ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Recipe Web Scraper

Άννα Α. Πετρίδου

Επιβλέπων: **Αλέξανδρος Ντούλας,** Επίκουρος Καθηγητής

ΑΘΗΝΑ

ΙΑΝΟΥΑΡΙΟΣ 2023

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Recipe Web Scraper

Άννα Α. Πετρίδου
Α.Μ.: 1115201600135

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Αλέξανδρος Ντούλας,** Επίκουρος Καθηγητής

ΠΕΡΙΛΗΨΗ

Σκοπός της εργασίας αυτής είναι η δημιουργία μίας εφαρμογής, που θα συλλέγει δεδομένα από πολλές διαφορετικές πηγές και θα τα προσφέρει συλλογικά στο χρήστη με τρόπο και λειτουργικότητες που θα τον διευκολύνουν.

Η εργασία αυτή λοιπόν αφορά μία εφαρμογή η οποία σαρώνει 5 μεγάλους ιστότοπους με συνταγές μαγειρικής, μαζεύει όση πληροφορία χρειάζεται από αυτούς και την περνάει σε μία βάση δεδομένων. Στη συνέχεια, μέσω του backend και του frontend που έχουν δημιουργηθεί, η εφαρμογή προσφέρει στο χρήστη μία σειρά από λειτουργίες και δυνατότητες τις οποίες μπορεί να εκμεταλλευτεί κάνοντας χρήση της διεπαφής χρήστη που έχουμε υλοποιήσει (user interface).

Πιο συγκεκριμένα, αρχικά δημιουργήσαμε τους 5 web crawlers, που ουσιαστικά σαρώνουν τις ιστοσελίδες, φιλτράροντας όλους τους συνδέσμους τους και αποθηκεύοντας μόνο όσους αφορούν συνταγές, σε ένα ξεχωριστό αρχείο κειμένου. Η υλοποίηση των crawlers έγινε με python χρησιμοποιώντας το “scrapy”.

Έπειτα, οι 5 web scrapers που έχουμε υλοποιήσει σε java, σαρώνουν κάθε σύνδεσμο που διαβάζουν από τα αρχεία κειμένου και αποθηκεύουν τις επιθυμητές πληροφορίες για την κάθε συνταγή, σε μία βάση δεδομένων MySQL. Το scraping των επιθυμητών δεδομένων από κάθε site με java γίνεται με τη βοήθεια του εργαλείου “Jsoup”.

Αφού περαστεί όλη η απαραίτητη πληροφορία στη βάση δεδομένων, το REST API που υλοποιήσαμε σε java με “spring boot”, παρέχει μια σειρά από λειτουργίες αναζήτησης στον χρήστη, που φροντίζει να παραδοθούν στο frontend που υλοποιήσαμε.

Τέλος, η διεπαφή χρήστη, δηλαδή το frontend, για το οποίο χρησιμοποιήσαμε “Vue.js”, φροντίζει τα δεδομένα και οι λειτουργίες της εφαρμογής να παρουσιάζονται με τρόπο εύχρηστο στο χρήστη.

Αποτέλεσμα αυτής της εργασίας είναι, η δημιουργία ενός project, περνώντας από όλες τις φάσεις του development, βλέποντας και μαθαίνοντας την πορεία για έναν πλήρη κύκλο σχεδιασμού και υλοποίησης μιας εφαρμογής, από τον οποίο προκύπτει μία εφαρμογή λειτουργική αλλά και χρήσιμη.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Απόξεση – Σάρωση Ιστού

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: εξόρυξη, σάρωση δεδομένων, διεπαφή χρήστη, web crawling, web scraping, backend, frontend

ABSTRACT

The purpose of this project is to create an application, that will collect data from many different sources and offer them collectively to the user in such a way and with functionalities that will facilitate him.

So, this work concerns an application that scans 5 large websites with cooking recipes, collects as much information as it needs from them and passes it to a database. Then, through the backend and frontend that have been created, the application offers the users a series of functions and features that they can take advantage of using the user interface that we have implemented.

More specifically, we initially created the 5 web crawlers, which basically scan the websites, filtering all their links and saving only those related to recipes, in a separate text file. The implementation of the crawlers was done with python using "scrapy".

Then, the 5 web scrapers that we have implemented in java, scan each link they read from the text files and store the desired information for each recipe, in a MySQL database. The scraping of the desired data from each java site is done with the help of the "Jsoup" tool.

After passing all the necessary information to the database, the REST API that we implemented in java with "spring boot", provides a series of search functions to the user, which takes care of being delivered to the frontend that we implemented.

Finally, the user interface, i.e., the frontend, for which we used “Vue.js”, makes sure that the data and functions of the application are presented in a user-friendly way.

The result of this work is the creation of a project, going through all the phases of development, seeing, and learning the course for a complete cycle of planning and implementing an application, from which a functional but also useful application results.

SUBJECT AREA: Web Scraping

KEYWORDS: scraping, crawling, backend, frontend, user-interface

Στην οικογένεια και ιδιαίτερα στον παππού μου, Γεώργιο Τσαγκαλάκη.

ΕΥΧΑΡΙΣΤΙΕΣ

Για τη διεκπεραίωση της παρούσας πτυχιακής εργασίας, θα ήθελα να ευχαριστήσω τον επιβλέποντα, επίκουρο καθηγητή Αλέξανδρο Ντούλα, για την αρχική ιδέα, την καθοδήγησή, μα και συνολικά για την πολύτιμη συνεισφορά του στην ολοκλήρωση της.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΠΡΟΛΟΓΟΣ

Η πτυχιακή εργασία πραγματοποιήθηκε στην Αθήνα το ακαδημαϊκό έτος 2022 και έχει στόχο τον σχεδιασμό και την υλοποίηση μίας εφαρμογής που θα ανακτά πληροφορία μέσω σάρωσης ιστού και θα παρέχει στο χρήστη μία διεπαφή με μία σειρά από δυνατότητες. Ευχαριστώ θερμά τον κύριο Α. Ντούλα για τη βοήθεια και καθοδήγησή του καθόλη τη διάρκεια εκπόνησης της εργασίας.

1. ΕΙΣΑΓΩΓΗ

1.1 Γενική Περιγραφή Έργου

Σήμερα υπάρχουν άπειρες εφαρμογές στο διαδίκτυο, δεδομένου ότι ο καθένας πλέον μπορεί να φτιάξει, γεγονός το οποίο έχει αποφέρει τον κατακερματισμό της πληροφορίας.

Ένα από τα συνήθη προβλήματα με τα οποία έρχεται αντιμέτωπος κάθε χρήστης, είναι η ανάκτηση πληροφορίας όταν αυτή είναι διαμοιρασμένη σε πολλές διαφορετικές πηγές. Υπάρχουν τόσες πολλές εφαρμογές και ιστοσελίδες με διαφορετική πληροφορία, που είναι αρκετά δύσκολο να αναζητήσουμε αυτό που θέλουμε, καθώς πρέπει να ψάξουμε σε κάθε πηγή ξεχωριστά.

Για παράδειγμα, όσον αφορά τις συνταγές μαγειρικής, υπάρχουν πάρα πολλά sites(ιστοσελίδες) με διαφορετικό περιεχόμενο και ο χρήστης για να μπορέσει να αναζητήσει μέσα στην πληροφορία των μεγάλων sites, θα πρέπει να μπει να ψάξει σε καθένα ξεχωριστά.

Για να επιλύσουμε το πρόβλημα αυτό, θέλαμε να δημιουργήσουμε μία εφαρμογή η οποία συγκεντρώνει πληροφορία από διαφορετικές πηγές (δηλαδή από διαφορετικές βάσεις δεδομένων) με αυτοματοποιημένο τρόπο, την επεξεργάζεται και προσφέρει στο χρήστη, μέσω μίας εύχρηστης διεπαφής, μια σειρά από δυνατότητες.

Ειδικότερα, αυτό που έχουμε υλοποιήσει ανακτά πληροφορία από διαφορετικές βάσεις δεδομένων και αφού την ενοποιήσει, την εισάγει σε μία δική μας βάση δεδομένων. Στη συνέχεια, έχουμε μία διεπαφή χρήστη σε μορφή ιστοσελίδας, που προσφέρει στο χρήστη τη δυνατότητα να αναζητά συνταγές με βάση πολλά διαφορετικά φίλτρα όπως π.χ. αριθμός θερμίδων, συστατικά, πλήθος υλικών κτλ. με τα οποία ο χρήστης μπορεί να κάνει την αναζήτησή του πιο στοχευμένη.

1.2 Γενικές Απαιτήσεις

Αρχικά, η επιλογή των κατάλληλων εργαλείων και τεχνολογιών αποτέλεσε σε ένα βαθμό δυσκολία, καθώς χρειάστηκε χρόνος και έρευνα για να αναζητήσουμε, να συγκρίνουμε και να αποφασίσουμε τι ταιριάζει καλύτερα στις ανάγκες μας.

Σχετικά με την υλοποίηση της εφαρμογής, η διαδικασία αποτέλεσε πρόκληση, διότι δεν έχουμε προσπέλαση στις βάσεις δεδομένων τις οποίες θέλουμε να ενοποιήσουμε. Αν είχαμε προσπέλαση στις ίδιες τις βάσεις των 5 ιστοσελίδων που επιλέξαμε, θα ήταν αρκετά απλό να μεταφέρουμε πληροφορία από τη μία βάση στην άλλη.

Ωστόσο, εμείς έπρεπε να «σαρώσουμε» την πληροφορία, για το οποίο αποφασίσαμε να χρησιμοποιήσουμε scraping [1], το οποίο έχει τις δικές του δυσκολίες καθώς το ίντερνετ είναι δυναμικό. Η δυσκολία στην ανάκτηση της πληροφορίας έγκειται στις διαφορές που έχουν τα sites μεταξύ τους τόσο στο περιεχόμενο όσο και στην οργάνωση της πληροφορίας τους. Συνεπώς, εμείς έπρεπε να λάβουμε την πληροφορία από τα sites αυτά που έχουν διαφορετικό σκελετό και να την ενοποιήσουμε σε ένα δικό μας κοινό σχήμα. Κληθήκαμε δηλαδή να παράγουμε ένα κοινό σχήμα για βάσεις δεδομένων των οποίων δε γνωρίζουμε το δικό τους.

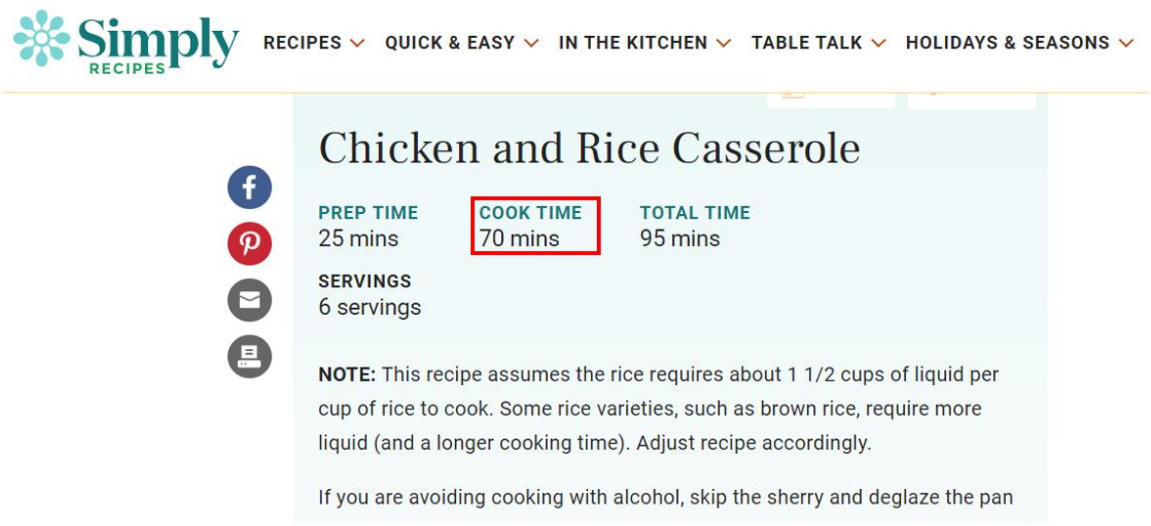
Ακόμη και στην περίπτωση που γνωρίζαμε την εικόνα των βάσεων αυτών, η υλοποίηση θα μπορούσε να αποτελέσει πρόκληση, πόσο μάλλον τώρα που όλη την πληροφορία την λαμβάνουμε από τη διεπαφή τους.

Συνεπώς, χρειάστηκε ο σχεδιασμός της δίκης μας βάσης να είναι αρκετά ευέλικτος, ώστε το σχήμα της να ικανοποιεί όλους τους περιορισμούς και να μπορεί να αποθηκεύσει την πληροφορία από όλα τα αλλά sites.

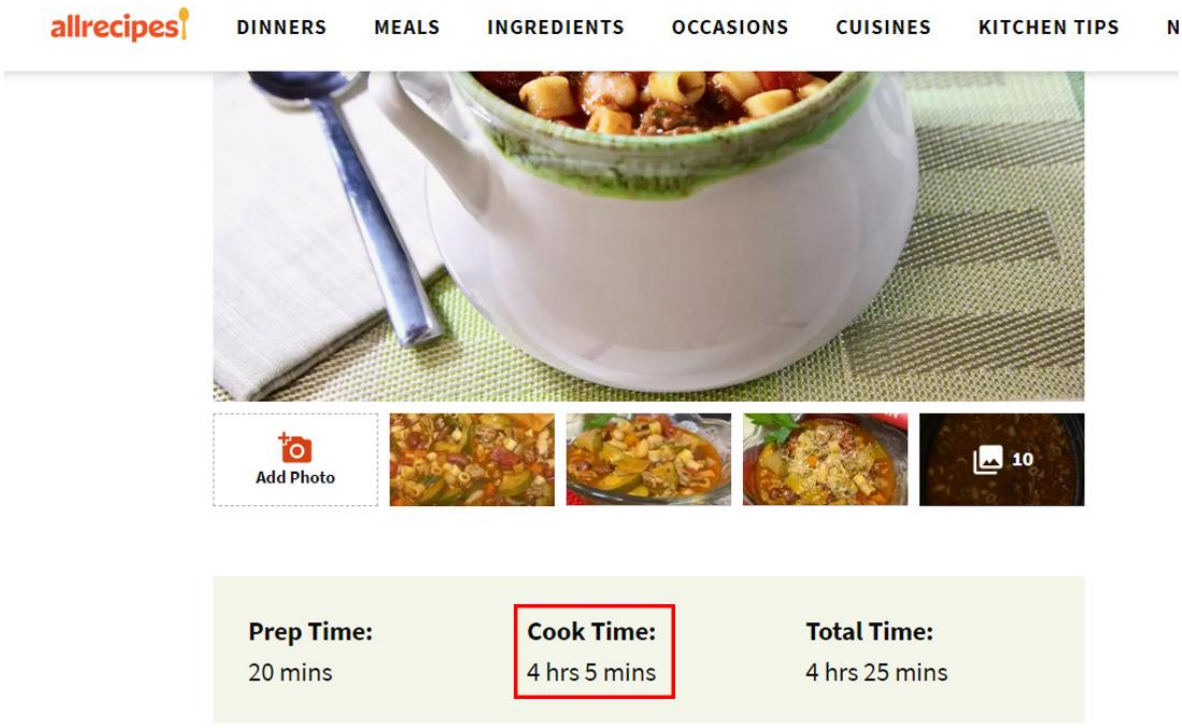
Η υλοποίηση μας προσφέρει τη δυνατότητα ενημέρωσης της βάσης δεδομένων αυτόματα, αφού το crawling(ανίχνευση) [2] και αντιστοίχως το scraping(σάρωση) μπορούν να εκτελούνται συνέχεια στο παρασκήνιο και να ενημερώνουν διαρκώς τη βάση.

Επόμενη πρόκληση αποτέλεσε, τόσο η δημιουργία των scrapers(σαρωτές) όσο και του backend(πίσω μέρος μιας εφαρμογής). Πιο συγκεκριμένα, αφού αποφασίσαμε να αντλήσουμε δεδομένα για τις συνταγές από 5 διαφορετικούς ιστότοπους, έπρεπε να φτιάξουμε και 5 διαφορετικούς scrapers. Τα sites αυτά περιέχουν από εκατοντάδες έως και χιλιάδες συνταγές το καθένα, τις οποίες εμείς θέλαμε να συλλέξουμε με έναν ομοιόμορφο τρόπο και να τις αποθηκεύσουμε στη βάση. Μία από τις βασικές προκλήσεις ήταν η επιτυχής ανάκτηση των δεδομένων μέσω των scrapers, καθώς όλα τα πεδία και ο κώδικας των sites είναι διαφορετικός.

Για παράδειγμα, όπως βλέπουμε στις παρακάτω εικόνες, ο χρόνος μαγειρέματος απεικονίζεται διαφορετικά στους 2 ιστότοπους (στον έναν σε λεπτά στον άλλον σε ώρες).



Εικόνα 1: Χρόνος εκτέλεσης από τον ιστότοπο "simplyrecipes.com"



Εικόνα 2: Χρόνος εκτέλεσης από τον ιστότοπο "allrecipes.com"

Συνεπώς εμείς έπρεπε τόσο να καταφέρουμε να ανακτήσουμε και να αναπαραστήσουμε σωστά τα δεδομένα, παρόλο που ήταν διαφορετικά οργανωμένα

αλλά και να κάνουμε τις κατάλληλες μετατροπές πριν την εισαγωγή τους στη βάση δεδομένων.

Επίσης, έπρεπε να δημιουργήσουμε κατάλληλους αλγοριθμικούς μηχανισμούς, ώστε να αναγνωρίζουμε από ποιες συνταγές ενδεχομένως λείπουν (από το συγγραφέα) βασικές πληροφορίες που όμως περιέχονται σε άλλες. Για να μη χαλάσει η ομοιομορφία λοιπόν και για να λειτουργήσει σωστά μετέπειτα η αναζήτησή στο backend, έπρεπε να σκεφτούμε κριτήρια και μηχανισμούς με τους οποίους θα παραλείπονται αυτές οι συνταγές “εξαιρέσεις” τελείως από τη διαδικασία. Αυτό απαιτούσε ειδικούς τρόπους διαχωρισμού και επισήμανσης των σελίδων αυτών ως μη κατάλληλες.

Όσον αφορά τον crawler(ανιχνευτή), έπρεπε να σχεδιαστεί μία λύση η οποία θα σαρώνει τις ιστοσελίδες και θα βρίσκει όλους τους συνδέσμους που περιέχονται σε αυτή, εξαιρώντας σωστά όσους είναι εξωτερικοί(σύνδεσμοι προς άλλα site), αλλά και των εσωτερικών συνδέσμων που όμως, δεν αφορούν συνταγές. Όλους αυτούς του συνδέσμους τους εξάγουμε σε ένα αρχείο κειμένου, το οποίο δίνουμε αργότερα ως input(εισαγόμενα δεδομένα) στους scrapers για να σαρώσουν τις σελίδες.

Αρκετές απαιτήσεις υπήρχαν φυσικά και στο σωστό σχεδιασμό της βάσης δεδομένων, καθώς εκεί θα βασίζονταν αργότερα όλες οι αναζητήσεις, αλλά και στη δημιουργία των φίλτρων αναζήτησης, ώστε να επιστρέφουν καλά αποτελέσματα σε ό,τι αναζητήσεις μπορεί να πραγματοποιήσει ο χρήστης.

Ακόμη, ήταν αρκετά περίπλοκη η υλοποίηση του user interface(διεπαφή χρήστη) σε συνδυασμό με την υλοποίηση του backend. Έπρεπε να υλοποιηθούν όλα τα features(δυνατότητες) ξεχωριστά και έπειτα να παρουσιαστούν τα αποτελέσματα με τον κατάλληλο τρόπο, μέσα από τη διεπαφή.

Τέλος, το project(έργο) αυτό εμπλέκει τεχνολογίες που είναι end-to-end(από άκρη σε άκρη). Χρησιμοποιήσαμε διαφορετικές τεχνολογίες για το crawling, το scraping, το frontend(μπροστά μέρος μιας εφαρμογής), το backend, τη βάση δεδομένων κτλ, τις οποίες μετά έπρεπε να συντονίσουμε και να ενώσουμε.

Το project αυτό επιλέχθηκε και υλοποιήθηκε για να δώσει αξία στο χρήστη, να τον διευκολύνει και να επιταχύνει τον τρόπο με τον οποίο πραγματοποιεί τις αναζητήσεις του. Δημιουργώντας κάτι χρήσιμο, και εύχρηστο ταυτόχρονα, για τον άνθρωπο, μάθαμε να υλοποιούμε scrapers και crawlers, που χρησιμοποιούνται πολύ σε όλες τις

σύγχρονες εφαρμογές/ μηχανές αναζήτησης. Επιπλέον, ο συνδυασμός αυτών των τεχνολογιών μαζί με τη διεπαφή χρήστη, παρέχει μία πλήρως λειτουργική εφαρμογή που μπορεί να χρησιμοποιήσει οποιοσδήποτε θέλει να βρει κάτι γρήγορα και στοχευμένα ανάμεσα σε χιλιάδες συνταγές.

Η δημιουργία και χρήση της βάσης δεδομένων και του REST API, είναι γνώσεις υψίστης σημασίας και μέσα από το project αυτό μας δόθηκε η ευκαιρία να τις ενισχύσουμε.

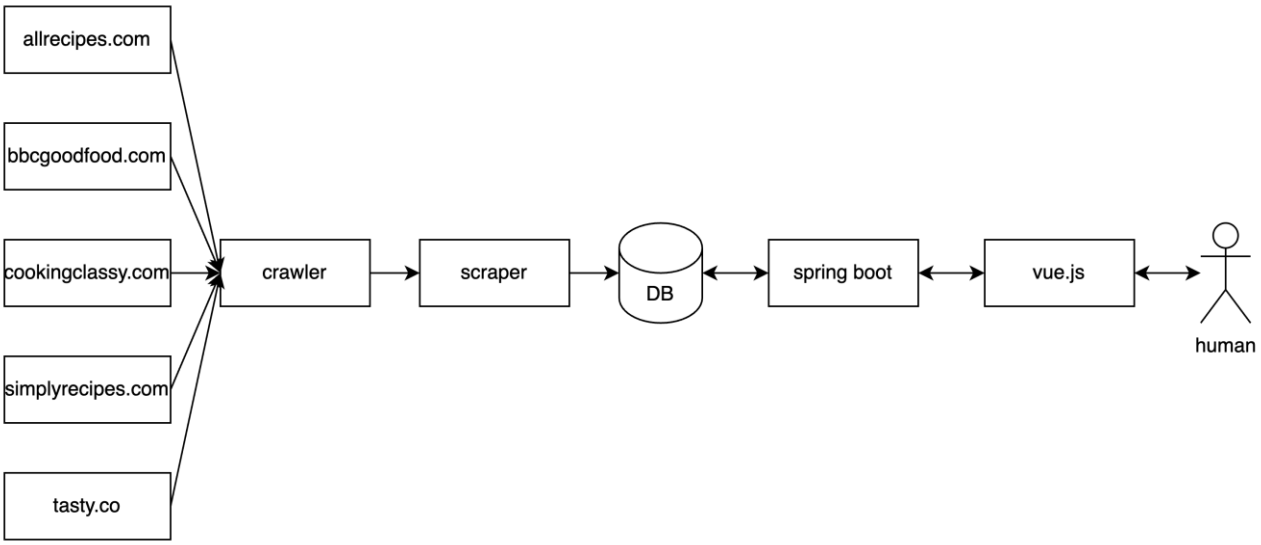
Αρχικά χρειάστηκε προετοιμασία και αναζήτησή για το ποιες τεχνολογίες και εργαλεία υπάρχουν διαθέσιμα, να τα δοκιμάσουμε και να τα συγκρίνουμε μεταξύ τους. Χρειάστηκαν γνώσεις HTML και κατανόηση του πώς κάνουμε extract τα data(απόσπαση δεδομένων) από τις σελίδες, χρειάστηκαν κάποιες βασικές γνώσεις σχεδιασμού βάσεων δεδομένων και φυσικά κατανόηση των REST αρχιτεκτονικών. Έπρεπε να συγκρίνουμε ποια εργαλεία ήταν διαθέσιμα για να υλοποιήσουμε ένα REST backend (e.g. Laravel, Django etc), από τα οποία τελικά εμείς καταλήξαμε στο “Spring Boot”. Τέλος χρειάστηκε μελέτη και κατανόηση των frontend τεχνολογιών και να σχεδιάσουμε το user interface. Συνολικά για την ολοκλήρωση του project, χρειαστήκαμε γνώσεις γύρω από scrapers, crawlers, REST API [3], frontend [4] και άλλες τεχνολογίες. Επιπλέον χρειαστήκαμε μία βάση δεδομένων, για τα 5 μεγάλα web sites στα οποία βασιστήκαμε και σαρώσαμε δεδομένα. Ακόμη, χρειαστήκαμε ένα template site(πρότυπο ιστοσελίδας) το οποίο χρησιμοποιήσαμε ως βάση για τη δημιουργία της διεπαφής χρήστη.

Τα 5 sites από τα οποία ανακτούμε συνταγές είναι τα παρακάτω:

- allrecipes.com
- bbcgoodfood.com
- tasty.co
- cookingclassy.com
- simplyrecipes.com

1.3 Ροή Πληροφορίας

Το γενικό σχήμα ροής της πληροφορίας είναι το παρακάτω:



Σχήμα 1: Το γενικό σχήμα ροής της πληροφορίας

2. ΕΡΓΑΛΕΙΑ ΚΑΙ ΤΕΧΝΟΛΟΓΙΕΣ

2.1 Διαθέσιμα εργαλεία και τεχνολογίες

2.1.1 Βάση Δεδομένων

Υπάρχουν διάφοροι τύποι βάσεων δεδομένων και το ποιος είναι ο καταλληλότερος εξαρτάται κάθε φορά από τις απαιτήσεις και ανάγκες του κάθε project.

Ενδεικτικά υπάρχουν οι εξής:

1. Αντικειμενοστραφής βάση δεδομένων: Σε μια αντικειμενοστραφή βάση δεδομένων, οι πληροφορίες αποθηκεύονται με τρόπο όμοιο με αντικείμενα.

2. Σχεσιακή βάση δεδομένων [5]: Μια σχεσιακή βάση δεδομένων είναι βασισμένη σε πίνακες όπου κάθε bit δεδομένων έχει μια σύνδεση με κάθε άλλο bit δεδομένων.

3. Μη σχεσιακή ή NoSQL βάση δεδομένων [6]: Μια βάση δεδομένων χωρίς SQL χρησιμοποιεί μια ποικιλία μορφών, όπως έγγραφα, γραφήματα, μεγάλες στήλες κτλ, γεγονός που προσφέρει μεγάλη ευελιξία και επεκτασιμότητα σε ένα σχεδιασμό βάσης δεδομένων.

Παρόλα αυτά οι βάσεις δεδομένων είναι ευρέως διαχωρισμένες σε δύο μεγάλες κατηγορίες, τις Σχεσιακές Βάσεις Δεδομένων (Relational or Sequence Databases) και τις Μη Σχεσιακές Βάσεις Δεδομένων(Non-relational/Non-sequence databases/No SQL databases). Οποιοσδήποτε μπορεί να τις χρησιμοποιήσει ξεχωριστά και σε συνδυασμό μεταξύ τους, ανάλογα από τη φύση των δεδομένων και τη λειτουργικότητα που απαιτείται.

Οι σχεσιακές βάσεις δεδομένων είναι ο πιο συνηθισμένος τύπος βάσης. Χρησιμοποιεί ένα σχήμα(schema) το οποίο είναι το πρότυπο(template) που χρησιμοποιείται για να καθορίσει τη δομή των δεδομένων που θα αποθηκευτούν στη βάση.

2.1.2 Scrapers

Για τη δημιουργία των scrapers υπάρχουν πολλά διαθέσιμα εργαλεία. Τα πιο γνωστά είναι:

- το Scrapy: Το Scrapy είναι ένα δωρεάν και ανοιχτού κώδικα framework(δομή) ανίχνευσης ιστού (web-crawling) γραμμένο σε Python. Αρχικά σχεδιασμένο για web scraping, μπορεί επίσης να χρησιμοποιηθεί για εξαγωγή δεδομένων χρησιμοποιώντας API ή ως web crawler γενικής χρήσης.
- το Octoparse: Το Octoparse είναι μια cloud-based(βασισμένη στο «σύννεφο») λύση εξαγωγής δεδομένων ιστού που βοηθά τους χρήστες να εξάγουν σχετικές πληροφορίες από διάφορους τύπους ιστότοπων. Επιτρέπει σε χρήστες να ανακτούν μη δομημένα δεδομένα και να τα αποθηκεύουν σε διαφορετικές μορφές, όπως Excel, απλό κείμενο και HTML.
- το BeautifulSoup: Το Beautiful Soup είναι ένα πακέτο Python για την ανάλυση(parsing) εγγράφων HTML και XML (συμπεριλαμβανομένης της ύπαρξης λανθασμένης σήμανσης, δηλαδή μη κλειστών ετικετών). Δημιουργεί ένα δέντρο ανάλυσης(parse tree) για αναλυμένες σελίδες που μπορεί να χρησιμοποιηθεί για την εξαγωγή δεδομένων από HTML, που είναι χρήσιμο για το web scraping.
- το Selenium: Το Selenium είναι ένα έργο “ομπρέλα” ανοιχτού κώδικα για μια σειρά εργαλείων και βιβλιοθηκών που στοχεύουν στην υποστήριξη της αυτοματοποίησης του προγράμματος περιήγησης. Παρέχει ένα εργαλείο αναπαραγωγής για τη σύνταξη λειτουργικών δοκιμών στα περισσότερα σύγχρονα προγράμματα περιήγησης ιστού, χωρίς την ανάγκη εκμάθησης μιας δοκιμαστικής γλώσσας δέσμης ενεργειών (Selenium IDE).
- και το Jsoup: Το jsoup είναι μια βιβλιοθήκη Java ανοιχτού κώδικα που έχει σχεδιαστεί για την ανάλυση, εξαγωγή και χειρισμό δεδομένων που είναι αποθηκευμένα σε έγγραφα HTML.

2.1.3 Crawlers

Για τη δημιουργία των crawlers υπάρχουν πολλά διαθέσιμα εργαλεία.

Τα πιο γνωστά είναι:

- το Scrapy [7]: όπως περιγράφηκε παραπάνω.
- το Apache Nutch [8]: Το Apache Nutch είναι ένα εξαιρετικά μεγάλο, αναλυτικό και επεκτάσιμο έργο λογισμικού ανιχνευτή ιστού(web crawler) ανοιχτού κώδικα.
- το Heritrix [9]: Το Heritrix είναι ένας ανιχνευτής ιστού(web crawler) σχεδιασμένος για αρχειοθέτηση ιστού(web archiving). Γράφτηκε από το Internet Archive. Είναι διαθέσιμο με άδεια ελεύθερου λογισμικού και γραμμένο σε Java.

2.1.4 Rest API

Για τη δημιουργία του REST API υπάρχουν πολλά διαθέσιμα εργαλεία. Τα πιο γνωστά είναι:

- Spring Boot(Java) [10]: Το Spring Boot είναι ένα framework ανοιχτού κώδικα που βασίζεται σε Java που χρησιμοποιείται για τη δημιουργία ενός micro Service(μικρο υπηρεσίας api). Αναπτύχθηκε από την Pivotal Team και χρησιμοποιείται για την κατασκευή αυτόνομων και έτοιμων για παραγωγή spring εφαρμογών.
- Laravel [11]: Το Laravel είναι ένα δωρεάν PHP web framework και ανοιχτού κώδικα, που δημιουργήθηκε από τον Taylor Otwell και προορίζεται για την ανάπτυξη διαδικτυακών εφαρμογών σύμφωνα με το αρχιτεκτονικό μοτίβο model–view–controller (MVC).
- Django(Python) [12]: Το Django είναι ένα δωρεάν και ανοιχτού κώδικα web framework βασισμένο σε Python που ακολουθεί το αρχιτεκτονικό μοτίβο model–template–views (MTV).

2.1.5 User Interface

Για τη δημιουργία των user interface υπάρχουν πολλά διαθέσιμα εργαλεία. Τα πιο γνωστά είναι:

- React [13]: Η React (γνωστή και ως React.js ή ReactJS) είναι μια δωρεάν και ανοιχτού κώδικα βιβλιοθήκη JavaScript front-end για τη δημιουργία διεπαφών χρήστη με βάση UI components(συστατικά). Συντηρείται από τη Meta (πρώην Facebook) και μια κοινότητα μεμονωμένων προγραμματιστών και εταιρειών.
- Vue.js [14]: Η Vue.js (κοινώς αναφέρεται ως Vue, προφέρεται "προβολή") είναι ένα JavaScript framework ανοιχτού κώδικα για τη δημιουργία διεπαφών χρήστη και εφαρμογών μιας σελίδας.
- Angular [15]: Η Angular (αναφέρεται επίσης ως "Angular 2+") είναι ένα web application framework δωρεάν και ανοιχτού κώδικα που βασίζεται σε TypeScript, το οποίο διευθύνεται από την Angular Team της Google και από μια κοινότητα ατόμων και εταιρειών.

2.2 Εργαλεία και τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση και οι λόγοι που επιλέχθηκαν

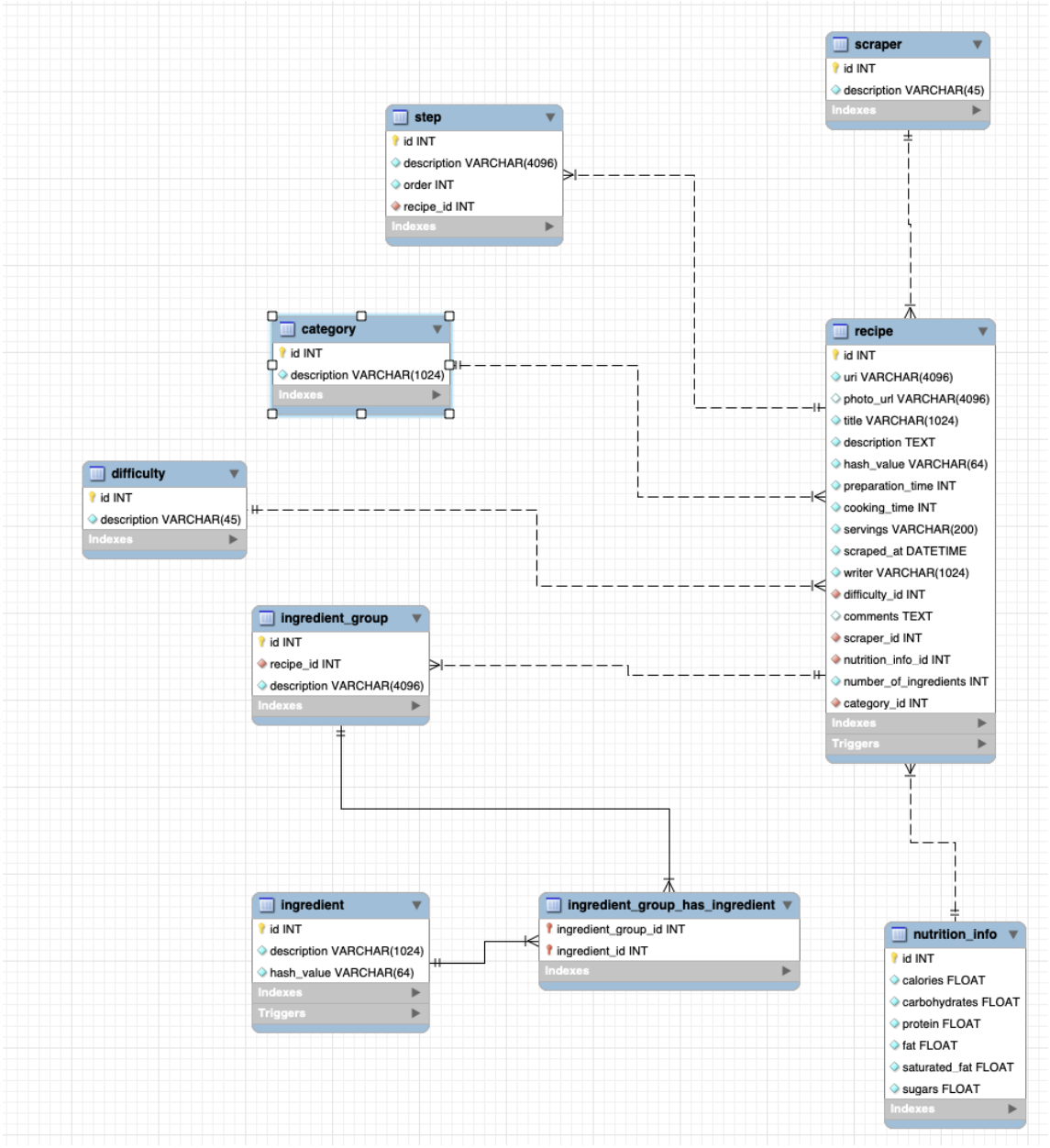
2.2.1 Βάση Δεδομένων

Όσον αφορά τη βάση δεδομένων, η επιλογή ήταν ανάμεσα σε MySQL[16] και PostgreSQL[17] διότι αυτές ήταν δωρεάν και είχαμε εξοικειωθεί με τη χρήση των sql βάσεων μέσα από μαθήματα του τμήματος. Στο συγκεκριμένο project θεωρήσαμε πως δε θα είχαν μεγάλη διαφορά και πως ήταν και οι δύο εξίσου κατάλληλες. Οι βάσεις αυτές είναι και οι δύο δωρεάν και δεν υπήρχε λόγος στο πλαίσιο αυτής της υλοποίησης να χρησιμοποιηθεί κάποια άλλη, καθώς αυτές υπερκαλύπτουν τις ανάγκες μας.

Ένας από τους λόγους που προτιμήσαμε τη MySQL, είναι πως έχει ευκολότερο deployment(ανάπτυξη). Πιο συγκεκριμένα, ήταν πολύ εύκολο να βρούμε δωρεάν δοκιμαστικούς servers(διακομιστές) στο διαδίκτυο και είχαμε και ήδη εμπειρία μέσα από τα μαθήματα του τμήματος, τόσο στην ίδια τη MySQL όσο και στο εργαλείο “MySQL Workbench” που χρησιμοποιήσαμε. Επομένως, επωφεληθήκαμε των γνώσεων που είχαμε ήδη για να προχωρήσει ομαλότερα και γρηγορότερα η ανάπτυξη της εφαρμογής.

Το “MySQL Workbench” [18], είναι ένα πολύ βολικό και εύχρηστο εργαλείο που έχουμε δει μέσα από διαφορά μαθήματα της σχολής και θα ήταν κρίμα να μην το εκμεταλλευτούμε. Επομένως, η επιλογή έγινε για πρακτικούς λόγους, παρόλο που τα εργαλεία ήταν τελείως ισοδύναμα μπροστά σε αυτό που θέλαμε να υλοποιήσουμε.

Η εικόνα της βάσης δεδομένων σε διάγραμμα είναι όπως φαίνεται παρακάτω:



Σχήμα 2: Σχήμα Βάσης Δεδομένων

Πιο συγκεκριμένα, η βάση περιέχει τους εξής πίνακες:

recipe:

Περιλαμβάνει μια σειρά από δεδομένα της κάθε συνταγής και συνδέεται με σχέσεις με όλους τους άλλους πίνακες εκτός του ingredient και ingredient_group_has_ingredient, με τα οποία συνδέεται έμμεσα μέσω του ingredient_group. Με τους πίνακες step και ingredient_group συνδέεται με σχέση 1-N (δηλαδή μία συνταγή μπορεί να έχει πολλά βήματα/step και πολλές ομάδες συστατικών/ingredient_group). Με τους υπόλοιπους πίνακες συνδέεται με σχέση N-1, δηλαδή η καταχώριση κάθε πίνακα(πχ στον difficulty, scraper κτλ) μπορεί να αντιστοιχεί σε πάνω από μία συνταγή και πρωτεύον κλειδί είναι το id.

scraper:

Περιλαμβάνει ένα id(ακέραιος αριθμός, auto increment) και μία περιγραφή(VARCHAR4096), που περιλαμβάνει το όνομα του scraper που χρησιμοποιήθηκε για τη σάρωση και εισαγωγή στη βάση της συγκεκριμένης συνταγής. Κανένα από τα δύο πεδία δε μπορεί να είναι κενό(null) και πρωτεύον κλειδί είναι το id.

step:

Περιλαμβάνει ένα id(ακέραιος αριθμός, auto increment), μία περιγραφή(VARCHAR45) που αποτελεί το κείμενο του βήματος, ένα order(ακέραιος αριθμός) που δηλώνει τη σειρά του βήματος μέσα στη συνταγή και ένα recipe_id(ακέραιος αριθμός) που δηλώνει τη συνταγή στην οποία αντιστοιχεί το βήμα. Κανένα από τα πεδία δε μπορεί να είναι κενό(null) και πρωτεύον κλειδί είναι το id. Η περιγραφή της κατηγορίας(description) είναι υποχρεωτικά μοναδική (unique).

category:

Περιλαμβάνει ένα id(ακέραιος αριθμός, auto increment) και μία περιγραφή(VARCHAR1024), που περιλαμβάνει το όνομα της κατηγορίας της συγκεκριμένης συνταγής. Κανένα από τα δύο πεδία δε μπορεί να είναι κενό(null) και πρωτεύον κλειδί είναι το id. Η περιγραφή της κατηγορίας(description) είναι υποχρεωτικά μοναδική (unique).

difficulty:

Περιλαμβάνει ένα id(ακέραιος αριθμός, auto increment) και μία περιγραφή(VARCHAR45), που περιλαμβάνει το όνομα της δυσκολίας της συγκεκριμένης συνταγής. Κανένα από τα δύο πεδία δε μπορεί να είναι κενό(null) και πρωτεύον κλειδί είναι το id. Η περιγραφή της δυσκολίας(description) είναι υποχρεωτικά μοναδική (unique).

ingredient_group:

Περιλαμβάνει ένα id(ακέραιος αριθμός, auto increment), μία περιγραφή(VARCHAR4096) που αποτελεί το όνομα της ομάδας συστατικών και ένα recipe_id(ακέραιος αριθμός) που δηλώνει τη συνταγή στην οποία αντιστοιχεί το βήμα. Κανένα από τα πεδία δε μπορεί να είναι κενό(null) και πρωτεύον κλειδί είναι το id. Η περιγραφή της κατηγορίας(description) είναι υποχρεωτικά μοναδική (unique).

ingredient:

Περιλαμβάνει ένα id(ακέραιος αριθμός, auto increment), μία περιγραφή(VARCHAR1024) που αποτελεί το όνομα του συστατικού και ένα hash_value(VARCHAR64) που δηλώνει τη συνταγή στην οποία αντιστοιχεί το βήμα. Κανένα από τα πεδία δε μπορεί να είναι κενό(null) και πρωτεύον κλειδί είναι το id. Η περιγραφή της κατηγορίας(description) είναι υποχρεωτικά μοναδική (unique).

i**ngredient_group_has_ingredient:**

Περιλαμβάνει ένα ingredient_group_id(ακέραιος αριθμός) που είναι πρωτεύον κλειδί και το παίρνει από τον πίνακα ingredient_group, ένα ingredient_id(ακέραιος αριθμός) που είναι πρωτεύον κλειδί και το παίρνει από τον πίνακα ingredient. Κανένα από τα πεδία δε μπορεί να είναι κενό(null). Η σχέση του πίνακα με τον ingredient_group είναι 1-N, δηλαδή μία ομάδα υλικών μπορεί να έχει πολλά υλικά.

nutrition_info:

Περιλαμβάνει ένα id(ακέραιος αριθμός, auto increment) καθώς και τις διατροφικές πληροφορίες της συνταγής. Πιο συγκεκριμένα, περιλαμβάνει calories(float), carbohydrates(float), protein(float), fat(float), saturated_fat(float), sugars(float). Καθένα από αυτά είναι υποχρεωτικά μοναδικό και πρωτεύον κλειδί είναι το id.

Επιπρόσθετα, χρησιμοποιήθηκαν 2 triggers:

1. Στον πίνακα “ingredient” χρησιμοποιήθηκε πριν την εισαγωγή του στοιχείου(BEFORE INSERT trigger)[19], το οποίο κάνει hashing το στοιχείο με τον αλγόριθμο MD5[20].

2. Στον πίνακα “recipe” χρησιμοποιήθηκε πριν την εισαγωγή του στοιχείου(BEFORE INSERT trigger), το οποίο κάνει hashing το στοιχείο με τον αλγόριθμο MD5.

2.2.2 Scrapers

Για τον scraper χρησιμοποιήσαμε μια βιβλιοθήκη στη Java, την “Jsoup” [21]. Εδώ δώσαμε προτεραιότητα στην ταχύτητα του scraping και για αυτό προτιμήσαμε εργαλείο Java έναντι κάποιου σε Python. Παρόλο που και το κομμάτι του scraping, όπως και

αυτό του crawling, γίνεται ασύγχρονα, η βιβλιοθήκη Jsoup ήταν αρκετά εύχρηστη, όποτε προτιμήθηκε έναντι κάποιας σε Python που θα ήταν σίγουρα και πιο αργή. Η υλοποίηση σε Java ήταν αρκετά πιο φλύαρη από ό,τι θα ήταν ενδεχομένως σε Python, που θα υλοποιούσαμε το ίδιο σε λιγότερες γραμμές κώδικα, όμως το αποτέλεσμα παραμένει η υψηλότερη ταχύτητα.

Για τα παραπάνω δεν χρησιμοποιήθηκαν συγκεκριμένα Design Patterns(μοτίβα σχεδίασης), αλλά χρησιμοποιήθηκε γενικά στοιχειώδης αντικειμενοστραφής προγραμματισμός.

2.2.3 Crawlers

Για την υλοποίηση του crawler, επιλέξαμε να χρησιμοποιήσουμε το “Scrapy” που είναι εργαλείο σε Python. Ο λόγος που το προτιμήσαμε είναι η ευκολία, η απλότητα και η ταχύτητα της υλοποίησης. Για την κατασκευή ενός crawler μπορούσαμε να επιλέξουμε ανάμεσα σε διάφορα άλλα εργαλεία όπως είναι πχ το “Apache Nutch” το οποίο είναι γραμμένο σε Java.

Ωστόσο, η Python έχει εξελιχθεί πάρα πολύ και έχει πολύ καλές βιβλιοθήκες, οι οποίες παρέχουν με κάποια ελάχιστα imports(εισαγωγές) μια σειρά από δυνατότητες και λειτουργίες. Για τις αντίστοιχες δυνατότητες και λειτουργίες αν χρησιμοποιούσαμε κάποιο άλλο εργαλείο όπως το Nutch, όλη η διαδικασία θα ήταν αρκετά πιο συνθέτη και χρονοβόρα, καθώς μόνο και μόνο για τα αρχικά settings(ρυθμίσεις) και configurations(διαμορφώσεις) πρέπει να μελετήσουμε και να κατανοήσουμε έναν πυκνό όγκο του documentation(τεκμηρίωση) του εργαλείου. Παρόλο που η υλοποίηση ενός crawler χρησιμοποιώντας άλλο εργαλείο θα ήταν πολύ δυσκολότερη, αξίζει να αναφέρουμε πως ενδεχομένως η ταχύτητα του crawling αυτού καθαυτού να ήταν γρηγορότερη, αν χρησιμοποιούσαμε εργαλείο γραμμένο σε Java (όπως το Apache Nutch).

Ωστόσο, στο πλαίσιο της δίκης μας εργασίας, η ταχύτητα του χρόνου ανάπτυξης του crawler τάχθηκε πιο σημαντική από ότι η ταχύτητα εκτέλεσης, καθώς ο crawler μπορεί να τρέχει ασύγχρονα από την υπόλοιπη εφαρμογή. Πιο συγκεκριμένα, το crawling δεν εκκινείται τη στιγμή που κάνει κάποιο αίτημα ο χρήστης (κάποια αναζήτηση), αλλά γίνεται εκ των προτέρων και μπορεί να επαναληφθεί ανά πάσα στιγμή ανεξάρτητα ή και παράλληλα με την χρήση της υπόλοιπης εφαρμογής από τον άνθρωπο.

2.2.4 Rest API

Για το backend υλοποιήσαμε ένα REST API με Spring Boot. Για το REST API είχαμε να διαλέξουμε ανάμεσα σε πολλά και γνωστά εργαλεία όπως το Django που είναι σε Python, το Spring Boot που είναι σε Java, μπορούσαμε με Node.js που είναι σε Javascript, Laravel που είναι σε PHP, μπορούσαμε με .NET Core που είναι σε C# και άλλες πολλές τεχνολογίες.

Η επιλογή της Java έγινε γιατί είναι portable(«φορητή»), τρέχει σε οποιοδήποτε υπολογιστή (σε αντίθεση με τη .NET Core ας πούμε). Επίσης, η ταχύτητα εδώ παίζει πολύ σημαντικό ρόλο και για αυτό προτιμήσαμε Java, σε αντίθεση με το Django για παράδειγμα ,που είναι σε python και είναι αρκετά πιο αργό.

Τέλος ένας ακόμη λόγος, ήταν ρως το Spring Boot χρησιμοποιείται πολύ στην αγορά. Σε σχέση δηλαδή με το Django που θα έλεγε κανείς ότι είναι το ανταγωνιστικό του, η ζήτηση για Spring Boot φαίνεται να είναι πολύ μεγαλύτερη. Συνολικά, η υλοποίηση REST API ξεκινώντας από το μηδέν γίνεται αρκετά γρήγορα σε Spring Boot, το οποίο αποτελεί επίσης ένα βασικό πλεονέκτημά του.

Ωστόσο εννοείται πως και το σύνολο των άλλων γνωστών εργαλείων όπως το Laravel(PHP) ή το Django(Python) θα μπορούσαν να καλύψουν κάθε μας ανάγκη για την εργασία αυτή.

2.2.5 User Interface

Στην επιλογή του frontend, καθοριστικό ρόλο έπαιξε η ευκολία του εργαλείου. Για το λόγο αυτόν επιλέξαμε Vue.js. Στις βασικές μας επιλογές υπήρχαν η React(Meta) και η Angular(Google). Στο παρελθόν δεν είχε τύχει να αποκτήσουμε εμπειρία σε κάποια από αυτές τις τεχνολογίες, οπότε η Angular κρίθηκε πολύ χρονοβόρα στην εκμάθηση της. Η συγκεκριμένη χρησιμοποιείται από μεγάλες εταιρίες και θα απαιτούσε αρκετή εκμάθηση. Την είδαμε και τη μελετήσαμε, παρόλα αυτά για το πλαίσιο της εργασίας θεωρήθηκε μάλλον ακατάλληλη καθώς θα καθυστερούσε την ανάπτυξη σημαντικά. Η React είναι η ίσως πρώτη σε ζήτηση αυτή τη στιγμή, όμως ήταν κι αυτή αρκετά σύνθετη, οπότε επιλέξαμε τη Vue.js που ήταν η πιο απλή από τις διαθέσιμες επιλογές.

Ένας ακόμη βασικός παράγοντας και εδώ είναι φυσικά η ταχύτητα. Η Vue.js είναι η γρηγορότερη από τις επιλογές μας.

3. ΜΕΘΟΔΟΛΟΓΙΑ ΥΛΟΠΟΙΗΣΗΣ

3.1 Backend

Σα μεθοδολογία κατά την υλοποίηση μου, θα μπορούσαμε να πούμε πως τα πήγαμε από τα αριστερά προς τα δεξιά. Παρατηρώντας το διάγραμμα παραπάνω δηλαδή, ξεκινήσαμε με τους crawlers και τελειώσαμε με το frontend. Γενικά θεωρήσαμε πως όταν πραγματοποιούμε περίπλοκες εφαρμογές πολλαπλών επιπέδων όπως αυτή, βολεύει η υλοποίηση με συνεχή τρόπο (με τη σειρά).

Ξεκινώντας την υλοποίηση του crawler, πειραματιστήκαμε για αρκετό χρονικό διάστημα με το “Apache Nutch”, το οποίο όπως προαναφέραμε είναι ένα εργαλείο σε Java με πολλές δυνατότητες. Δυστυχώς το εργαλείο αυτό αποδείχτηκε πολύ σύνθετο τόσο στην κατανόηση όσο και στη χρήση του καθώς ήθελε μια σειρά από settings και configurations για κάθε λειτουργικότητα του και το documentation του, αν και πυκνό, δε φάνηκε πολύ εύχρηστο.

Μετά από αρκετό πειραματισμό και αφού είχαμε καταφέρει ένα κομμάτι του crawling, αλλά όχι όλη τη λειτουργικότητα που θέλαμε, αποφασίσαμε να αναζητήσουμε κάποιο άλλο εργαλείο για crawling.

Ουσιαστικά αυτό που είχαμε κατά νου να υλοποιήσουμε, ήταν ένας crawler, ο οποίος θα μπορεί ξεκινώντας από ένα αρχικό σύνδεσμο ενός ιστοτόπου με συνταγές(πχ allrecipes.com), να βρίσκει όλες τις υποσελίδες, φιλτράροντας και αγνοώντας όσα links δεν αφορούν συνταγές και να τα γράφει σε ένα αρχείο txt(κειμένου). Με τη χρήση του Apache Nutch δυστυχώς είχαμε καταφέρει τη μισή λειτουργικότητα αυτού και έτσι αποφασίσαμε να στραφούμε σε κάποιο εργαλείο στην Python, το οποίο θα ήταν μεν πιο αργό, αλλά θα ήταν αποτελεσματικό.

Πράγματι, μετά από έρευνα επιλέξαμε το “Scrapy” το οποίο με λίγες γραμμές κώδικα κάνει όσα ακριβώς περιγράφουμε παραπάνω, δημιουργώντας τα txt αρχεία με τα links του κάθε site, τα οποία αργότερα θα φορτώσουμε στους scrapers.

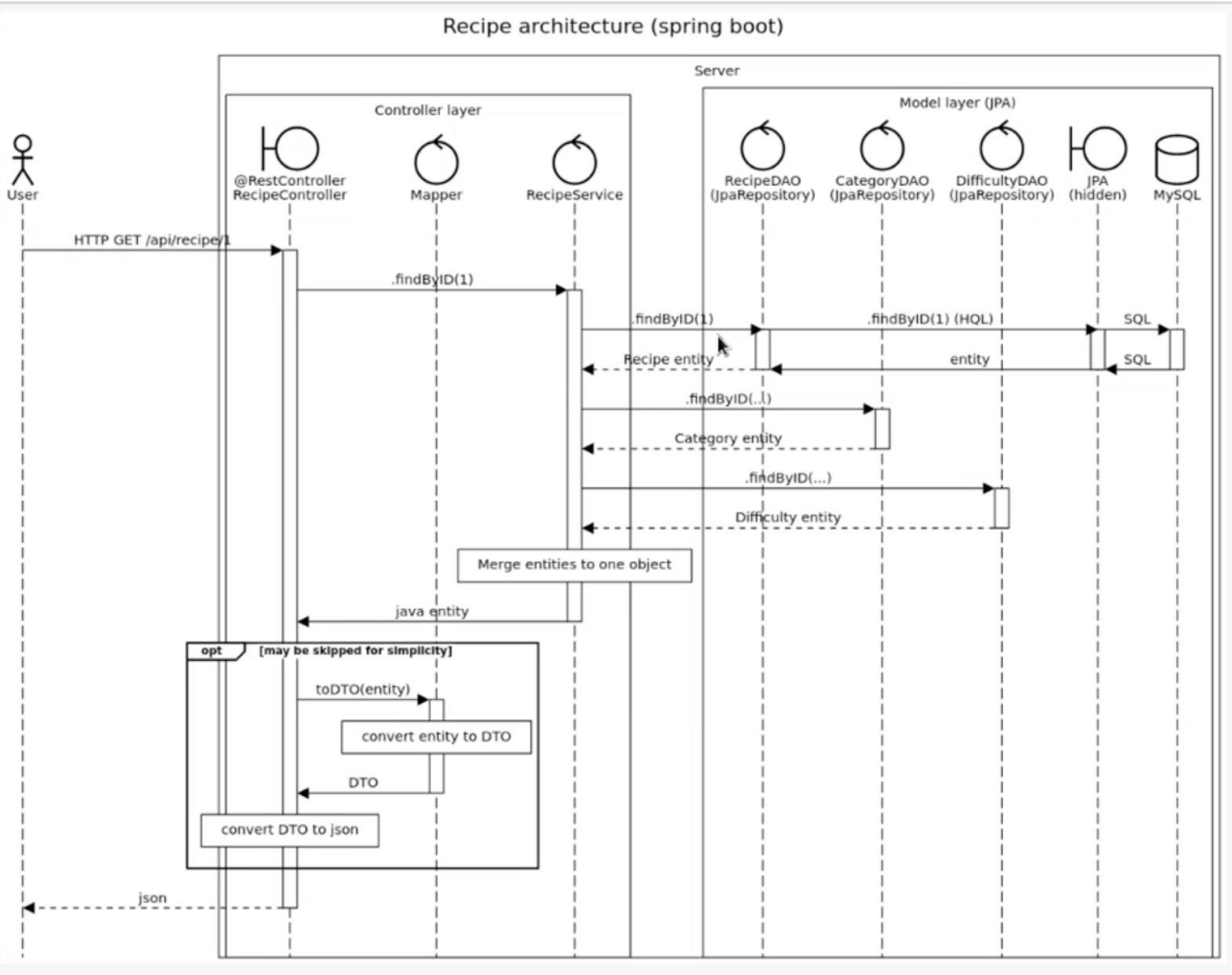
Έπειτα ξεκινήσαμε την υλοποίηση του πρώτου scraper, μελετήσαμε τις ιστοσελίδες, είδαμε τη μορφή τους και τι δεδομένα μπορούμε να εξάγουμε και γράψαμε τον κώδικα που ανακτά τα πρώτα data(δεδομένα) από το ένα site. Στο σημείο αυτό πήγαμε στο MySQL Workbench και σχεδιάσαμε τη βάση δεδομένων μας. Αφού φτιάξαμε το σχήμα και κάναμε forward engineering μέσα από το workbench, η βάση μας ήταν έτοιμη και μπορούσαμε να συνεχίσουμε την υλοποίηση των scrapers.

Υλοποιήσαμε 5 scrapers για τα 5 διαφορετικά sites συνταγών. Ο κάθε scraper διαβάζει από το αντίστοιχο αρχείο .txt του ένα-ένα τα links και από το καθένα ανακτά την επιθυμητή πληροφορία κάνοντας όλες τις κατάλληλες μετατροπές πριν τα εισαγάγει στη βάση δεδομένων, έτσι ώστε να έχουν όλα τα data από όλα τα sites όμοια μορφή.

Για την υλοποίηση του backend χρησιμοποιήσαμε Spring Boot, IntelliJ IDEA [22] και Maven [23].

3.1.1 Sequence Diagram

Ακολουθεί το Sequence Diagram της αρχιτεκτονικής του backend:



Σχήμα 3: Sequence Diagram

3.1.2 Περιγραφή αρχιτεκτονικής spring-boot

Για το σχεδιασμό του backend χρησιμοποιήσαμε τα παρακάτω:

- DAO(Data Access Object) [24]: η δουλειά τους είναι μόνο να παίρνουν data από τη database.
- Rest Controller [25]: κλάση που δέχεται τις αιτήσεις από το χρήστη(HTTP GET). Η δουλειά του είναι όταν πάρει μια αίτηση να ελέγξει αν η αίτηση είναι σωστή. Κάνει validation(επικύρωση) δηλαδή στο request(αίτημα) και επιστρέφει σφάλμα σε περίπτωση που λείπουν για παράδειγμα δεδομένα από κάποιο πεδίο, στην αναζήτηση που έχει υποχρεωτικά τιμές. Από τη στιγμή που η αίτηση περνάει, ο Rest Controller προωθεί την αίτηση στο Service.
- RecipeService: δουλειά του service είναι να πάρει αυτή την αίτηση του χρήστη και να τη στείλει στη βάση δεδομένων. Επιπλέον, αν θέλουμε να συνδυάζουμε δεδομένα από διαφορετικούς πίνακες, είναι δουλειά του να τα μαζέψει.

Πιο συγκεκριμένα, για κάθε πίνακα της database έχουμε ένα διαφορετικό DAO(data Access Object) που είναι κλάση JPA Repository του Spring Boot, δηλαδή μία διαφορετική κλάση που αναλαμβάνει την επικοινωνία με τη βάση δεδομένων. Το Service με το που θα πάρει την αίτηση καλεί ξεχωριστά τη μέθοδο του αντίστοιχου DAO για να πάρει τα αποτελέσματα και να τα ενώσει. Μόλις τα ενώσει, παίρνει αυτό το ενοποιημένο αποτέλεσμα, το στέλνει πίσω στο Rest Controller όπου ο Rest Controller πρέπει να πάρει αυτό το entity(οντότητα) και να το μετατρέψει σε DTO(data transaction object) [26][27].

Η διαφορά των entities από τα DTOs(data transaction objects):

Τα entities μας δείχνουν πώς παρουσιάζεται η πληροφορία στη βάση δεδομένων, ενώ τα DTOs είναι ουσιαστικά τα δεδομένα που ανταλλάζουμε με τον πελάτη μας. Είναι η επικοινωνία δηλαδή ανάμεσα στον controller και το user και όχι η επικοινωνία ανάμεσα στο controller και τη database. Το DTO δηλαδή αφορά το πώς παρουσιάζεται η πληροφορία στο χρήστη.

Άρα πρώτα μαζεύουμε τις πληροφορίες από τη βάση με το Service, μετά δίνουμε το Entity αυτό στον Controller. Ο Controller πλέον έχει όλη την πληροφορία από τη ΒΔ, επομένως από την πληροφορία αυτή πρέπει να αφαιρέσει πράγματα. Ο controller το στέλνει στο mapper, ο οποίος παίρνει το entity και το μετατρέπει σε DTO. Αντιγράφει δηλαδή από το entity όσα χρειάζεται και συνδυάζει όλα τα δεδομένα(αν είναι από

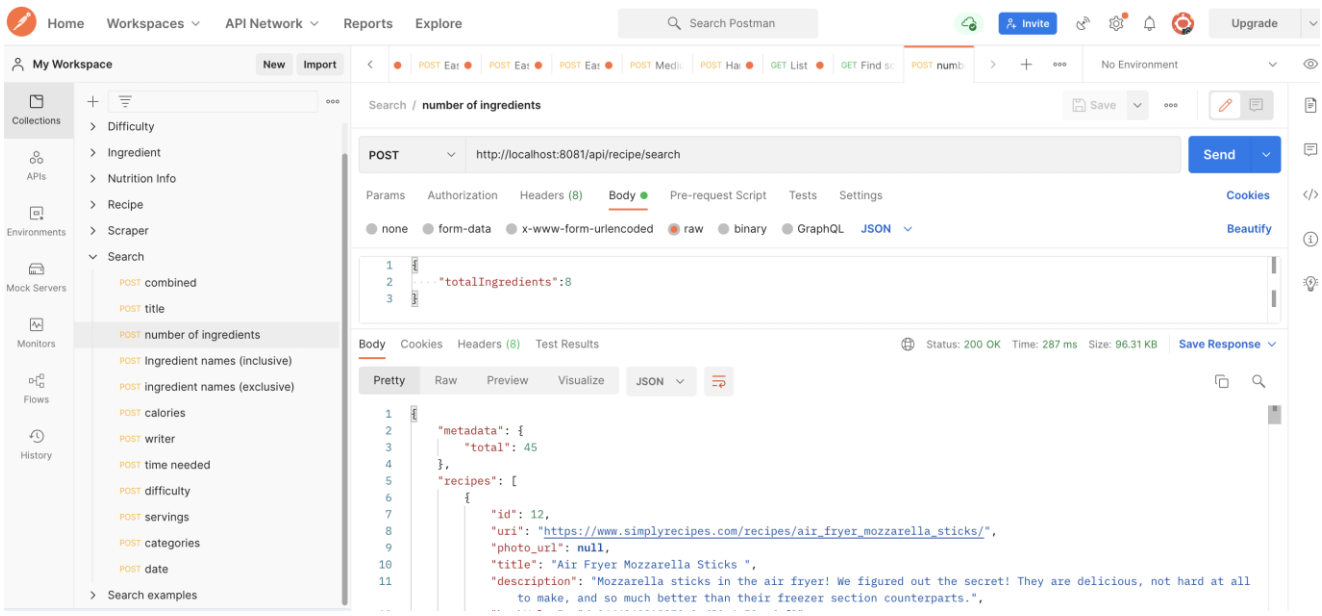
παραπάνω πίνακες) σε ένα τελικό αντικείμενο. Το αντικείμενο αυτό θα μετατραπεί σε json για να σταλεί στον πελάτη.

Οπότε μετά ο mapper το έχει μετατρέψει σε json, το δίνει πίσω στο Rest Controller, ο Rest Controller το μετατρέπει σε json και το στέλνει πίσω στον πελάτη.

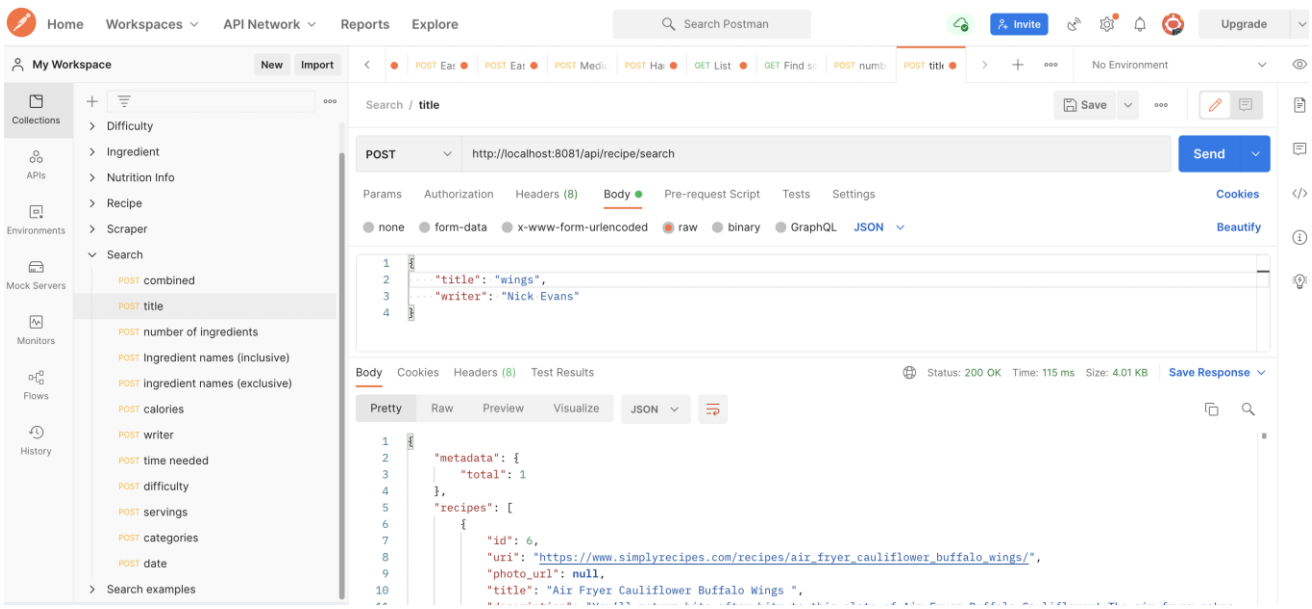
Άρα ο χρήστης στέλνει την αίτηση σε json στον controller, ο controller ελέγχει αν λείπει κάποιο πεδίο ή αν οι τιμές της αίτησης δεν είναι σωστές και αν είναι τις προωθεί στο service. Το service πηγαίνει και επικοινωνεί με τη βάση δεδομένων, παίρνει όλη την πληροφορία που χρειάζεται από έναν ή παραπάνω πίνακες και τα ενώνει όλα σε ένα τελικό αντικείμενο που έχει όλες τις πληροφορίες της database. Το service το δίνει πίσω στον controller, ο controller το δίνει στον mapper. Ο mapper αυτό το entity το μετατρέπει στο αντίστοιχο DTO όπου φιλτράρει τα δεδομένα ή και τα κάνει πιο ευανάγνωστα, αφαιρεί δεδομένα(αν υπάρχουν κάποια που δεν χρειάζονται) και μετά τα προωθεί πίσω στον controller, που τα παίρνει το κάνει json και το στέλνει στον πελάτη.

Μια από τις δυσκολίες ήταν η κατανόηση των αρχιτεκτονικών πολλαπλών επιπέδων όπως αυτή, ώστε να μπορέσουμε να χωρίσουμε τον κώδικα σωστά και να κάνει κάθε κλάση τη δουλειά της.

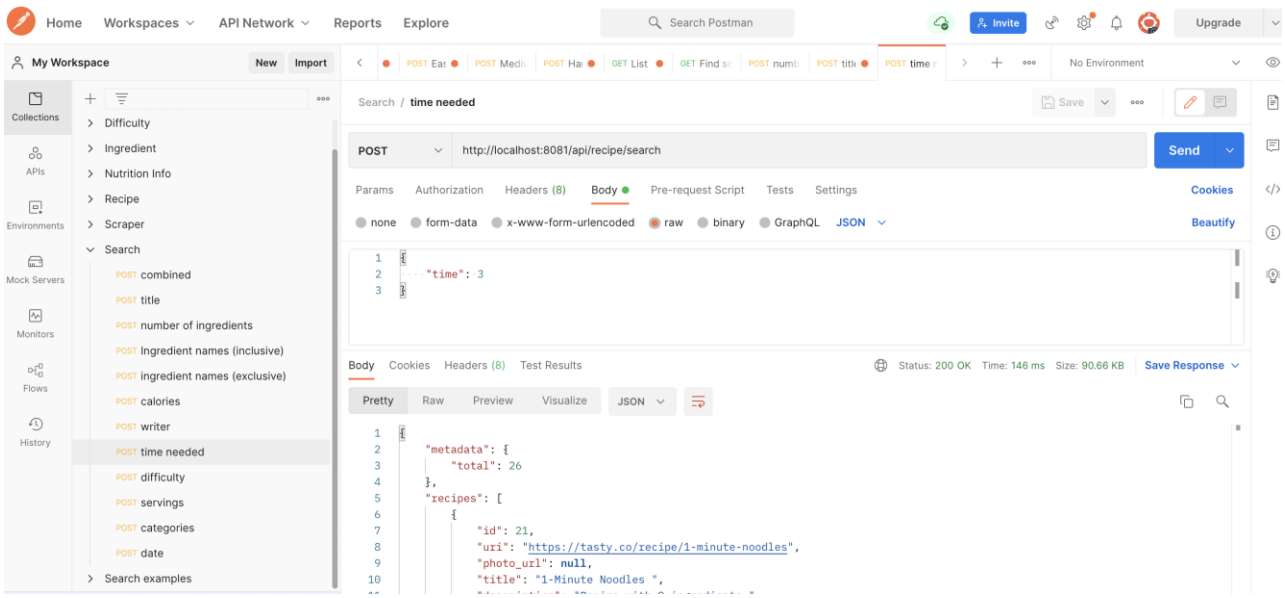
Πριν προχωρήσουμε στο frontend τεστάραμε την εφαρμογή στο postman για να βεβαιωθούμε ότι λειτουργεί σωστά. Πιο συγκεκριμένα, φτιάξαμε διάφορα test παραδείγματα για τις αναζητήσεις και ελέγξαμε αν τα αποτελέσματα που λάβαμε αντιστοιχούσαν με αυτά στη βάση μας.



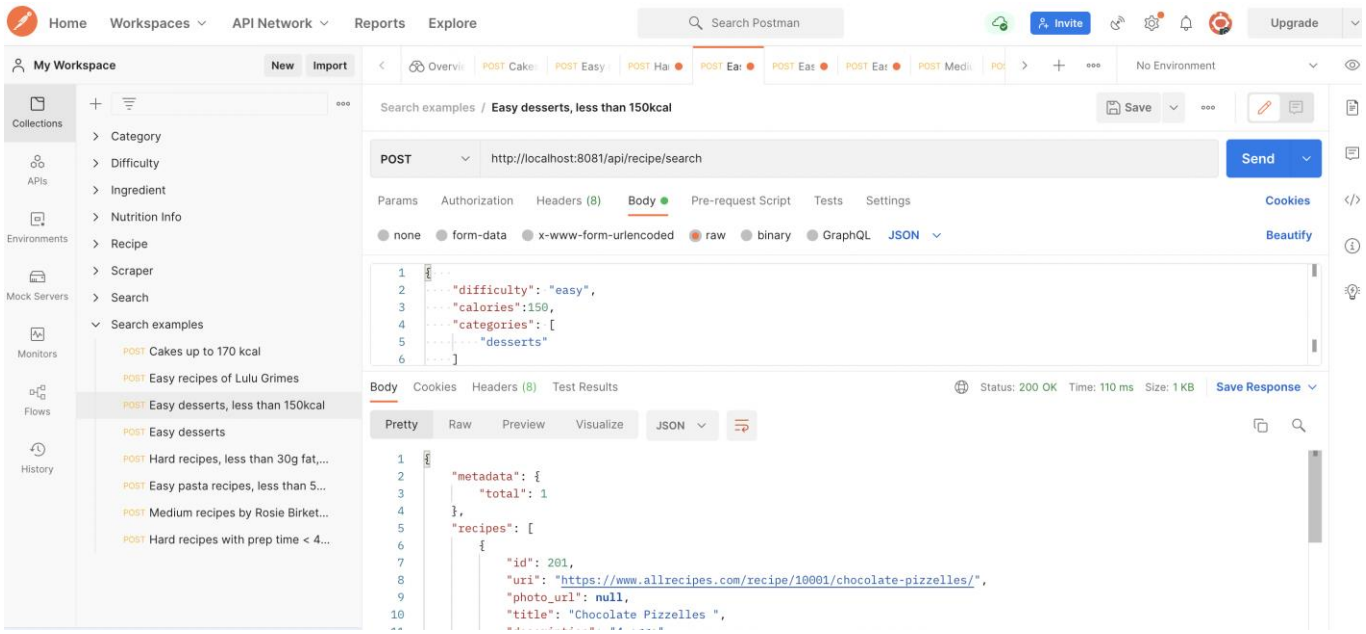
Εικόνα 3: Αναζήτηση με βάση το πλήθος των συστατικών



Εικόνα 4: Αναζήτηση με βάση τον τίτλο και τον συγγραφέα



Εικόνα 5: Αναζήτηση με βάση τον χρόνο της συνταγής



Εικόνα 6: Παράδειγμα πιο σύνθετης αναζήτησης, με βάση τις θερμίδες, την κατηγορία και τη δυσκολία

3.2 Frontend

Για την υλοποίηση του frontend χρησιμοποιήσαμε Nodejs [28], Vuejs, Vuecli [29], Visual Studio Code [30], Nuxt [31] με χρήση Options API (αντί του Composition). Επίσης για τη δημιουργία του frontend χρησιμοποιήσαμε ένα template το οποίο βρήκαμε στο διαδίκτυο και προσαρμόσαμε ώστε να παρουσιάζονται αποτελεσματικά και καλαίσθητα τα αποτελέσματα στον χρήστη. Το template είναι το “FoodHub - Food Delivery Template" [32], από αυτήν την πηγή “https://store.vuetifyjs.com/products/foodhub-food-delivery-template?utm_source=vuetifyjs.com&utm_medium=themes" και με το license “Personal Use”. Τα design patterns που χρησιμοποιήθηκαν για το development του frontend ήταν όπως αυτά προϋπήρχαν στο template. Οι τεχνολογίες που χρησιμοποιεί το template είναι οι Vue, Nuxt.js και vuetify. Επίσης για να χρησιμοποιήσουμε REST στη Vuejs χρησιμοποιήσαμε και τη βιβλιοθήκη Axios. Επίσης χρησιμοποιήσαμε Nuxt, Babel και Sass.

- Vuetify[33]: μια πολύ διαδεδομένη βιβλιοθήκη που αποτελεί ένα σύνολο από έτοιμα components.
- Babel[34]: μεταγλωττιστής(transpiler) που ελέγχει και μετατρέπει τη javascript ώστε να παίζει καταλληλά σε όλους τους browsers(προγράμματα περιήγησης).

Με τη χρήση των παραπάνω δηλαδή το site μας παίζει σε όλους τους browsers και παρουσιάζεται ωραία και σε pc(υπολογιστές) αλλά και σε κινητές συσκευές.

- Nuxt: επιτρέπει το σάιτ να σκαναριστεί εύκολα από μηχανές αναζήτησης ώστε να φαίνεται στο google.

Τα παραπάνω περιλαμβάνονταν στο template και εμείς προσθέσαμε και το nudejs/ axios το οποίο είναι μια βιβλιοθήκη της vuejs που χρησιμοποιείται για να επικοινωνούμε με το server, για να κάνουμε δηλαδή κλήσεις και να στέλνουμε τα ερωτήματα της αναζήτησης του χρήστη και να λαμβάνουμε την απάντηση.

Για τον crawler και τους scrapers δεν έχει χρησιμοποιηθεί κάποια συγκεκριμένη αρχιτεκτονική ή κάποιο συγκεκριμένο design pattern. Όσον αφορά τη βάση, έχουμε κάνει κανονικοποίηση της ΒΔ μέχρι κανονική μορφή BCNF[35].

Άλλα υποστηρικτικά εργαλεία που χρησιμοποιήσαμε:

Postman[36], HTTPie[37].

Για την ανάπτυξη κώδικα σε Java χρησιμοποιήσαμε το IntelliJ IDEA, για Python το Pycharm[38] και για Vue.js το Visual Studio Code.

3.3 Testing

Όσον αφορά τις δοκιμές των διάφορων κομματιών της εφαρμογής, κάναμε τα παρακάτω:

Για τον κάθε Crawler, η υλοποίηση έγινε σε Pycharm και για να το τεστάρουμε τρέξαμε όλο το crawling και ελέγξαμε ότι το txt αρχείο που παράγεται για κάθε site έχει σωστά urls. Στο σημείο αυτό χρειάστηκε πειραματισμός και μετατροπές ώστε να μην υπάρχουν πολλαπλότητα ή σύνδεσμοι του site που δεν αφορούν συνταγές.

Για τον κάθε Scraper, η υλοποίηση έγινε σε IntelliJ IDEA και για να τεστάρουμε το scraping πηγαίναμε στη βάση δεδομένων και ελέγχαμε ότι τα δεδομένα που ανακτήθηκαν για κάθε link είναι τα αναμενόμενα.

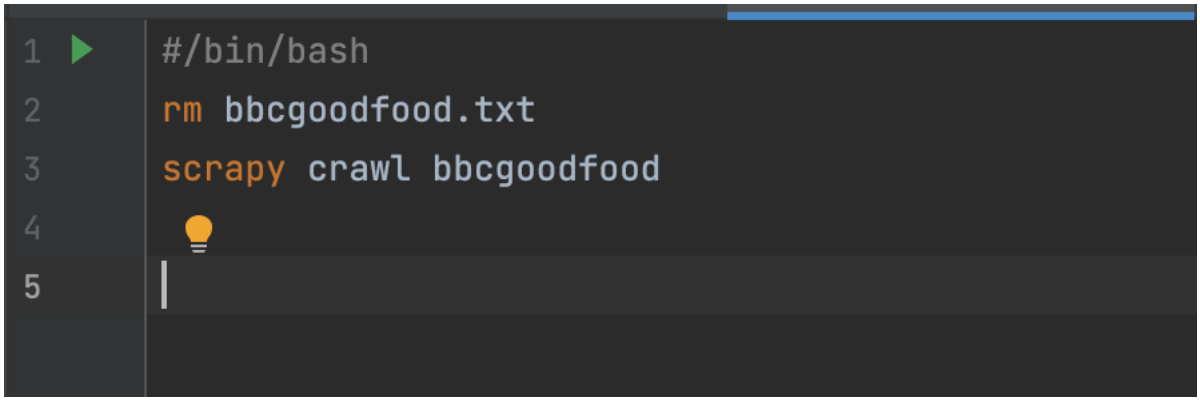
Για το backend(Spring Boot), η υλοποίηση έγινε σε Intellij IDEA και το τεστάραμε με το Postman.

Για το frontend(Vue.js), η υλοποίηση έγινε με Visual Studio Code και το τεστάραμε χειροκίνητα σαν ανθρώπινος χρήστης.

4. USER MANUAL – APPLICATION DEMO

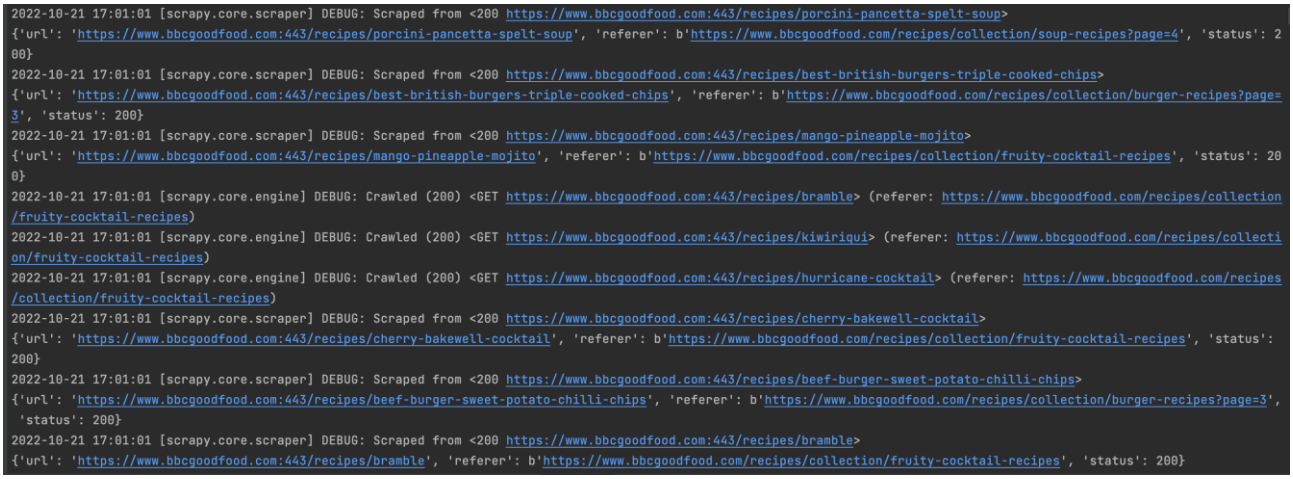
4.1 Παρουσίαση Εκτέλεσης

Για να τρέξουμε τους crawlers, έχουμε φτιάξει ξεχωριστά bash scripts αρχεία και χρησιμοποιώντας το περιβάλλον του Pycharm, πατάμε απλά Run(εκτέλεση) στο script και τρέχει ο αντίστοιχος crawler. Για παράδειγμα, για να ανακτήσουμε όλα τα links του bbcgoodfood.com, τρέχουμε το script “scrape_bbcgoodfood.sh”.



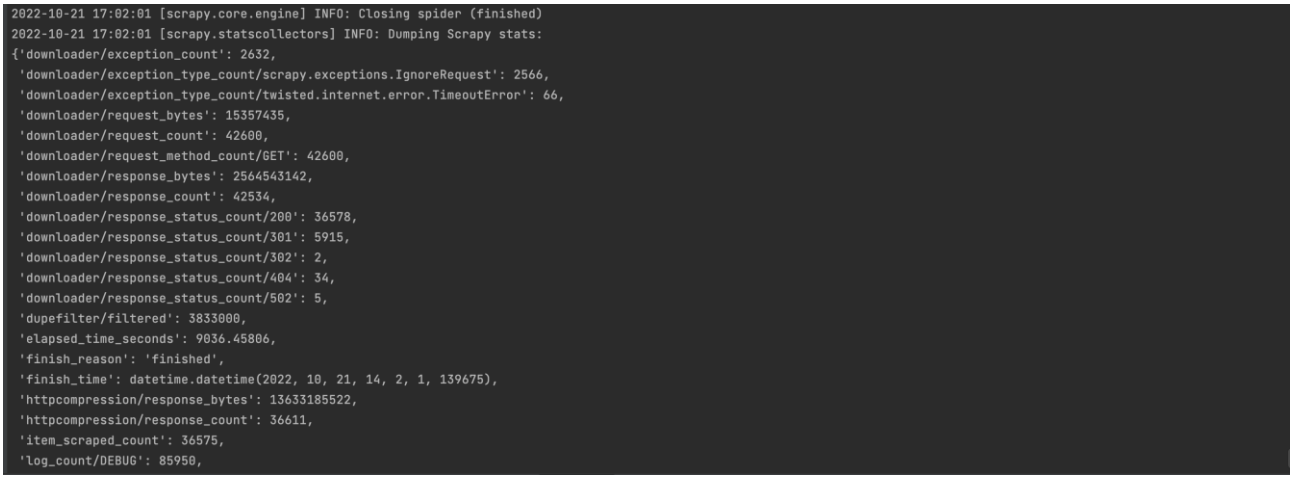
Εικόνα 7: Εκτέλεση του script για το crawling του bbcgoodfood.com

Έτσι ξεκινάει το crawling και καταγράφονται όλα τα links που πληρούν τις πορυποθέσεις μας στο “bbcgoodfood.txt”. Το crawling συνεχίζει κάπως έτσι (εικόνα από το terminal(τερματικό) του pycharm):

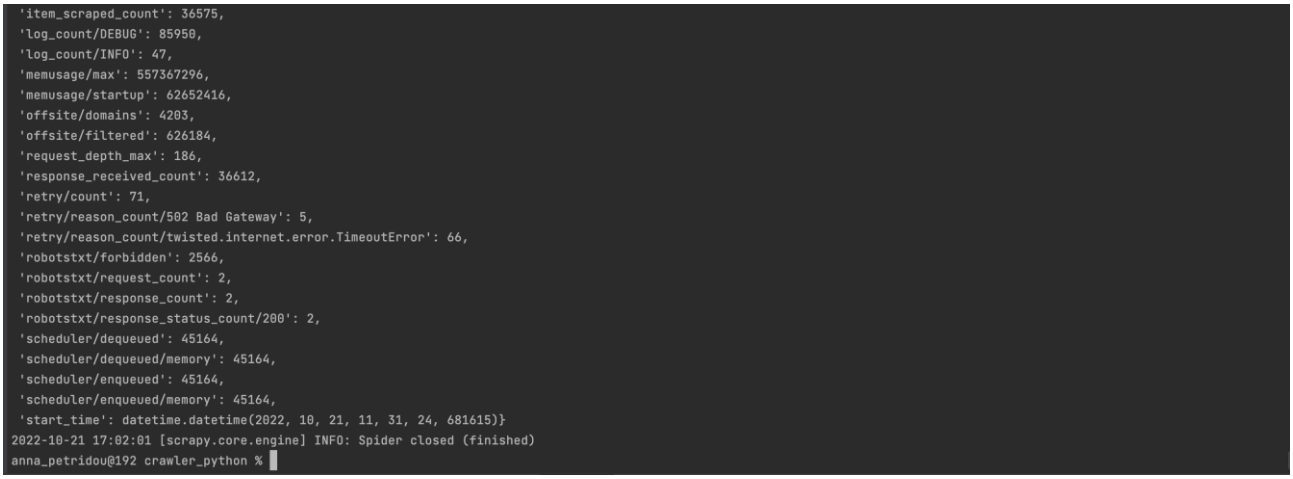


Εικόνα 8: Κατά τη διάρκεια του crawling του bbcgoodfood.com

Τέλος η εικόνα στο terminal είναι αυτή:



Εικόνα 9: Η εικόνα στο terminal μετά την ολοκλήρωση του crawling

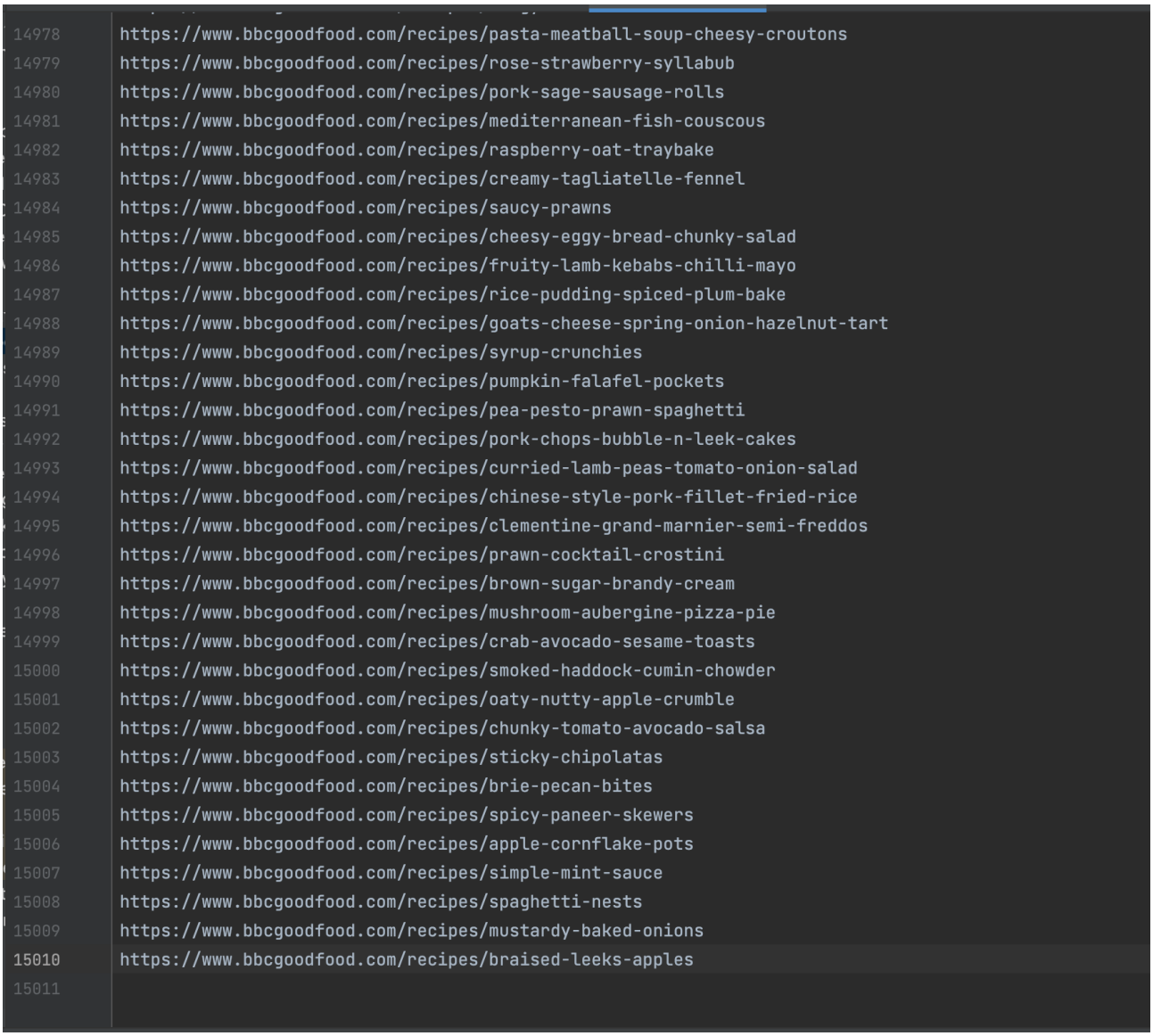


Εικόνα 10: : Η τελική εικόνα στο terminal μετά την ολοκλήρωση του crawling

Το text αρχείο δημιουργείται στον ίδιο φάκελο και είναι της μορφής:

```
https://www.bbcgoodfood.com/recipes/pork-stroganoff
https://www.bbcgoodfood.com/recipes/lamb-ragu
https://www.bbcgoodfood.com/recipes/marble-bundt-cake
https://www.bbcgoodfood.com/recipes/vegan-waffles
https://www.bbcgoodfood.com/recipes/cabbage-rolls
https://www.bbcgoodfood.com/recipes/chana-daal-chaat-with-tamarind-herbs
https://www.bbcgoodfood.com/recipes/slow-cooker-hot-chocolate
https://www.bbcgoodfood.com/recipes/slow-cooker-bio-yogurt
https://www.bbcgoodfood.com/recipes/poached-apricots-rosewater
https://www.bbcgoodfood.com/recipes/slow-cooker-spiced-root-lentil-casserole
https://www.bbcgoodfood.com/recipes/madeira-roast-quince-fool
https://www.bbcgoodfood.com/recipes/membrillo-glazed-halloumi-skewers
https://www.bbcgoodfood.com/recipes/venison-quince
https://www.bbcgoodfood.com/recipes/membrillo-chorizo-cheddar-toastie
https://www.bbcgoodfood.com/recipes/toffee-apple-bread-butter-pudding
https://www.bbcgoodfood.com/recipes/open-face-pear-berry-pie
https://www.bbcgoodfood.com/recipes/squidgy-chocolate-pear-pudding
https://www.bbcgoodfood.com/recipes/roasted-aubergine-salad
https://www.bbcgoodfood.com/recipes/sweetcorn-chowder
https://www.bbcgoodfood.com/recipes/coconut-flour-cookies
https://www.bbcgoodfood.com/recipes/fat-rascals
https://www.bbcgoodfood.com/recipes/pear-almond-pavlova-trifle
https://www.bbcgoodfood.com/recipes/maple-pear-pecan-mascarpone-roulade
https://www.bbcgoodfood.com/recipes/chocolate-coconut-cake
https://www.bbcgoodfood.com/recipes/chicken-sausage-pasta
https://www.bbcgoodfood.com/recipes/salted-caramel-apple-pudding
https://www.bbcgoodfood.com/recipes/chicken-leek-brown-rice-stir-fry
https://www.bbcgoodfood.com/recipes/orecchiette-with-butter-beans-parsley-chilli-lemon
https://www.bbcgoodfood.com/recipes/5-day-tagine
https://www.bbcgoodfood.com/recipes/spinach-pancakes-with-harissa-yogurt-poached-eggs
https://www.bbcgoodfood.com/recipes/smoked-trout-tartlets
https://www.bbcgoodfood.com/recipes/lamb-with-olive-herb-stuffing
https://www.bbcgoodfood.com/recipes/celeriac-pancetta-thyme-soup
```

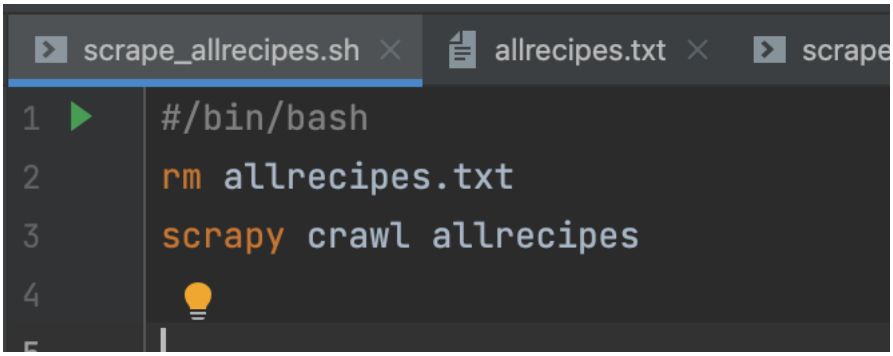
Εικόνα 11: Τα links που ανακτήθηκαν για το bbcgoodfood.com



Εικόνα 12: Συνολικά για αυτό τον ιστότοπο ανακτήθηκαν 15010 σύνδεσμοι.

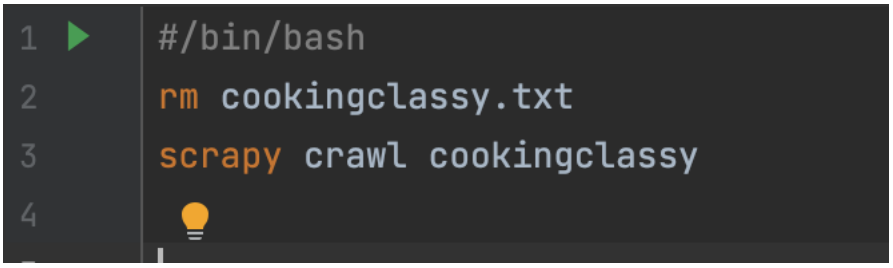
Το συγκεκριμένο txt αρχείο περιλαμβάνει συνολικά 15010 links από συνταγές.
Με τα αντίστοιχα scripts τρέχουμε και τους άλλους 4 crawlers όπως φαίνεται παρακάτω:

- scrape_allrecipes.sh



Εικόνα 13: Εκτέλεση του script για το crawling του allrecipes.com

- scrape_cookingclassy.sh

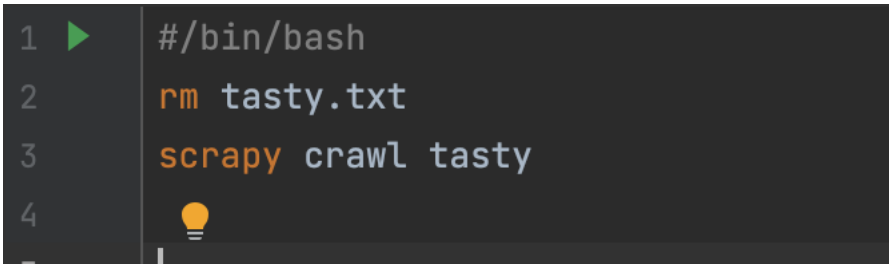


Εικόνα 14: Εκτέλεση του script για το crawling του cookingclassy.com

- scrape_simplyrecipes.sh

Εικόνα 15: Εκτέλεση του script για το crawling του simplyrecipes.com

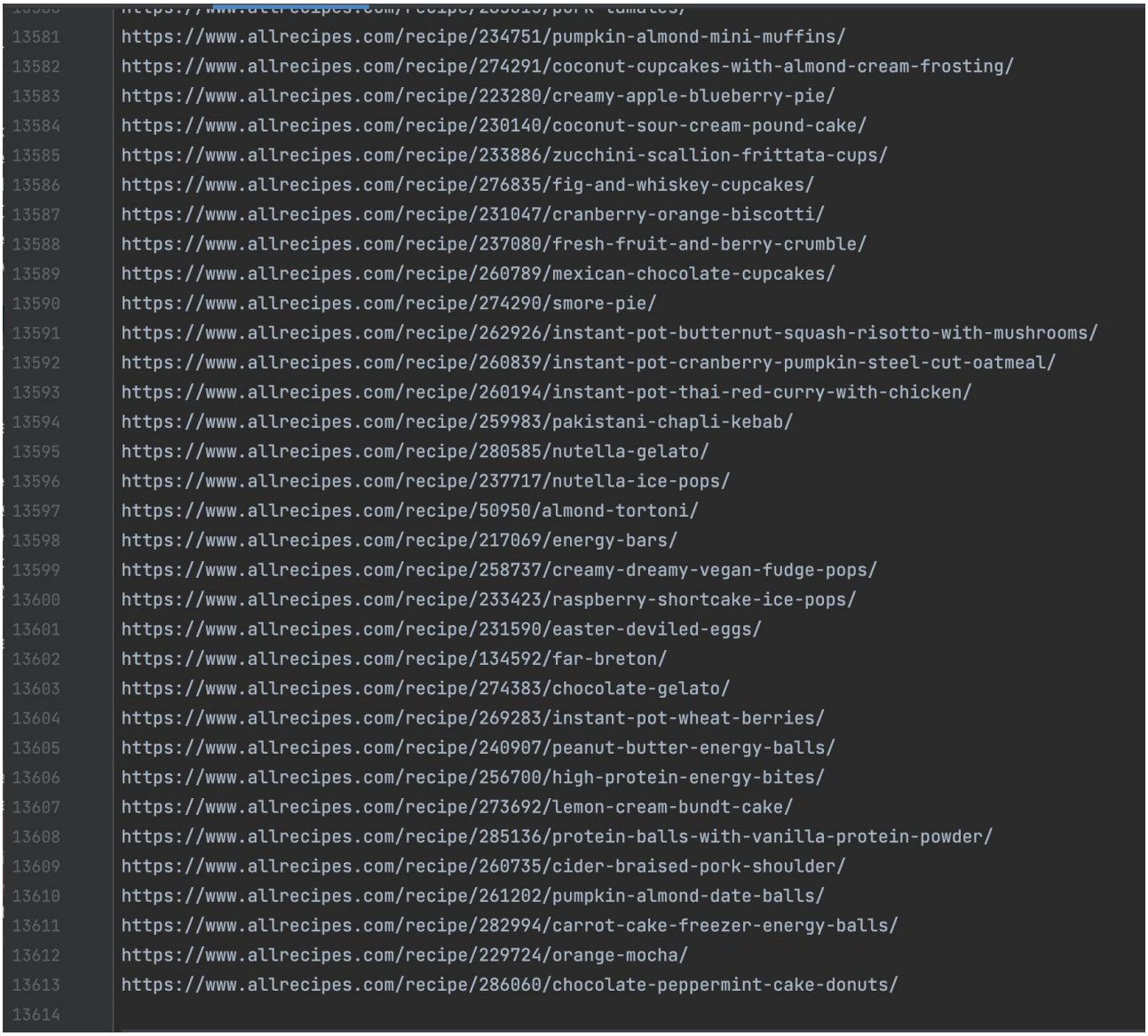
- scrape_tasty.sh



Εικόνα 16: Εκτέλεση του script για το crawling του tasty.co

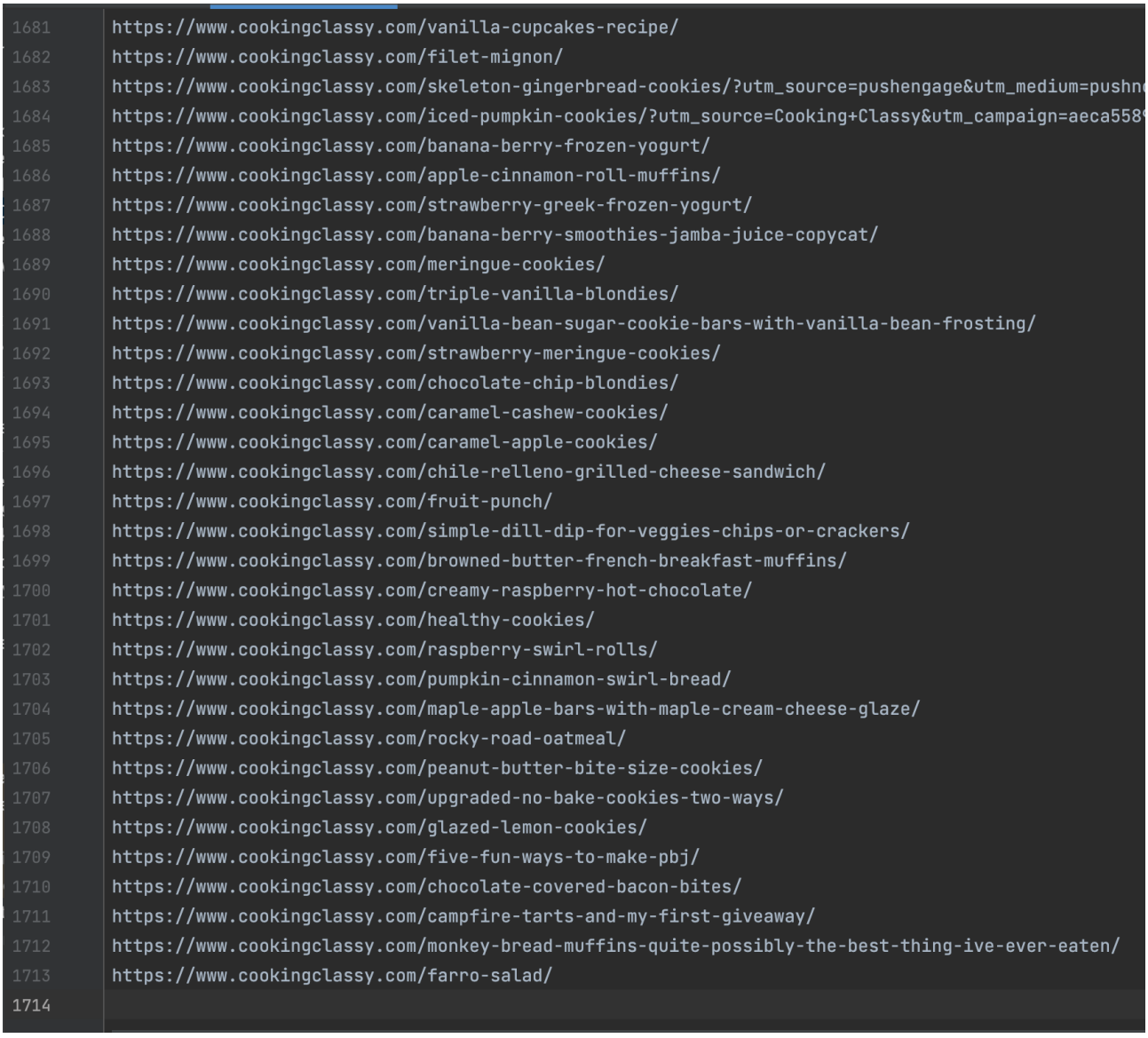
Τα αντίστοιχα txt αρχεία τους είναι της μορφής:

Για το allrecipes.txt, ανακτήθηκαν 13613 links.



Εικόνα 17: Συνολικά για αυτό τον ιστότοπο ανακτήθηκαν 13613 σύνδεσμοι.

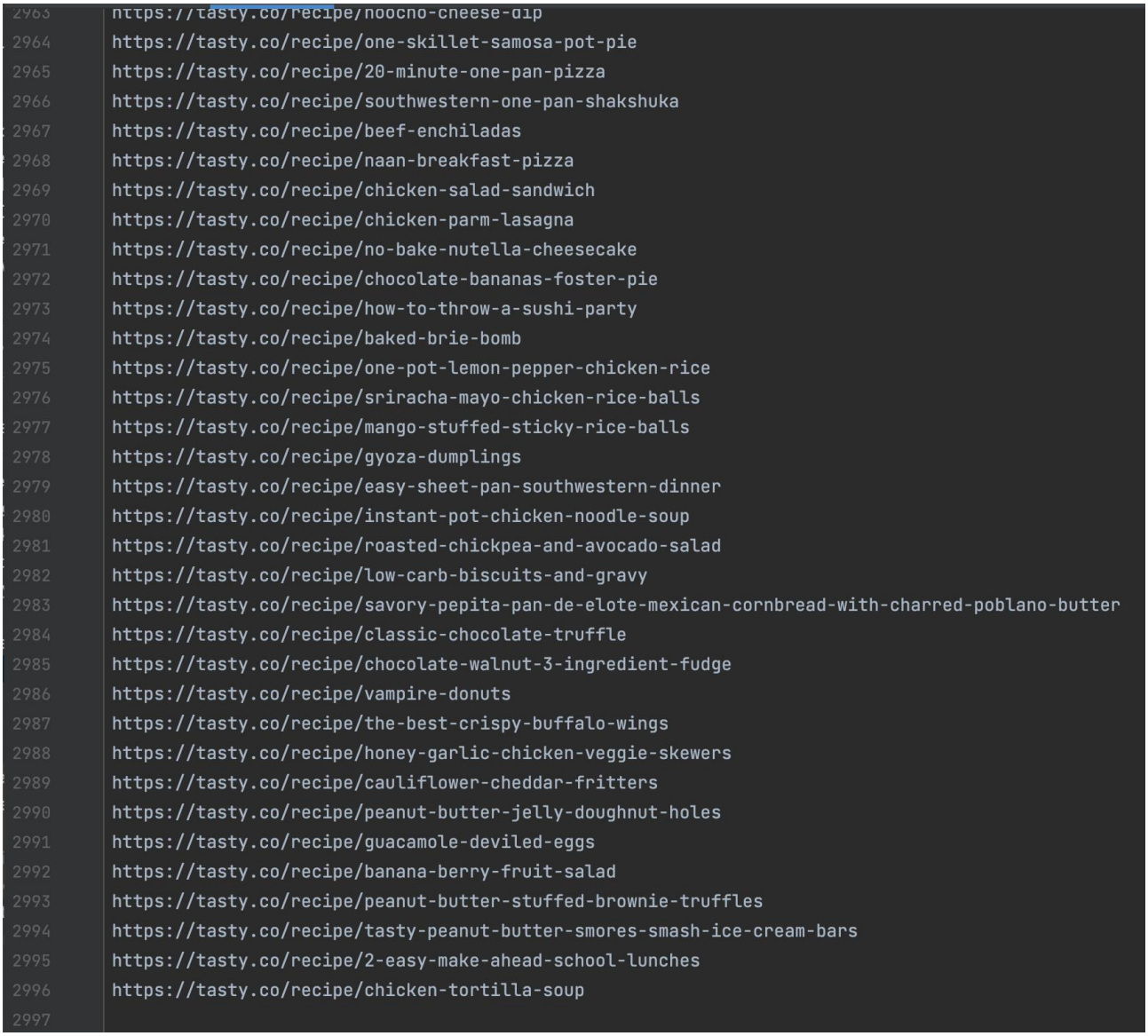
Για το cookingclassy.com ανακτήθηκαν 1713 links:



```
https://www.cookingclassy.com/vanilla-cupcakes-recipe/
https://www.cookingclassy.com/filet-mignon/
https://www.cookingclassy.com/skeleton-gingerbread-cookies/?utm_source=pushengage&utm_medium=pushn
https://www.cookingclassy.com/iced-pumpkin-cookies/?utm_source=Cooking+Classy&utm_campaign=aeca558
https://www.cookingclassy.com/banana-berry-frozen-yogurt/
https://www.cookingclassy.com/apple-cinnamon-roll-muffins/
https://www.cookingclassy.com/strawberry-greek-frozen-yogurt/
https://www.cookingclassy.com/banana-berry-smoothies-jamba-juice-copycat/
https://www.cookingclassy.com/meringue-cookies/
https://www.cookingclassy.com/triple-vanilla-blondies/
https://www.cookingclassy.com/vanilla-bean-sugar-cookie-bars-with-vanilla-bean-frosting/
https://www.cookingclassy.com/strawberry-meringue-cookies/
https://www.cookingclassy.com/chocolate-chip-blondies/
https://www.cookingclassy.com/caramel-cashew-cookies/
https://www.cookingclassy.com/caramel-apple-cookies/
https://www.cookingclassy.com/chile-relleno-grilled-cheese-sandwich/
https://www.cookingclassy.com/fruit-punch/
https://www.cookingclassy.com/simple-dill-dip-for-veggies-chips-or-crackers/
https://www.cookingclassy.com/browned-butter-french-breakfast-muffins/
https://www.cookingclassy.com/creamy-raspberry-hot-chocolate/
https://www.cookingclassy.com/healthy-cookies/
https://www.cookingclassy.com/raspberry-swirl-rolls/
https://www.cookingclassy.com/pumpkin-cinnamon-swirl-bread/
https://www.cookingclassy.com/maple-apple-bars-with-maple-cream-cheese-glaze/
https://www.cookingclassy.com/rocky-road-oatmeal/
https://www.cookingclassy.com/peanut-butter-bite-size-cookies/
https://www.cookingclassy.com/upgraded-no-bake-cookies-two-ways/
https://www.cookingclassy.com/glazed-lemon-cookies/
https://www.cookingclassy.com/five-fun-ways-to-make-pbj/
https://www.cookingclassy.com/chocolate-covered-bacon-bites/
https://www.cookingclassy.com/campfire-tarts-and-my-first-giveaway/
https://www.cookingclassy.com/monkey-bread-muffins-quite-possibly-the-best-thing-ive-ever-eaten/
https://www.cookingclassy.com/farro-salad/
```

Εικόνα 18: Συνολικά για αυτό τον ιστότοπο ανακτήθηκαν 1713 σύνδεσμοι.

Για το tasty.co ανακτήθηκαν 2996 links:



Εικόνα 19: Συνολικά για αυτό τον ιστότοπο ανακτήθηκαν 2996 σύνδεσμοι.

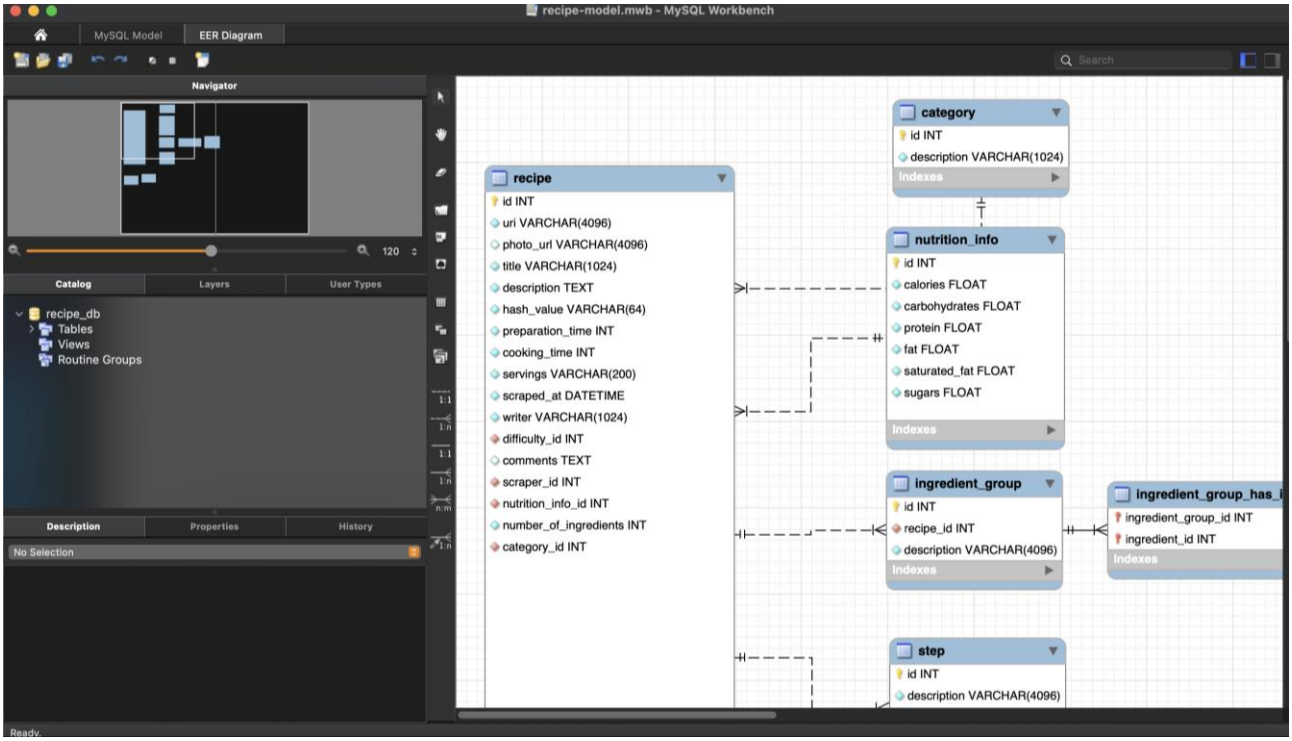
Για το simplyrecipes.com ανακτήθηκαν 2642 links:

```
https://www.simplyrecipes.com/recipes/grilled_marinated_flank_steak/
https://www.simplyrecipes.com/recipes/honey_glazed_lemon_roast_chicken/
https://www.simplyrecipes.com/recipes/how_to_dry_brine_and_roast_a_turkey/
https://www.simplyrecipes.com/recipes/cowboy_steak_with_chimichurri_sauce/
https://www.simplyrecipes.com/recipes/giant_ginger_cookies/
https://www.simplyrecipes.com/recipes/coconut_chocolate_chip_cookies/
https://www.simplyrecipes.com/recipes/rum_balls/
https://www.simplyrecipes.com/recipes/candy_cane_cookies/
https://www.simplyrecipes.com/recipes/hot_chocolate/
https://www.simplyrecipes.com/recipes/chocolate_peppermint_swiss_roll/
https://www.simplyrecipes.com/recipes/chipotle_flourless_chocolate_cake/
https://www.simplyrecipes.com/recipes/bittersweet_chocolate_cake/
https://www.simplyrecipes.com/recipes/how_to_make_whipped_cream/
https://www.simplyrecipes.com/recipes/chocolate_layer_cake/
https://www.simplyrecipes.com/recipes/kale_and_shaved_brussels_sprout_salad_with_bacon/
https://www.simplyrecipes.com/recipes/cheesy_potato_casserole/
https://www.simplyrecipes.com/recipes/steak_diane/
https://www.simplyrecipes.com/recipes/roasted_cauliflower_and_mushroom_bolognese/
https://www.simplyrecipes.com/recipes/pork_chops_with_braised_cabbage/
https://www.simplyrecipes.com/recipes/low_carb_cranberry_cooler/
https://www.simplyrecipes.com/recipes/pork_chops_with_dijon_sauce/
https://www.simplyrecipes.com/recipes/easy_smothered_pork_chops/
https://www.simplyrecipes.com/recipes/mexican_chocolate_ice_cream/
https://www.simplyrecipes.com/recipes/triple_chocolate_cheesecake/
https://www.simplyrecipes.com/recipes/chocolate_covered_pretzels/
https://www.simplyrecipes.com/recipes/christmas_cracker_candy/
https://www.simplyrecipes.com/recipes/english_toffee/
https://www.simplyrecipes.com/recipes/almond_roca/
https://www.simplyrecipes.com/recipes/chocolate_peanut_butter_bars/
https://www.simplyrecipes.com/recipes/chocolate_ganache_torte/
https://www.simplyrecipes.com/recipes/butter_pecan_cookies/
https://www.simplyrecipes.com/recipes/peppermint_bark_chocolate_cookies/
https://www.simplyrecipes.com/recipes/walnut_snowball_cookies/
https://www.simplyrecipes.com/recipes/cinnamon_snap_cookies/
```

Εικόνα 20: Συνολικά για αυτό τον ιστότοπο ανακτήθηκαν 2642 σύνδεσμοι.

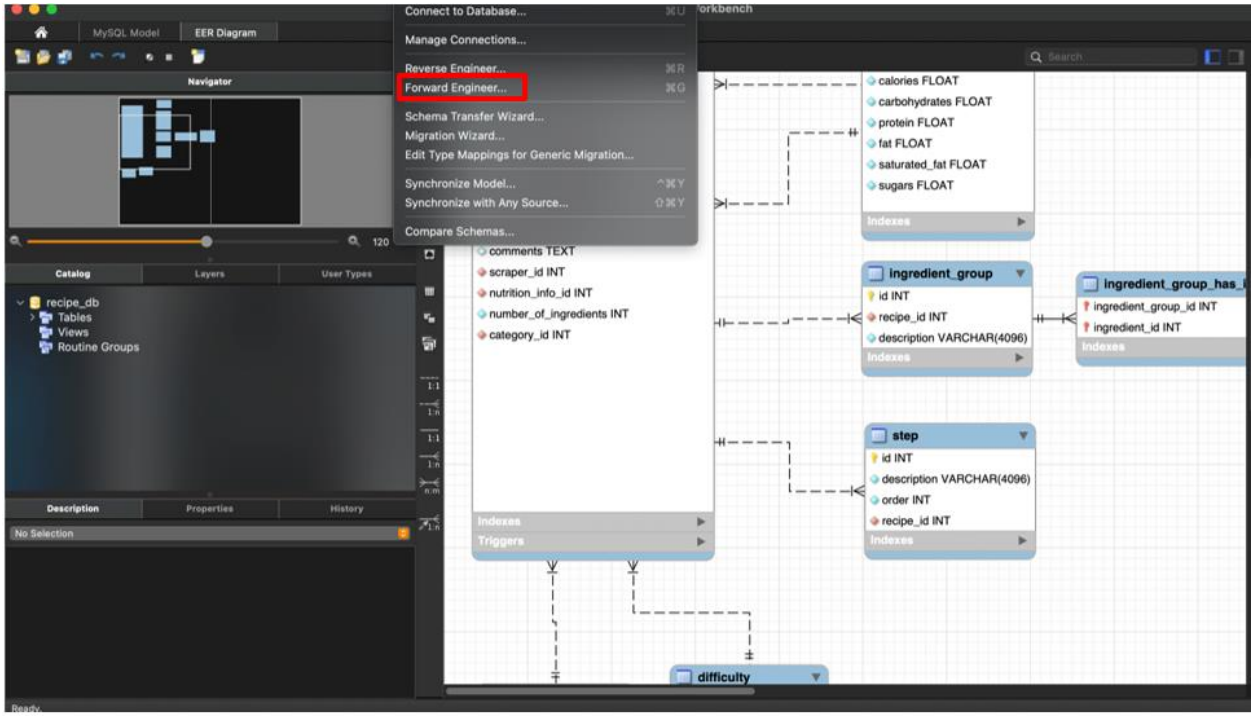
Μετά τους crawlers, προχωράμε και τρέχουμε τους scrapers. Για την εκτέλεση αυτών, πηγαίνουμε στο Intellij IDEA και κάνουμε run τη main του project στο φάκελο “scraper”. Πριν εκτελέσουμε τους scrapers, πηγαίνουμε στη βάση δεδομένων και κάνουμε forward engineering. Για τη βάση δεδομένων χρησιμοποιούμε το MySQL Workbench όπως φαίνεται παρακάτω:

Ανοίγουμε το database model που βρίσκεται στο φάκελο database και πηγαίνουμε στο workbench και επιλέγουμε “Database”.



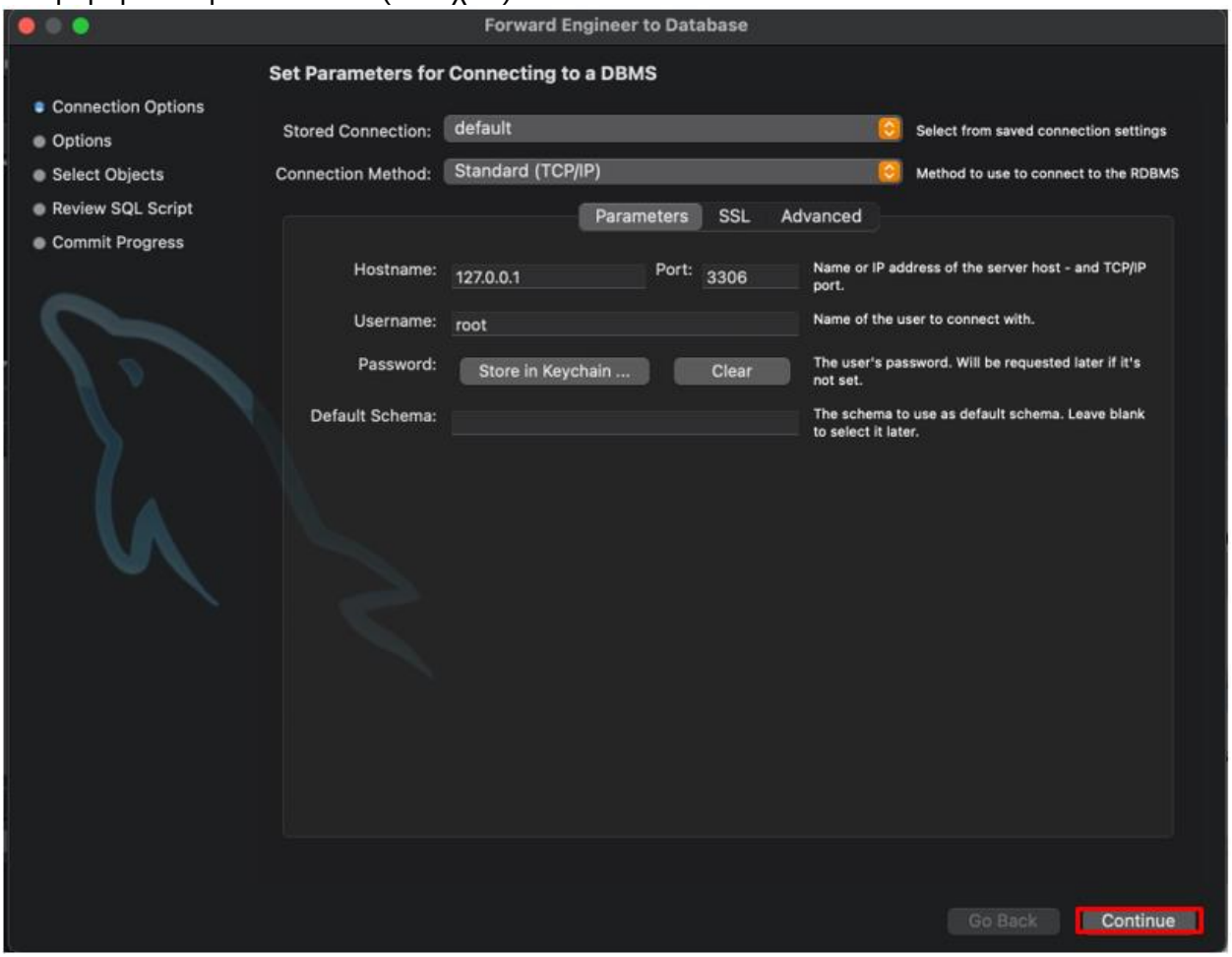
Εικόνα 21: Database model μέσα στο MySQL Workbench

Από το dropdown menu επιλέγουμε το “forward engineer” για να αρχικοποιήσουμε τη βάση δεδομένων.



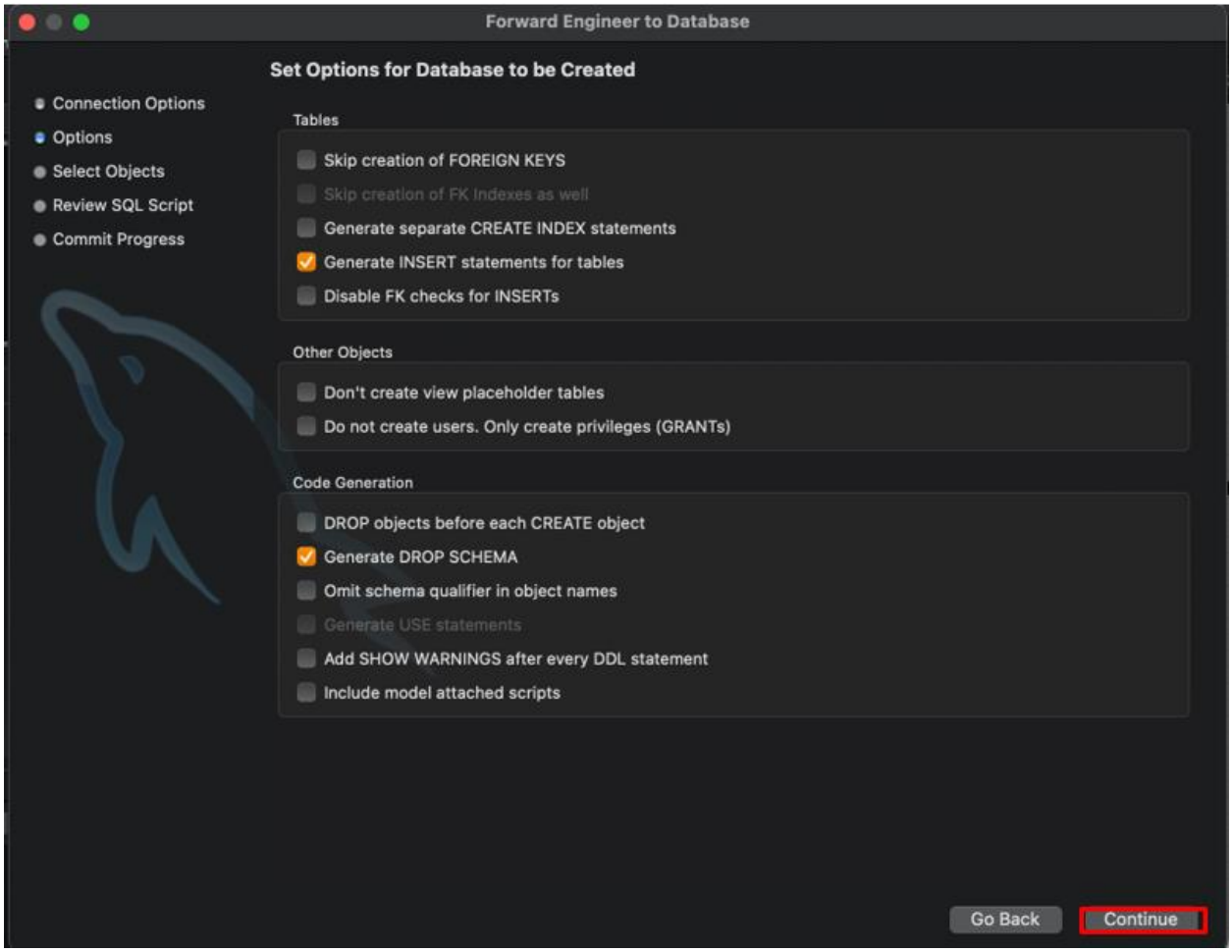
Εικόνα 22: Κλικ στο “Forward Engineer”

Στο popup πατάμε “Continue”(συνέχεια).

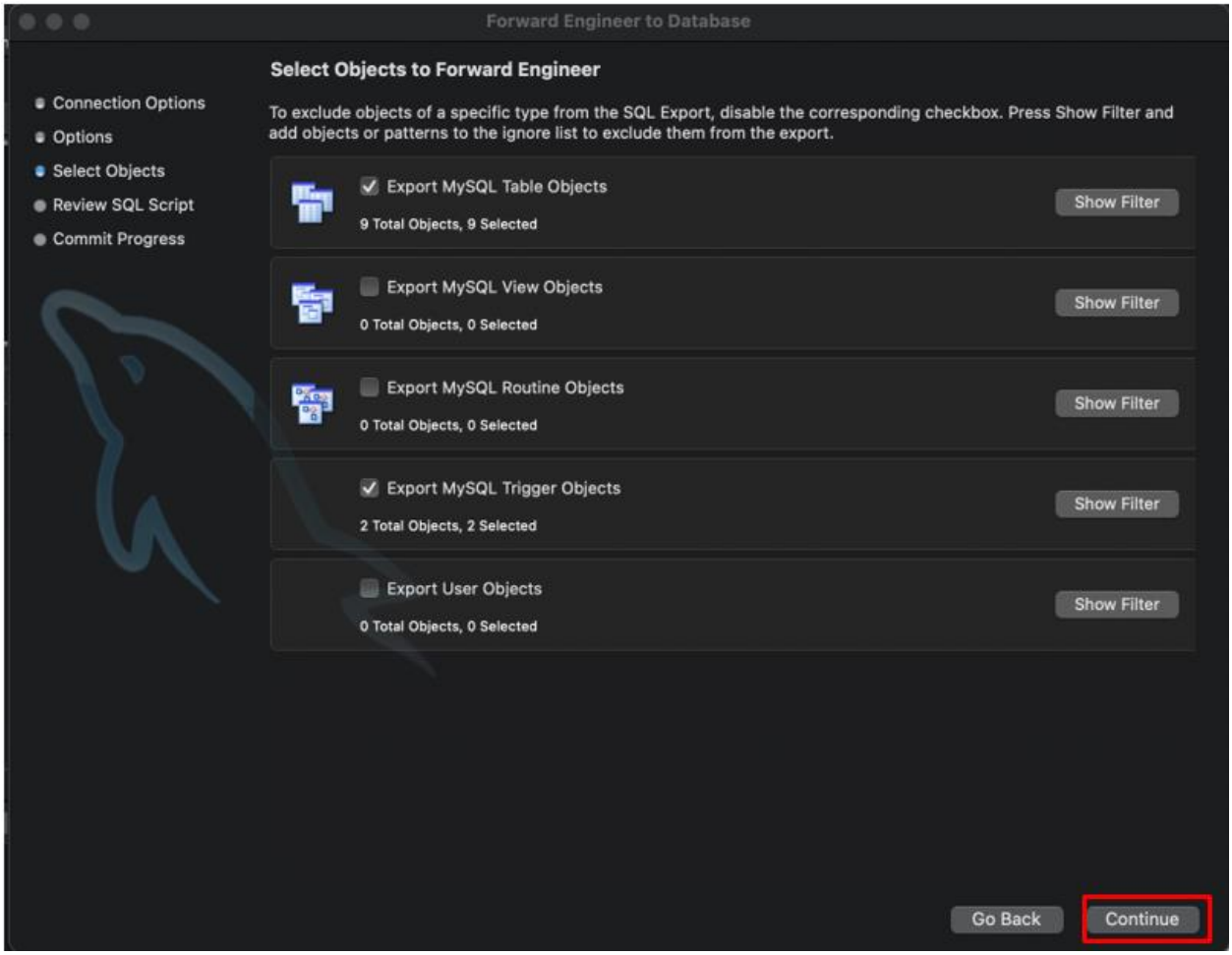


Εικόνα 23: Κλικ στο “Continue”

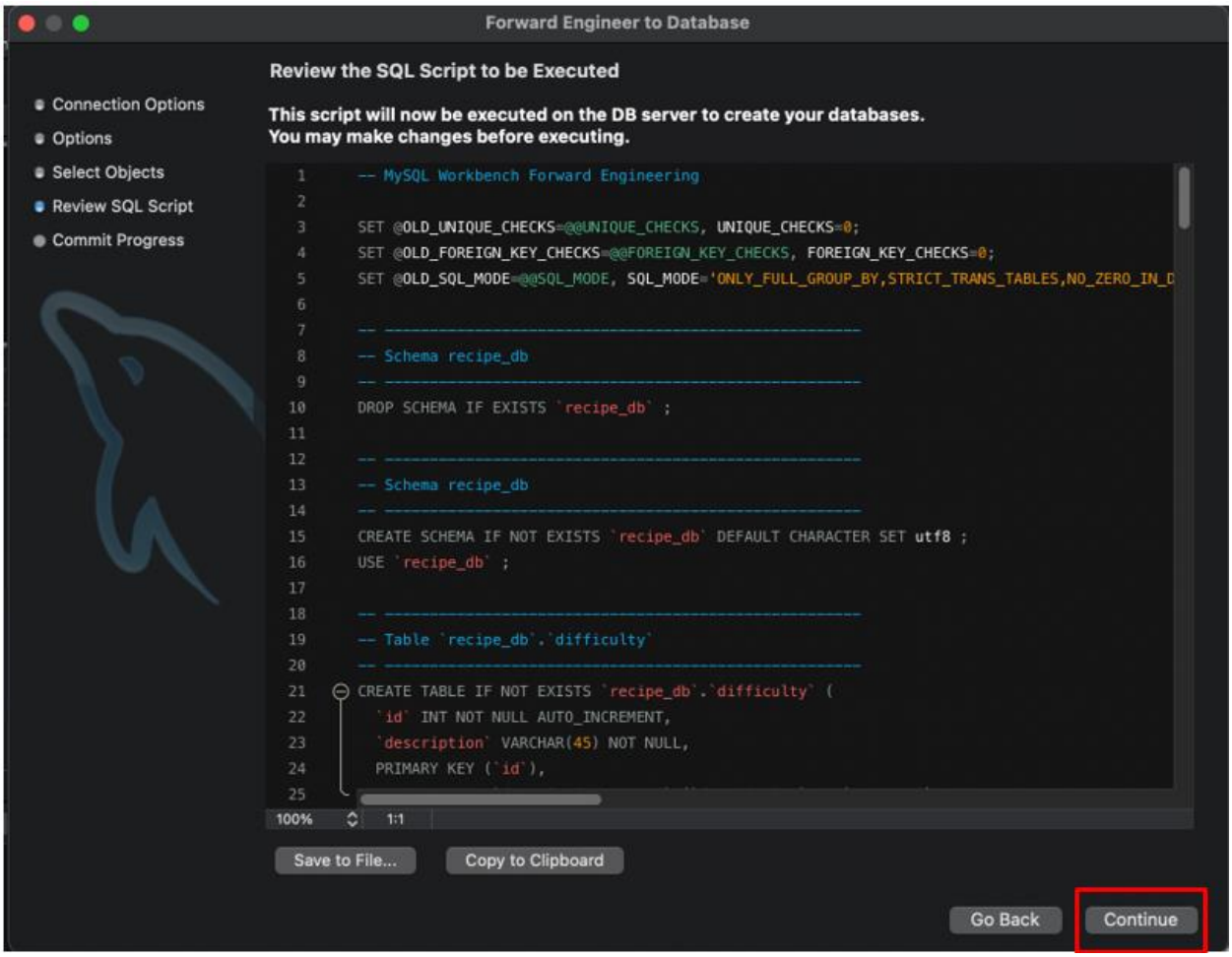
Και ξανά continue σε όλα τα επόμενα.



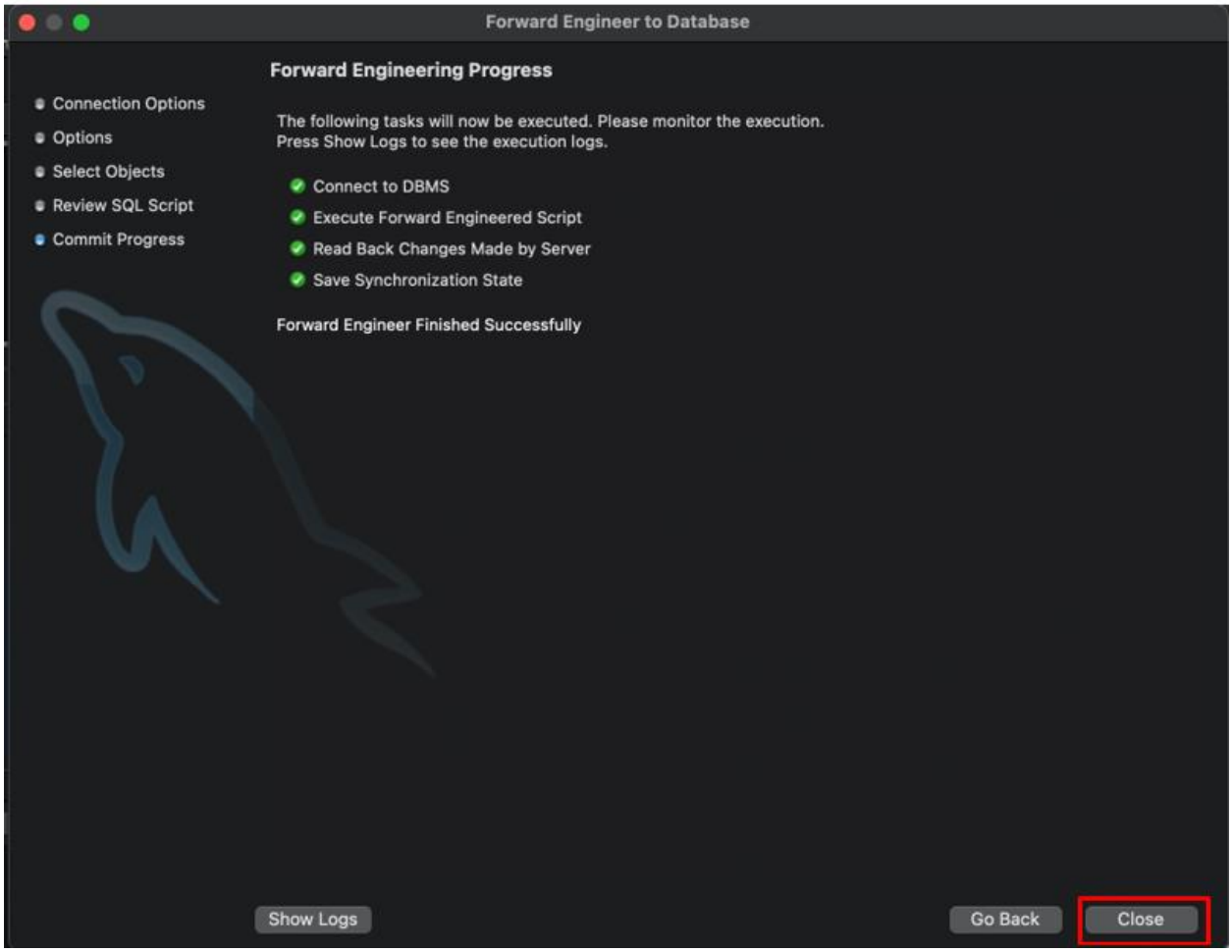
Εικόνα 24: Κλικ στο “continue”



Εικόνα 25: Κλικ στο “continue”

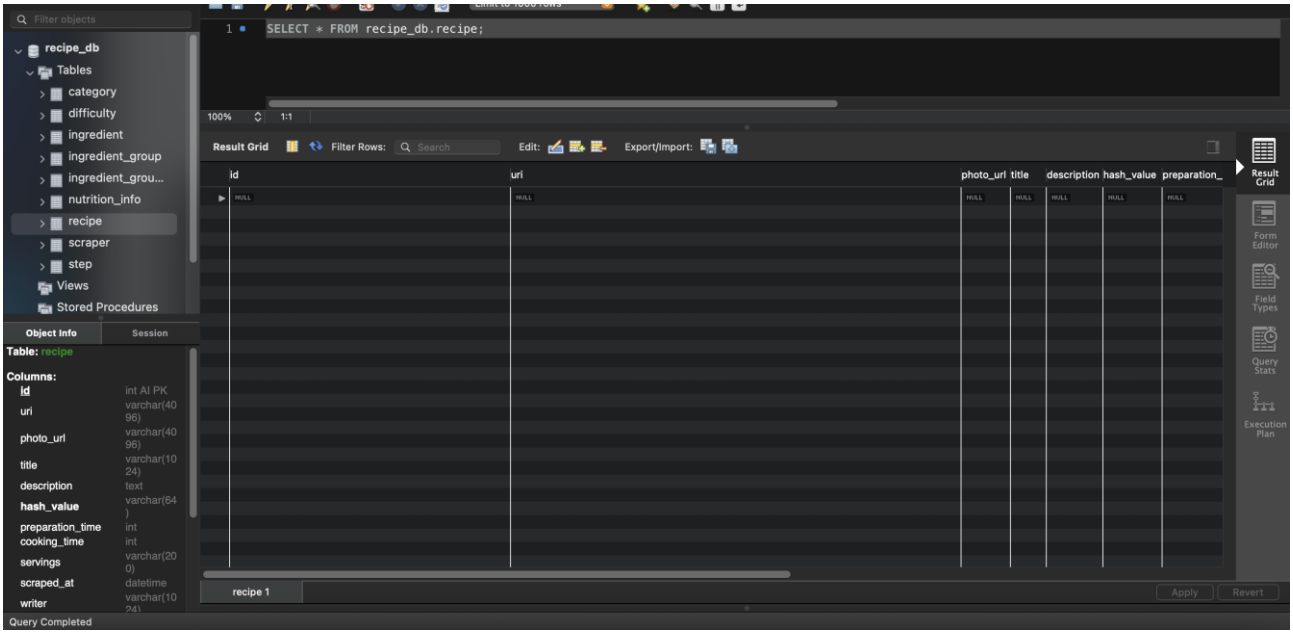


Εικόνα 26: Κλικ στο “continue”



Εικόνα 27: Κλικ στο “close”

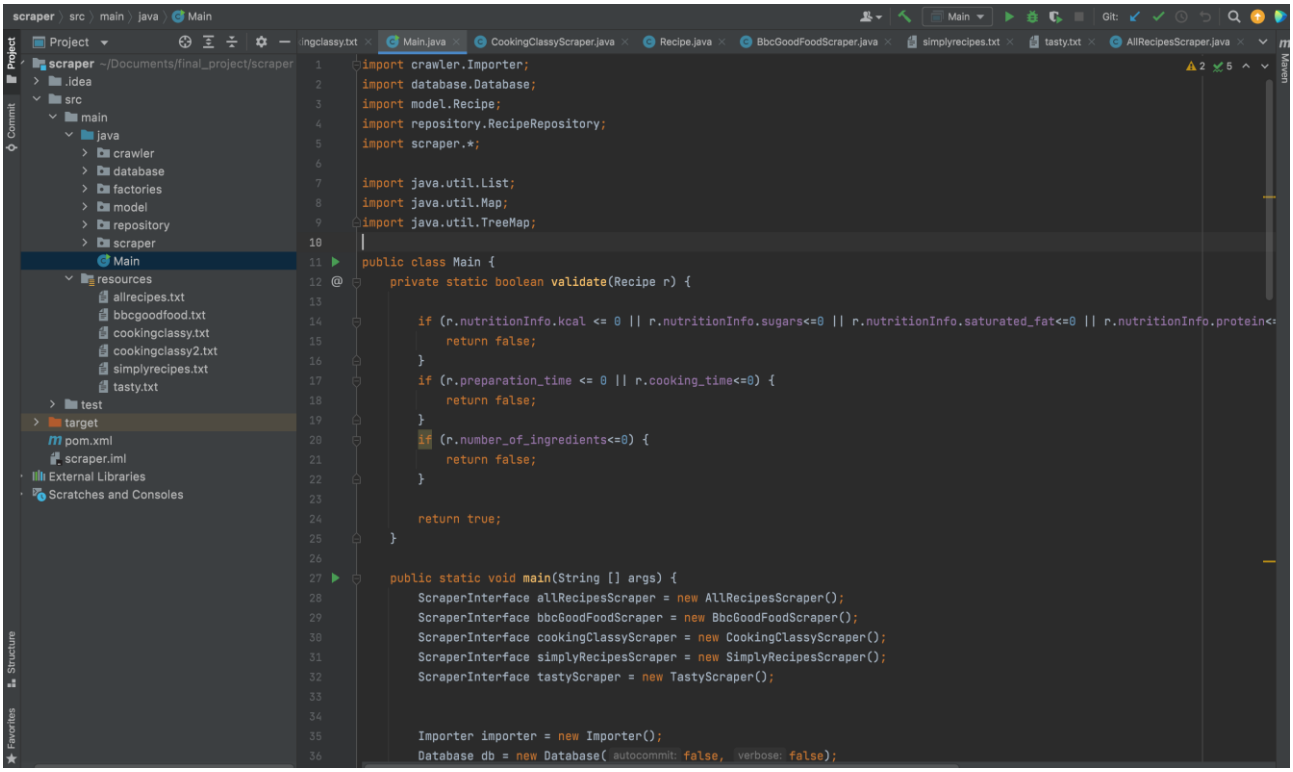
Πατάμε “close”(κλείσιμο). Με ένα απλό query(ερώτημα) μπορούμε να ελέγξουμε ότι η βάση είναι άδεια αυτή τη στιγμή.



Εικόνα 28: Ο άδειος πίνακας που επιστρέφει το query

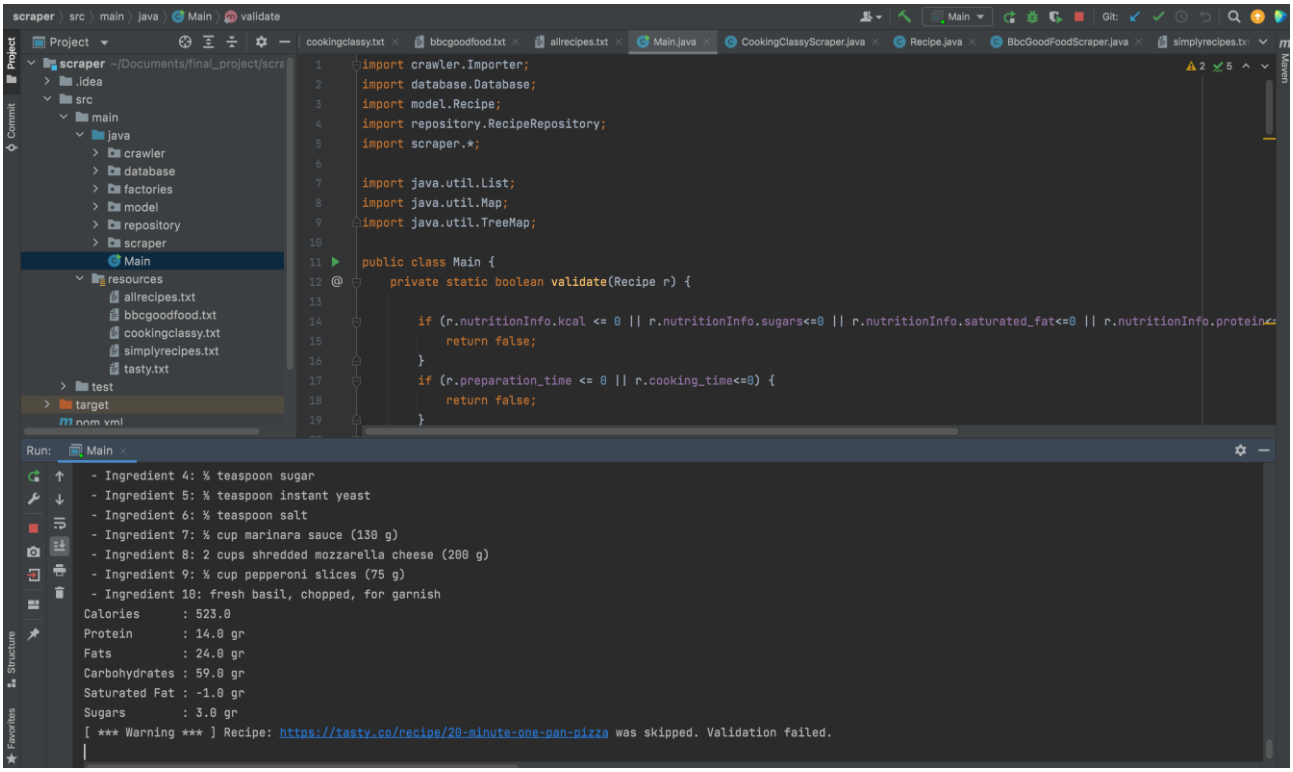
Έπειτα, αντιγράφουμε τα txt αρχεία από το φάκελο
“…/final_project/crawler_python/crawler_python/”
στον φάκελο
“…/final_project/scraper/src/main/resources/“.

Έτσι πάμε στη main στον φάκελο scraper και τρέχουμε το project:



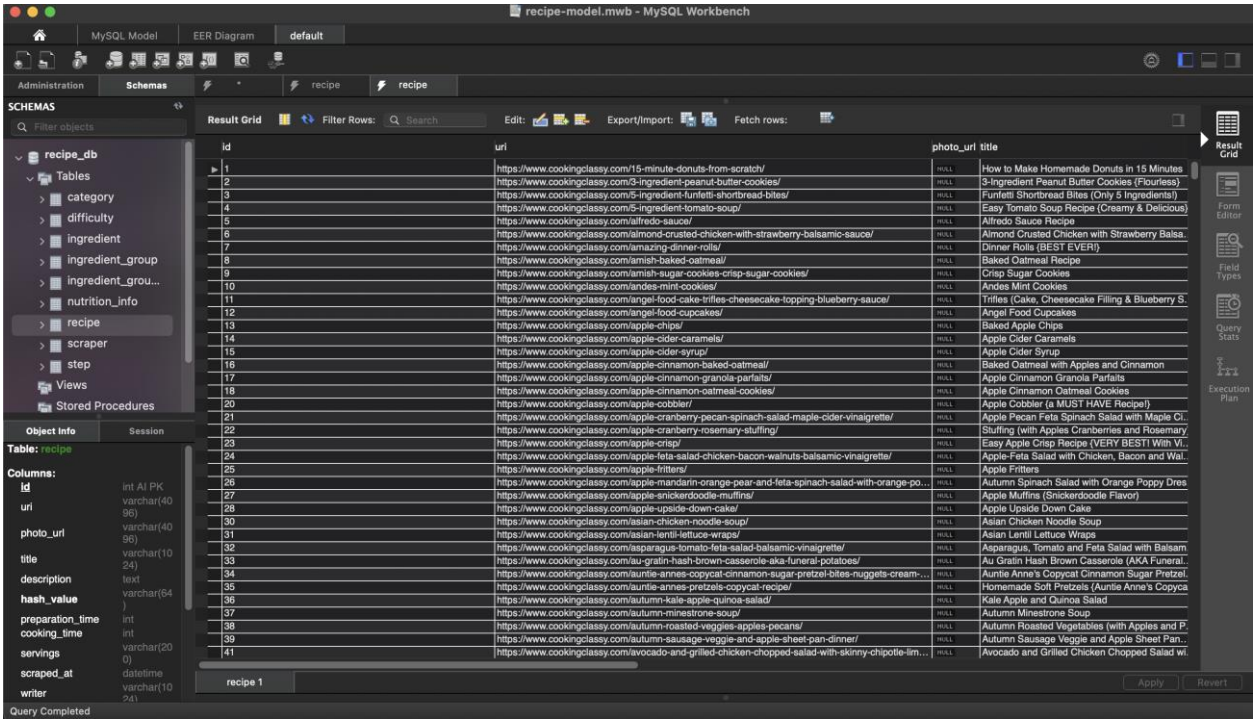
Εικόνα 29: Ο κώδικας της main.java

Όσο γίνεται το scraping:



Εικόνα 30: Κατά τη διάρκεια του scraping

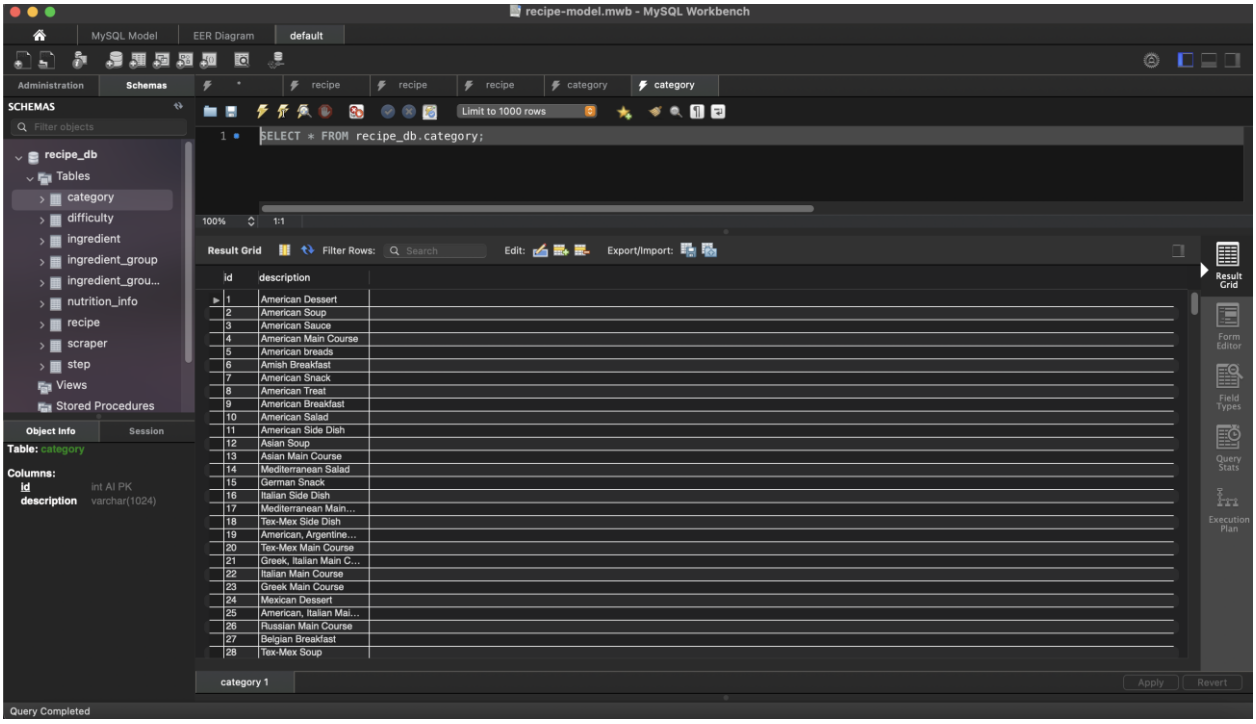
Η εικόνα της βάσης μας μετά το scraping είναι αυτή:



Εικόνα 31: Ο πίνακας “recipe” μετά το scraping

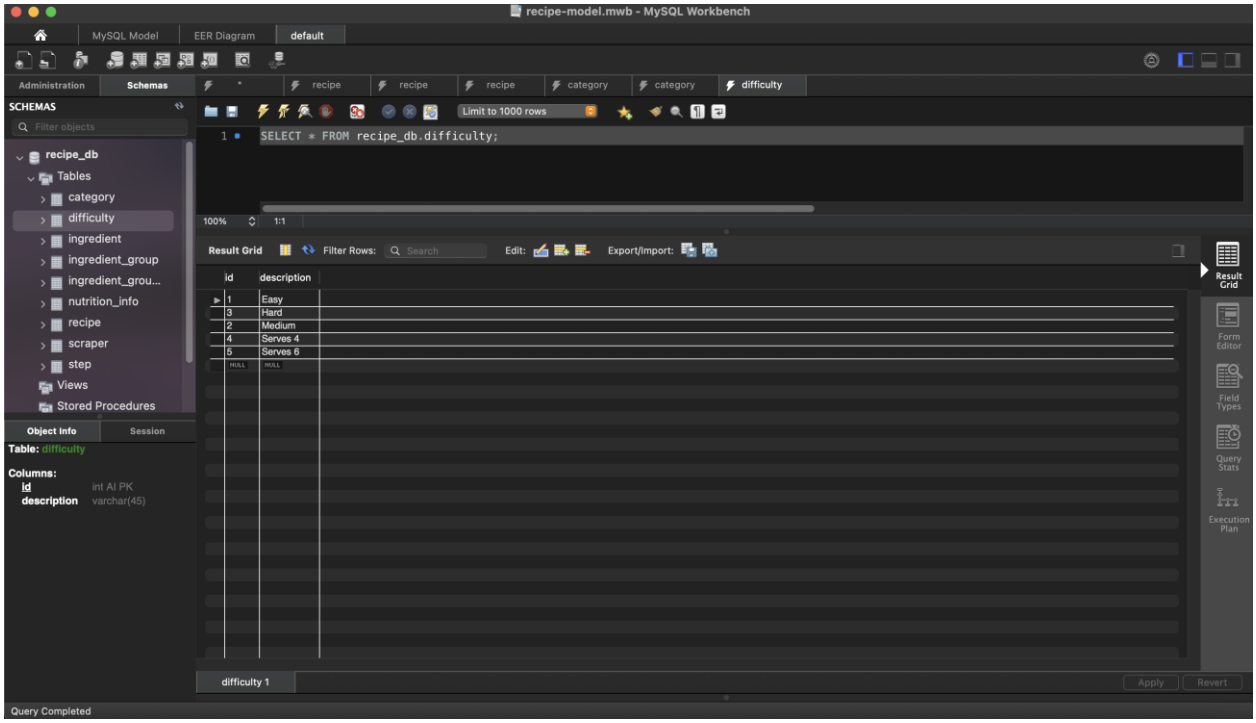
Η εικόνα κάθε πίνακα είναι η ακόλουθη:

Για τον category:



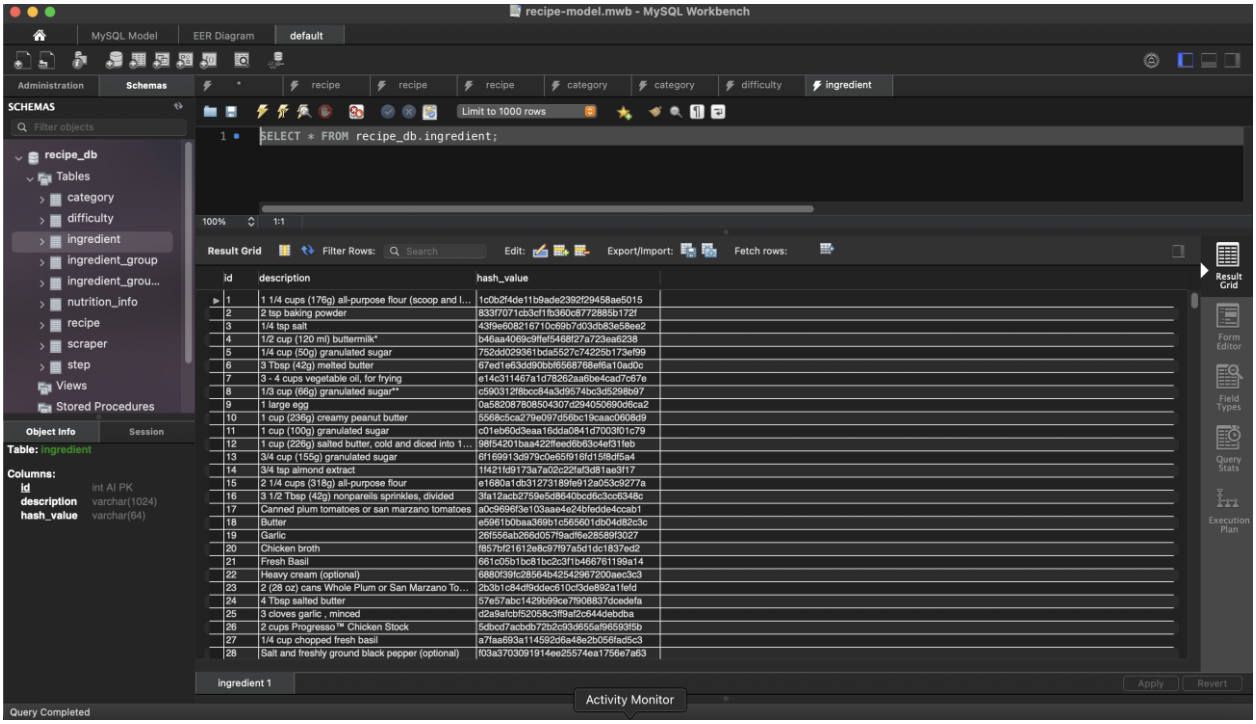
Εικόνα 32: Ο πίνακας “category”

Για τον difficulty:



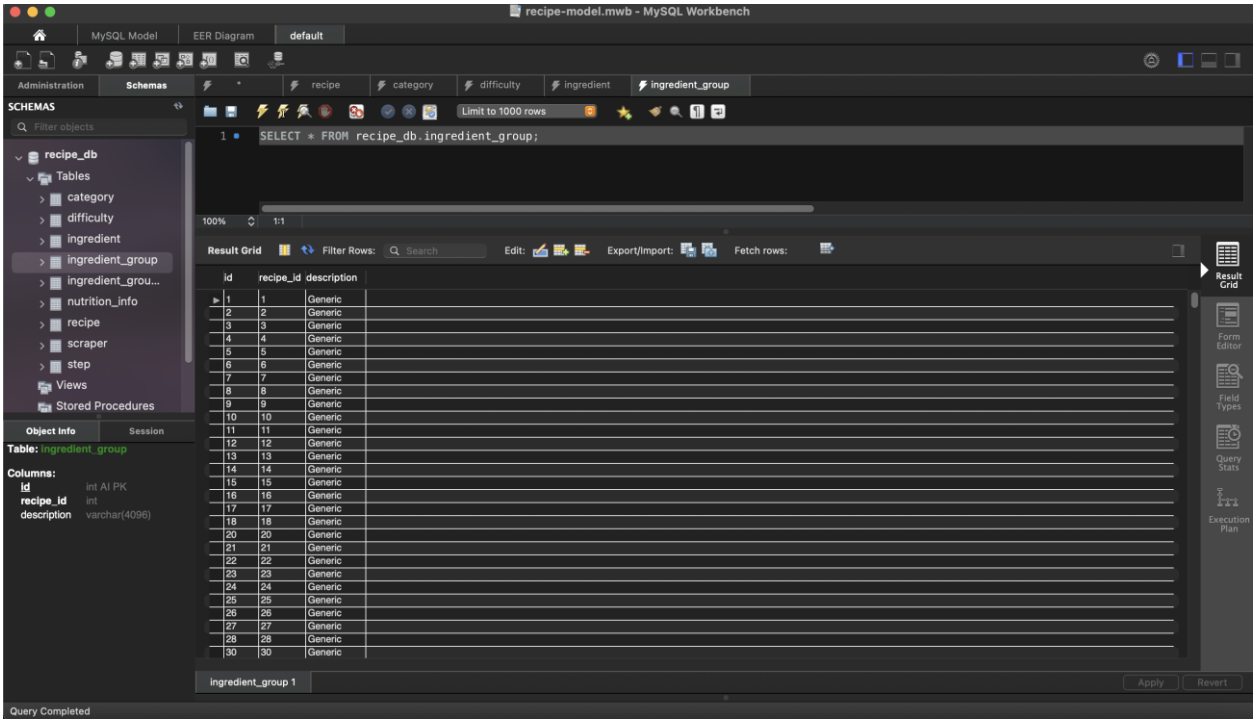
Εικόνα 33: Ο πίνακας “difficulty”

Για τον ingredient:



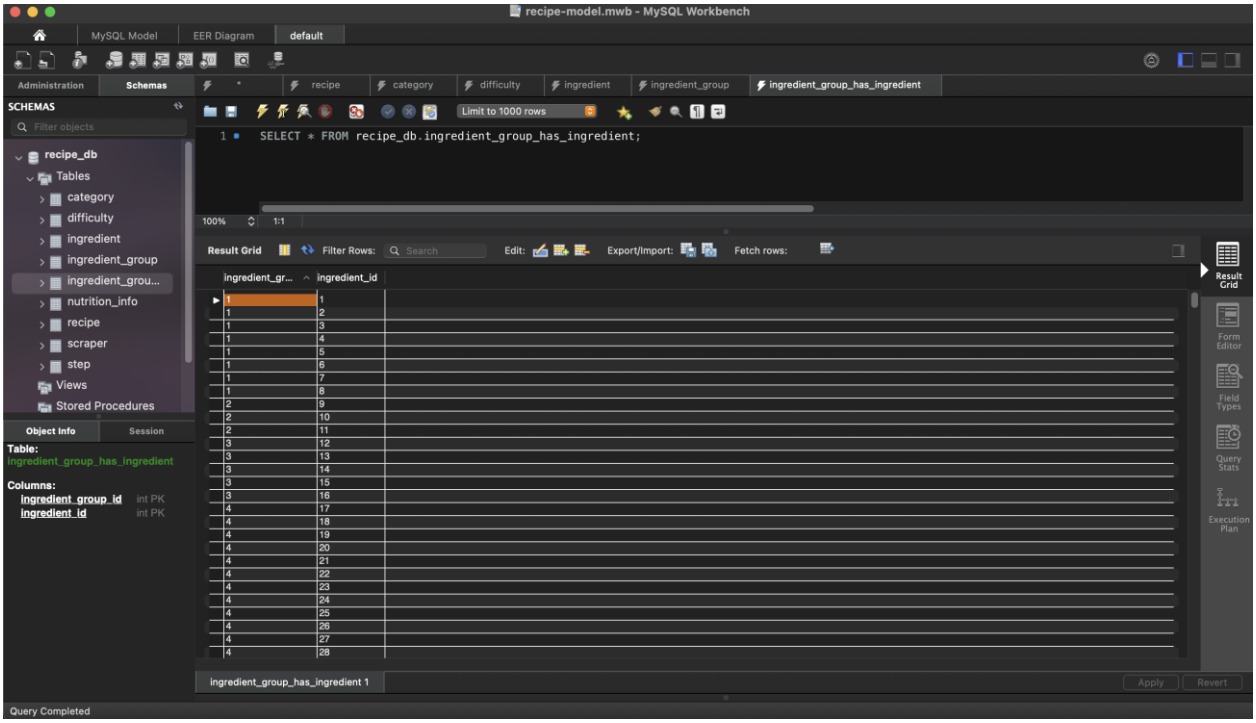
Εικόνα 34: Ο πίνακας “ingredient”

Για τον ingredient_group:



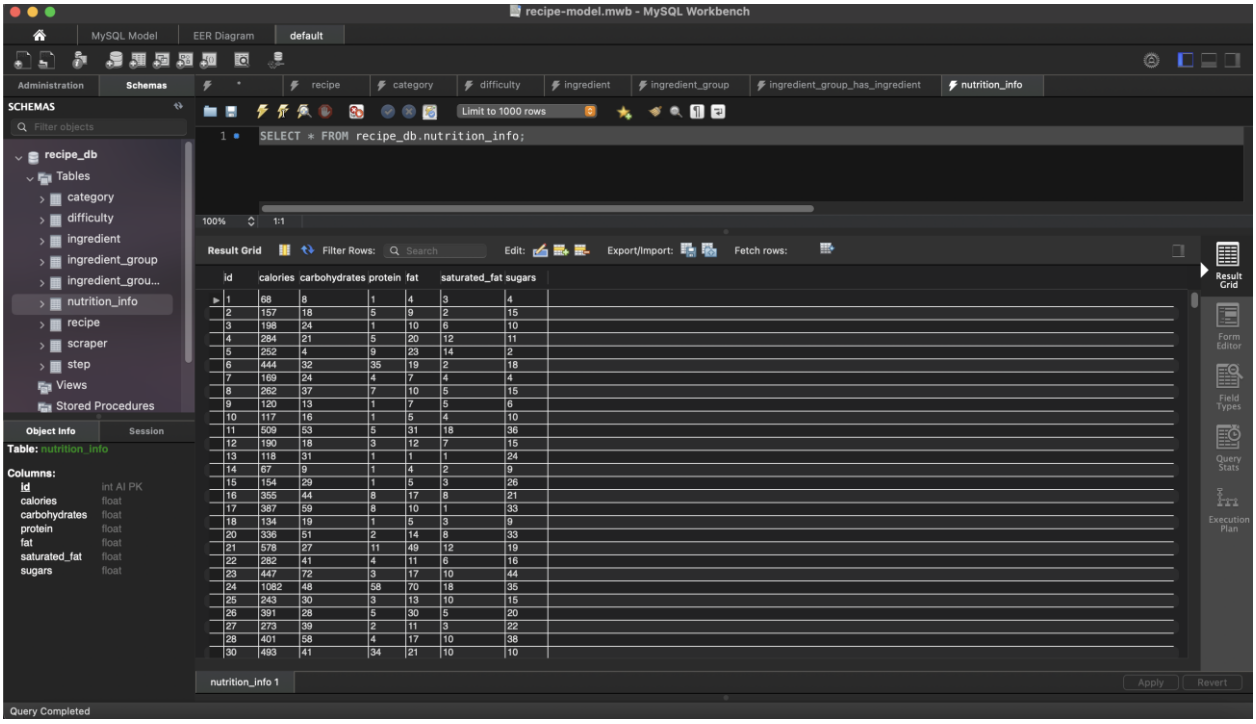
Εικόνα 35: Ο πίνακας “ingredient_group”

Για τον ingredient_group_has_ingredient:



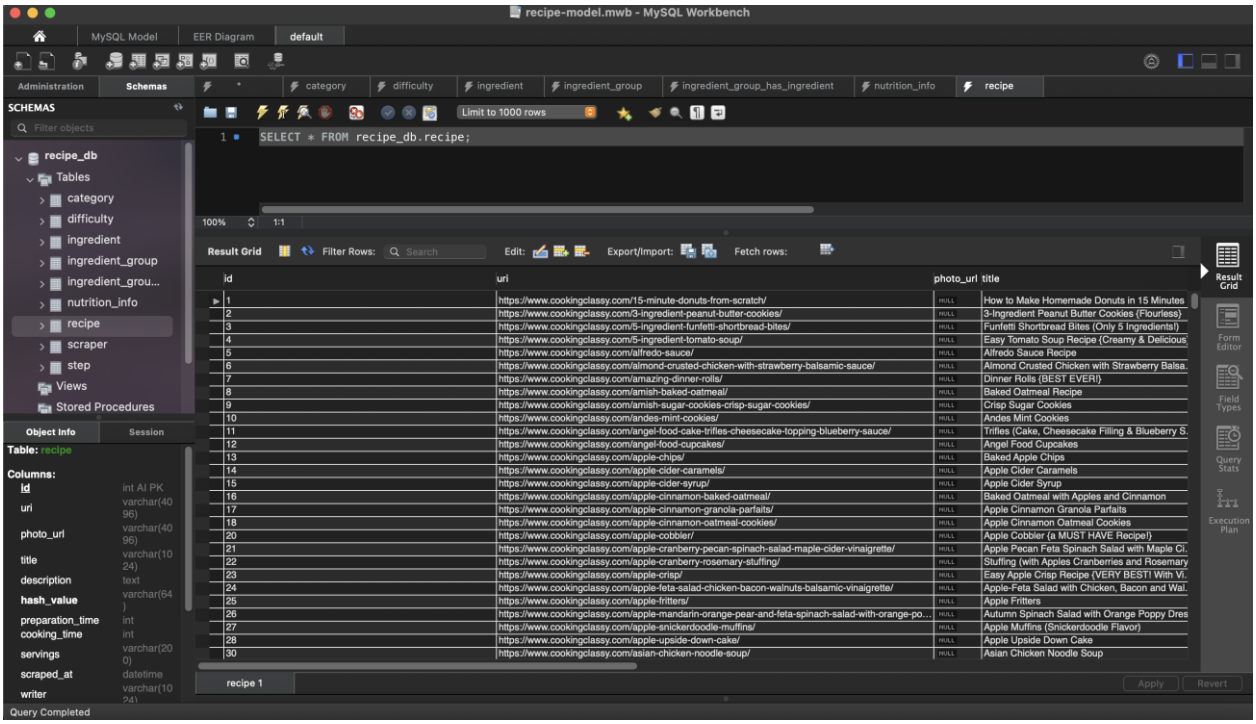
Εικόνα 36: Ο πίνακας “ingredient_group_has_ingredient”

Για τον nutrition_info:



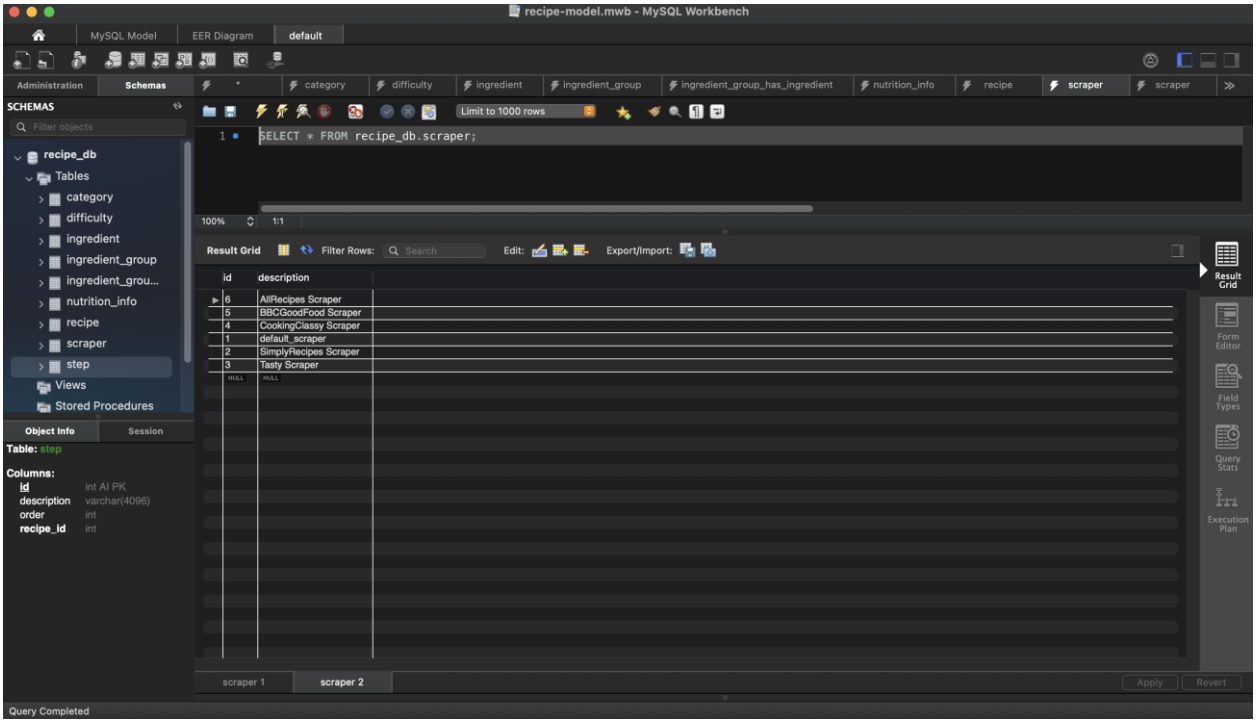
Εικόνα 37: Ο πίνακας “nutrition_info”

Για τον recipe:



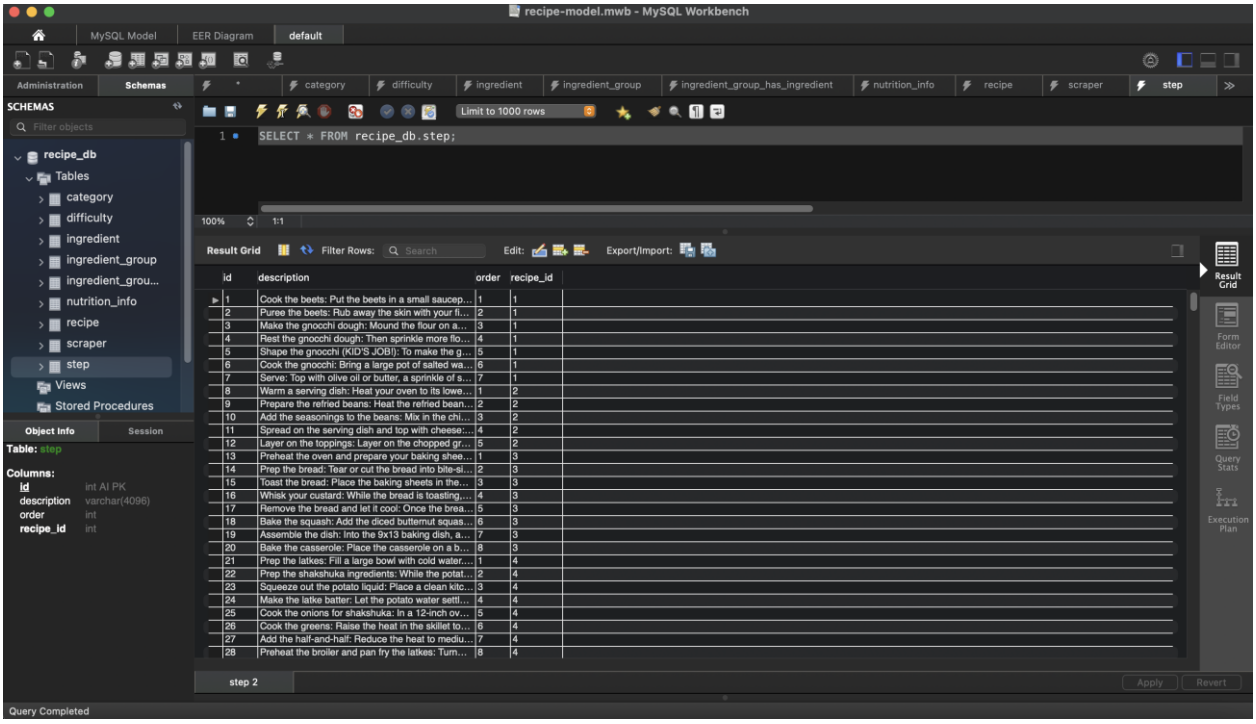
Εικόνα 38: Ο πίνακας “recipe”

Για τον scraper:



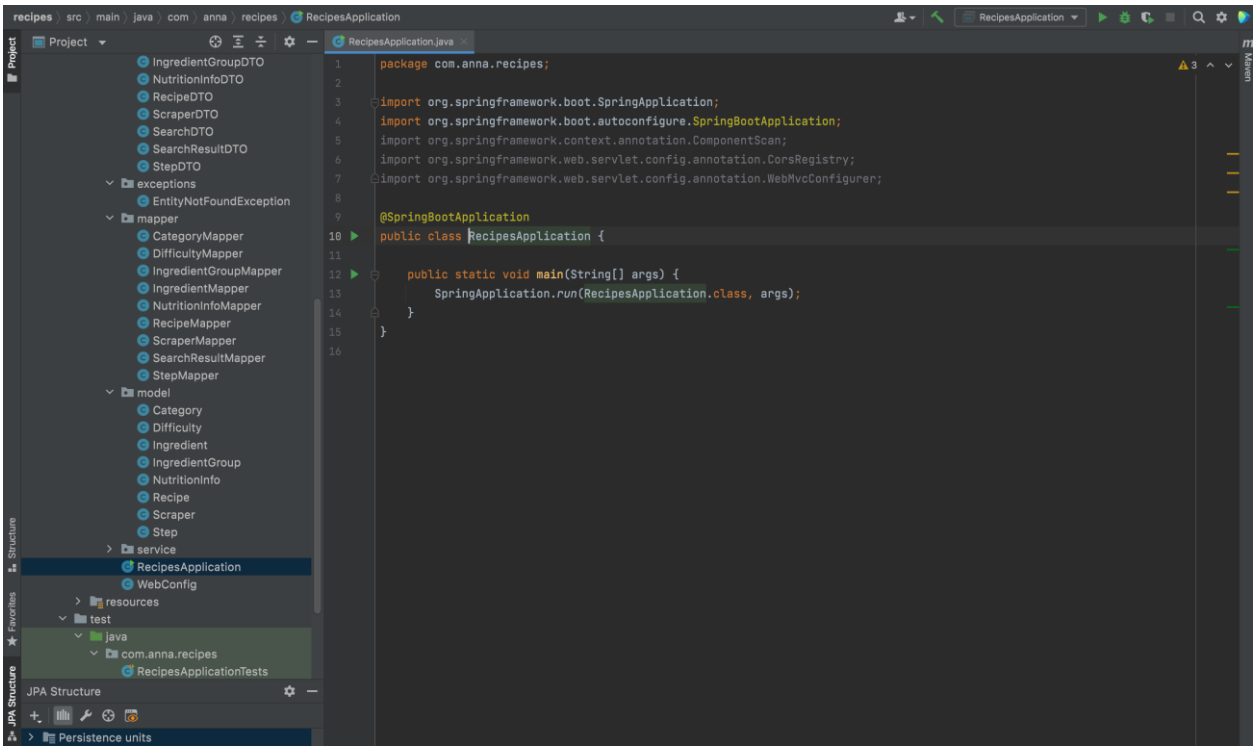
Εικόνα 39: Ο πίνακας “scraper”

Για τον step:



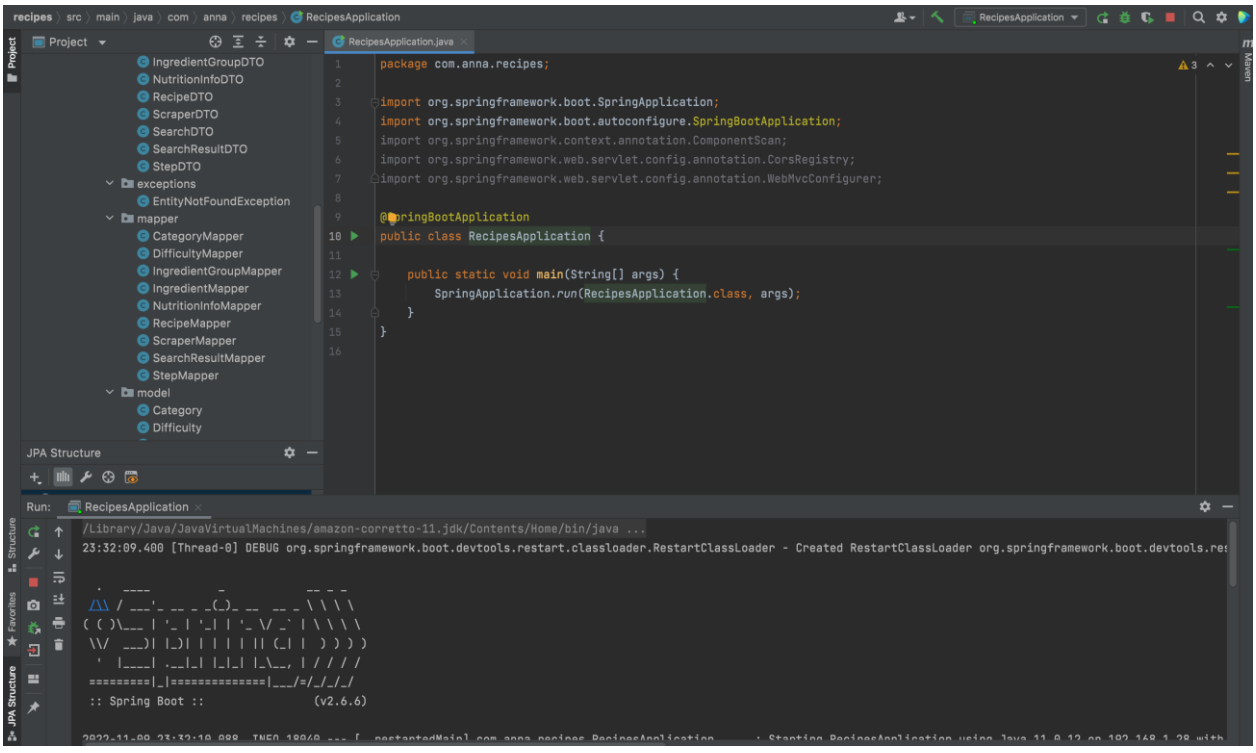
Εικόνα 40: Ο πίνακας “step”

Στη συνέχεια πηγαίνουμε στο IntelliJ IDEA και τρέχουμε το project “Recipes”, δηλαδή το backend.



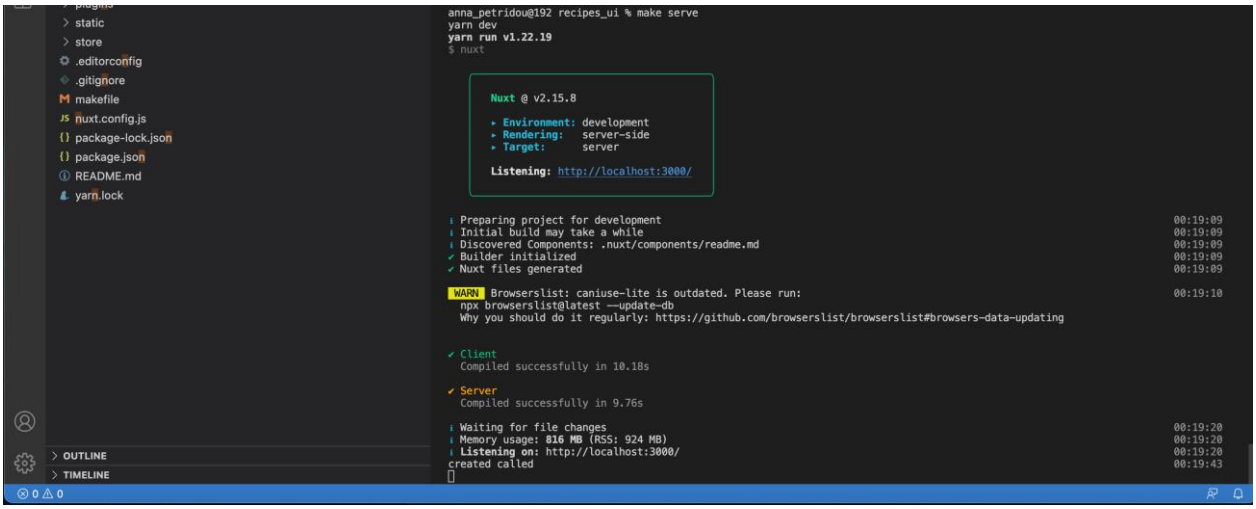
Εικόνα 41: Το αρχείο κώδικα “RecipesApplication.java”

Εκκινείται το spring boot όπως φαίνεται παρακάτω:



Εικόνα 42: Το terminal όταν εκκινείται το spring boot

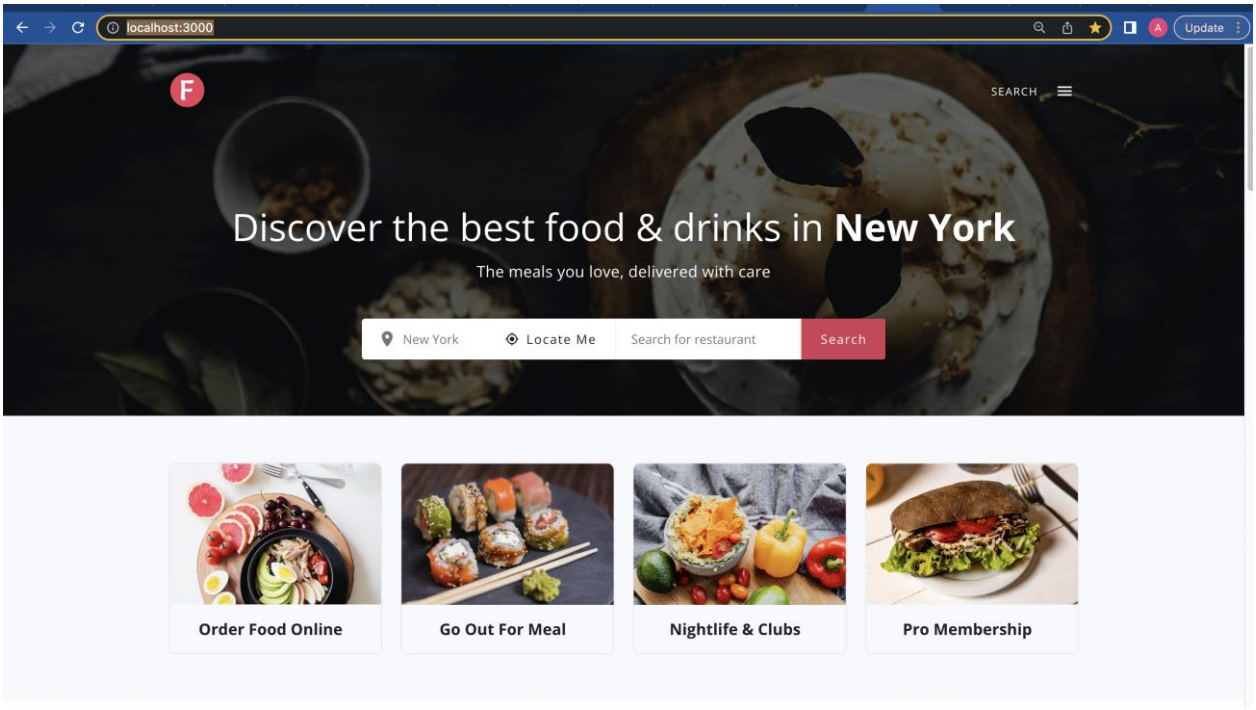
Έπειτα πάμε στο Visual Studio Code για να τρέξουμε το front end. Ανοίγουμε το φάκελο “recipes_ui” μέσα από το Visual Studio Code και τρέχουμε “make serve” στο terminal.



Εικόνα 43: Make serve στο visual studio code

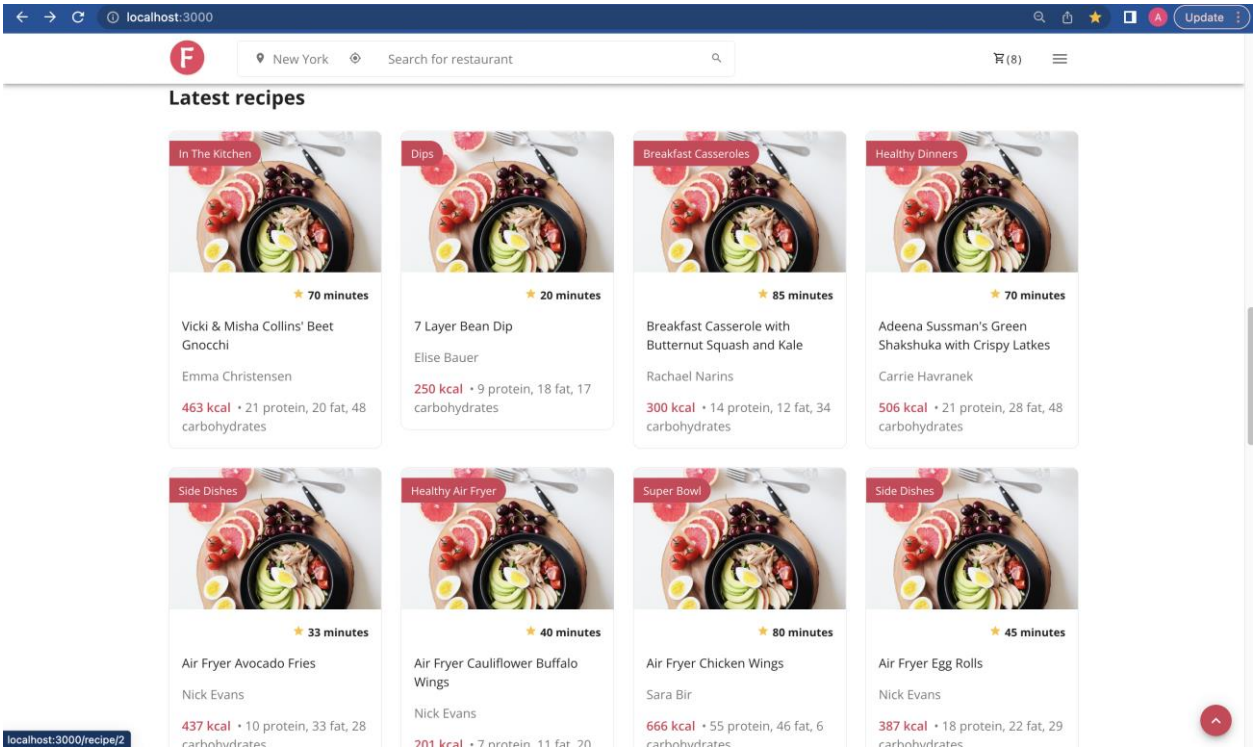
Στο σημείο αυτό, μπορούμε να πάμε στην ιστοσελίδα μας στο http://localhost:3000/ και να δούμε την τελική εικόνα του user interface μας.

Πιο συγκεκριμένα, η αρχική οθόνη είναι αυτή:



Εικόνα 44: Η homepage σελίδα του user interface

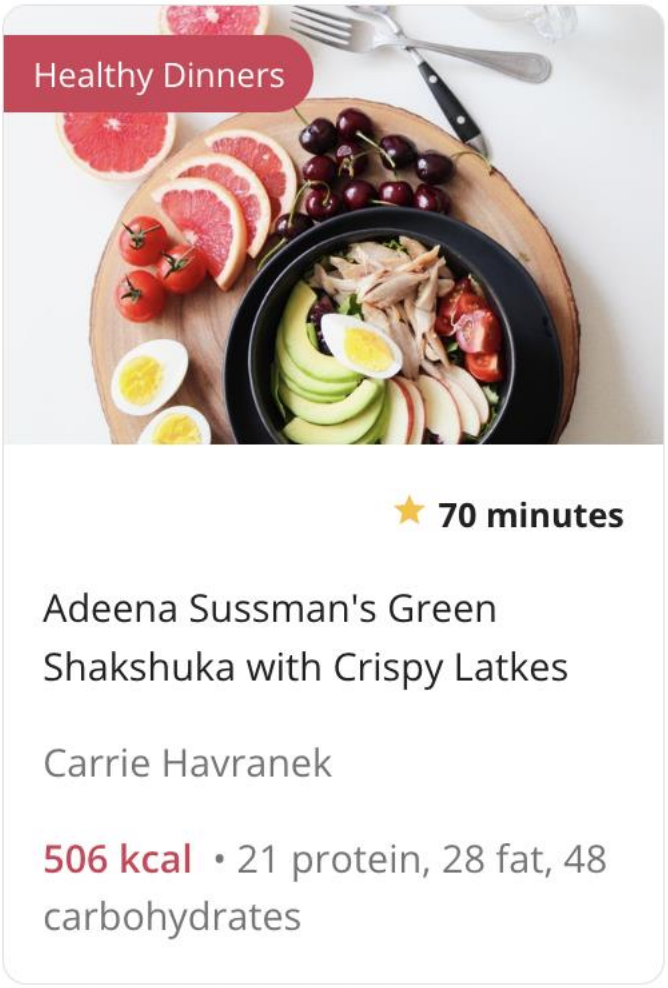
Αν κάνουμε scroll down, μπορούμε να δούμε τις πιο πρόσφατες συνταγές:



Εικόνα 45: Η αρχική σελίδα μετά από scroll down

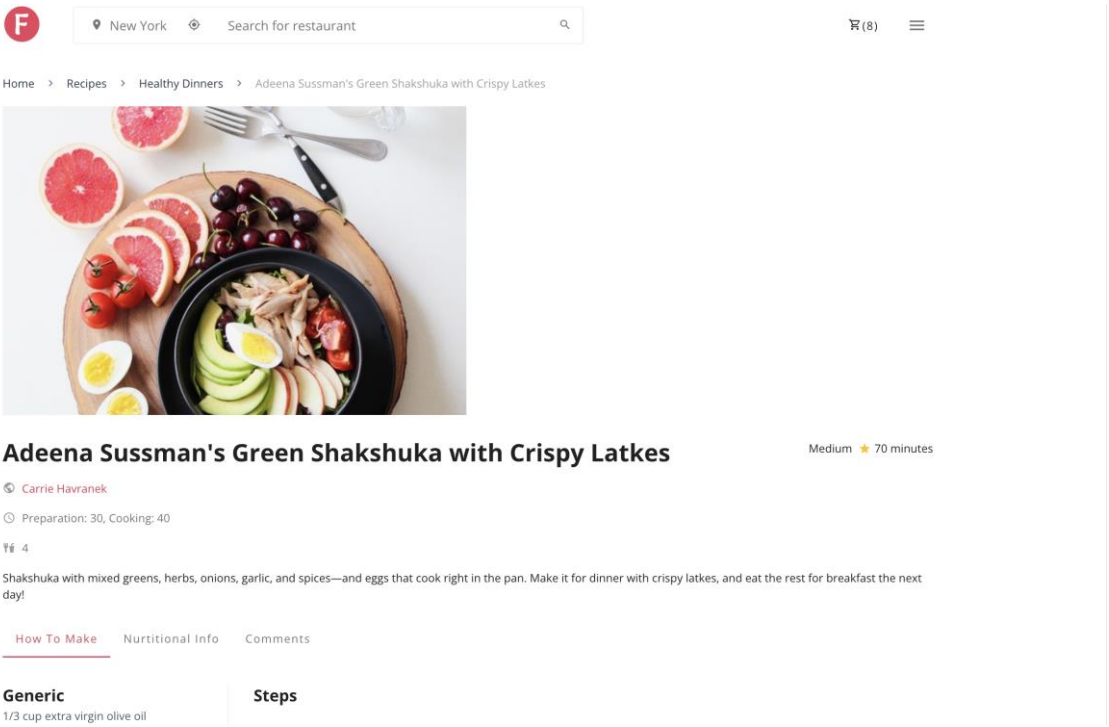
Μια από τις πιθανές επεκτάσεις που θα μπορούσαν να γίνουν μελλοντικά στην εφαρμογή, είναι η προσθήκη φωτογραφίας για κάθε συνταγή. Το αντίστοιχο πεδίο έχει ήδη δημιουργηθεί στη βάση δεδομένων.

Σε κάθε μία από αυτές τις συνταγές μπορούμε να παρατηρήσουμε και διάφορες πληροφορίες τους, όπως την κατηγορία, τον τίτλο, το συγγραφέα, τις θερμίδες, τις διατροφικές πληροφορίες και το χρόνο εκτέλεσης.

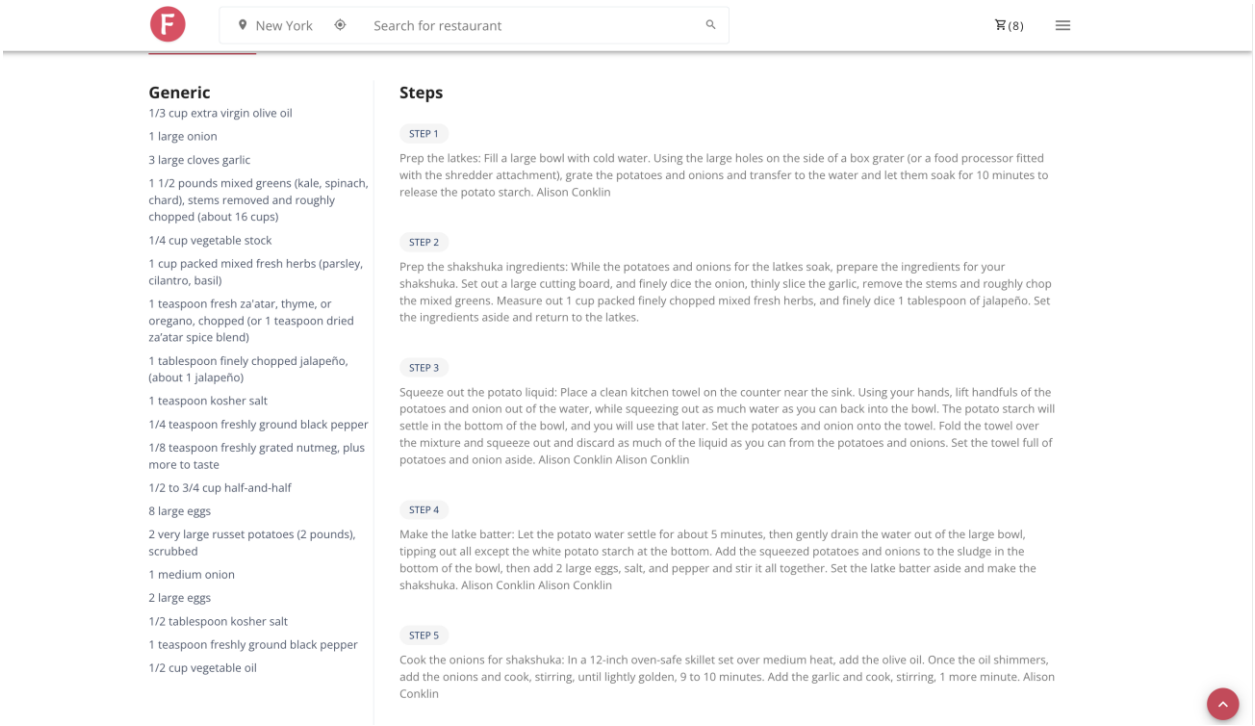


Εικόνα 46: Αποτέλεσμα ενδεικτικής συνταγής

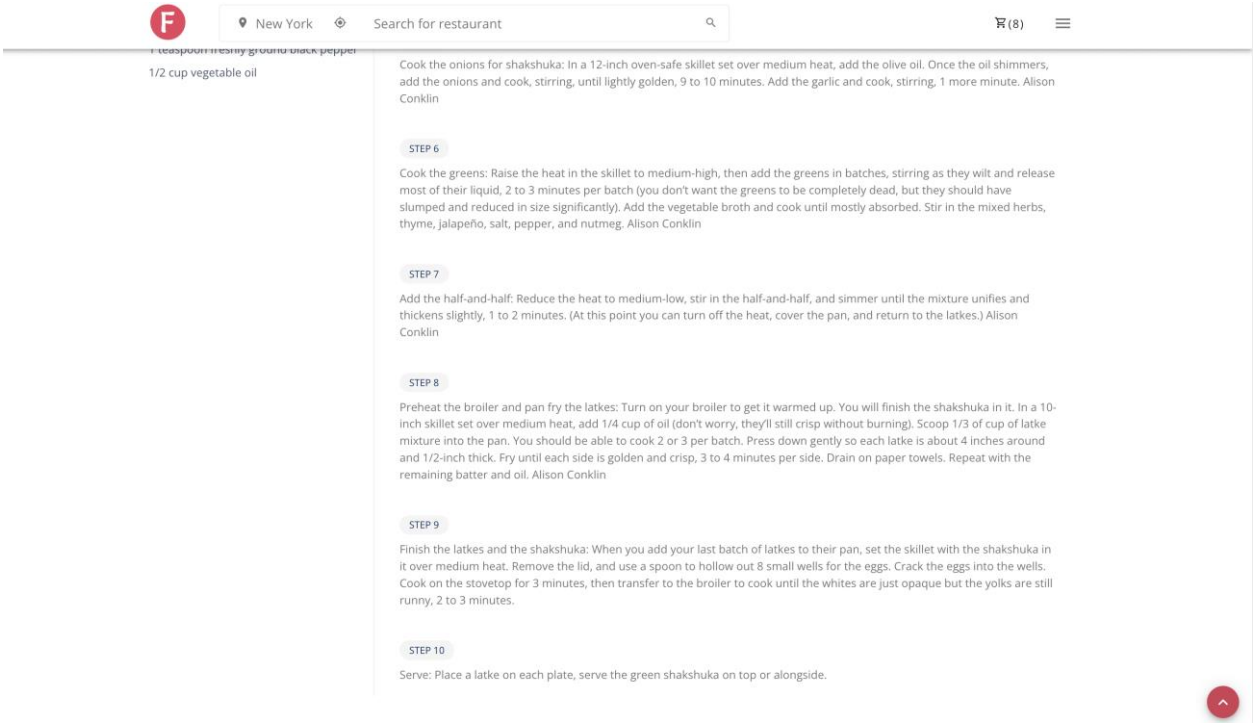
Αν κάνουμε κλικ σε κάποια από αυτές, θα μας ανοίξει στην παρακάτω μορφή, όπου μπορούμε να διαβάσουμε όλη την πληροφορία της συνταγής:



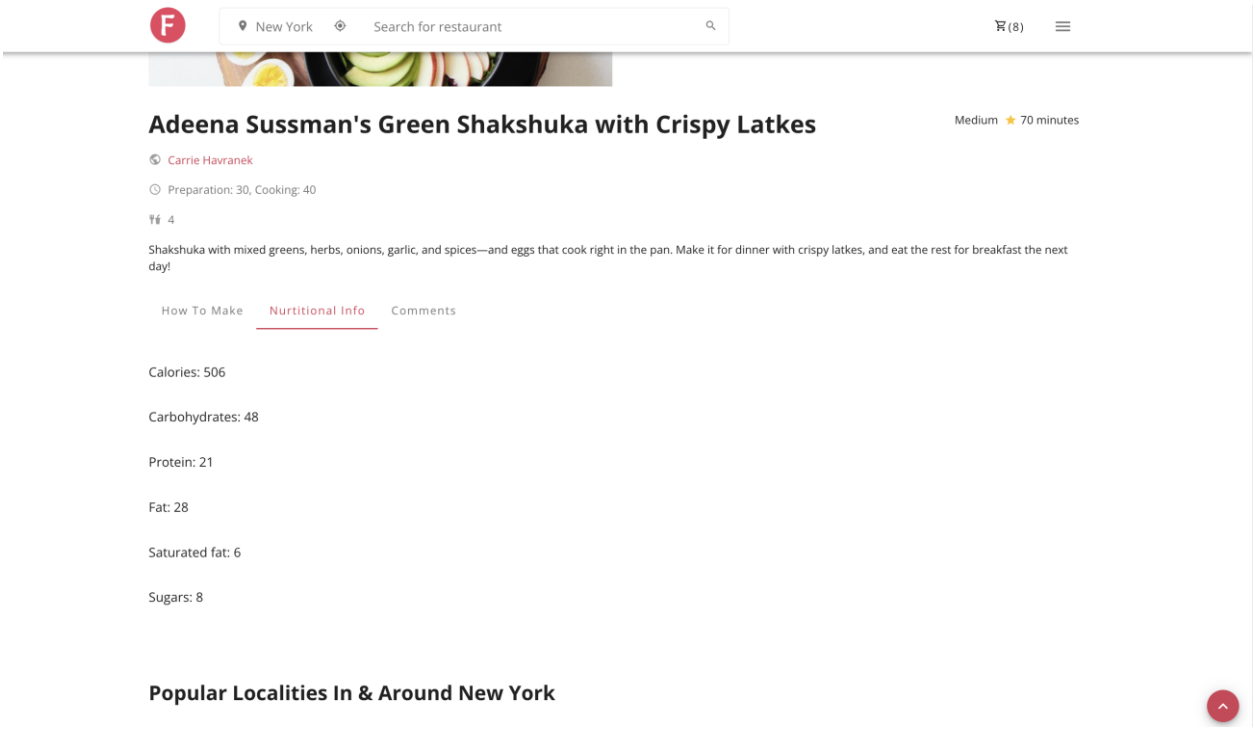
Εικόνα 47: Σελίδα συνταγής με όλη τη σχετική πληροφορία



Εικόνα 48: Σελίδα συνταγής μετά από scroll down

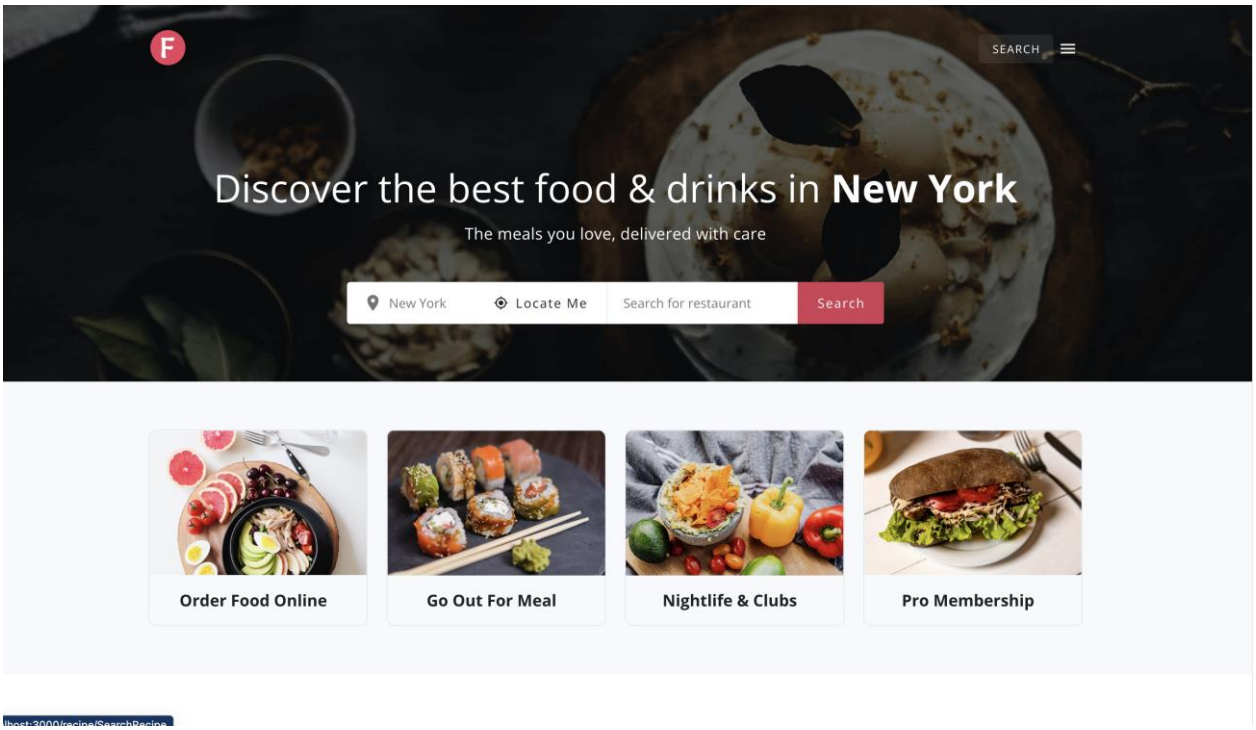


Εικόνα 49: Σελίδα συνταγής μετά από περισσότερο scroll down



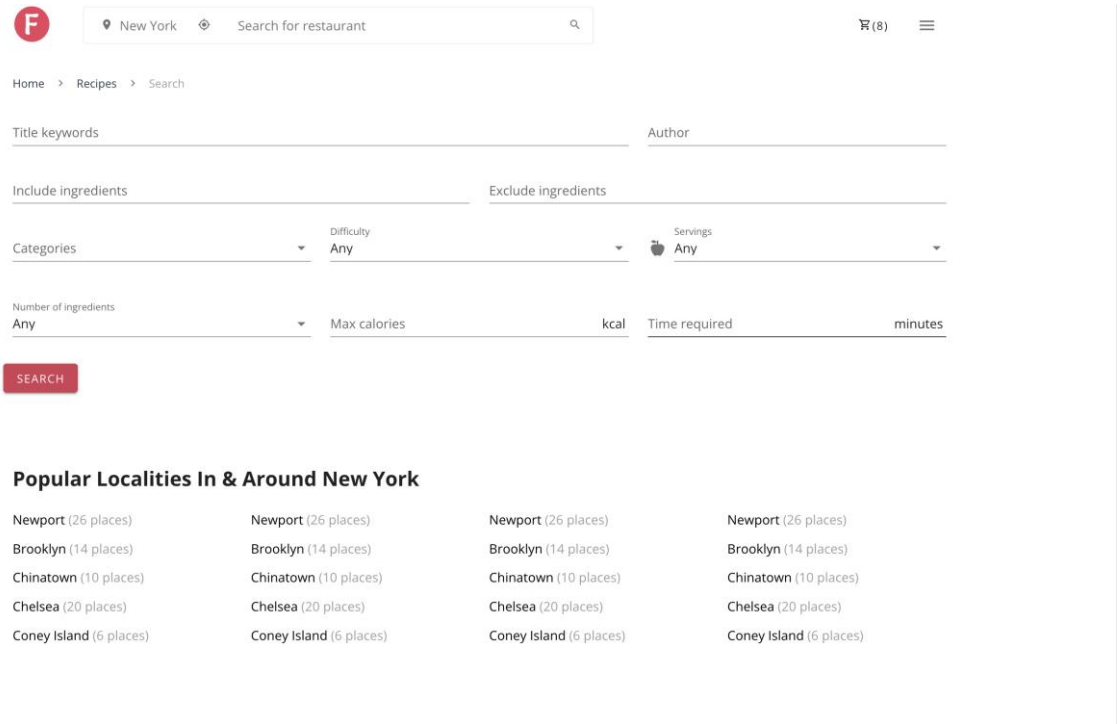
Εικόνα 50: Το tab του "nutritional info"

Μπορούμε να πατήσουμε το "Search"(αναζήτηση) πάνω δεξιά για να πραγματοποιήσουμε αναζητήσεις.



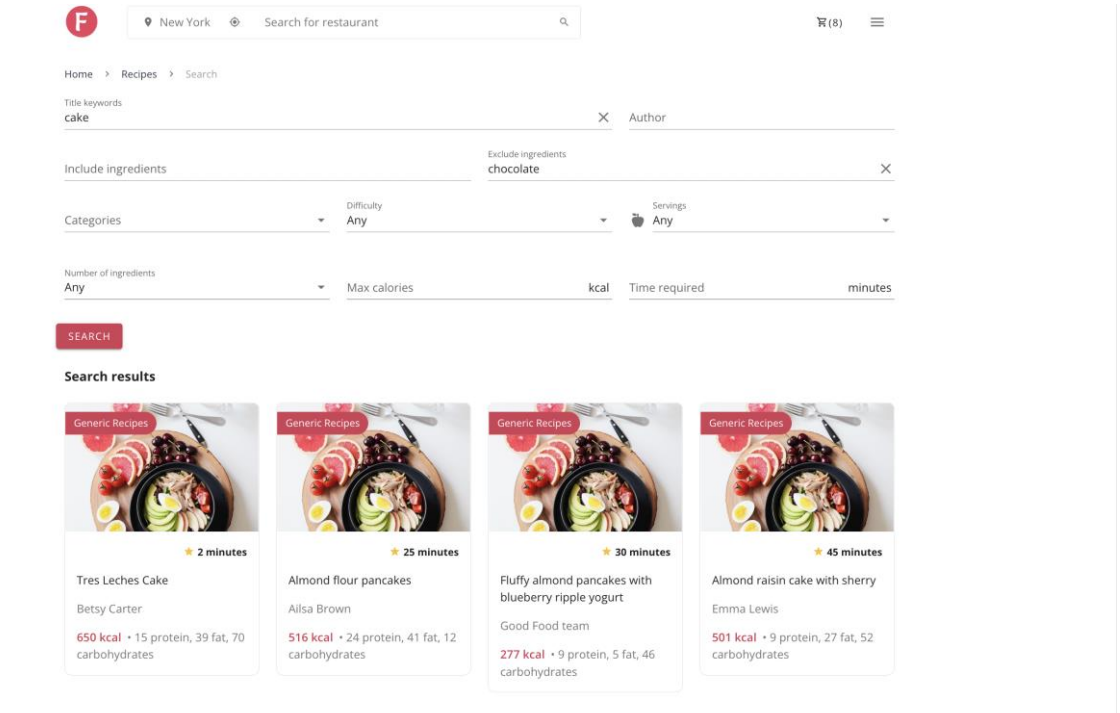
Εικόνα 51: Κουμπί αναζήτησης στην αρχική σελίδα

Μας μεταφέρει στη σελίδα με τα φίλτρα από όπου μπορούμε να επιλέξουμε όσα φίλτρα θέλουμε για να περιορίσουμε τα αποτελέσματα και να κάνουμε την αναζήτησή μας πιο στοχευμένη.



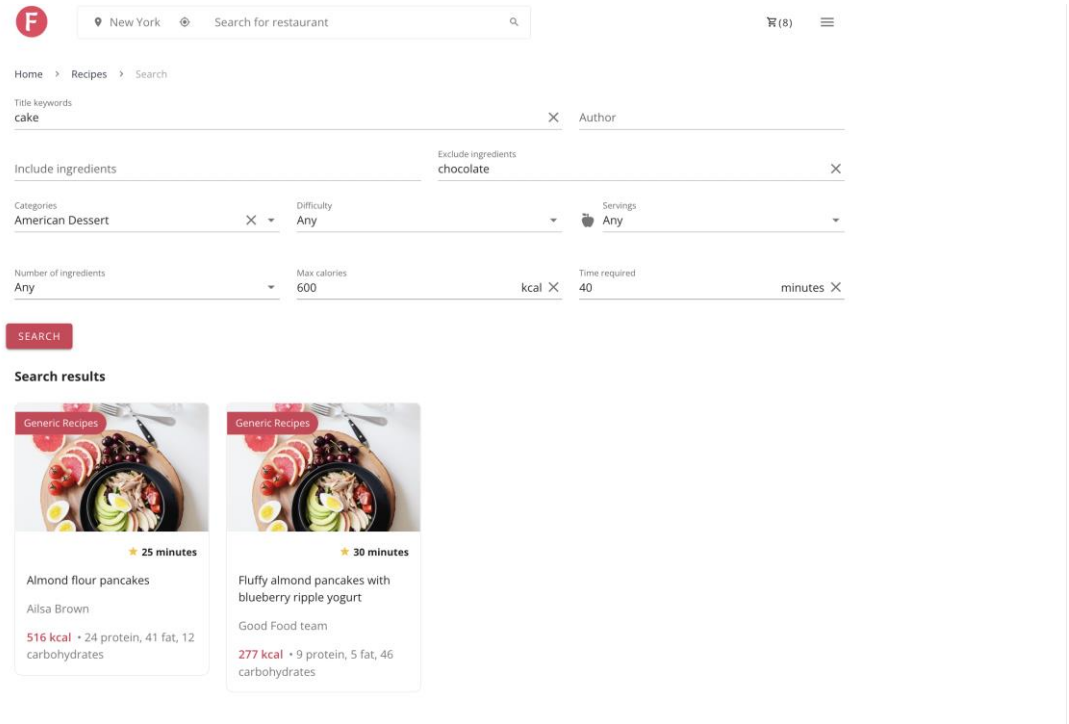
Εικόνα 52: Σελίδα με φίλτρα

Για παράδειγμα, έστω ότι βάζουμε τη λέξη κλειδί “cake” στον τίτλο και αποκλείουμε από τα συστατικά τη σοκολάτα, τα πρώτα αποτελέσματα που θα πάρουμε είναι αυτά.



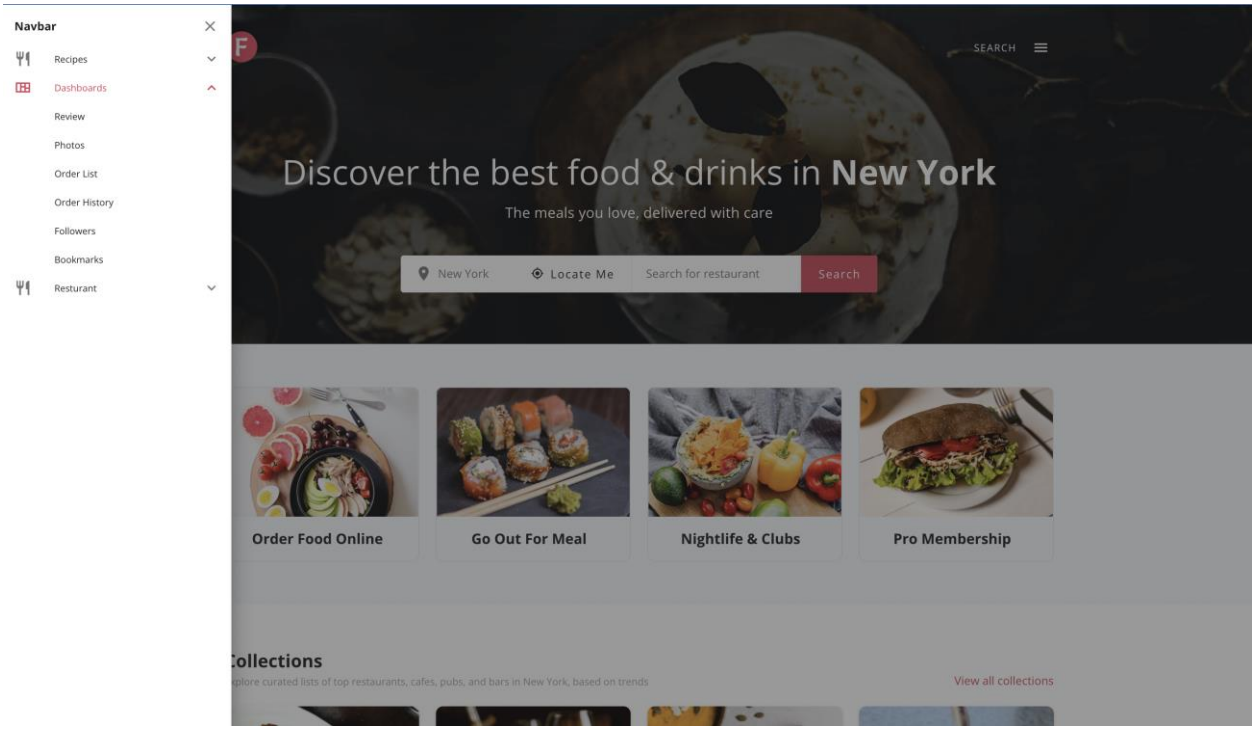
Εικόνα 53: Αποτελέσματα φίλτρων

Αν στη συνέχεια προσθέσουμε κατηγορία, max θερμίδων και χρόνο εκτέλεσης, τα αποτελέσματα περιορίζονται ακόμα περισσότερο.

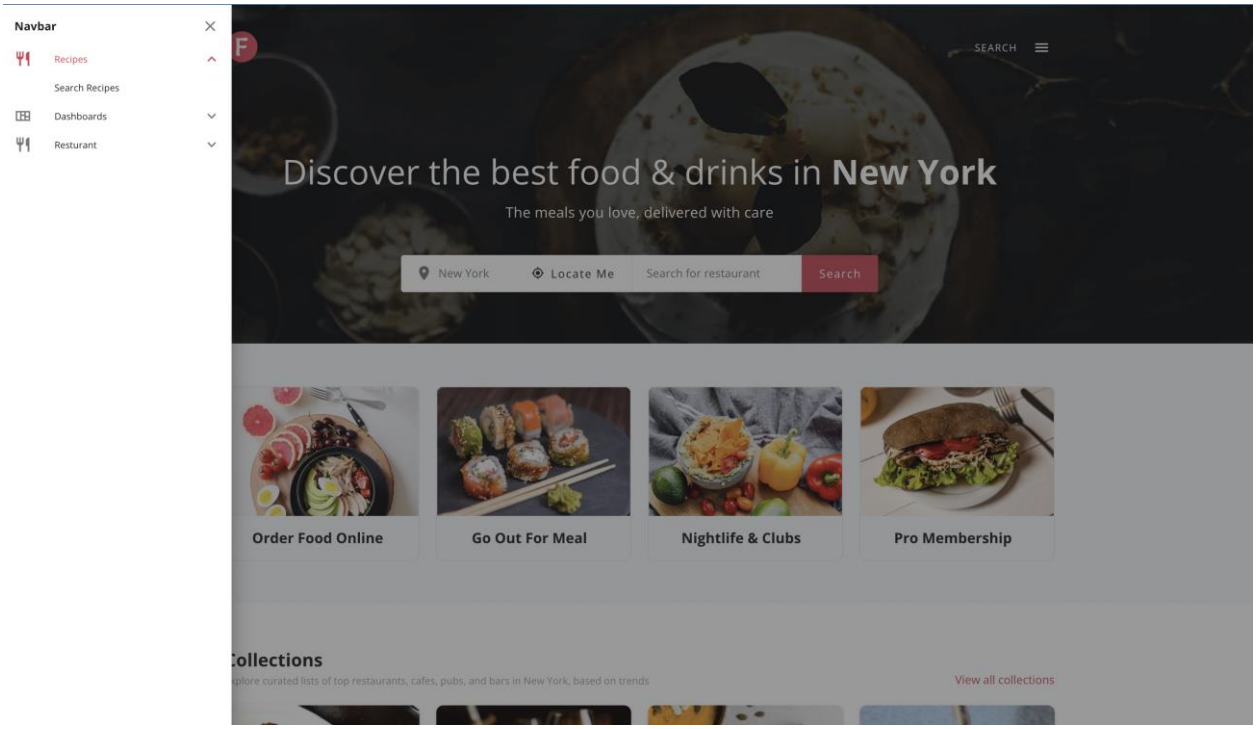


Εικόνα 54: Αποτελέσματα μετά από περισσότερα φίλτρα

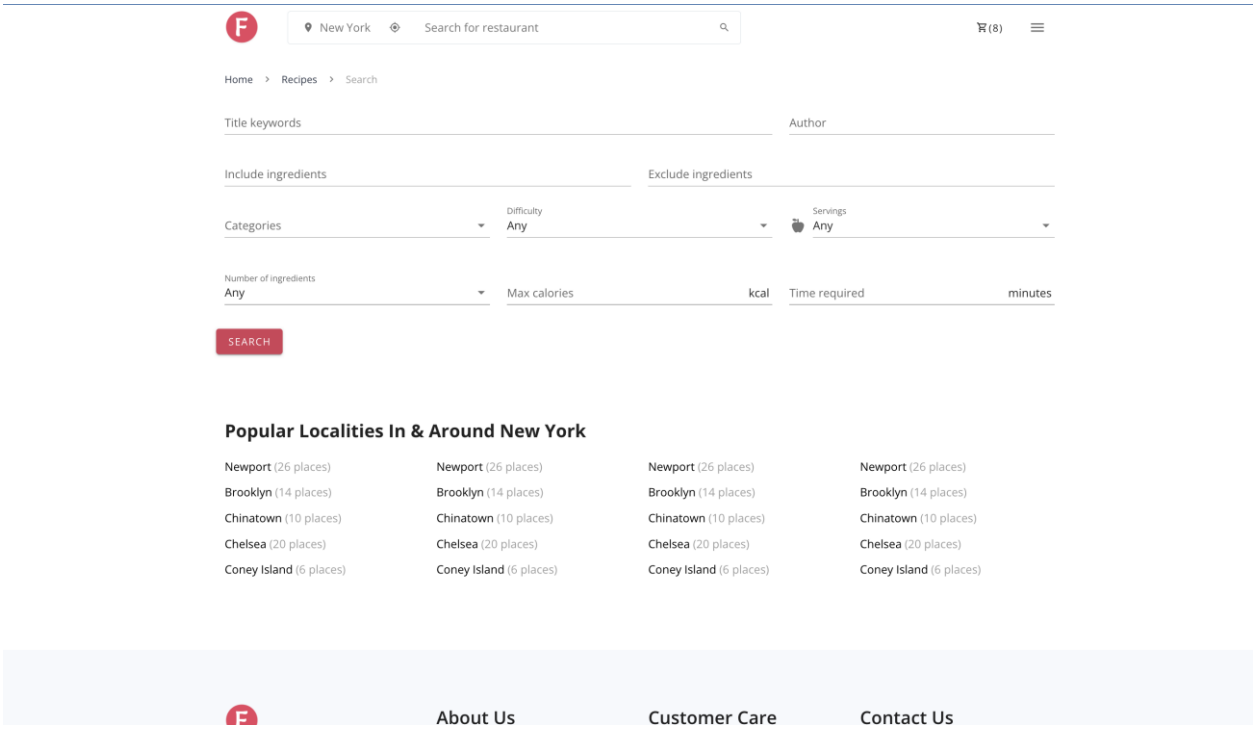
Ο χρήστης μπορεί να καταλήξει στην αναζήτηση είτε από το “Search” πάνω δεξιά στην αρχική οθόνη, είτε ανοίγοντας το tab με τις επιλογές και επιλέγοντας “Recipes” και μετά “Search Recipes”



Εικόνα 55: Τρόπος αναζήτησης από το μενού στα αριστερά

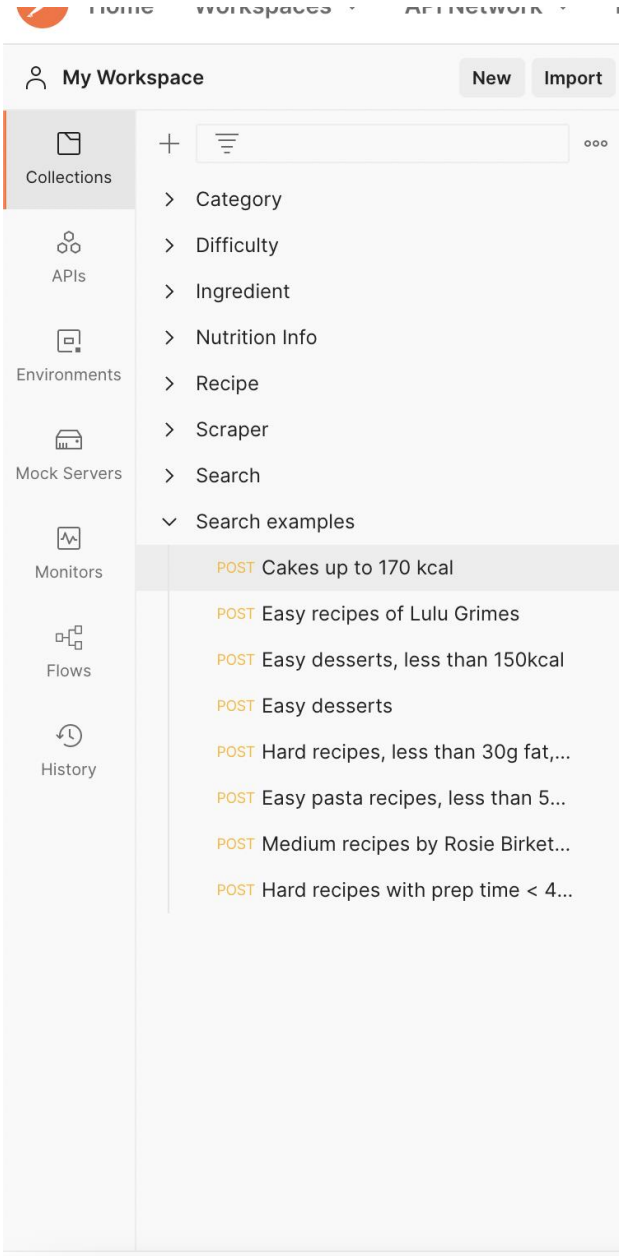


Εικόνα 56: Κλικ στο “recipes” και “search recipes”



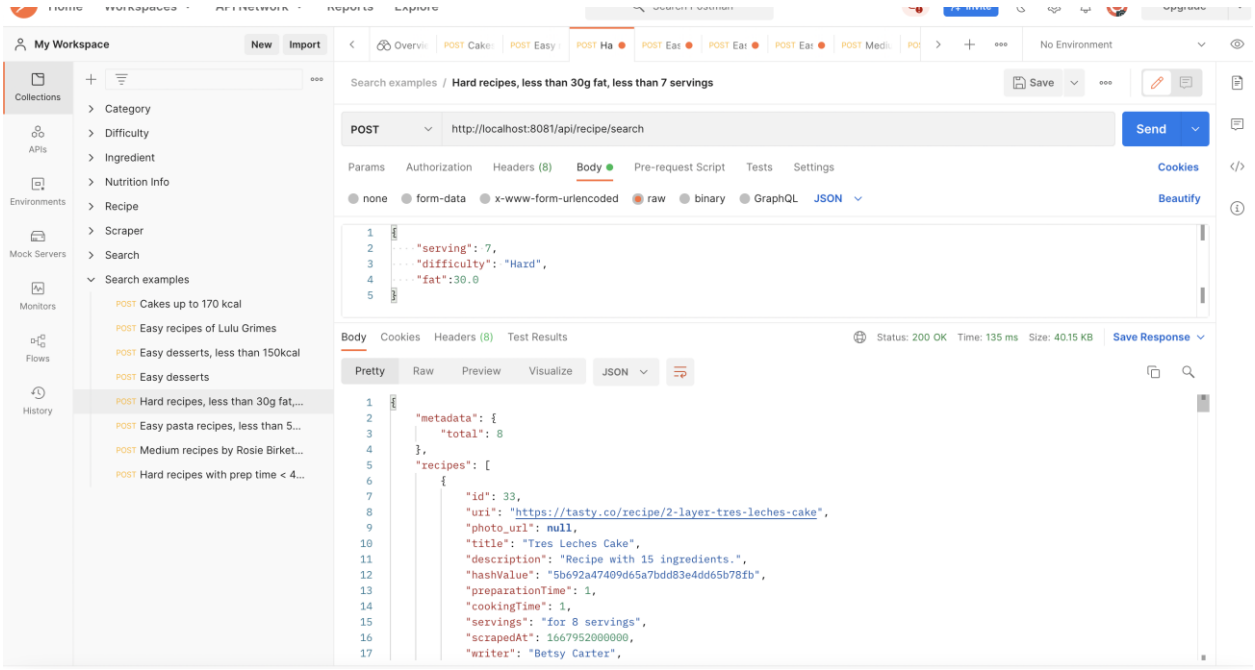
Εικόνα 57: Τα πεδία των φίλτρων

Τα αποτελέσματα μας έχουν τεσταριστεί και στο postman. Εκεί θα βλέπαμε τα αποτελέσματα άλλωστε αν δεν υπήρχε το user interface. Μερικά παραδείγματα αναζητήσεων στο postman είναι τα παρακάτω:



Εικόνα 58: Search examples στο postman

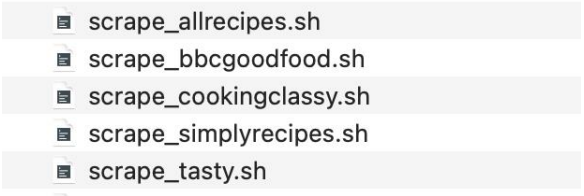
Βλέπουμε εδώ ας πούμε τα αποτελέσματα της αναζήτησης για συνταγές δυσκολίας “Hard”, που έχουν λιγότερα από 30 γρ λίπους και βγάζουν λιγότερες από 7 μερίδες.



Εικόνα 59: Παράδειγμα αναζήτησης με περιορισμούς στο postman

Όσον αφορά τον κώδικα:

Αρχικά, έχουμε τους 5 crawlers υλοποιημένους σε python με χρήση του scrapy. Τα αρχεία των crawlers είναι τα παρακάτω:



Εικόνα 60: Τα αρχεία των crawlers

Τα αρχεία ακολουθούν όλα την παρακάτω μορφή με κάποιες διαφοροποιήσεις/ προσαρμογές ανάλογα το site, ώστε να καταγράφονται μόνο οι σύνδεσμοι που μας ενδιαφέρουν.

Για το AllRecipesCrawler.py:

```python
from scrapy import Item, Field
from scrapy.spiders import CrawlSpider, Rule
from scrapy.linkextractors import LinkExtractor

class AllRecipesCrawler(CrawlSpider):
    name = 'allrecipes'
    allowed_domains = ['www.allrecipes.com']
    start_urls = ['https://www.allrecipes.com']
    rules = (Rule(LinkExtractor(), callback='parse_item', follow=True),)
    filename = 'allrecipes.txt'
    accept_url = "https://www.allrecipes.com/recipe/"
    reject_urls = ["printview", "page"]

    def parse_item(self, response):
        if response.status == 200:
            url = response.url

            item = { }
            item['url'] = response.url
            item['referer'] = response.request.headers.get('Referer')
            item['status'] = response.status

            if url.startswith(self.accept_url):
                matches = True
                for prefix in self.reject_urls:
                    if prefix in url:
                        matches = False

                if matches:
                    with open(self.filename, 'at') as f:
                        f.write(url +"\n")

            return item
```

Εικόνα 61: Το αρχείο του crawler για το allrecipes.com

Για τον CookingClassyCrawler.py:

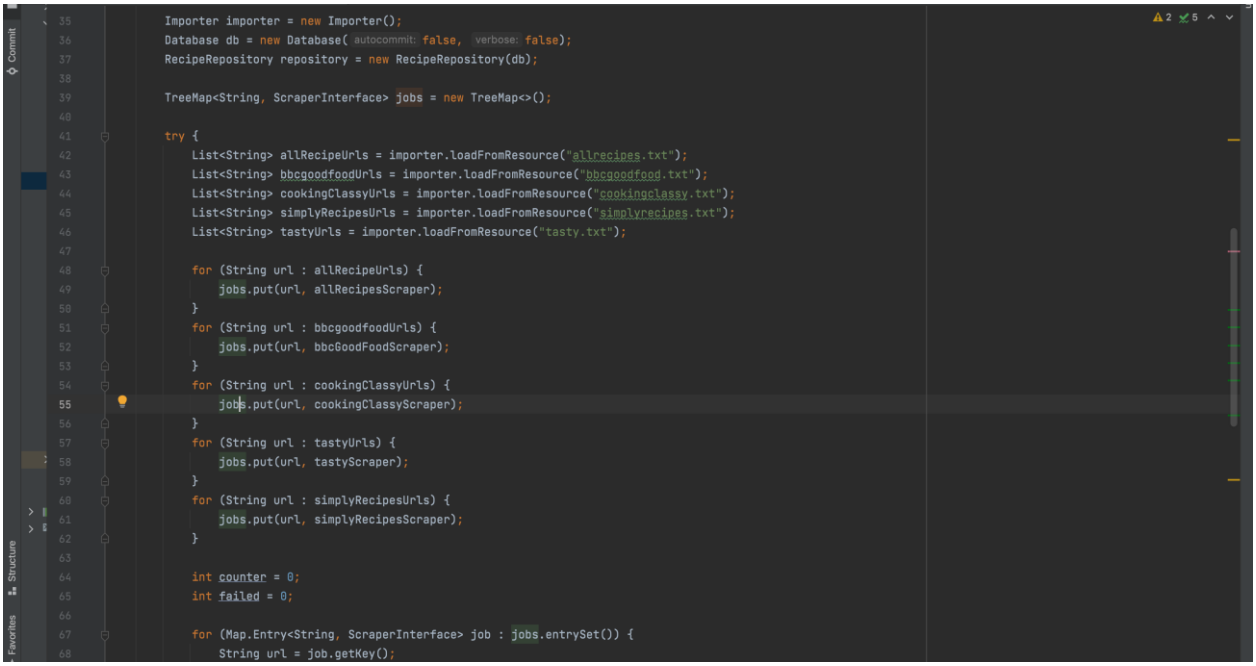
```python
from scrapy import Item, Field
from scrapy.spiders import CrawlSpider, Rule
from scrapy.linkextractors import LinkExtractor

class CookingClassyCrawler(CrawlSpider):
    name = 'cookingclassy'
    allowed_domains = ['www.cookingclassy.com']
    start_urls = ['https://www.cookingclassy.com/recipes']
    rules = (Rule(LinkExtractor(), callback='parse_item', follow=True),)
    filename = 'cookingclassy.txt'
    accept_url = "https://www.cookingclassy.com/"
    reject_urls = ["https://www.cookingclassy.com/tag/", "https://www.cookingclass

    def parse_item(self, response):
        if response.status == 200:
            url = response.url

            item = { }
            item['url'] = response.url
            item['referer'] = response.request.headers.get('Referer')
            item['status'] = response.status

            if url.startswith(self.accept_url) and not "comment" in url:
                matches = True
                for prefix in self.reject_urls:
                    if url.startswith(prefix):
                        matches = False

                if matches:
                    with open(self.filename, 'at') as f:
                        f.write(url +"\n")

            return item
```

Εικόνα 62: Το αρχείο του crawler για το cookingclassy.com

Η main των scrapers είναι όπως φαίνεται παρακάτω:

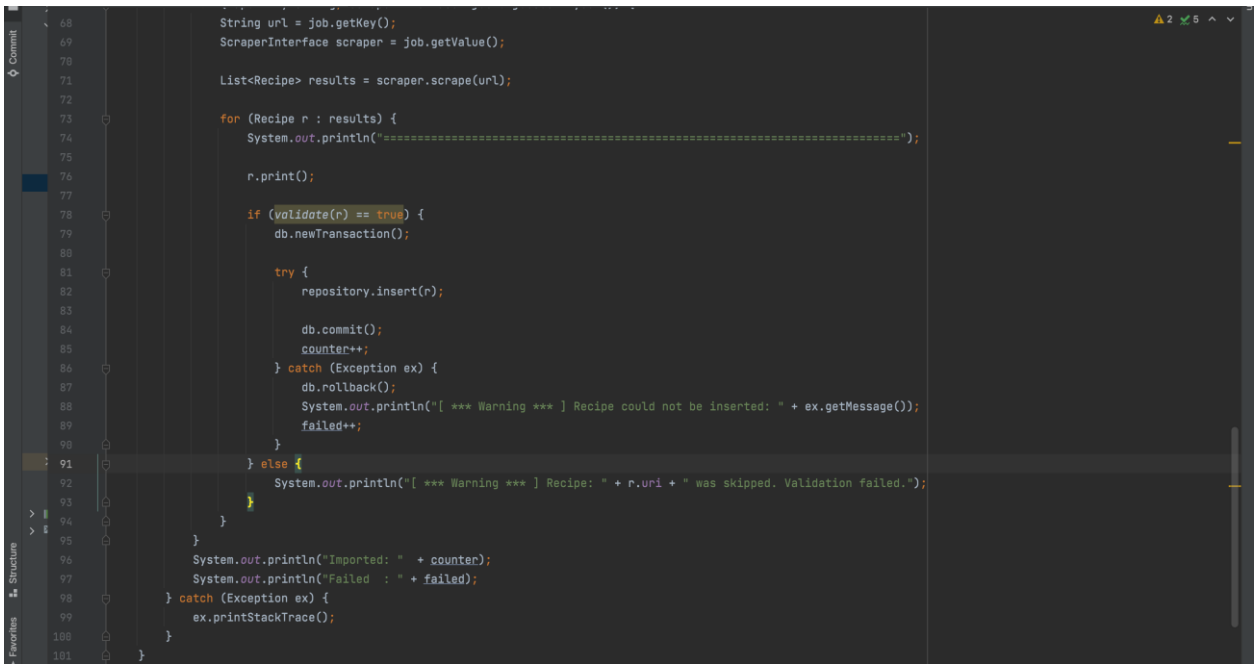
```java
import crawler.Importer;
import database.Database;
import model.Recipe;
import repository.RecipeRepository;
import scraper.*;

import java.util.List;
import java.util.Map;
import java.util.TreeMap;

public class Main {
    private static boolean validate(Recipe r) {

        if (r.nutritionInfo.kcal <= 0 || r.nutritionInfo.sugars<=0 || r.nutritionInfo.saturated_fat<=0 || r.nutritionInfo.protein<=0 || r.nutritionInfo.carbohydrates<
            return false;
        }
        if (r.preparation_time <= 0 || r.cooking_time<=0) {
            return false;
        }
        if (r.number_of_ingredients<=0) {
            return false;
        }

        return true;
    }

    public static void main(String [] args) {
        ScraperInterface allRecipesScraper = new AllRecipesScraper();
        ScraperInterface bbcGoodFoodScraper = new BbcGoodFoodScraper();
        ScraperInterface cookingClassyScraper = new CookingClassyScraper();
        ScraperInterface simplyRecipesScraper = new SimplyRecipesScraper();
        ScraperInterface tastyScraper = new TastyScraper();
```

Εικόνα 63: Η main των scrapers



```java
        Importer importer = new Importer();
        Database db = new Database( autocommit: false,  verbose: false);
        RecipeRepository repository = new RecipeRepository(db);

        TreeMap<String, ScraperInterface> jobs = new TreeMap<>();

        try {
            List<String> allRecipeUrls = importer.loadFromResource("allrecipes.txt");
            List<String> bbcgoodfoodUrls = importer.loadFromResource("bbcgoodfood.txt");
            List<String> cookingClassyUrls = importer.loadFromResource("cookingclassy.txt");
            List<String> simplyRecipesUrls = importer.loadFromResource("simplyrecipes.txt");
            List<String> tastyUrls = importer.loadFromResource("tasty.txt");

            for (String url : allRecipeUrls) {
                jobs.put(url, allRecipesScraper);
            }
            for (String url : bbcgoodfoodUrls) {
                jobs.put(url, bbcGoodFoodScraper);
            }
            for (String url : cookingClassyUrls) {
                jobs.put(url, cookingClassyScraper);
            }
            for (String url : tastyUrls) {
                jobs.put(url, tastyScraper);
            }
            for (String url : simplyRecipesUrls) {
                jobs.put(url, simplyRecipesScraper);
            }

            int counter = 0;
            int failed = 0;

            for (Map.Entry<String, ScraperInterface> job : jobs.entrySet()) {
                String url = job.getKey();
```

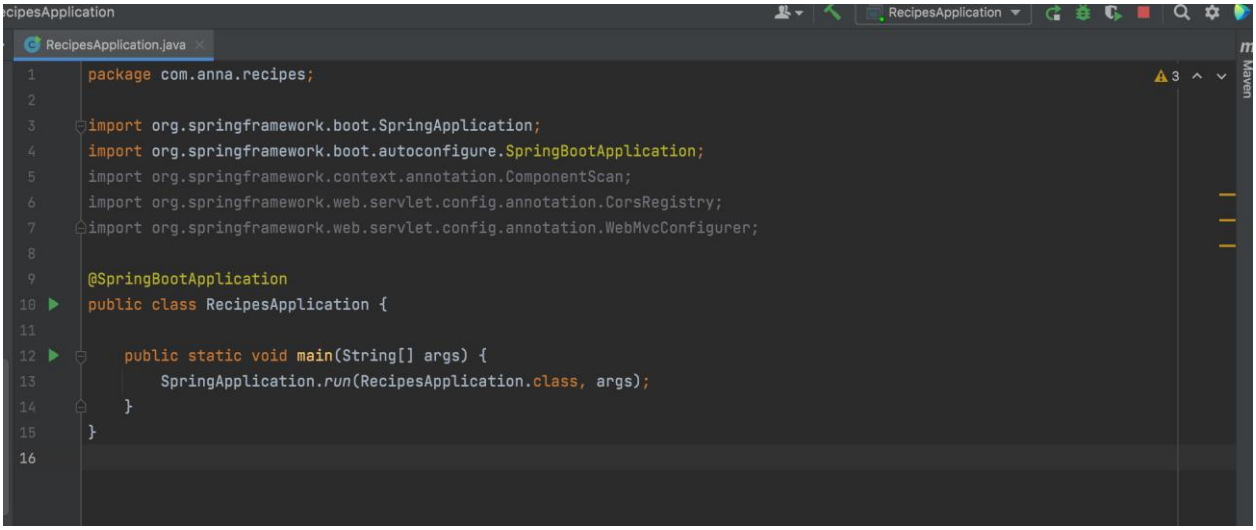
Εικόνα 64: Συνέχεια της main



```
                String url = job.getKey();
                ScraperInterface scraper = job.getValue();

                List<Recipe> results = scraper.scrape(url);

                for (Recipe r : results) {
                    System.out.println("================================================================================");

                    r.print();

                    if (validate(r) == true) {
                        db.newTransaction();

                        try {
                            repository.insert(r);

                            db.commit();
                            counter++;
                        } catch (Exception ex) {
                            db.rollback();
                            System.out.println("[ *** Warning *** ] Recipe could not be inserted: " + ex.getMessage());
                            failed++;
                        }
                    } else {
                        System.out.println("[ *** Warning *** ] Recipe: " + r.uri + " was skipped. Validation failed.");
                    }
                }
            }
            System.out.println("Imported: "  + counter);
            System.out.println("Failed  : " + failed);
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
```

Εικόνα 65: Συνέχεια της main

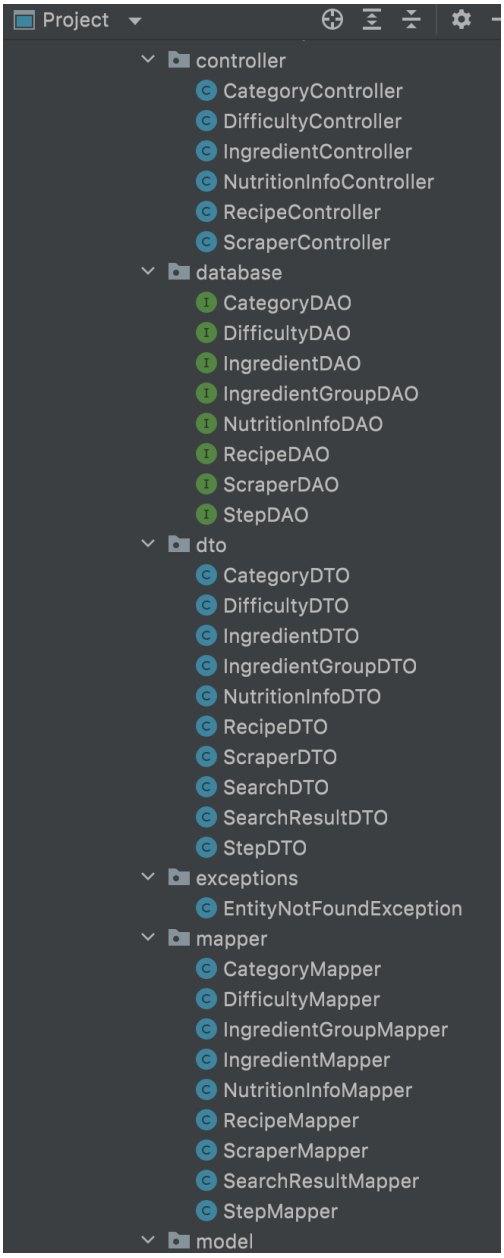
Ο κώδικας του κάθε scraper είναι διαφορετικός.
Η “main” του backend είναι η “RecipesApplication”, που φαίνεται παρακάτω:



```
package com.anna.recipes;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@SpringBootApplication
public class RecipesApplication {

    public static void main(String[] args) {
        SpringApplication.run(RecipesApplication.class, args);
    }
}
```

Εικόνα 66: Backend - Main

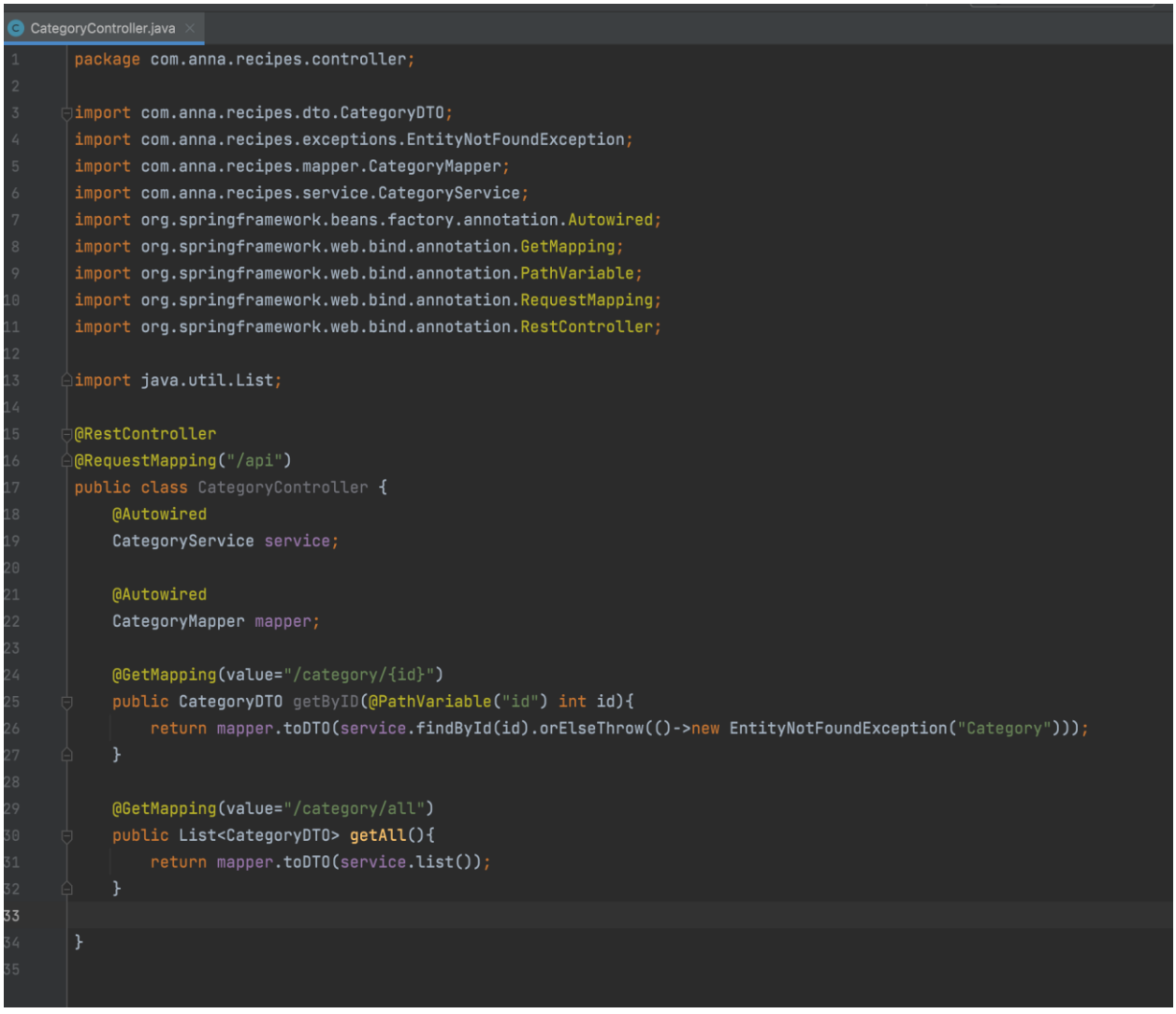
Στο πλαίσιο του backend έχουμε χρησιμοποιήσει DAOs,DTOs,Mappers,Controllers, για κάθε οντότητα της βάσης όπως φαίνεται παρακάτω:



Εικόνα 67: Αρχεία του backend

Ενδεικτικά ακολουθεί ο κώδικας καθενός αρχείου για την οντότητα “category” :

Controller:



Εικόνα 68: Κώδικας controller για category

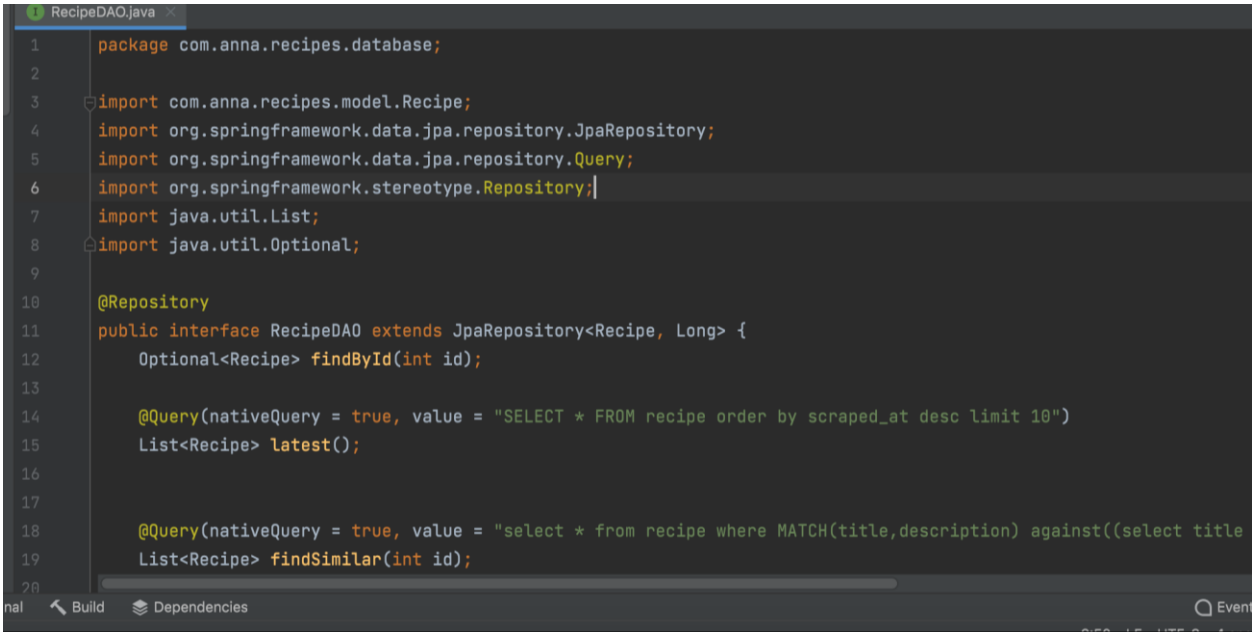
DAO(Data Access Object):

CategoryDAO.java

```java
package com.anna.recipes.database;

import com.anna.recipes.model.Category;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.Optional;

@Repository
public interface CategoryDAO extends JpaRepository<Category, Long> {
    Optional<Category> findById(int id);

}
```

Εικόνα 69: Κώδικας DAO για category

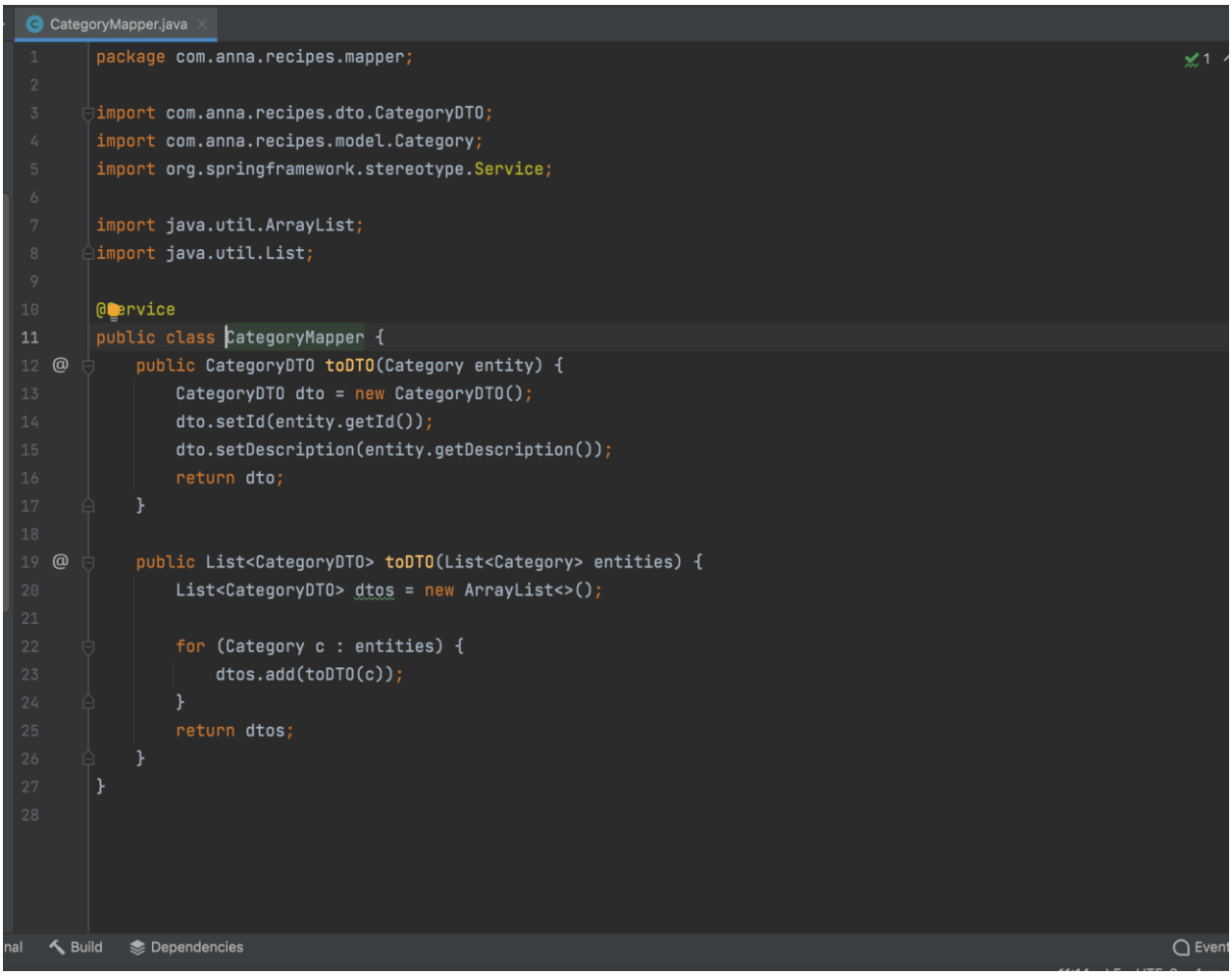
Ακολουθεί επίσης το DAO του “Recipe” το οποίο κάνει παραπάνω queries(δεν κάνει απλά εύρεση με βάση Id όπως τα υπόλοιπα). Τα φέρνει σε σειρά με βάση τη χρονική στιγμή που έγιναν scraped και κάνει και τα απαραίτητα «ταιριάσματα» στον τίτλο και την περιγραφή.



Εικόνα 70: Κώδικας DAO για recipe

DTO(data transaction object):

Mapper:



Εικόνα 71: Κώδικας mapper για category

4.2 Πίνακες με χρήσιμα στατιστικά και πληροφορίες

Πίνακας 1: Links που εισήχθησαν στη βάση δεδομένων

Allrecipes.com	Bbcgoodfood.com	Tasty.co	Cookingclassy.com	Simplyrecipes.com
9750	8307	286	1064	1439

Πίνακας 2: Μνήμη που απαιτήθηκε για κάθε αρχείο txt που περιλαμβάνει τα links με τις συνταγές του κάθε σάιτ.

Simplyrecipes.txt	Tasty.txt	Allrecipes.txt	Bbcgoodfood.txt	Cookingclassy.txt
177.012 bytes	154.282 bytes	925.037 bytes	947.762 bytes	106.697 bytes

Πίνακας 3: Μνήμη που απαιτήθηκε για τη βάση δεδομένων

Recipe_db	MySql
77.3437500 MB	2.5937500 MB

Πίνακας 4: Μέσος χρόνος scraping και crawling

Scraping	Crawling(fetching links of each site)
> 2 recipes / 1 sec	>2500 /1 hour

5. ΣΥΜΠΕΡΑΣΜΑΤΑ

5.1 Γενικά Συμπεράσματα

Οι τεχνολογίες έχουν εξελιχθεί πολύ. Ένα απλό site πια δεν είναι απλά μια html σελίδα, αλλά είναι πολλές τεχνολογίες που αλληλοεπιδρούν μεταξύ τους με τους τρόπους που είδαμε.

Ουσιαστικά, σκοπός του project είναι να διευκολύνει το χρήστη προσφέροντάς του πρόσβαση σε συνταγές από πολλές διαφορετικές πηγές, καθώς και ευέλικτους και βολικούς τρόπους αναζήτησης που θα τον διευκολύνουν πολύ καλύτερα από το να έψαχνε σε κάθε site ξεχωριστά. Αντί να αναζητά δηλαδή σε κάθε μηχανή αναζήτησης και έπειτα σε κάθε διαφορετικό ιστότοπο, έχουμε συγκεντρώσει την πληροφορία από πολλές διαφορετικές πηγές, ώστε να επιταχύνουμε και να διευκολύνουμε τον χρήστη.

5.2 Δυνατότητες Επέκτασης

Η εφαρμογή αυτή είναι δυνατό να επεκταθεί στο μέλλον και να προστεθούν κι άλλα features. Για παράδειγμα, οι scrapers μπορούν να σαρώνουν τις εικόνες των συνταγών, να τις περνάμε με links στη ΒΔ και στη συνέχεια να τις στέλνουμε κι αυτές στο frontend ώστε να παρουσιάζονται πιο όμορφα τα αποτελέσματα στο χρήστη. Επιπλέον, θα μπορούσαν να προστεθούν και άλλες περιπτώσεις αναζητήσεων, να επεκταθεί ο σχεδιασμός της διεπαφής ώστε να παρέχει περισσότερες δυνατότητες στο χρήστη ή και να φτιαχτούν και άλλοι scrapers για άλλα sites, έτσι ώστε να υπάρχει ακόμη μεγαλύτερος όγκος συνταγών στη βάση μας.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Web Crawling	Ανίχνευση Ιστού
Web Scraping	Σάρωση Ιστού
Database	Βάση Δεδομένων
Site	Ιστότοπος
User Interface	Διεπαφή Χρήστη
Link	Σύνδεσμος
Feature	Δυνατότητα/ Χαρακτηριστικό
Input	Εισαγόμενα δεδομένα
Crawler	Ανιχνευτής
Development	Ανάπτυξη
Scraper	Σαρωτής
Project	Έργο
Design pattern	Μοτίβο Σχεδίασης
Deployment	Ανάπτυξη
Components	Συστατικά
Framework	Δομή
Template	Πρότυπο
Data	Δεδομένα
Extract data	Απόσπαση δεδομένων
End-to-end	Από άκρη σε άκρη
Servers	Διακομιστές
Import	Εισαγωγές
Portable	Φορητή
Configurations	Διαμορφώσεις
Settings	Ρυθμίσεις
Documentation	Τεκμηρίωση
Validation	Επικύρωση
Run	Εκτέλεση
Entity	Οντότητα
Request	Αίτημα
Browser	Προγράμματα περιήγησης
Query	Ερώτημα
Continue	Συνέχεια
Search	Αναζήτηση
Close	Κλείσιμο
Terminal	Τερματικό

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

UI	User Interface
Txt	Text file
DAO	Data Access Object
DTO	Data Transaction Object
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
Json	JavaScript Object Notation
PC	Personal computer
BCNF	Boyce–Codd Normal Form
MVC	Model view controller
MTV	Model template views
API	Application programming interface
Rest	Representational state transfer
ΕΚΠΑ	Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

ΑΝΑΦΟΡΕΣ

[1] *Web scraping* (2023) *Wikipedia*. Wikimedia Foundation. Διαθέσιμο στο: https://en.wikipedia.org/wiki/Web_scraping (Προσπελάστηκε: 1/11/2021).
[2] *Web Crawler* (2022) *Wikipedia*. Wikimedia Foundation. Διαθέσιμο στο: https://en.wikipedia.org/wiki/Web_crawler (Προσπελάστηκε: 1/11/2021).
[3] *Representational state transfer* (2022) *Wikipedia*. Wikimedia Foundation. Διαθέσιμο στο: https://en.wikipedia.org/wiki/Representational_state_transfer (Προσπελάστηκε: 12/2/2022).
[4] *Frontend and backend* (2022) *Wikipedia*. Wikimedia Foundation. Διαθέσιμο στο: https://en.wikipedia.org/wiki/Frontend_and_backend (Προσπελάστηκε: 5/3/2022).
[5] *Relational database* (2022) *Wikipedia*. Wikimedia Foundation. Διαθέσιμο στο: https://en.wikipedia.org/wiki/Relational_database (Προσπελάστηκε: 19/1/2022).
[6] *NoSQL* (2022) *Wikipedia*. Wikimedia Foundation. Διαθέσιμο στο: https://en.wikipedia.org/wiki/NoSQL (Προσπελάστηκε: 19/1/2022).
[7] *A fast and powerful scraping and web crawling framework* (no date) *Scrapy*. Διαθέσιμο στο: https://scrapy.org/ (Προσπελάστηκε: 1/3/2022).
[8] Committee, A.N.P.M. (no date) *Apache nutch™*, *Apache Nutch™*. Διαθέσιμο στο: https://nutch.apache.org/ (Προσπελάστηκε: 21/11/2021).
[9] *Heritrix 3 documentation* (no date) *Heritrix 3 Documentation - Heritrix 3 documentation*. Διαθέσιμο στο: https://heritrix.readthedocs.io/en/latest/ (Προσπελάστηκε: 10/1/2022).
[10] *Spring Boot* (no date) *Spring.* Διαθέσιμο στο: https://spring.io/projects/spring-boot (Προσπελάστηκε: 26/3/2022).
[11] *The PHP framework for web artisans* (no date) *Laravel*. Διαθέσιμο στο: https://laravel.com/ (Προσπελάστηκε: 30/3/2022).
[12] *Django* (no date) *Django Project*. Διαθέσιμο στο:https://www.djangoproject.com/. (Προσπελάστηκε: 19/1/2022).
[13] *React – a JavaScript library for building user interfaces* (no date) *– A JavaScript library for building user interfaces*. Διαθέσιμο στο: https://reactjs.org/ (Προσπελάστηκε: 1/5/2022).
[14] *The progressivejavascript framework* (no date) *Vue.js - The Progressive JavaScript Framework | Vue.js*. Διαθέσιμο στο:https://vuejs.org/ (Προσπελάστηκε: 1/6/2022).
[15] (no date) *Angular*. Διαθέσιμο στο: https://angular.io/ (Προσπελάστηκε: 18/5/2022).
[16] (no date) *MySQL*. Διαθέσιμο στο:https://www.mysql.com/ (Προσπελάστηκε: 15/1/2022).
[17] Group, P.S.Q.L.G.D. (2023) *PostgreSQL*. Διαθέσιμο στο:https://www.postgresql.org/ (Προσπελάστηκε: 15/1/2022).
[18] *MySQL Workbenchdownload now "* (no date) *MySQL*. Διαθέσιμο στο: https://www.mysql.com/products/workbench/ (Προσπελάστηκε: 17/1/2022).
[19] *MySQL after insert trigger by practical examples* (2020) *MySQL Tutorial*. Διαθέσιμο στο: https://www.mysqltutorial.org/mysql-triggers/mysql-after-insert-trigger/. (Προσπελάστηκε: 26/1/2022).
[20] *MD5* (2022) *Wikipedia*. Wikimedia Foundation. Διαθέσιμο στο: https://en.wikipedia.org/wiki/MD5 (Προσπελάστηκε: 9/2/2022).
[21] *Java HTML parser, built for html editing, cleaning, scraping, and XSS Safety* (no date) *jsoup*. Διαθέσιμο στο: https://jsoup.org/ (Προσπελάστηκε: 17/2/2022).
[22] *IntelliJ IDEA – the leading Java and Kotlin Ide* (no date) *JetBrains*. Διαθέσιμο στο: https://www.jetbrains.com/idea/ (Προσπελάστηκε: 1/9/2021).
[23] Porter, B., Zyl, J.van and Lamy, O. (no date) *Welcome to Apache Maven*, *Maven*. Διαθέσιμο στο: https://maven.apache.org/ (Προσπελάστηκε: 1/9/2021).
[24] *Data Access Object* (no date) *Design Patterns: Data Access Object*. Διαθέσιμο στο: https://www.oracle.com/java/technologies/data-access-object.html (Προσπελάστηκε: 16/4/2022).
[25] *Spring - Rest Controller* (2021) *GeeksforGeeks*. Διαθέσιμο στο: https://www.geeksforgeeks.org/spring-rest-controller/ (Προσπελάστηκε: 20/3/2022).
[26] Seniuk, V. (2020) *Data Transfer Object Pattern in java - implementation and mapping*, *Stack Abuse*. Stack Abuse. Διαθέσιμο στο: https://stackabuse.com/data-transfer-object-pattern-in-java-implementation-and-mapping/ (Προσπελάστηκε: 22/3/2022).
[27] Lubowa, E. (2022) *Why you need to use DTO's in your rest api*, *Medium*. Medium. Διαθέσιμο στο: https://medium.com/@enocklubowa/why-you-need-to-use-dtos-in-your-rest-api-d9d6d7be5450 (Προσπελάστηκε: 12/4/2022).
[28] Node.js (no date) *Node.js*. Διαθέσιμο στο: https://nodejs.org/en/ (Προσπελάστηκε: 7/6/2022).
[29] *Vue Cli* (no date) *Vue CLI*. Διαθέσιμο στο: https://cli.vuejs.org/ (Προσπελάστηκε: 8/6/2022).
[30] Microsoft (2021) *Visual studio code - code editing. redefined*, *RSS*. Microsoft. Διαθέσιμο στο: https://code.visualstudio.com/ (Προσπελάστηκε: 14/4/2022).
[31] *The Intuitive Vue Framework* (no date) *Nuxt*. Διαθέσιμο στο: https://nuxtjs.org/ (Προσπελάστηκε: 28/4/2022).

[32] *FoodHub - Food Delivery Template* (no date) *Vuetify Store*. Διαθέσιμο στο: https://store.vuetifyjs.com/products/foodhub-food-delivery-template?utm_source=vuetifyjs.com&utm_medium=themes (Προσπελάστηκε: 3/5/2022).
[33] *A material design framework for vue.js* (no date) *Vuetify*. Διαθέσιμο στο: https://vuetifyjs.com/en/ (Προσπελάστηκε: 4/5/2022).
[34] *Babel · The compiler for next generation javascript* (no date) *Babel*. Διαθέσιμο στο: https://babeljs.io/ (Προσπελάστηκε: 7/5/2022).
[35] *Boyce–Codd Normal Form* (2022) *Wikipedia*. Wikimedia Foundation. Διαθέσιμο στο: https://en.wikipedia.org/wiki/Boyce%E2%80%93Codd_normal_form (Προσπελάστηκε: 6/1/2022).
[36] (no date) *Postman*. Διαθέσιμο στο: https://www.postman.com/ (Προσπελάστηκε: 1/3/2022).
[37] *API testing client that flows with you* (no date) *HTTPie*. Διαθέσιμο στο: https://httpie.io/ (Προσπελάστηκε: 12/5/2022).
[38] *PyCharm: The python IDE for professional developers by jetbrains* (no date) *JetBrains*. Διαθέσιμο στο: https://www.jetbrains.com/pycharm/ (Προσπελάστηκε: 18/12/2021).