, τα οποία επιστρέφουν τον κωδικό κατάστασης HTTP, το περιεχόμενο της απόκρισης σε μορφή bytes (bytestring) και τις κεφαλίδες της απόκρισης αντίστοιχα.

Η μέθοδος `request()` διαθέτει, επίσης, το όρισμα `headers` στο οποίο μπορούν να δηλωθούν οι κεφαλίδες HTTP με τη μορφή λεξικού. Για αιτήματα GET, HEAD και DELETE, ο καθορισμός των query παραμέτρων γίνεται με τη μορφή λεξικού και μέσω του ορίσματος `fields` της `request()`. Για αιτήματα POST και PUT, οι query παράμετροι θα πρέπει πρώτα να κωδικοποιηθούν και έπειτα να προστεθούν στο URL. Για την αποστολή των πεδίων μιας φόρμας HTTP με POST ή PUT, τα δεδομένα εισάγονται με τη μορφή λεξικού στο όρισμα `fields` της `request()`. Η βιβλιοθήκη urllib3 υποστηρίζει, επίσης, την αποστολή δεδομένων JSON, την αποστολή αρχείων και binary δεδομένων.

```
# Headers
>>> r = http.request(
        'GET',
        'http://httpbin.org/headers',
         headers = {
            'X-Something': 'value'
        }
)
# Query Parameters (GET, HEAD και DELETE)
>>> r = http.request(
        'GET',
        'http://httpbin.org/get',
        fields = {'arg': 'value'}
)

# Query Parameters (POST, PUT)
>>> from urllib.parse import urlencode
>>> encoded_args = urlencode({'arg': 'value'})
>>> url = 'http://httpbin.org/post?' + encoded_args
>>> r = http.request('POST', url)

# Form Data (POST, PUT)
>>> r = http.request(
        'POST',
        'http://httpbin.org/post',
        fields = {'field': 'value'}
)
```

Κώδικας 4. Ορισμός κεφαλίδων και query παραμέτρων με τη βιβλιοθήκη urllib3

Από την έκδοση 1.25 της βιβλιοθήκης και έπειτα, οι συνδέσεις HTTPS επαληθεύονται από προεπιλογή με την παράμετρο `cert_reqs='CERT_REQUIRED'`. Η urllib3 φορτώνει τα πιστοποιητικά SSL που βρίσκονται αποθηκευμένα στο σύστημα, εκτός αν ορίζεται διαφορετικά. Ωστόσο, η πιο αξιόπιστη επιλογή είναι η χρήση του package certifi το οποίο παρέχει το πακέτο

των root πιστοποιητικών της Mozilla. Το αντικείμενο `PoolManager` αναλαμβάνει την επαλήθευση του πιστοποιητικού και εγείρει μια εξαίρεση SSLError αν η επαλήθευση αποτύχει.

Ένα άλλο χαρακτηριστικό της urllib3 είναι ότι υποστηρίζει timeouts, δηλαδή ορίζεται ο χρόνος (σε δευτερόλεπτα) που επιτρέπεται να εκτελούνται τα αιτήματα προτού ακυρωθούν. Ο χρόνος αυτός περνάει ως float αριθμός στο όρισμα `timeout` της `request()`. Για καλύτερο έλεγχο μπορεί να χρησιμοποιηθεί ένα αντικείμενο `Timeout`, προκειμένου να καθοριστούν διαφορετικοί χρόνοι σύνδεσης και ανάγνωσης. Η urllib3 μπορεί αυτόματα να ξαναδοκιμάζει idempotent αιτήματα και με τον ίδιο μηχανισμό διαχειρίζεται τις ανακατευθύνσεις. Ο αριθμός των επαναλήψεων δηλώνεται στην παράμετρο retries της `request()`. Από προεπιλογή η urllib3 ξαναδοκιμάζει τα αιτήματα τρεις φορές και ακολουθεί έως τρεις ανακατευθύνσεις [14].

3.2.4 Requests

Η βιβλιοθήκη requests της Python χρησιμοποιείται για την εύκολη αποστολή HTTP/1.1 αιτημάτων. Ορισμένες από τις λειτουργίες που υποστηρίζει είναι:

- Παραμένουσες συνδέσεις HTTP (keep-alive) και caching συνδέσεων (connection pooling)
- Διεθνή ονόματα τομέα (domains) και διευθύνσεις url
- Σύνοδοι (sessions) με παραμένοντα (persistent) cookies
- Έλεγχος SSL όπως σε πρόγραμμα περιήγησης
- Αυτόματη αποκωδικοποίηση περιεχομένου
- Basic/Digest ταυτοποίηση
- Cookies κλειδιού/τιμής
- Αυτόματη αποσυμπίεση
- Unicode περιεχόμενο απόκρισης
- Υποστήριξη διακομιστή μεσολάβησης (proxy) HTTP(S)
- Μεταφορτώσεις αρχείων με multipart αιτήματα (multipart requests)
- Λήψη ροής περιεχομένου (streaming)
- Διακοπές σύνδεσης (connection timeouts)
- Διάσπαση αιτημάτων (chunked requests)
- Υποστήριξη .netrc.

Για να γίνει ένα αίτημα HTTP, αρχικά εισάγεται η βιβλιοθήκη με την εντολή `import requests`, στη συνέχεια επιλέγεται η κατάλληλη μέθοδος (GET, POST, PUT, DELETE, HEAD, OPTIONS) και εισάγεται το URL, όπως φαίνεται στον Κώδικα 5. Ως αποτέλεσμα λαμβάνεται ένα αντικείμενο `Response` (στο παράδειγμα ονομάζεται r), από το οποίο μπορούμε να αντλήσουμε τις πληροφορίες που χρειαζόμαστε.

```
import requests
r = requests.get('https://api.github.com/events')
r = requests.post('https://httpbin.org/post', data={'key': 'value'})
```

Εάν πρόκειται να σταλούν δεδομένα, το URL περιλαμβάνει query παραμέτρους οι οποίες προστίθενται μετά από το ερωτηματικό (?) και είναι ζεύγη κλειδιών-τιμών, όπως για παράδειγμα το httpbin.org/get?key=val. Η βιβλιοθήκη requests επιτρέπει την εισαγωγή αυτών των παραμέτρων ως λεξικό συμβολοσειρών με τη βοήθεια του ορίσματος `params`. Για παράδειγμα, εάν θέλουμε να περάσουμε τα ζεύγη key1=value1 και key2=value2 στο URL httpbin.org/get, θα χρησιμοποιήσουμε τον εξής κώδικα:

```
payload = {'key1': 'value1', 'key2': 'value2'}
r = requests.get('https://httpbin.org/get', params=payload)
```

Εάν πρόκειται να σταλούν κωδικοποιημένα δεδομένα, όπως μια φόρμα HTML, το λεξικό περνάει στο όρισμα `data` και κωδικοποιείται αυτόματα με την αποστολή του αιτήματος. Ο κώδικας έχει ως εξής:

```
payload = {'key1': 'value1', 'key2': 'value2'}
r = requests.post("https://httpbin.org/post", data=payload)
```

Το όρισμα `data` μπορεί να έχει πολλαπλές τιμές για κάθε κλειδί. Αυτό μπορεί να υλοποιηθεί χρησιμοποιώντας είτε μια λίστα από πλειάδες είτε ένα λεξικό με λίστες ως τιμές. Είναι ιδιαίτερα χρήσιμο στην περίπτωση που η φόρμα διαθέτει πολλά στοιχεία με το ίδιο κλειδί. Ο κώδικας έχει ως εξής:

```
payload_tuples = [('key1', 'value1'), ('key1', 'value2')]
r1 = requests.post('https://httpbin.org/post', data=payload_tuples)
payload_dict = {'key1': ['value1', 'value2']}
r2 = requests.post('https://httpbin.org/post', data=payload_dict)
```

Η απάντηση του διακομιστή μπορεί να διαβαστεί με την ιδιότητα `r.text`. Η βιβλιοθήκη requests αποκωδικοποιεί αυτόματα το περιεχόμενο που λαμβάνει από τον διακομιστή «μαντεύοντας» το πρότυπο κωδικοποίησης από τις κεφαλίδες HTTP. Με την ιδιότητα `r.encoding` μπορούμε να μάθουμε την κωδικοποίηση που χρησιμοποίησε η requests και να την αλλάξουμε. Στην περίπτωση που το περιεχόμενο της απόκρισης είναι binary και όχι κείμενο, μπορεί να διαβαστεί ως bytes με την ιδιότητα `r.content`. Υπάρχει, επίσης, ένας ενσωματωμένος αποκωδικοποιητής JSON ο οποίος καλείται με τη μέθοδο `r.json()`, όταν η απόκριση είναι αντικείμενο JSON. Για να λάβουμε την αρχική socket απόκριση του διακομιστή, χρησιμοποιούμε την ιδιότητα `r.raw` αφού έχουμε προσθέσει το όρισμα `stream=True` στο αίτημα [15].

3.2.5 Beautiful Soup

Η Beautiful Soup είναι μια βιβλιοθήκη της Python για εξαγωγή δεδομένων από αρχεία HTML και XML. Λειτουργεί με διάφορους αναλυτές (parsers) και παρέχει τρόπους πλοήγησης, αναζήτησης και τροποποίησης του δέντρου DOM. Η Beautiful Soup υποστηρίζει τον HTML parser που περιλαμβάνεται στην standard βιβλιοθήκη της Python (html.parser) αλλά και τρίτους parsers, όπως ο lxml και ο html5lib.

Το πρώτο βήμα για την ανάλυση ενός εγγράφου είναι να περάσουμε το έγγραφο στον constructor BeautifulSoup είτε ως συμβολοσειρά είτε ως αρχείο. Το έγγραφο μετατρέπεται σε unicode και οι οντότητες HTML μετατρέπονται σε unicode χαρακτήρες. Στη συνέχεια η Beautiful Soup αναλύει το έγγραφο χρησιμοποιώντας τον καλύτερο διαθέσιμο parser. Από προεπιλογή η ανάλυση γίνεται με HTML parser, εκτός εάν οριστεί να γίνει με XML parser.

```
>>> from bs4 import BeautifulSoup
>>> with open("index.html") as fp:
            soup = BeautifulSoup(fp, 'html.parser')
>>> soup = BeautifulSoup("<html>a web page</html>", 'html.parser')

>>> print(BeautifulSoup("<html><head></head><body>Sacr&eacute;
bleu!</body></html>", "html.parser"))
<html><head></head><body>Sacré bleu!</body></html>
```

Κώδικας 5. Ανάλυση εγγράφου HTML με τη βιβλιοθήκη Beautiful Soup

Η Beautiful Soup μετατρέπει ένα έγγραφο HTML σε ένα δέντρο αντικειμένων Python. Τα αντικείμενα αυτά είναι τεσσάρων ειδών: Tag, NavigableString, BeautifulSoup και Comment. Ένα αντικείμενο `Tag` αντιστοιχεί σε μια ετικέτα XML ή HTML στο αρχικό έγγραφο. Τα αντικείμενα `Tag` διαθέτουν διάφορα χαρακτηριστικά και μεθόδους, όπως για παράδειγμα το `.name` που επιστρέφει το όνομα της ετικέτας και το `.attrs` που επιστρέφει ένα λεξικό με τα χαρακτηριστικά (attributes) της ετικέτας.

Η κλάση `NavigableString` αντιστοιχεί στο κείμενο (συμβολοσειρά) που περιέχει μια ετικέτα. Ένα αντικείμενο `NavigableString` είναι όπως ακριβώς μια unicode συμβολοσειρά στην Python. Υποστηρίζει τις περισσότερες μεθόδους πλοήγησης και αναζήτησης στο δέντρο αλλά όχι όλες. Αυτό συμβαίνει διότι οι συμβολοσειρές δεν μπορούν να περιέχουν άλλα στοιχεία, όπως συμβαίνει με τις ετικέτες, οι οποίες μπορεί να περιέχουν συμβολοσειρές ή άλλες ετικέτες. Επομένως, οι συμβολοσειρές δεν υποστηρίζουν τα χαρακτηριστικά `.contents` ή `.string` ή τη μέθοδο `find()`.

Το αντικείμενο `BeautifulSoup` αντιστοιχεί στο έγγραφο HTML ή XML σαν σύνολο. Στην πλειονότητα των περιπτώσεων αντιμετωπίζεται ως ένα αντικείμενο `Tag` και υποστηρίζει τις

περισσότερες μεθόδους για πλοήγηση και αναζήτηση στο δέντρο. Το αντικείμενο `Comment` είναι ένας ειδικός τύπος του `NavigableString` για τα σχόλια σε HTML και XML έγγραφα. Παραδείγματα για τα παραπάνω αντικείμενα της Beautiful Soup φαίνονται στον Κώδικα 6.

```
>>> soup = BeautifulSoup('<b id="boldest">Extremely bold</b>',
'html.parser')
>>> tag = soup.b
>>> type(tag)
<class 'bs4.element.Tag'>

>>> tag.name
'b'

>>> tag['id']
'boldest'

>>> tag.attrs
{'id': 'boldest'}

>>> tag.string
'Extremely bold'
>>> type(tag.string)
<class 'bs4.element.NavigableString'>

>>> doc = BeautifulSoup("<document><content/>INSERT FOOTER HERE</document",
"xml")
>>> footer = BeautifulSoup("<footer>Here's the footer</footer>", "xml")
>>> doc.find(text="INSERT FOOTER HERE").replace_with(footer)
'INSERT FOOTER HERE'
>>> print(doc)
<?xml version="1.0" encoding="utf-8"?>
<document><content/><footer>Here's the footer</footer></document>

>>> markup = "<b><!--Hey, buddy. Want to buy a used parser?--></b>"
>>> soup = BeautifulSoup(markup, 'html.parser')
>>> comment = soup.b.string
>>> type(comment)
<class 'bs4.element.Comment'>
```

Κώδικας 6. Αντικείμενα Tag, NavigableString, BeautifulSoup και Comment της Beautiful Soup

Η Beautiful Soup ορίζει, επίσης, τις κλάσεις Stylesheet, Script και TemplateString, για ενσωματωμένους κανόνες CSS (συμβολοσειρές που περιέχονται στην ετικέτα <style>), ενσωματωμένο κώδικα JavaScript (συμβολοσειρές που περιέχονται στην ετικέτα <script>), και πρότυπα HTML (συμβολοσειρές που περιέχονται στην ετικέτα <template>) αντίστοιχα. Οι κλάσεις αυτές λειτουργούν ακριβώς όπως η NavigableString. Σκοπός τους είναι να διευκολύνουν τον εντοπισμό του κύριου σώματος της σελίδας, αγνοώντας τις συμβολοσειρές που αντιπροσωπεύουν κάτι διαφορετικό. Τέλος υπάρχουν οι κλάσεις CData, ProcessingInstruction, Declaration και Doctype οι οποίες σχετίζονται με επιπλέον πληροφορίες που περιέχουν τα έγγραφα XML. Οι κλάσεις αυτές, όπως και η Comment, είναι υποκλάσεις της NavigableString [16].

3.2.6 pyspider

Η pyspider είναι μια βιβλιοθήκη ανίχνευσης ιστού ή αλλιώς ένας web crawler για τη γλώσσα Python. Τα κύρια χαρακτηριστικά της είναι:

- Διαθέτει web περιβάλλον χρήσης (WebUI), το οποίο περιλαμβάνει editor για τη συγγραφή των scripts, πίνακα παρακολούθησης των tasks και πίνακα προβολής των αποτελεσμάτων.
- Υποστηρίζει MySQL, MongoDB, Redis, SQLite, Elasticsearch και PostgreSQL με SQLAlchemy ως backend βάση δεδομένων.
- Υποστηρίζει RabbitMQ, Beanstalk, Redis και Kombu ως ουρά μηνυμάτων (message queue).
- Δίνει τη δυνατότητα για προτεραιοποίηση των tasks, επαναπροσπάθεια (retry) και περιοδική ανίχνευση.
- Διαθέτει κατανεμημένη αρχιτεκτονική, δυνατότητα ανίχνευσης σελίδων JavaScript και είναι συμβατή με Python 2 και 3.

Η εγκατάσταση της βιβλιοθήκης γίνεται με την εντολή `pip install pyspider`. Με εκτέλεση της εντολής pyspider, το WebUI είναι διαθέσιμο για τον χρήστη στη διεύθυνση http://localhost:5000/. Μόλις ο χρήστης δημιουργήσει ένα νέο project, εμφανίζεται στον editor ο Κώδικας 7 προς συμπλήρωση [17].

```
from pyspider.libs.base_handler import *
class Handler(BaseHandler):
    crawl_config = {
    }

    @every(minutes=24 * 60)
    def on_start(self):
        self.crawl('__START_URL__', callback=self.index_page)

    @config(age=10 * 24 * 60 * 60)
    def index_page(self, response):
        for each in response.doc('a[href^="http"]').items():
            self.crawl(each.attr.href, callback=self.detail_page)

    @config(priority=2)
    def detail_page(self, response):
        return {
            "url": response.url,
            "title": response.doc('title').text(),
        }
```

Κώδικας 7. Παράδειγμα χρήσης της βιβλιοθήκης pyspider

3.2.7 Scrapy

Το Scrapy είναι ένα framework για ανίχνευση (crawling) ιστοσελίδων και εξαγωγή δομημένων δεδομένων, το οποίο μπορεί να χρησιμοποιηθεί σε ένα ευρύ φάσμα εφαρμογών, όπως η εξόρυξη δεδομένων (data mining), η επεξεργασία πληροφοριών (information processing) ή η αρχειοθέτηση πληροφοριών (historical archival). Παρόλο που το Scrapy σχεδιάστηκε αρχικά για εξαγωγή δεδομένων ιστού (web scraping), μπορεί να χρησιμοποιηθεί, επίσης, για εξαγωγή δεδομένων με χρήση API (όπως το Amazon Associates Web Services) ή ως πρόγραμμα ανίχνευσης ιστού γενικού σκοπού.

Το Scrapy διαθέτει ένα σύνολο χαρακτηριστικών για εύκολη και αποτελεσματική εξαγωγή δεδομένων ιστού, όπως είναι τα παρακάτω:

- Ενσωματωμένη υποστήριξη για επιλογή και εξαγωγή δεδομένων από πηγές HTML/XML με επιλογείς CSS και εκφράσεις XPath, και με βοηθητικές μεθόδους για εξαγωγή των δεδομένων με κανονικές εκφράσεις (regular expressions).
- Διαδραστική κονσόλα κελύφους (Scrapy shell) για τη δοκιμή των εκφράσεων CSS και XPath για την εξαγωγή δεδομένων, η οποία είναι ιδιαίτερα χρήσιμη κατά τη σύνταξη ή την αποσφαλμάτωση των αραχνών (spiders).
- Ενσωματωμένη υποστήριξη για τη δημιουργία ροών εξαγωγής σε διάφορες μορφές (JSON, CSV, XML) και την αποθήκευση τους σε backend συστήματα (FTP, S3, τοπικό σύστημα αρχείων).
- Υποστήριξη κωδικοποίησης και αυτόματη αναγνώριση κωδικοποίησης για τη διαχείριση ξένων γλωσσών, ανεπίσημων ή ελλιπών δηλώσεων κωδικοποίησης.
- Δυνατότητα επεκτασιμότητας που επιτρέπει στον χρήστη να προσθέσει λειτουργικότητα χρησιμοποιώντας signals και ένα καλά ορισμένο API (middlewares, extensions και pipelines).
- Μεγάλη γκάμα ενσωματωμένων επεκτάσεων (extensions) και ενδιάμεσου λογισμικού (middlewares) που αφορούν:
    - διαχείριση cookies και sessions
    - συμπίεση, ταυτοποίηση (authentication) και caching
    - αλλαγή της κεφαλίδας User-Agent (user-agent spoofing)
    - ανάγνωση του αρχείου robots.txt
    - περιορισμό του βάθους ανίχνευσης κ.α.
- Κονσόλα Telnet, η οποία συνδέεται με την κονσόλα Python που εκτελείται μέσα στη διαδικασία Scrapy, για έλεγχο και αποσφαλμάτωση της αράχνης (spider).
- Μια σειρά επιπλέον εργαλείων, όπως επαναχρησιμοποιήσιμες αράχνες για ανίχνευση ιστοσελίδων από χάρτες ιστοτόπου (sitemaps) και ροές XML/CSV, pipeline πολυμέσων για

αυτόματη λήψη εικόνων (ή άλλων πολυμέσων), έναν caching DNS resolver και πολλά ακόμη [18].

Η εγκατάσταση του Scrapy γίνεται με την εντολή `pip install scrapy`. Η εγκατάσταση συνιστάται να γίνεται σε “εικονικό περιβάλλον” (virtual environment) και όχι globally (δηλαδή σε όλο το σύστημα). Με τη χρήση ενός virtual environment δεν υπάρχει conflict με τα ήδη εγκατεστημένα πακέτα συστήματος της Python (που θα μπορούσαν να σπάσουν ορισμένα από τα εργαλεία και τα script του συστήματος) και ο χρήστης μπορεί να εγκαθιστά νέα πακέτα με pip (χωρίς sudo).

Για να χρησιμοποιήσει κανείς το Scrapy, δημιουργεί αρχικά έναν κατάλογο για την αποθήκευση του κώδικα, τον ανοίγει και ξεκινά ένα νέο Scrapy project με την εντολή `scrapy startproject` και το όνομα του project. Για παράδειγμα, αν θέλουμε να δημιουργήσουμε το project με το όνομα tutorial, γράφουμε `scrapy startproject tutorial.` Με τον τρόπο αυτό δημιουργείται ο κατάλογος tutorial με τα εξής περιεχόμενα:

```
tutorial/
    scrapy.cfg            # deploy configuration file
    tutorial/             # project's Python module, you'll import your code
                          # from here
        __init__.py
        items.py          # project items definition file
        middlewares.py    # project middlewares file
        pipelines.py      # project pipelines file
        settings.py       # project settings file
        spiders/          # a directory where you'll later put your spiders
            __init__.py
```

Οι αράχνες (spiders) είναι κλάσεις που ορίζει ο χρήστης και τις οποίες χρησιμοποιεί το Scrapy για την εξαγωγή δεδομένων από έναν ή περισσότερους ιστότοπους. Οι αράχνες καθορίζουν που πρέπει να σταλούν τα αρχικά request, πως γίνεται το διαδοχικό άνοιγμα των συνδέσμων στις σελίδες (προαιρετικά) και πως γίνεται η ανάλυση της σελίδας που έχει ληφθεί για να εξάγουμε τα δεδομένα. Στον Κώδικα 8 φαίνεται μια αράχνη, ως παράδειγμα. Το αρχείο που περιέχει τον κώδικα της αράχνης αποθηκεύεται στον κατάλογο tutorial/spiders του project.

```
from pathlib import Path

import scrapy

class QuotesSpider(scrapy.Spider):
    name = "quotes"

    def start_requests(self):
        urls = [
            'https://quotes.toscrape.com/page/1/',
            'https://quotes.toscrape.com/page/2/',
        ]
        for url in urls:
```

```
            yield scrapy.Request(url=url, callback=self.parse)

    def parse(self, response):
        page = response.url.split("/")[-2]
        filename = f'quotes-{page}.html'
        Path(filename).write_bytes(response.body)
        self.log(f'Saved file {filename}')
```

Κώδικας 8. Παράδειγμα αράχνης (spider) του Scrapy

Η αράχνη είναι υποκλάση της scrapy.Spider και έχει τα παρακάτω χαρακτηριστικά και μεθόδους:

- `name`: το χαρακτηριστικό αυτό προσδιορίζει την το όνομα της αράχνης και πρέπει να είναι μοναδικό σε κάθε project, δηλαδή δεν μπορούμε να έχουμε αράχνες με το ίδιο όνομα.
- `start_requests()`: η μέθοδος αυτή επιστρέφει ένα iterable του αντικειμένου Requests (μπορεί να επιστρέφει μια λίστα με αιτήματα ή να περιέχει μια συνάρτηση γεννήτορα), από το οποίο θα ξεκινήσει η αράχνη την ανίχνευση. Τα επόμενα αιτήματα θα δημιουργηθούν διαδοχικά από τα αρχικά αιτήματα που ορίζονται εδώ.
- `parse()`: η μέθοδος αυτή χειρίζεται την απάντηση που λαμβάνεται από κάθε αίτημα. Η παράμετρος `response` είναι instance του αντικειμένου TextResponse, το οποίο κρατά το περιεχόμενο της σελίδας και διαθέτει χρήσιμες μεθόδους για τη διαχείρισή του. Η μέθοδος `parse()` αναλύει την απόκριση, εξάγει τα δεδομένα σε λεξικά, εντοπίζει νέες διευθύνσεις URL για ανίχνευση και δημιουργεί νέα requests από αυτά [18].

Για να τρέξουμε την αράχνη, θα πρέπει να μεταβούμε στον root κατάλογο του project και να εκτελέσουμε την εντολή `scrapy crawl quotes`.

3.3 Βιβλιοθήκες Python για ασύγχρονα αιτήματα

3.3.1 GRequests

Η βιβλιοθήκη requests είναι εύχρηστη αλλά αργή σε περιπτώσεις εξαγωγής δεδομένων από πολυάριθμες σελίδες. Αυτό συμβαίνει γιατί τα αιτήματα HTTP αποστέλλονται σύγχρονα. Για παράδειγμα, εάν πρέπει να ανακτηθούν 25 διευθύνσεις URL, η ανάκτηση τους γίνεται μια προς μια. Αν ο χρόνος ανάκτησης μιας σελίδας είναι 10 δευτερόλεπτα, χρειάζονται πάνω από 4 λεπτά για τη λήψη και των 25 σελίδων.

Ένας τρόπος για να επιταχυνθεί η διαδικασία είναι η αποστολή πολλαπλών αιτημάτων ταυτόχρονα. Αυτό σημαίνει ότι τα αιτήματα δεν αποστέλλονται διαδοχικά, αλλά σε ομάδες των 5 αιτημάτων, για παράδειγμα. Σε αυτή την περίπτωση πραγματοποιείται εξαγωγή δεδομένων από 5

URL σε 10 δευτερόλεπτα, αντί για 50 δευτερόλεπτα, και ο συνολικός χρόνος για τα 25 URL είναι 50 δευτερόλεπτα αντί για 250.

Συνήθως, η υλοποίηση γίνεται με την τεχνική του παραλληλισμού με νήματα (thread-based parallelism) και με χρήση της built-in βιβλιοθήκης threading που διαθέτει η Python. Επειδή όμως η χρήση νημάτων είναι περίπλοκη και προϋποθέτει εμπειρία στον προγραμματισμό, μπορεί να χρησιμοποιηθεί εναλλακτικά η βιβλιοθήκη grequests, η οποία απλοποιεί τη διαδικασία. Βασίζεται στη βιβλιοθήκη requests και παράλληλα ενσωματώνει το Gevent, ένα ασύγχρονο API της Python που χρησιμοποιείται ευρέως για web εφαρμογές. Η grequests επιτρέπει την αποστολή ασύγχρονων αιτημάτων HTTP με εύκολο τρόπο.

Ένα παράδειγμα χρήσης της βιβλιοθήκης grequests φαίνεται στον Κώδικα 9. Το module εισάγεται με την εντολή `import grequests` και στη συνέχεια ορίζεται ο πίνακας με τα URL προς ανάκτηση. Έπειτα δημιουργείται ένα σύνολο αιτημάτων HTTP, τα οποία αποστέλλονται ασύγχρονα με τη μέθοδο `map`.

```
import grequests

urls = [
    'http://www.heroku.com',
    'http://python-tablib.org',
    'http://httpbin.org',
    'http://python-requests.org',
    'http://fakedomain/',
    'http://kennethreitz.com'
]
rs = (grequests.get(u) for u in urls)
grequests.map(rs)
# [<Response [200]>, <Response [200]>, <Response [200]>, <Response [200]>,
None, <Response [200]>]
```

Κώδικας 9. Βιβλιοθήκη GRequests: αποστολή ασύγχρονων αιτημάτων

```
def exception_handler(request, exception):
        print("Request failed")

reqs = [
        grequests.get('http://httpbin.org/delay/1', timeout=0.001),
        grequests.get('http://fakedomain/'),
        grequests.get('http://httpbin.org/status/500')
]
grequests.map(reqs, exception_handler=exception_handler)
# Request failed
# Request failed
# [None, None, <Response [500]>]
```

Κώδικας 10. Βιβλιοθήκη GRequests: προσθήκη exception handler

Για τη διαχείριση διακοπών (timeouts) ή εξαιρέσεων κατά τη σύνδεση με τον εξυπηρετητή μπορεί να χρησιμοποιηθεί προαιρετικά ένας χειριστής εξαίρεσης (exception handler), όπως φαίνεται στον Κώδικα 10.

Για βελτίωση της ταχύτητας μπορεί να χρησιμοποιηθεί η μέθοδος `imap` στη θέση της μεθόδου `map`. Η `imap` επιστρέφει έναν γεννήτορα (generator) απαντήσεων. Η σειρά των απαντήσεων δεν ακολουθεί τη σειρά με την οποία υποβλήθηκαν τα αιτήματα HTTP. Το API της `imap` είναι αντίστοιχο με το API της `map`. Οι μέθοδοι `map` και `imap` διαθέτουν την παράμετρο `size` η οποία μπορεί να οριστεί κατάλληλα προκειμένου να αυξηθεί το μέγεθος του gevent pool.

```
for resp in grequests.imap(reqs, size=10):
    print(resp)
```

Τέλος, σημειώνεται ότι επειδή η βιβλιοθήκη grequests χρησιμοποιεί το gevent (το οποίο με τη σειρά του χρησιμοποιεί monkey patching για να επιτευχθεί συγχρονισμός), θα πρέπει να εισάγεται πριν από τις άλλες βιβλιοθήκες, ειδικά πριν από τη requests για να αποφευχθούν πιθανά προβλήματα [19].

```
# GOOD PRACTICE
import grequests
import requests

# BAD PRACTICE
import requests
import grequests
```

3.3.2 aiohttp

Η βιβλιοθήκη aiohttp της Python είναι ένας ασύγχρονος HTTP Client/Server. Για τις ανάγκες του web scraping μας ενδιαφέρει μόνο η λειτουργία του client, η οποία περιγράφεται παρακάτω.

Υποβολή ασύγχρονου αιτήματος

Για να υποβληθεί ένα αίτημα HTTP εισάγονται τα modules aiohttp και asyncio και ορίζεται η συνάρτηση `main()`, όπως φαίνεται στον Κώδικα 11. Οι συναρτήσεις με τη σύνταξη `async def` ονομάζονται coroutine συναρτήσεις. Έπειτα, δημιουργείται ένα `ClientSession` αντικείμενο με το όνομα `session`, το οποίο μπορεί να διατηρεί συνδέσεις με έως και 100 διακομιστές ταυτόχρονα, και ένα `ClientResponse` αντικείμενο με το όνομα `resp`. Για να τρέξει η συνάρτηση `main()`, χρησιμοποιείται η `asyncio.run()`. Το αίτημα αποστέλλεται στον διακομιστή και αναμένεται η απόκριση.

Η λέξη-κλειδί `async` δηλώνει στον διερμηνέα της Python ότι η coroutine συνάρτηση θα πρέπει να εκτελεστεί ασύγχρονα σε ένα βρόχο συμβάντων (event loop). Η λέξη-κλειδί `await` επιστρέφει τον έλεγχο στο βρόχο συμβάντων αναστέλλοντας την εκτέλεση της συνάρτησης και αφήνοντας το βρόχο συμβάντων να τρέξει άλλες διεργασίες μέχρι να ληφθεί το αποτέλεσμα που αναμένεται.

```
import aiohttp
import asyncio

async def main():
    async with aiohttp.ClientSession() as session:
        async with session.get('http://httpbin.org/get') as resp:
            print(resp.status)
            print(await resp.text())

asyncio.run(main)
```

Κώδικας 11. Βιβλιοθήκη aiohttp: αποστολή ασύγχρονου αιτήματος

Η εισαγωγή query παραμέτρων στο URL γίνεται με τη βοήθεια του ορίσματος params. Ένα παράδειγμα φαίνεται στον κώδικα που ακολουθεί, στον οποίο εισάγονται τα ζεύγη key1=value1 και key2=value2 στο URL httpbin.org/get [20].

```
params = {'key1': 'value1', 'key2': 'value2'}
async with session.get('http://httpbin.org/get', params=params) as resp:
    expect = 'http://httpbin.org/get?key1=value1&key2=value2'
    assert str(resp.url) == expect
```

Σύγκριση με τη βιβλιοθήκη requests

Με τη βιβλιοθήκη aiohttp ένα απλό αίτημα HTTP εκτελείται σε τρία βήματα, όπως φαίνεται στον παρακάτω κώδικα:

```
async with aiohttp.ClientSession() as session:
    async with session.get('http://python.org') as response:
        print(await response.text())
```

Ο αντίστοιχος κώδικας με χρήση της βιβλιοθήκης requests έχει ως εξής:

```
response = requests.get('http://python.org')
print(response.text)
```

Η περιπλοκότητα του module aiohttp οφείλεται στο γεγονός ότι είναι ασύγχρονο και το API του είναι σχεδιασμένο για να αξιοποιεί τις non-blocking λειτουργίες δικτύου. Στο προηγούμενο παράδειγμα όταν χρησιμοποιείται η βιβλιοθήκη requests ο κώδικας μπλοκάρεται τρεις φορές, ενώ αντίθετα η aiohttp δίνει στο βρόχο συμβάντων (event loop) τρεις φορές τη δυνατότητα για αλλαγή ροής, όπως θα εξηγήσουμε παρακάτω [20]:

- Με τη μέθοδο `get()` και οι δύο βιβλιοθήκες στέλνουν ένα αίτημα GET στον διακομιστή. Για την aiohttp αυτό σημαίνει ασύγχρονη είσοδος/έξοδος (I/O) και επισημαίνεται με την έκφραση `async with`.
- Όσον αφορά τη βιβλιοθήκη requests, η εντολή `response.text` επιστρέφει απλά ένα χαρακτηριστικό. Με την κλήση της `get()` που προηγήθηκε, έχει ήδη φορτωθεί και αποκωδικοποιηθεί το payload της απόκρισης. Αντίθετα, η aiohttp φορτώνει μόνο τις κεφαλίδες όταν εκτελείται η `get()`, επιτρέποντας τη φόρτωση του body σε μεταγενέστερο χρόνο, με μια δεύτερη ασύγχρονη λειτουργία. Για αυτό το λόγο η εντολή είναι `await response.text()`.
- Η εντολή `async with aiohttp.ClientSession()` δεν εκτελεί I/O κατά την έναρξη του μπλοκ, αλλά στο τέλος του μπλοκ και διασφαλίζει ότι όλοι οι υπόλοιποι πόροι έχουν κλείσει σωστά. Και πάλι αυτό γίνεται ασύγχρονα και επισημαίνεται με `async with`. Η χρήση του session είναι, επίσης ένα εργαλείο για καλύτερη απόδοση. Ένα session μπορεί να διαχειρίζεται ένα σύνολο συνδέσεων και επιτρέπει την επαναχρησιμοποίηση τους, αντί να πραγματοποιείται άνοιγμα και κλείσιμο μιας νέας σύνδεσης για κάθε αίτημα.

Εδώ θα πρέπει να διευκρινιστεί ότι και η βιβλιοθήκη requests υποστηρίζει συνόδους (sessions). Ένα παράδειγμα φαίνεται στο κώδικα που ακολουθεί. Η χρήση του session έχει θετική επίδραση στην απόδοση, αλλά δε συνηθίζεται, καθώς η βιβλιοθήκη requests εστιάζει περισσότερο στην απλότητα παρά στην απόδοση. Αντίθετα, η βιβλιοθήκη aiohttp για ασύγχρονο προγραμματισμό παρουσιάζει μεγαλύτερη πολυπλοκότητα με αντάλλαγμα καλύτερη απόδοση [20].

```
with requests.Session() as session:
    response = session.get('http://python.org')
    print(response.text)
```

Υποβολή πολλαπλών ασύγχρονων αιτημάτων

Ο χρόνος εκτέλεσης βελτιώνεται ακόμα και με την πραγματοποίηση ενός μόνο ασύγχρονου αιτήματος, καθώς ο βρόχος συμβάντων μπορεί να εκτελεί άλλες εργασίες και δεν μπλοκάρεται ολόκληρο το thread όσο περιμένουμε την απόκριση του διακομιστή. Αλλά η χρήση ενός ασύγχρονου client έχει πραγματικά νόημα όταν υποβάλλεται ένας μεγάλος αριθμός αιτημάτων. Ένα τέτοιο παράδειγμα φαίνεται στον Κώδικα 12.

```
import aiohttp
import asyncio

async def get_page(session, url):
        async with session.get(url) as r:
                return await r.text()

async def get_all(session, urls):
        tasks = []
```

```
        for url in urls:
                task = asyncio.create_task(get_page(session, url))
                tasks.append(task)
        results = await asyncio.gather(*tasks)
        return results

async def main(urls):
        async with aiohttp.ClientSession() as session:
                data = await get_all(session, urls)
                return data

if __name__ == '__main__':
        urls = [
                'https://books.toscrape.com/catalogue/page-1.html',
                'https://books.toscrape.com/catalogue/page-2.html',
                'https://books.toscrape.com/catalogue/page-3.html'
        ]
        results = asyncio.run(main(urls))
```

Κώδικας 12. Βιβλιοθήκη aiohttp: αποστολή πολλαπλών ασύγχρονων αιτημάτων

Στον Κώδικα 12 ορίζονται τρεις coroutine συναρτήσεις. Η συνάρτηση `get_page()` στέλνει το αίτημα στον εξυπηρετητή και επιστρέφει τον κώδικα HTML της σελίδας. Η συνάρτηση `get_all()` δημιουργεί μια λίστα από tasks, τα οποία εκτελούνται ταυτόχρονα με κλήση της `gather`. Τέλος, ορίζεται η συνάρτηση `main()`, η οποία καλεί την `get_all()` και ελέγχει το session [20].

Να σημειώσουμε ότι ο ιστότοπος https://toscrape.com/ που χρησιμοποιήθηκε παραπάνω είναι ένα δημοφιλές sandbox για web scraping, δηλαδή ένας ιστότοπος που χρησιμοποιείται για τη δοκιμή εργαλείων web scraping. Ο ιστότοπος αποτελείται από δύο μέρη. Το πρώτο (books.toscrape.com) είναι ένα εικονικό βιβλιοπωλείο με χιλιάδες βιβλία, που περιέχει μόνο στατικό περιεχόμενο και προσφέρεται για την εξαγωγή δεδομένων με χρήση απλών βιβλιοθηκών. Το δεύτερο μέρος (quotes.toscrape.com), στο οποίο παρατίθενται λόγια διάσημων προσωπικοτήτων, απαιτεί πιο σύνθετες τεχνικές web scraping, καθώς το περιεχόμενο φορτώνεται με JavaScript, η εμφάνιση περισσότερων αποτελεσμάτων γίνεται με κύλιση (infinite scrolling) αντί για σελιδοποίηση (pagination), διαθέτει φόρμα login και άλλα χαρακτηριστικά που συναντάμε σε σύγχρονους ιστότοπους.

3.3.3 HTTPX

Η βιβλιοθήκη HTTPX είναι ένας HTTP client για την Python 3 που παρέχει σύγχρονα και ασύγχρονα APIs και υποστήριξη για HTTP/1.1 και HTTP/2. Βασίζεται στη χρηστικότητα της γνωστής βιβλιοθήκης requests και διαθέτει εκτός από τα χαρακτηριστικά της requests, τα εξής [21]:

- Ένα API αντίστοιχο με εκείνο της βιβλιοθήκης requests
- Πρότυπη (standard) σύγχρονη διεπαφή αλλά με ασύγχρονη υποστήριξη εάν απαιτείται

- Υποστήριξη για HTTP/1.1 και HTTP/2
- Δυνατότητα υποβολής αιτημάτων απευθείας σε WSGI ή ASGI εφαρμογές
- Υποστήριξη timeouts
- Δυνατότητα χρήσης δηλώσεων τύπων (fully type annotated)
- 100% κάλυψη testing (test coverage)

Υποβολή σύγχρονου αιτήματος HTTP

Η εισαγωγή της βιβλιοθήκης HTTPX γίνεται με την εντολή `import httpx`. Για την υποβολή ενός απλού GET αιτήματος χρησιμοποιείται η μέθοδος `get()`, όπως φαίνεται παρακάτω:

```
>>> import httpx
>>> r = httpx.get('https://httpbin.org/get')
>>> r
<Response [200 OK]>
```

Για την αποστολή αιτημάτων με τις υπόλοιπες μεθόδους (POST, PUT, DELETE, HEAD, OPTIONS) χρησιμοποιείται η αντίστοιχη συνάρτηση και τα δεδομένα αποστέλλονται μέσω της παραμέτρου data με τη μορφή λεξικού.

```
>>> r = httpx.post('https://httpbin.org/post', data={'key': 'value'})
>>> r = httpx.put('https://httpbin.org/put', data={'key': 'value'})
>>> r = httpx.delete('https://httpbin.org/delete')
>>> r = httpx.head('https://httpbin.org/get')
>>> r = httpx.options('https://httpbin.org/get')
```

Για να συμπεριληφθούν query παράμετροι στο URL χρησιμοποιείται το όρισμα params, όπως φαίνεται στο επόμενο παράδειγμα. Στο όρισμα params εισάγεται ένα λεξικό, το οποίο μπορεί να παίρνει ως τιμή μια συμβολοσειρά ή μια λίστα.

```
>>> params = {'key1': 'value1', 'key2': ['value2', 'value3']}
>>> r = httpx.get('https://httpbin.org/get', params=params)
>>> r.url
URL('https://httpbin.org/get?key1=value1&key2=value2&key2=value3')
```

Η HTTPX αποκωδικοποιεί αυτόματα το περιεχόμενο της απόκρισης σε κείμενο unicode. Με την ιδιότητα `encoding` μπορούμε να δούμε την κωδικοποίηση που χρησιμοποίησε η HTTPX ή να ορίσουμε διαφορετική κωδικοποίηση. Εάν το περιεχόμενο της απόκρισης είναι binary και όχι κείμενο, μπορεί να διαβαστεί ως bytes με την ιδιότητα `content`.

```
>>> r = httpx.get('https://www.example.org/')
>>> r.text
'<!doctype html>\n<html>\n<head>\n<title>Example Domain</title>...'

>>> r.encoding
```

```
'UTF-8'
>>> r.encoding = 'ISO-8859-1'

>>> r.content
b'<!doctype html>\n<html>\n<head>\n<title>Example Domain</title>...'
```

Υποβολή ασύγχρονου αιτήματος HTTP

Η HTTPX υποστηρίζει είτε τη βιβλιοθήκη asyncio είτε τη βιβλιοθήκη trio ως ασύγχρονο περιβάλλον.

Η asyncio είναι μια built-in βιβλιοθήκη της Python που χρησιμοποιείται για τη συγγραφή μονονηματικού ταυτόχρονου (concurrent) κώδικα με τη χρήση coroutines, για πολυπλεξία εισόδου-εξόδου μέσω υποδοχών (sockets) και άλλων πόρων, για την εκτέλεση προγραμμάτων πελατών και διακομιστών δικτύου κλπ. Για την αποστολή ενός ασύγχρονου αιτήματος με τις βιβλιοθήκες asyncio και httpx, γίνονται import τα δύο modules και δημιουργείται ένα αντικείμενο AsyncClient, όπως φαίνεται στον Κώδικα 13. Για την εκτέλεση της `main()` χρησιμοποιείται η μέθοδος `run()` της βιβλιοθήκης asyncio.

```
import asyncio
import httpx

async def main():
    async with httpx.AsyncClient() as client:
        response = await client.get('https://www.example.com/')
        print(response)

asyncio.run(main())
```

Κώδικας 13. Αποστολή ασύγχρονου αιτήματος με τις βιβλιοθήκες httpx και asyncio

Η βιβλιοθήκη trio βοηθά στη συγγραφή προγραμμάτων με λειτουργίες που εκτελούνται ταυτόχρονα με παράλληλη είσοδο-έξοδο, όπως για παράδειγμα μια αράχνη ιστού που ανακτά πολλές σελίδες παράλληλα, ένας εξυπηρετητής που διαχειρίζεται ταυτόχρονα πολλές λήψεις και συνδέσεις websocket, ένας επόπτης διεργασιών που παρακολουθεί πολλαπλές υποδιεργασίες κλπ. Για την αποστολή ενός ασύγχρονου request χρησιμοποιώντας τις βιβλιοθήκες trio και httpx, εισάγονται τα δύο modules και ακολουθείται παρόμοια διαδικασία, όπως φαίνεται στον Κώδικα 14.

```
import httpx
import trio

async def main():
    async with httpx.AsyncClient() as client:
        response = await client.get('https://www.example.com/')
        print(response)

trio.run(main)
```

Κώδικας 14. Αποστολή ασύγχρονου αιτήματος με τις βιβλιοθήκες httpx και trio

Υποβολή πολλαπλών ασύγχρονων αιτημάτων

Στον Κώδικα 15 φαίνεται ένα παράδειγμα αποστολής τριών ασύγχρονων αιτημάτων με χρήση της βιβλιοθήκης HTTPX.

```
import httpx
import asyncio

async def get_page(client, url):
        r = await client.get(url)
        return r.text()

async def get_all(client, urls):
        tasks = []
        for url in urls:
                task = asyncio.create_task(get_page(client, url))
                tasks.append(task)
        results = await asyncio.gather(*tasks)
        return results

async def main(urls):
        async with httpx.AsyncClient() as client:
                data = await get_all(client, urls)
                return data

if __name__ == '__main__':
        urls = [
                'https://books.toscrape.com/catalogue/page-1.html',
                'https://books.toscrape.com/catalogue/page-2.html',
                'https://books.toscrape.com/catalogue/page-3.html'
        ]
        results = asyncio.run(main(urls))
```

Κώδικας 15. Βιβλιοθήκη httpx: αποστολή πολλαπλών ασύγχρονων αιτημάτων

3.4 Βιβλιοθήκες Python για αυτοματοποίηση προγράμματος περιήγησης

Η εξαγωγή δεδομένων ιστού με χρήση προγράμματος περιήγησης (browser-based web scraping) αποτελεί την πιο εύκολη και γρήγορη λύση για ιστοσελίδες JavaScipt που εκτελούνται στην πλευρά του client. Υπάρχουν πολλά διαθέσιμα frameworks για τη δημιουργία ενός web scraper που χρησιμοποιεί πρόγραμμα περιήγησης. Τα πιο δημοφιλή από αυτά είναι τα Selenium, Playwright και Puppeteer.

3.4.1 Selenium

Το Selenium περιλαμβάνει μια σειρά από διαφορετικά έργα ανοιχτού κώδικα για την αυτοματοποίηση του προγράμματος περιήγησης. Υποστηρίζει bindings για όλες τις δημοφιλείς γλώσσες προγραμματισμού, μεταξύ αυτών και η Python. Το Selenium API χρησιμοποιεί το πρωτόκολλο WebDriver για τον έλεγχο προγραμμάτων περιήγησης όπως Chrome, Firefox ή Safari. Το Selenium μπορεί να ελέγξει είτε ένα τοπικά εγκατεστημένο instance ενός προγράμματος περιήγησης είτε ένα instance που εκτελείται σε απομακρυσμένο μηχάνημα μέσω του δικτύου.

Αρχικά το Selenium προοριζόταν για δοκιμές μεταξύ προγραμμάτων περιήγησης (cross-browser, end-to-end testing). Ωστόσο, σήμερα χρησιμοποιείται ως μια γενική πλατφόρμα αυτοματισμού του προγράμματος περιήγησης εξυπηρετώντας επίσης τους σκοπούς της ανίχνευσης (crawling) και της εξαγωγής (scraping) δεδομένων ιστού.

Η εγκατάσταση του Selenium στην Python γίνεται με την εντολή `pip install selenium`. Το Selenium χρειάζεται έναν webdriver για διασύνδεση με το επιλεγμένο πρόγραμμα περιήγησης. Ο χρήστης θα πρέπει να κατεβάσει τον webdriver για τον αντίστοιχο browser που θέλει να χρησιμοποιήσει. Ο Firefox, για παράδειγμα, χρειάζεται τον geckodriver.

Το Selenium διαθέτει τη μέθοδο `find_element` για τον εντοπισμό ενός στοιχείου σε μια ιστοσελίδα. Για την αναζήτηση περισσότερων στοιχείων χρησιμοποιείται η μέθοδος `find_elements`, η οποία επιστρέφει μια λίστα. Παρακάτω φαίνεται ένα παράδειγμα χρήσης αυτών των μεθόδων:

```
from selenium.webdriver.common.by import By
driver.find_element(By.XPATH, '//button[text()="Some text"]')
driver.find_elements(By.XPATH, '//button')
```

Τα χαρακτηριστικά (attributes) της κλάσης By χρησιμοποιούνται για την αναζήτηση στοιχείων σε μια ιστοσελίδα. Η αναζήτηση μπορεί να γίνει με βάση την ιδιότητα id ή class του στοιχείου, με βάση μια έκφραση XPath, έναν css selector κλπ. Στον Πίνακα 7 φαίνονται όλα τα διαθέσιμα attributes για την κλάση By.

Πίνακας 7. Selenium: attributes της κλάσης By για την επιλογή στοιχείων HTML

Τύπος	**Περιγραφή**	**Παράδειγμα DOM**	**Παράδειγμα**
By.ID	Αναζήτηση στοιχείων με βάση την ιδιότητα id		find_element(By.ID, "myID")
By.NAME	Αναζήτηση στοιχείων με βάση την ιδιότητα name	<input name="myNAME">	find_element(By.NAME, "myNAME")
By.XPATH	Αναζήτηση στοιχείων με βάση μια έκφραση XPath	My <a>Link	find_element(By.XPATH, "//span/a")

By.LINK_TEXT	Αναζήτηση στοιχείων <a> με βάση το κείμενο που περιέχουν	<a>My Link	find_element(By.LINK_TEXT, "My Link")
By.PARTIAL_LINK_TEXT	Αναζήτηση στοιχείων <a> με βάση μια υποσυμβολοσειρά (substring) του κειμένου	<a>My Link	find_element(By.PARTIAL_LINK_TEXT, "Link")
By.TAG_NAME	Αναζήτηση στοιχείων με βάση το όνομα της ετικέτας τους	<h1>	find_element(By.TAG_NAME, "h1")
By.CLASS_NAME	Αναζήτηση στοιχείων με βάση την ιδιότητα class		find_element(By.CLASSNAME, "myCLASS")
By.CSS_SELECTOR	Αναζήτηση στοιχείων με βάση έναν επιλογέα css	My <a>Link	find_element(By.CSS_SELECTOR, "span > a")

Στον Κώδικα 16 φαίνεται ένα παράδειγμα χρήσης του Selenium, το οποίο και θα εξηγήσουμε [22]:

- Αρχικά εισάγεται το module selenium.webdriver και στη συνέχεια οι κλάσεις `Keys` και `By`. Το module selenium.webdriver παρέχει όλες τις υλοποιήσεις του WebDriver. Αυτή τη στιγμή υποστηρίζονται οι Firefox, Chrome, IE και Remote. Η κλάση `Keys` χρησιμοποιείται για την εισαγωγή εντολών από το πληκτρολόγιο, όπως RETURN, F1, ALT κλπ. Η κλάση `By` χρησιμοποιείται για τον εντοπισμό στοιχείων μέσα σε ένα έγγραφο HTML.
- Στη συνέχεια δημιουργείται το instance του Firefox WebDriver με την εντολή `webdriver.Firefox()`.
- Η μέθοδος `driver.get` χρησιμοποιείται για την μετάβαση στην ιστοσελίδα με τη δεδομένη διεύθυνση URL. Ο WebDriver περιμένει μέχρι να φορτωθεί πλήρως η σελίδα (δηλαδή, να ενεργοποιηθεί το event "onload") και μετά επιστρέφει τον έλεγχο στο test ή στο script.
- Η επόμενη γραμμή του κώδικα είναι ένα assertion που επιβεβαιώνει ότι το στοιχείο title περιέχει την λέξη “Python”.
- Στη συνέχεια γίνεται αναζήτηση με τη μέθοδο `find_element`. Εδώ ο WebDriver αναζητά στοιχεία που έχουν την ιδιότητα name με τιμή q.
- Έπειτα εισάγουμε δεδομένα από το πληκτρολόγιο με τη μέθοδο `send_keys`. Προηγουμένως διαγράφουμε οποιοδήποτε προσυμπληρωμένο κείμενο στο πεδίο search (με τη μέθοδο clear), ώστε να μην επηρεάσει τα αποτελέσματα της αναζήτησης.
- Μετά την υποβολή του όρου αναζήτησης λαμβάνουμε τα αποτελέσματα εάν υπάρχουν. Για να βεβαιωθούμε ότι επέστρεψαν ορισμένα αποτελέσματα, γράφουμε το assertion που φαίνεται στον κώδικα.
- Στο τέλος κλείνουμε το παράθυρο του browser με τη μέθοδο `close`. Εναλλακτικά μπορεί να χρησιμοποιηθεί η μέθοδος `quit`. Η `quit` κλείνει τον browser, ενώ η `close` κλείνει μόνο μια καρτέλα. Αν υπάρχει μόνο μια καρτέλα ανοιχτή, οι περισσότεροι browsers θα κλείσουν από προεπιλογή και με τη μέθοδο `close`.

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By

driver = webdriver.Firefox()
driver.get("http://www.python.org")
assert "Python" in driver.title
elem = driver.find_element(By.NAME, "q")
elem.clear()
elem.send_keys("pycon")
elem.send_keys(Keys.RETURN)
assert "No results found." not in driver.page_source
driver.close()
```

Κώδικας 16. Παράδειγμα χρήσης του πακέτου Selenium

3.4.2 Playwright

Το Playwright είναι framework για αυτοματισμό του προγράμματος περιήγησης και διαθέτει API για JavaScript, Python, .NET και Java. Η απλότητα του και οι μεγάλες δυνατότητες αυτοματισμού το καθιστούν ιδανικό εργαλείο για web scraping. Επίσης, παρέχει υποστήριξη για headless browser.

Ένας browser χαρακτηρίζεται headless όταν δεν περιλαμβάνει διεπαφή χρήστη. Τα πλεονεκτήματα ενός headless browser είναι ότι απαιτεί λιγότερους πόρους και ότι μπορεί να τρέξει εύκολα σε διακομιστή (server). Δεδομένου ότι υπάρχει μικρότερη απαίτηση για πόρους, μπορούν να δημιουργηθούν πολλά instances ταυτόχρονα και να γίνει εξαγωγή δεδομένων από πολλές ιστοσελίδες. Ο κύριος λόγος που χρησιμοποιούνται headless browsers για web scraping είναι γιατί όλο και περισσότεροι ιστότοποι κατασκευάζονται με Single Page Application (SPA) frameworks, όπως ReactJS, VueJS, AngularJS κλπ. Αν προσπαθήσουμε να εξάγουμε πληροφορίες από τέτοιου είδους ιστότοπους με έναν συνηθισμένο HTTP client, θα λάβουμε μια κενή σελίδα HTML, καθώς οι ιστότοποι αυτοί είναι χτισμένοι με κώδικα JavaScript. Οι headless browsers αντιμετωπίζουν αυτό το πρόβλημα εκτελώντας τον κώδικα JavaScript όπως ένας κοινός browser.

Τα κυριότερα χαρακτηριστικά της βιβλιοθήκης Playwright είναι τα εξής [23]:

- Cross-browser. Το Playwright υποστηρίζει όλους τους σύγχρονους browsers, συμπεριλαμβανομένων των Google Chrome, Microsoft Edge (με Chromium), Apple Safari (με WebKit) και Mozilla Firefox. Επίσης, δίνει στον χρήστη τη δυνατότητα επιλογής προσαρμοσμένων web drivers με το όρισμα executable_path. Επιτρέπει την ταυτόχρονη εξαγωγή δεδομένων από πολλαπλούς browsers και μπορεί να παρακάμπτει τα προγράμματα ανίχνευσης bot, χρησιμοποιώντας διαφορετικούς browsers και λειτουργικά συστήματα. Επιπλέον, το Playwright βοηθά τον χρήστη να επιλέξει τον καλύτερο browser με βάση την ταχύτητα.

- Cross-platform. Με το Playwright ο χρήστης μπορεί να ελέγξει την απόδοση των εφαρμογών του σε διαφορετικές εκδόσεις των προγραμμάτων περιήγησης για Windows, Linux και macOS.
- Cross-language. Το Playwright υποστηρίζει πολλές γλώσσες προγραμματισμού (TypeScript, JavaScript, Python, .NET, Java), διαθέτει τεκμηρίωση και ενεργή κοινότητα υποστήριξης.
- Auto-wait (αυτόματη αναμονή). Το Playwright πραγματοποιεί μια σειρά ελέγχων πριν εκτελέσει μια ενέργεια, για να διασφαλίσει ότι λειτουργεί όπως αναμένεται. Για να εκτελεστεί η απαιτούμενη ενέργεια, το Playwright περιμένει να περάσουν όλοι οι σχετικοί έλεγχοι. Εάν οι έλεγχοι δεν ολοκληρωθούν σε ένα καθορισμένο χρονικό διάστημα, η ενέργεια αποτυγχάνει με σφάλμα TimeoutError.
- Web-first assertion. Το Playwright δημιουργεί ισχυρισμούς (assertions) για τους δυναμικούς ιστότοπους. Ελέγχει αν η συνθήκη ισχύει, και αν δεν ισχύει, εξετάζει ξανά τον κόμβο μέχρι να ικανοποιηθεί η συνθήκη ή να λήξει το χρονικό όριο. Το χρονικό όριο για τα assertions δεν ορίζεται από προεπιλογή, επομένως το Playwright αναμένει μέχρι να λήξει ο χρόνος για τη συνολική δοκιμή.
- Proxies. Το Playwright υποστηρίζει τη χρήση διακομιστών διαμεσολάβησης (proxies). Ο proxy μπορεί να ρυθμιστεί είτε καθολικά για τον browser είτε για κάθε περιβάλλον (context) του browser ξεχωριστά.
- Browser contexts. Με το Playwright μπορούμε να δημιουργήσουμε μεμονωμένα περιβάλλοντα (contexts) για κάθε δοκιμή σε ένα μόνο instance του browser. Το context του προγράμματος περιήγησης ισοδυναμεί με ένα καινούριο προφίλ browser. Αυτό είναι χρήσιμο όταν εκτελούμε λειτουργίες πολλών χρηστών και web scraping με πλήρη απομόνωση (isolation). Στην περίπτωση αυτή επιτυγχάνεται πλήρης απομόνωση για τις δοκιμές με μηδενικό overhead. Για κάθε context μπορούμε να ρυθμίσουμε cookies, user agent, viewport, proxy και να ενεργοποιήσουμε ή να απενεργοποιήσουμε την JavaScript.

Η εγκατάσταση της βιβλιοθήκης Playwright στη γλώσσα Python γίνεται ως εξής:

```
pip install playwright
playwright install
```

Στον Κώδικα 17 φαίνεται ένα παράδειγμα web scraping στον ιστότοπο quotes.toscrape.com με χρήση του Playwright. Ο κώδικας ανοίγει τον ιστότοπο, επιλέγει τα αντικείμενα με την κλάση 'quote' και με έναν βρόχο εξάγει το απόφθεγμα (text) και το όνομα του συγγραφέα (author) για κάθε αντικείμενο, και τα αποθηκεύει σε ένα λεξικό. Τέλος, τυπώνει το λεξικό στο terminal, περιμένει για 10000 ms και κλείνει τον browser.

```
from playwright.sync_api import sync_playwright

def main():
        with sync_playwright() as p:
                browser = p.chromium.launch(headless=False)
                page = browser.new_page()
                page.goto('https://quotes.toscrape.com/')
                all_quotes = page.query_selector_all('.quote')

                for quote in all_quotes:
                        text = quote.query_selector('.text').inner_text()
                        author = quote.query_selector('.author').inner_text()
                        print({'Author': author, 'Quote': text})

                page.wait_for_timeout(10000)
                browser.close()

if __name__ == '__main__':
        main()
```

Κώδικας 17. Παράδειγμα χρήσης του πακέτου Playwright

3.4.3 Pyppeteer

Το Pyppeteer είναι wrapper της Python για τη βιβλιοθήκη JavaScript (Node), Puppeteer. Λειτουργεί όπως το Selenium, υποστηρίζοντας headless και non-headless λειτουργία, αν και το Pyppeteer περιορίζεται σε προγράμματα περιήγησης chrome/chromium. Η λειτουργία headless αναφέρεται στην εκτέλεση του browser χωρίς τη γραφική διεπαφή χρήστη και προτιμάται σε εργασίες όπως η αυτοματοποίηση ιστού (web automation), οι αυτοματοποιημένοι έλεγχοι (automated testing) και το web scraping, καθώς με τον τρόπο αυτό μειώνεται ο χρόνος φόρτωσης του browser και η απαιτούμενη υπολογιστική ισχύς [24].

Εργαλεία όπως οι βιβλιοθήκες requests και BeautifulSoup είναι ιδανικά για την εξαγωγή δεδομένων από στατικές ιστοσελίδες, αλλά δεν είναι κατάλληλα για δυναμικούς ιστότοπους που περιέχουν JavaScript στο UI και έχουν κατασκευαστεί με frameworks όπως ReactJS, AngularJS, ή VueJS. Το Pyppeteer δίνει στο χρήστη τη δυνατότητα να ελέγχει ολόκληρο το πρόγραμμα περιήγησης και τα στοιχεία του αντί να χρησιμοποιεί βιβλιοθήκες HTTP, όπως η requests, για να λάβει το περιεχόμενο της σελίδας. Με τον τρόπο αυτό ο χρήστης έχει μεγαλύτερη ευελιξία στο τι μπορεί να επιτύχει. Ορισμένες από τις περιπτώσεις χρήσης του Pyppeteer είναι [24]:

- Δημιουργία στιγμιότυπων οθόνης (screenshots) ή pdf από ιστοσελίδες.
- Αυτοματοποιημένη εισαγωγή δεδομένων από το πληκτρολόγιο, υποβολή φόρμας, UI testing κ.α.
- Ανίχνευση (crawling) single-page εφαρμογής για την παραγωγή προ-φορτωμένου περιεχομένου (π.χ. server-side rendering).

- Δημιουργία περιβάλλοντος για αυτοματοποιημένο testing χρησιμοποιώντας τις τελευταίες εκδόσεις του Chrome και της JavaScript.

Το Pyppeteer καταναλώνει λιγότερο χρόνο CPU και λιγότερη RAM σε σύγκριση με το Selenium, αλλά δεν υποστηρίζει άλλους browsers (εκτός από Chromium) και δεν διαθέτει πλήρη τεκμηρίωση, καθώς αποτελεί ανεπίσημο κέλυφος (shell) του Puppeteer. Επομένως, μπορεί να χρησιμοποιηθεί στη θέση του Selenium σε περιπτώσεις που χρειάζεται να γράψει κανείς έναν ισχυρό scraper με μικρή κατανάλωση πόρων.

Η χρήση της βιβλιοθήκης Pyppeteer προϋποθέτει python 3.6+ και η εγκατάσταση μπορεί να γίνει είτε από το repository του PyPI με την εντολή:

```
python3 -m pip install pyppeteer
```

είτε από το repository του GitHub με την εντολή:

```
python3 -m pip install -U git+https://github.com/miyakogi/pyppeteer.git@dev
```

Όταν ο χρήστης τρέχει τη βιβλιοθήκη για πρώτη φορά, κατεβαίνει μια πρόσφατη έκδοση του Chromium (~100MB). Διαφορετικά μπορεί να χρησιμοποιηθεί πρώτα η εντολή `pyppeteer-install` για τη λήψη του browser, και έπειτα ο χρήστης να τρέξει τα scripts που χρησιμοποιούν τη βιβλιοθήκη. Στον Κώδικα 18 φαίνεται ένα απλό παράδειγμα χρήσης του Pyppeteer για τη λήψη screenshot από μια ιστοσελίδα [25].

```
import asyncio

from pyppeteer import launch
async def main():
    browser = await launch()
    page = await browser.newPage()
    await page.goto('http://example.com')
    await page.screenshot({'path': 'example.png'})
    await browser.close()

asyncio.get_event_loop().run_until_complete(main())
```

Κώδικας 18. Παράδειγμα χρήσης της βιβλιοθήκης Pyppeteer

4. ΕΤΟΙΜΑ ΕΡΓΑΛΕΙΑ WEB SCRAPING

Στο κεφάλαιο αυτό γίνεται αναφορά σε έτοιμα εργαλεία web scraping που μπορεί να χρησιμοποιήσει κανείς για να εξάγει δεδομένα από το διαδίκτυο. Παρουσιάζονται τρεις κατηγορίες εργαλείων: SaaS εργαλεία, desktop εφαρμογές και επεκτάσεις προγραμμάτων περιήγησης.

4.1 SaaS εργαλεία

Οι SaaS (Software as a Service) πλατφόρμες για web scraping προσφέρουν ολοκληρωμένες υπηρεσίες και ένα σύνολο εργαλείων που μπορεί να χρησιμοποιήσει ο χρήστης για την εξαγωγή των δεδομένων από το διαδίκτυο και τον μετασχηματισμό τους στη μορφή που επιθυμεί. Συνήθως διατίθενται με συνδρομή και προσφέρουν πρόσθετες υπηρεσίες (όπως διαχείριση proxy, υποστήριξη προγραμμάτων περιήγησης κ.α.) που άλλες λύσεις είτε δεν υποστηρίζουν καθόλου είτε υποστηρίζουν μέσω plugin τρίτων. Γενικότερα, η επιλογή μιας SaaS πλατφόρμας προσφέρει ένα ολοκληρωμένο πακέτο με δυνατότητες κλιμάκωσης και συντήρησης.

4.1.1 ScraperAPI

Το ScraperAPI είναι σχεδιασμένο για web scraping σε μεγάλη κλίμακα, καθώς προσφέρει ένα σύνολο λειτουργιών, όπως εύρεση και εναλλαγή των διακομιστών διαμεσολάβησης (proxies), εντοπισμός απαγόρευσης πρόσβασης (ban), επίλυση των CAPTCHA, γεωγραφική στόχευση (geotargeting) και εκτέλεση της JavaScript [26].

Πιο αναλυτικά, τα κύρια χαρακτηριστικά του ScraperAPI είναι τα εξής [26]:

- Εκτέλεση JavaScript. Το API δρομολογεί το αίτημα μέσω ενός instance του Chromium, τρέχει την JavaScript στην ιστοσελίδα και επιστρέφει την HTML απόκριση.
- Γεωγραφική στόχευση. Το API παρέχει γεωγραφική στόχευση σε 13 χώρες. Σε περίπτωση επαναλαμβανόμενης αποτυχίας ενός αιτήματος, χρησιμοποιούνται οικιακοί διακομιστές διαμεσολάβησης. Αν ο χρήστης το επιθυμεί, μπορεί να χρησιμοποιεί αποκλειστικά οικιακές IP (residential IPs) ή IP κινητών συσκευών (mobile IPs).
- Anti-bots & CAPTCHAs. Το API διαθέτει μηχανισμούς παράκαμψης των εργαλείων anti-bots που χρησιμοποιούν πολλοί ιστότοποι. Μόλις παραληφθεί η απάντηση HTML από την ιστοσελίδα-στόχο, τρέχουν αυτόματα οι αλγόριθμοι εντοπισμού ban ή CAPTCHA. Εάν το API εντοπίσει CAPTCHA, επαναλαμβάνει το αίτημα με διαφορετική IP και παράλληλα ξεμπλοκάρει την αποκλεισμένη IP. Με τον τρόπο αυτό δεν χρειάζεται να περιμένουμε την επίλυση του CAPTCHA για να δοκιμάσουμε εκ νέου το αίτημα.

Το ScraperAPI μπορεί να χρησιμοποιηθεί για τη συλλογή δεδομένων από ιστοσελίδες ή API, για τη συλλογή εικόνων, εγγράφων, pdf ή άλλων αρχείων με όριο 2MB για κάθε αίτημα. Υπάρχουν πέντε διαθέσιμοι τρόποι για την αποστολή GET αιτημάτων στο ScraperAPI:

- Μέσω της υπηρεσίας Async Scraper `http://async.scraperapi.com`
- Μέσω του API endpoint `http://api.scraperapi.com?`
- Μέσω του SDK (διαθέσιμο για Python, NodeJS, PHP, Ruby και Java)
- Μέσω της θύρας του proxy `http://scraperapi:APIKEY@proxy-server.scraperapi.com:8001`
- Μέσω της υπηρεσίας Structured Data `https://api.scraperapi.com/structured/`

Ο χρήστης αποστέλλει το URL που τον ενδιαφέρει μαζί με το κλειδί API (API key) και το API επιστρέφει την HTML απόκριση από το συγκεκριμένο URL. Το ScraperAPI χρησιμοποιεί κλειδιά API για τον έλεγχο ταυτότητας των αιτημάτων. Προκειμένου να χρησιμοποιηθεί το API, ο χρήστης θα πρέπει να δημιουργήσει λογαριασμό, και να συμπληρώνει το μοναδικό API key σε κάθε αίτημά του.

Στη συνέχεια θα δούμε ένα παράδειγμα χρήσης του Async Scraper σε Python. Η ασύγχρονη εξαγωγή δεδομένων είναι η πιο κατάλληλη μέθοδος για “δύσκολους” ιστότοπους και για περιπτώσεις που το ποσοστό επιτυχίας είναι σημαντικότερο από τον χρόνο απόκρισης.

Το απλό παράδειγμα που φαίνεται στον Κώδικα 19 δείχνει πώς ο χρήστης μπορεί να υποβάλει μια ασύγχρονη εργασία scraping και να λάβει ένα endpoint URL, μέσω του οποίου θα μπορεί να δει το status και αργότερα το αποτέλεσμα της εργασίας. Στην απάντηση (response) εμφανίζεται το πεδίο statusUrl, το οποίο περιέχει ένα μοναδικό URL για την ανάκτηση των αποτελεσμάτων.

```
import requests
r = requests.post(url = 'https://async.scraperapi.com/jobs', json={
'apiKey': 'xxxxxx', 'url': 'https://example.com' })
print(r.text)

Response:
{
"id":"0962a8e0-5f1a-4e14-bf8c-5efcc18f0953",
"status":"running",
"statusUrl":"https://async.scraperapi.com/jobs/0962a8e0-5f1a-4e14-bf8c-
5efcc18f0953",
"url":"https://example.com"
}
```

Κώδικας 19. ScraperAPI: υποβολή ασύγχρονης εργασίας scraping

Η αποστολή αιτήματος στο συγκεκριμένο endpoint επιστρέφει αρχικά το status και αφού ολοκληρωθεί η εργασία, η απόκριση αλλάζει και επιστρέφει το αποτέλεσμα της εργασίας (Κώδικας 20).

```
import requests
r = requests.get(url = 'https://async.scraperapi.com/jobs/0962a8e0-5f1a-
4e14-bf8c-5efcc18f0953')
print(r.text)

Response:
{
"id":"0962a8e0-5f1a-4e14-bf8c-5efcc18f0953",
"status":"running",
"statusUrl":"https://async.scraperapi.com/jobs/0962a8e0-5f1a-4e14-bf8c-
5efcc18f0953",
"url":"https://example.com"
}
{
Response:
"id":"0962a8e0-5f1a-4e14-bf8c-5efcc18f0953",
"status":"finished",
"statusUrl":"https://async.scraperapi.com/jobs/0962a8e0-5f1a-4e14-bf8c-
5efcc18f0953",
"url":"https://example.com",
"response":{
"headers":{
"date":"Thu, 14 Apr 2022 11:10:44 GMT",
"content-type":"text/html; charset=utf-8",
"content-length":"1256",
"connection":"close",
"x-powered-by":"Express",
"access-control-allow-origin":"undefined","access-control-allow-
headers":"Origin, X-Requested-With, Content-Type, Accept",
"access-control-allow-methods":"HEAD,GET,POST,DELETE,OPTIONS,PUT",
"access-control-allow-credentials":"true",
"x-robots-tag":"none",
"sa-final-url":"https://example.com/",
"sa-statuscode":"200",
"etag":"W/\"4e8-Sjzo7hHgkd15I/TYxuW15B7HwEc\"",
"vary":"Accept-Encoding"
},
"body":"<!doctype html>\n<html>\n
<head>\n
<title>Example Domain</title>\n\n
<meta charset=\"utf-8\" />\n
<meta http-equiv=\"Content-type\" content=\"text/html; charset=utf-8\" />\n
<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\" />\n
<style type=\"text/css\"> ... </style>\n
</head>\n\n
<body> ... </body>\n</html>\n",
"statusCode":200
}
}
```

Κώδικας 20. ScraperAPI: αποστολή αιτήματος στο endpoint και απόκριση

Το ScraperAPI προσφέρει δωρεάν και επί πληρωμή πακέτα χρήσης. Κάθε πακέτο χρήσης έχει συγκεκριμένο αριθμό ταυτόχρονων νημάτων περιορίζοντας τον αριθμό των αιτημάτων που μπορούν να γίνουν παράλληλα (το API δεν δέχεται ομαδικά αιτήματα/batch requests). Όσα περισσότερα ταυτόχρονα νήματα διαθέτει το πακέτο του χρήστη, τόσο αυξάνεται η ταχύτητα εξαγωγής δεδομένων από έναν ιστότοπο [26].

4.1.2 ScrapingBee

Το ScrapingBee διαθέτει ένα REST API με βιβλιοθήκες υποστήριξης για δημοφιλείς γλώσσες προγραμματισμού (Python, cURL, NodeJS, Java, Ruby, Php, Go) το οποίο παρέχει εύκολη πρόσβαση σε όλες τις δυνατότητες της πλατφόρμας. Υποστηρίζει εξαγωγή δεδομένων με CSS selectors, λήψη στιγμιότυπων οθόνης (screenshots), πρόσβαση στο API αναζήτησης της Google, χρήση κοινών διακομιστών διαμεσολάβησης (datacenter proxies) καθώς και premium οικιακών διακομιστών (residential proxies). Προσφέρει, επίσης, πρόσβαση σε μια πλήρη μηχανή προγράμματος περιήγησης Chrome, η οποία είναι χρήσιμη για την εξαγωγή δεδομένων από ιστότοπους με JavaScript.

Το ScrapingBee είναι κατάλληλο για προγραμματιστές και εταιρείες τεχνολογίας που θέλουν να χειρίζονται οι ίδιες τη διαδικασία εξαγωγής των δεδομένων, αλλά χωρίς να εμπλέκονται με διακομιστές διαμεσολάβησης και headless προγράμματα περιήγησης. Το ScrapingBee λειτουργεί σαν μαύρο κουτί αναλαμβάνοντας τη διαχείριση των διακομιστών διαμεσολάβησης και του δικτύου, ενώ ο χρήστης χρειάζεται να προσθέσει μόνο τις διευθύνσεις URL και τις κατάλληλες παραμέτρους για τα request.

Για τη γλώσσα Python υπάρχει διαθέσιμο ένα SDK το οποίο είναι wrapper για τη βιβλιοθήκη requests και υποστηρίζει αιτήματα GET και POST. Η εγκατάσταση του Python SDK του ScrapingBee γίνεται με την εντολή `pip install scrapingbee`. Για την εξαγωγή δεδομένων από έναν ιστότοπο είναι απαραίτητο το API key (το οποίο λαμβάνει ο χρήστης με την εγγραφή του στο ScrapingBee) και η διεύθυνση URL. Στον Κώδικα 21 φαίνεται μια απλή κλήση GET στο API για την εξαγωγή δεδομένων από το URL που ορίζεται στο string YOUR-URL. Κάθε αίτημα που αποτυγχάνει επαναλαμβάνεται όσο το δυνατόν περισσότερες φορές για 30 δευτερόλεπτα. Το API επιστρέφει τον κώδικα HTML του ιστότοπου [27].

```
from scrapingbee import ScrapingBeeClient

client = ScrapingBeeClient(api_key='YOUR-API-KEY')
response = client.get("YOUR-URL")

print('Response HTTP Status Code: ', response.status_code)
print('Response HTTP Response Body: ', response.content)
```

Κώδικας 21. ScrapingBee: αποστολή αιτήματος στο API

Ορισμένα από τα χαρακτηριστικά του ScrapingBee είναι τα εξής [27]:

- Headless browser. Από προεπιλογή το ScrapingBee ανακτά τη διεύθυνση URL που ορίζει ο χρήστης μέσω ενός headless browser που εκτελεί τον κώδικα JavaScript στη σελίδα. Η διεύθυνση URL μπορεί να ανακτηθεί και χωρίς τη χρήση headless browser με την προσθήκη της κατάλληλης παραμέτρου στο αίτημα GET.
- Αλληλεπίδραση με τον ιστότοπο. Το ScrapingBee δίνει τη δυνατότητα αλληλεπίδρασης του χρήστη με τον ιστότοπο και συγκεκριμένα υποστηρίζει τις εξής ενέργειες: πάτημα κουμπιού, αναμονή για ορισμένο χρόνο, αναμονή μέχρι να εμφανιστεί ένα στοιχείο, αναμονή εμφάνισης ενός στοιχείου και κλικ πάνω σε αυτό, scroll οριζόντια ή κάθετα, συμπλήρωση πεδίου input και εκτέλεση προσαρμοσμένου κώδικα JavaScript.
- Αποκλεισμός εικόνων, CSS και διαφημίσεων. Προκειμένου να μειωθεί ο χρόνος των αιτημάτων, το ScrapingBee μπλοκάρει τις εικόνες και τον κώδικα CSS από προεπιλογή. Αν ο χρήστης επιθυμεί να εξάγει τις εικόνες από έναν ιστότοπο, μπορεί να αλλάξει την προεπιλεγμένη λειτουργία, προσθέτοντας την κατάλληλη παράμετρο στο request. Για να μειωθεί ο χρόνος απόκρισης, υπάρχει και η δυνατότητα αποκλεισμού των διαφημίσεων.
- Προσαρμοσμένες κεφαλίδες HTTP. Το ScrapingBee προσθέτει κεφαλίδες σε όλα τα request για να μοιάζουν πραγματικά, αλλά υποστηρίζει και την αποστολή προσαρμοσμένων κεφαλίδων από τον χρήστη. Στην περίπτωση αυτή αποστέλλονται οι κεφαλίδες που ορίζει ο χρήστης, αλλά και όλες οι υπόλοιπες που θεωρούνται απαραίτητες. Αν ο χρήστης δεν επιθυμεί την προσθήκη άλλων κεφαλίδων από το ScrapingBee, προσθέτει την κατάλληλη παράμετρο στο request.

4.1.3 Scrapestack

Το scrapestack είναι ένα απλό REST API που επιτρέπει στον χρήστη να εξάγει δεδομένα από ιστότοπους χωρίς να ανησυχεί για τεχνικά ζητήματα όπως γεωγραφική στόχευση, αλλαγή διευθύνσεων IP ή διαχείριση των CAPTCHA. Το API υποστηρίζει μια σειρά από λειτουργίες για την εξαγωγή δεδομένων ιστού, όπως εκτέλεση κώδικα JavaScript, προσαρμοσμένες κεφαλίδες HTTP, γεωγραφική στόχευση, αιτήματα POST/PUT, καθώς και την επιλογή για χρήση οικιακών διακομιστών διαμεσολάβησης (residential proxies) αντί για διακομιστές διαμεσολάβησης κέντρου δεδομένων (datacenter proxies).

Το scrapestack διαθέτει μια δεξαμενή με περισσότερα από 35 εκατομμύρια διευθύνσεις IP (residential/datacenter) σε δεκάδες ISP ανά τον κόσμο και υποστηρίζει πραγματικές συσκευές, επαναπροσπάθεια (retry) σε περίπτωση αποτυχίας και εναλλαγή IP. Επίσης, υποστηρίζει σχεδόν 200 τοποθεσίες στις οποίες συμπεριλαμβάνονται πολλές μεγαλουπόλεις σε όλο τον κόσμο. Ένα ακόμα χαρακτηριστικό είναι ότι διαθέτει cloud υποδομή που το καθιστά εξαιρετικά επεκτάσιμο [28].

Για την εξαγωγή δεδομένων με το API του scrapestack, ο χρήστης θα πρέπει να προσθέσει στο endpoint του API το URL του ιστότοπου και το κλειδί πρόσβασης στο API (μοναδικό κλειδί το οποίο παρέχεται στον χρήστη με τη δημιουργία λογαριασμού στο scrapestack). Παρακάτω φαίνεται ένα παράδειγμα για τη συλλογή δεδομένων από το URL https://apple.com.

```
https://api.scrapestack.com/scrape
    ? access_key = YOUR_ACCESS_KEY
    & url = https://apple.com
```

Εάν το αίτημα είναι επιτυχημένο, το API θα επιστρέψει τον HTML κώδικα της σελίδας. Σε περίπτωση που έχουν προστεθεί κεφαλίδες HTTP στο αίτημα, η απόκριση του API θα περιλαμβάνει και τις κεφαλίδες αυτές. Στον Κώδικα 22 φαίνεται ένα αίτημα GET σε γλώσσα Python.

```
import requests

params = {
  'access_key': 'YOUR_ACCESS_KEY',
  'url': 'http://scrapestack.com'
}

api_result = requests.get('http://api.scrapestack.com/scrape', params)
website_content = api_result.content

print(website_content)
```

Κώδικας 22. Scrapestack: αποστολή αιτήματος στο API

Το API του scrapestack δίνει επίσης τη δυνατότητα αποστολής δεδομένων σε φόρμες ή API endpoint μέσω αιτημάτων POST/PUT. Παρακάτω φαίνεται ένα POST request με curl [28].

```
curl -d 'foo=bar' \
-X POST \
"https://api.scrapestack.com/scrape?access_key=YOUR_ACCESS_KEY&url=https://app
le.com"
```

4.1.4 Diffbot

Το Diffbot διαθέτει το Extract API, το οποίο χρησιμοποιεί μηχανική όραση (computer vision) και επεξεργασία φυσικής γλώσσας (NLP) για την αυτόματη κατηγοριοποίηση ενός ιστότοπου και την εξαγωγή του περιεχομένου σε μορφή JSON. Μπορεί να χρησιμοποιηθεί για την εξαγωγή δεδομένων από πολλούς διαφορετικούς ιστότοπους και διευκολύνει σημαντικά τον χρήστη, γιατί ο ορισμός κανόνων για κάθε μεμονωμένη σελίδα είναι μια διαδικασία χρονοβόρα και δύσκολη στη συντήρηση σε μεγάλη κλίμακα.

Το Extract API του Diffbot διαθέτει ένα σύνολο APIs για την εξαγωγή δεδομένων ανάλογα με τον τύπο του ιστότοπου, τα οποία είναι τα εξής [29]:

- Article API: για την εξαγωγή πληροφοριών από ειδησεογραφικά άρθρα, αναρτήσεις ιστολογίου ή άλλο γραπτό περιεχόμενο. Το Diffbot μπορεί να αναγνωρίσει τους συγγραφείς, τις εικόνες και τους συνδέσμους του προφίλ τους, τις ημερομηνίες δημοσίευσης, τις εικόνες των άρθρων, τα σχόλια, τη γλώσσα στην οποία είναι γραμμένο το περιεχόμενο κ.ά.
- Product API: για τη συλλογή πληροφοριών από προϊόντα όπως χρώματα, διαθεσιμότητα, τιμές, εκπτωτικές τιμές, επιλογές αποστολής, περιγραφή, κριτικές κ.ά.
- Image API: για την εξαγωγή πληροφοριών από εικόνες όπως οι διαστάσεις και τα URL για τη λήψη των εικόνων μέχρι και πληροφορίες για το περιεχόμενο των εικόνων μέσω image recognition.
- Video API: είναι αντίστοιχο του Image API αλλά για βίντεο.
- Discussion API: για την εξαγωγή δεδομένων από thread περιεχομένου. Αυτό μπορεί να είναι η κριτική ενός προϊόντος (το Product API χρησιμοποιεί το Discussion API για να εξάγει σχόλια πελατών), ένα φόρουμ ή ένα thread του Reddit, ή μια ενότητα για σχόλια σε ένα ιστολόγιο.

Εάν ο χρήστης δεν είναι σίγουρος για τον τύπο του ιστότοπου, μπορεί να χρησιμοποιήσει το Analyze API το οποίο χρησιμοποιεί μηχανική μάθηση, προκειμένου να ταξινομήσει αυτόματα το URL που εισάγει ο χρήστης και να το οδηγήσει σε ένα από τα παραπάνω API. Υπάρχει, επίσης, και το Custom API που μπορεί να χρησιμοποιηθεί για τη διόρθωση των δεδομένων που εξάγονται ή για την εξαγωγή δεδομένων με χρήση κανόνων. Διαθέτει μια διεπαφή point-and-click που επιτρέπει στον χρήστη τη συλλογή δεδομένων με CSS selectors, κανονικές εκφράσεις και φίλτρα χαρακτηριστικών (attributes). Τέλος, το Custom API μπορεί να χρησιμοποιηθεί και μέσω προγραμματισμού [29].

Ορισμένα από τα χαρακτηριστικά των Extract APIs του Diffbot είναι [29]:

- Αποστολή προσαρμοσμένων κεφαλίδων HTTP. Το Diffbot υποστηρίζει τη ρύθμιση των κεφαλίδων User-Agent, Referer, Cookie, Accept-Language και X-Evaluate.
- Εισαγωγή προσαρμοσμένου κώδικα JavaScript προτού το Extract API επεξεργαστεί μια σελίδα.
- Χρήση διακομιστών διαμεσολάβησης (proxies). Το Diffbot προσφέρει δύο επίπεδα διακομιστών διαμεσολάβησης, τους προεπιλεγμένους και τους δυναμικούς. Οι προεπιλεγμένοι proxies χρησιμοποιούνται για τους περισσότερους ιστότοπους, ενώ οι δυναμικοί proxies παρέχουν μια νέα διεύθυνση IP για κάθε αίτημα και είναι χρήσιμοι στις πιο δύσκολες περιπτώσεις εξαγωγής δεδομένων.

Στον Κώδικα 23 φαίνεται ένα παράδειγμα χρήσης του Analyze API σε Python.

```
import requests

url  =  "https://api.diffbot.com/v3/analyze?url=https%3A%2F%2Fwww.technology
review.com%2F2020%2F09%2F04%2F1008156%2Fknowledge-graph-ai-reads-web-
machine-learning-natural-language-processing%2F"

headers = {"accept": "application/json"}

response = requests.get(url, headers=headers)

print(response.text)
```

Κώδικας 23. Diffbot: αποστολή αιτήματος στο Analyze API

4.1.5 Σύγκριση των SaaS εργαλείων

Στον Πίνακα 8 γίνεται σύγκριση μεταξύ των κύριων χαρακτηριστικών των SaaS εργαλείων που παρουσιάστηκαν παραπάνω.

Πίνακας 8. Σύγκριση των SaaS εργαλείων

	ScraperAPI	**ScrapingBee**	**Scrapestack**	**Diffbot**
Εκτέλεση JavaScript	X	X	X	X
Προσαρμοσμένες κεφαλίδες HTTP	X	X	X	X
Αυτόματη εναλλαγή IP	X	X	X	X
Γεωγραφική στόχευση (geotargeting)	X	Διαθέσιμη μόνο για residential proxies	X	X
Οικιακοί διακομιστές διαμεσολάβησης (residential proxies)	X	X	X	X
Διαχείριση CAPTCHA	X	-	X	X
Αιτήματα POST/PUT	X	X	X	X
Ταυτόχρονα αιτήματα HTTP (concurrent requests)	X	X	X	X
User agent για σταθερές & κινητές συσκευές	X	X	-	X
Επανάληψη αιτημάτων HTTP (retries)	X	X	X	X
Sessions (χρήση του ίδιου proxy για πολλαπλά αιτήματα HTTP)	X	X	-	X

4.2 Desktop εφαρμογές

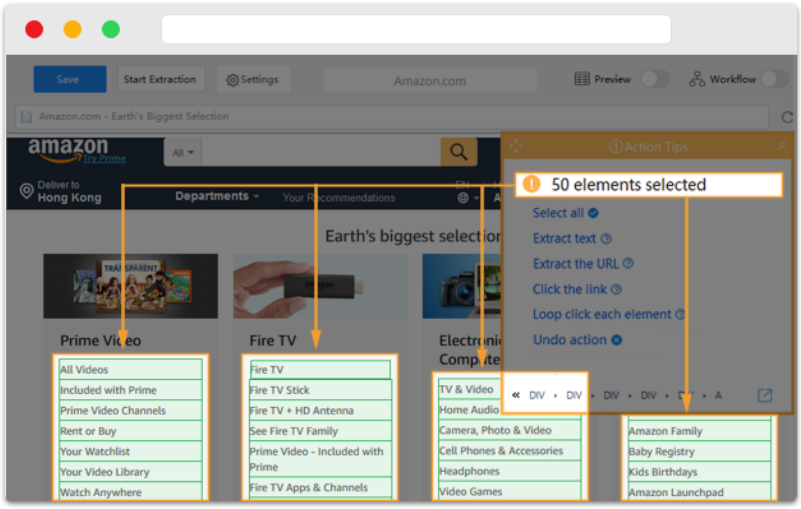
Εκτός από τις εφαρμογές SaaS υπάρχουν και οι εφαρμογές που εγκαθίστανται τοπικά στον υπολογιστή του χρήστη. Συνήθως δε χρειάζονται συνδρομή, αλλά είτε διατίθενται δωρεάν είτε απαιτούν την αγορά άδειας χρήσης. Στην περίπτωση αυτή, ο χρήστης είναι υπεύθυνος για την παροχή του υλικού, τη σύνδεση και τη συνολική συντήρηση του συστήματος, ενώ μπορεί να αντιμετωπίσει δυσκολίες στην κλιμάκωση. Η λύση αυτή είναι κατάλληλη για μικρότερα έργα.

4.2.1 Octoparse

Το Octoparse είναι desktop εφαρμογή για Windows και Mac. Μπορεί να χρησιμοποιηθεί για εξαγωγή δεδομένων από διάφορα είδη ιστότοπων όπως μέσα κοινωνικής δικτύωσης, ηλεκτρονικά καταστήματα ή ιστοσελίδες καταχώρισης ακινήτων, και βοηθά τον χρήστη στην αυτόματη παρακολούθηση προϊόντων και τιμών, στην προσέλκυση δυνητικών πελατών (lead generation) κλπ. Τα κύρια χαρακτηριστικά του Octoparse είναι τα εξής:

- Εύχρηστη διεπαφή. Ο χρήστης ανοίγει τον ιστότοπο στο ενσωματωμένο πρόγραμμα περιήγησης του Octoparse και επιλέγει με κλικ τα δεδομένα που τον ενδιαφέρουν. Το Octoparse εφαρμόζει έναν προηγμένο αλγόριθμο μηχανικής μάθησης για να εντοπίζει με ακρίβεια τα δεδομένα.
- Κατάλληλο για ιστότοπους κάθε είδους. Το Octoparse μπορεί να εξάγει δεδομένα από ιστότοπους με JavaScript ή AJAX και γενικά από οποιονδήποτε δυναμικό ιστότοπο. Δίνει, επίσης, τη δυνατότητα για σύνδεση (login), συμπλήρωση φόρμας, εισαγωγή όρων αναζήτησης, άπειρη κύλιση (infinite scroll), επιλογή από dropdown κ.α.
- Cloud υποδομή. Η cloud υποδομή απαλλάσσει τον χρήστη από την ανάγκη αγοράς και συντήρησης hardware υψηλού κόστους ή τα προβλήματα σύνδεσης στο διαδίκτυο. Η cloud πλατφόρμα του Octoparse επιτρέπει 6 έως 20 φορές πιο γρήγορη εξαγωγή δεδομένων σε σύγκριση με την desktop εφαρμογή και εκτέλεση 24/7. Τα δεδομένα εξάγονται και αποθηκεύονται στο cloud και είναι προσβάσιμα από οποιοδήποτε μηχάνημα.
- Αυτόματη εναλλαγή IP. Η υπηρεσία Octoparse Cloud υποστηρίζεται από εκατοντάδες cloud servers, ο καθένας με μια μοναδική διεύθυνση IP. Όταν μια εργασία εξαγωγής δεδομένων εκτελείται στο cloud, τα αιτήματα υποβάλλονται στον ιστότοπο-στόχο με διαφορετικές IP, έτσι ώστε να μειώνονται οι πιθανότητες αποκλεισμού.
- Προγραμματισμός των εργασιών scraping. Για ιστότοπους που ενημερώνονται διαρκώς υπάρχει η δυνατότητα προγραμματισμού των εργασιών εξαγωγής σε οποιαδήποτε χρονική στιγμή της ημέρας, της εβδομάδας ή του μήνα. Για την υλοποίηση real-time scraping μπορεί να προγραμματιστεί η εκτέλεση μιας εργασίας ανά λεπτό.

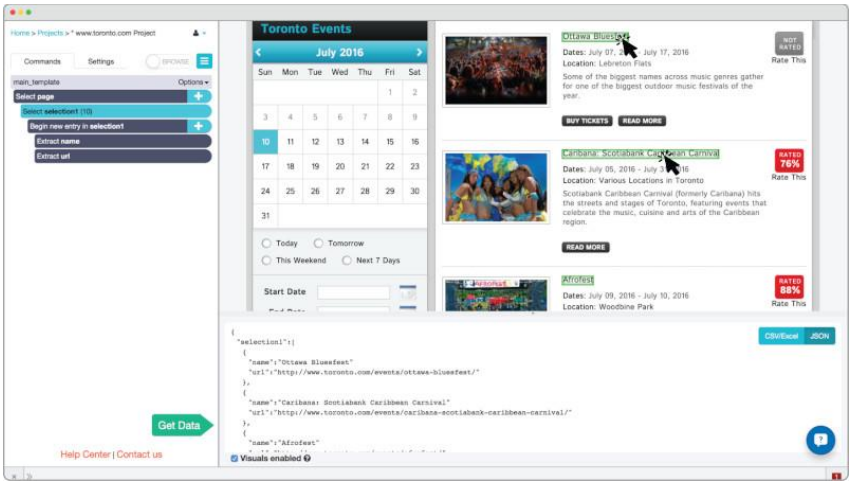
- API, CSV, Excel, βάση δεδομένων. Το Octoparse προσφέρει πρόσβαση στα εξαγόμενα δεδομένα με Excel, API ή δυνατότητα εξαγωγής στη βάση δεδομένων του χρήστη. Με σύνδεση στο Octoparse API ο χρήστης μπορεί εύκολα να μεταφέρει τα δεδομένα στο δικό του σύστημα [30].



Εικόνα 6. Octoparse [30]

4.2.2 ParseHub

Το ParseHub είναι desktop εφαρμογή διαθέσιμη για Windows, Mac και Linux. Διαθέτει, επίσης, επέκταση για προγράμματα περιήγησης προσφέροντας εύκολη και άμεση εξαγωγή δεδομένων ιστού. Μπορεί να χρησιμοποιηθεί για εξαγωγή δεδομένων από παράθυρα pop-up, χάρτες, εικόνες κ.α. και διαθέτει αρκετά καλή τεκμηρίωση για τους χρήστες. Ο χρήστης μπορεί να επιλέξει δωρεάν πακέτο ή με πληρωμή ανάλογα τις ανάγκες του.



Εικόνα 7. ParseHub [31]

Ορισμένα από τα χαρακτηριστικά του ParseHub είναι:

- Εύκολη επιλογή δεδομένων. Ο χρήστης κάνει κλικ πάνω στα δεδομένα που θέλει να εξάγει από τον ιστότοπο και το ParseHub συλλέγει τα δεδομένα.
- ParseHub API. Το API δίνει τη δυνατότητα στον χρήστη να δημιουργήσει τον δικό του scraper και να κατεβάσει τα δεδομένα σε μορφή CSV ή JSON.
- Ευελιξία. Το ParseHub διαθέτει έναν έξυπνο αλγόριθμο που αναγνωρίζει μοτίβα στα δεδομένα. Ο χρήστης μπορεί να χρησιμοποιήσει κανονικές εκφράσεις ή να τροποποιήσει τους CSS selectors και να επεξεργαστεί τις ιδιότητες (attributes) των στοιχείων.
- Υποστήριξη για διαδραστικούς και περίπλοκους ιστότοπους. Το ParseHub είναι κατάλληλο για κάθε είδους ιστότοπο. Ο χρήστης μπορεί να συνδυάσει τα διαθέσιμα εργαλεία και να χειριστεί εύκολα ανακατευθύνσεις, φόρμες, dropdowns, χάρτες, άπειρη κύλιση (infinite scroll), σύνδεση με credentials (login) και γενικά ιστότοπους που χρησιμοποιούν AJAX και JavaScript.
- Άμεση εμφάνιση αποτελεσμάτων. Το ParseHub εμφανίζει αμέσως ένα δείγμα των δεδομένων και έτσι ο χρήστης δεν χρειάζεται να περιμένει μέχρι να ολοκληρωθεί η εκτέλεση για να δει τα αποτελέσματα.
- Πλοήγηση μεταξύ των σελίδων. Το ParseHub μπορεί να χρησιμοποιηθεί σε ιστότοπους με χιλιάδες συνδεδεμένες σελίδες και σελιδοποίηση (pagination). Ο χρήστης μπορεί σχεδιάσει ένα σύνολο κανόνων για κάθε σελίδα, να συνδέσει σελίδες ή να κάνει ανακατεύθυνση σε ένα διαφορετικό URL, χωρίς να χρειάζεται να εισάγει τα URL ένα προς ένα.
- Αυτόματη εναλλαγή IP. Τα αιτήματα δρομολογούνται μέσω μιας μεγάλης δεξαμενής διαθέσιμων IP προκειμένου ο χρήστης να διατηρήσει το απόρρητο και την ανωνυμία του.
- Cloud υποδομή. Τα δεδομένα των χρηστών αποθηκεύονται στο cloud και είναι διαθέσιμα οποιαδήποτε στιγμή. Υπάρχει, επίσης, η δυνατότητα προγραμματισμού της ανάκτησης των δεδομένων ανά λεπτό, ώρα, ημέρα, εβδομάδα ή μήνα [31].

4.2.3 ScrapeStorm

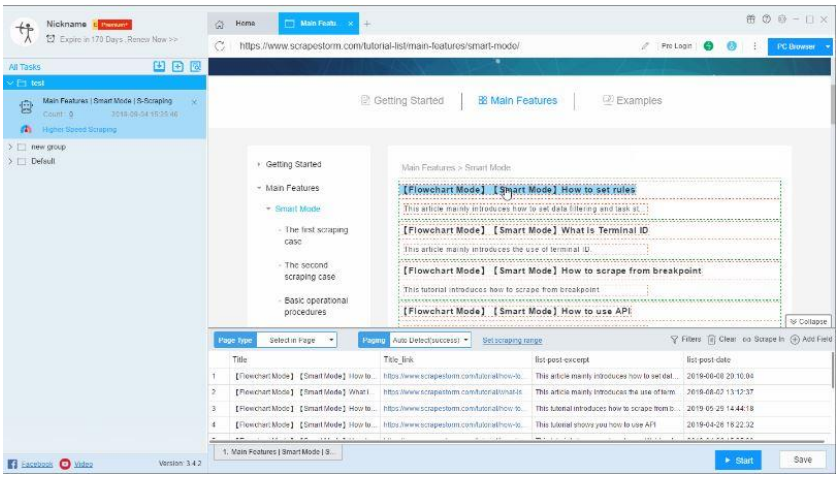
Το ScrapeStorm είναι desktop εφαρμογή διαθέσιμη για Windows, Mac και Linux που βασίζεται στην τεχνολογία της τεχνητής νοημοσύνης. Είναι ένα εύχρηστο λογισμικό κατάλληλο για χρήστες χωρίς γνώσεις προγραμματισμού που μπορεί να χρησιμοποιηθεί σε πολλούς τομείς, όπως πωλήσεις, οικονομικά, ειδήσεις, ηλεκτρονικό εμπόριο, ανάλυση δεδομένων, καθώς και από κρατικούς φορείς ή για ακαδημαϊκή έρευνα.

Το ScrapeStorm υποστηρίζει δύο διαφορετικές λειτουργίες εξαγωγής δεδομένων. Η πρώτη λειτουργία, που ονομάζεται “Smart Mode”, είναι εξαιρετικά απλή και ο χρήστης χρειάζεται να εισάγει μόνο το URL του ιστότοπου. Το ScrapeStorm αναγνωρίζει το περιεχόμενο της σελίδας και

εξάγει τα δεδομένα χωρίς ο χρήστης να δημιουργήσει κανόνες. Στη δεύτερη λειτουργία, που ονομάζεται “Flowchart Mode”, ο χρήστης ανοίγει τον ιστότοπο και ακολουθεί τις υποδείξεις του λογισμικού για να δημιουργήσει αυτόματα κανόνες εξαγωγής των δεδομένων με μερικά κλικ.

Τα δεδομένα που συλλέγονται μπορούν να αποθηκευτούν τοπικά σε αρχείο ή σε cloud server. Το ScrapeStorm υποστηρίζει εξαγωγή των δεδομένων σε Excel, CSV, TXT, HTML, MySQL, MongoDB, SQL Server, PostgreSQL, WordPress και Google Sheets. Ορισμένα ακόμη χαρακτηριστικά του ScrapeStorm είναι η δυνατότητα προγραμματισμού της εξαγωγής των δεδομένων, η αυτόματη εναλλαγή IP, η διάθεση ενός RESTful API, η δυνατότητα λήψης αρχείων κ.ά.

Ο χρήστης μπορεί να δημιουργήσει λογαριασμό και να αποθηκεύει όλες τις εργασίες εξαγωγής δεδομένων στο cloud. Η cloud πλατφόρμα δίνει στον χρήστη τη δυνατότητα να συνδέεται στο λογαριασμό του και να χρησιμοποιεί το εργαλείο από οποιονδήποτε υπολογιστή [32].



Εικόνα 8. ScrapeStorm [32]

4.2.4 Σύγκριση των desktop εφαρμογών

Στον Πίνακα 9 γίνεται σύγκριση των desktop εφαρμογών για web scraping που παρουσιάστηκαν προηγουμένως.

Πίνακας 9. Σύγκριση των desktop εφαρμογών

	Octoparse	**ParseHub**	**ScrapeStorm**
Λειτουργικό σύστημα	Windows, Mac	Windows, Mac, Linux	Windows, Mac, Linux
Cloud πλατφόρμα	X	X	X
Αυτόματη εναλλαγή IP	X	X	X
Υποστήριξη API	X	X	X

Επιλογή δεδομένων	Point-and-click, XPath	Point-and-click, XPath, CSS selectors	Point-and-click
Λήψη αρχείων & εικόνων	Μόνο τοπικά στο μηχάνημα του χρήστη	X	X
Εξαγωγή δεδομένων	Excel, CSV, JSON, HTML	Excel, CSV, JSON	Excel, CSV, TXT, HTML
Αποθήκευση δεδομένων	SqlServer, MySql, Oracle	DropBox, Amazon S3	MySQL, MongoDB, SQL Server, PostgreSQL
Προγραμματισμός των tasks	X	X	X
Integrations	-	Google Sheets, Tableau	Google Sheets, WordPress

4.3 Επεκτάσεις προγραμμάτων περιήγησης

Μια άλλη δημοφιλής κατηγορία εργαλείων web scraping βασίζεται σε πρόσθετα (extensions) των προγραμμάτων περιήγησης. Τα εργαλεία αυτά τρέχουν απευθείας στον browser και χρησιμοποιούν πλήρως τη μηχανή του browser και τις ενσωματωμένες τεχνολογίες ιστού (DOM, CSS selectors και JavaScript). Για τους browsers Chrome και Firefox υπάρχουν διαθέσιμα πολλά εργαλεία τέτοιου είδους.

4.3.1 Webscraper.io

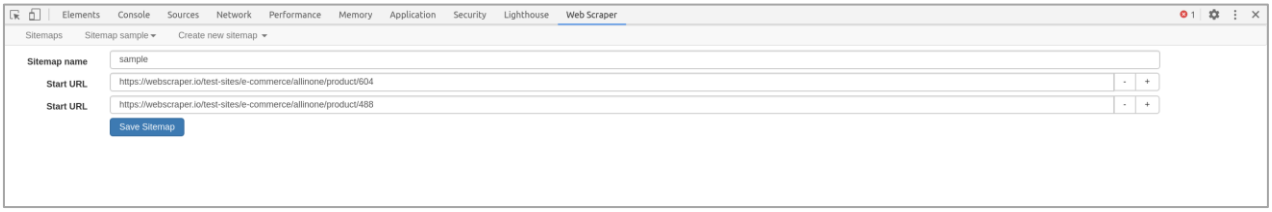
Το εργαλείο Web Scraper είναι επέκταση (extension) για Chrome και Firefox. Διατίθεται και ως cloud extension, που ονομάζεται Web Scraper Cloud, και είναι μια premium υπηρεσία που εμπλουτίζει το Web Scraper με λειτουργίες αυτοματισμού, είναι επεκτάσιμο και επιτρέπει την παρακολούθηση των εργασιών εξαγωγής δεδομένων.

Για να χρησιμοποιήσει κανείς το Web Scraper χρειάζεται αρχικά να προσθέσει την επέκταση στον browser και στη συνέχεια να ανοίξει την καρτέλα Web Scraper στο παράθυρο Developers Tools. Τα επόμενα βήματα είναι:

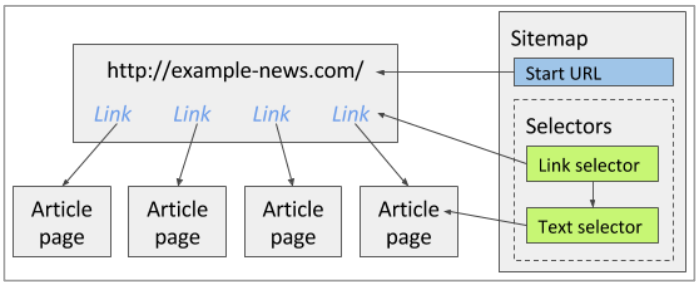
- Δημιουργία χάρτη ιστοτόπου (sitemap). Ο χρήστης προσθέτει το URL από το οποίο θα ξεκινήσει η εξαγωγή των δεδομένων (βλ. Εικόνα 10). Μπορεί να προσθέσει περισσότερα URL αν ενδιαφέρεται για δεδομένα από πολλαπλές διευθύνσεις, για παράδειγμα δεδομένα από διαφορετικά αποτελέσματα αναζήτησης. Αν ο ιστότοπος χρησιμοποιεί αρίθμηση για τις σελίδες, ο χρήστης μπορεί να ορίσει το εύρος των σελίδων που τον ενδιαφέρουν.
- Ορισμός επιλογέων (selectors). Αφού δημιουργηθεί ο χάρτης ιστοτόπου, ο χρήστης μπορεί να προσθέσει επιλογείς οι οποίοι οργανώνονται σε μορφή δέντρου. Ο Web Scraper εκτελεί τους επιλογείς με τη σειρά που βρίσκονται στο δέντρο. Για παράδειγμα, αν ο χρήστης θέλει

να εξάγει τα άρθρα από την αρχική σελίδα ενός ειδησεογραφικού site, θα ορίσει έναν Link selector για την επιλογή των συνδέσμων και θα προσθέσει έναν θυγατρικό Text selector για να εξάγει το περιεχόμενο του κάθε συνδέσμου/άρθρου. Στην Εικόνα 11 φαίνεται πως θα πρέπει να χτιστεί ο χάρτης ιστοτόπου σε αυτή την περίπτωση. Στο παράθυρο “Selector graph” ο χρήστης μπορεί να δει με γραφικό τρόπο τη δενδρική δομή με το σύνολο των επιλογέων. Ένα παράδειγμα φαίνεται στην Εικόνα 12.

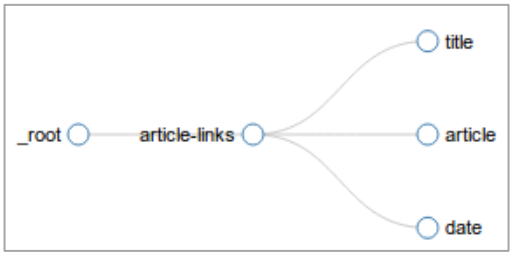
- Εξαγωγή των δεδομένων. Στο παράθυρο “Scrape” ο χρήστης μπορεί να ξεκινήσει τη συλλογή των δεδομένων. Προαιρετικά, μπορεί να οριστεί ο χρόνος μεταξύ των αιτημάτων (request interval) και ο χρόνος αναμονής για τη φόρτωση των σελίδων (page load delay). Στη συνέχεια, εμφανίζεται ένα popup παράθυρο στο οποίο ο Web Scraper φορτώνει τις σελίδες για να εξάγει τα δεδομένα. Μόλις ολοκληρωθεί η διαδικασία, ο χρήστης μπορεί να δει τα αποτελέσματα στο παράθυρο “Browse” και να τα κατεβάσει σε αρχείο CSV ανοίγοντας το παράθυρο “Export data as CSV” [33].



Εικόνα 9. Web Scraper: εισαγωγή url [33]



Εικόνα 10. Web Scraper: χάρτης ιστοτόπου (sitemap) [33]



Εικόνα 11. Web Scraper: selector graph [33]

Η premium υπηρεσία που ονομάζεται Web Scraper Cloud έχει τα εξής χαρακτηριστικά:

- Υποστήριξη διακομιστή διαμεσολάβησης (proxy).
- Εξαγωγή δεδομένων σε CSV, XLSX ή JSON. Υπάρχει, επίσης, η δυνατότητα για αυτόματη εξαγωγή των δεδομένων σε Dropbox, Google Sheets ή Amazon S3.
- Υποστήριξη API. Η διαχείριση του Web Scraper Cloud μπορεί να γίνει μέσω ενός HTTPS JSON API. Το API επιτρέπει τη διαχείριση των sitemap, τη δημιουργία εργασιών scraping

και τη λήψη των δεδομένων. Για την ανάπτυξη εφαρμογών σε γλώσσα PHP υπάρχει διαθέσιμο ένα PHP SDK.

- Παράλληλη εκτέλεση εργασιών scraping. Ο χρήστης μπορεί να τρέχει παράλληλες εργασίες τις οποίες μπορεί να διακόψει και να τις συνεχίσει αργότερα ή να αλλάξει τη σειρά με την οποία θα εκτελεστούν.
- Δύο τύποι drivers. Ο Web Scraper Cloud υποστηρίζει δύο διαφορετικούς drivers για την εκτέλεση των εργασιών scraping. Ο "Full driver" φορτώνει τις σελίδες όπως και η επέκταση Web Scraper για τα προγράμματα περιήγησης. Στην περίπτωση αυτή φορτώνονται όλα τα στοιχεία και η JavaScript εκτελείται πριν ξεκινήσει η συλλογή των δεδομένων. Ο "Fast driver" δεν εκτελεί την JavaScript, τα δεδομένα εξάγονται από τον HTML κώδικα της σελίδας και ο χρόνος φόρτωσης είναι μικρότερος.
- Παρακολούθηση των εργασιών. Ο χρήστης μπορεί να ελέγχει την πρόοδο κάθε εργασίας scraping, όπως για παράδειγμα τον αριθμό των σελίδων που έχει διατρέξει ο scraper, τις σελίδες που φορτώθηκαν με κωδικό σφάλματος 4xx ή 5xx, το πλήθος των δεδομένων που έχουν εξαχθεί κλπ. Ο Web Scraper Cloud επαναλαμβάνει τις εργασίες scraping στις σελίδες που απέτυχε να φορτώσει (failed pages) και στις σελίδες που φορτώθηκαν με επιτυχία αλλά οι επιλογείς δεν εξήγαγαν δεδομένα (empty pages).
- Καταγραφή και επίλυση προβλημάτων. Ο χρήστης έχει στη διάθεση του τη λίστα με τις σελίδες από τις οποίες δεν συλλέχθηκαν δεδομένα για κάθε εργασία scraping. Όταν μια εργασία βρίσκεται σε εξέλιξη δημιουργείται ένα γράφημα απόδοσης, τα δεδομένα του οποίου διατηρούνται για 7 μέρες [33].

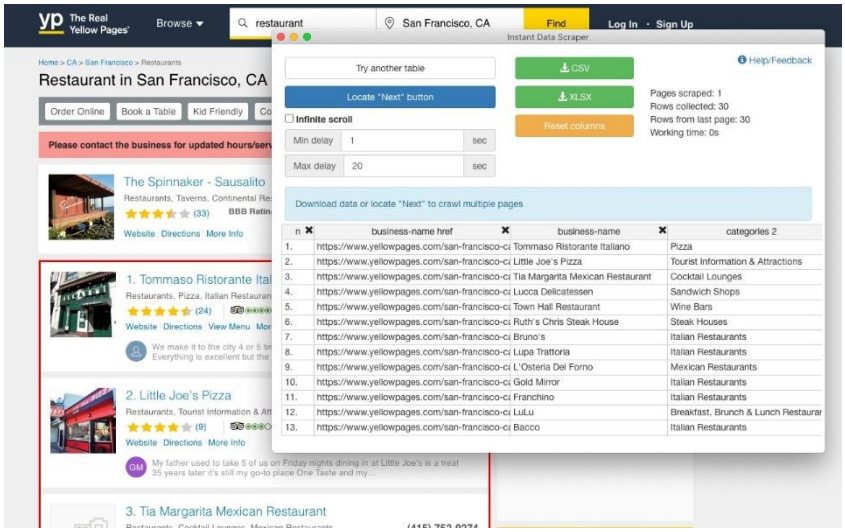
4.3.2 Instant Data Scraper

Το Instant Data Scraper είναι επέκταση για τα προγράμματα περιήγησης Chrome και Edge. Είναι ένα εργαλείο για εξαγωγή δεδομένων που χρησιμοποιεί τεχνητή νοημοσύνη (AI) για να προβλέψει ποια δεδομένα χρειάζεται να συλλεχθούν από μια σελίδα HTML και επιτρέπει την αποθήκευσή τους σε αρχείο Excel ή CSV. Δεν απαιτεί τη συγγραφή συγκεκριμένου script για κάθε ιστότοπο, καθώς πραγματοποιεί ευριστική ανάλυση της δομής της HTML με τεχνητή νοημοσύνη για να εντοπίσει τα κατάλληλα δεδομένα για εξαγωγή. Εάν η πρόβλεψη δεν είναι ικανοποιητική, ο χρήστης μπορεί να προσαρμόσει τις επιλογές για μεγαλύτερη ακρίβεια.

Τα κυριότερα χαρακτηριστικά του είναι:

- Ανίχνευση δεδομένων για εξαγωγή με AI.
- Προσαρμογή του χρόνου καθυστέρησης (delay time) και του μέγιστου χρόνου αναμονής (wait time).
- Υποστήριξη για ιστότοπους με σελιδοποίηση (pagination).

- Αυτόματη πλοήγηση στην επόμενη σελίδα μέσω κουμπιών ή συνδέσμων.
- Υποστήριξη για άπειρη κύλιση (infinite scroll).
- Προεπισκόπηση των εξαγόμενων δεδομένων με δυνατότητα αντιγραφής και επικόλλησης.
- Εξαγωγή δεδομένων σε αρχείο Excel ή CSV.
- Δυνατότητα για μετονομασία και φιλτράρισμα των στηλών των εξαγόμενων δεδομένων [34].



Εικόνα 12. Instant Data Scraper [34]

4.3.3 Data Miner

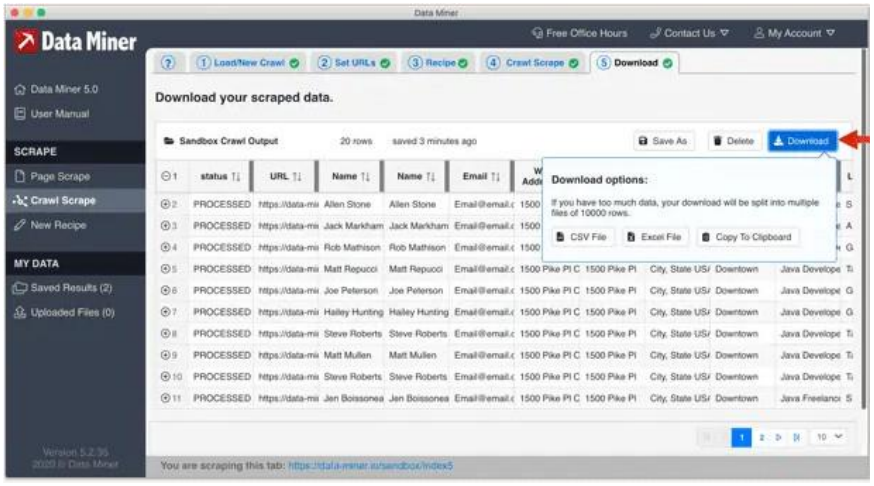
Το Data Miner είναι επέκταση για τα προγράμματα περιήγησης Chrome και Edge. Μπορεί να εξάγει δεδομένα από μια μόνο σελίδα ή να ακολουθεί τους συνδέσμους ενός ιστότοπου και να εξάγει δεδομένα από περισσότερες σελίδες. Το Data Miner μπορεί να εξάγει οποιοδήποτε κείμενο βλέπει ο χρήστης στο πρόγραμμα περιήγησής του, να κάνει αυτόματα κλικ σε κουμπιά και συνδέσμους, να ανοίγει popup παράθυρα και να συλλέγει δεδομένα από αυτά. Ο χρήστης μπορεί να κατεβάσει τα δεδομένα σε αρχείο CSV ή Excel.

Μερικές ακόμη από τις δυνατότητες του Data Miner είναι οι εξής:

- Εύκολη εξαγωγή δεδομένων από πίνακες και λίστες.
- Συλλογή δεδομένων από σελίδες που προϋποθέτουν σύνδεση (login) ή βρίσκονται πίσω από firewall.
- Εκτέλεση οδηγιών που γράφονται από τον χρήστη για τη συλλογή των δεδομένων. Οι οδηγίες αυτές μπορούν να περιλαμβάνουν μετάβαση στην επόμενη σελίδα, προσαρμοσμένη JavaScript, ενέργειες όπως κλικ και scroll κ.ά.

- Εκτέλεση προσαρμοσμένου κώδικα JavaScript για την επεξεργασία των δεδομένων, για παράδειγμα για την εξαγωγή διευθύνσεων email.
- Αυτόματη συμπλήρωση φορμών με δεδομένα τα οποία παρέχει ο χρήστης μέσω αρχείου Excel.

Το Data Miner διατίθεται σαν δωρεάν υπηρεσία αλλά με περιορισμούς όσον αφορά τα domain και το πλήθος των σελίδων από τις οποίες μπορεί να εξάγει δεδομένα (500 σελίδες/μήνα), ενώ η χρήση προσαρμοσμένου κώδικα JavaScript δεν υποστηρίζεται σε αυτή την περίπτωση. Εκτός από το δωρεάν πακέτο υπάρχει και μια σειρά premium πακέτων με πρόσθετες λειτουργίες από τα οποία ο χρήστης μπορεί να επιλέξει [35].



Εικόνα 13. Data Miner [35]

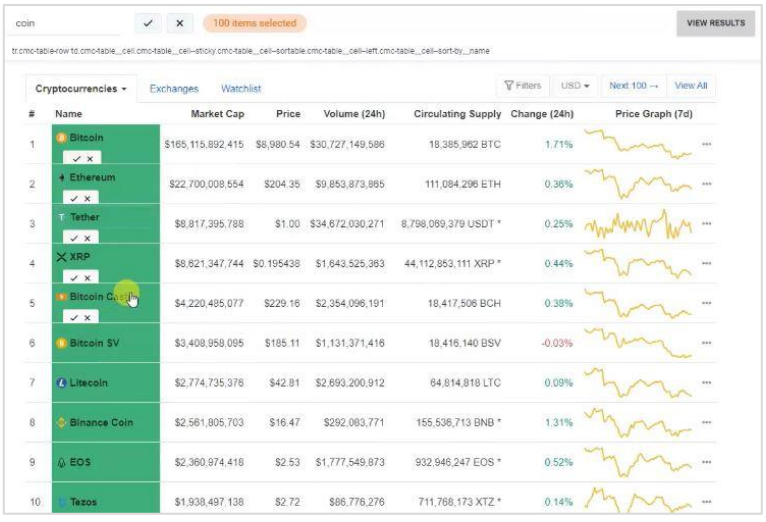
4.3.4 Simplescraper

Μια ακόμη δημοφιλής επέκταση για τον Chrome είναι το Simplescraper. Πρόκειται για ένα εργαλείο που επιτρέπει την εύκολη και γρήγορη εξαγωγή δεδομένων από οποιονδήποτε ιστότοπο και τη μετατροπή τους σε δομημένα δεδομένα.

Το Simplescraper μπορεί να χρησιμοποιηθεί για τοπική εξαγωγή δεδομένων (local scraping) κατά την οποία ο χρήστης μπορεί να συλλέξει τα δεδομένα που είναι ορατά στο πρόγραμμα περιήγησής του. Μπορεί, όμως, να χρησιμοποιηθεί και για cloud scraping μέσω προγραμμάτων περιήγησης που βρίσκονται στο cloud. Εάν ο χρήστης θέλει να κατεβάσει τα δεδομένα που βλέπει σε μια μεμονωμένη ιστοσελίδα, μπορεί να εγκαταστήσει και να χρησιμοποιήσει την επέκταση τοπικά στον browser του υπολογιστή του. Εάν, όμως, χρειάζεται περισσότερες λειτουργίες συμπεριλαμβανομένου ενός API, εξαγωγή από πολλαπλές σελίδες ή ιστότοπους με άπειρη κύλιση (infinite scroll) και δυνατότητα προγραμματισμού των εργασιών εξόρυξης, θα πρέπει να χρησιμοποιήσει την cloud υπηρεσία.

Τα κυριότερα χαρακτηριστικά του Simplescraper είναι:

- Εύκολη επιλογή των δεδομένων προς εξαγωγή με κλικ πάνω στα δεδομένα.
- Λήψη των δεδομένων σε μορφή CSV ή JSON.
- Απευθείας εξαγωγή των δεδομένων σε Google Sheets, Airtable, Zapier ή Make (Integromat).
- Συλλογή δεδομένων από ιστότοπους με pagination και infinite scroll.
- Δυνατότητα αποθήκευσης των εργασιών scraping για να μπορεί ο χρήστης να τις εκτελέσει ξανά, χωρίς να χρειάζεται εκ νέου επιλογή των δεδομένων.
- Δημιουργία API endpoint μετά την εκτέλεση μιας εργασίας scraping. Ο χρήστης μπορεί να καλεί το URL και να λαμβάνει τα δεδομένα που συλλέχθηκαν την τελευταία φορά ή να κάνει real-time εξαγωγή των δεδομένων.
- Εξαγωγή δεδομένων από ιστότοπους που χρειάζονται login. Στην περίπτωση αυτή θα πρέπει να εγκατασταθεί και η επέκταση EditThisCookie για την ανάκτηση των cookies. Ωστόσο, ορισμένοι ιστότοποι δεν χρησιμοποιούν cookies για τον έλεγχο ταυτότητας του χρήστη και επομένως αυτή η μέθοδος μπορεί να μην λειτουργεί πάντα.
- Αυτόματη εκτέλεση των εργασιών scraping με δυνατότητα προγραμματισμού από τον χρήστη.
- Λίστα με έτοιμα scraping tasks για την εξαγωγή δεδομένων από δημοφιλείς ιστοσελίδες και εφαρμογές [36].



Εικόνα 14. Simplescraper [36]

4.3.5 Σύγκριση των επεκτάσεων για προγράμματα περιήγησης

Στον Πίνακα 10 γίνεται σύγκριση των desktop εφαρμογών για web scraping που παρουσιάστηκαν προηγουμένως.

Πίνακας 10. Σύγκριση των browser extensions για web scraping

	Webscraper.io	**Instant Data Scraper**	**Data Miner**	**Simplescraper**
Browser	Chrome, Firefox	Chrome, Edge	Chrome, Edge	Chrome
Αυτόματη εναλλαγή IP	X	-	-	X
Επιλογή δεδομένων	Point-and-click	Point-and-click	Point-and-click	Point-and-click, CSS selectors
Εξαγωγή δεδομένων	Excel, CSV, JSON	Excel, CSV	Excel, CSV	CSV, JSON
Υποστήριξη API	X	-	-	X
Προγραμματισμός των tasks	X	-	X	X
Integrations	Dropbox, Google Sheets, Amazon S3	-	Google Sheets	Google Sheets, Airtable, Zapier, Make (Integromat)

5. ΚΑΛΕΣ ΠΡΑΚΤΙΚΕΣ WEB SCRAPING

Στο κεφάλαιο αυτό γίνεται αναφορά στη διαδικασία που πρέπει να ακολουθείται κατά την εξαγωγή δεδομένων από έναν ιστότοπο και σε ζητήματα που θα πρέπει να λαμβάνονται υπόψη όπως η προσαρμογή των κεφαλίδων HTTP, η διαχείριση των cookies, η υποβολή φορμών HTML, η διαχείριση ιστότοπων με JavaScript κ.ά.

5.1 Προσαρμογή των κεφαλίδων HTTP

Οι κεφαλίδες HTTP (HTTP headers) είναι λίστες χαρακτηριστικών ή προτιμήσεων που αποστέλλονται κάθε φορά που γίνεται αίτημα σε έναν διακομιστή. Το HTTP ορίζει δεκάδες τύπους κεφαλίδων, ωστόσο οι παρακάτω επτά κεφαλίδες είναι αυτές που χρησιμοποιούν οι κυριότεροι browsers κατά την εκκίνηση οποιασδήποτε σύνδεσης. Οι τιμές των κεφαλίδων δίνονται ως παράδειγμα.

`Host`	`https://www.google.com/`
`Connection`	`keep-alive`
`Accept`	`text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8`
`User-Agent`	`Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/ 39.0.2171.95 Safari/537.36`
`Referrer`	`https://www.google.com/`
`Accept-Encoding`	`gzip, deflate, sdch`
`Accept-Language`	`en-US,en;q=0.8`

Οι κεφαλίδες HTTP που αποστέλλονται από έναν συνηθισμένο scraper της Python που χρησιμοποιεί τη βιβλιοθήκη urllib είναι οι εξής:

`Accept-Encoding`	`Identity`
`User-Agent`	`Python-urllib/3.4`

Στην περίπτωση αυτή, ο διαχειριστής του ιστότοπου μπορεί εύκολα να διαπιστώσει ότι πρόκειται για bot και να αποκλείσει τη συγκεκριμένη σύνδεση. Οι ιστότοποι έχουν τη δυνατότητα να ελέγχουν αν τα αιτήματα γίνονται από απλούς χρήστες ή bot εξετάζοντας οποιαδήποτε από τις κεφαλίδες HTTP, αλλά η σημαντικότερη από αυτές είναι η `User-Agent`. Γι' αυτό θα πρέπει να αλλάζουμε την τιμή της κεφαλίδας `User-Agent` με μια τιμή που παραπέμπει σε πραγματικό χρήστη. Οι κεφαλίδες HTTP μπορούν να προσαρμοστούν κατάλληλα με βιβλιοθήκες όπως η requests. Σε άλλους ιστότοπους ίσως απαιτείται και ο ορισμός της κεφαλίδας `Accept-Language` για να αποκτήσουμε πρόσβαση στο περιεχόμενο [42].

Πολλοί ιστότοποι εμφανίζουν το περιεχόμενό τους μεταφρασμένο σε διαφορετική γλώσσα με βάση τις γλωσσικές προτιμήσεις που δηλώνονται στις κεφαλίδες. Για παράδειγμα, αλλάζοντας την κεφαλίδα `Accept-Language:en-US` σε `Accept-Language:fr` ο χρήστης μπορεί να δει το περιεχόμενο στη γαλλική γλώσσα, εφόσον υποστηρίζεται από τον ιστότοπο. Οι κεφαλίδες χρησιμοποιούνται, επίσης, για την αλλαγή της μορφής του περιεχομένου. Για παράδειγμα, στις κινητές συσκευές μπορεί να εμφανίζεται μια διαφορετική έκδοση του ιστότοπου χωρίς διαφημιστικά πλαίσια, flash κλπ. Αλλάζοντας την κεφαλίδα `User-Agent` ώστε να υποδηλώνει χρήστη κινητού τηλεφώνου, όπως φαίνεται παρακάτω, η εξαγωγή δεδομένων θα μπορούσε να γίνει ευκολότερη.

```
User-Agent : Mozilla/5.0 (iPhone; CPU iPhone OS 7_1_2 like Mac OS X)
AppleWebKit/537.51.2 (KHTML, like Gecko) Version/7.0 Mobile/11D257
Safari/9537.53
```

5.2 Διαχείριση των cookies

Οι περισσότεροι σύγχρονοι ιστότοποι χρησιμοποιούν cookies για να παρακολουθούν αν οι χρήστες είναι συνδεδεμένοι. Μόλις ο ιστότοπος επαληθεύσει τα στοιχεία σύνδεσης του χρήστη, τα αποθηκεύει στο cookie του προγράμματος περιήγησης. Το cookie συνήθως περιέχει ένα token που παράγεται από τον server, χρονικό όριο λήξης (timeout) και πληροφορίες παρακολούθησης του χρήστη. Στη συνέχεια ο ιστότοπος χρησιμοποιεί αυτό το cookie σαν αποδεικτικό της ταυτότητας του χρήστη, το οποίο και ελέγχει σε κάθε σελίδα που επισκέπτεται ο χρήστης.

Παρόλο που τα cookies είναι χρήσιμα για τους web developers, μπορεί να αποτελέσουν εμπόδιο για όσους ασχολούνται με την εξαγωγή δεδομένων ιστού. Για παράδειγμα, αν γίνει υποβολή μιας φόρμας σύνδεσης με κάποιο εργαλείο web scraping και δεν αποθηκευτεί το cookie που επιστρέφει η φόρμα, η επόμενη σελίδα που θα επισκεφτούμε θα λειτουργεί σαν να μην έχει προηγηθεί login.

Η διαχείριση των cookies μπορεί να γίνει εύκολα με τη βιβλιοθήκη requests, όπως φαίνεται στον Κώδικα 23. Στο παράδειγμα αυτό, αποστέλλονται οι παράμετροι σύνδεσης στη σελίδα welcome.php, γίνεται ανάκτηση των cookies από τα αποτελέσματα του request, τυπώνονται τα cookies για επαλήθευση και, τέλος, αποστέλλονται στη σελίδα profile.php με το όρισμα cookies.

```
import requests
params = {'username': 'Ryan', 'password': 'password'}
r = requests.post('http://pythonscraping.com/pages/cookies/welcome.php',
params)
print('Cookie is set to:')
print(r.cookies.get_dict())
print('Going to profile page...')
r = requests.get('http://pythonscraping.com/pages/cookies/profile.php',
cookies=r.cookies)
print(r.text)
```

Κώδικας 23. Διαχείριση των cookies με τη βιβλιοθήκη requests (όρισμα cookies)

Η μέθοδος αυτή λειτουργεί καλά σε απλούς ιστότοπους, ωστόσο υπάρχουν και ιστότοποι που αλλάζουν συχνά τα cookies χωρίς προειδοποίηση. Στις περιπτώσεις αυτές μπορεί να χρησιμοποιηθεί η κλάση `Session` της βιβλιοθήκης requests, όπως φαίνεται στον Κώδικα 24. Σε αυτό το παράδειγμα το αντικείμενο `Session` κρατά τις πληροφορίες της συνόδου, όπως τα cookies και τις κεφαλίδες, καθώς και πληροφορίες σχετικά με τα πρωτόκολλα που μπορεί να εκτελούνται πάνω από το HTTP (πχ HTTPAdapters) [42].

```
import requests

session = requests.Session()
params = {'username': 'username', 'password': 'password'}
s = session.post('http://pythonscraping.com/pages/cookies/welcome.php',
params)
print('Cookie is set to:')
print(s.cookies.get_dict())
print('Going to profile page...')
s = session.get('http://pythonscraping.com/pages/cookies/profile.php')
print(s.text)
```

Κώδικας 24. Διαχείριση των cookies με τη βιβλιοθήκη requests (χρήση session)

5.3 Υποβολή φόρμας HTML

Οι φόρμες HTML και κυρίως οι φόρμες που σχετίζονται με τη δημιουργία λογαριασμών και τη σύνδεση των χρηστών αποτελούν σημαντική απειλή για την ασφάλεια και το υπολογιστικό overhead, αν είναι εκτεθειμένες στην αλόγιστη χρήση από bots. Γι' αυτό το λόγο πολλοί διαχειριστές ιστοσελίδων προσπαθούν να περιορίσουν την πρόσβαση από φόρμες σύνδεσης και λαμβάνουν μέτρα προστασίας που μπορεί να αποτελέσουν εμπόδιο για τα προγράμματα web scraping. Παρακάτω περιγράφεται τι θα πρέπει να προσέχουμε κατά την εξαγωγή δεδομένων από ιστότοπους με φόρμες HTML.

5.3.1 Κρυφά πεδία σε φόρμες HTML

Όταν αναφερόμαστε σε "κρυφά" πεδία στις φόρμες HTML εννοούμε πεδία η τιμή των οποίων είναι ορατή για το πρόγραμμα περιήγησης αλλά αόρατη για τον χρήστη (εκτός αν κοιτάξει τον πηγαίο κώδικα της σελίδας). Στην Εικόνα 14 φαίνεται ένα παράδειγμα κρυφών πεδίων από τη σελίδα σύνδεσης του Facebook. Παρόλο που η φόρμα έχει μόνο τρία ορατά πεδία (Username, Password, και το κουμπί Submit), στην πραγματικότητα μεταφέρει στον διακομιστή περισσότερες πληροφορίες.

Εικόνα 15. Κρυφά πεδία στη σελίδα σύνδεσης του Facebook [42]

Τα κρυφά πεδία χρησιμοποιούνται για να εμποδίσουν ενέργειες web scraping με δύο τρόπους. Ο πρώτος είναι ότι το κρυφό πεδίο συμπληρώνεται με μια τυχαία μεταβλητή, την οποία ο διακομιστής αναμένει στη σελίδα επεξεργασίας της φόρμας. Αν η μεταβλητή δεν εμφανίζεται, ο διακομιστής μπορεί εύλογα να υποθέσει ότι η αποστολή των δεδομένων δεν έγινε μέσω της φόρμας, αλλά έγινε από bot απευθείας στη σελίδα επεξεργασίας της φόρμας. Για να αποφύγει κανείς το συγκεκριμένο μέτρο προστασίας, θα πρέπει πρώτα να εξάγει την τυχαία μεταβλητή από το κρυφό πεδίο και έπειτα να κάνει αίτημα POST στη σελίδα επεξεργασίας της φόρμας.

Η δεύτερη μέθοδος είναι ένα είδος “honeypot”. Αν η φόρμα περιέχει ένα κρυφό πεδίο με το όνομα Username ή Email Address, υπάρχει η πιθανότητα ένα κακογραμμένο bot να συμπληρώσει το πεδίο και να προσπαθήσει να το αποστείλει παρόλο που είναι κρυφό για τον χρήστη. Κρυφά πεδία που αποστέλλονται συμπληρωμένα με τιμές (ή με τιμές διαφορετικές από τις προεπιλεγμένες) αγνοούνται και ο χρήστης μπορεί ακόμη και να αποκλειστεί από τον ιστότοπο.

Γενικότερα, αν η φόρμα διαθέτει κρυφά πεδία που περιέχουν συχνά μεγάλες και τυχαίες συμβολοσειρές, ο διακομιστής πιθανότατα θα ελέγξει την παρουσία τους κατά την υποβολή της φόρμας. Επίσης, ενδέχεται να γίνονται κι άλλοι έλεγχοι για να διαπιστωθεί αν οι μεταβλητές της φόρμας έχουν χρησιμοποιηθεί μόνο μια φορά ή αν έχουν δημιουργηθεί πρόσφατα (διότι υπάρχει η πιθανότητα οι μεταβλητές να είναι αποθηκευμένες σε script και να χρησιμοποιούνται ξανά και ξανά) ή και τα δύο [42].

5.3.2 Αποφυγή των honeypots

Αν και το CSS είναι χρήσιμο για τη διάκριση των χρήσιμων πληροφοριών (με βάση το id ή το class, για παράδειγμα) σε έναν ιστότοπο, ορισμένες φορές μπορεί να αποτελέσει παγίδα για τα προγράμματα web scraping. Αν ένα πεδίο σε μια φόρμα HTML έχει κρυφτεί με κανόνα CSS, ο μέσος χρήστης που επισκέπτεται τον ιστότοπο δεν θα μπορεί να συμπληρώσει το πεδίο καθώς

δεν εμφανίζεται στον browser. Αν όμως το πεδίο συμπληρωθεί, είναι πολύ πιθανό ότι κάποιο bot υπέβαλε τη φόρμα και το αίτημα απορρίπτεται.

Το ίδιο ισχύει όχι μόνο για τις φόρμες αλλά και για τους συνδέσμους, τις εικόνες, τα αρχεία και για οποιοδήποτε άλλο στοιχείο του ιστότοπου που μπορεί να διαβαστεί από ένα bot, αλλά είναι κρυφό για τον μέσο χρήστη. Για παράδειγμα, η μετάβαση σε μια σελίδα με κρυφό σύνδεσμο μπορεί εύκολα να ενεργοποιήσει ένα script στον πλευρά του διακομιστή που θα μπλοκάρει την IP του χρήστη, θα αποσυνδέσει τον χρήστη από τον ιστότοπο ή θα προβεί σε κάποια άλλη ενέργεια για να αποτρέψει περαιτέρω πρόσβαση.

Στον Κώδικα 25 φαίνεται ο πηγαίος κώδικας μιας σελίδας που περιέχει δύο συνδέσμους, ο ένας εκ των οποίων είναι κρυμμένος με CSS και ο άλλος είναι ορατός. Στη σελίδα υπάρχει, επίσης, μια φόρμα με δύο κρυφά πεδία.

```
<html>
   <head>
      <title>A bot-proof form</title>
   </head>
   <style>
      body {
      overflow-x:hidden;
      }
      .customHidden {
      position:absolute;
      right:50000px;
      }
   </style>
   <body>
      <h2>A bot-proof form</h2>
      <a href=
         "http://pythonscraping.com/dontgohere" style="display:none;">Go
here!</a>
      <a href="http://pythonscraping.com">Click me!</a>
      <form>
         <input type="hidden" name="phone"
value="valueShouldNotBeModified"/>
         <p/>
            <input type="text" name="email" class="customHidden"
               value="intentionallyBlank"/>
         <p/>
            <input type="text" name="firstName"/>
         <p/>
            <input type="text" name="lastName"/>
         <p/>
            <input type="submit" value="Submit"/>
         <p/>
      </form>
   </body>
</html>
```

Κώδικας 25. Παράδειγμα σελίδας με κρυφά πεδία

Στο συγκεκριμένο παράδειγμα, τα στοιχεία HTML είναι κρυμμένα από τον χρήστη με τους εξής τρόπους:

- Ο πρώτος σύνδεσμος είναι κρυμμένος με έναν απλό κανόνα CSS (display:none).
- Το πεδίο για το τηλέφωνο είναι ένα κρυφό πεδίο εισαγωγής (type="hidden").
- Το πεδίο για το email δεν είναι ορατό στον χρήστη γιατί έχει μετακινηθεί 50.000 pixel προς τα δεξιά με αποτέλεσμα να βρίσκεται εκτός οθόνης.

Η επίσκεψη κρυφών συνδέσμων δεν είναι συνηθισμένη, ωστόσο η υποβολή φόρμας με κρυφά πεδία και προσυμπληρωμένες τιμές είναι μια συνηθισμένη περίπτωση που θα πρέπει να ελέγχεται κατά τη συγγραφή προγραμμάτων web scraping.

5.4 Διαχείριση ιστότοπων με JavaScript

Οι γλώσσες σεναρίων που εκτελούνται στην πλευρά του client (ή αλλιώς client-side scripting languages) είναι γλώσσες προγραμματισμού που τρέχουν στο πρόγραμμα περιήγησης και όχι στον διακομιστή. Οι γλώσσες που χρησιμοποιούνται συνήθως είναι η ActionScript (που χρησιμοποιείται σε εφαρμογές Flash) και η JavaScript. Η ActionScript συναντάται σήμερα πολύ λιγότερο από ότι συνέβαινε 10 χρόνια πριν, ενώ η JavaScript είναι η πιο κοινή και καλά υποστηριζόμενη γλώσσα scripting. Μπορεί να χρησιμοποιηθεί για τη συλλογή πληροφοριών σχετικά με τη δραστηριότητα του χρήστη (user tracking), για την υποβολή φορμών χωρίς επαναφόρτωση της σελίδας, για την ενσωμάτωση πολυμέσων, μέχρι και για τη δημιουργία ολόκληρων διαδικτυακών παιχνιδιών. Ακόμα και φαινομενικά απλές ιστοσελίδες μπορεί να περιέχουν κώδικα JavaScript, ο οποίος περικλείεται σε ετικέτες script στον πηγαίο κώδικα της σελίδας, όπως για παράδειγμα:

```
<script>
alert("This creates a pop-up using JavaScript");
</script>
```

Παλιότερα ο μόνος τρόπος επικοινωνίας με έναν διακομιστή ήταν η αποστολή ενός αιτήματος HTTP και η ανάκτηση μιας νέας σελίδας. Ωστόσο, στους σύγχρονους ιστότοπους είναι δυνατή η υποβολή μιας φόρμας ή η ανάκτηση πληροφοριών από τον διακομιστή χωρίς επαναφόρτωση της σελίδας κι αυτό γίνεται με τη χρήση AJAX. Η AJAX δεν είναι γλώσσα προγραμματισμού, αλλά ένα σύνολο τεχνολογιών που χρησιμοποιούνται για την αποστολή και λήψη πληροφοριών από τον διακομιστή χωρίς αίτημα για νέα σελίδα.

Η DHTML (dynamic HTML), όπως και η AJAX, είναι μια ομάδα τεχνολογιών που χρησιμοποιούνται για τον ίδιο σκοπό. Η DHTML είναι κώδικας HTML ή/και γλώσσα CSS που αλλάζει όταν τα script στην πλευρά του client αλλάζουν στοιχεία της HTML. Για παράδειγμα, ένα κουμπί μπορεί να

εμφανίζεται μόνο μετά από μετακίνηση του κέρσορα, ένα χρώμα φόντου μπορεί να αλλάζει μετά από κλικ ή ένα αίτημα AJAX μπορεί να ενεργοποιεί την εμφάνιση ενός νέου μπλοκ περιεχομένου. Παρόλο που η λέξη “δυναμική” σχετίζεται με κίνηση ή αλλαγή, η ύπαρξη διαδραστικών στοιχείων HTML, κινούμενων εικόνων ή ενσωματωμένων πολυμέσων δεν καθιστά απαραίτητα μια σελίδα DHTML, αν και μπορεί να μοιάζει δυναμική. Αντίθετα, υπάρχουν σελίδες στο διαδίκτυο που μπορεί να μοιάζουν στατικές αλλά τρέχουν διεργασίες DHTML στο παρασκήνιο που βασίζονται στη χρήση JavaScript για τον χειρισμό της HTML και του CSS.

Κατά την εξαγωγή δεδομένων ιστού μπορεί να συναντήσει κανείς περιπτώσεις στις οποίες το περιεχόμενο που βλέπει στο πρόγραμμα περιήγησης δεν ταιριάζει με τον κώδικα HTML που λαμβάνει μέσω του scraper. Επίσης, μια σελίδα μπορεί φαινομενικά να ανακατευθύνει σε μια νέα σελίδα, αλλά η διεύθυνση URL της σελίδας δεν αλλάζει. Και στις δύο αυτές περιπτώσεις ο scraper δεν μπορεί να εξάγει τα δεδομένα, γιατί αποτυγχάνει να εκτελέσει την JavaScript.

Επίσης, η σελίδα μπορεί να χρησιμοποιεί AJAX ή DHTML για να αλλάζει/φορτώνει το περιεχόμενο. Στις περιπτώσεις αυτές υπάρχουν δύο επιλογές: είτε να γίνει συλλογή των δεδομένων απευθείας από την JavaScript είτε να χρησιμοποιηθεί κάποιο package της Python για την εκτέλεση της JavaScript, και στη συνέχεια να γίνει εξαγωγή των δεδομένων από τον ιστότοπο, όπως αυτός εμφανίζεται στο πρόγραμμα περιήγησης. Για τον σκοπό αυτό μπορούν να χρησιμοποιηθούν packages όπως τα Selenium, Playwright και Pyppeteer που περιγράφονται στο Κεφάλαιο 3.

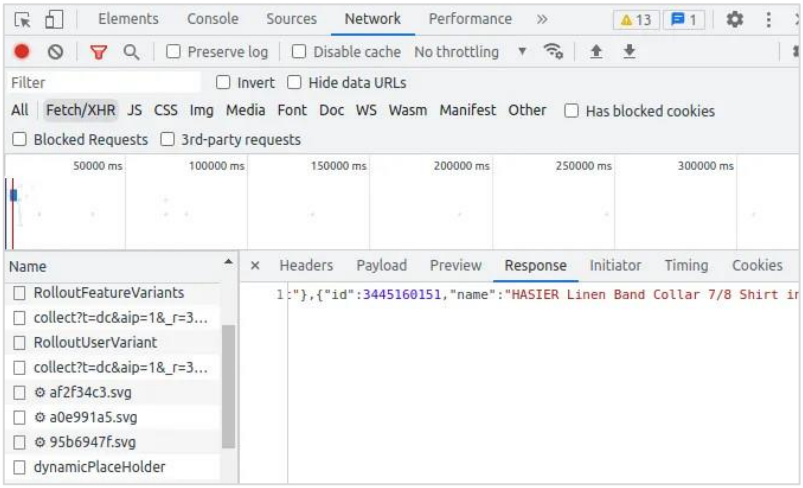
5.5 Αναζήτηση των “κρυμμένων” API

Όταν θέλουμε να εξάγουμε δεδομένα από έναν ιστότοπο, θα πρέπει αρχικά να διερευνήσουμε τον τρόπο με τον οποίο φορτώνει ο ιστότοπος τα δεδομένα. Αν τα δεδομένα που μας ενδιαφέρουν φορτώνονται με JavaScript, χρησιμοποιούμε έναν headless browser που θα εκτελέσει την JavaScript και στη συνέχεια εξετάζουμε τον κώδικα HTML και εξάγουμε τα δεδομένα.

Πολλοί σύγχρονοι ιστότοποι διαθέτουν backend APIs που συνδέονται με τις frontend εφαρμογές. Αν ο ιστότοπος-στόχος διαθέτει ένα τέτοιο “κρυμμένο” API για τα δεδομένα που μας ενδιαφέρουν, μπορούμε να το χρησιμοποιήσουμε απευθείας. Με τον τρόπο αυτό μειώνεται η απαιτούμενη υπολογιστική ισχύς, καθώς αποφεύγεται η εκτέλεση ενός headless browser. Επίσης, τα API έχουν γρήγορο χρόνο απόκρισης, επομένως μπορούμε να υλοποιήσουμε έναν web scraper με μεγαλύτερη ταχύτητα.

Για να ελέγξουμε αν ένας ιστότοπος διαθέτει API, ανοίγουμε το παράθυρο Developer Tools στον browser και επιλέγουμε την καρτέλα Network. Μπορούμε να φιλτράρουμε τα αιτήματα επιλέγοντας XHR. Το XHR είναι ένα JavaScript API για τη δημιουργία αιτημάτων AJAX. Διαθέτει μεθόδους για

την αποστολή αιτημάτων μεταξύ του browser και του server. Στη συνέχεια, φορτώνουμε ξανά τη σελίδα και παρακολουθούμε τα αιτήματα. Κάνοντας κλικ σε ένα αίτημα μπορούμε να δούμε τις κεφαλίδες αιτήματος και απόκρισης (request/response headers), το payload του αιτήματος και την απόκριση. Αν η απόκριση περιλαμβάνει τα δεδομένα που θέλουμε να συλλέξουμε (βλ. Εικόνα 16), κάνουμε δεξί κλικ πάνω στο όνομα του αιτήματος και επιλέγουμε “Copy as cURL”. Με τον τρόπο αυτό λαμβάνουμε το endpoint του API, τις κεφαλίδες και το σώμα του αιτήματος. Τέλος, μπορούμε να χρησιμοποιήσουμε μια βιβλιοθήκη, όπως η requests, για να κάνουμε αίτημα στο API και να λάβουμε τα δεδομένα σε μορφή JSON.



Εικόνα 16. Παράδειγμα ιστότοπου με backend API

5.6 Ανάγνωση του αρχείου robots.txt

Το αρχείο robots.txt βρίσκεται στον root κατάλογο κάθε ιστότοπου και περιέχει οδηγίες σχετικά με το ποιες σελίδες επιτρέπεται να επισκέπτονται τα bot. Τα bot που επιλέγουν να ακολουθήσουν τις οδηγίες του robots.txt, ανακτούν το αρχείο και διαβάζουν τις οδηγίες πριν ξεκινήσουν την προσπέλαση του ιστότοπου. Αν ο ιστότοπος δε διαθέτει αρχείο robots.txt, θεωρείται ότι ο κάτοχος του ιστότοπου δεν έχει θέσει κανένα περιορισμό στην προσπέλαση [43]. Το αρχείο robots.txt είναι ιδιαίτερα σημαντικό για τα bot των μηχανών αναζήτησης, όπως η Google, καθώς ορίζει ποια μέρη του ιστότοπου πρέπει ή δεν πρέπει να προσπελάσουν, προκειμένου να κατατάξουν στη συνέχεια το περιεχόμενό του στα αποτελέσματα αναζήτησης, μια διαδικασία που ονομάζεται indexing.

Η σύνταξη του αρχείου robots.txt γίνεται με βάση το πρότυπο Robots Exclusion Protocol, το οποίο προτάθηκε το 1994 από τον Martijn Koster. Όπως ισχύει στην Python και σε άλλες γλώσσες, τα σχόλια ξεκινούν με το σύμβολο #, τελειώνουν με νέα γραμμή και μπορούν να βρίσκονται οπουδήποτε μέσα στο αρχείο. Η πρώτη γραμμή του αρχείου, εκτός από τυχόν σχόλια, ξεκινά με `User-agent` προσδιορίζοντας το bot για το οποίο ισχύουν οι κανόνες που ακολουθούν. Στην

συνέχεια ορίζονται οι κανόνες με `Allow` ή `Disallow`, ανάλογα με το αν επιτρέπεται ή όχι στο bot να επισκέπτεται το συγκεκριμένο τμήμα του ιστότοπου. Ο αστερίσκος (*) μπορεί να χρησιμοποιηθεί για να περιγράψει είτε έναν `User-agent` είτε μια διεύθυνση URL [42].

Παρακάτω φαίνεται ένα παράδειγμα με πολλά bot καθένα με τους δικούς του κανόνες [43]:

```
User-agent: Googlebot          # all Google services
Disallow: /private/            # disallow this directory

User-agent: googlebot-news     # only the news service
Disallow: /                    # disallow everything

User-agent: *                  # any robot
Disallow: /something/          # disallow this directory
```

Αν υπάρχουν δύο κανόνες που έρχονται σε αντίθεση, ο τελευταίος κανόνας έχει προτεραιότητα. Για παράδειγμα [42]:

```
User-agent: *
Disallow: *

User-agent: Googlebot
Allow: *
Disallow: /private
```

Σε αυτή την περίπτωση κανένα bot δεν επιτρέπεται να προσπελάσει τον ιστότοπο, εκτός από το Googlebot το οποίο μπορεί να έχει πρόσβαση οπουδήποτε εκτός από τον κατάλογο /private [42].

Η συμμόρφωση με τις οδηγίες του robots.txt επαφίεται στον χρήστη. Πολλά bot όχι μόνο δεν ακολουθούν το robots.txt αλλά ξεκινούν την προσπέλαση από τα τμήματα του ιστότοπου στα οποία δεν επιτρέπεται η πρόσβαση. Τέτοιου είδους bot είναι, για παράδειγμα, τα bot που συλλέγουν email, τα spam bot, τα κακόβουλα προγράμματα ή τα bot που σκανάρουν για ευπάθειες όσον αφορά την ασφάλεια [43].

Αρκετές βιβλιοθήκες web scraping υποστηρίζουν συμμόρφωση με τους κανόνες του αρχείου robots.txt, αλλά στις περισσότερες περιπτώσεις αυτή είναι μια προεπιλεγμένη ρύθμιση που μπορεί να αλλάξει [42]. Για παράδειγμα, στη βιβλιοθήκη Scrapy υπάρχει η επιλογή `'ROBOTSTXT_OBEY': True` στο αρχείο settings.py που μπορούμε να αλλάξουμε σε False, αν θέλουμε να παρακάμψουμε το robots.txt ενός ιστότοπου.

Μια καλή πρακτική είναι να σεβόμαστε τους περιορισμούς που θέτει ο κάτοχος του ιστότοπου και να ακολουθούμε τις οδηγίες που αναγράφονται στο αρχείο robots.txt.

5.7 Γενικές οδηγίες

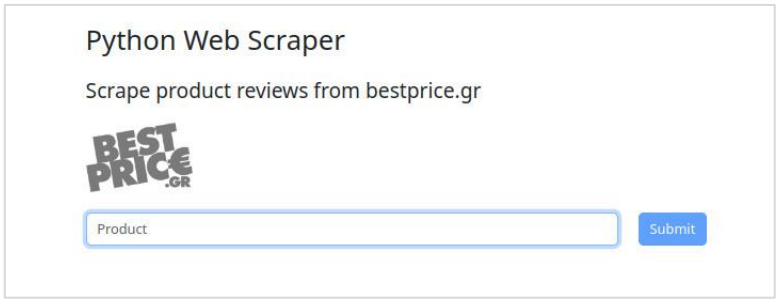
- Αν η σελίδα που λαμβάνουμε από τον διακομιστή είναι κενή ή λείπουν πληροφορίες ή δεν είναι αυτή που αναμένουμε (ή βλέπουμε στο δικό μας πρόγραμμα περιήγησης), αυτό πιθανώς να οφείλεται στην εκτέλεση κώδικα JavaScript για τη δημιουργία της σελίδας.
- Αν πρέπει να υποβληθεί μια φόρμα ή ένα αίτημα POST, θα πρέπει να ελέγξουμε αν στέλνουμε όλες τις πληροφορίες που αναμένει ο ιστότοπος και στην κατάλληλη μορφή. Με τη βοήθεια ενός εργαλείου, όπως ο Chrome Inspector, μπορούμε να δούμε πως αποστέλλεται ένα αίτημα POST στον ιστότοπο, και να βεβαιωθούμε ότι ένα “οργανικό” αίτημα μοιάζει με τα αιτήματα που στέλνει ο scraper που σχεδιάσαμε.
- Αν προσπαθούμε να κάνουμε log in σε έναν ιστότοπο και διαπιστώνουμε ότι δεν μπορούμε να διατηρήσουμε τη σύνδεση, θα πρέπει να ελέγξουμε τα cookies. Τα cookies θα πρέπει να διατηρούνται μεταξύ των σελίδων και να αποστέλλονται στον ιστότοπο με κάθε νέο αίτημα.
- Αν λαμβάνουμε σφάλματα HTTP, κυρίως σφάλματα με κωδικό 403, πιθανότατα ο ιστότοπος θεωρεί πως η συγκεκριμένη διεύθυνση IP είναι bot και σταματά να δέχεται άλλα αιτήματα. Σε αυτή την περίπτωση θα πρέπει είτε να περιμένουμε μέχρι η IP να αφαιρεθεί από τη λίστα αποκλεισμού είτε να αλλάξουμε IP.

Μερικές καλές πρακτικές που μπορούμε να ακολουθούμε για να αποφύγουμε τον αποκλεισμό της IP είναι οι εξής:

- Οι scrapers που γράφουμε δεν θα πρέπει να περιηγούνται στον ιστότοπο με μεγάλη ταχύτητα. Το γρήγορο scraping είναι κακή πρακτική που επιβαρύνει τους διακομιστές και μπορεί να οδηγήσει ακόμα και σε νομικά προβλήματα.
- Θα πρέπει να στέλνουμε κατάλληλες κεφαλίδες HTTP με κάθε αίτημα, διότι πολλοί ιστότοποι μπλοκάρουν αιτήματα με κεφαλίδες που υποδηλώνουν την ύπαρξη bot. Ένας εύκολος τρόπος είναι η αντιγραφή των κεφαλίδων που βλέπουμε στο δικό μας πρόγραμμα περιήγησης.
- Θα πρέπει να βεβαιωθούμε ότι δεν κάνουμε κλικ και δεν έχουμε πρόσβαση σε στοιχεία που είναι μη προσβάσιμα για τον μέσο χρήστη που επισκέπτεται τον ιστότοπο μέσω browser.

6. ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ ΣΕ ΓΛΩΣΣΑ PYTHON

Στο πλαίσιο της διπλωματικής εργασίας αναπτύχθηκε μια web εφαρμογή η οποία εξάγει κριτικές προϊόντων από τον ιστότοπο σύγκρισης τιμών bestprice.gr. Η εφαρμογή διαθέτει ένα πολύ απλό UI με φόρμα αναζήτησης, στην οποία ο χρήστης εισάγει το προϊόν που τον ενδιαφέρει (βλ. Εικόνα 17). Η εφαρμογή επιστρέφει τις κριτικές του προϊόντος σε μορφή πίνακα και δίνει τη δυνατότητα στον χρήστη να κατεβάσει τα αποτελέσματα σε αρχείο csv.



Εικόνα 17. UI εφαρμογής

6.1 Εργαλεία που χρησιμοποιήθηκαν

Παρακάτω παρουσιάζονται τα εργαλεία που χρησιμοποιήθηκαν για την δημιουργία της εφαρμογής. Η εξαγωγή των δεδομένων έγινε με τις βιβλιοθήκες scrapy και scrapy-playwright. Στη συνέχεια χρησιμοποιήθηκε η βιβλιοθήκη scrapyrt για τον έλεγχο της αράχνης μέσω αιτημάτων HTTP. Η web εφαρμογή αναπτύχθηκε με το framework flask και οι HTML σελίδες δημιουργήθηκαν με Bootstrap. Τέλος, χρησιμοποιήθηκε η βιβλιοθήκη pandas για την εμφάνιση των αποτελεσμάτων σε μορφή πίνακα.

6.1.1 Scrapy

Το Scrapy είναι ένα framework για ανίχνευση (crawling) ιστοσελίδων και εξαγωγή δομημένων δεδομένων, το οποίο μπορεί να χρησιμοποιηθεί σε ένα ευρύ φάσμα εφαρμογών. Στην ενότητα 3.2.7 αναφέρονται περισσότερες πληροφορίες για το Scrapy.

6.1.2 Scrapy-playwright

Το scrapy-playwright είναι download handler του Scrapy που εκτελεί αιτήματα HTTP χρησιμοποιώντας τη βιβλιοθήκη Playwright για τη γλώσσα Python. Μπορεί να χρησιμοποιηθεί για

τον χειρισμό σελίδων με JavaScript, ενώ διατηρεί το workflow του Scrapy (χωρίς να παρεμβαίνει στον προγραμματισμό των αιτημάτων, στην επεξεργασία των στοιχείων κλπ) [37].

Το scrapy-playwright προϋποθέτει Python >= 3.7, Scrapy >= 2.0 (!= 2.4.0) και Playwright >= 1.15, και δεν λειτουργεί σε Windows. Η εγκατάσταση του πακέτου γίνεται με την εντολή:

```
pip install scrapy-playwright
```

Το Playwright είναι dependency, επομένως εγκαθίσταται αυτόματα, ωστόσο μπορεί να χρειαστεί η εγκατάσταση των browser που θα χρησιμοποιηθούν, και αυτό γίνεται με την εντολή:

```
playwright install
```

Το Playwright κατεβάζει τα προγράμματα περιήγησης Chromium, Firefox και WebKit από προεπιλογή. Ωστόσο, ο χρήστης μπορεί να εγκαταστήσει ένα υποσύνολο των διαθέσιμων προγραμμάτων περιήγησης περνώντας τα ως ορίσματα κατά την εγκατάσταση, όπως για παράδειγμα:

```
playwright install firefox chromium
```

Για να χρησιμοποιηθεί το scrapy-playwright χρειάζεται να γίνουν οι παρακάτω ρυθμίσεις:

1. Θα πρέπει να αντικατασταθούν οι προεπιλεγμένοι download handlers http ή/και https.

```
DOWNLOAD_HANDLERS = {
   "http":"scrapy_playwright.handler.ScrapyPlaywrightDownloadHandler",
   "https":"scrapy_playwright.handler.ScrapyPlaywrightDownloadHandler",
}
```

2. Θα πρέπει να εγκατασταθεί ο asyncio-based Twisted reactor.

```
TWISTED_REACTOR = "twisted.internet.asyncioreactor.AsyncioSelectorReactor"
```

6.1.3 ScrapyRT

Το Scrapyrt (Scrapy realtime) είναι μια επέκταση ανοιχτού κώδικα για το Scrapy που δίνει στον χρήστη τη δυνατότητα να ελέγχει τις αράχνες (spiders) του Scrapy με αιτήματα HTTP. Όταν το Scrapyrt συνδεθεί με το project λειτουργεί σαν web service. Ο χρήστης υποβάλλει ένα request, το οποίο περιέχει το URL και το όνομα της αράχνης, και λαμβάνει τα δεδομένα σε μορφή JSON.

Η αρχιτεκτονική του Scrapyrt είναι πολύ απλή. Είναι ένας web server γραμμένος σε Python Twisted μαζί με ένα προσαρμοσμένο αντικείμενο Crawler του Scrapy. Το Twisted είναι ένα από τα πιο ισχυρά ασύγχρονα framework της Python, είναι κατάλληλο για ασύγχρονη ανίχνευση (crawling) και χρησιμοποιείται από το Scrapy για όλη την HTTP επικοινωνία. Με αυτό τον τρόπο διασφαλίζεται εύκολη ενσωμάτωση (integration) με το Scrapy [38].

Η εγκατάσταση του Scrapyrt γίνεται με την εντολή: `pip install scrapyrt`. Στη συνέχεια ο χρήστης μεταβαίνει στον κατάλογο που βρίσκεται το Scrapy project και τρέχει το Scrapyrt με την εντολή: `scrapyrt`

Αιτήματα GET και POST

Ο server τρέχει στην πόρτα 9080 από προεπιλογή. Το Scrapyrt υποστηρίζει το endpoint /crawl.json και τα αιτήματα μπορούν να γίνουν με δύο μεθόδους, GET ή POST. Ένα παράδειγμα αιτήματος GET είναι το εξής:

```
curl "localhost:9080/crawl.json?spider_name=toscrape-
css&url=http://quotes.toscrape.com/"
```

Οι παράμετροι `spider_name` και `url` είναι υποχρεωτικές, ενώ υπάρχει και ένα σύνολο προαιρετικών παραμέτρων τις οποίες μπορεί να προσθέσει ο χρήστης και είναι οι `callback`, `errback`, `max_requests`, `start_requests` και `crawl_args`. Όταν χρησιμοποιείται η παράμετρος `start_requests`, η παράμετρος `url` μπορεί να παραληφθεί.

Στην περίπτωση που χρησιμοποιηθεί η μέθοδος POST, το σώμα του αιτήματος θα πρέπει να περιέχει σε μορφή JSON τις απαραίτητες πληροφορίες για το αίτημα HTTP και το όνομα της αράχνης. Στο JSON κλειδί `request` μπορούν να χρησιμοποιηθούν όλες οι παράμετροι του αντικειμένου `Request` του Scrapy. Ένα παράδειγμα φαίνεται παρακάτω:

```
{
    "request": {
        "url": "http://www.target.com/p/-/A-13631176",
        "callback": "parse_product",
        "dont_filter": "True"
    },
    "spider_name": "target.com_products"
}
```

Το αντικείμενο JSON θα πρέπει υποχρεωτικά να περιέχει τα κλειδιά (keys) `request` και `spider_name`, και προαιρετικά το κλειδί `max_requests`. Το JSON αντικείμενο `request` θα πρέπει υποχρεωτικά να περιέχει το κλειδί `url`. Ένα παράδειγμα αιτήματος POST είναι το εξής:

```
curl localhost:9080/crawl.json \
    -d '{"request":{"url":"http://quotes.toscrape.com/"}, "spider_name":
"toscrape-css"}'
```

Απόκριση (Response)

Το endpoint /crawl.json επιστρέφει ένα αντικείμενο JSON. Τα πεδία του JSON μπορούν να διαφέρουν ανάλογα με το αν το αίτημα είναι επιτυχές ή όχι. Αν το αίτημα είναι επιτυχές, η απάντηση JSON περιέχει τα κλειδιά `status` (με την τιμή “ok”), `spider_name`, `stats`, `items` και `items_dropped`. Ένα τέτοιο παράδειγμα φαίνεται στον Κώδικα 26.

```
$ curl "http://localhost:9080/crawl.json?spider_name=toscrape-
css&url=http://quotes.toscrape.com/"
{
    "status": "ok"
    "spider_name": "toscrape-css",
    "stats": {
        "start_time": "2019-12-06 13:01:31",
        "finish_time": "2019-12-06 13:01:35",
        "finish_reason": "finished",
        "downloader/response_status_count/200": 10,
        "downloader/response_count": 11,
        "downloader/response_bytes": 24812,
        "downloader/request_method_count/GET": 11,
        "downloader/request_count": 11,
        "downloader/request_bytes": 2870,
        "item_scraped_count": 100,
        "log_count/DEBUG": 111,
        "log_count/INFO": 9,
        "response_received_count": 11,
        "scheduler/dequeued": 10,
        "scheduler/dequeued/memory": 10,
        "scheduler/enqueued": 10,
        "scheduler/enqueued/memory": 10,
    },
    "items": [
        {
            "text": ...,
            "author": ...,
            "tags": ...
        },
        ...
    ],
    "items_dropped": [],
}
```

Κώδικας 26. Scrapyrt: απόκριση επιτυχημένου request

Αν το αίτημα αποτύχει, η απάντηση JSON περιέχει τα κλειδιά status (με την τιμή “error”), code και message. Ένα παράδειγμα φαίνεται στον Κώδικα 27.

```
$ curl
"http://localhost:9080/crawl.json?spider_name=foo&url=http://quotes.toscrape
.com/"
{
    "status": "error"
    "code": 404,
    "message": "Spider not found: foo",
}
```

Κώδικας 27. Scrapyrt: απόκριση αποτυχημένου request

6.1.4 Flask

Το Flask είναι ένα micro web framework γραμμένο σε Python. Χαρακτηρίζεται microframework, διότι δεν απαιτεί συγκεκριμένα εργαλεία ή βιβλιοθήκες, δεν διαθέτει επίπεδο αφαίρεσης βάσης δεδομένων (database abstraction layer), επικύρωση φόρμας (form validation) κλπ. Ωστόσο, το Flask υποστηρίζει επεκτάσεις που μπορούν να προσθέσουν στις εφαρμογές λειτουργίες σαν να είχαν υλοποιηθεί στο ίδιο το Flask. Υπάρχουν επεκτάσεις για ORM (object-relational mapping), επικύρωση φόρμας, χειρισμό upload, διάφορες ανοιχτές τεχνολογίες ελέγχου ταυτότητας και αρκετά ακόμη εργαλεία. Το Flask βασίζεται στην εργαλειοθήκη Werkzeg WSGI και στη μηχανή προτύπων Jinja2.

Το WSGI χρησιμοποιείται ως πρότυπο για την ανάπτυξη web εφαρμογών σε Python. Είναι η προδιαγραφή ενός κοινού interface μεταξύ διακομιστών ιστού και web εφαρμογών. Το Werkzeug είναι μια εργαλειοθήκη WSGI που υποστηρίζει αιτήματα HTTP, αντικείμενα απόκρισης και άλλες βοηθητικές λειτουργίες.

Το Jinja2 είναι μια δημοφιλής μηχανή προτύπων (template engine) για την Python. Ένα σύστημα για web templates συνδυάζει ένα template με μια πηγή δεδομένων για να εμφανίσει το περιεχόμενο μιας δυναμικής ιστοσελίδας. Με τον τρόπο αυτό μπορούμε να περάσουμε μεταβλητές της Python σε ένα template HTML όπως στο παράδειγμα:

```
<html>
    <head>
        <title>{{ title }}</title>
    </head>
    <body>
        <h1>Hello {{ username }}</h1>
    </body>
</html>
```

Η εγκατάσταση του Flask γίνεται με την εντολή: `pip install Flask`. Στον Κώδικα 28 φαίνεται μια πολύ απλή εφαρμογή Flask που εμφανίζει το μήνυμα ‘Hello World!’ στον browser του χρήστη.

1. Αρχικά εισάγεται η κλάση `Flask`.
2. Στη συνέχεια δημιουργείται ένα instance της κλάσης. Το πρώτο όρισμα είναι το όνομα του module ή του package της εφαρμογής. `__name__` είναι η συντόμευση που χρησιμοποιείται στις περισσότερες περιπτώσεις. Το όρισμα είναι απαραίτητο ώστε το Flask να γνωρίζει που θα αναζητήσει τους πόρους, όπως τα template ή τα στατικά αρχεία.
3. Στη συνέχεια χρησιμοποιείται ο decorator `route()` για να οριστεί το URL που θα ενεργοποιεί τη συνάρτηση.

4. Η συνάρτηση επιστρέφει το μήνυμα ‘Hello World!’ στον browser του χρήστη. Ο προεπιλεγμένος τύπος περιεχομένου είναι HTML, επομένως η HTML που βρίσκεται στη συμβολοσειρά θα γίνει render στον browser.

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"
```

Κώδικας 28. Παράδειγμα εφαρμογής Flask

Ο κώδικας της εφαρμογής θα πρέπει να αποθηκευτεί σε αρχείο με όνομα διαφορετικό του flask.py, γιατί τότε θα προκύψει conflict με το Flask. Αν η εφαρμογή αποθηκευτεί στο αρχείο hello.py, για παράδειγμα, μπορούμε να την εκτελέσουμε με την εντολή `flask --app hello run`. Αν το αρχείο ονομαστεί app.py ή wsgi.py, δεν χρειάζεται η επιλογή `-app`. Με την εντολή αυτή εκκινεί ένας απλός server και ο χρήστης μπορεί να δει το μήνυμα ‘Hello World!’ στη διεύθυνση http://127.0.0.1:5000/ [39].

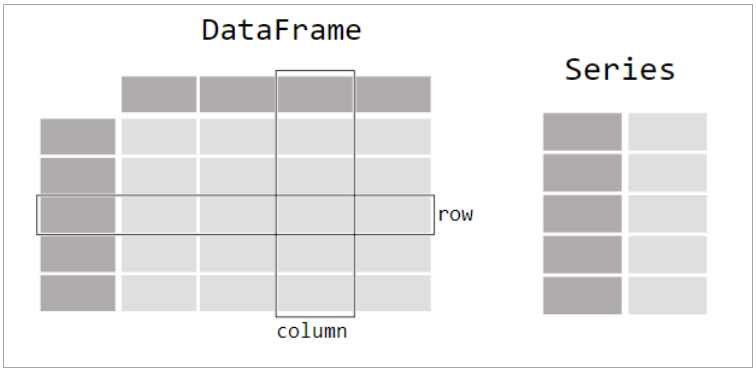
6.1.5 Pandas

Η pandas είναι μια βιβλιοθήκη ανοιχτού κώδικα της Python που διαθέτει συναρτήσεις και δομές δεδομένων για την εύκολη ανάγνωση και τον χειρισμό δεδομένων από αρχεία. Για παράδειγμα, επιτρέπει τον εύκολο χειρισμό δεδομένων που λείπουν (missing values), τη λήψη υποσυνόλου δεδομένων που πληρούν συγκεκριμένα κριτήρια, τον χειρισμό χρονοσειρών κλπ.

Η pandas είναι κατάλληλη για διάφορα είδη δεδομένων όπως:

- Δεδομένα πίνακα με ετερογενείς στήλες, όπως για παράδειγμα τα δεδομένα ενός πίνακα SQL ή ενός υπολογιστικού φύλλου Excel.
- Δεδομένα χρονοσειρών (διατεταγμένα ή μη, και όχι απαραίτητα σταθερής συχνότητας).
- Δεδομένα πίνακα με ετικέτες σειρών και στηλών.
- Οποιαδήποτε άλλη μορφή στατιστικών δεδομένων ή δεδομένων παρατήρησης.

Οι δύο κύριες δομές δεδομένων του pandas, Series (1 διάσταση) και DataFrame (2 διαστάσεις), χρησιμοποιούνται στη συντριπτική πλειοψηφία των περιπτώσεων στα οικονομικά, στη στατιστική, στις κοινωνικές επιστήμες και σε πολλούς τομείς της μηχανικής.



Εικόνα 18. Data structures στη βιβλιοθήκη pandas

Για να αποθηκεύσουμε δεδομένα σε έναν πίνακα, δημιουργούμε ένα DataFrame. Αν τα δεδομένα βρίσκονται σε ένα λεξικό με λίστες, τα κλειδιά του λεξικού θα χρησιμοποιηθούν ως κεφαλίδες στις στήλες και οι τιμές κάθε λίστας ως στήλες του DataFrame. Το DataFrame είναι μια δισδιάστατη δομή δεδομένων που μπορεί να αποθηκεύσει δεδομένα διαφορετικών τύπων (χαρακτήρες, ακέραιους, αριθμούς κινητής υποδιαστολής κ.α.) σε στήλες. Είναι παρόμοιο με ένα υπολογιστικό φύλλο, έναν πίνακα SQL ή με το data.frame στη γλώσσα R. Ένα παράδειγμα φαίνεται στον Κώδικα 29.

```
>>> import pandas as pd
>>> df = pd.DataFrame(
        {
            "Name": [
                "Braund, Mr. Owen Harris",
                "Allen, Mr. William Henry",
                "Bonnell, Miss. Elizabeth",
            ],
            "Age": [22, 35, 58],
            "Sex": ["male", "male", "female"],
        }
    )
>>> df
>>>                          Name  Age     Sex
    0   Braund, Mr. Owen Harris   22    male
    1  Allen, Mr. William Henry   35    male
    2  Bonnell, Miss. Elizabeth   58  female
```

Κώδικας 29. Δημιουργία DataFrame με τη βιβλιοθήκη Pandas

Αν επιλέξουμε μια στήλη από ένα pandas DataFrame, το αποτέλεσμα θα είναι ένα pandas Series. Μπορούμε επίσης να δημιουργήσουμε ένα Series, όπως φαίνεται στον Κώδικα 30.

```
>>> df["Age"]
>>> 0    22
    1    35
    2    58
    Name: Age, dtype: int64
```

```
>>> ages = pd.Series([22, 35, 58], name="Age")
>>> ages
>>> 0    22
    1    35
    2    58
    Name: Age, dtype: int64
```

Κώδικας 30. Δημιουργία Series με τη βιβλιοθήκη Pandas

Το pandas αναπαριστά τα δεδομένα ως πίνακα με γραμμές και στήλες, και υποστηρίζει τον χειρισμό των δεδομένων και την πραγματοποίηση υπολογισμών όπως θα έκανε ο χρήστης και με ένα λογισμικό υπολογιστικών φύλλων [40].

6.1.6 Bootstrap

Το Bootstrap είναι το πιο δημοφιλές front-end framework για εύκολη και γρήγορη ανάπτυξη web εφαρμογών. Περιέχει templates για τυπογραφία, φόρμες, κουμπιά, πίνακες, πλοήγηση, modals, καρουζέλ εικόνων και πολλά άλλα που βασίζονται σε HTML και CSS, καθώς και πρόσθετα JavaScript.

Το Bootstrap 5 είναι η νεότερη έκδοση του Bootstrap. Η έκδοση αυτή υποστηρίζει τις πιο πρόσφατες, σταθερές εκδόσεις των κυριότερων προγραμμάτων περιήγησης και πλατφορμών. Ωστόσο, δεν υποστηρίζει τον Internet Explorer 11 και τις προγενέστερες εκδόσεις του, και η jQuery έχει αντικατασταθεί με vanilla JavaScript.

Για να χρησιμοποιήσει κανείς το Bootstrap θα πρέπει είτε να κατεβάσει τον μεταγλωττισμένο κώδικα CSS και JavaScript, είτε να συμπεριλάβει τον κώδικα μέσω ενός CDN. Στον Κώδικα 31 φαίνεται πως προσθέτουμε το CSS του Bootstrap στην ετικέτα <link> (μέσα στο <head>), και τον κώδικα JavaScript στην ετικέτα <script> (αμέσως πριν κλείσει το <body>).

Το Bootstrap 5 χρησιμοποιεί στοιχεία HTML και ιδιότητες CSS που απαιτούν HTML5 doctype. Γι' αυτό η δήλωση τύπου εγγράφου (<!doctype html>) θα πρέπει να περιέχεται στην αρχή της σελίδας μαζί με τις ιδιότητες lang και charset. Το Bootstrap 5 είναι σχεδιασμένο για λειτουργία και στις κινητές συσκευές. Για να διασφαλιστεί σωστό rendering και touch zooming στις κινητές συσκευές, θα πρέπει το στοιχείο <head> να περιλαμβάνει την ετικέτα:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
```

```
    <title>Bootstrap demo</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-
GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
crossorigin="anonymous">
  </head>
  <body>
    <h1>Hello, world!</h1>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js" integrity="sha384-
w76AqPfDkMBDXo30jS1Sgez6pr3x5MlQ1ZAGC+nuZB+EYdgRZgiwxhTBTkF7CXvN"
crossorigin="anonymous"></script>
  </body>
</html>
```

Κώδικας 31. Παράδειγμα χρήσης του Bootstrap μέσω CDN

6.2 Έλεγχος του ιστότοπου

Αρχικά θα πρέπει να εξετάσουμε τον ιστότοπο από τον οποίο θέλουμε να συλλέξουμε τα δεδομένα. Το πρώτο βήμα είναι να ελέγξουμε το αρχείο robots.txt, το οποίο βρίσκεται στον σύνδεσμο https://www.bestprice.gr/robots.txt. Ένα τμήμα του αρχείου φαίνεται στον Κώδικα 32. Με βάση τις οδηγίες του αρχείου, απαγορεύεται η πρόσβαση στις σελίδες με τις κριτικές των προϊόντων, γιατί το path αυτών των σελίδων έχει τη μορφή /item/*/reviews. Επειδή η εφαρμογή σχεδιάστηκε για ερευνητικό και όχι εμπορικό σκοπό, επιλέγουμε να παραβλέψουμε τις οδηγίες του αρχείου robots.txt, αν και δεν αποτελεί καλή πρακτική. Γι' αυτό το λόγο θα αλλάξουμε την default επιλογή `'ROBOTSTXT_OBEY':True` στο αρχείο settings.py, όπως θα δείξουμε στη συνέχεια.

```
User-agent: *
Disallow: /cart
Disallow: /404
Disallow: /oops
Disallow: /error
Disallow: /env
Disallow: /legacy
Disallow: /login
Disallow: /logout
Disallow: /api
Disallow: /to
Disallow: /deals-new
Disallow: /collection
Disallow: /compare
Disallow: /badge
Disallow: /chrome
Disallow: /notifications
Disallow: /badgetFetch
Disallow: /item/*/reviews
Disallow: /m/*/reviews
Disallow: /search?q=*&deals=*&f1=*
```

Κώδικας 32. Τμήμα του αρχείου robots.txt του ιστότοπου bestprice.gr

Επόμενο βήμα είναι να εξετάσουμε αν το περιεχόμενο των σελίδων είναι απλός κώδικας HTML ή κώδικας JavaScript που εκτελείται στον browser. Ένας τρόπος για να το διαπιστώσουμε είναι να απενεργοποιήσουμε την JavaScript και να δούμε αν το περιεχόμενο που μας ενδιαφέρει είναι ορατό. Στον Chrome η απενεργοποίηση της JavaScript μπορεί να γίνει ως εξής: ανοίγουμε το παράθυρο Developer Tools, πατάμε Ctrl+Shift+P και πληκτρολογούμε Disable JavaScript.

Στη συγκεκριμένη εφαρμογή μας ενδιαφέρει η σελίδα του bestprice με τα αποτελέσματα αναζήτησης και η σελίδα με τις κριτικές των προϊόντων. Μετά την απενεργοποίηση της JavaScript, παρατηρούμε ότι η σελίδα με τα αποτελέσματα αναζήτησης εμφανίζεται ακριβώς ίδια, ενώ η σελίδα που περιέχει τις κριτικές εμφανίζεται κενή. Αυτό σημαίνει ότι στην πρώτη περίπτωση έχουμε απλό κώδικα HTML, ενώ στη δεύτερη περίπτωση το περιεχόμενο είναι αποτέλεσμα εκτέλεσης κώδικα JavaScript. Γι' αυτό το λόγο αποφασίσαμε να χρησιμοποιήσουμε τη βιβλιοθήκη scrapy-playwright.

6.3 Περιβάλλον προγραμματισμού

Η εφαρμογή υλοποιήθηκε σε λειτουργικό σύστημα Ubuntu 22.04.2 LTS και χρησιμοποιήθηκε ο editor Visual Studio Code. Η εγκατάσταση όλων των απαραίτητων βιβλιοθηκών έγινε σε virtual environment. Τα βήματα που ακολουθήσαμε ήταν τα εξής:

Εγκατάσταση του εργαλείου virtualenv

```
pip install virtualenv
```

Δημιουργία και άνοιγμα του καταλόγου project

```
mkdir project
cd project
```

Δημιουργία και ενεργοποίηση ενός virtual environment με το όνομα venv1

```
python3 -m venv venv1
source venv1/bin/activate
```

Εγκατάσταση των βιβλιοθηκών scrapy, scrapyrt, scrapy-playwright και εγκατάσταση των browser για τη λειτουργία του scrapy-playwright

```
pip install scrapy scrapyrt scrapy-playwright
playwright install
```

Εγκατάσταση των βιβλιοθηκών flask, requests και pandas

```
pip install flask requests pandas
```

Δημιουργία ενός νέου scrapy project με το όνομα reviews και μετάβαση στον κατάλογο reviews

```
scrapy startproject reviews
```

```
cd reviews
```

Δημιουργία μιας νέας αράχνης με το όνομα scrapereviews

```
scrapy genspider scrapereviews bestprice.gr
```

Αφού ολοκληρώσουμε τη συγγραφή του κώδικα, τρέχουμε το Scrapyrt μέσα στον κατάλογο του Scrapy project (κατάλογος reviews) με την εντολή `scrapyrt`. Στη συνέχεια τρέχουμε την εφαρμογή flask με την εντολή `flask run –debug` ή με την εντολή `python3 app.py`, όπως φαίνεται στην Εικόνα 18. Η εφαρμογή είναι έτοιμη προς χρήση στη διεύθυνση http://127.0.0.1:5000/.

Εικόνα 19. Εκκίνηση της εφαρμογής

6.4 Επεξήγηση του κώδικα της αράχνης

Αρχικά θα εξηγήσουμε τον κώδικα της αράχνης scrapereviews, ο οποίος βρίσκεται στο αρχείο scrapereviews.py. Όπως φαίνεται στον Κώδικα 33, έχουμε ορίσει τη συνάρτηση `should_abort_request` με την οποία ακυρώνουμε τη λήψη των εικόνων για να επιτύχουμε μεγαλύτερη ταχύτητα.

Στο λεξικό `custom_settings` κάνουμε τις ρυθμίσεις με βάση τις οποίες θα τρέχει η αράχνη. Οι ρυθμίσεις αυτές μπορούν να γίνουν και στο αρχείο settings.py, αλλά προτιμήσαμε να βρίσκονται στο ίδιο αρχείο με τον κώδικα της αράχνης.

- Με την επιλογή `'ROBOTSTXT_OBEY':False`, παρακάμπτουμε τις οδηγίες του αρχείου robots.txt προκειμένου να εξάγουμε τις κριτικές των προϊόντων από τις κατάλληλες σελίδες.
- Με την επιλογή `'AUTOTHROTTLE_ENABLED':True`, ενεργοποιούμε την επέκταση AutoThrottle του Scrapy, η οποία προσαρμόζει τις καθυστερήσεις μεταξύ των αιτημάτων ανάλογα με τον φόρτο του server. Με τον τρόπο αυτό η ταχύτητα προσαρμόζεται αυτόματα στο βέλτιστο επίπεδο και μειώνονται οι πιθανότητες αποκλεισμού από τον ιστότοπο.

- Η επιλογή `'PLAYWRIGHT_LAUNCH_OPTIONS':{"headless":False}` χρησιμοποιήθηκε στο στάδιο της ανάπτυξης της εφαρμογής για να ελέγξουμε αν η αράχνη εκτελεί τις σωστές ενέργειες (άνοιγμα σελίδων, scroll, πάτημα κουμπιών κλπ). Ένας headed browser έχει UI και είναι ιδανικός για debugging, ενώ ένας headless browser απαιτεί λιγότερους πόρους και είναι πιο γρήγορος.
- Στην επόμενη γραμμή αντικαθιστούμε τους default http και https download handlers με τους αντίστοιχους από τη βιβλιοθήκη scrapy-playwright.
- Με την επιλογή `'PLAYWRIGHT_DEFAULT_NAVIGATION_TIMEOUT':100000`, αυξάνουμε τον χρόνο timeout κατά την υποβολή αιτημάτων από το playwright. Η default τιμή είναι 30000 ms.
- Με την επιλογή `'PLAYWRIGHT_ABORT_REQUEST':should_abort_request`, ακυρώνουμε τα αιτήματα για τη λήψη των εικόνων, καθώς δεν μας ενδιαφέρει η εξαγωγή τους.
- Στην επόμενη γραμμή ορίζουμε ότι τα δεδομένα που εξάγονται από την αράχνη περνούν στο pipeline ReviewsPipeline για επεξεργασία.
- Στην τελευταία γραμμή αντικαθιστούμε τον default user agent που αποστέλλει το Scrapy κάθε φορά που υποβάλει ένα αίτημα, ο οποίος έχει τη μορφή Scrapy/VERSION (+https://scrapy.org). Από τη default τιμή του user agent ο ιστότοπος μπορεί εύκολα να αναγνωρίσει ότι τα αιτήματα υποβάλλονται από κάποιο bot και να τα μπλοκάρει.

Συνάρτηση start_requests

Στη συνάρτηση `start_requests` (Κώδικας 34) χρησιμοποιείται η παράμετρος `product`, η οποία αντιστοιχεί στον όρο που εισάγει ο χρήστης στη φόρμα αναζήτησης της εφαρμογής. Με βάση την παράμετρο αυτή σχηματίζεται το URL και υποβάλλεται το αίτημα στον ιστότοπο. Για παράδειγμα, αν ο χρήστης εισάγει στη φόρμα τον όρο «iphone 14», το URL διαμορφώνεται ως εξής: https://www.bestprice.gr/search?q=iphone+14. Στη συνέχεια στέλνουμε δύο αιτήματα HTTP στο συγκεκριμένο URL με διαφορετικές callback συναρτήσεις, τις `parse` και `parse_products`. Για να γίνει δύο φορές αίτημα στο ίδιο URL, χρησιμοποιούμε την παράμετρο `dont_filter=True`, διότι το Scrapy διαθέτει ένα ενσωματωμένο φίλτρο που μπλοκάρει την αποστολή του ίδιου request δεύτερη φορά.

```python
import scrapy
from scrapy_playwright.page import PageMethod
from scrapy.selector import Selector

# no images loaded (for increased speed)
def should_abort_request(request):
    return (
        request.resource_type == "image"
        or ".jpg" in request.url
    )

class ScrapereviewsSpider(scrapy.Spider):
    name = "scrapereviews"
    allowed_domains = ["bestprice.gr"]

    custom_settings = {
        'ROBOTSTXT_OBEY' : False,
        'AUTOTHROTTLE_ENABLED' : True,
        # 'PLAYWRIGHT_LAUNCH_OPTIONS' : {"headless": False},
        'DOWNLOAD_HANDLERS' : {
            'http': 'scrapy_playwright.handler.ScrapyPlaywrightDownloadHandler',
            'https': 'scrapy_playwright.handler.ScrapyPlaywrightDownloadHandler'
        },
        'PLAYWRIGHT_DEFAULT_NAVIGATION_TIMEOUT' : 100000,
        'PLAYWRIGHT_ABORT_REQUEST' : should_abort_request,
        'ITEM_PIPELINES' : {'reviews.pipelines.ReviewsPipeline': 300},
        'USER_AGENT' : 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.36'
    }

    headers = {
        'accept': 'application/json',
        'accept-encoding': 'gzip, deflate, br',
        'accept-language': 'en-US,en;q=0.9,el;q=0.8,fr;q=0.7',
        'referer' : 'https://www.bestprice.gr/',
        'user-agent': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.36'
    }
```

Κώδικας 33. Αράχνη scrapereviews - μέρος 1ο

```python

    def start_requests(self):
        query_parameters = self.product.strip().replace(" ", "+")
        full_url = "https://www.bestprice.gr/search?q=" + query_parameters

        yield scrapy.Request(
            url = full_url,
            # headers = self.headers,
            dont_filter = True,
            callback = self.parse
        )

        yield scrapy.Request(
            url = full_url,
            # headers = self.headers,
            dont_filter = True,
            callback = self.parse_products
        )

```

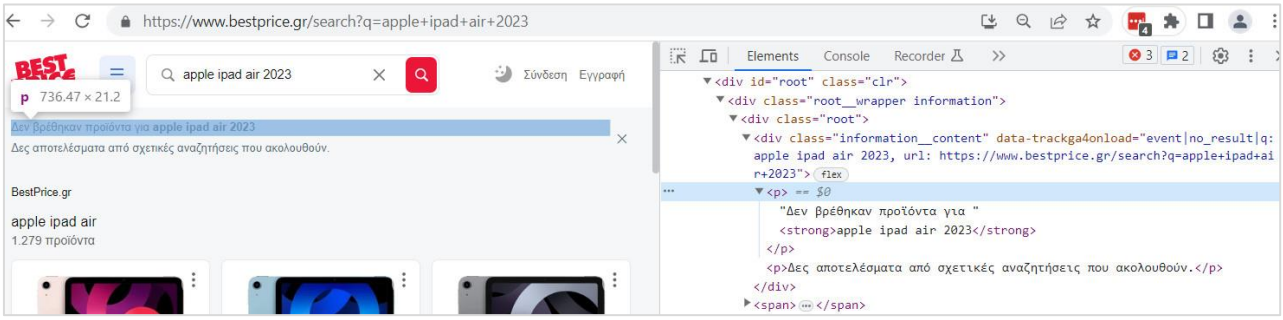
Κώδικας 34. Αράχνη scrapereviews - μέρος 2ο

Συνάρτηση parse

Στη συνάρτηση `parse` (Κώδικας 35) ελέγχουμε το αποτέλεσμα του αιτήματος. Εάν η αναζήτηση δεν έχει κάποιο αποτέλεσμα, εξάγουμε το μήνυμα που εμφανίζεται στον ιστότοπο χρησιμοποιώντας τον CSS selector που φαίνεται στην Εικόνα 20.

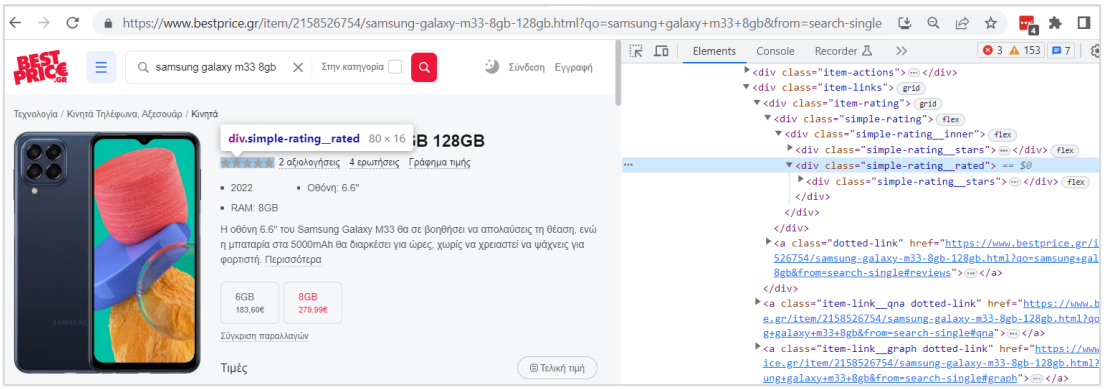
```
56
57      def parse(self, response):
58          # check if product not found
59          if response.css('div.root > div.information__content').get() is not None:
60              yield {
61                  'message' : "".join(response.css('div.information__content p:nth-child(1) *::text').getall())
62              }
63
64          # check if single-product page is displayed
65          elif response.css('div.item-layout').get() is not None:
66
67              # check if product has ratings
68              if response.css('div.simple-rating__rated').get() is not None:
69                  response_url = response.url
70                  reviews_url = response_url[:response_url.index(".html?")] + "/reviews"
71
72                  yield scrapy.Request(
73                      url = reviews_url,
74                      callback = self.parse_reviews,
75                      meta = {
76                          "playwright" : True,
77                          "playwright_include_page" : True,
78                          "playwright_page_methods" : [
79                              PageMethod("wait_for_load_state", "domcontentloaded"),
80                              PageMethod("evaluate", "window.scrollBy(0, document.body.scrollHeight)"),
81                              PageMethod("wait_for_timeout", 5000)
82                          ]
83                          },
84                      errback = self.errback_close_page
85                  )
86
```

Κώδικας 35. Αράχνη scrapereviews - μέρος 3ο



Εικόνα 20. Κανένα προϊόν στα αποτελέσματα αναζήτησης

Αν η αναζήτηση επιστρέφει μόνο ένα προϊόν, ελέγχουμε τον CSS selector που φαίνεται στην Εικόνα 21, για να διαπιστώσουμε αν το προϊόν έχει κριτικές. Αν υπάρχουν κριτικές, γίνεται νέο αίτημα HTTP για να λάβουμε τη σελίδα με τις κριτικές του προϊόντος. Όπως φαίνεται στην Εικόνα 21, ο ιστότοπος ανακατευθύνει τον χρήστη σε διαφορετικό URL. Το Scrapy χειρίζεται αυτόματα τις ανακατευθύνσεις, επομένως δεν χρειάζεται να προσθέσουμε κώδικα. Η σελίδα με τις κριτικές του προϊόντος βρίσκεται στη διεύθυνση https://www.bestprice.gr/item/2158526754/samsung-galaxy-m33-8gb-128gb/reviews. Επομένως, σχηματίζουμε κατάλληλα το URL για το νέο αίτημα και χρησιμοποιούμε τη βιβλιοθήκη scrapy-playwright.



Εικόνα 21. Μοναδικό προϊόν στα αποτελέσματα αναζήτησης

Όπως αναφέρθηκε σε προηγούμενη ενότητα, οι σελίδες με τις αξιολογήσεις των προϊόντων προκύπτουν από την εκτέλεση κώδικα JavaScript, γι' αυτό θα χρησιμοποιήσουμε έναν headless browser για να τις ανοίξουμε και να αντλήσουμε τα απαραίτητα δεδομένα. Για να γίνει αυτό εισάγουμε στο λεξικό `Request.meta` το ζεύγος κλειδιού-τιμής `"playwright":True`. Όταν η τιμή είναι True, το αίτημα πραγματοποιείται μέσω του Playwright.

Στο κλειδί `playwright_include_page` δίνουμε την τιμή True. Με τον τρόπο αυτό το αντικείμενο `Page` είναι διαθέσιμο στην callback συνάρτηση `parse_reviews` (Κώδικας 37) στο `response.meta['playwright_page']` και μπορούμε να επιλέξουμε και να εξάγουμε τα δεδομένα. Όταν η τιμή είναι False, η σελίδα κλείνει αμέσως μετά την ολοκλήρωση του αιτήματος.

Στο κλειδί `playwright_page_methods` αντιστοιχίζουμε μια λίστα αντικειμένων `PageMethod`. Με τον τρόπο αυτό μπορούμε να καλούμε μεθόδους στο αντικείμενο `Page`, πριν επιστραφεί το τελικό αντικείμενο `Response` στην callback συνάρτηση. Εδώ χρησιμοποιήσαμε τις μεθόδους `wait_for_load_state`, `evaluate` και `wait_for_timeout`. Με την πρώτη μέθοδο περιμένουμε το event DOMContentLoaded της JavaScript, δηλαδή να φορτωθεί το DOM. Με τη δεύτερη μέθοδο κάνουμε scroll μέχρι το τέλος της σελίδας για να εμφανιστεί το κουμπί «Φόρτωσε περισσότερες αξιολογήσεις», το οποίο χρησιμοποιούμε στη συνάρτηση `parse_reviews` για να εμφανίσουμε όλες τις κριτικές. Με την τρίτη μέθοδο ορίζουμε χρόνο timeout 5 δευτερόλεπτα.

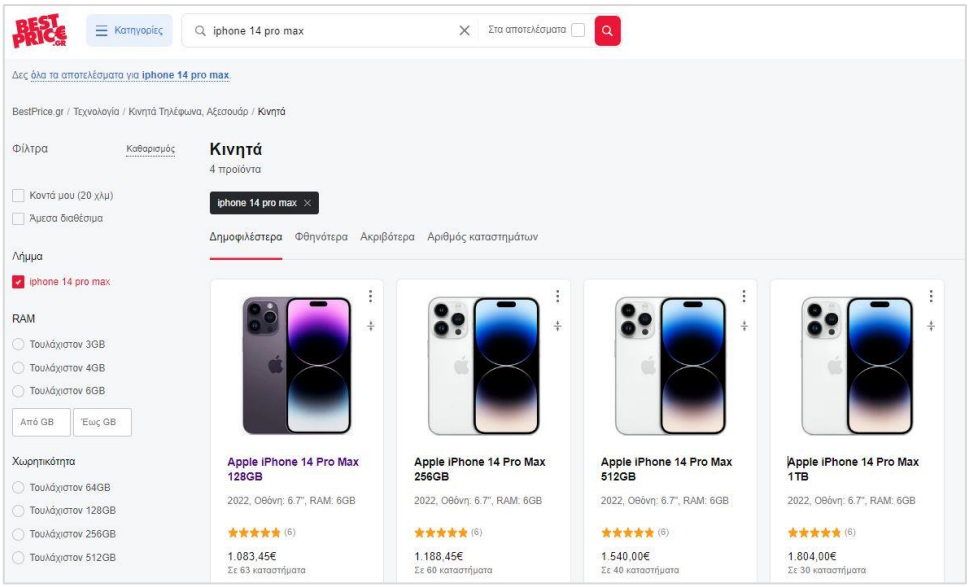
Όταν ορίζουμε `playwright_include_page=True`, είναι σημαντικό να κλείνουμε τις σελίδες όταν δεν τις χρειαζόμαστε πλέον. Η συνάρτηση `errback_close_page` χρησιμοποιείται για το κλείσιμο των σελίδων όταν ένα αίτημα αποτυγχάνει. Δηλαδή, ακόμα και αν έχουμε `playwright_include_page=False`, οι σελίδες κλείνουν αυτόματα μόλις προκύψει exception. Το κλείσιμο των σελίδων είναι απαραίτητο γιατί υπάρχει όριο στον αριθμό των ανοιχτών σελίδων, το οποίο δηλώνεται στην παράμετρο `PLAYWRIGHT_MAX_PAGES_PER_CONTEXT`. Η default τιμή της παραμέτρου είναι 4. Αν οι ανοιχτές σελίδες φτάσουν το όριο, η αράχνη μπορεί να κολλήσει.

```
87
88      def parse_products(self, response):
89          # check page structure
90          if response.css('div.page-products__main-wrapper div.p__products').get() is not None:
91
92              for product in response.css('div.page-products__main-wrapper div.p__products > div'):
93
94                  # check if product has ratings
95                  if product.css('div.p__rating').get() is not None:
96                      product_link = "https://www.bestprice.gr" + product.css('a.p__cover::attr(href)').get()
97                      reviews_url = product_link[:product_link.index(".html?")] + "/reviews"
98
99                      yield scrapy.Request(
100                         url = reviews_url,
101                         callback = self.parse_reviews,
102                         meta = {
103                             "playwright" : True,
104                             "playwright_include_page" : True,
105                             "playwright_page_methods" : [
106                                 PageMethod("wait_for_load_state", "domcontentloaded"),
107                                 PageMethod("evaluate", "window.scrollBy(0, document.body.scrollHeight)"),
108                                 PageMethod("wait_for_timeout", 5000)
109                             ]
110                             },
111                         errback = self.errback_close_page
112                     )
113
114             # check for pagination
115             next_page = response.css('li.pagination-action.pagination-action-next a::attr(href)').get()
116
117             if next_page is not None:
118                 next_page_url = "https://www.bestprice.gr" + next_page
119
120                 yield scrapy.Request(
121                     url =  next_page_url,
122                     callback = self.parse_products
123                 )
124
```

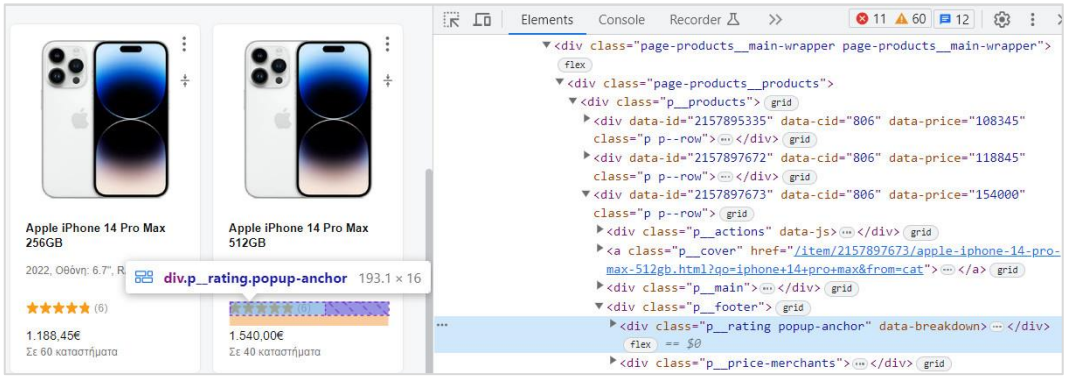
Κώδικας 36. Αράχνη scrapereviews - μέρος 4ο

Συνάρτηση parse_products

Η συνάρτηση `parse_products` (Κώδικας 36) χρησιμοποιείται όταν ο ιστότοπος επιστρέφει μια σελίδα με τη μορφή που φαίνεται στην Εικόνα 22. Στην περίπτωση αυτή, ελέγχουμε για κάθε προϊόν αν υπάρχουν αξιολογήσεις, αναζητώντας τον CSS selector που φαίνεται στην Εικόνα 23. Αν υπάρχουν αξιολογήσεις, παίρνουμε το URL του προϊόντος και σχηματίζουμε το URL της σελίδας που περιέχει τις κριτικές. Στη συνέχεια, υποβάλλουμε νέο αίτημα HTTP με τον ίδιο τρόπο που περιγράψαμε και στη συνάρτηση `parse`.

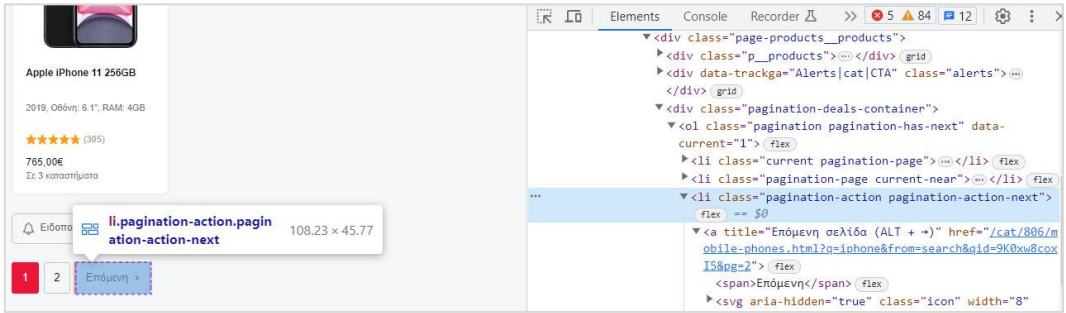


Εικόνα 22. Πολλαπλά προϊόντα στα αποτελέσματα αναζήτησης



Εικόνα 23. Έλεγχος για την ύπαρξη αξιολογήσεων

Στο επόμενο τμήμα του κώδικα ελέγχουμε αν υπάρχει σελιδοποίηση (pagination) στα αποτελέσματα αναζήτησης (βλ. Εικόνα 24). Αν ναι, βρίσκουμε το URL της επόμενης σελίδας και υποβάλλουμε νέο HTTP αίτημα με callback συνάρτηση την `parse_products`. Η διαδικασία επαναλαμβάνεται μέχρι την τελευταία σελίδα όπου παύει να εμφανίζεται το κουμπί «Επόμενη».



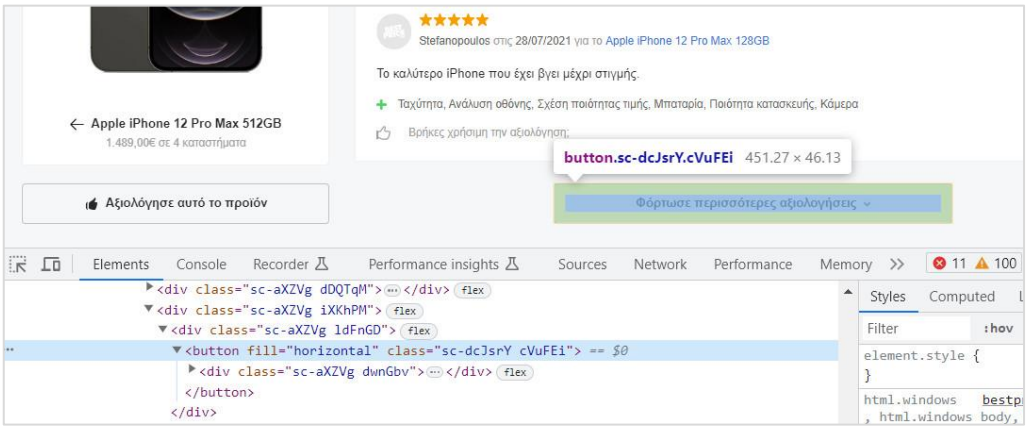
Εικόνα 24. Έλεγχος για σελιδοποίηση

Συνάρτηση parse_reviews

Η συνάρτηση `parse_reviews` (Κώδικας 37) εξάγει τα δεδομένα σχετικά με τις αξιολογήσεις των προϊόντων. Αρχικά, ελέγχει αν υπάρχει το κουμπί «Φόρτωσε περισσότερες αξιολογήσεις», το οποίο φαίνεται στην Εικόνα 25, και με τη μέθοδο click του Playwright κάνει κλικ στο κουμπί, όσο αυτό είναι ορατό στη σελίδα. Στη συνέχεια, για κάθε μια αξιολόγηση εξάγει το όνομα του αγοραστή, την ημερομηνία, τη βαθμολογία και την κριτική.

```
125
126     async def parse_reviews(self, response):
127         page = response.meta['playwright_page']
128
129         try:
130             while button := page.locator('button[fill="horizontal"]:only-child'):
131                 await button.click()
132                 await page.wait_for_timeout(2000)
133         except:
134             pass
135
136         s = Selector(text = await page.content())
137
138         product_name = s.css('div[data-testid="product-info"] span::text').get()
139
140         for review in s.css('div[data-testid="reviewItem"]'):
141             yield {
142                 'Product name' : product_name,
143                 'Author' : review.css('span[itemprop="name"]::text').get(),
144                 'Date' : review.css('small > small::attr(title)').get(),
145                 'Rating' : review.css('meta[itemprop="ratingValue"]::attr(content)').get(),
146                 'Review' : "".join(review.css('span[itemprop="reviewBody"] *::text').getall()).replace("\n", " ")
147             }
148         await page.close()
149
150
151     async def errback_close_page(self, failure):
152         page = failure.request.meta["playwright_page"]
153         await page.close()
154
```

Κώδικας 37. Αράχνη scrapereviews - μέρος 5ο



Εικόνα 25. Κουμπί φόρτωσης περισσότερων αξιολογήσεων

Αφαίρεση διπλότυπων αξιολογήσεων

Επόμενο βήμα είναι να αφαιρέσουμε τις αξιολογήσεις που εμφανίζονται δύο ή περισσότερες φορές. Αυτό συμβαίνει όταν στα αποτελέσματα αναζήτησης εμφανίζονται παραλλαγές του ίδιου προϊόντος (βλ. Εικόνα 22). Για το σκοπό αυτό χρησιμοποιήθηκε ένα Item Pipeline.

Κάθε φορά που εξάγεται ένα σύνολο δεδομένων (item) από την αράχνη, αποστέλλεται στο Item Pipeline για επεξεργασία. Ένα Item Pipeline είναι μια κλάση Python που υλοποιεί τη μέθοδο `process_item`. Η μέθοδος `process_item` λαμβάνει ένα item, εκτελεί μια ενέργεια και αποφασίζει αν το item θα συνεχίσει μέσω του pipeline ή αν θα απορριφθεί. Όλες οι pipeline κλάσεις γράφονται στο αρχείο pipelines.py. Για να τις ενεργοποιήσουμε, θα πρέπει να τις προσθέσουμε στην επιλογή `ITEM_PIPELINES` στο αρχείο settings.py.

Για να αφαιρέσουμε τις διπλότυπες αξιολογήσεις, δημιουργήσαμε την κλάση `ReviewsPipeline` (Κώδικας 38), η οποία ελέγχει την τιμή ‘Author’ και αν υπάρχει ήδη, απορρίπτει το item (δεν επιστρέφεται στην έξοδο).

```
6   from itemadapter import ItemAdapter
7   from scrapy.exceptions import DropItem
8
9   class ReviewsPipeline:
10
11      def __init__(self):
12          self.authors_seen = set()
13
14
15      def process_item(self, item, spider):
16          adapter = ItemAdapter(item)
17
18          if adapter.get('Author'):
19              if adapter.get('Author') in self.authors_seen:
20                  raise DropItem(f"Duplicate item found: {item!r}")
21              else:
22                  self.authors_seen.add(adapter.get('Author'))
23                  return item
24          else:
25              return item
26
```

Κώδικας 38. Αρχείο pipelines.py

6.5 Επεξήγηση του κώδικα της εφαρμογής Flask

Στον Κώδικα 39 φαίνεται το αρχείο app.py που περιέχει την εφαρμογή. Αρχικά εισάγονται οι απαραίτητες βιβλιοθήκες και δημιουργείται ένα instance της κλάσης Flask.

Ο decorator `route()` χρησιμοποιείται για να συνδέσουμε μια συνάρτηση με μια διεύθυνση URL. Η μέθοδος `render_template()` χρησιμοποιείται για την απόδοση ενός template. Οι παράμετροι που δέχεται είναι το όνομα του template και οι μεταβλητές (με τη μορφή keyword arguments) που θέλουμε να περάσουμε στο template engine. Η συνάρτηση `index()` κάνει render το template base.html (Κώδικας 40) στο root URL, δηλαδή στο http://127.0.0.1:5000/.

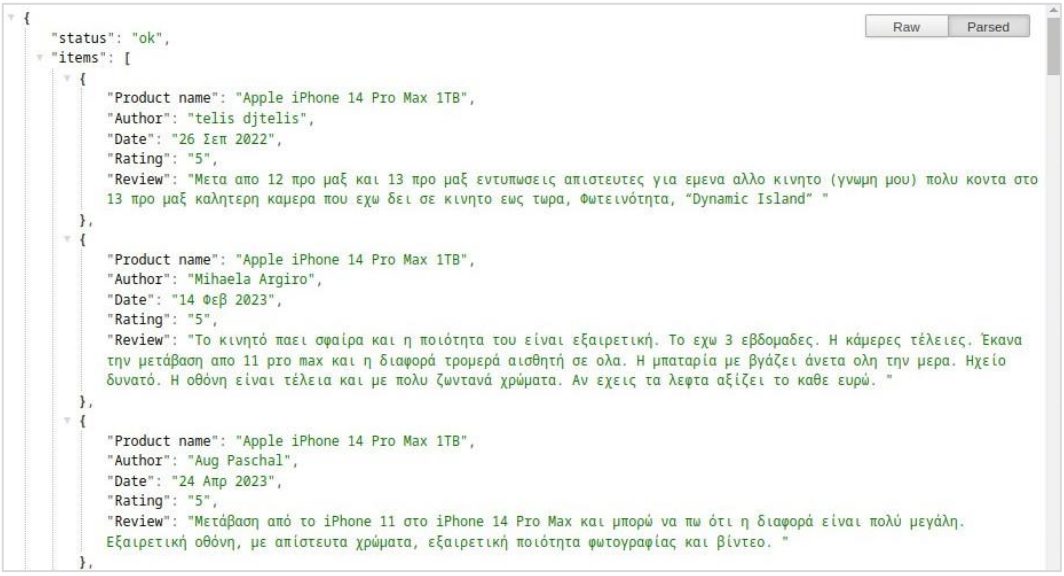
Η συνάρτηση `scrape()` λαμβάνει την τιμή που εισάγει ο χρήστης στο πεδίο input της φόρμας και σχηματίζει το URL για να τρέξει η αράχνη scrapereviews μέσω του Scrapyrt. Στη συνέχεια στέλνει ένα GET request στο συγκεκριμένο URL με χρήση της βιβλιοθήκης requests. Το Scrapyrt λειτουργεί σαν web service και εμφανίζει τα αποτελέσματα σε μορφή json, όπως φαίνεται στην Εικόνα 26. Για να λάβουμε αυτά τα αποτελέσματα χρησιμοποιούμε τη μέθοδο `response.json()`.

```python
from flask import Flask, render_template, request, send_file, url_for
import requests
import pandas as pd

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('base.html')

@app.route('/', methods=['GET', 'POST'])
def scrape():
    product = request.form['product']
    url = 'http://localhost:9080/crawl.json?spider_name=scrapereviews&start_requests=true&crawl_args={"product":"' + product + '"}'
    response = requests.get(url)
    data = response.json()

    message = ''
    items_num = len(data['items'])

    # procuct not found
    if items_num != 0 and 'message' in data['items'][0]:
        message = data['items'][0]['message']
        return render_template('home.html', message=message)

    # product has no reviews
    elif items_num == 0:
        return render_template('home.html', message=message, items_num=items_num)

    else:
        df = pd.DataFrame(data=data['items'])

        # save the DataFrame as a csv file
        df.to_csv('reviews.csv', index=False)
        return render_template('home.html', message=message, items_num=items_num, tables=[df.to_html(justify='left', classes='table table-stripped')])

@app.route('/download')
def download_file():
    path = 'reviews.csv'
    return send_file(path, as_attachment=True)


if __name__ == "__main__":
    app.run(host="localhost", debug=True)

```

Κώδικας 39. Αρχείο app.py

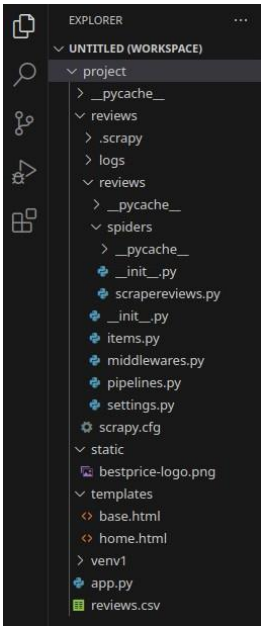
Στη συνέχεια διακρίνουμε τρεις περιπτώσεις. Εάν το προϊόν δεν υπάρχει στα αποτελέσματα αναζήτησης του bestprice ή αν δε διαθέτει αξιολογήσεις, η συνάρτηση `scrape()` κάνει render το template home.html περνώντας τις μεταβλητές `message` και `items_num`. Αν το προϊόν έχει αξιολογήσεις, φορτώνουμε τα δεδομένα σε ένα DataFrame και με τη μέθοδο `to_csv()` αποθηκεύουμε το DataFrame σε αρχείο csv με το όνομα reviews. Στην περίπτωση αυτή, γίνεται render το template home.html με τις μεταβλητές `message`, `items_num` και `tables`. Εδώ χρησιμοποιείται η μέθοδος `to_html()` για να μετατρέψουμε το DataFrame σε πίνακα HTML.

Η συνάρτηση `download_file()` επιστρέφει το αρχείο reviews.csv, όταν ο χρήστης κάνει κλικ στον σύνδεσμο download. Για το κατέβασμα του αρχείου χρησιμοποιείται η μέθοδος `send_file()` με όρισμα το path του αρχείου.



Εικόνα 26. Αποτελέσματα του Scrapyrt

Τα template αποθηκεύονται στον κατάλογο /templates. Στη συγκεκριμένη εφαρμογή χρησιμοποιούνται δύο template αρχεία (base.html, home.html). Τα στατικά αρχεία της εφαρμογής (αρχεία CSS, αρχεία JavaScript, εικόνες κλπ) αποθηκεύονται στον κατάλογο /static. Στην Εικόνα 27 φαίνεται ο κατάλογος της εφαρμογής με όλα τα περιεχόμενά του.



Εικόνα 27. Κατάλογος αρχείων της εφαρμογής flask

Templates και template inheritance

Το front-end της εφαρμογής δημιουργήθηκε με Bootstrap 5. Στον Κώδικα 40 φαίνεται το αρχείο base.html, στο οποίο έχουμε συμπεριλάβει τον κώδικα CSS και JavaScript του Bootstrap μέσω CDN. Στις γραμμές 9-18 του αρχείου base.html χρησιμοποιήθηκε η βιβλιοθήκη jQuery, προκειμένου να ελεγχθεί η λειτουργία του κουμπιού Submit. Όταν το πεδίο input της φόρμας είναι κενό ή περιέχει μόνο space χαρακτήρες, το κουμπί Submit έχει το attribute `disabled` και είναι ανενεργό. Σε αντίθετη περίπτωση το attribute αφαιρείται, το κουμπί ενεργοποιείται και ο χρήστης μπορεί να υποβάλει τη φόρμα.

Για να γίνει αναφορά στα στατικά αρχεία της εφαρμογής, χρησιμοποιείται η συνάρτηση `url_for()` με ορίσματα το όνομα του καταλόγου (static) και το όνομα του αρχείου. Στη γραμμή 25 του αρχείου base.html φαίνεται πως χρησιμοποιήθηκε η συνάρτηση `url_for()` για να εμφανίσουμε το λογότυπο του bestprice στην εφαρμογή.

Στη συνέχεια ακολουθεί μια απλή φόρμα αναζήτησης που αποτελείται από ένα input πεδίο και το κουμπί Submit. Στις γραμμές 38-39 χρησιμοποιούνται οι ετικέτες `{% block %}` μέσα στις οποίες εμφανίζεται το περιεχόμενο του αρχείου home.html (Κώδικας 41). Αυτή η ιδιότητα του Jinja ονομάζεται template inheritance και μας επιτρέπει να έχουμε ένα κύριο template που περιέχει τα κοινά στοιχεία όλων των σελίδων της εφαρμογής, και block τα οποία γεμίζουν περιεχόμενο από τα θυγατρικά template.

Το αρχείο home.html ξεκινά με την ετικέτα `{% extends %}` η οποία ενημερώνει το template engine πως το συγκεκριμένο template ΄επεκτείνει΄ ένα άλλο. Αν το προϊόν που αναζητά ο χρήστης

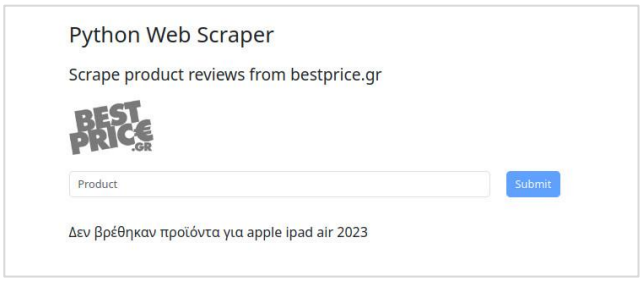
δεν υπάρχει, η αράχνη εξάγει το αντίστοιχο μήνυμα από το bestprice και η εφαρμογή το εμφανίζει στον browser (βλ. Εικόνα 28). Αυτό γίνεται περικλείοντας τη μεταβλητή `message` σε διπλές αγκύλες, όπως φαίνεται στη γραμμή 5 του Κώδικα 41. Αν το προϊόν δεν έχει αξιολογήσεις, η εφαρμογή εμφανίζει το μήνυμα ‘Product has no reviews!’, όπως φαίνεται στην Εικόνα 29. Αν το προϊόν διαθέτει αξιολογήσεις, η εφαρμογή επιστρέφει τις αξιολογήσεις σε μορφή πίνακα και δίνει στον χρήστη τη δυνατότητα να κατεβάσει τα αποτελέσματα σε αρχείο csv, όπως δείχνουν οι Εικόνες 30 και 31 αντίστοιχα. Για τη λήψη του αρχείου χρησιμοποιείται η συνάρτηση `url_for()` με ορίσματα το όνομα της συνάρτησης και το όνομα του αρχείου, όπως φαίνεται στη γραμμή 17 του Κώδικα 41.

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="utf-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1">
6      <title>Python Web Scraper</title>
7      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet"
8      integrity="sha384-KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ" crossorigin="anonymous">
9      <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.4/jquery.min.js"></script>
10     <script>
11         $(function(){
12             $("#inputProduct").on("keyup", function(e) {
13               if (e.target.value.replace(/\s/g, '').length) {
14                 $("#submit").removeAttr("disabled")
15               }
16             });
17         });
18     </script>
19 </head>
20 <body>
21     <div class="container mt-4">
22         <div class="row gy-3">
23             <h2>Python Web Scraper</h2>
24             <h4>Scrape product reviews from bestprice.gr</h4>
25             <img src="{{ url_for('static', filename='bestprice-logo.png') }}" style="width:150px; height:auto;">
26             <form class="row mt-4" method="POST">
27                 <div class="col-6">
28                     <label for="inputProduct" class="visually-hidden">Product</label>
29                     <input type="text" class="form-control" id="inputProduct" placeholder="Product" name="product">
30                 </div>
31                 <div class="col-auto">
32                     <button type="submit" class="btn btn-primary mb-3" id="submit" disabled>Submit</button>
33                 </div>
34             </form>
35         </div>
36     </div>
37     <div class="container mt-4">
38         {% block content %}
39         {% endblock %}
40     </div>
41     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js"
42     integrity="sha384-ENjdO4Dr2bkBIFxQpeoTz1HIcje39Wm4jDKdf19U8gI4ddQ3GYNS7NTKfAdVQSZe" crossorigin="anonymous"></script>
43 </body>
44 </html>
```

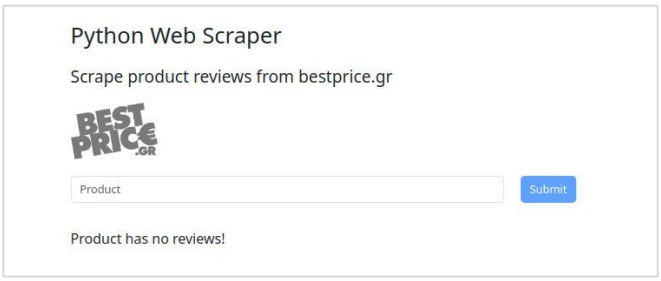
Κώδικας 40. Αρχείο base.html

```
{% extends 'base.html' %}
{% block content %}
    {% if message %}
    <div class="row">
    <h5>{{ message }}</h5>
    </div>
    {% elif items_num == 0 %}
    <div class="row">
    <h5>Product has no reviews!</h5>
    </div>
    {% else %}
    <div class="row">
        <div class="col-auto">
        <h5>Download csv file:</h5>
        </div>
        <div class="col-auto">
        <a href = "{{ url_for('download_file', filename='reviews.csv') }}">download</a>
        </div>
    </div>
    <div class="row mt-3">
        {% for table in tables %}
        {{ table|safe }}
        {% endfor %}
    </div>
    {% endif %}
{% endblock %}

```

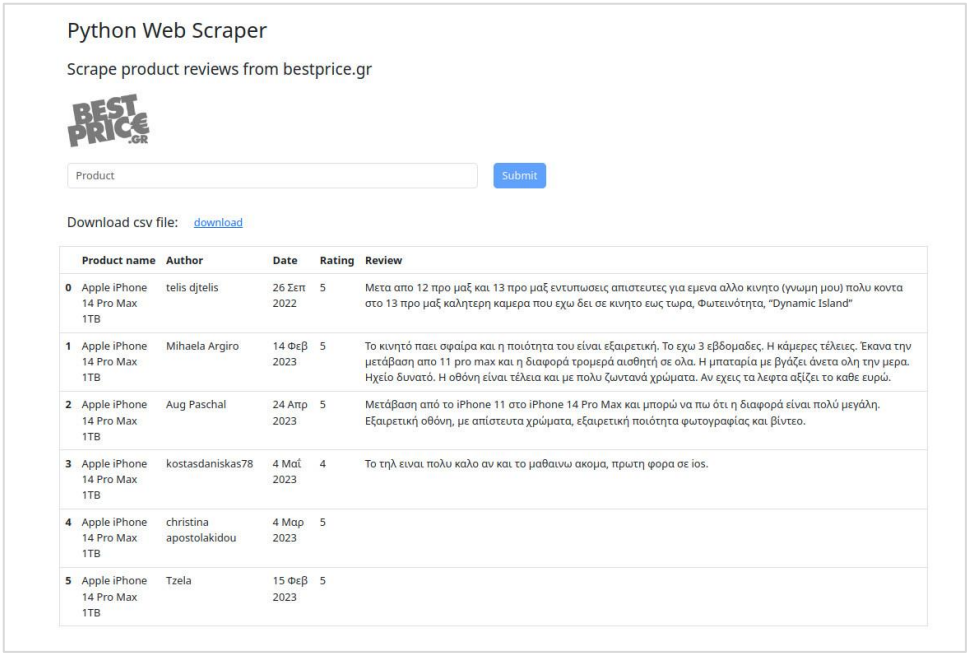
Κώδικας 41. Αρχείο home.html



Εικόνα 28. Αποτέλεσμα για προϊόν που δεν βρέθηκε

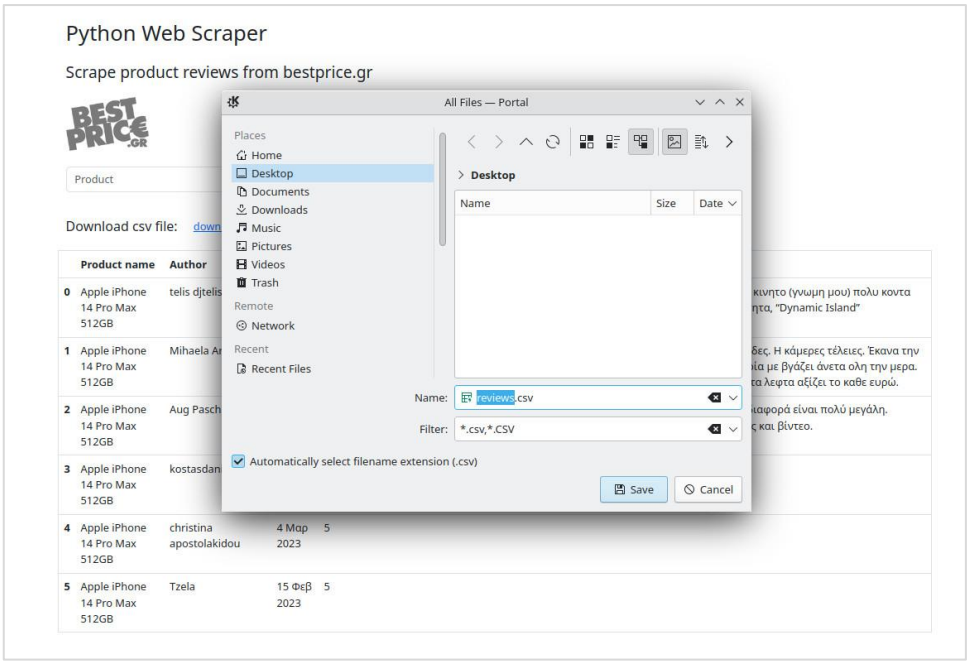


Εικόνα 29. Αποτέλεσμα για προϊόν χωρίς αξιολογήσεις



	Product name	Author	Date	Rating	Review
0	Apple iPhone 14 Pro Max 1TB	telis djtelis	26 Σεπ 2022	5	Μετα απο 12 προ μαξ και 13 προ μαξ εντυπωσεις απιστευτες για εμενα αλλο κινητο (γνωμη μου) πολυ κοντα στο 13 προ μαξ καλητερη καμερα που εχω δει σε κινητο εως τωρα, Φωτεινότητα, "Dynamic Island"
1	Apple iPhone 14 Pro Max 1TB	Mihaela Argiro	14 Φεβ 2023	5	Το κινητό παει σφαίρα και η ποιότητα του είναι εξαιρετική. Το εχω 3 εβδομαδες. Η κάμερες τέλειες. Έκανα την μετάβαση απο 11 pro max και η διαφορά τρομερά αισθητή σε ολα. Η μπαταρία με βγάζει άνετα ολη την μερα. Ηχείο δυνατό. Η οθόνη είναι τέλεια και με πολυ ζωντανά χρώματα. Αν εχεις τα λεφτα αξίζει το καθε ευρώ.
2	Apple iPhone 14 Pro Max 1TB	Aug Paschal	24 Απρ 2023	5	Μετάβαση από το iPhone 11 στο iPhone 14 Pro Max και μπορώ να πω ότι η διαφορά είναι πολύ μεγάλη. Εξαιρετική οθόνη, με απίστευτα χρώματα, εξαιρετική ποιότητα φωτογραφίας και βίντεο.
3	Apple iPhone 14 Pro Max 1TB	kostasdaniskas78	4 Μαΐ 2023	4	Το τηλ ειναι πολυ καλο αν και το μαθαινω ακομα, πρωτη φορα σε ios.
4	Apple iPhone 14 Pro Max 1TB	christina apostolakidou	4 Μαρ 2023	5	
5	Apple iPhone 14 Pro Max 1TB	Tzela	15 Φεβ 2023	5	

Εικόνα 30. Αποτέλεσμα για προϊόν με αξιολογήσεις



Εικόνα 31. Λήψη αρχείου csv

7. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΟΠΤΙΚΕΣ

Το τελευταίο κεφάλαιο περιλαμβάνει μια σύντομη σύνοψη της διπλωματικής εργασίας και τις προοπτικές για μετέπειτα συνέχιση της εργασίας.

7.1 Σύνοψη της διπλωματικής εργασίας

Ο Ιστός αποτελεί μια τεράστια και πανίσχυρη βάση δεδομένων, καθώς ένας μεγάλος όγκος δεδομένων παράγεται και προστίθεται καθημερινά. Με την άνοδο των big data και της επιστήμης δεδομένων, τα δεδομένα είναι πιο πολύτιμα από ποτέ και χρησιμοποιούνται για την εκπαίδευση αλγορίθμων μηχανικής μάθησης, την παραγωγή γνώσης, την πρόβλεψη γεγονότων κ.ά.

Η εξαγωγή των δεδομένων με μη αυτόματο τρόπο είναι μια πολύ αργή και χρονοβόρα διαδικασία. Γι' αυτό χρησιμοποιείται η μέθοδος του web scraping, η οποία επιτρέπει τη συλλογή των δεδομένων με χρήση προγραμματισμού. Επίσης, η αυτοματοποίηση των προγραμμάτων περιήγησης που μιμείται τις ενέργειες του χρήστη, όπως είναι το κλικ και η κύλιση της σελίδας, μας δίνει τη δυνατότητα να συλλέξουμε χρήσιμα δεδομένα απλά και αποτελεσματικά.

Αντικείμενο της παρούσας διπλωματικής εργασίας ήταν η μελέτη της εξαγωγής δεδομένων ιστού και η ανάπτυξη μιας εφαρμογής για τη συλλογή δεδομένων από το διαδίκτυο. Παρουσιάστηκαν οι τεχνικές, οι εφαρμογές και τα νομικά θέματα που σχετίζονται με το web scraping και στη συνέχεια

έγινε αναφορά στα διαθέσιμα εργαλεία. Η εργασία επικεντρώθηκε στην εξαγωγή δεδομένων με Python και έγινε παράθεση των σημαντικότερων βιβλιοθηκών που διαθέτει η γλώσσα για αυτόν το σκοπό. Επίσης, έγινε αναφορά σε εμπορικά εργαλεία web scraping, όπως SaaS εφαρμογές, desktop εφαρμογές και επεκτάσεις για τα προγράμματα περιήγησης. Τα εργαλεία αυτά απευθύνονται σε χρήστες με ή χωρίς γνώσεις προγραμματισμού και απλοποιούν τη διαδικασία συλλογής των δεδομένων. Τέλος, παρουσιάστηκαν οι καλές πρακτικές που πρέπει να ακολουθούνται κατά το σχεδιασμό εργαλείων web scraping, όπως η προσαρμογή των κεφαλίδων HTTP, η διαχείριση των cookies κ.ά.

Η εφαρμογή που αναπτύχθηκε, συλλέγει αξιολογήσεις προϊόντων από ένα δημοφιλή ιστότοπο σύγκρισης τιμών. Στο UI της εφαρμογής περιέχεται μια φόρμα HTML, όπου ο χρήστης εισάγει το προϊόν που τον ενδιαφέρει. Οι αξιολογήσεις του προϊόντος εξάγονται από τον ιστότοπο και παρουσιάζονται σε μορφή πίνακα, και ο χρήστης έχει τη δυνατότητα να κατεβάσει τα αποτελέσματα σε αρχείο csv. Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκαν τα frameworks Scrapy και Flask.

7.2 Προοπτικές

Στην παρούσα εργασία αναπτύχθηκε μια εφαρμογή που εξάγει κριτικές πελατών από ένα συγκεκριμένο ιστότοπο. Η ανατροφοδότηση από τους πελάτες είναι μια σημαντική πηγή πληροφοριών για τις επιχειρήσεις, για να βελτιώσουν τα προϊόντα και τις υπηρεσίες τους. Ωστόσο, για να εξαχθούν χρήσιμες πληροφορίες από τις κριτικές των πελατών, θα πρέπει να γίνει ανάλυση κειμένου με text mining.

Η τεχνική του text mining περιλαμβάνει τη χρήση επεξεργασίας φυσικής γλώσσας (NLP) και μηχανικής μάθησης (ML) για την ανάλυση και την ερμηνεία κειμένου. Μέσω του text mining μπορούμε να εκτελέσουμε διάφορες εργασίες, όπως:

- *Ανάλυση συναισθήματος (sentiment analysis)*, για να διακρίνουμε τη χροιά του μηνύματος (θετική, αρνητική ή ουδέτερη).
- *Εντοπισμό θέματος (topic modeling)*, για να αναγνωρίσουμε τα κύρια θέματα ή τις κατηγορίες του κειμένου.
- *Εξαγωγή λέξεων-κλειδιών (keyword extraction)*, για να βρούμε τις πιο σχετικές ή συχνές λέξεις και φράσεις στο κείμενο.
- *Περίληψη κειμένου (text summarization)*, για να παράγουμε τη σύνοψη του κειμένου.

Ωστόσο, η τεχνική του text mining έχει αρκετούς περιορισμούς και προκλήσεις που πρέπει να ληφθούν υπόψη, όπως είναι η ποσότητα και η ποιότητα των δεδομένων, η γλώσσα κλπ. Θα πρέπει

να υπάρχουν αρκετά δεδομένα για την εκπαίδευση και τη δοκιμή των μεθόδων text mining, και να διασφαλιστεί ότι τα δεδομένα είναι σχετικά, πλήρη και αμερόληπτα. Επίσης, θα πρέπει να ληφθούν υπόψη παραλλαγές και ασάφειες της ανθρώπινης γλώσσας, όπως η αργκό, οι ιδιωματισμοί, ο σαρκασμός ή η ειρωνεία.

Η παρούσα εργασία θα μπορούσε να συνεχιστεί ώστε μετά τη συγκέντρωση των αξιολογήσεων να ακολουθήσει η ανάλυσή τους για την εξαγωγή χρήσιμων συμπερασμάτων. Η Python διαθέτει το εργαλείο NLTK (Natural Language Toolkit), το οποίο περιλαμβάνει ένα σύνολο βιβλιοθηκών για την επεξεργασία κειμένου. Ωστόσο, η ανάλυση κειμένου στην ελληνική γλώσσα παρουσιάζει μεγαλύτερη δυσκολία, γιατί τα περισσότερα εργαλεία έχουν σχεδιαστεί για την αγγλική γλώσσα. Γι' αυτό το λόγο θα μπορούσε να μελετηθεί η εργασία των Papantoniou και Tzitzikas (2020), στην οποία παρατίθενται έργα, πηγές και εργαλεία που αφορούν στην επεξεργασία της ελληνικής γλώσσας [61]. Μια ακόμη ενδιαφέρουσα εργασία είναι αυτή των Prokopidis και Piperidis (2020), στην οποία παρουσιάζεται ένα νευρωνικό NLP toolkit για την ελληνική γλώσσα, το οποίο διαθέτει modules για προσθήκη ετικετών POS (Part-of-Speech), λημματοποίηση (lemmatization), ανάλυση εξάρτησης (dependency parsing) και ταξινόμηση κειμένου [62].

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Khder, M. (2021). Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application. *International Journal of Advances in Soft Computing and its Applications*. http://dx.doi.org/10.15849/IJASCA.211128.11

[2] Web scraping. https://en.wikipedia.org/wiki/Web_scraping

[3] Dallmeier, E. C. (2021). Computer Vision-based Web Scraping for Internet Forums. *2021 7th International Conference on Optimization and Applications (ICOA)*. https://doi.org/10.1109/ICOA51614.2021.9442634

[4] Krotov, V., & Silva, L. (2018). Legality and Ethics of Web Scraping. *Conference: Twenty-fourth Americas Conference on Information Systems.*

[5] What is the Document Object Model? https://www.w3.org/TR/WD-DOM/introduction.html

[6] Document Object Model. https://en.wikipedia.org/wiki/Document_Object_Model#DOM_tree_structure

[7] Χατζηλυγερούδης, Ι. (2022). Αναπαράσταση γνώσης στον παγκόσμιο ιστό (XML-XPath). Ανακτήθηκε από https://ceidnotes.net/notes/Αναπαράσταση+Γνώσης+στον+Παγκόσμιο+Ιστό/Διαφάνειες+Θεωρίας/04%20XML-XPath.pdf

[8] CSS Selectors. https://www.w3schools.com/css/css_selectors.asp

[9] Python RegEx. https://www.w3schools.com/python/python_regex.asp

[10] What is Python Used For: Web Scraping and Other Use Cases (2020). Ανακτήθηκε από https://www.linkedin.com/pulse/what-python-used-web-scraping-other-use-cases-iveta-vi%C5%A1torskyt%C4%97?trk=public_profile_article_view

[11] 5 Preferred Programming Languages for Web Scraping (2022). Ανακτήθηκε από https://opendatascience.com/5-preferred-programming-languages-for-web-scraping/

[12] Socket Programming HOWTO. https://docs.python.org/3/howto/sockets.html

[13] re - Regular expression operations. https://docs.python.org/3/library/re.html#functions

[14] urllib3. https://urllib3.readthedocs.io/en/stable/

[15] Requests: HTTP for Humans. https://requests.readthedocs.io/en/latest/

[16] Beautiful Soup Documentation. https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[17] pyspider. http://docs.pyspider.org/en/latest/

[18] Scrapy 2.8 documentation. https://docs.scrapy.org/en/latest/index.html

[19] GRequests: Asynchronous Requests. https://github.com/spyoungtech/grequests

[20] AIOHTTP. https://docs.aiohttp.org/en/stable/

[21] HTTPX. https://www.python-httpx.org/

[22] Selenium with Python. https://selenium-python.readthedocs.io/getting-started.html

[23] Playwright for Python. https://playwright.dev/python/

[24] Pyppeteer: The Puppeteer For Python Developers. https://www.scrapingbee.com/blog/pyppeteer/

[25] pyppeteer. https://miyakogi.github.io/pyppeteer/

[26] ScraperAPI. https://www.scraperapi.com/documentation/python/

[27] ScrapingBee. https://www.scrapingbee.com/documentation/

[28] scrapestack. https://scrapestack.com/documentation

[29] Diffbot - Extract API. https://docs.diffbot.com/reference/extract-introduction

[30] Octoparse. https://www.octoparse.com/product

[31] ParseHub. https://www.parsehub.com/

[32] ScrapeStorm. https://www.scrapestorm.com/

[33] Web Scraper. https://webscraper.io/documentation/

[34] Instant Data Scraper. https://webrobots.io/instantdata/

[35] Data Miner. https://dataminer.io/

[36] Simplescraper. http://simplescraper.io/

[37] Playwright integration for Scrapy. https://github.com/scrapy-plugins/scrapy-playwright

[38] Scrapyrt HTTP API. https://scrapyrt.readthedocs.io/en/stable/api.html#scrapyrt-http-api

[39] Flask Documentation (2.2.x). https://flask.palletsprojects.com/en/2.2.x/

[40] pandas documentation. https://pandas.pydata.org/docs/index.html

[41] Bootstrap 5 Tutorial. https://www.w3schools.com/bootstrap5/index.php

[42] Mitchell, R. (2018). *Web Scraping with Python* (2nd ed.). USA: O'Reilly Media, Inc.

[43] robots.txt. https://en.wikipedia.org/wiki/Robots.txt

[44] Lotfi, C., Srinivasan, S., Ertz, M., & Latrous, I. (2021). Web Scraping Techniques and Applications: A Literature Review. *SCRS Conference Proceedings on Intelligent Systems*. http://dx.doi.org/10.52458/978-93-91842-08-6-38

[45] Melchor, R. A., Fonseca, M., Rey, B., Hernandez, A., Puertas, B., Gomez, S., Palomino, D., Román, L. G., Peña, A. F., & Mateos, M. V. (2020). CT-152: Application of Web-Scraping Techniques for Autonomous Massive Retrieval of Hematologic Patients' Information During SARS-CoV2 Pandemic. *Clinical Lymphoma, Myeloma and Leukemia, 20, S214.*

[46] Dascalu, D., Paraschiv, I., Bogdan, N., Dascalu, M., Trausan-Matu, S., & Nuta, A. (2019). Intelligent Platform for the Analysis of Drug Leaflets Using NLP Techniques. *18th RoEduNet Conference: Networking in Education and Research (RoEduNet).* http://dx.doi.org/10.1109/ROEDUNET.2019.8909606

[47] Himawan, A., Priadana, A., & Murdiyanto, A. (2020). Implementation of Web Scraping to Build a Web-Based Instagram Account Data Downloader Application. *IJID (International Journal on Informatics for Development) 9(2):59-65*. http://dx.doi.org/10.14421/ijid.2020.09201

[48] Axenbeck, J., & Breithaupt, P. (2021). Innovation indicators based on firm websites—Which website characteristics predict firm-level innovation activity? PLoS ONE 16(4): e0249583. https://doi.org/10.1371/journal.pone.0249583

[49] Balendran, T., Liyanapathirana, C., Sampath, K., & Rupasinghe, P. (2018). Extracting Unstructured Data and Analysis and Prediction of Financial Event Modeling. *The Institution of Engineering and Technology (IET)*.

[50] Boegershausen, J., Datta, H., Borah, A., & Stephen, A. (2022). Fields of Gold: Scraping Web Data for Marketing Insights. *Journal of Marketing 86(5):1-20.* http://dx.doi.org/10.1177/00222429221100750

[51] Saranya, G., Gopinath, N., Geetha, G., Meenakshi, K., & Nithya, M. (2020). Prediction of Customer Purchase Intention Using Linear Support Vector Machine in Digital Marketing. *Journal of Physics Conference Series. 1712. 1-10*. http://dx.doi.org/10.1088/1742-6596/1712/1/012024

[52] Nguyen, V., Sukunesan, S., & Huynh, M. (2021). Analyzing Australian SME Instagram Engagement via Web Scraping. *Pacific Asia Journal of the Association for Information Systems 13 (2)*. https://doi.org/10.17705/1pais.13202

[53] Deng, S. (2020). Research on the Focused Crawler of Mineral Intelligence Service Based on Semantic Similarity. *Journal of Physics Conference Series 1575(1):012142*. http://dx.doi.org/10.1088/1742-6596/1575/1/012142

[54] Kotouza, M., Tsarouchis, S. F., Kyprianidis, A. C., Chrysopoulos, A., & Mitkas, P. (2020). Towards Fashion Recommendation: An AI System for Clothing Data Retrieval and Analysis. *Artificial Intelligence Applications and Innovations*. https://doi.org/10.1007/978-3-030-49186-4_36

[55] Wang, H., & Song, J. (2019). Fast Retrieval Method of Forestry Information Features Based on Symmetry Function in Communication Network. *Symmetry 11(3):416*. http://dx.doi.org/10.3390/sym11030416

[56] Seliverstov, Y., Seliverstov, S., Malygin, I., & Korolev, O. (2020). Traffic safety evaluation in Northwestern Federal District using sentiment analysis of Internet users' reviews. *Transportation Research Procedia 50(3):626-635*. http://dx.doi.org/10.1016/j.trpro.2020.10.074

[57] Suganya, E., & Vijayarani, S. (2021). Firefly Optimization Algorithm Based Web Scraping for Web Citation Extraction. *Wireless Personal Communications 118(2)*. https://doi.org/10.1007/s11277-021-08093-z

[58] Rahmatulloh, A., & Gunawan, R. (2020). Web Scraping with HTML DOM Method for Data Collection of Scientific Articles from Google Scholar. Indonesian Journal of Information Systems 2(2):16. http://dx.doi.org/10.24002/ijis.v2i2.3029

[59] Li, R.Y.M. (2020). Building updated research agenda by investigating papers indexed on Google Scholar: A natural language processing approach. *Advances in Artificial Intelligence, Software and Systems Engineering. AHFE 2020*. https://doi.org/10.1007/978-3-030-51328-3_42

[60] Santos, B. S., Silva, I., da Câmara Ribeiro-Dantas, M., Alves, G., Endo, P. T., & Lima, L. (2020). COVID-19: A scholarly production dataset report for research analysis. *Data in Brief, 32, 106178*. http://dx.doi.org/10.1016/j.dib.2020.106178

[61] Papantoniou, K., & Tzitzikas, Y. (2020). NLP for the Greek Language: A Brief Survey. *11th Hellenic Conference on Artificial Intelligence (SETN 2020)*, 101-109. http://dx.doi.org/10.1145/3411408.3411410

[62] Prokopidis, P., & Piperidis, S. (2020). A Neural NLP toolkit for Greek. *11th Hellenic Conference on Artificial Intelligence (SETN 2020)*, 125-128. https://doi.org/10.1145/3411408.3411430