Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Επικοινωνιών, Ηλεκτρονικής και
Συστημάτων Πληροφορικής

Δικτυακή εφαρμογή εύρεσης προσφορών σε δεδομένα του Skroutz API

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΖΑΝΝΗΣ ΚΑΛΑΜΠΟΥΚΗΣ

Επιβλέπων : Ευστάθιος Συκάς
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2017

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Επικοινωνιών, Ηλεκτρονικής και
Συστημάτων Πληροφορικής

Δικτυακή εφαρμογή εύρεσης προσφορών σε δεδομένα του Skroutz API

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΖΑΝΝΗΣ ΚΑΛΑΜΠΟΥΚΗΣ

Επιβλέπων : Ευστάθιος Συκάς
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 19η Ιουνίου 2017.

.....	.....	.....
Ευστάθιος Συκάς Καθηγητής Ε.Μ.Π.	Γεώργιος Στασινόπουλος Καθηγητής Ε.Μ.Π.	Ιωάννα Ρουσσάκη Επίκ. Καθηγήτρια Ε.Μ.Π.

Αθήνα, Ιούνιος 2017

.....

Ζαννής Καλαμπούκης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Στα πλαίσια της παρούσας διπλωματικής εργασίας, σχεδιάστηκε και αναπτύχθηκε μία πλήρης δικτυακή εφαρμογή εύρεσης προσφορών σε προϊόντα που παρέχονται απο το δικτυακό τόπο Skroutz.gr.

Σκοπός της εφαρμογής είναι να λύσει μία πραγματική ανάγκη, να παρέχει σε υποψήφιους αγοραστές τη δυνατότητα να εντοπίζουν πραγματικά καλές ευκαιρίες δεδομένου ότι η τιμολόγηση των παρεχόμενων προϊόντων μεταβάλλεται σε τακτά χρονικά διαστήματα. Με τον τρόπο αυτό, ο χρήστης αποκτά άμεσα την πληροφορία για πιθανές συμφέρουσες αγορές, αποφεύγοντας την ενοχλητική διαδικασία της χειροκίνητης αναζητησης.

Η εφαρμογή σχεδιάστηκε με γνώμονα τη λειτουργικότητα, την αμεσότητα και την αξιοπιστία. Δόθηκε ιδιαίτερη έμφαση στη συχνή ανανέωση των δεδομένων και στην εύκολη χρήση της διεπαφής, καθώς κύρια επιδίωξη μας είναι μία ταχεία και αποτελεσματική αλληλεπίδραση, η οποία θα οδηγεί σε προστιθέμενη αξία και ανάπτυξη αισθήματος εμπιστοσύνης στο χρήστη. Σχεδιαστικά, βασιστήκαμε στην αποδεδειγμένη λύση της μηχανής αναζήτησης, εμπλουτίζοντάς τη με πληροφορία για τις βέλτιστες προσφορές τη δεδομένη στιγμή.

Για την υλοποίηση των παραπάνω, με βάση την αρχιτεκτονική τριών επιπέδων, ο σχεδιασμός της εφαρμογής έγινε εκ νέου σε όλα τα επίπεδα (Εξυπηρετητής, Βάση Δεδομένων, Μοντελοποίηση, Λογική λειτουργία, Διεπαφή χρήστη). Έγινε χρήση τελευταίων τεχνολογιών σε κάθε ένα από αυτά, ενώ κάθε αρχιτεκτονική απόφαση σε θέμα αρχιτεκτονικής της παρούσας εργασίας ελήφθη με σεβασμό στις βέλτιστες πρακτικές σχεδιασμού, ασφάλειας, επεκτασιμότητας και ακολουθώντας με συνέπεια τις οδηγίες ορθού προγραμματισμού. Ταυτόχρονα, το εγχείρημα θα ήταν αδύνατο χωρίς τη προγραμματιστική υλοποίηση ώριμων στατιστικών ελέγχων για εύρεση ακραίων τιμών, στην ορθότητα και εφαρμοσιμότητα των οποίων δόθηκε ιδιαίτερη έμφαση καθώς αποτελούν τον ερευνητικό σκελετό της εφαρμογής. Με τον τρόπο αυτό δημιουργήσαμε ένα σύστημα αποτελούμενο από αυτοτελείς μονάδες που αλληλεπιδρούν μεταξύ τους, δημιουργώντας μία πλήρως λειτουργική εφαρμογή, ενώ ταυτόχρονα μπορούν να επαναχρησιμοποιηθούν με ελάχιστες επεμβάσεις, για εξολοκλήρου νέες εφαρμογές.

Λέξεις κλειδιά

Στατιστικός έλεγχος, Εντοπισμός εκτροπών, Δικτυακή εφαρμογή, Βάση δεδομένων, Διεπαφή Προγραμματισμού Εφαρμογών

Abstract

Within the scope of this diploma thesis, a full-stack Web Application was designed and developed. The application aims to locate and suggest possible offers on products available on the online shopping aggregator Skroutz.gr.

The main goal of this application is to address a real-world need, which is to enable consumers make optimal purchases based on product pricing patterns, given that prices are subject to frequent change. This way, users are able to instanly receive the information of favorable options, avoiding the annoying process of manual market research.

The web application is designed with emphasis on the completeness of the desired functionality, responsiveness and reliability. Special attention was given to the robustness of the service, frequent dataset updates and the user experience, as we prioritized achieving quick and responsive human-machine interaction, leading to added value and a feeling of trustworthiness to the user. On the subject of design, we built upon the proven solution of a search engine, enriching it with information about the best offers at any given time.

In order to implement the previous points, we planned a three tier architecture, designing and implementing it throughout the web stack including the Web Server, Database, Data Modelling, Business Logic and User Interface. We used modern technologies on each of the forementioned modules and all critical decisions were made with respect to the best practices on design, security, extensibility, scalability and consistently following the guidelines of robust programming. Admittedly, developing this application would be impossible without programmatically performing proven statistical tests on outlier detection, whose correct and sound application was of very high importance throughout this operation, as they are effectively the cornerstone of research in this diploma thesis. As a result, we built a system comprising of independent, loosely-coupled modules which together result in a fully functioning web application that is reusable in new, extended applications with minimal effort.

Key words

Statistical test, Outlier detection, Web Application, Database, REST API

Ευχαριστίες

Ευχαριστώ θερμά τον επιβλέποντα καθηγητή αυτής της διατριβής, κ. Ευστάθιο Συκά, καθώς και τον υποψήφιο διδάκτορα Βασίλη Ασθενόπουλο για την εμπιστοσύνη τους και την καθοδήγησή τους όπου χρειάστηκε. Επίσης θέλω να ευχαριστήσω τους γονείς μου, την κοπέλα μου και τους φίλους μου για την ασταμάτητη υποστήριξή και υπομονή που μου έδωσαν καθόλη τη φοιτητική μου ζωή και ιδιαίτερα κατά την περίοδο εκπόνησης της παρούσας διπλωματικής.

Ζαννής Καλαμπούκης,

Αθήνα, 19η Ιουνίου 2017

Περιεχόμενα

Κατάλογος σχημάτων

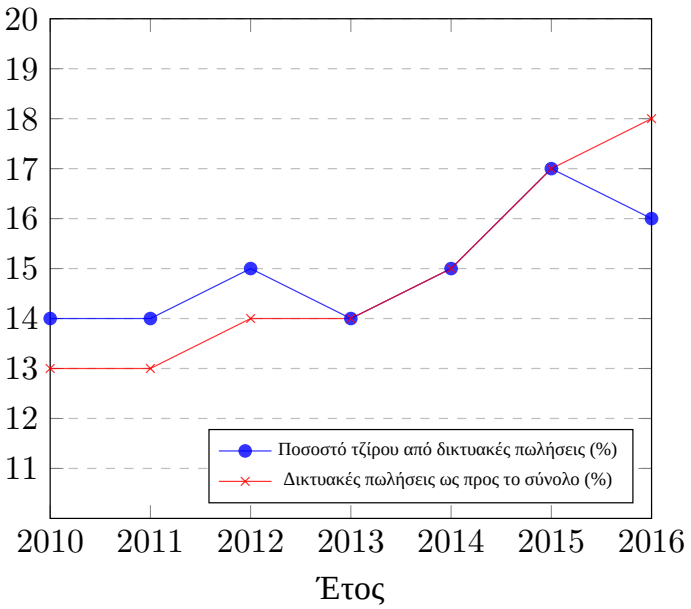
Κατάλογος αποσπασμάτων κώδικα

Κεφάλαιο 1

Εισαγωγή

1.1 Διαδικτυακό εμπόριο και ευκαιρίες

Είναι γεγονός πως την τελευταία δεκαετία, η εξέλιξη του διαδικτυακού εμπορίου κυλά με ταχύτατους ρυθμούς σε ευρωπαϊκό αλλά και παγκόσμιο επίπεδο. Έχει ξεπεραστεί η εποχή όπου υπήρχαν αμφιβολίες ως προς την ασφάλεια και την αξιοπιστία των αγοραπωλήσιών στο μέσο αυτό και μάλιστα, εντύπωση προκαλεί το γεγονός ότι ορισμένα δικτυακά καταστήματα καταλαμβάνουν μεγαλύτερο μερίδιο της αγοράς σε σχέση με εδραιωμένα φυσικά καταστήματα. Στο διάγραμμα 1.1 φαίνονται οι ανοδικές τάσεις ενσωμάτωσης του δικτυακού εμπορίου σύμφωνα με ετήσιες έρευνες της Eurostat [1].



Σχήμα 1.1: Δικτυακό εμπόριο στις 28 χώρες της Ευρώπης

Δημιουργήθηκε με τον τρόπο αυτό η ευκαιρία ανάπτυξης ενός οικοσυστήματος γύρω από την ηλεκτρονική επιχειρηματικότητα. Την τελευταία δεκαπενταετία αναδύονται καθημερινά καινοτόμες δικτυακές εφαρμογές σχεδιασμένες να εξυπηρετήσουν κάθε είδους ανάγκη. Η κατάσταση αυτή συνοδεύεται φυσικά από έντονο ανταγωνισμό λόγω της προσιτότητάς του σε πολλών ειδών τεχνικούς και επιχειρηματίες, ενώ βασίζεται κατά μεγάλο βαθμό στα κοινωνικά δίκτυα και στις κοινότητες που δημιουργούνται σε αυτά. Τα παραπάνω, σε συνδυασμό με την κουλτούρα του ανοιχτού λογισμικού, δημιουργούν ένα συλλογικό, ιεραρχικό πλαίσιο ανάπτυξης, το οποίο εξελίσσεται με ραγδαίους ρυθμούς.

Μία κατηγορία τέτοιων εφαρμογών είναι γνωστή ως Συλλέκτες (aggregators). Οι τελευταίοι ουσιαστικά ανέλαβαν να κατηγοριοποιήσουν και να παρουσιάσουν ορισμένες πληροφορίες, δηλαδή,

[1] http://ec.europa.eu/eurostat/data/database

στην περιπτωσή μας τα διαθέσιμα καταστήματα, προϊόντα ή υπηρεσίες προς πώληση. Εφόσον ολοκληρωθεί αυτή η κατηγοριοποίηση, γίνεται δυνατή η αναζήτηση προϊόντων με βάση διάφορα μεταδεδομένα, όπως η τιμή, τα χαρακτηριστικά, η διαθεσιμότητα, ο τρόπος πληρωμής και η απόσταση από το κατάστημα. Παρατηρούμε, λοιπόν, ότι με τον τρόπο αυτό, δημιουργήθηκε υπερβάλλουσα αξία και για τους ιδιοκτήτες των καταστημάτων, οι οποίοι προβλήθηκαν ξαφνικά σε μία πολύ μεγάλη βάση χρηστών, αλλά και στους καταναλωτές, οι οποίοι με τη σειρά τους απέκτησαν πολλαπλές επιλογές, τις οποίες πιθανότατα αγνοούσαν. Το εμπορικό μοντέλο άλλαξε άρδην χάρη στη διείσδυση που είχαν οι Συλλέκτες στην αγορά. Οι αποστάσεις πλέον έγιναν αμελητέες και αυτό έδωσε κίνητρα και ευκαιρίες σε μικρούς κεφαλαιακά εμπόρους να αποκτήσουν ένα αξιοπρεπές μερίδιο της αγοράς. Βασικά παραδείγματα ιστότοπων που συλλέγουν και παρουσιάζουν τέτοια πληροφορία στην Ελλάδα είναι το Skroutz.gr και Bestprice.gr, ενώ παγκοσμίως τα πιο γνωστά παραδείγματα είναι το amazon.com και το ebay.com.

Παρατηρούμε, λοιπόν, ότι η εισαγωγή ενός νέου επιχειρηματικού ιστότοπου δημιουργεί αξία για κάθε επαγγελματία του χώρου καθώς αποτελεί ένα νέο κανάλι μετάδοσης δεδομένων και ταυτόχρονα ένα μηχανισμό εξαγωγής χρήσιμων συμπερασμάτων. Τελευταία γίνονται πολλές αναφορές στην “Ανάλυση Μεγάλων Δεδομένων” (Big Data Analytics) ή διαφορετικά την εξαγωγή πληροφορίας από την προγραμματιστική επεξεργασία μεγάλων όγκων δεδομένων με χρήση κατάλληλων αλγορίθμων. Όσο καλύτερη η ποιότητα των δεδομένων, τόσο πιο ασφαλή και ουσιαστικά είναι τα στατιστικά αποτελέσματα που εξάγονται με κατάλληλη διαδικασία από αυτό. Στην επιστήμη αλλά και την αγορά των υπηρεσιών Πληροφορικής, η εκμαίευση δεδομένων και η εξαγωγή συμπερασμάτων είναι ένα ερευνητικό πεδίο με μεγάλο πλήθος εφαρμογών και αυτό είναι το πεδίο στο οποίο στοχεύει να κατευθυνθεί η δική μας εφαρμογή.

1.2 Αντικείμενο της διπλωματικής

Η παρούσα διπλωματική εργασία αποτελεί την ερευνητική μελέτη και τυπική τεκμηρίωση της δικτυακής εφαρμογής συλλέκτη BargainHunt, όπως αυτή αναπτύχθηκε από το συγγραφέα. Αναγνωρίζοντας την αξία των αντίστοιχων υπηρεσιών όπως αναφέρθηκε παραπάνω, επιλέγουμε να δημιουργήσουμε μία μονάδα εύρεσης προσφορών, εμπλουτίζοντας την υπάρχουσα λειτουργία αντίστοιχων εφαρμογών όπως εμπλουτίσουμε τη λειτουργικότητά τους με την προσθήκη μίας μονάδας εύρεσης προσφορών. Δίνουμε τη δυνατότητα στο χρήστη να κάνει συμφέρουσες συναλλαγές εξαλείφοντας την ανάγκη για χρονοβόρες αναζητήσεις που είναι δύσκολο να επαναλαμβάνονται στο ακαίραιο και πιθανώς να προβούν άκαρπες. Μία τέτοια μονάδα εξυπηρετεί άριστα τη μοντέρνα τάση αρχιτεκτονικής δικτυακών εφαρμογών; αυτή της αποκεντροποίησης και της ανάμειξης ετερογενών Διεπαφών Προγραμματισμού για τη δημιουργία νέων, καινοτόμων εφαρμογών.

Επιλέξαμε να δημιουργήσουμε μία απλή αλλά αποτελεσματική διεπαφή χρήστη σε μορφή πρωτοτύπου, η οποία θα δύνει τη δυνατότητα στο χρήστη να αναζητήσει τα προϊόντα που επιθυμεί και να ενημερωθεί εαν βρίσκεται σε προσφορά, ενώ παράλληλα πληροφορείται για τις τελευταίες διαθέσιμες προσφορές αυτή τη στιγμή από όλα τα συνεργαζόμενα ψηφιακά καταστήματα.

Για το παραπάνω λογισμικό χρησιμοποιήσαμε τη γλώσσα προγραμματισμού Java Enterprise Edition, σε συνδυασμό με το πλαίσιο προγραμματισμού Web εφαρμογών Vaadin και πληθώρα άλλων τρίτων βιβλιοθηκών και εργαλείων. Επιπλέον, κάναμε χρήση εδραιωμένων στατιστικών ελέγχων για ύπαρξη ακροτάτων σε κανονικά στατιστικά δείγματα. Ανακαλύψαμε προγραμματιστικά το βέλτιστο τρόπο λήψης απόφασης σχετικά με την ύπαρξη ή μη κάποιας προσφοράς με βάση τα συνδυαστικά αποτελέσματα των παραπάνω.

Στα πλαίσια ανάπτυξης αυτής της εφαρμογής, λάβαμε αποφάσεις σε πλήρη κλίμακα ανάπτυξης, ξεκινώντας από το χαμηλότερο επίπεδο των δεδομένων συνεχίζοντας στη λογική των υπηρεσιών και καταλήγοντας στις τελικές διεπαφές χρήστη. Με τον τρόπο αυτό, μειώσαμε τις εξαρτήσεις από τρίτα συστήματα και στοχεύσαμε στην επεκτασιμότητα και χρησιμότητα της εφαρμογής στο σύγχρονο οικοσύστημα Διεπαφών Προγραμματισμού.

1.3 Οργάνωση της διπλωματικής

Το παρόν έγγραφο περιέχει 8 βασικά κεφάλαια στα οποία θα αναπτύξουμε όλα τα θέματα που αφορούν στο σχεδιασμό και την υλοποίηση της εφαρμογής BargainHunt. Επιπλέον περιλαμβάνει ένα Ευρετήριο Τεχνικών Όρων και την επίσημη τεχνική

Στο κεφάλαιο 2 θα ορίσουμε το πρόβλημα προς επίλυση και θα αναλύσουμε τα χαρακτηριστικά του με ακρίβεια, έτσι ώστε να γίνει αντιληπτό το εύρος των ύποπροβλημάτων που θα πρέπει να αντιμετωπίσυμε.

Στη συνέχεια, στο κεφάλαιο 3, θα μιλήσουμε για τις τεχνολογίες που χρησιμοποιήσαμε σε κάθε επίπεδο του σχεδιασμού και της υλοποίησης της εφαρμογής. Θα αιτιολογηθούν οι επιλογές μας και θα γίνει μία ενδεικτική σύγκριση μεταξύ των εργαλείων που επιλέξαμε σε σχέση με άλλα διαθέσιμα.

Το κεφάλαιο 4 είναι αφιερωμένο στην αρχιτεκτονική των διαφόρων ενοτήτων. Μία δικτυακή εφαρμογή αποτελείται από πολλαπλά επίπεδα, καθένα εκ των οποίων απαιτεί έναν πλήρη σχεδιασμό, ως μονάδα αλλά και ως μέλος ενός περίπλοκου συστήματος που βασίζεται στην ορθή και άμεση επικοινωνία των επιμέρους συνιστωσών του.

Η επόμενη ενότητα ανάλυσης αφορά στην υλοποίηση της εφαρμογής. Στο κεφάλαιο 5 θα συζητηθούν θέματα προγραμματισμού, σημεία συμφόρησης και μέθοδοι αντιμετώπισής τους, ζητήματα ασφάλειας και, τέλος, οι αλγόριθμοι που χρησιμοποιήσαμε στα πλαίσια αυτής της διπλωματικής εργασίας.

Το κεφάλαιο 6 περιέχει οδηγίες και προαπαιτούμενα για τη σωστή εγκατάσταση και διαμόρφωση όλων των μονάδων που αποτελούν την εφαρμογή, ενώ στο κεφάλαιο 7 γίνεται τεκμηρίωση και καλύπτεται πλήθος σεναρίων χρήσης.

Τέλος, στο κεφάλαιο 8, γίνεται ένας απολογισμός της παραπάνω διαδικασίας και αποτυπώνονται ορισμένες προτάσεις για μελλοντικές επεκτάσεις της εφαρμογής.

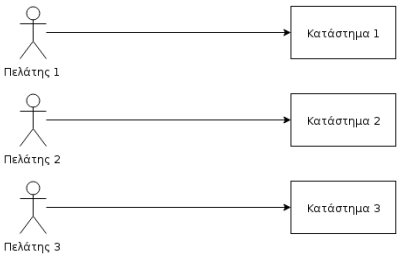
Κεφάλαιο 2

Περιγραφή του προβλήματος

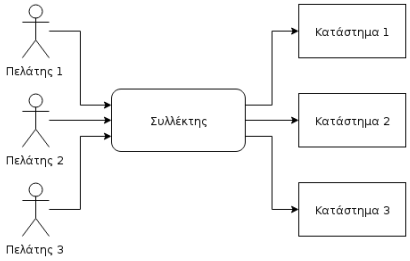
2.1 Τρέχουσα κατάσταση

Η εξάπλωση των δικτυακών αγορών έχει επηρεάσει έντονα το κλασσικό εμπορικό μοντέλο. Είναι φανερό πως στην περίοδο που ζούμε, όπου κυριαρχεί η δύναμη της πληροφορίας, τα κριτήρια επιλογής ενός αγαθού έχουν μεταβληθεί αισθητά. Οι αγορές έχουν παγκοσμιοποιηθεί και είναι διαθέσιμες από κάθε σημείο του πλανήτη, οι αποστάσεις εκμηδενίζονται και οι ανάγκες αυξάνονται. Αυτό πιθανώς να δημιουργεί ευκαιρίες για κάθε οργανισμό που επιθυμεί να συμμετάσχει στη διαδικασία. Εντύπωση, για παράδειγμα, προκαλεί η άνθιση μετα-καταστημάτων όπως οι λεγόμενοι Συλλέκτες (aggregators).

Με τον όρο Συλλέκτης αναφερόμαστε ουσιαστικά σε μία υπηρεσία, καθώς δεν διαθέτει φυσική υπόσταση, την επιχειρηματική ιδέα δηλαδή του να ορίζεις ένα σημείο στο οποίο συσσωρεύεται κατηγοριοποιημένη πληροφορία. Προστίθεται δηλαδή ένα αφαιρετικό επίπεδο συσσωρεύοντας δεδομένα, που υπό άλλες συνθήκες θα ήταν δύσκολο να αποκτήσει κανείς με συμβατικό τρόπο. Ο νέος αυτός κόμβος του δικτύου εμπεριέχει προστιθέμενη αξία, η οποία μεταφράζεται σε οικονομικό συμφέρον όλων των ενδιαφερόμενων μελών. Ίσως τα παραπάνω γίνουν περισσότερο κατανοητά μέσω του σχήματος 2.3.



Σχήμα 2.1: Παλαιό Μοντέλο



Σχήμα 2.2: Νέο Μοντέλο

Σχήμα 2.3: Εφαρμογή συλλέκτης ως αφαιρετικό επίπεδο μεταξύ καταναλωτή και καταστημάτων

Παρατηρούμε παραπάνω ότι ο νέος κόμβος που εισήχθη στο σχήμα ουσιαστικά εκμηδενίζει την περιπλοκότητα της εύρεσης του ιδανικού καταστήματος για τον καταναλωτή, προσφέροντάς του ένα αξιοσημείωτο πλεονέκτημα. Είναι φανερό ότι πλέον ο καταναλωτής, με μία απλή αναζήτηση, έχει τη δυνατότητα να συλλέξει στοιχεία από πηγές του διαδικτύου είτε του κοινωνικού του περιβάλλοντος έτσι ώστε να λάβει μία ενημερωμένη απόφαση για το προϊόν που τον ενδιαφέρει. Το καίριο σημείο είναι πλέον το προϊόν σε αντίθεση με το κατάστημα.

Αντίστοιχα, οι ιδιοκτήτες ηλεκτρονικών καταστημάτων, επωφελούνται εξίσου απο την ευρύτατη εκπροσώπηση που λαμβάνουν από τους ιστότοπους συλλέκτες όπως το Skroutz.gr. Οι τελευταίοι λειτουργούν ως ένα ψευδοκατάστημα, ένας διαμεσολαβητής μεταξύ του καταστηματάρχη και του αγοραστή. Έτσι δημιουργούνται οι προϋποθέσεις για την εδραίωση μίας ανταγωνιστικής συμπεριφοράς του πωλητή, με σκοπό τη διεκδίκηση της πρώτης θέσης στη σελίδα των αποτελεσμάτων. Με τον τρόπο που έχει διαμορφωθεί η υπηρεσία, για να κατακτήσει ένα κατάστημα την πρώτη θέση στα αποτελέσματα θα πρέπει να παρέχει το προϊόν προς αγορά στην ελάχιστη τιμή. Παρατηρούμε, συνεπώς,

ότι οι ιστότοποι Συλλέκτες λειτουργούν προσφέροντας μία διμερή υπηρεσία και κερδοφορώντας από αυτή.

Ταυτόχρονα, το ίδιο φαινόμενο παρατηρείται και σε άλλες περιπτώσεις ανακάλυψης πληροφορίας. Κλασσικό παράδειγμα συλλέκτη είναι η μηχανή αναζήτησης της Google που πλέον μπορεί να θεωρηθεί ενδεχομένως ένας από τους πιο κεντρικούς κόμβους του διαδικτύου, με στόχο την ανακατεύθυνση του χρήστη στην πληροφορία που επιθυμεί. Αντίστοιχη εφαρμογή μπορεί να θεωρηθεί η εφαρμογή TripAdvisor, όπου περιλαμβάνει ταξιδιωτικούς προορισμούς σε συνδυασμό με αξιολογήσεις για κάθε λογής αξιοθέατα. Άλλες αντίστοιχες εφαρμογές είναι το IMDb.com, Metacritic.com, Feedly.com για συσσώρευση ταινιών, μέσων ψυχαγωγίας και κοινωνικών δικτύων αντίστοιχα.

2.2 Περιγραφή της διπλωματικής

Η παρούσα διπλωματική εργασία έχει ως τελικό στόχο τη δημιουργία μίας υπηρεσίας εύρεσης προσφορών κάνοντας χρήση των δεδομένων που διατίθενται από τη Διεπαφή Προγραμματιστικών Εφαρμογών του ιστότοπου Skroutz.gr. Για την επίτευξη αυτού του σκοπού, θα επικαλεστούμε την αναλογία της προσφοράς σε ένα προϊόν με την έκτοπη τιμή σε ένα στατιστικό δείγμα τιμών. Στη συνέχεια η τεχνοτροπία αυτή θα εφαρμοστεί προγραμματιστικά και θα διατεθεί για χρήση ώς δικτυακή υπηρεσία.

Το παρόν έργο υλοποιήθηκε ως μία ολοκληρωμένη διαδικτυακή εφαρμογή, ένα σύστημα από μονάδες λογισμικού, το οποίο εκτίθεται στο χρήστη μέσω μίας ιστοσελίδας. Ο βασικός μηχανισμός που τη χαρακτηρίζει είναι η συλλογή δεδομένων από τη Δ.Π.Ε. που αναφέραμε προηγουμένως και η κατηγοριοποίηση των προϊόντων ως προσφορές ή μη, με βάση τα αποτελέσματα εφαρμογής αλγορίθμων και στατιστικών τεχνικών εύρεσης ακρότατων τιμών σε στατιστικά δείγματα. Δείγμα θεωρούμε το πλήθος των διαθέσιμων τιμών στις οποίες διατίθεται ένα συγκεκριμένο προϊόν, ενώ το ερευνητικό αντικείμενο μας είναι η απόδοση των συγκρινόμενων τεχνικών στην ορθή και αναμενόμενη διεξαγωγή συμπερασμάτων.

Η αρχική ιστοσελίδα περιέχει ένα τμήμα με το λογότυπο της εφαρμογής, μία επιφάνεια εμφάνισης των υψηλότερων ποσοστιαία προσφορών της τελευταίας εβδομάδας και μία μπάρα αναζήτησης, με χρήση της οποίας, ο χρήστης μπορεί να μάθει εάν ένα προϊόν υπάρχει σε προσφορά ή όχι, να δει τις διακυμάνσεις στην τιμή του και να πλοηγηθεί στο επιθυμητό κατάστημα για την ολοκλήρωση της αγοράς του. Παράλληλα, αναπτύχθηκε και η διεπαφή ελέγχου, στην οποία ο διαχειριστής έχει τη δυνατότητα να πραγματοποιήσει μαζική ενημέρωση των προϊόντων και την κατηγοριοποίησή τους σε προσφορές ή μη με αυτοματοποιημένο τρόπο.

Στα πλαίσια της ανάπτυξης μίας αποτελεσματικής, εύρωστης και αποκρίσιμης εφαρμογής, προγραμματίσαμε ένα σύνολο από μονάδες απαραίτητες για τη σωστή λειτουργία της. Τα βασικά στοιχεία είναι τέσσερα, ο Συλλέκτης Δεδομένων, ο Εξυπηρετητής Εφαρμογών, το Λογισμικό Διαδικτυακής Εφαρμογής σε γλώσσα Java και η Βάση Δεδομένων. Επιλέξαμε τεχνολογίες ισχυρές και ώριμες έτσι ώστε να δημιουργήσουμε μία εφαρμογή ασφαλή, γρήγορη και επεκτάσιμη εκτός πλαισίου Πολυτεχνειακού διπλώματος.

2.3 Λειτουργικές Απαιτήσεις

Οι λειτουργικές απαιτήσεις της εφαρμογής είναι οι εξής:

- Λειτουργία αναζήτησης προϊόντος.
- Λειτουργία εμφάνισης βέλτιστων οικονομικά προσφορών για μία χρονική περίοδο.
- Λειτουργία αναζήτησης κάθε πληροφοριακής μονάδας που παρέχεται από τον ιστότοπο Skroutz.gr.
- Λειτουργία μαζικής αποθήκευσης, επεξεργασίας και αποτίμησης δεδομένων.

- Απόφαση εάν ένα προϊόν βρίσκεται σε προσφορά με εφαρμογή στατιστικών τεχνικών εύρεσης έκτοπων τιμών και σύγκρισή τους ως προς τα αποτελέσματα.

- Δυνατότητα μετάβασης στο επιθυμητό ηλεκτρονικό κατάστημα για ολοκλήρωση της αγοράς.

2.4 Προδιαγραφές–Σενάρια Χρήσης

Οι προδιαγραφές για τη συγκεκριμένη εφαρμογή ορίστηκαν σε συννενόηση με το Διδάσκοντα, θα περιγραφούν παρακάτω σε φυσική γλώσσα και θα γίνει χρήση διαγραμμάτων Ακολουθίας και Σεναρίου Χρήσης για οπτική απεικόνιση της λειτουργίας όπου αυτό είναι απαραίτητο. Ουσιαστικά αναφέρονται τα απαιτούμενα χαρακτηριστικά των βασικών δομικών μονάδων της εφαρμογής, σε συνδυασμό με τις λύσεις που δόθηκαν με σειρά λειτουργικής προτεραιότητας.

2.4.1 Συλλέκτης Δεδομένων

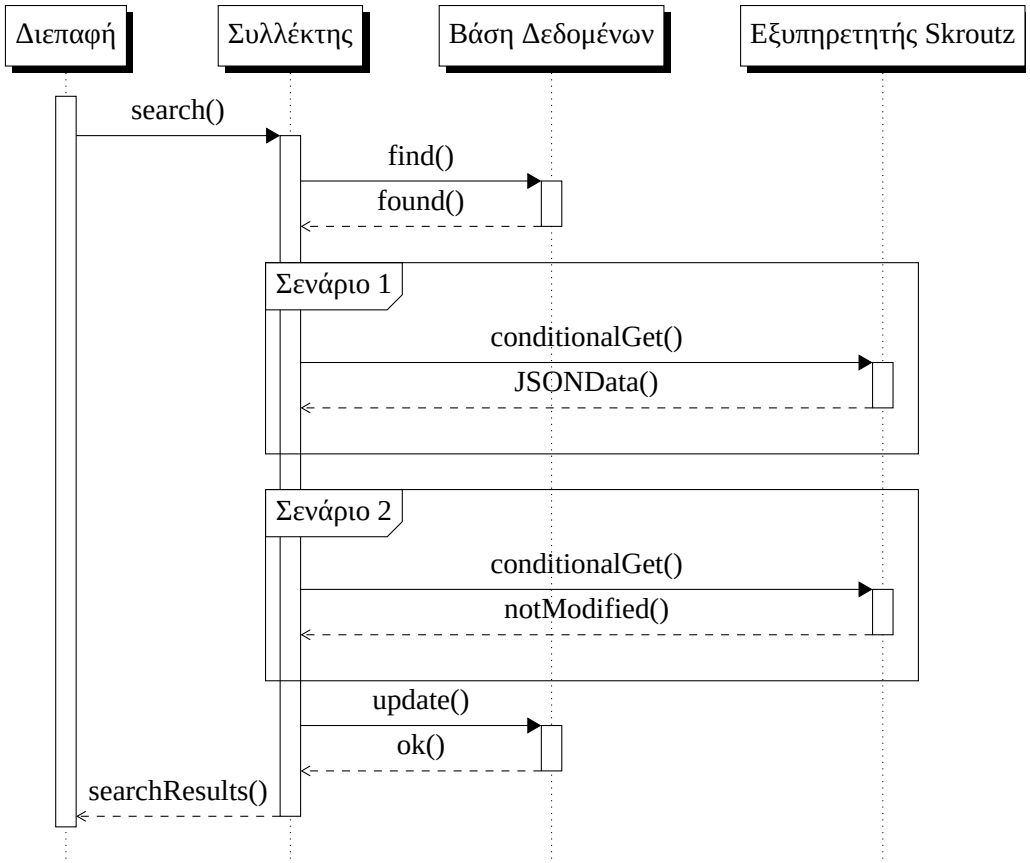
Η πρώτη σε σειρά λειτουργία που κληθήκαμε να υλοποιήσουμε είναι η συλλογή δεδομένων από τη Διεπαφή Προγραμματισμού Εφαρμογών του Skroutz.gr. Λαμβάνοντας υπόψη το πλήθος των δεδομένων που πρέπει να χειριστούμε, τη διαθεσιμότητα και τη μορφή στην οποία τα λαμβάνουμε, επιλέξαμε μία τεχνοτροπία Εξυπηρετητή–Πελάτη για το σχεδιασμό του λογισμικού Συλλογής Δεδομένων. Κάθε φορά που ο χρήστης ζητάει οποιαδήποτε εμπορική πληροφορία, η μονάδα αυτή αναλαμβάνει να επικοινωνήσει με τον Εξυπηρετητή του Skroutz.gr και να την επιστρέψει επιτυχώς, διαφορετικά επιστρέφει μήνυμα λάθους. Ο συλλέκτης, δεδομένης της επαναληπτικότητας ορισμένων δεδομένων, θα έχει τη δυνατότητα να χρησιμοποιεί τη βάση δεδομένων και ορισμένες δυνατότητες του πρωτόκολλου HTTP (HyperText Transfer Protocol) για να αποφεύγει τη μετάδοση περιττών πληροφοριών.

Ο συλλέκτης αυτός τεχνικά δρα ως καταναλωτής της RESTful διεπαφής, όπως αυτή τεκμηριώνεται αναλυτικά στη σχετική σελίδα [Skro]. Ουσιαστικά προγραμματίζει μία νέα διεπαφή, προσβάσιμη στην εφαρμογή μας, η οποία αναλαμβάνει να λάβει τις οντότητες `category`, `shop`, `sku`, `product` και `manufacturer`, να τις αποσειριοποιήσει και να τις χρησιμοποιήσει για ό,τι κρίνεται απαραίτητο.

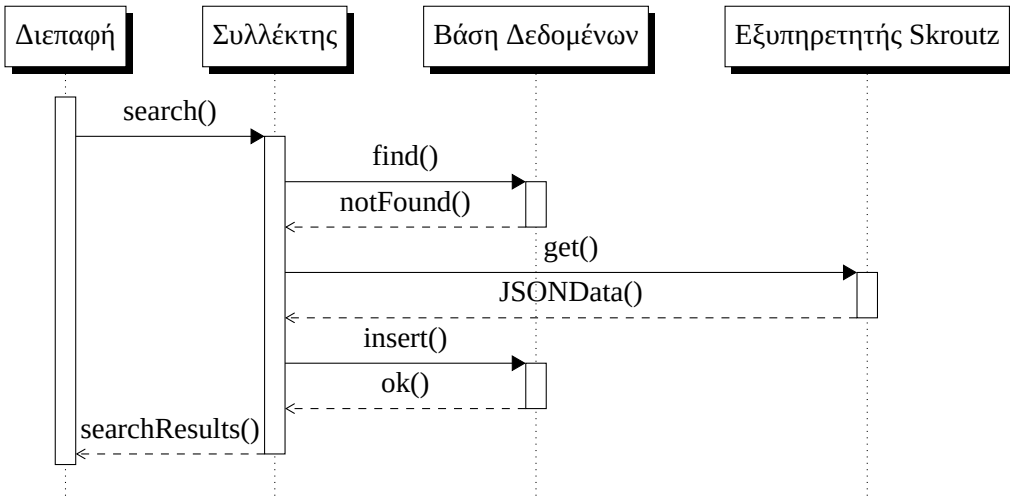
Στο παρακάτω ακολουθιακό διάγραμμα φαίνονται τα διάφορα σενάρια επικοινωνίας του συλλέκτη με τις υπόλοιπες μονάδες, ανάλογα με το είδος της αναζήτησης. Παρατηρούμε ότι η επικοινωνία με τον εξυπηρετητή γίνεται μέσω αιτημάτων HTTP, αξιοποιώντας την σημαία “Conditional” όπου αυτό είναι δυνατό. Η συγκεκριμένη σημαία, χρησιμοποιείται όταν μία πληροφορία έχει ανακτηθεί στο παρελθόν και υπάρχει στη βάση δεδομένων, συνδυάζοντας τα μετα-δεδομένα του με την ημερομηνία ανάκτησης. Έτσι, η εφαρμογή μας επικοινωνεί με τον εξυπηρετητή ενημερώνοντάς τον ότι έχει ήδη λάβει την πληροφορία, και είτε επιβεβαιώνεται ότι έχει την τελευταία, είτε λαμβάνει την ανανεωμένη έκδοση. Το επιτυχημένο σενάριο αναπαρίσταται στο σχήμα 2.4 και το αποτυχημένο στο σχήμα 2.5. Η διαφοροποίηση βρίσκεται στην ύπαρξη ή μη ενός πόρου στη Βάση Δεδομένων. Στην πρώτη περίπτωση, φαίνονται δύο σενάρια, όπου εκτελείται αναζήτηση υπό όρους (conditional). Η λογική είναι ότι αν γίνει επιτυχημένη ανάκτηση από τη Βάση Δεδομένων, τότε εκμεταλλευόμαστε το πεδίο Etag το οποίο χαρακτηρίζει σε ένα στιγμιότυπο του πόρου μία δεδομένη χρονική στιγμή. Με βάση το πεδίο αυτό εκτελούμε την αναζήτηση στη Διεπαφή του Skroutz, και είτε λαμβάνουμε την ανανεωμένη έκδοσή του είτε επιβεβαιώνεται η επικαιροποίησή του με τον κατάλληλο κωδικό απάντησης από τον εξυπηρετητή. Διαφορετικά, γίνεται κλασσική ανάκτηση από τον εξυπηρετητή, και ενημέρωση της βάσης χωρίς κάποια περαιτέρω λογική.

2.4.2 Βάση δεδομένων

Η βάση δεδομένων σχεδιάστηκε με γνώμονα τη βέλτιστη διαθεσιμότητα των δεδομένων και την ελαχιστοποίηση της πολυπλοκότητας των αναζητήσεων. Γνωρίζουμε ότι η κρίσιμη πληροφορία αφορά στα προϊόντα και τις τιμές που τα χαρακτηρίζουν, συνεπώς επιλέγουμε ένα σχήμα που να επιτρέπει την άμεση πρόσβαση σε αυτά. Ταυτόχρονα, καλούμαστε να εμφανίζουμε τις διαθέσιμες



Σχήμα 2.4: Ακολουθιακό διάγραμμα Συλλέκτη Δεδομένων – Περίπτωση επιτυχούς αναζήτησης



Σχήμα 2.5: Ακολουθιακό διάγραμμα Συλλέκτη Δεδομένων – Περίπτωση απουσίας αποτελεσμάτων

προσφορές στην αρχική σελίδα της εφαρμογής, άρα ένας πίνακας προσφορών είναι επίσης απαραίτητος. Μία ιδιομορφία της υλοποίησής μας είναι η χρήση δύο κύριων κλειδιών για κάθε εγγραφή που λαμβάνουμε από τρίτο ιστότοπο. Ο λόγος που επιλέξαμε τη συγκεκριμένη τεχνοτροπία είναι διττός. Αρχικά, τα αιτήματα που στέλνουμε στον εξυπηρετητή του Skroutz γίνονται με βάση τα δεδομένα κλειδιά τους, άρα και απαραίτητη η αποθήκευσή τους. Κατά δεύτερον, αποτελούν εξωτερικό παράγοντα του συστήματός μας, στον οποίο δεν έχουμε έλεγχο σε πιθανές αλλαγές ή διαγραφές, οπότε χρειαζόμαστε την ύπαρξη ενός δεύτερου μοναδικού αναγνωριστικού για την αποφυγή απρόβλεπτων καταστάσεων.

Κατά την προσπάθεια της συχνής επικαιροποίησης δεδομένων, εμφανίζεται το ενδεχόμενο συχνής επικαιροποίησης των διαθέσιμων τιμών ενός προϊόντος. Κάτι τέτοιο επηρεάζει το σχήμα μας, οι επιμέρους τιμές μπορούν να θεωρηθούν σαν εξαρτημένες οντότητες, αλλά η πληθικότητά τους σε σχέση με την ιεραρχική τους εξάρτηση από τα προϊόντα, προστάζει την προσθήκη ενός πίνακα τιμών.

Αντίστοιχα, δημιουργήσαμε τον πίνακα `requests`, ο οποίος καταγράφει όλα τα μοναδικά αιτήματα που έγιναν προς τον εξυπηρετητή μαζί με τις φορές που επαναλήφθηκαν. Ένας τέτοιος πίνακας μπορεί να αποβεί χρήσιμος για την εξαγωγή στατιστικών και συμπερασμάτων ως προς τις πληροφορίες που διαχειρίζεται η εφαρμογή.

Τέλος, επιλέξαμε τη δημιουργία ενός πίνακα προσφορών, ο οποίος καθορίζεται από δύο εξωτερικά κλειδιά, ένα προς τον πίνακα των προϊόντων και ένα προς των τιμών. Ο λόγος που λάβαμε τη συγκεκριμένη αρχιτεκτονική επιλογή είναι για λόγους ταχύτητας εκτέλεσης ορισμένων επαναλαμβανόμενων υπολογισμών. Η έννοια “προσφορά” είναι κύριας σημασίας στην παρούσα εργασία και η ανάκτησή της θα είναι μία από τις λειτουργίες που θα επαναλαμβάνονται με τη μεγαλύτερη συχνότητα κατά τη χρήση της. Έτσι, με την διάθεση ενός ξεχωριστού πίνακα για τις συγκεκριμένες οντότητες, επιτυγχάνουμε ταχύτατα και χαμηλής πολυπλοκότητας ερωτήματα στη βάση μας.

Με βάση τα παραπάνω στοιχεία, επιλέξαμε την προσθήκη ευρετηρίων (indexes) στα κρίσιμα πεδία των πινάκων, ανάλογα πάντα με τα κριτήρια αναζήτησης που εφαρμόζονται συχνότερα. Για παράδειγμα, σε όλους τους πίνακες που υπάρχει πεδίο `skroutz_id`, το πεδίο αυτό είναι και κλειδί ευρετηρίου. Με τον τρόπο αυτό αιτιολογείται και αποδίδει η χρήση διπλών αναγνωριστικών στους αντίστοιχους πίνακες.

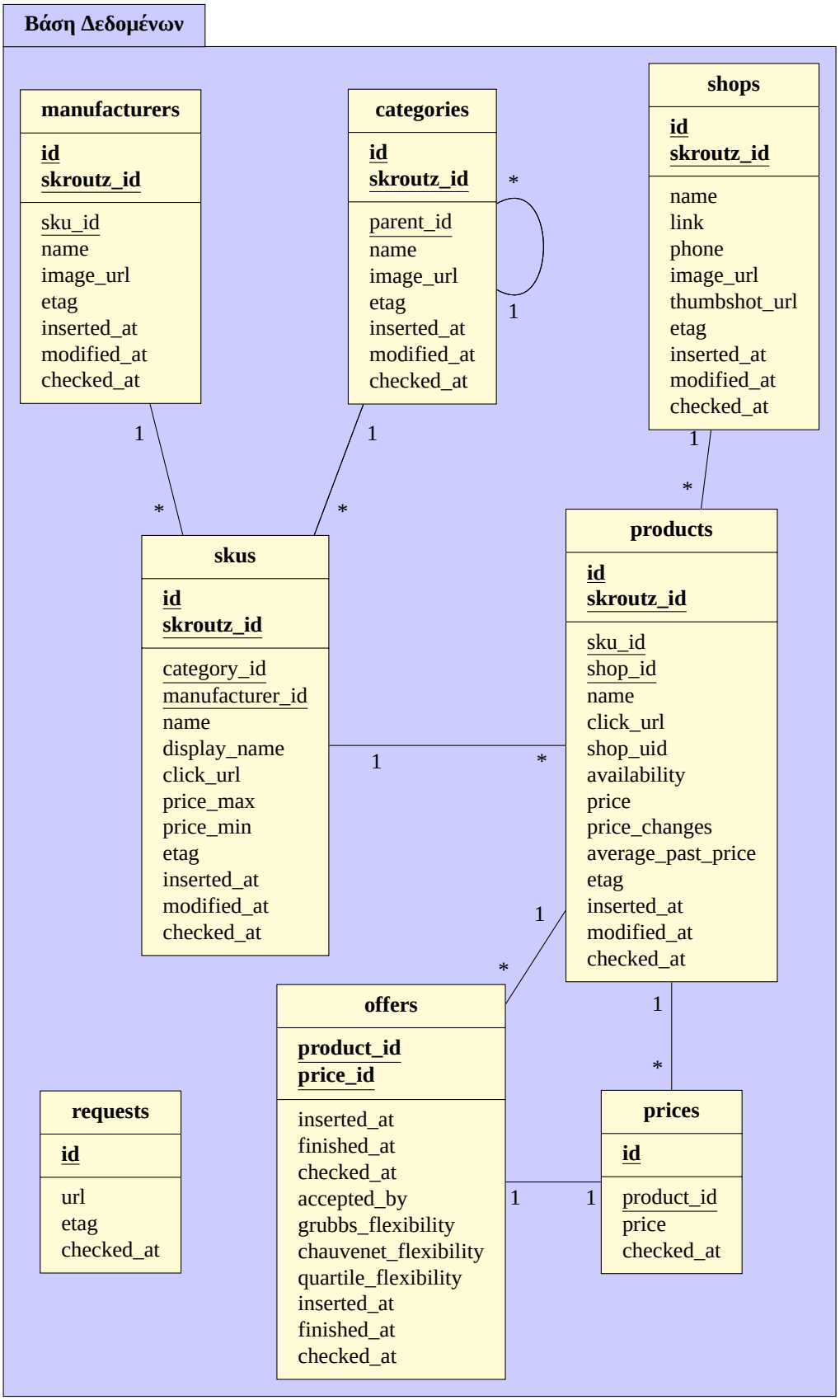
Το σχήμα της βάσης δεδομένων φαίνεται στο σχεσιακό διάγραμμα 2.6. Αξίζει να αναφέρουμε ότι έγιναν αρκετές αναθεωρήσεις του παραπάνω σχήματος έτσι ώστε να ανταποκρίνεται πλήρως στις παρούσες ανάγκες της διπλωματικής και να μπορεί να υποστηρίξει πιθανές επεκτάσεις. Χαρακτηριστικό παράδειγμα είναι οι πίνακες `requests` και `prices` οι οποίοι αυτή τη στιγμή δεν προσφέρουν κάτι ουσιαστικό στην παρούσα μορφή της διπλωματικής, αλλά μπορούν μελλοντικά να αποβούν ιδιαίτερα χρήσιμοι στην εξαγωγή στατιστικών και στην εξόρυξη κρίσιμων δεδομένων για την ανάλυση της απόδοσης της εφαρμογής.

2.4.3 Διεπαφή χρήστη

Όσον αφορά τη διεπαφή χρήστη, δηλαδή την εμφάνιση και τη μορφοποίηση της ιστοσελίδας που δημιουργήσαμε, επιλέξαμε μία μινιμαλιστική λογική. Σε κάθε σελίδα φαίνονται μόνο οι απολύτως απαραίτητες πληροφορίες για να μήν αποσπάται η προσοχή του χρήστη και διότι μία εμπλουτισμένη διεπαφή δεν ήταν άμεσος στόχος αυτής της διπλωματικής εργασίας.

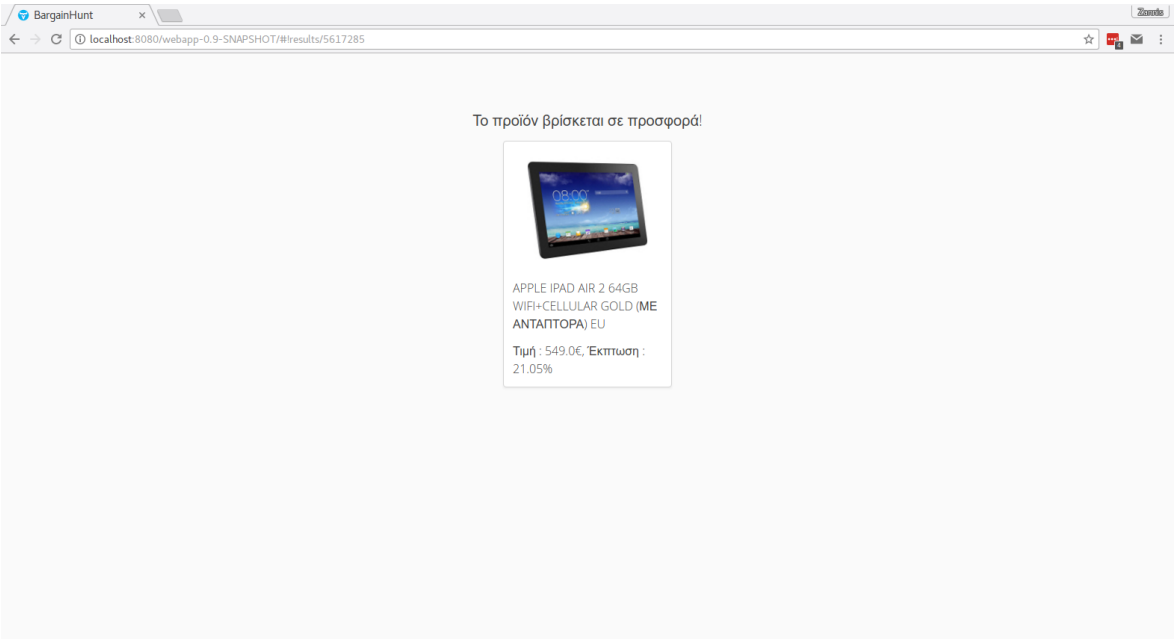
Κατά την είσοδο, υπάρχουν δύο επιτρεπτές ενέργειες. Η πρώτη είναι η αναζήτηση ενός προϊόντος μέσω της μπάρας αναζήτησης και η δεύτερη, η επιλογή μίας από τις υπάρχουσες προσφορές με σκοπό την επίδειξη περισσότερων πληροφοριών σχετικά με αυτή. Στην πρώτη περίπτωση, η αναζήτηση ακολουθεί πλήρως την πορεία του Skroutz, υλοποιώντας μία ασαφή λογική, ικανή να ταιριάξει τη συμβολοσειρά αναζήτησης με οποιαδήποτε εκ των βασικών οντοτήτων της εφαρμογής. Ιδανικά, θα γίνει αναζήτηση για κάποιο συγκεκριμένο προϊόν, όμως αυτό δεν είναι πάντα δυνατό είτε λόγω τυπογραφικού λάθους, λόγω διαφορετικής επιλογής του χρήστη ή και απουσίας αποτελεσμάτων. Σε αυτή την περίπτωση, η εφαρμογή θα εντοπίσει το πρότυπο και θα το εμφανίσει στο χρήστη εφόσον αυτό είναι δυνατό. Παράλληλα, θα εμφανίσει τις κατηγορίες προϊόντων που ταιριάζουν όσο το δυνατόν περισσότερο στο δεδομένο πρότυπο. Η παραπάνω διαδικασία επαναλαμβάνεται μέχρι ο χρήστης να πατήσει στο πλαίσιο που αντιστοιχεί σε κάποιο προϊόν. Όταν συμβεί αυτό, θα ανοίξει η σελίδα αποτελεσμάτων, όπου θα είναι φανερό εάν το επιλεγμένο προϊόν βρίσκεται σε προσφορά, οι διάφορες τιμές και τα καταστήματα στα οποία είναι διαθέσιμο, σύνδεσμοι για να μεταβεί ο χρήστης σε αυτά καθώς και ένα ιστορικό τιμών, ανάλογα πάντα με τη διαθεσιμότητα και το πλήθος των αναζητήσεων που έχουν γίνει για αυτό στα πλαίσια της δοκιμαστικής λειτουργίας της εφαρμογής μας. Στην εικόνα 2.7 φαίνεται η διεπαφή προσφορών που αναφέραμε.

Στη δεύτερη περίπτωση, ο χρήστης θα επιλέξει μία από τις βέλτιστες διαθέσιμες προσφορές. Τότε, ο φυλλομετρητής θα μεταβεί κατευθείαν στην αντίστοιχη σελίδα του προϊόντος, όπου θα φαίνονται



Σχήμα 2.6: Σχεσιακό διάγραμμα Βάσης Δεδομένων

όλα τα χαρακτηριστικά που αναφέραμε στην σελίδα αποτελεσμάτων προηγουμένως. Ουσιαστικά η



Σχήμα 2.7: Παράδειγμα προϊόντος σε προσφορά

επιλογή αυτή έγινε για τον εμπλουτισμό της εμπειρίας χρήσης της εφαρμογής μας, σε μία προσπάθεια να ελαχιστοποιήσουμε τις αναποτελεσματικές αναζητήσεις που δεν καταλήγουν σε κάποια προσφορά. Αυτό είναι και το μεγαλύτερο πρόβλημα που αντιμετωπίζει η εφαρμογή μας, η αναποτελεσματική χρήση της δηλαδή. Ιδανικά σε κάθε αναζήτηση, θα πρέπει αν καταλήξουμε ότι το υπό εξέταση προϊόν δεν διατίθεται σε χαμηλή τιμή, να γίνονται προτάσεις για συγγενικά προϊόντα για τα οποία έχουμε εξακριβώσει από πρίν ότι είναι σε προσφορά.

Ένας τρόπος να αντιμετωπιστεί το παραπάνω πρόβλημα είναι η χρήση της αρχικής σελίδας για προώθηση των καλύτερων προσφορών της ημέρας. Έτσι, ο χρήστης δεν χρειάζεται να μπει στη διαδικασία της αναζήτησης, αλλά έχει πρόσβαση σε χρήσιμες πληροφορίες με την ελάχιστη δυνατή προσπάθεια, γεγονός που χαρακτηρίζει κάθε επιτυχημένη εφαρμογή.

2.5 Παραδοτέα

Στα πλαίσια αυτής της εφαρμογής, θα παραδόσουμε ορισμένα ψηφιακά αρχεία με τα οποία μπορεί να δημιουργηθεί από του μηδενός το απαραίτητο περιβάλλον για την εγκατάσταση και εκτέλεσή της. Τα παραδοτέα θα είναι τα εξής:

- Ο πηγαίος κώδικας της εφαρμογής. Πρόκειται για ένα σύνολο κλάσσεων οι οποίες υλοποιούν όλες τις απαραίτητες οντότητες και τις σχέσεις μεταξύ τους. Ο κώδικας αυτός, καθώς είναι γραμμένος σε γλώσσα Java, έχει ως προαπαιτούμενο την ύπαρξη περιβάλλοντος Java (Java Runtime) διαμορφωμένου κατάλληλα. Για τη μεταγλώττισή του σε εκτελέσιμο κώδικα, είναι απαραίτητη η ύπαρξη του Java Development Kit, το οποίο διατίθεται δωρεάν από τον ιστότοπο Java.com.

- Το αρχείο διαμόρφωσης του Εξυπηρετητή Εφαρμογών Java Wildfly. Το συγκεκριμένο αρχείο περιέχει ρυθμίσης απαραίτητες για τη σωστή λειτουργία της εφαρμογής και της βάσης δεδομένων.

- Το αρχείο δημιουργίας της βάσης δεδομένων, το οποίο περιέχει όλες τις αναγκαίες εντολές για τη δημιουργία και αρχικοποίηση μίας βάσης δεδομένων όπου θα αποθηκεύουμε τα δεδομένα μας.

- Τα αρχεία τεκμηρίωσης των κλάσσεων της εφαρμογής, στην ενδεδειγμένη μορφή Javadoc. Εδώ περιέχονται διάφορες τεχνικές πληροφορίες για τη λειτουργία των μεθόδων που υλοποιήσαμε. Το συγκεκριμένο αρχείο είναι και αναπόσπαστο κομμάτι κάθε εφαρμογής, στα πλαίσια των ορθών τεχνικών ανάπτυξης λογισμικού.

Με τα παραπάνω και φυσικά το παρόν έγγραφο, μπορεί ο οποιοσδήποτε που διαθέτει ένα σύγχρονο υπολογιστή με σύνδεση στο διαδίκτυο να εγκαταστήσει, να θέσει σε λειτουργία αλλά και να δημιουργήσει ενημερώσεις για την εφαρμογή. Η μόνη προϋπόθεση είναι η απόκτηση στοιχείων ταυτοποίησης OAuth2 στον ιστότοπο Skroutz.

Κεφάλαιο 3

Τεχνολογίες

Στο παρόν κεφάλαιο θα αναλύσουμε τις τεχνολογίες που επιλέξαμε να χρησιμοποιήσουμε στα πλαίσια του σχεδιασμού και ανάπτυξης της παρούσας εφαρμογής, τους λόγους που υπαγόρευσαν τις επιλογές μας και τα βασικότερα χαρακτηριστικά της κάθε μίας από αυτές. Λεπτομέρειες σχετικά με την ίδια την υλοποίηση θα δοθούν αναλυτικά σε επόμενο κεφάλαιο.

Με τον γενικό όρο "τεχνολογίες" αναφερόμαστε σε λογισμικά και εργαλεία όπως η γλώσσα προγραμματισμού της επιλογής μας, βιβλιοθήκες, προγραμματιστικά εργαλεία, πρακτικές και πλαίσια ανάπτυξης δικτυακών εφαρμογών.

Η λεπτομερής ανάλυση που ακολουθεί, θα οργανωθεί με βάση τη λογική σειρά στην οποία λήφθηκαν οι αποφάσεις για τη χρήση των εκάστοτε τεχνολογιών. Ξεκινώντας από το στάδιο του σχεδιασμού των απαιτήσεων, συνεχίζουμε με την αρχιτεκτονική και καταλήγουμε στην ανάπτυξη των δομικών επιπέδων της εφαρμογής (Βάση Δεδομένων, Μοντελοποίηση, Λειτουργικότητα, Εμφάνιση). Συγκεκριμένα, η γλώσσα προγραμματισμού που θα χρησιμοποιήσουμε αποτέλεσε την πρώτη σημαντική απόφαση που λάβαμε, για τον απλούστατο λόγο ότι με αυτή θα καθόριζε και μεγάλο πλήθος από τις ακόλουθες, όπως η επιλογή του εξυπηρετητή και του πλαισίου προγραμματισμού. Στη συνέχεια, χρειαστήκαμε ένα εργαλείο διαχείρισης και ελέγχου εκδόσεων κώδικα για την οργάνωση και κατ' επέκταση, τη διάθεση της εφαρμογής στον ιστότοπο "κοινωνικού προγραμματισμού" GitHub. Έπειτα, έγινε ο σχεδιασμός των μονάδων με βάση τις σωστές πρακτικές ανάπτυξης εφαρμογών και με τον αναλογισμό των πλέον κατάλληλων τεχνοτροπιών προγραμματισμού, σύμφωνα πάντα με την υπηρεσιοκεντρική αρχιτεκτονική. Τέλος, οι αποφάσεις μας ολοκληρώθηκαν με την επιλογή των εφαρμογών ανοικτού λογισμικού που θα χρησιμοποιήσουμε σαν εξυπηρετητή εφαρμογών, της διαχειριστικής εφαρμογής της βάσης δεδομένων και του πλαισίου προγραμματισμού. Αξίζει να αναφέρουμε ότι χωρίς ομάδες ανάπτυξης που διαθέτουν λογισμικό με άδειες ανοιχτού κώδικα, το παρόν σύγγραμμα δε θα ήταν δυνατό να ολοκληρωθεί.

3.1 Προσχεδιαστικό στάδιο

Στη συνέχεια γίνεται μία γρήγορη αναφορά στις γλώσσες που χρησιμοποιήσαμε έμμεσα ή άμεσα κατά την ανάληψη αυτής της διπλωματικής εργασίας καθώς επίσης και στα εργαλεία διαχείρισης κώδικα που χρησιμοποιήσαμε για αρχειοθέτηση και αυτόματη διάθεση της εφαρμογής.

3.1.1 Γλώσσες Προγραμματισμού

Java

Παραπάνω αναφέραμε ότι η πρώτη ουσιαστική επιλογή που κάναμε ήταν αυτή της γλώσσας προγραμματισμού. Από τις γλώσσες που ενδείκνυνται για την ανάπτυξη δικτυακών εφαρμογών (Java, JavaScript, PHP, Python, Ruby, C#, Go) με κριτήριο την ήδη υπάρχουσα υποστήριξη και τις γνώσεις μας, επιλέξαμε τη Java, καθώς αποτελεί μία ώριμη και πλήρη αντικειμενοστρεφή γλώσσα, με διάχυτη υποστήριξη και ευρεία χρήση στο διαδίκτυο, η οποία επιτρέπει την αφαίρεση επιπέδων πολυπλοκότητας όπως η συγγραφή αρχείων διαμόρφωσης εμφάνισης, η διαμόρφωση της βάσης δεδομένων και ο εμπλουτισμός των ιστοσελίδων με σεναριακό κώδικα για την καλύτερη εμπειρία χρήσης.

Βασικά της χαρακτηριστικά είναι η Αυτόματη Διαχείριση Μνήμης (Garbage Collection) και η εκτέλεση των εφαρμογών σε περιβάλλον απομόνωσης από το λειτουργικό σύστημα που τις φιλοξενεί. Έτσι διασφαλίζεται η ασφάλεια του χρήστη και η αποφυγή επηρεασμού του συστήματος σε περίπτωση λανθασμένης ενέργειας. Ταυτόχρονα, η γλώσσα Java μπορεί να λειτουργήσει σε επεξεργαστές διάφορων αρχιτεκτονικών σε σταθερές και κινητές συσκευές. Χαρακτηριστικό παράδειγμα είναι το λειτουργικό σύστημα Android [1].

Συγκεκριμένα, επιλέξαμε την εμπλουτισμένη έκδοση Java Enterprise Edition, η οποία υποστηρίζει ειδικές κλάσσεις με όνομα "Enterprise Java Beans", Αντικειμενο-Σχεσιακή Αντιστοίχηση (Object Relational Mapping), Εισαγωγή Εξαρτήσεων Πλαισίου (Context Dependency Injections) και πολλές άλλες προχωρημένες λειτουργίες που αυτοματοποιούν επαναλαμβανόμενες διαδικασίες και τον προγρ αμματιστή στη ανάπτυξη και συντήρηση, αποκρύπτοντας ορισμένες τεχνικές λεπτομέριες. Παράλληλα, είχαμε πρόσβαση σε επαγγελματικά εργαλεία απο το οικοσύστημα εφαρμογών σε γλώσσα Java, τα οποία διατίθενται προς χρήση δωρεάν ως ελεύθερο λογισμικό. Δύο παραδείγματα είναι ο εξυπηρετητής εφαρμογών `Wildfly` και το πλαίσιο ανάπτυξης εφαρμογών `Vaadin` τα οποία θα αναλυθούν περαιτέρω στη συνέχεια.

SASS

Η γλώσσα SASS (Syntactically Awesome StyleSheets) [Catl06] είναι μία σεναριακή γλώσσα διαμόρφωσης κειμένου και αντικειμένων που ανήκουν στο Αντικειμενοστρεφές Μοντέλο Εγγράφου (Document Object Model) [2], η οποία μεταγλωττίζεται σε κανόνες CSS (Cascading Style Sheets). Σχεδιάστηκε σε πρώτο στάδιο το 2006 από τον Hampton Catlin [Wiki15b] και αναπτύχθηκε από τους Natalie Weizenbaum και Chris Eppstein. Αυτή τη στιγμή η γλώσσα αναπτύσσεται σε δημόσιο αποθετήριο στο GitHub [3]. Χρησιμοποιεί δύο διαφορετικά συντακτικά, το πρώτο που βασίζεται στη διάταξη του κειμένου και θυμίζει αυτό της Python, και το δεύτερο που είναι όμοιο με το αντίστοιχο της CSS, το οποίο και έχει επικρατήσει λόγω της οικειότητας που έχουν αναπτύξει οι προγραμματιστές με αυτό. Ο σκοπός της είναι η δυναμική ανάπτυξη κανόνων CSS και ο εμπλουτισμός τους με μηχανισμούς που εμφανίζονται κατά κύριο λόγο σε αντικειμενοστραφείς γλώσσες προγραμματισμού. Η υλοποίησή της έγινε αρχικά στη γλώσσα Ruby, όμως ακολούθησαν και άλλες σε PHP, C και Java. Η τελευταία εσωκλείεται στο πλαίσιο ανάπτυξης που θα χρησιμοποιήσουμε, το Vaadin.

Τα πλέον αξιοσημείωτα χαρακτηριστικά της SASS είναι η ύπαρξη βρόχων, η ύπαρξη μεταβλητών σε τέσσερις βασικούς τύπους (αριθμοί, συμβολοσειρές, κωδικοί χρωμάτων και λογικές αποτιμήσεις) και η δυνατότητα εμφώλευσης. Παράλληλα, υπάρχουν τα `mixins`, δηλαδή μία απλή υλοποίηση συλλογών από κανόνες που μπορούν να προστεθούν σε ένα ήδη υπάρχον σύνολο με μία γραμμή. Ένα απλό παράδειγμα στην τυπική μορφή γραφής SASS φαίνεται στο απόσπασμα 1, ενώ το ίδιο παράδειγμα σε πιο μοντέρνα μορφή παρουσιάζεται στο απόσπασμα 2.

Δεν θα επεκταθούμε παραπάνω στις δυνατότητες της SASS διότι τη διαχειρίζεται αυτόματα το πλαίσιο που θα χρησιμοποιήσουμε. Παράλληλα όμως, αν είναι απαραίτητο, έχουμε τη δυνατότητα να τροποποιήσουμε ορισμένους κανόνες σύμφωνα με τις ανάγκες μας. Το σημαντικό είναι να κατανοήσουμε ότι μέσω αυτής της γλώσσας επιτρέπεται η γρήγορη και εύκολη θεματική παραμετροποίηση της εφαρμογής.

Javascript

Η Javascript [Flan06] είναι μία σεναριακή γλώσσα υψηλού επιπέδου, με δυναμική αποτίμηση τύπων, που συνδυάζει το προστακτικό συντακτικό με χαρακτηριστικά αντικειμενοστραφούς και συναρτησιακού προγραμματισμού. Σε συνδυασμό με την HTML και τη CSS, αποτελούν το βασικό τρίπτυχο που χαρακτηρίζει την εμφάνιση και λειτουργικότητα του διαδίκτυο όπως το γνωρίζουμε τα τελευταία

[1] https://www.android.com

[2] Περισσότερες πληροφορίες για το Document Object Model στο Παράρτημα Α.

[3] https://github.com/sass/sass

```scss
@mixin table-base {
    th {
        text-align: center;
        font-weight: bold;
    }
    td, th {padding: 2px}
}

#data {
    @include table-base;
}
```

Απόσπασμα 1: Παράδειγμα μορφοποίησης με συντακτικό SCSS

```sass
@mixin table-base
    th
        text-align: center
        font-weight: bold
    td, th
        padding: 2px

#data
    @include table-base
```

Απόσπασμα 2: Παράδειγμα μορφοποίησης με συντακτικό SASS

χρόνια. Παρά την κριτική που δέχεται για ορισμένες ασάφειες που τη χαρακτηρίζουν, έχει γίνει πρότυπο μέσω της ευρύτατης υποστήριξης της από όλους τους φυλλομετρητές. Πνευματικός της πατέρας είναι ο Brendan Eich [Wiki15a], ο οποίος την σχεδίασε και ανέπτυξε το Μάϊο του 1995, όντας υπάλληλος της εταιρείας Netscape (που αργότερα μετονομάστηκε σε ίδρυμα Mozilla) σε μία προσπάθεια να ανταγωνιστεί τις διαδικτυακές πλατφόρμες της Microsoft, που απευθύνονταν σε προγραμματιστές με περιορισμένο τεχνικό παρελθόν. Παρά το όνομά της, δεν παρουσιάζει κάποια ομοιότητα με τη γλώσσα Java και εικάζεται ότι τελικά ονομάστηκε έτσι για εμπορικούς λόγους.

Με την πάροδο των χρόνων και την εκθετική άνθιση του διαδικτύου, όλο και περισσότεροι επαγγελματίες ξεκίνησαν να ασχολούνται με τη γλώσσα αυτή, εμπλουτίζοντάς τη με βιβλιοθήκες και πλαίσια προγραμματισμού, ορίζοντας ορθές πρακτικές και επεκτείνοντας το εύρος των εφαρμογών της με αποτέλεσμα σήμερα να είναι ίσως η πλέον διαδεδομένη γλώσσα λόγω της διαθεσιμότητάς της σε κινητές και σταθερές συσκευές. Εντύπωση επίσης προκαλεί, το οικοσύστημα που έχει δημιουργηθεί γύρω από την εν λόγω γλώσσα με την ελεύθερη διάθεση εργαλείων από μεγάλες εταιρίες όπως η Google, το Facebook, η IBM, τα οποία δημιούργησαν τη βάση πάνω στην οποία αναπτύχθηκε η πανσπερμία εφαρμογών και πλαισίων ανάπτυξης που υπάρχουν αυτή τη στιγμή. Πλαίσια όπως το Angular [4], το React [5] το Backbone.js [6], παράλληλα με τη δημιουργία του εργαλείου Node.js [7], οδήγησαν στο να είναι δυνατή η χρήση της Javascript στη συγγραφή εφαρμογών από την πλευρά του εξυπηρετητή αλλά και σε πληθώρα άλλων σεναρίων. Ομοίως με τη SASS, οι λειτουργικές απαιτήσεις για κώδικα Javascript από την εφαρμογή μας καλύπτονται πλήρως από το πλαίσιο Vaadin και συγκεκριμένα το μεταγλωττιστή Google Web Toolkit [8].

3.1.2 Εργαλεία Ανάπτυξης

Τα κύρια εργαλεία που χρησιμοποιήσαμε πέραν του Ολοκληρωμένου Περιβάλλοντος Ανάπτυξης (IDE), τη σουίτα λογισμικών δηλαδή που χρησιμοποιήσαμε για τη συγγραφή και τη δοκιμή της εφαρμογής, είναι το `git` και το `Maven`. Ακολουθεί μία αναφορά στις βασικές λειτουργίες και των δύο, σε συνδυασμό με παραδείγματα και αποσπάσματα κώδικα από τη χρήση τους.

[4] https://angularjs.org
[5] https://facebook.github.io/react
[6] http://backbonejs.org
[7] https://nodejs.org/
[8] http://www.gwtproject.org/

Git

Το git αποτελεί ένα κατανεμημένο σύστημα ελέγχου εκδόσεων λογισμικού (Version Control System), ιδανικό για την οργάνωση και εποπτεία της εφαρμογής, για συνεργατική ανάπτυξη και αποτίμηση του συγγραφικού ιστορικού. Αναπτύχθηκε αρχικά από τον δημιουργό του πυρήνα Linux, τον Linus Torvalds [Wiki16] και πλέον συντηρείται και ανανεώνεται δημόσια [9] από πλήθος προγραμματιστών, με υπεύθυνο τον Junio Hamano [Wiki15c]. Είναι το πρώτο τέτοιο σύστημα που υιοθέτησε την κατανεμημένη αρχιτεκτονική. Αυτό σημαίνει ότι ο κάθε συνεισφέρων στην εφαρμογή, διατηρεί στον τοπικό του υπολογιστή ένα πλήρες αντίγραφο του συνολικού κώδικα, στο οποίο μπορεί να επιφέρει αλλαγές χωρίς να επηρεάζεται η εργασία κανενός άλλου και όταν το επιθυμεί, πιθανώς αποστέλλει τις αλλαγές του στον κεντρικό κόμβο, ο οποίος κατά παράδοση περιέχει την νεότερη και πληρέστερη έκδοση της εφαρμογής. Διαθέτει πλήθος ενεργειών και δομών για την διευκόλυνση της συνεργασίας μεγάλων ομάδων, χωρίς αυτό να σημαίνει ότι περιορίζονται οι δυνατότητές του σε άλλες χρήσεις. Η κάθε ενέργεια αντιστοιχεί σε μία εντολή τερματικού και για την καλύτερη κατανόηση θα δώσουμε ένα παράδειγμα ροής εργασίας αμέσως παρακάτω. Ο ίδιος ακριβώς τρόπος διαχείρισης εκδόσεων κώδικα χρησιμοποιήθηκε στην εφαρμογή BargainHunt.

Κατά την έναρξη της υλοποίησης μίας νέας ιδέας, θέλουμε αρχικά να δημιουργήσουμε ένα νέο καταθετήριο στο φάκελο που επιθυμούμε. Αυτό γίνεται ως εξής σε περιβάλλον γραμμής εντολών UNIX:

```
cd /path/to/project/root/
git init
```

Εφόσον επεξεργαστούμε ή προσθέσουμε τα αρχεία που ανήκουν στην εφαρμογή μας και είμαστε έτοιμοι να τα αποθηκεύσουμε, καλούμαστε να ενημερώσουμε το git για τις αλλαγές και δημιουργήσουμε ένα νέο στιγμιότυπο του κώδικα με ένα χαρακτηριστικό μήνυμα όπως φαίνεται παρακάτω:

```
git add .
git commit -m "Το μήνυμα που επιθυμούμε"
```

Στην περίπτωση όπου κάνουμε χρήση των κατανεμημένων ιδιοτήτων του git, έχουμε τη δυνατότητα να στείλουμε και να λάβουμε τον κώδικά μας από και προς ένα απομακρυσμένο καταθετήριο. Αρχικά πρέπει να ορίσουμε την τοποθεσία του ως εξής:

```
git remote add <ονομασία> <σύνδεσμος προς το καταθετήριο>
```

ή στην προκειμένη περίπτωση

```
git remote add origin https://github.com/zannis/BargainHunt
```

Πλέον μπορούμε να χρησιμοποιήσουμε το νέο απομακρυσμένο καταθετήριο που εγκαταστήσαμε, με τη χρήση των εντολών:

```
git push -u <branch_name>
git pull <branch_name>
```

Για αναμετάδωση και λήψη της τελευταίας αποθηκευμένης έκδοσης κώδικα αντίστοιχα. Τέλος, η συνιστώμενη ροή εργασίας στα πλαίσια του git προτάσσει ο κάθε χρήστης που εργάζεται σε ξεχωριστή λογική μονάδα, να δημιουργεί μία νέα διακλάδωση και να εργάζεται αποκλειστικά σε αυτή, δημιουργώντας έτσι ένα πλαίσιο μη γραμμικής ενημέρωσης ιστορικού. Η δημιουργία μίας νέας διακλάδωσης γίνεται με την εντολή:

```
    git branch <όνομα διακλάδωσης>
```

Ενώ η εναλλαγή μεταξύ διακλαδώσεων με την εντολή:

[9] https://git-scm.com

```
git checkout -b <όνομα διακλάδωσης>
```

Στην περίπτωσή μας, αξίζει να αναφέρουμε ότι παρά το γεγονός ότι οι ενημερώσεις έγιναν αποκλειστικά από ένα χρήστη, χρησιμοποιήθηκαν διαφορετικές διακλαδώσεις για τον συλλέκτη δεδομένων και για τη δικτυακή εφαρμογή έτσι ώστε τυχόν αλλαγές στη μία μονάδα να μην επηρεάσουν την άλλη. Για παράδειγμα, στη διακλάδωση με όνομα `statistics` αναπτύχθηκαν οι στατιστικοί έλεγχοι ενώ στη διακλάδωση `crawler` υλοποιήθηκε ο μηχανισμός μαζικής λήψης πληροφοριών από τη διεπαφή του Skroutz. Τελικά, οι αλλαγές που έγιναν σε κάθε μία διακλάδωση, ενσωματώθηκαν στον "κορμό" ή `master` ως ορίζει η ολοκληρωμένη παράδοση του προγράμματος.

Επιπλέον, το `git` περιέχει τρόπους ορισμού αναφοράς σε εκδόσεις κώδικα με αυξημένη σημασία προς διευκόλυνση του χρήστη, χρησιμοποιώντας "ετικέτες", ή tags με τον εξής τρόπο:

```
git tag -a <όνομα ετικέτας> <hash κωδικός κάποιας έκδοσης>
```

Ταυτόχρονα, μπορεί κάποιος να λάβει όλες τις ετικέτες που υπάρχουν σε ένα αποθετήριο κώδικα `git` και να στείλει τις δικές του με τις παρακάτω εντολές αντίστοιχα:

```
# λίστα με όλες τις διαθεσιμες ετικέτες
git tag
# για να στείλει την έκδοση με ετικέτα 1.1 στον κόμβο origin
git push origin v1.1
```

Εδώ παρουσιάστηκε ένα μικρό μέρος των δυνατοτήτων του `git`, με έμφαση στα εργαλεία που χρησιμοποιήσαμε για τη διαχείριση της συγκεκριμένης εφαρμογής. Το πλήρες μέγεθος των χαρακτηριστικών του είναι εντυπωσιακό αλλά δε θα μας απασχολήσει παραπάνω.

Maven

Το `Maven` είναι ένα υψηλού επιπέδου εργαλείου που χρησιμεύει σε πλήθος τομέων σχετικά με την ανάπτυξη λογισμικού σε γλώσσες που εξυπηρετούνται από το Java Virtual Machine. Αναπτύσσεται από τον οργανισμό ανοιχτού λογισμικού `Apache` (https://www.apache.org/) αρχικά ως εργαλείο ανάπτυξης εφαρμογών Java, ανάλογο του `make` για τη `C` και ως διάδοχος του `Ant` και του `Ivy`. Η βασική χρησιμότητά του είναι η διαχείριση εξαρτήσεων και η αυτοματοποίηση προγραμματιστικών διαδικασιών όπως ο έλεγχος μονάδων (unit testing), η επανεκτέλεση και "συσκευασία" της εφαρμογής σε εκτελέσιμη μορφή. Επιτρέπει στον προγραμματιστή να μην ανησυχεί για το αν οι εξαρτήσεις της εφαρμογής του βρίσκονται στο σωστό φάκελο και τη σωστή μορφή, καθώς αναλαμβάνει να τις λάβει από το κεντρικό του καταθετήριο [href15] και να τις αποθηκεύσει με κατάλληλο τρόπο έτσι ώστε να λειτουργούν πάντα, ανεξαρτήτως αλλαγών στη δομή της εφαρμογής. Περαιτέρω, χάρη στην υπάρχουσα κοινότητα, έχει μπορεί να καλύψει πληθώρα άλλων τεχνικών λειτουργιών που επιτυγχάνονται με τη χρήση εξωτερικών μονάδων (plugins).

Για τη χρήση του, αρκεί η λήψη από τον ιστότοπο [Foun15b] και η αποθήκευσή του στον τοπικό υπολογιστή. Στη συνέχεια, ο χρήστης καλείται να χρησιμοποιήσει την προτεινόμενη δομή στην εφαρμογή του και να ορίσει τις εξαρτήσεις και τις μακροεντολές του στο αρχείο `pom.xml` με τον τρόπο που έχει οριστεί από το σχηματικό πρωτόκολλο του `Maven`. Παρομοίως με το `git`, χρησιμοποιείται και αυτό κυρίως μέσω τερματικού, όμως υπάρχει ευρεία υποστήριξή του από όλα τα Ολοκληρωμένα Περιβάλλοντα Ανάπτυξης. Στη συνέχεια θα δώσουμε μερικά αποσπάσματα από το περιεχόμενο του αρχείου `pom.xml` και τη σημασία του καθενός από αυτά, αφού πρώτα μιλήσουμε για τον τρόπο αρχικοποίησης ενός πρότζεκτ.

Πλέον, τα περισσότερα πλαίσια ανάπτυξης εφαρμογών έχουν υιοθετήσει τη χρήση του `Maven` και διαθέτουν "αρχέτυπα" εφαρμογών στο κεντρικό καταθετήριο που αναφέραμε παραπάνω. Παρέχουν δηλαδή έτοιμες λειτουργικές εφαρμογές με απλές ρυθμίσεις, δίνοντας ένα εφαλτήριο στην παραγωγικότητα του προγραμματιστή. Για να κατεβάσουμε και να εκτελέσουμε σε φυλλομετρητή το "αρχέτυπο" εφαρμογής του Vaadin, πληκτρολογούμε το εξής σε περιβάλλον εντολών UNIX:

```
$ mvn -B archetype:generate
     -DarchetypeGroupId=com.vaadin
     -DarchetypeArtifactId=vaadin-archetype-application
     -DarchetypeVersion=8.0.5 -DgroupId=org.test
     -DartifactId=vaadin-app
     -Dversion=1.0-SNAPSHOT
$ cd vaadin-app
$ mvn package jetty:run
```

Με το παραπάνω, πρώτα γίνεται ανάκτηση των απαραίτητων αρχείων και εξαρτήσεων και στη συνέχεια δημιουργείται μία ολοκληρωμένη δομή αρχείων για μία βασική εφαρμογή. Έπειτα, μεταβαίνουμε στον αντίστοιχο φάκελο και αφού μεταγλωττίσουμε και συσκευάσουμε το προτζεκτ, εκτελούμε μέσω plugin τον εξυπηρετητή εφαρμογών Jetty και εκκινούμε την εφαρμογή.

Έχοντας δει στην πράξη το απλούστερο λειτουργικό παράδειγμα μπορούμε να ξεκινήσουμε τη συγγραφή του κώδικα με την επεξεργασία του αρχείου ρυθμίσεων. Για προσθήκη νέων εξαρτήσεων στο λογισμικό μας προσθέτουμε τις γραμμές που φαίνονται στο απόσπασμα 3, στο σώμα του `pom.xml`:

```
[...]
<dependencies>
  <!-- Postgres JDBC driver -->
  <dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>9.3-1103-jdbc41</version>
    <scope>provided</scope>
  </dependency>
  <!--  Jackson for JSON parsing  -->
  <dependency>
    <groupId>org.glassfish.jersey.media</groupId>
    <artifactId>jersey-media-json-jackson</artifactId>
    <version>2.16</version>
  </dependency>
  [...]
</dependencies>
[...]
```

Απόσπασμα 3: `pom.xml` - Ορισμός εξαρτήσεων

Με χρήση της εκφραστικής γλώσσας XML, ορίζουμε με πολύ ευνόητο τρόπο, τη βιβλιοθήκη ή τη μονάδα που θέλουμε να εντάξουμε στο πρόγραμμά μας. Για την ορθή αναζήτηση και ανάκτηση της εξάρτησης, οφείλουμε να ορίσουμε `groupId`, `artifactId`, `version`, και προαιρετικά `scope`, δηλαδή το αναγνωριστικό της ομάδας στην οποία ανήκει, το όνομά της, την έκδοση και προαιρετικά το πλαίσιο εφαρμογής της. Στη συνέχεια, εφόσον τα στοιχεία που εισάγαμε υπάρχουν σε μορφή αρχείων στο κεντρικό καταθετήριο, αποθηκεύονται σε προσωπικό φάκελο του προγραμματιστή για μελλοντική χρήση.

Παρακάτω βλέπουμε την ένταξη δύο αρθρωμάτων (plugins), δηλαδή συστατικών που προσθέτουν ιδιαίτερες δυνατότητες σε ένα ήδη υπάρχων λογισμικό [Βι15]:

```
[...]
<plugins>
  [...]
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-war-plugin</artifactId>
    <version>2.3</version>
    <configuration>
      <failOnMissingWebXml>false</failOnMissingWebXml>
      <!--
       Exclude some unnecessary files generated by the GWT compiler.
      -->
      <packagingExcludes>
      WEB-INF/classes/VAADIN/gwt-unitCache/**,
      WEB-INF/classes/VAADIN/widgetsets/WEB-INF/**
      </packagingExcludes>
    </configuration>
  </plugin>
  [...]
  <plugin>
    <groupId>org.wildfly.plugins</groupId>
    <artifactId>wildfly-maven-plugin</artifactId>
    <version>\${wildfly.plugin.version}</version>
    <configuration>
      <jboss-home>C:\Java\wildfly-8.2.0.Final</jboss-home>
      <server-config>standalone.xml</server-config>
    </configuration>
  </plugin>
</plugins>
[...]
```

Απόσπασμα 4: `pom.xml` - Plugins

Συγκεκριμένα, το πρώτο άρθρωμα αναλαμβάνει μετά τη μεταγλώττιση της εφαρμογής μέσω της εντολής:

```
mvn package
```

Να ενθυλακώσει τα αρχεία γλώσσας μηχανής στη συμπιεσμένη μορφή `war`, κατάλληλη για την άμεση εκτέλεση της εφαρμογής στον εξυπηρετητή. Εδώ, έχουμε εξαιρέσει ορισμένους φακέλους με στατικά αρχεία διότι δεν είναι απαραίτητα, ενώ παράλληλα δηλώνουμε ότι λείπει το αρχείο `web.xml` και αυτό δε θα πρέπει να σταματήσει τη διαδικασία. Το δεύτερο άρθρωμα, δίνει τη δυνατότητα στο `Maven` να ελέγξει τη λειτουργία του εξυπηρετητή `Wildfly`. Μας επιτρέπει λοιπόν, με μία εντολή, όπως αυτή που ακολουθεί, να εκκινήσουμε τον εξυπηρετητή, να μεταγλωττίσουμε και να συμπιέσουμε την εφαρμογή μας, και να τη θέσουμε για εκτέλεση στο κατάλληλο περιβάλλον.

```
mvn install wildfly:start wildfly:deploy
```

Παραπάνω αποδόθηκαν όλοι οι τρόποι που το Maven μας διευκόλυνε κατά την υλοποίηση της παρούσας λύσης λογισμικού. Περαιτέρω πληροφορίες για τις λειτουργίες που παρέχει το σύστημα Maven μπορούν να βρεθούν στον ιστότοπό του.

3.2 Σχεδιασμός

3.2.1 Εξυπηρετητής

Μία δικτυακή εφαρμογή εξαρτάται άμεσα από τις δυνατότητες του εξυπηρετητή στον οποίο φιλοξενείται. Στη δική μας περίπτωση, λόγω του γεγονότος ότι χρησιμοποιήσαμε τη γλώσσα Java για την υλοποίησή μας, θα πρέπει να επιλέξουμε και έναν συμβατό εξυπηρετητή εφαρμογών. Στο διαδίκτυο υπάρχουν αρκετές επιλογές επαγγελματικού επιπέδου οι οποίες παρέχονται δωρεάν (`Apache Tomcat`, `Jetty`, `Wildfly`, `Glassfish`) και άλλες υπό πληρωμή (`IBM Websphere`, `Oracle Weblogic`). Δεδομένου ότι οι δύο τελευταίες επιλογές εστιάζονται σε εμπορικές εφαρμογές, έπρεπε να επιλέξουμε μία από τις υπόλοιπες. Εν τέλει, ο μόνος εξυπηρετητής εφαρμογών που παρέχει πλήρη κάλυψη στις δυνατότητες που επιθυμούσαμε είναι ο `Wildfly` ο οποίος αναπτύσσεται και διατίθεται δωρεάν από την εταιρία Red Hat. Συγκεκριμένα, υλοποιεί πλήρως τη ΔΠΕ της Java σχετικά με το Αντικειμενικό-Σχεσιακό Συσχετισμό (Object Relational Mapping) μέσω του πλαισίου `Hibernate` αλλά και άλλων ελεύθερων υλοποιήσεων όπως το `EclipseLink`. Επιπλέον διαθέτει μία πανίσχυρη διεπαφή ελέγχου με αναλυτικότατη καταγραφή γεγονότων, στατιστικών, διαχείριση συνδέσεων σε βάσεις δεδομένων και τέλος επιδέχεται πλήρη παραμετροποίηση μέσω της επεξεργασίας του αρχείου `standalone.xml`.

Στη μεγάλη λίστα των χαρακτηριστικών του προστίθεται η εντυπωσιακή ταχύτητα εκκίνησης, χάρη στην παραλληλοποίηση των απαιτούμενων διεργασιών, η οποία είναι εμφανής σε συστήματα με πολλούς επεξεργαστές. Ταυτόχρονα, έχει σχεδιαστεί με κύριο γνώμονα την επεκτασιμότητα επιτρέποντας σε εφαρμογές κάθε μεγέθους να λειτουργήσουν ιδανικά. Είτε μιλάμε για μία εφαρμογή ερευνητικού χαρακτήρα όπως η δική μας με επισκεψιμότητα μερικών δεκάδων, είτε για μία με εκατομμύρια ταυτόχρονες συνδέσεις, το ίδιο λογισμικό εξυπηρέτησης συμπεριφέρεται αξιόπιστα σύμφωνα με ελέγχους τρίτων.

Ένα ακόμα χαρακτηριστικό που μας ώθησε στη συγκεκριμένη επιλογή είναι το χαμηλό αποτύπωμα μνήμης που απαιτείται για τη λειτουργία του. Αφού επιλέξαμε την εκτέλεση του πλήρες συστήματός μας σε ένα μηχάνημα, κρίσιμης σημασίας είναι η αξιοποίηση της μνήμης του. Στον Wildfly, οι ζωτικής σημασίας λειτουργίες έχουν αναπτυχθεί με τρόπο ώστε να καταλαμβάνουν το ελάχιστο δυνατό χώρο στη μνήμη, και αξιοποιούν βέλτιστα τα επίπεδα κρυφής μνήμης (cache) του υπολογιστή. Με τον τρόπο αυτό ελαχιστοποιούνται οι διακοπές για τη συλλογή απορριμάτων και επιτρέπεται η λειτουργία του ακόμα και σε ενσωματωμένα συστήματα όπως το Raspberry Pi. Επιπλέον, ακολουθεί τη λογική της τμηματοποίησης, απομονώνοντας όλα τα ανεξάρτητα τμήματά του, και επιτρέποντας στον προγραμματιστή να τα απενεργοποιήσει, για να δημιουργήσει ένα μινιμαλιστικό σύστημα, σχεδιασμένο αποκλειστικά για την εξυπηρέτηση των αναγκών του.

Ο εξυπηρετητής, στα πλαίσια αυτής της εφαρμογής, λειτουργεί αδιάκοπα σε ένα ψευδομηχάνημα που μας διατίθεται στον `okeanos` [SA], την ελληνική πλατφόρμα Υποδομής ως Υπηρεσία (Infrastructure as a Service).

3.2.2 Βάση δεδομένων

Η τελευταία επιλογή που λάβαμε σχετιζόταν με την πηγή των δεδομένων μας. Είτε θα επιλέγαμε μια υλοποίηση στα πλαίσια του σχεσιακού μοντέλου που εκφράζεται από τη γλώσσα SQL (Structured Query Language) είτε στα πλαίσια του μοντέρνου Μη-Σχεσιακού μοντέλου, όπου η πληροφορία είναι πιο συγκεχυμένη και ασαφής και αποτελείται από συλλογές δεδομένων σε μορφή JSON (JavaScript Object Notation). Οι δύο τακτικές έχουν αμφότερες τα προτερήματα και τα μειονεκτήματα τους. Το σχεσιακό μοντέλο επιτρέπει τον ακριβή ορισμό των δεδομένων και του χαρακτηρισμού τους από πεδία συγκεκριμένης μορφής. Τα δεδομένα βρίσκονται σε πίνακες που αποτελούνται από γραμμές, όπου κάθε γραμμή αντιστοιχεί σε ένα αντικείμενο ή μία σχέση μεταξύ οντοτήτων. Αντίθετα, στο νέο μοντέλο, η αποθήκευση των δεδομένων μοντελοποιείται διαφορετικά, με χρήση εγγράφων, λεξικών, γράφων αλλά και πινάκων. Στο σχεσιακό μοντέλο, υπάρχει προκαθορισμένο σχήμα στο οποίο πρέπει να υπακούν όλες οι εγγραφές, γεγονός που το καθιστά δυσπροσάρμοστο σε συχνές αλλαγές αντίθετα με το μη-σχεσιακό όπου ακολουθείται μία δυναμική λογική σχημάτων, ανάλογα με τις ανάγκες της

εφαρμογής και χωρίς αυστηρούς κανόνες. Διαφορές παρατηρούμε, τέλος, στο θέμα της επεκτασιμότητας, όπου είναι φανερό ότι καθώς συσσωρεύονται τα δεδομένα, στην πρώτη περίπτωση χρειαζόμαστε εξυπηρετητές μεγαλύτερης χωρητικότητας και υπολογιστικής δύναμης κάτι που σε μεγάλα μεγέθη είναι ιδιαίτερα ακριβό, ενώ στη δεύτερη, μπορούμε να μοιράσουμε την πληροφορία σε πολλούς μικρής ισχύος εξυπηρετητές με ίδια ή και καλύτερη απόδοση στην αποκρισιμότητα και το κόστος.

Αφού αναλογιστήκαμε τα παραπάνω, αλλά και τις υπάρχουσες γνώσεις καταλήξαμε σε μία σχεσιακή λύση. Αυτό σημαίνει ότι χρειαστήκαμε ένα Σύστημα Διαχείρισης Σχεσιακών Βάσεων Δεδομένων (RDBMS) μεταξύ των `MySQL`, `SQLite`, `PostgreSQL`, `Oracle Database`, `Microsoft IIS Server`. Τα δύο τελευταία, αν και ισχυρότατα, βρέθηκαν εκτός συναγωνισμού διότι αποτελούν εμπορικές και όχι ερευνητικές επιλογές. Από τα υπόλοιπα, επιλέξαμε το σύστημα `PostgreSQL`, το οποίο αποτελεί μία προσπάθεια για την συγκάλυψη των ικανοτήτων της `Oracle Database` σε ελεύθερο λογισμικό. Είναι, επίσης, γνωστό για την αξιοπιστία του, την υποστήριξη μεγάλου φάσματος τύπων δεδομένων, καθώς επίσης και όλων των βέλτιστων πρακτικών κατά τη μοντελοποίηση και αποθήκευση πληροφοριών. Σε πιο τεχνικό επίπεδο, επιτρέπει δημιουργία ευρετηρίων με βάση λογικές εκφράσεις, χρήση πινάκων, παραμετροποιημένες αθροιστικές συναρτήσεις και τύπους, ελέγχους τιμών σε πεδία, εσωτερική βελτιστοποίηση αναζητήσεων και σύνταξη εκφράσεων με εξαιρέσεις.

3.2.3 Μεταβίβαση δεδομένων

Όσον αφορά στην επικοινωνία των μονάδων του συστήματός μας με το εξωτερικό περιβάλλον, λόγω της διαδικτυακής μορφής της εφαρμογής, χρησιμοποιήσαμε την υπηρεσία επικοινωνίας βασισμένη στο HTTP πρωτόκολλο και τις μεθόδους που υποστηρίζει, συγκεκριμένα GET, POST, PUT, DELETE, HEAD και OPTIONS, γνωστή και ως REST (Representational State Transfer) [Fiel00]. Εμπλουτίζοντας σημασιολογικά αυτές τις μεθόδους με λειτουργίες στα δεδομένα, όπως ανάγνωση με τη GET, προσθήκη με την POST, ενημέρωση με την PUT και διαγραφή με την DELETE, παρατηρούμε ότι όλες οι επιθυμητές ενέργειες ικανοποιούνται με αποκλειστική χρήση ενός πρωτοκόλλου, ενθυλακώνοντας όλη την πληροφορία στο σώμα κάθε σχετικού αιτήματος ή απάντησης. Η πληροφορία αυτή, κωδικοποιείται στη μορφή JSON (Javascript Object Notation) η οποία είναι μία συνοπτική και αποτελεσματική μορφή σειριοποίησης αντικειμένων που συντακτικά μοιάζει με την αποτίμηση δεδομένων στη γλώσσα Javascript.

3.3 Υλοποίηση

Η υλοποίηση περιλαμβάνει όλες τις απαραίτητες ενέργειες για να φτάσουμε από την ιδέα μας σε μία λειτουργική εφαρμογή. Η συγγραφή κώδικα, η κατανόηση και βέλτιστη χρήση πλαισίων προγραμματισμού, η χρήση έτοιμου κώδικα στη μορφή βιβλιοθηκών και τέλος η μεταφορά στατιστικών αλγορίθμων σε κωδικοποιημένη μορφή και προσαρμογή στα δεδομένα μας είναι κατ' εξοχήν θέματα υλοποίησης.

3.3.1 Πλαίσια ανάπτυξης

Στα πλαίσια της ανάπτυξης δικτυακών εφαρμογών, εφόσον μιλάμε για δυναμικό περιεχόμενο και όχι στατικό, θα πρέπει να λάβουμε υπ' όψιν και να χειριστούμε κατάλληλα όλα τα χαρακτηριστικά και ιδιοτροπίες του πρωτοκόλλου HTTP, του εξυπηρετητή και της βάσης δεδομένων. Κάτι τέτοιο, για το πεδίο δράσης αυτής της διπλωματικής εργασίας, θα ήταν αχρείαστα δύσκολο, και θα οδηγούσε σε μία εφαρμογή με πολλαπλά πιθανά σημεία εμφάνισης προβλημάτων. Συνεπώς, ακολουθήσαμε την καλή πρακτική της υιοθέτησης πλαισίων ανάπτυξης όπου κρίναμε ότι αυτό είναι απαραίτητο. Ειδικότερα, μεταξύ των διαθέσιμων ελεύθερων πλαισίων ανάπτυξης δικτυακών εφαρμογών σε γλώσσα Java, `Spring`, `Grails`, `GWT`, `Play`, `Struts` και `Vaadin`, επιλέξαμε το τελευταίο, διότι έχει ισορροπημένη σχέση δυνατοτήτων και πολυπλοκότητας, επιτρέπει την αφαιρετική αντιμετώπιση και την απλούστευση πολλών διαδικασιών αφού αναλαμβάνει τη μεταγλώττιση του δικού μας κώδικα Java

σε Javascript και CSS και κατ' επέκταση τη δημιουργία όμορφων και εύχρηστων διεπαφών με την ελάχιστη δυνατή προσπάθεια. Ένα δεύτερο πλαίσιο που χρησιμοποιήσαμε, είναι το `EclipseLink`, το οποίο έχει περιορισμένο εύρος χρήσης, όμως είναι απαραίτητο για τη σωστή συσχέτιση της βάσης δεδομένων με τα αντικείμενα που χρησιμοποιούμε στην εφαρμογή. Στη συνέχεια θα δόσουμε περισσότερες πληροφορίες και θα ερευνήσουμε θέματα που αφορούν τα δύο αυτά πλαίσια.

Vaadin

Το Vaadin [Ltd15] είναι ένα πλαίσιο ανάπτυξης πλούσιων δικτυακών εφαρμογών Java που αναπτύσσεται από το 2002 από την εταιρία Vaadin Ltd. το οποίο και διατίθεται δωρεάν υπό την άδεια χρήσης Apache License 2.0 [Foun15a]. Χρησιμοποιεί την πλατφόρμα `Google Web Toolkit (GWT)` για τη μεταγλώττιση των μονάδων σε μορφή κατάλληλη για την κατανάλωση από φυλλομετρητές, και ως αποτέλεσμα, η σύνταξή του μοιάζει αρκετά με αυτή του πακέτου `Swing`, το οποίο εξειδικεύεται στη δημιουργία γραφικών διεπαφών σε τοπικό περιβάλλον. Το τελικό αποτέλεσμα μπορεί εύκολα να τροποποιηθεί και να επεκταθεί χρησιμοποιώντας είτε έτοιμα πακέτα θεματικής μορφοποίησης (themes), είτε με ρητό κώδικα SASS ή CSS.

Αρχιτεκτονικά, υποστηρίζει την πρακτική των περιορισμένων σελίδων, οι οποίες λειτουργούν με διαφορετικό τρόπο ανάλογα με το πλαίσιο δεδομένων και την κατάσταση στην οποία βρίσκεται η εφαρμογή. Οι πληροφορίες μεταδίδονται σύγχρονα είτε ασύγχρονα, με τη μορφή αιτημάτων HTTP και ακολουθώντας το REST μοντέλο επικοινωνίας. Έτσι, εκμεταλλεύεται τη γνωστή πληροφορία για να διαχειρίζεται με το βέλτιστο τρόπο τη μνήμη και να επιτυγχάνει πολύ γρήγορη απόκριση στην αλληλεπίδραση με το χρήστη. Χάρη στην ενθυλάκωση του μεταγλωττιστή GWT, αναλαμβάνει να δημιουργήσει όλη τη λογική και θεματοποίηση που απαιτείται από την πλευρά του δέκτη σε HTML5, CSS3 και Javascript μόνο στην έναρξη της επικοινωνίας. Ο προγραμματιστής έτσι απαλλάσσεται από την αντιπαραγωγική συγγραφή κώδικα για υποστήριξη των διάφορων φυλλομετρητών και συσκευών. Αντίθετα, καταλήγει με μία εφαρμογή αποκρίσιμη σε σταθερές και φορητές συσκευές με ελάχιστη παρέμβαση. Για την επίτευξη του τελευταίου παραθέτουμε χαρακτηριστικά τον αντίστοιχο κώδικα.

```
/**
 * Main UI class for BargainHunt. Initializes the {@link Navigator}
 * and the EJB classes we are using inside the project.
 */
@CDIUI("")
@Push
@Theme("bargainhunt")
public class BargainHuntUI extends UI {
  @Override
    protected void init(VaadinRequest vaadinRequest) {
        Responsive.makeResponsive(this);
        setLocale(vaadinRequest.getLocale());
        getPage().setTitle("BargainHunt");
        initNavigator();
    }
}
```

Απόσπασμα 5: Η κλάση που αρχικοποιεί μια εφαρμογή στο Vaadin

Εντύπωση προκαλεί επίσης η δυνατότητα χρήσης λειτουργίας "push" δηλαδή η ασύγχρονη και απρόκλητη μετάδοση πληροφοριών από τον εξυπηρετητή στον καταναλωτή, επίσης υλοποιήσιμη με την ένταξη ενός πρόσθετου συστατικού στο Vaadin, το "vaadin-push", όπως φαίνεται στο απόσπασμα 6. Τη χρησιμοποιήσαμε στην εφαρμογή για να ενημερώνουμε ασύγχρονα τη διεπαφή χρήστη

με πληροφορίες για την κατάσταση της εφαρμογής καθώς αυτή εκτελεί χρονοβόρες διαδικασίες. Για τη χρήση της αρκεί η επισημείωση `@Push` στην κλάσση `BargainHuntUI` και η χρήση της μεθόδου `UI.getCurrent().push()` κάθε φορά που υπάρχει νέα πληροφορία για μετάδοση στο χρήστη.

```
<dependency org="com.vaadin" name="vaadin-push"
            rev="&vaadin.version;" conf="default->default">
    <exclude org="org.slf4j" name="slf4j-api"/>
</dependency>
```

Απόσπασμα 6: `pom.xml` - Vaadin Push dependency

Χάρη στην ενεργή κοινότητα που υποστηρίζει αφιλοκερδώς την πλατφόρμα, δημιουργώντας και ανανεώνοντας πρόσθετες μονάδες (add-ons) ωφελούμαστε από προηγουμένως δυσπρόσβατη λειτουργικότητα. Συγκεκριμένα, εμείς θα χρησιμοποιήσουμε τα πρόσθετα για υποστήριξη των τεχνολογιών CDI, JPA και δυναμικής στοιχειοθέτησης αντικειμένων HTML.

EclipseLink

Το λογισμικό EclipseLink [Mull12] είναι μία ελεύθερη υλοποίηση ορισμένων ΔΠΕ συσχετισμού αντικειμένων και σχέσεων στη γλώσσα Java, συγκεκριμένα των JPA (Java Persistence API), JAXB (Java Architecture for XML Binding), JCA (Java Connector Architecture) και SDO (Service Data Objects). Αναπτύσσεται από το ίδρυμα Eclipse και διαθέτει πλήρη τεκμηρίωση. Οι ΔΠΕ που αναφέραμε, είναι ένα σύνολο μεθόδων και σχολιασμών που επιτρέπουν την άμεση συσχέτιση δομών της βάσης δεδομένων με αντικείμενα Java, τη σειριακή τους αναπαράσταση σε μορφή XML και JSON και διάφορες προδιαγραφές για τη σωστή σύνδεση μεταξύ δομικών μονάδων μιας εφαρμογής. Παράλληλα, το περιβάλλον ανάπτυξης που χρησιμοποιούμε επιτρέπει τη λειτουργία της αυτόματης ανάπτυξης όλου του απαραίτητου κώδικα για την παραμετροποίηση αυτής της λειτουργίας, διαβάζοντας το τρέχον σχήμα της βάσης και παράγοντας προτυποποιημένο κώδικα. Είναι γεγονός ότι η πλέον διαδεδομένη υλοποίηση είναι αυτή της εταιρίας Red Hat, υπό το όνομα Hibernate, η οποία και είναι προεγκατεστημένη στην τελευταία της έκδοση στον εξυπηρετητή Wildfly. Επιλέξαμε όμως τη συγκεκριμένη υλοποίηση λόγω της καθολικής της υποστήριξης ανεξαρτήτως υποδομής, καθώς επιθυμούμε το σύστημα να λειτουργεί σε κάθε δυνατή σύνθεση υλικού. Επιπλέον, η λειτουργικότητα των δύο αυτών υλοποιήσεων είναι ίδια, όμως διαφοροποιείται σε τεχνικές λεπτομέρειες και τον τρόπο σύνταξης.

Τέλος, για την ενεργοποίησή του, αρκεί να προσθέσουμε το εκτελέσιμο αρχείο που διατίθεται από την ιστοσελίδα λήψης [Foun15c], στον κατάλληλο φάκελο εσωτερικά του εξυπηρετητή μαζί με ένα αρχείο `module.xml` που παρατίθεται στη συνέχεια.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Η άδεια χρήσης παραλείπεται -->
<!-- Represents the EclipseLink module  -->
<module xmlns="urn:jboss:module:1.3" name="org.eclipse.persistence">
    <resources>
        <resource-root path="eclipselink.jar"/>
    </resources>

    <dependencies>
      <module name="asm.asm"/>
      <module name="javax.api"/>
      <module name="javax.ws.rs.api"/>
      <module name="javax.annotation.api"/>
      <module name="javax.enterprise.api"/>
      <module name="javax.persistence.api"/>
      <module name="javax.transaction.api"/>
      <module name="javax.validation.api"/>
      <module name="javax.xml.bind.api"/>
      <module name="org.antlr"/>
      <module name="org.apache.commons.collections"/>
      <module name="org.dom4j"/>
      <module name="org.javassist"/>
      <module name="org.jboss.as.jpa.spi"/>
      <module name="org.jboss.logging"/>
      <module name="org.jboss.vfs"/>
    </dependencies>
</module>
```

Απόσπασμα 7: `module.xml` - Παραμετροποίηση EclipseLink

Μετά την ανάρτηση αυτού του αρχείου, αρκεί να συμπεριλάβουμε στον αρχείο `pom.xml` την κατάλληλη εξάρτηση, και να υποδείξουμε ότι αυτή συμπεριλαμβάνεται στα αρχεία του εξυπηρετητή, όπως φαίνεται παρακάτω.

```xml
<dependencies>
    <!--EclipseLink for JPA-->
    <dependency>
        <groupId>org.eclipse.persistence</groupId>
        <artifactId>eclipselink</artifactId>
        <version>2.6.0</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

Απόσπασμα 8: `pom.xml` - Ενσωμάτωση EclipseLink

3.3.2 JUnit

JUnit [10] ονομάζεται το πλέον εδραιωμένο πλαίσιο υλοποίησης Ελέγχου Μονάδων (Unit Tests) για εφαρμογές Java. Παρέχει δηλαδή στον προγραμματιστή ένα σύνολο ειδικών μεθόδων και κλάσσεων, οι οποίες επιτρέπουν τη συγγραφή ελέγχων επεκτείνοντας την ομόνυμη μητρική κλάσση. Οι έλεγχοι αυτοί υποδηλώνουν κατά πόσο οι λειτουργίες της αναπτυσσόμενης εφαρμογής λειτουργούν με τον αναμενόμενο τρόπο. Κατά την ώρα γραφής του παρόντος βρίσκεται στην έκδοση 4.12 από το 2014, το οποίο σε συνδυασμό με το γεγονός ότι μαζί με το `slf4j` είναι οι δύο εξωτερικές εξαρτήσεις με το μεγαλύτερο ποσοστό ενσωμάτωσης σε εφαρμογές Java [11], υποδηλώνει τη σταθερότητά του σαν εργαλείο. Το Junit υπηρετεί τον "ελεγχοστραφή προγραμματισμό" (Test-driven development), ο οποίος αναλύεται περαιτέρω στο Παράρτημα Α. Η φιλοσοφία και ο τρόπος έκφρασης που επιτρέπει, ανήκει στην οικογένεια λογισμικού `xUnit`, όπου `x` αντιστοιχεί σε κάποιο γράμμα που χαρακτηρίζει την αντίστοιχη υλοποίηση σε γλώσσα προγραμματισμού C# [12]. Ένα παράδειγμα κλάσσης ελέγχου φαίνεται στο Απόσπασμα 9. Στο συγκεκριμένο παράδειγμα, ελέγχουμε το ορθό αποτέλεσμα της μεθόδου `evaluate(String)`, η οποία σχεδιάστηκε έτσι ώστε να διαβάζει συμβολοσειρές που εκφράζουν πρόσθεση αριθμών και να εκτελεί την πράξη. Αυτό γίνεται με τη μέθοδο `assertEquals(Object, Object)`, η οποία διατίθεται από το JUnit και εκφράζει την απαίτηση να είναι ίσα τα δύο ορίσματά της. Εαν η συνθήκη ικανοποιείται, τότε και ο έλεγχος είναι επιτυχής.

```
1   import static org.junit.Assert.assertEquals;
2   import org.junit.Test;
3
4   public class CalculatorTest {
5       @Test
6       public void evaluatesExpression() {
7           Calculator calculator = new Calculator();
8           int sum = calculator.evaluate("1+2+3");
9           assertEquals(6, sum);
10      }
11  }
```

Απόσπασμα 9: Παράδειγμα κλάσσης ελέγχου

Συνήθως μετά τη συγγραφή τους, οι έλεγχοι μονάδων εκτελούνται αυτόματα αμέσως μετά τη μεταγγλώτισση της εφαρμογής, και μόνο σε περίπτωση πλήρους επιτυχίας συνεχίζεται η διαδικασία συμπίεσης και εκτέλεσης. Για να γίνει μια παρόμοια διαδικασία με χρήση του Maven θα πληκτρολογούσαμε το εξής σε περιβάλλον εντολών UNIX:

```
1 mvn test package
```

3.3.3 Βιβλιοθήκες και Διεπαφές Προγραμματισμού Εφαρμογών

Στη συνέχεια θα παραθέσουμε συνοπτικά τα ονόματα και τη σημασία των διάφορων βιβλιοθηκών που χρησιμοποιήσαμε στην εφαρμογή μας. Όλες αυτές οι βιβλιοθήκες, ουσιαστικά είναι προμεταγλωττισμένος κώδικας Java, ο οποίος επιτρέπει την ενεργοποίηση δυνατοτήτων που υπό άλλες συνθήκες θα έπρεπε να υλοποιήσουμε μόνοι μας, με όλους τους κινδύνους που συνεπάγεται αυτό. Αντίθετα, βασιζόμαστε στην εργασία επαγγελματικών ομάδων ελεύθερου λογισμικού, οι οποίοι κατέχουν πλήρως το εκάστοτε αντικείμενο και είναι υπεύθυνοι για τη συντήρηση και ανανέωση του.

[10] http://junit.org/
[11] http://blog.takipi.com/we-analyzed-30000-github-projects-here-are-the-top-100-libraries-in-java-js-and-ruby
[12] https://xunit.github.io/

Ξεκινώντας από τη βάση δεδομένων, χρειαζόμαστε έναν συμβατό “σύνδεσμο” προς και από την εφαρμογή μας, ο οποίος είναι ο οδηγός PostgreSQL JDBC (Java DataBase Connector). Αυτός αναλαμβάνει να καθοδηγήσει την εφαρμογή έτσι ώστε να λειτουργήσει με το λογισμικό διαχείρισης PostgreSQL και με την αντικατάστασή του θα μπορούσαμε να αλλάξουμε τη βάση δεδομένων με οποιαδήποτε άλλη. Παράλληλα, ενεργοποιούμε τη διεπαφή JPA έτσι ώστε να προετοιμαστούν οι κατάλληλες μέθοδοι για το συσχετισμό αντικειμένων Java με εγγραφών SQL και την αυτόματη διαχείριση του επιπέδου δεδομένων.

Όσον αφορά στις τεχνολογίες μεταφοράς δεδομένων, θέλουμε να χειριστούμε τη μετάδοση αιτημάτων και απαντήσεων υπό το πρωτόκολλο HTTP, ακολουθώντας το REST μοντέλο, και τη σειριοποίηση των δεδομένων από και προς τη μορφή JSON. Για το πρώτο χρησιμοποιήσαμε την πρότυπη βιβλιοθήκη της Java, Jersey, και για το δεύτερο, την υλοποίηση της Red Hat με το όνομα Jackson 2.

Οι διεπαφές που παρέχονται στα πλαίσια της εκτεταμένης έκδοσης της Java και θέσαμε στην υπηρεσία μας είναι οι CDI, EJB, Annotations API και Servlet API. Για το εύρος της δικής μας εργασίας κρίθηκαν απαραίτητες, διότι ενεργοποιούν κρίσιμες λειτουργίες στα πλαίσια της δικτυακής αρχιτεκτονικής. Τέλος, οι βιβλιοθήκες γενικού σκοπού που χρησιμοποιήσαμε είναι η SLF4J για την επέκταση του συστήματος καταγραφής της λειτουργίας, η Lang3 και Math3 που αποτελούν υποσύνολα της βιβλιοθήκης Apache Commons για καλύτερο χειρισμό συμβολοσειρών και μαθηματικών πράξεων αντίστοιχα.

3.3.4 Στατιστικές τεχνικές

Η παρούσα εργασία θα ήταν αδύνατο να ολοκληρωθεί χωρίς την υλοποίηση και σύγκριση στατιστικών μεθόδων για την εύρεση ακρότατων σε κανονικές κατανομές αριθμών. Στη συγκεκριμένη περίπτωση, ως κατανομές θεωρούμε το σύνολο των τιμών στις οποίες διατίθεται ένα προϊόν, οι οποίες είναι περιορισμένες σε πλήθος και κυμαίνονται από μία έως λίγες εκατοντάδες και επιθυμούμε να απομονώσουμε τα χαμηλά ακρότατα και συγκεκριμένα το ελάχιστο από αυτά. Εάν πληρούνται οι προϋποθέσεις για την εκτίμηση των διαφόρων τεχνικών, τότε απομονώνουμε την ελάχιστη τιμή και την εμφανίζουμε ως προσφορά. Στα πλαίσια αυτής της διπλωματικής, αρκούμαστε στην παρατήρηση της απόδοσης των μεθόδων και τη σύγκρισή τους.

Η πρώτη μέθοδος που εξετάσαμε είναι ο Έλεγχος Ακραίων Τιμών του Grubbs [Grub69]. Ονομάστηκε από το δημιουργό της, τον Frank E. Grubbs, ο οποίος και την εξέδωσε στην πλήρη της μορφή το 1969. Προυποθέτει την ύπαρξη ενός δείγματος που υπακούει στην κανονική κατανομή και είναι ταξινομημένο κατάλληλα, και με τη γραμμική εξέταση του δείγματος, αποφαίνεται εάν η πρώτη τιμή αποτελεί ακρότατο ή όχι. Συγκεκριμένα, σε δείγμα *n* τιμών, για τον έλεγχο της σημαντικότητας της μέγιστης τιμής προτείνει τη μετρική

$$\frac{S_n}{S} = \frac{\sum_{i=1}^{n-1}(x_i - \bar{x}_n)}{\sum_{i=1}^{n-1}(x_i - \bar{x})} \tag{3.1}$$

όπου

$$x_1 \leq x_2 \leq \cdots \leq x_n, \bar{x}_n = \tfrac{1}{n-1}\sum_{i=1}^{n-1} x_i$$

και

$$\bar{x} = \tfrac{1}{n}\sum_{i=1}^{n} x_i.$$

Παρομοίως, για τον έλεγχο της σημασίας της μικρότερης τιμής μπορούμε να χρησιμοποιήσουμε το μέγεθος S_1/S. Αυτή είναι και η μετρική που μας ενδιαφέρει στα πλαίσια αυτής της εργασίας. Για

την επιβεβαίωση ότι η εξεταζόμενη τιμή αποτελεί έκτοπο, θα πρέπει να ικανοποιεί τη συνθήκη

$$\frac{\overline{x} - x_1}{\sigma} > \frac{N-1}{\sqrt{N}} \cdot \sqrt{\frac{(t_{\alpha/N,N-2})^2}{N-2+(t_{\alpha/N,N-2})^2}} \tag{3.2}$$

όπου $t_{\alpha/N,N-2}$ η κρίσιμη τιμή με $N-2$ βαθμούς ελευθερίας και βαθμό αξιοπιστίας α/N της κατανομής Student–T. Στο παράρτημα C παρατίθεται πίνακας με μεγάλο πλήθος αυτών των τιμών σε προσεγγιστικό βαθμό.

Η δεύτερη μέθοδος είναι γνωστή ως έλεγχος των τεταρτημορίων. Αποτελεί μία τεχνική για να χωρίσουμε ένα δειγματικό χώρο σε τέσσερα όσο το δυνατόν περισσότερο ίσα μέρη. Επιλέγει χαρακτηριστικές τιμές της κατανομής και τις ορίζει τεταρτημόρια με τον εξής τρόπο. Ως πρώτο τεταρτημόριο Q_1 ορίζεται η παρατήρηση που χωρίζει το 25% των χαμηλότερων τιμών με το υπόλοιπο 75%, δεύτερο τεταρτημόριο Q_2 αποτελεί η διάμεσος τιμή, τρίτο Q_3 είναι ό αριθμός που χωρίζει το χαμηλότερο 75% των τιμών με το υψηλότερο 25%. Τα παραπάνω μεγέθη μπορούν να υπολογιστούν μονοσήμαντα για δείγματα με άρτιο πλήθος παρατηρήσεων, στα υπόλοιπα θα πρέπει να υιοθετήσουμε μία από τις διαθέσιμες μεθόδους [RobJ96] για την αποτίμησή τους. Τέλος, ως διατεταρτημοριακή απόσταση IQR ορίζεται η διαφορά $Q_3 - Q_1$.

Με βάση τα παραπάνω μεγέθη, μπορούμε να χρησιμοποιήσουμε τις σχέσεις

$$\begin{aligned} \Phi_{low} &= Q_1 - k \cdot IQR \\ \Phi_{high} &= Q_3 + k \cdot IQR \end{aligned} \tag{3.3}$$

όπου $k = 1.5$ αλλά ανάλογα με την ελαστικότητα που θέλουμε να έχει ο αλγόριθμός μας μπορούμε να το μεταβάλλουμε κατά λίγα δεκαδικά ψηφία. Μετά την ανατροφοδότηση που λάβαμε, για αξιόπιστα αποτελέσματα θα πρέπει να ισχύει $1.2 \leq k \leq 1.7$ όπου γραμμικά με το k αυξάνεται και η αυστηρότητα του ελέγχου. Οι τιμές Φ_{low} και Φ_{high} αποτελούν τα κάτω και άνω φράγματα αντίστοιχα για τις αναμενόμενες παρατηρήσεις. Εμάς για ακόμα μια φορά μας ενδιαφέρει εάν υπάρχουν τιμές x_i, όπου $x_i < \Phi_{low}$ και απο αυτές η χαμηλότερη.

Με τη συγκεκριμένη υλοποίηση, παρατηρούμε ότι για τον υπολογισμό κάθε μίας από τις κρίσιμες τιμές πρέπει να διασχίσουμε το δείγμα μας αρκετές φορές, το οποίο σημαίνει ότι λόγω της γραμμικής πολυπλοκότητας ο χρόνος εκτέλεσης αυξάνεται ανεπίτρεπτα για μεγάλες τιμές παρατηρήσεων. Στη δική μας περίπτωση, το πλήθος των τιμών που πρέπει να εξετάσουμε δεν θα υπερβαίνει τη μία εκατοντάδα οπότε το παραπάνω δεν αποτελεί λόγο ανησυχίας.

Η τρίτη μέθοδος που εξετάσαμε ονομάζεται κριτήριο του Chauvenet προς τιμήν του William Chauvenet, Αμερικανού καθηγητή μαθηματικών και αστρονομίας που έζησε το $19^ο$ αιώνα και είχε μείζονα ρόλο στη θέσπιση της Σχολής Αμερικανικού Ναυτικού όπου και δίδασκε για χρόνια. Χρησιμοποιεί και αυτή μεγέθη όπως ο διάμεσος, η μέση τιμή και η τυπική απόκλιση για τον ορισμό του ελέγχου εαν ένα δείγμα περιέχει ακραίες τιμές. Προσπαθεί να ορίσει ένα εύρος τιμών για τα αποδεκτά όρια της τυπικής απόκλισης με βάση τη μέση τιμή, για το οποίο με βάση τις κρίσιμες τιμές της κανονικής κατανομής έχουμε ορθές παρατηρήσεις. Στο παράρτημα D παρατίθεται ο πίνακας που περιέχει τις προσεγγίσεις αυτών των τιμών.

Εδώ, για να συμπεράνουμε εάν η τιμή x_i αποτελεί ακρότατο σε δείγμα **x** με μέγεθος n, μέση τιμή μ και τυπική απόκλιση σ, θα πρέπει να ισχύει η σχέση

$$n \cdot \text{erfc}\,(\frac{|x_i - \mu|}{\sigma}) < \frac{1}{2} \tag{3.4}$$

όπου erfc η συμπληρωματική τιμή της συνάρτησης σφάλματος της κανονικής κατανομής. Αποτιμάται από τη σχέση

$$\text{erfc}\,(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \tag{3.5}$$

Είναι φανερό πως η μετρική που χρησιμοποιεί αυτή η τεχνική θυμίζει αρκετά αυτή του Grubbs. Αντικαθιστά όμως τις κρίσιμες τιμές της κατανομής Student–T με αυτές της κανονικής και υιοθετεί αυθαίρετα τη σταθερά 1/2 για τη διεξαγωγή του συμπεράσματος.

Όλες οι μέθοδοι που αναλύσαμε παραπάνω είναι ευαίσθητες ως προς με το μέγεθος του δείγματος. Κάτι τέτοιο είναι ευνόητο, διότι η επίδραση μίας ακραίας τιμής σε μικρό δείγμα θα επηρεάζει έντονα τα μεγέθη στα οποία βασιζόμαστε για τους υπολογισμούς μας. Εδώ, τα δείγματα που χρησιμοποιούμε ποικίλλουν σε μέγεθος και δεν έχουμε κάποιο έλεγχο πάνω σε αυτό οπότε λαμβάνουμε τα κατάλληλα μέτρα για να περιορίσουμε τα περιθώρια λανθασμένων εκτιμήσεων. Στην προσπάθειά μας να εξαλείψουμε τα σφάλματα, σε κάθε νέο δείγμα που λαμβάνουμε, εκτελούμε και τους τρεις ελέγχους και επιζητούμε να επιτύχουν τουλάχιστον δύο από αυτούς για να λάβουμε την απόφαση ως προς την ύπαρξη προσφοράς. Παρατηρούμε όμως από τα αποτελέσματα ότι πολλές φορές δεν υπάρχει απόλυτη συμφωνία στη συμπεριφορά των αλγορίθμων και θα παρουσίαζε ενδιαφέρον να μελετήσουμε την περαιτέρω αιτιολόγηση της συμπεριφοράς τους.

Παράλληλα, για ακόμα μεγαλύτερη ευελιξία επιλέξαμε να παραμετροποιήσουμε τους τρεις αυτούς ελέγχους σε τρεις βαθμούς ευελιξίας ή ανοχής. Ορίσαμε την αυστηρή, κανονική και χαλαρή ανοχή, οι οποίες χρησιμοποιούν διαφορετικές σταθερές στον υπολογισμό των κρίσιμων τιμών. Με τον τρόπο αυτό μπορούμε να εντοπίσουμε πόσο “ισχυρή” ή όχι είναι μια προσφορά. Ο τρόπος που το επιτύχαμε αυτό είναι η επεξεργασία των μετρικών ελέγχου, όπως αυτές λήφθηκαν σε κάθε έλεγχο. Συγκεκριμένα, στον έλεγχο Grubbs, χρησιμοποιήσαμε μεταβλητό μέγεθος μεταξύ των τιμών 0.1, 0.05 και 0.01 το οποίο εκφράζει την πιθανότητα λανθασμένης μέτρησης. Παρατηρούμε ότι όσο μικρότερη η τιμή του, τόσο μεγαλύτερος ο βαθμός εμπιστοσύνης μας στο αποδιδόμενο αποτέλεσμα.

Αντίστοιχα, στον έλεγχο Chauvenet, βασιζόμαστε στη σύγκριση της κρίσιμης μετρικής με τον αριθμό 0.5. Με ανάλογο τρόπο, μετατοπίσαμε την τιμή αυτή προς τα πάνω και κάτω κατά 10% έτσι ώστε να δημιουργηθούν οι τρείς διαφορετικές της εκφάνσεις. Τέλος, στον έλεγχο Τεταρτημορίων, επιλέχθηκε η μεταβολή του κατά 0.3 πάνω και κάτω για την αποτίμηση των διάφορων εκφάνσεων του ελέγχου. Οι τιμές αυτές επιλέχθηκαν μετά από την εφαρμογή Unit Tests στην κάθε μονάδα έτσι ώστε να γίνεται πάντα ορθή αναγνώριση των αποτελεσμάτων σε εύρος δειγμάτων με ποικιλόμορφα χαρακτηριστικά.

Κεφάλαιο 4

Σχεδιασμός εφαρμογής

Ένα από τα πιο σημαντικά στάδια ανάπτυξης κάθε δικτυακής εφαρμογής είναι ο σχεδιασμός. Ένα εύρωστο πλάνο σχεδιασμού θα περιορίσει κατά πολύ τους κύκλους ανατροφοδότησης και τις αχρείαστες αλλαγές λόγω λανθασμένων ή ελλιπών διασαφήσεων. Συνεπώς, προσπαθήσαμε καθ' όλη τη διάρκεια της εκπόνησης αυτής της διπλωματικής εργασίας να ορίσουμε τις απαιτήσεις με τον πληρέστερο τρόπο για να καλύπτεται πλήρως η επιθυμητή λειτουργικότητα. Ταυτόχρονα, κύριο μέλημα ήταν το μοντέλο μας να είναι ανεκτικό και εύκολα ευέλικτο σε πιθανές μελλοντικές αλλαγές ή επεκτάσεις.

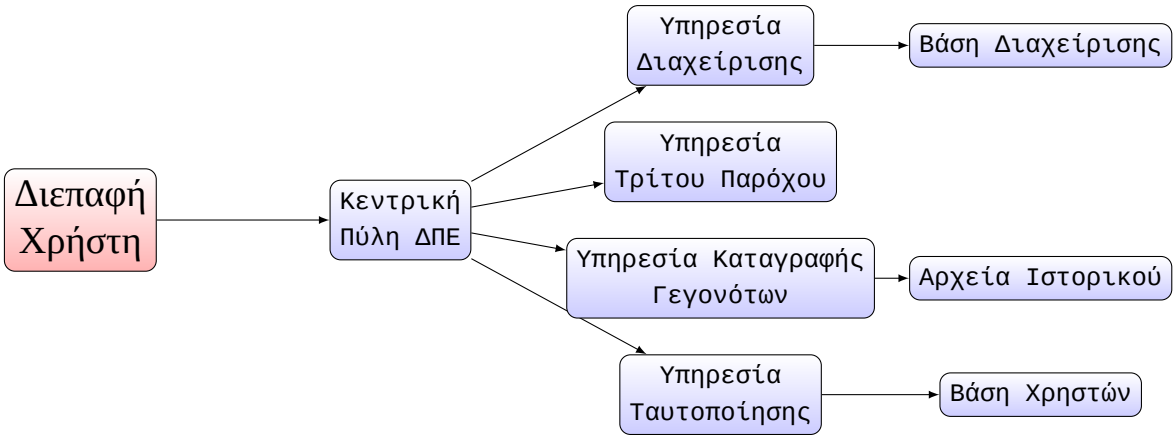
Στο παρόν κεφάλαιο θα γίνει μία προσπάθεια ανάλυσης της αρχιτεκτονικής του συνολικού συστήματος αλλά ταυτόχρονα και των χαρακτηριστικών της κάθε δομικής μονάδας από τη σκοπιά του μηχανικού λογισμικού. Θα περιγραφούν οι τεχνοτροπίες που χρησιμοποιήσαμε και οι λόγοι που οδήγησαν στην επιλογή τους, ακολουθώντας την κατηγοριοποίηση που έχουμε επιλέξει μέχρι στιγμής.

4.1 Αρχιτεκτονική

Στο κόσμο της ανάπτυξης λογισμικού, δίνεται ιδιαίτερη σημασία στην επιλογή της αρχιτεκτονικής κάθε συστήματος. Ο όρος αρχιτεκτονική, προερχόμενος από τον ομώνυμο τομέα, εκφράζει τη στρατηγική επιλογή της τοπολογίας ενός συστήματος, καθορίζοντας τις δυνατότητες του, και τα επιμέρους υποσυστήματα που το αποτελούν, σε υψηλό επίπεδο αφαίρεσης, έτσι ώστε να μην προκληθούν τεχνικοί περιορισμοί κατά την υλοποίηση. Ανάλογα με το πεδίο εφαρμογής, τις απαιτούμενες λειτουργίες αλλά και τη φύση της εφαρμογής, η κάθε αρχιτεκτονική μπορεί να διαφέρει ριζικά. Χαρακτηριστικά παραδείγματα που χρησιμοποιούν ετερογενείς αρχιτεκτονικες είναι εφαρμογές διαδικτύου, γραμμής εντολών, παιχνίδια, οδηγοί υλικού, γλώσσες προγραμματισμού αλλά τελικά κάθε λογισμικό οριοθετείται και συμπεριφέρεται με βάση τον αρχιτεκτονικό σχεδιασμό του.

Ειδικότερα, στο πεδίο του δικτυακού προγραμματισμού, ο οποίος, την ώρα συγγραφής του παρόντος κειμένου, αποτελεί την πλέον διαδεδομένη μορφή ανάπτυξης εφαρμογών [Over17], υπάρχει μία πανσπερμία από απόψεις σχετικά με την βέλτιστη αρχιτεκτονική, ανάλογα πάντα με το μοντέλο προγραμματισμού και τις παραμέτρους που αναφέραμε παραπάνω. Εθιμοτυπικά υπήρχε η άποψη της μονολιθικής εφαρμογής, δηλαδή η ανάπτυξη όλων των τεχνικών ικανοτήτων με χρήση μίας πλατφόρμας, η οποία οδηγεί σε δυσκίνητα, δυσνόητα συστήματα καθώς οι απαιτήσεις αλλάζουν και επεκτείνονται. Τα τελευταία χρόνια, η παγκόσμια προγραμματιστική κοινότητα έχει διαθέσει πληθώρα εργαλείων ανάπτυξης, με άδειες χρήσης ανοικτού λογισμικού, τα οποία έχουν προκαλέσει τη δημιουργία νέων αντιλήψεων και μια τάση για πιο κατανεμημένες αρχιτεκτονικές, αποτελούμενες από ατομικά εξαρτήματα, τα οποία έχουν απολύτως οριοθετημένες και σημασιολογικά συγκεκριμένες ευθύνες, γνωστά και ως μικροϋπηρεσίες (microservices) [Drag16]. Το σχήμα 4.1 παρουσιάζει ένα σχετικό δείγμα δικτυακής εφαρμογής.

Ταυτόχρονα, ένα εξίσου σημαντικό πλήθος προγραμματιστούν υποστηρίζουν την υπηρεσιοκεντρική αρχιτεκτονική, η οποία με τη σειρά της διακρίνει και δίνει βάρος στις διάφορες υπηρεσίες που παρέχονται μέσω μίας εφαρμογής [Perr03]. Και σε αυτή την αντιμετώπιση είναι φανερό ότι υπάρχει διαχωρισμός και κατανομή των συστημάτων σε μικρότερα, με ειδοποιό διαφορά την ύπαρξη λιγότερων, πιο "ισχυρών", με την έννοια της λογικής, τμημάτων, τα οποία έχουν έντονη αλληλεξάρτηση και



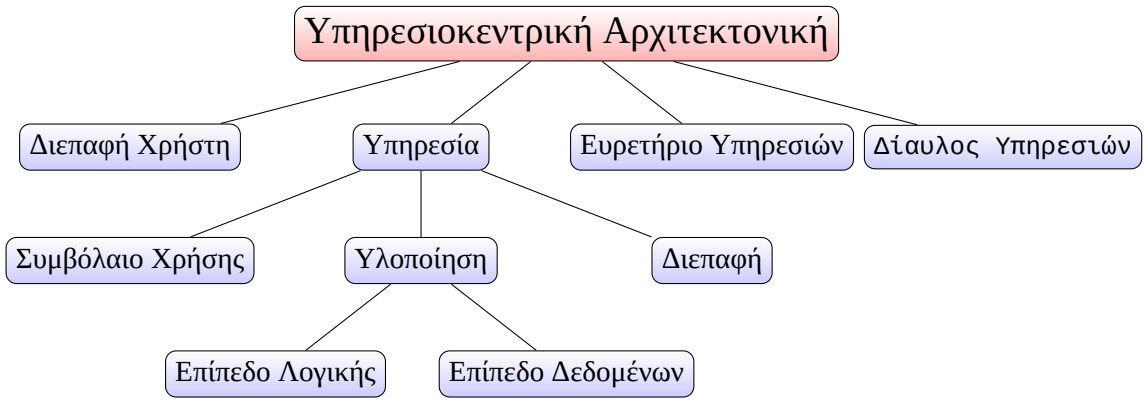
Σχήμα 4.1: Τοπολογία Αρχιτεκτονικής Μικροϋπηρεσιών

αυξημένες ευθύνες σε σχέση με την αρχιτεκτονική μικρο-υπηρεσιών. Δύο βασικές οντότητες ενορχηστρώνουν τη λειτουργικότητα σε ένα τέτοιο σύστημα, οι Πάροχοι Υπηρεσιών (service providers) και οι Καταναλωτές Υπηρεσιών (service consumers), μεταξύ των οποίων δημιουργούνται σημασιολογικές σχέσεις εξάρτησης. Χαρακτηριστικό παράδειγμα Καταναλωτή είναι μία διεπαφή χρήστη. Το σχήμα **??** αποδίδει μία ιεραρχική αναπαράσταση των στοιχείων μίας υπηρεσιοκεντρικής αρχιτεκτονικής σε εφαρμογές επαγγελματικών απαιτήσεων κατά τους Krafzig, Banke και Slama [Kraf05]. Παρατηρούμε ότι υπό αυτή τη νοοτροπία, μία εφαρμογή αποτελείται από τη Διεπαφή Χρήστη, το Ευρετήριο Υπηρεσιών που λειτουργεί ως ο βασικός τρόπος ανακάλυψης των διαθέσιμων υπηρεσιών στο περιβάλλον που έχει πρόσβαση η εφαρμογή και το Δίαυλο Υπηρεσιών, ο οποίος διαχειρίζεται την εγγυημένη μετάδοση μηνυμάτων μεταξύ της Διεπαφής Χρήστη και των διαθέσιμων Υπηρεσιών. Οι τελευταίες λειτουργούν ως αυτόνομες οντότητες καθώς καθορίζουν εσωτερικά τη διεπαφή τους με το περιβάλλον και αναλαμβάνουν να ορίσουν τα συμβόλαια επικοινωνίας, όπως τα πρωτόκολλα μετάδοσης, τις μορφές σειριοποίησης δεδομένων και φυσικά την ίδια την υλοποίησή τους με ό,τι αυτό συνεπάγεται. Δεν θα προχωρήσουμε σε περαιτέρω ανάλυση διότι ξεπερνά τους σκοπούς του παρόντος κειμένου.

Φυσικά, οι τάσεις στις μεθοδολογίες προγραμματισμού εφαρμογών αλλάζουν, καθώς ανακαλύπτονται οι τυχόν αδυναμίες τους και συνέχεια βγαίνουν στην επιφάνεια υβριδικά μοντέλα με διαφορετική σημασιολογία, τα οποία εξυπηρετούν συγκεκριμένες ανάγκες, όπως για παράδειγμα η παραλληλοποίηση διεργασιών, η εγγυημένη μετάδοση μηνυμάτων σε πολλαπλούς παραλήπτες και η ανάθεση ρόλων σε διαφορετικούς χρήστες σε συνδυασμό με την εξασφάλιση της σωστής δικαιοδότησης.

Σε αυτή τη διπλωματική εργασία βασιστήκαμε σε υπηρεσιοκεντρική αρχιτεκτονική. Τα κύρια συστατικά είναι ο Καταναλωτής της Διεπαφής Προγραμματισμού Εφαρμογών που υλοποιήθηκε με μόνη ευθύνη την κατανάλωση της Διεπαφής Προγραμματισμού Εφαρμογών (API) που διατίθεται δωρεάν από το Skroutz.gr, ο Πάροχος στατιστικών ελέγχων, ο Πάροχος που εξυπηρετεί τη λήψη αντικειμένων από τη Βάση Δεδομένων της εφαρμογής και τέλος η ίδια η Διεπαφή Χρήστη η οποία καταναλώνει της υπηρεσίες που παρέχουν τα δύο προηγούμενα στοιχεία. Κρίσιμοι παράγοντες που λήφθηκαν υπόψη στη συγκεκριμένη αρχιτεκτονική είναι η απομόνωση των μονάδων λογισμικού σε συνδυασμό με την ελάττωση των αλληλεξαρτήσεων. Ιδανικά, θα πρέπει όλα τα στοιχεία να μπορούν να λειτουργήσουν αυτόνομα στο βαθμό που αυτό είναι δυνατό. Με αυτό το σκεπτικό, ο Πάροχος στατιστικών ελέγχων, για παράδειγμα, σχεδιάστηκε και υλοποιήθηκε έτσι ώστε να εντοπίζει ακραίες τιμές σε δείγματα τιμών μεταβλητής υποδιαστολής και όχι στα ίδια τα προϊόντα έτσι ώστε θεωρητικά να είναι επαναχρησιμοποιήσιμος και σε εφαρμογές με διαφορετικό ερευνητικό αντικείμενο. Παρατηρούμε ταυτόχρονα ότι ο Καταναλωτής της ΔΠΕ του Skroutz.gr έχει άμεση και αναντικατάστατη εξάρτηση από τις δομές δεδομένων που λαμβάνει από το σχετικό Πάροχο, άρα υπάγεται σε περιορισμούς. Σε περίπτωση αλλαγής των δυνατοτήτων ή των δομών της διεπαφής, θα χρειασθεί μεταβολή και της αντίστοιχης υλοποίησης. Παρατηρούμε λοιπόν ότι ο προγραμματιστής έχει μεγαλύτερη ευελιξία στην υλοποίηση ενός

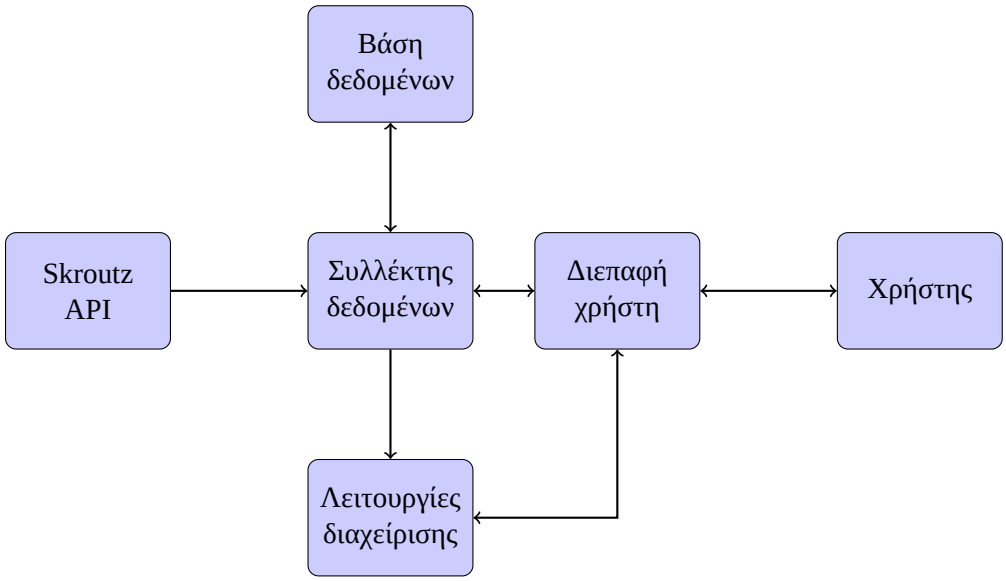
Παρόχου Υπηρεσιών παρά ενός Καταναλωτή. Παρ' όλα αυτά, η αρχιτεκτονική αυτή καθιστά όλα τα συστατικά εύκολα στην αντικατάσταση. Σε περίπτωση που υπήρχαν διαφορετικές υλοποιήσεις μίας ΔΠΕ, θα μπορούσαν, με την κατάλληλη εγκατάσταση, να υποκαταστήσουν αυτές που αναπτύχθηκαν από το συγγραφέα χωρίς ιδιαίτερη παρέμβαση.



Σχήμα 4.2: Στοιχεία της Υπηρεσιοκεντρικής Αρχιτεκτονικής

4.2 Μοντέλο εφαρμογής

Μία από τις κυριότερες επιδιώξεις μας για την οργάνωση της εφαρμογής ήταν η δημιουργία ενός γραφικού μοντέλου που να περιγράφει τη διασύνδεση των διάφορων μονάδων. Έτσι μπορούμε να δούμε το εύρος λειτουργίας του κάθε συστατικού μέσω μίας αναπαράστασης υψηλού επιπέδου, χωρίς τεχνικές λεπτομέρειες. Η τελική μορφή αυτής της μοντελοποίησης φαίνεται στο σχήμα 4.3.



Σχήμα 4.3: Ροή δεδομένων στην εφαρμογή

Το διάγραμμα αποτελείται από τρεις βασικούς παράγοντες. Αριστερά φαίνεται η ΔΠΕ του Skroutz η οποία είναι η κύρια πηγή δεδομένων μας, στη μέση η εφαρμογή μας, αποτελούμενη από διάφορα υποσυστήματα και δεξιά ο τελικός χρήστης. Η κατεύθυνση των βελών αντιπροσωπεύει τη διεύθυνση της ροής δεδομένων μεταξύ κάθε συστατικού. Μπορούμε με τον τρόπο αυτό να κατανοήσουμε ότι το BargainHunt είναι ένας Πάροχος Υπηρεσιών ο οποίος δημιουργεί μία ροή πληροφορίας που δεν υπήρχε νωρίτερα μεταξύ των άλλων δύο παραγόντων. Εκεί έγκειται και η αξία του. Για την περαιτέρω

ανάλυση θα πρέπει να κατέβουμε ένα αφαιρετικό επίπεδο χαμηλότερα, για να βρεθούμε στα πλαίσια της δικής μας εφαρμογής. Αντιμετωπίζοντας την ως ένα σύστημα, θα πρέπει να αναγνωρίσουμε τις εισόδους του, τις εξόδους του, και τη συνάρτηση μεταφοράς ή αλλιώς την εσωτερική του λειτουργία.

Όσον αφορά στις εισόδους, υπάρχουν αρκετές παράμετροι που θα πρέπει να λάβουμε υπόψη. Αρχικά το κανάλι μετάδοσης είναι ένα, και είναι η ΔΠΕ του Skroutz. Αυτό το κανάλι όμως, πολυπλέκει στο εσωτερικό του πλήθος δεδομένων τα οποία θα πρέπει να καταγραφούν και να μεταφραστούν με ορθό τρόπο για να αποκτήσουν αξία. Συνεπώς, αναλάβαμε διαβάσουμε τη διαθέσιμη τεκμηρίωση [Skro] και να δημιουργήσουμε ένα δέκτη πληροφοριών που να μπορεί να χειριστεί κάθε δυνατή είσοδο και να την ερμηνεύσει με τον ανάλογο τρόπο. Ένα παράδειγμα τέτοιας εισόδου φαίνεται στο απόσπασμα κώδικα 10.

```
{
    "sku": {
        "id": 3690169,
        "ean": "",
        "pn": "D802 16GB",
        "name": "G2 (16GB)",
        "display_name": "LG G2 (16GB)",
        "category_id": 40,
        "first_product_shop_info": null,
        "click_url": null,
        "price_max": 599.0,
        "price_min": 288.74,
        "reviewscore": 4.63905,
        "shop_count": 37,
        "plain_spec_summary": "Single SIM, Δίκτυο Σύνδεσης: 2G / 3G / 4G -
        ↪  LTE, Κάμερα: 13 MP,  Οθόνη: 5.2\" , Ανάλυση: 1920 x 1080 pixels,
        ↪  Χειρισμός: Οθόνη αφής (Touch screen),  Εσωτερική Μνήμη: 16.0 GB,
        ↪  Λειτουργικό Σύστημα: Android, SmartPhone",
        "manufacturer_id": 30,
        "future": false,
        "reviews_count": 169,
        "virtual": false,
        "images": {
            "main": "http://b.scdn.gr/images/.../medium_LGG2_Front.jpg",
            "alternatives": [
                "http://a.scdn.gr/images/sku_images/.../LGG2_Side.jpg",
                "http://a.scdn.gr/images/sku_images/.../thum_gallery3.png",
                "http://a.scdn.gr/images/sku_images/.../thum_gallery6.png",
                "http://b.scdn.gr/images/sku_images/.../LGG2.jpg",
                "http://a.scdn.gr/images/sku_images/.../thum_gallery4.png"
            ]
        }
    }
}
```

Απόσπασμα 10: Παράδειγμα προϊόντος σε μορφή JSON

Ένα πρώτο χαρακτηριστικό που παρατηρούμε είναι ο τρόπος κωδικοποίησης της πληροφορίας. Βλέπουμε ότι χρησιμοποιείται η μορφή JSON (JavaScript Object Notation), η οποία παρά τη μικρή της ηλικία, τείνει να γίνει πρότυπο λόγω της απλότητας και της λακωνικότητας που διαθέτει σε σχέση με

τη γλώσσα XML. Με τη μορφή αυτή μπορούμε εύκολα να αναπαραστήσουμε δομές όπως αντικείμενα και λίστες και όλους τους βασικούς τύπους δεδομένων μέσα σε αυτά. Στη συγκεκριμένη περίπτωση βλέπουμε ότι μεταδίδεται η πληροφορία σχετικά με το προϊόν "(SKU) LG G2 (16GB)", και αυτή είναι μία από τις απλούστερες περιπτώσεις. Ανάλογα με την αναζήτηση που εκτελεί ο χρήστης, τον τρόπο γραφής του και την ύπαρξη ή μη του αντικειμένου που ψάχνει, δημιουργούνται πολλοί δυνατοί συνδυασμοί απαντήσεων, τους οποίους και προσπαθήσαμε να καλύψουμε από την πλευρά μας. Υπάρχει δυνατότητα να παραλείψουμε ή να ζητήσουμε πιο λεπτομερή δεδομένα για μία οντότητα μέσω των παραμέτρων που παρέχουμε κατά το αίτημα μας.

Χαρακτηριστικό παράδειγμα για το παραπάνω είναι η αναζήτηση με λανθασμένο λεκτικό, όπως για παράδειγμα "apple ipone". Στην περίπτωση αυτή, η απάντηση που θα λάβουμε θα μοιάζει με αυτή του αποσπάσματος 11.

Παρατηρούμε ότι τώρα η απάντηση περιέχει μία λίστα από κατηγορίες αλλά και το αντικείμενο `meta`, στο οποίο υπάρχουν πληροφορίες γύρω από την αναζήτησή μας. Συγκεκριμένα, βλέπουμε τον όρο που αναζητήσαμε, αν υπάρχουν εναλλακτικές ονομασίες ή οντότητες που ταιριάζουν ιδιαίτερα στην αναζήτησή μας και λεπτομέρειες σχετικά με το πλήθος των αποτελεσμάτων και των σελίδων στις οποίες είναι χωρισμένα αυτά. Με παρόμοιο τρόπο αντιμετωπίζονται αναζητήσεις που ταιριάζουν με κάποιο πρότυπο από τις οντότητες που ορίζει ο Skroutz.

Έχοντας καθορίσει τη ροή εισόδου που λαμβάνει η εφαρμογή μας, είμαστε σε θέση να χτίσουμε τη μονάδα συλλογής δεδομένων, τον συνδετικό κρίκο δηλαδή με τη ΔΠΕ του Skroutz. Η μονάδα αυτή σχεδιάστηκε με τέτοιο τρόπο έτσι ώστε να συμπεριφέρεται βέλτιστα με βάση τις διευκολύνσεις αλλά και τους περιορισμούς που μας παρέχει η τελευταία ΔΠΕ. Δεδομένης της σχετικά χαμηλής πολυπλοκότητας των διαδικασιών της εφαρμογής, η επικοινωνία με τρίτους παράγοντες είναι ένα από τα σημεία που προσθέτουν μεγάλη χρονική καθυστέρηση στην διεκπεραίωση μίας ολοκληρωμένης επικοινωνίας. Στην προσπάθειά μας να την ελαχιστοποιήσουμε, προσθέτουμε ένα επιπλέον επίπεδο μνήμης μεταξύ της υπηρεσίας δεδομένων του Skroutz, τη δική μας βάση δεδομένων. Στα πλαίσια αυτής της λογικής, χρησιμοποιούμε την πληροφορία που έχουμε περί "τελευταίας τροποποίησης" στα δεδομένα μας. Για κάθε μία οντότητα που αποθηκεύουμε τοπικά, κρατάμε, εφόσον αυτό είναι δυνατόν, μία μονοσήμαντη τιμή που κωδικοποιεί τη στιγμή της τελευταίας τροποποίησης που δέχθηκε η οντότητα. Έτσι, σε μελλοντικές αναζητήσεις, υλοποιούμε αιτήματα "υπό όρους", όπου παρέχουμε την τιμή αυτή και ο εξυπηρετητής απαντά με την κατάλληλη επικεφαλίδα για να μας ενημερώσει εάν η οντότητα έχει υποστεί αλλαγές, και ανάλογα με το ενδεχόμενο παρέχει τις αλλαγές ή όχι. Με τον τρόπο αυτό επιτυγχάνουμε σημαντική οικονομία στην επικοινωνία μας, γεγονός κρίσιμης σημασίας καθώς η ΔΠΕ του Skroutz έχει θέσει περιορισμούς σχετικά με τα διαθέσιμα αιτήματα ανά λεπτό. Φυσικά, στα ερευνητικά πλαίσια χρήσης, τα όρια αυτά είναι υπέρ αρκετά, όμως σε μία πραγματική εφαρμογή, η λύση μας θα ήταν απαραίτητη.

Στην προσπάθεια μας να χωρίσουμε την εφαρμογή σε καλά ορισμένες μονάδες, η καθεμία από τις οποίες έχει ξεχωριστές αρμοδιότητες, προσπαθήσαμε να οργανώσουμε την επικοινωνία με βάση τα επίπεδα της εφαρμογής. Χαρακτηριστικά, το επίπεδο εμφάνισης επικοινωνεί αποκλειστικά με τον συλλέκτη δεδομένων, ο οποίος με τη σειρά του λειτουργεί είτε ως λήπτης των πληροφοριών της ΔΠΕ, είτε ως ενδιάμεσος για την ανάκτηση και εγγραφή στη βάση δεδομένων.

Όσον αφορά στο επίπεδο των δεδομένων, αξίζει να αναφέρουμε ότι ουσιαστικά η δική μας βάση λειτουργεί σαν ένα επιπλέον επίπεδο μνήμης για τις οντότητες που χειριζόμαστε, εμπλουτίζοντας τες με πληροφορίες όπως εάν βρίσκονται σε προσφορά και το ιστορικό των τιμών τους. Έτσι περιορίζουμε την εξάρτησή μας από τρίτους και μειώνουμε τους χρόνους απόκρισης των αναζητήσεων. Όπως προστάζουν οι μοντέρνες τακτικές ανάπτυξης λογισμικού, το επίπεδο των δεδομένων επικοινωνεί αποκλειστικά με τον Ελεγκτή, ο οποίος στη δική μας περίπτωση είναι ο Συλλέκτης Δεδομένων.

Τα επίπεδα εμφάνισης, συγκεκριμένα ο ιστότοπος που βλέπει ο τελικός χρήστης καθώς και η διεπαφή διαχείρισης, έχουν δικαιώματα επικοινωνίας με το αμέσως χαμηλότερο επίπεδο. Ανάλογα με τη διεπαφή, διαφορετικές λειτουργίες ενεργοποιούνται στον ελεγκτή έτσι ώστε να είναι δυνατές οι επιθυμητές ενέργειες. Ο τελικός χρήστης έχει πρόσβαση μόνο στην πρώτη οθόνη, η οποία του δίνει τη δυνατότητα να πραγματοποιήσει τις αναζητήσεις που επιθυμεί και να λάβει τις αντίστοιχες

```
{
  "categories": [
    {
      "id": 754,
      "name": "Διάφορα Αξεσουάρ MP3",
      "children_count": 0,
      "image_url": "http://d.scdn.gr/images/categories/large/754.jpg",
      "parent_id": 230,
      "fashion": false,
      "layout_mode": "list",
      "web_uri": "http://api.skroutz.gr/c/754/diafora-aksesouar-mp3.html",
      "path": "76,1269,767,230,754",
      "show_specifications": false,
      "manufacturer_title": "Κατασκευαστές",
      "match_count": 1
    },
    {
      "id": 579,
      "name": "Θήκες Κινητών Τηλεφώνων",
      "children_count": 0,
      "image_url":
       ↪  "http://c.scdn.gr/images/categories/large...3958_42b293c7.jpg",
      "parent_id": 86,
      "fashion": false,
      "layout_mode": "tiles",
     "web_uri": "http://api.skroutz.gr/c/579/thikes-kinhtwn-thlefwnwn.html",
      "path": "76,1269,86,579",
      "show_specifications": false,
      "manufacturer_title": "Κατασκευαστές",
      "match_count": 2
    }
  ],
  "meta": {
    "q": "apple ipone",
    "alternatives": [

    ],
    "strong_matches": {
    },
    "pagination": {
      "total_results": 2,
      "total_pages": 1,
      "page": 1,
      "per": 2
    }
  }
}
```


Απόσπασμα 11: Παράδειγμα απάντησης στον όρο αναζήτησης “apple ipone”

πληροφορίες. Αντίθετα, η διεπαφή διαχείρισης έχει ως στόχο την απλό και γρήγορο συντονισμό των απαραίτητων ενεργειών για την ανανέωση του συστήματος με καινούριες πληροφορίες.

4.3 Διεπαφή Χρήστη

Ο σχεδιασμός της διεπαφής χρήστη βασίστηκε στην τεχνοτροπία που προτείνει το πλαίσιο Vaadin. Συγκεκριμένα, υπάρχει μία και μόνο σελίδα, της οποίας το περιεχόμενο ανανεώνεται ασύγχρονα, ανάλογα με την ενέργεια του χρήστη. Χάρη στους μηχανισμούς που λειτουργούν στο παρασκήνιο, κάθε φορά που κάνουμε ένα κλικ, μεταβάλλεται η τοποθεσία URL του φυλλομετρητή μέσα στην εφαρμογή μας. Τότε ο εξυπηρετητής, τη διαβάζει, και αποστέλλει την απαραίτητη πληροφορία στο χρήστη μέσω αντικειμένων JSON και μεθόδων σε γλώσσα JavaScript, οι οποίες αναλαμβάνουν να μεταποιήσουν τον πηγαίο κώδικα που μεταγλωττίζεται στο τερματικό του πελάτη, εμφανίζοντας τη νέα πληροφορία μόλις αυτή είναι έτοιμη και αποκρύπτοντας την παλαιά. Από πλευρά απόκρισης, ο σχεδιασμός αυτός αγγίζει τη βελτιστοποίηση. Αυτό συμβαίνει επειδή το μείζον πλήθος των εμφανών συστατικών έχουν προ-αρχικοποιηθεί και αποστέλλονται με τη μορφή κώδικα JavaScript στο χρήστη, κατά την πρώτη επαφή του με την εφαρμογή. Σίγουρα κάτι τέτοιο δε θα ήταν αποδοτικό στο ενδεχόμενο μίας περίπλοκης εφαρμογής με πολλούς χρήστες, λόγω της αυξημένης επιβάρυνσης που θα δεχόταν ο εξυπηρετητής κατά τη δημιουργία νέων συνόδων. Για την αντιμετώπιση του συγκεκριμένου προβλήματος έχουν αναπτυχθεί λογισμικά που λειτουργούν ως ένα επιπλέον επίπεδο κρυφής μνήμης και παρεμβάλλονται μεταξύ πελάτη και εξυπηρετητή, με σκοπό να ανακαλύπτουν διπλότυπα αιτήματα προς στατικό περιεχόμενο και να το ανακατευθύνουν κατάλληλα, επιτυγχάνοντας την αποσυμφόρηση του τελευταίου.

Επιλέξαμε να αφιερώσουμε μία ξεχωριστή οθόνη σε κάθε μία από τις ενέργειες που παρείχαμε στα πλαίσια απλότητας και ερευνητικής πληρότητας που θελήσαμε να επιτύχουμε κατά την ανάπτυξη. Έτσι καταλήξαμε με τέσσερις σελίδες.

- Η αρχική σελίδα που περιέχει το πλαίσιο με τις καλύτερες υπάρχουσες προσφορές και τη μπάρα αναζήτησης. Αν και αποτελεί εξαίρεση στον κανόνα που αναφέραμε προηγουμένως, αυτό έγινε σκόπιμα, για να είναι πιο εύκολα διακριτή η αξία που προσφέρει η εφαρμογή.

- Η σελίδα αποτελεσμάτων αναζήτησης, της οποίας το περιεχόμενο διαφοροποιείται ανάλογα με το στάδιο στο οποίο αυτή βρίσκεται. Ξεκινώντας από την παράθεση κατηγοριών, συνεχίζουμε μέχρι να καταλήξουμε σε ένα μοναδικό προϊόν (SKU). Για να γίνει δυνατή η χρήση μίας και μόνο σελίδας για την απεικόνιση ετερόμορφων δεδομένων, δημιουργήσαμε ορισμένα σχεδιαστικά πλαίσια στα οποία εμφανίζουμε τις βασικές πληροφορίες για τις οντότητες που εξετάζουμε με ομοιόμορφο τρόπο.

- Η σελίδα προϊόντος όπου πλέον βλέπουμε λεπτομέρειες για το προϊόν όπως τις τιμές στις οποίες διατίθεται, ένα ιστορικό τιμών του και φυσικά αν αυτό βρίσκεται σε προσφορά με βάση τις στατιστικές μεθόδους που χρησιμοποιήσαμε.

- Τέλος, αναπτύξαμε τη σελίδα διαχείρισης. Αν και κάτι τέτοιο δεν ήταν απαραίτητο, προτιμήσαμε τη λύση αυτή για την ομοιομορφία της εφαρμογής ως διαδικτυακή οντότητα. Η σελίδα αυτή είναι προσβάσιμη μόνο από το διαχειριστή και δίνει δυνατότητες ανανέωσης των δεδομένων που λαμβάνουμε από τη ΔΠΕ του Skroutz, καθώς επίσης και τη μαζική εύρεση προσφορών.

Τα θέματα που εξετάστηκαν στο παρόν κεφάλαιο είναι καθοριστικής σημασίας στην υλοποίηση που ακολούθησε μετέπειτα.

Κεφάλαιο 5

Θέματα υλοποίησης

Στη συνέχεια θα δοθεί έμφαση στην τεχνική όψη της εφαρμογής, θα γίνει λόγος για συγκεκριμένα λογισμικά και θα αναλυθεί το πως τα χρησιμοποιήσαμε και τι επιτύχαμε με αυτά. Ουσιαστικά θα γίνει μία παρουσίαση της ήδη υπάρχουσας τεκμηρίωσης σε προγραμματιστικό επίπεδο και μία πλήρης ανασκόπηση των ενεργειών που φέραμε σε πέρας για την δημιουργία του συγκεκριμένου προγράμματος. Για το διαχωρισμό των υποκεφαλαίων, θα ακολουθήσουμε τη λογική με την οποία διασπάσαμε την εργασία σε μονάδες, τις οποίες και θα αναλύσουμε ξεχωριστά, αναλύοντας με λεπτομέρεια τα πιο σημαντικά σημεία.

5.1 Συλλέκτης δεδομένων

Η συγκεκριμένη μονάδα, όπως έχουμε δει και νωρίτερα, είναι η πρώτη μονάδα που αναπτύξαμε κατά το παρόν εγχείρημα. Ο λόγος είναι απλός, διότι θέλαμε ένα μηχανισμό λήψης δεδομένων, ο οποίος να μπορεί να επικοινωνεί με έναν ιστότοπο και να καταναλώνει τη διεπαφή REST του τελευταίου με το βέλτιστο τρόπο. Θέλαμε επίσης να είναι ένα αυτοτελές συστατικό, με την έννοια ότι δεν θα έχει εξαρτήσεις από την υπόλοιπη εφαρμογή και ταυτόχρονα θα επικοινωνεί άμεσα με τη βάση δεδομένων μας. Το πρώτο στάδιο σε αυτή την προσπάθεια ήταν η συστηματική ανάγνωση και καταγραφή των δυνατοτήτων της παρεχόμενης ΔΠΕ από τον κατάλληλο ιστότοπο [Skro]. Εφόσον έγιναν σαφείς λοιπόν οι δυνατότητες που έχουμε στον τρόπο λήψης δεδομένων μπορέσαμε να σχεδιάσουμε μία εφαρμογή που να υποστηρίζει την εξαγωγή δεδομένων και αποσειριοποίηση (deserialization) τους σε αντικείμενα των κλάσεων της εφαρμογής μας.

Τα εφόδια που μας δίνει μία αντικειμενοστραφής γλώσσα προγραμματισμού μας επέτρεψαν να κάνουμε χρήση αρκετών τεχνικών για την αποφυγή επαναλήψεων κατά τη συγγραφή του προγράμματος. Ένα παράδειγμα για το παραπάνω είναι η χρήση ενός αφαιρετικού επιπέδου, μέσω μίας αφηρημένης κλάσης που χαρακτηρίζει όλες τις οντότητες που συναντούμε στην ιστοσελίδα του Skroutz. Οι τελευταίες μοιράζονται ορισμένα χαρακτηριστικά, όπως για παράδειγμα το μοναδικό αναγνωριστικό τους και πεδία όπως η ημερομηνία προσθήκης και επεξεργασίας. Για το λόγο αυτό, κρίναμε ότι η δημιουργία μίας κλάσης που να χαρακτηρίζει όλες αυτές τις οντότητες και να ορίζει την κοινή τους λειτουργικότητα ήταν απαραίτητη. Η τελευταία φαίνεται στο απόσπασμα 12.

Εφόσον αποκτήσαμε μία κλάση η οποία διαθέτει σημασιολογικό χαρακτήρα, μπορούμε να δημιουργήσουμε τις υπόλοιπες βασιζόμενοι σε αυτή. Στο παράδειγμα των κατασκευαστών που παραθέτουμε στο 13, χάρη στον ορισμό `extends SkroutzEntity`, μπορούμε να δημιουργήσουμε νέες κλάσεις που υιοθετούν την `SkroutzEntity`, παραλείποντας τον αντίστοιχο κώδικα.

Κάτι που είναι επίσης φανερό στα δύο προηγούμενα παραδείγματα, είναι η πολύπλεξη των κλάσεων με μεταδεδομένα αναφορικά με το συσχετισμό των αντικειμένων και των πινάκων της βάσης, η οποία επιτυγχάνεται με τη χρήση επισημάνσεων (annotations) όπως οι `@MappedSuperclass`, `@Entity` και `@Column`. Κρίσιμο ρόλο όμως έχουν και οι υπόλοιπες, όπως οι `@NotNull`, `@Size` και `@Temporal` για την ταυτοποίηση των πεδίων με τις σωστές τιμές. Οι επισημάνσεις αυτές είναι ο τρόπος με τον οποίο ενεργοποιούμε το πλαίσιο EclipseLink στην εφαρμογή μας. Μπορούμε έτσι να έχουμε μία μόνο κλάση, εμπλουτισμένη με όλους τους απαραίτητους ορισμούς για να μπορεί να ληφθεί και να καταγραφεί στη βάση δεδομένων και να αποσειριοποιηθεί κατάλληλα από τις αποκρίσεις

του Skroutz. Το συγκεκριμένο πλαίσιο χρησιμοποιεί επίσης το αρχείο `persistence.xml`, το οποίο περιέχει όλες της πληροφορίες που χρειάζεται ο οδηγός της Postgres για να δημιουργήσει αυτόματα τους συσχετισμούς και ενεργοποιείται κατά την έναρξη της εφαρμογής.

Στη συνέχεια, οδηγηθήκαμε στην υλοποίηση της τεχνοτροπίας DAO (Data Access Object), δηλαδή οντοτήτων άρρητα δεμένων με αυτές του επιπέδου δεδομένων, οι οποίες είναι υπεύθυνες για τις βασικές λειτουργίες ΔΑΕΔ Δημιουργίας Ανάγνωσης Ενημέρωσης Διαγραφής (Create, Read, Update, Delete) καθώς επίσης και για επιπλέον διαδικασίες που εμπίπτουν στα πλαίσια της επιχειρησιακής λογικής (business logic). Ομοίως με παραπάνω, όσον αφορά στις ανάγκες υλοποίησης, αρχικά ορίσαμε δύο διεπαφές, ως παρουσιάζεται στα παραδείγματα 14 και 15.

Οι δύο παραπάνω διεπαφές έχουν αναπτυχθεί με τη χρήση γενικεύσεων (generics). Αυτή η τεχνολογία έγινε διαθέσιμη στην έκδοση 5 της Java και επιτρέπει την εφαρμογή μεθόδων και διεργασιών σε αντικείμενα διαφορετικών τύπων. Στη δική μας περίπτωση, οι τύποι αυτοί είναι της μορφής `SkroutzEntity`, `Request` ή `Offer`. Στη συνέχεια, υλοποιήσαμε όλες τις βασικές CRUD λειτουργίες σε μία κλάση η οποία υλοποιεί τις παραπάνω διεπαφές και λειτουργεί καθολικά με όλες τις οντότητες που θα χρησιμοποιήσουμε. Οι εξειδικευμένες διεργασίες που χρειαστήκαμε ανάλογα με την κάθε οντότητα, υλοποιήθηκαν στις αντίστοιχες κλάσεις-παιδιά. Για παράδειγμα, οι μέθοδοι για να αναζητούμε την κλάση “Προϊόν” σε σχέση με την κατηγορία, το κατάστημα, ή τον κατασκευαστή του αποτυπώθηκαν στην ίδια την κλάσση και καλοούνται με δυναμικό τρόπο από την ύπερ-κλάσση γενίκευσης.

Η βιβλιοθήκη Jackson 2 χρησιμοποιήθηκε για τη διευκόλυνση της σειριοποίησης και αποσειριοποίησης αντικειμένων. Ομοίως με το EclipseLink, ο τρόπος που το επιτυγχάνουμε αυτό είναι να αντιστοιχήσουμε κάθε πεδίο των επιθυμητών κλάσεων σε μία συμβολοσειρά που θα αντιπροσωπεύει το πεδίο του αντικειμένου σε μορφή JSON. Αυτό υλοποιείται με τη χρήση επισημάνσεων όπως είδαμε παραπάνω, αλλά με διαφορετικό συντακτικό. Η συγκεκριμένη βιβλιοθήκη έχει σχεδιαστεί έτσι ώστε να συνεργάζεται αυτόματα με τη βιβλιοθήκη Jersey που χρησιμοποιείται για την επικοινωνία μέσω πρωτοκόλλου HTTP και ο συνδυασμός τους αποτελεί ιδανικό εργαλείο για τον προγραμματιστή για την ανάπτυξη εφαρμογών που συνδέονται με διεπαφές τύπου REST αλλά και για κάθε ανάγκη διαδικτυακής επαφής. Έτσι, λαμβάνοντας μία απάντηση HTTP που περιέχει ένα ή περισσότερα αντικείμενα, και έχοντας εφαρμόσει την κατάλληλη αντιστοίχηση, μπορούμε με την κλήση μόνο μίας μεθόδου να μεταβούμε από αντικείμενα JSON σε αντικείμενα Java που χρησιμοποιούμε σε όλα τα υπόλοιπα στάδια της εφαρμογής. Κατά την αποσειριοποίηση αυτή, καλείται ο κατασκευαστής που έχουμε επιλέξει και χαρτογραφούνται τα πεδία του με βάση τις τιμές που λάβαμε. Το μόνο σημείο που επεμβαίνουμε εμείς στον κώδικα είναι η λογική που εφαρμόζουμε για να καταλάβουμε τι είδους αντικείμενα περιέχει το μήνυμα που λάβαμε και ο ορισμός των αντιστοιχίσεων όπως φαίνεται στον κατασκευαστή του αποσπάσματος 16.

Στην περίπτωση που επιθυμούμε να αγνοήσουμε ορισμένα πεδία εφόσον χρειάζονται κάποια επεξεργασία ή απλά δεν τα χρειαζόμαστε, μπορούμε να το επιτύχουμε με την επισήμανση `@JsonIgnore`. Η συγκεκριμένη βιβλιοθήκη μπορεί να επιτύχει κάθε δυνατή μετατροπή στα δεδομένα μεταξύ των χρησιμοποιούμενων μορφών και βρίσκεται προεγκατεστημένη στα πακέτα του εξυπηρετητή Wildfly. Συνεχίζοντας, υλοποιήσαμε την κλάση που υλοποιεί όλες τις βασικές λειτουργίες και αναλαμβάνει την καταγραφή όλων των ενεργειών, η οποία σχεδιάστηκε έτσι ώστε να μπορεί να υιοθετηθεί από νέες κλάσεις με σκοπό τη λήψη πληροφοριών από άλλες υπηρεσίες εκτός του Skroutz.

Τα αντικείμενά που δημιουργούμε αποθηκεύουν μόνο τις πληροφορίες που θα χρειαστούν στα πλαίσια της εφαρμογής. Σύνδεσμοι ιστότοπων, ονόματα και περιγραφές προϊόντων συγκρατούνται στη βάση, ενώ τηλέφωνα και τα λεπτομερή χαρακτηριστικά παραλείπονται ως περιττά.

5.2 Διαδικασίες ταυτοποίησης

Μία από τις ιδιαιτερότητες της επικοινωνίας με υπηρεσίες REST είναι η διαδικασία της ταυτοποίησης. Είναι ευνόητο ότι για λόγους ασφάλειας, νομικής αλλά και πρακτικής φύσεως δεν μπορεί ένας ιστότοπος που διαχειρίζεται δεδομένα τρίτων, να τα παρέχει αδιάκριτα σε κάθε ενδιαφερό-

μενο. Για το λόγο αυτό θα πρέπει να υπάρχει μία μορφή ταυτοποίησης με μόνιμο χαρακτήρα (Single Sign-on) [Hurs97], για την άμεση επικοινωνία, η οποία στο σύγχρονο διαδίκτυο υλοποιείται μέσω του πρωτοκόλλου OAuth 2.0 [Send14] το οποίο αποτελεί μία αναθεώρηση και επαναδιατύπωση του OAuth το οποίο δημοσιεύτηκε πρώτη φορά το 2006. Συνίσταται για την επικοινωνία εξυπηρετητών με ένα σύνολο από ταυτοποιημένους και μη πελάτες και ορίζει τη ροή πληροφοριών προς αυτούς. Δίνει ιδιαίτερη ελευθερία στον προγραμματιστή ως προς τις ρυθμίσεις, όμως η βασική λογική που το διέπει είναι η εξής. Ο πελάτης πραγματοποιεί ένα αίτημα προς μία τοποθεσία που απαιτεί ταυτοποίηση, τότε ο εξυπηρετητής παράγει ένα αναγνωριστικό χρήστη (`client_id`) και μία ιστοσελίδα για ανακατεύθυνση (`redirect_uri`) και κατά κανόνα δίνει τη δυνατότητα στο χρήστη να εγγραφεί στην υπηρεσία, επικοινωνώντας του ένα κωδικό δικαιοδότησης (`auth_code`), ο οποίος αποθηκεύεται στον πελάτη και χρησιμοποιείται για την έκδοση ενός προσωρινού κλειδάριθμου ταυτοποίησης (`authorization_token`). Ο συνδυασμός αναγνωριστικού και κλειδαρίθμου χρήστη χρησιμοποιείται σε κάθε επικοινωνία με τον εξυπηρετητή μέσω προκαθορισμένων εγγραφών της επικεφαλίδας ενός αιτήματος HTTP και έχει συγκεκριμένη διάρκεια ζωής, η οποία καθορίζεται από το διαχειριστή της υπηρεσίας. Στη συγκεκριμένη υλοποίηση, τα στοιχεία που περιγράψαμε παραπάνω μας παραδόθηκαν μέσω ηλεκτρονικού ταχυδρομείου και χρησιμοποιούμε τις ενδεδειγμένες μεθόδους κατά την υλοποίηση για τη δημιουργία του κλειδαρίθμου. Στην προσπάθειά μας να αυτοματοποιήσουμε τη διαδικασία για το χρήστη, η μόνη απαίτηση για τη σωστή λειτουργία είναι η καταγραφή των προσωπικών στοιχείων στο αρχείο `config.properties` όπως φαίνεται στο απόσπασμα 17.

Με τα στοιχεία αυτά αποθηκευμένα, η εφαρμογή, κατά τη λειτουργία της, ελέγχει εάν υπάρχει ο απαραίτητος κλειδάριθμος ταυτοποίησης και είτε αρχικοποιείται κανονικά, είτε εκκινεί τη διαδικασία έκδοσης κλειδαρίθμου, όπως αυτή ορίζεται στον ιστότοπο με τις οδηγίες για τον προγραμματιστή. Τέλος, για την ορθή λειτουργία των αιτημάτων, στην εκάστοτε επικεφαλίδα παραθέτουμε το εξής λεκτικό.

```
Authorization: Bearer your_access_token_here
```

5.3 Βάση Δεδομένων

Για την υλοποίηση της βάσης δεδομένων εφαρμόσαμε το σχήμα που προέκυψε μετά την κανονικοποίηση της στην Τρίτη Κανονική Μορφή όπως προστάζουν οι βέλτιστες πρακτικές σε ένα τέτοιο εγχείρημα. Για τη δημιουργία των πρωτευόντων κλειδιών χρησιμοποιήσαμε ακολουθίες, ένα μηχανισμό παραγωγής σειριακών αριθμών που ενθυλακώνει τη λογική της αυτόματης προσαύξησης. Παράλληλα, αποθηκεύσαμε το σχήμα σε γλώσσα SQL για την καταγραφή της πορείας του και για την γρήγορη επανασύστασή του κατά τη διεξαγωγή δοκιμών. Τελικά, κατά την πρώτη εκτέλεση, δημιουργήσαμε όλους τους απαραίτητους πίνακες όπως έχουν περιγραφεί νωρίτερα.

Ένα θέμα κρίσιμης σημασίας για τη σωστή λειτουργία της βάσης ήταν η ύπαρξη ενός αρχείου περιγραφής του μοντέλου αποθήκευσης όπως αυτό απαιτείται από το πλαίσιο μονιμότητας δεδομένων EclipseLink. Οι προδιαγραφές του καταγράφονται ρητά στο έγγραφο προσδιορισμού απαιτήσεων της διεπαφής JPA με κωδικό όνομα JSR-338 [aiafc] στην παράγραφο 8.2.1. Το τελευταίο κωδικοποιείται σε γλώσσα XML, όπως συνηθίζεται σε αρχεία ρυθμίσεων της Java Extended Edition και περιέχει όλες τις απαραίτητες πληροφορίες που θα πρέπει να επικοινωνήσει στον εξυπηρετητή της εφαρμογής για να γίνει ορθά ο συσχετισμός μεταξύ σχεσιακού και αντικειμενοστραφούς μοντέλου και ονομάζεται `persistence.xml`. Η πληροφορία που μεταδίδεται περιλαμβάνει το όνομα της πηγής δεδομένων όπως αυτή έχει οριστεί κατά τη διαδικασία εγκατάστασης, το μηχανισμό διαχείρισης συναλλαγών, το πλήθος των κλάσεων που αναπαριστούν τις οντότητές του σχήματος μας και τέλος ορισμένες πρόσθετες ρυθμίσεις για τη βέλτιστη διαχείριση πόρων και επιθυμητή λειτουργία της μονάδας αυτής. Το αρχείο που χρησιμοποιήσαμε φαίνεται στο απόσπασμα 18.

Παρατηρούμε ότι στην ταμπέλα `persistence-unit` δίνουμε τον τύπο διαχείρισης συναλλαγών με όνομα `JTA`. Ο τελευταίος είναι ο επίσημος τρόπος διαχείρισης συναλλαγών με πηγές δεδομένων σε εφαρμογές Java Enterprise Edition και διευκολύνει τον προγραμματιστή, επιτρέποντάς του να μην

ανησυχεί για το πότε θα πρέπει να καταγραφούν ή αναιρεθούν τυχόν αλλαγές στη βάση, αλλά δημιουργεί ένα ολόκληρο αφαιρετικό επίπεδο, το οποίο εκθέτει μέσω των μεθόδων `persist()`, `find()` `merge()` και `remove()` που χρησιμοποιούνται για όλες τις βασικές διαδικασίες διάδρασης με τη βάση. Αυτό που συμβαίνει, είναι ότι πριν την εκτέλεση μίας διαδικασίας που χρησιμοποιεί μία ή περισσότερες από τις παραπάνω μεθόδους, αρχικοποιείται μία νέα συναλλαγή, δηλαδή ένα σύνολο εντολών σε γλώσσα SQL, η οποία κατά την εντολή επιστροφής είτε θα μονιμοποιηθεί στη βάση, είτε θα αναιρεθεί στο ακέραιο της σε περίπτωση λάθους ή ανεπιθύμητης ροής.

Στη συνέχεια, μέσω της ετικέτας `jta-data-source`, η οποία είναι απαραίτητη εφόσον επιλέξαμε την αυτόματη διαχείριση συναλλαγών, αποτυπώνουμε το όνομα της τοποθεσίας της βάσης, σε μορφή που ορίζει η διεπαφή JNDI (Java Naming and Directory Interface) [aiafd]. Η τελευταία είναι μία υπηρεσία καταλόγου που επιτρέπει σε εφαρμογές Java να ανακαλύπτουν και να αποκτούν πρόσβαση σε δεδομένα και αντικείμενα από τρίτες εφαρμογές μέσω ενός ονόματος. Συναντά εκτεταμένη χρήση σε δικτυακές εφαρμογές και πολλές φορές παραμένει κρυφή από τον προγραμματιστή για την απλοποίηση πολλών διαδικασιών. Το όνομα που ορίσαμε εδώ πρέπει να ταυτίζεται με αυτό που έχει δηλωθεί στον εξυπηρετητή εφαρμογών, έτσι ώστε η σύνδεση να γίνει σωστά.

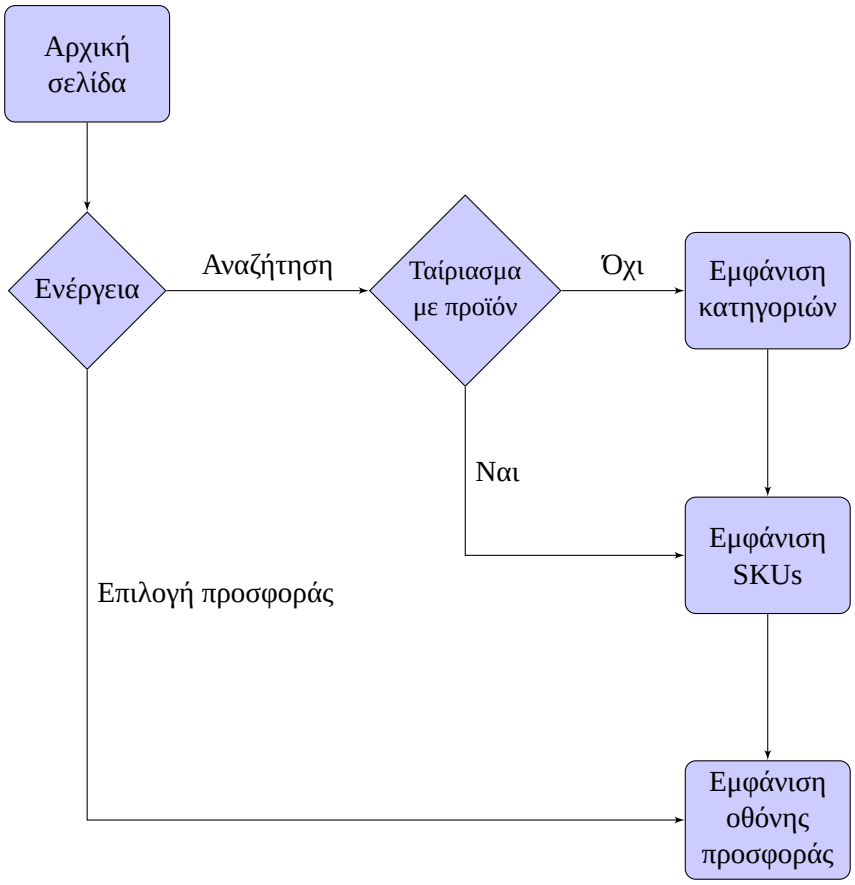
Απαραίτητο κομμάτι της συγγραφής αυτού του αρχείου είναι η απαρίθμηση όλων των οντοτήτων της εφαρμογής, δηλαδή όλων των κλάσεων που έχουν επισημανθεί με τις ετικέτες `@Entity` και `@MappedSuperclass` και αντιστοιχούν σε πίνακες της βάσης. Επιπλέον, αναφέρουμε τις ρυθμίσεις που θα χρησιμοποιήσει το EclipseLink για να αναγνωρίσει και να συντονίσει το περιβάλλον λειτουργίας του. Οι ρυθμίσεις αυτές είναι το επίπεδο αρχειοθέτησης, ο τύπος του λογισμικού RDBMS και ο τύπος του εξυπηρετητή εφαρμογών. Χάρη στις δυνατότητες των εργαλείων που χρησιμοποιήσαμε και τη συγγραφή του αρχείου σε γλώσσα ορισμού δεδομένων (Data Definition Language), δηλαδή ενός αρχείου όπου καταγράφονται όλες οι εντολές για τη δημιουργία και αρχικοποίηση της βάσης δεδομένων σε γλώσσα SQL, η υλοποίηση της συγκεκριμένης ενότητας δεν παρουσίασε ιδιαίτερες δυσκολίες. Τέλος, οι απαραίτητες ενέργειες για την εγκατάσταση και ρύθμιση της βάσης δεδομένων θα περιγραφούν αναλυτικά στο επόμενο κεφάλαιο.

5.4 Διεπαφή Χρήστη

Αυτό είναι ίσως το πιο τμήμα της διπλωματικής εργασίας. Είναι η υπηρεσία που θα εμφανίζεται στο χρήστη και στην οποία ο τελευταίος θα περιηγείται για να ανακαλύψει προσφορές σε προϊόντα που τον ενδιαφέρουν. Η υλοποίησή της έγινε με χρήση της πλατφόρμας Vaadin και του συνόλου εργαλείων που διαθέτει. Όπως έχουμε αναφέρει και νωρίτερα, η αρχική σελίδα περιέχει ένα πλαίσιο αναζήτησης και τις καλύτερες τρέχουσες προσφορές. Στην περίπτωση της αναζήτησης, ανάλογα με την είσοδο που θα παρέχει ο χρήστης, ξεκινά μία ροή για την εύρεση του επιθυμητού προϊόντος, που κρύβεται πίσω από το αφαιρετικό επίπεδο της οντότητας SKU. Αυτή η ροή μπορεί να είναι άμεση αν η αναζήτηση γίνει για ένα συγκεκριμένο μοντέλο συσκευής για παράδειγμα, ή έμμεση εάν το λεκτικό αναζήτησης είναι κάτι πιο γενικό, όπως π.χ. “Κινητό τηλέφωνο”. Εναλλακτικά, εάν ο χρήστης επιλέξει μία προσφορά από την αρχική σελίδα, μετακινείται αυτόματα στο σύνδεσμο του αντίστοιχου προϊόντος, δηλαδή προσπερνά τη ροή που περιγράψαμε παραπάνω. Για την καλύτερη κατανόηση, παραθέτουμε το διάγραμμα 5.1.

Αν και σχετικά απλό, το συγκεκριμένο διάγραμμα μας αρκεί για την πρωτοτυποποίηση της εφαρμογής ως προς τον τελικό χρήστη. Το διαχειριστικό κομμάτι, δηλαδή η ενημέρωση του περιεχομένου της βάσης, υλοποιήθηκε με τη μορφή μίας ακόμα διεπαφής, η οποία βρίσκεται στην τοποθεσία και περιλαμβάνει τη λίστα με όλες τις διαθέσιμες κατηγορίες που έχουμε καταγράψει σε συνδυασμό με την επιλογή αναδρομικής λήψης όλων των προϊόντων που περιέχουν. Τα δεδομένα που λαμβάνουμε από αυτή τη διαδικασία καταγράφονται πλήρως σε ξεχωριστή διεύθυνση αρχειοθέτησης.

Η υλοποίηση των παραπάνω έγινε με τη χρήση Όψεων (Views) του Vaadin οι οποίες περιέχουν όλα τα απαραίτητα συστατικά για επεξεργασία κειμένου, παρουσίαση εικόνας, παρουσίαση οντοτήτων και αναζήτηση. Μία όψη, για να λειτουργήσει σωστά, πρέπει να προστεθεί στον Πλοηγό (Navigator) της εφαρμογής μας. Ο τελευταίος αποτελεί μία τεχνοτροπία του Vaadin για τη συνεπή



Σχήμα 5.1: Διάγραμμα ροής της δικτυακής διεπαφής

απαρίθμηση και λειτουργία των διεπαφών που θέλουμε να έχει η εφαρμογή μας. Υπηρετώντας τη λογική της μίας σελίδας με πολλές εκφάνσεις που αναλύσαμε στο προηγούμενο κεφάλαιο, χρησιμοποιήσαμε ξεχωριστές κλάσεις για να αναπαραστήσουμε την κάθε σελίδα, τις οποίες και εντάξαμε στο πεδίο έκτασης του πλοηγού κατά την αρχικοποίηση του όπως φαίνεται στο 19.

```java
@MappedSuperclass
public class SkroutzEntity implements Serializable {

    protected long skroutzId;
    protected Date insertedAt;
    protected Date checkedAt;
    protected Date modifiedAt;
    protected String etag;

    public SkroutzEntity() {
        Date now = new Date();
        this.checkedAt = now;
        this.insertedAt = now;
    }

    @NotNull
    @Column(name = "skroutz_id")
    public long getSkroutzId() {
        return skroutzId;
    }

    @Size(max = 32)
    @Column(name = "etag")
    public String getEtag() {
        return etag;
    }

    @NotNull
    @Temporal(TemporalType.TIMESTAMP)
    @Column(name = "inserted_at")
    public Date getInsertedAt() {
        return insertedAt;
    }

    @Temporal(TemporalType.TIMESTAMP)
    @Column(name = "checked_at")
    public Date getCheckedAt() {
        return checkedAt;
    }

    @Temporal(TemporalType.TIMESTAMP)
    @Column(name = "modified_at")
    public Date getModifiedAt() {
        return modifiedAt;
    }
}
```

Απόσπασμα 12: Κώδικας MappedSuperclass στο JPA

```java
@JsonRootName("manufacturer")
@Entity
@Table(name = "manufacturers", schema = "public", catalog =
 ↪  "bargainhunt")
@NamedQuery(name = "Manufacturer.findAll", query = "select m from
 ↪  Manufacturer m")
public class Manufacturer extends SkroutzEntity {

    protected static final long serialVersionUID = -1L;

    private long id;
    private String name;
    private String imageUrl;

    public Manufacturer(@JsonProperty("id") long id,
                        @JsonProperty("name") String name,
                        @JsonProperty("image_url") String imageUrl) {
        super();
        this.skroutzId = id;
        this.name = name;
        this.imageUrl = imageUrl;
    }

    public Manufacturer() {
    }

    @Id
    @GeneratedValue(generator = "ManufacturerSequence")
    @SequenceGenerator(name = "ManufacturerSequence", sequenceName =
     ↪  "manufacturer_seq", allocationSize = 1)
    @Column(name = "id")
    public long getId() {
        return id;
    }

    @NotNull
    @Size(max = 100)
    @Column(name = "name")
    public String getName() {
        return name;
    }

    @Size(max = 300)
    @Column(name = "image_url")
    public String getImageUrl() {
        return imageUrl;
    }

    ...
}
```

Απόσπασμα 13: Παράδειγμα κλάσης τύπου SkroutzEntity

```java
/**
 * Generic Data Access Object interface.
 * @author zannis <zannis.kal@gmail.com>
 */
public interface GenericDao <T> {

    /** Persist a transient entity object into database */
    T persist(T transientEntity);

    /** Retrieve a persistent object from the database using its id */
    T find(long id);

    /**
     * Retrieve all the objects from the given type.
     * @return A list containing all the results.
     */
    List<T> findAll();

    /** Save changes made to a persistent object.  */
    T update(T transientEntity);

    /** Remove an object from persistent storage in the database */
    void remove(T persistentEntity);
}
```

Απόσπασμα 14: Ορισμός γενικευμένης διεπαφής Data Access

```java
/**
 * An interface made specifically for managing Skroutz entities. That is,
 ↪  entities
 * that contain a <code>skroutzId</code>.
 * @author zannis <zannis.kal@gmail.com>
 */
public interface GenericSkroutzDao<T> extends GenericDao<T> {

    /** Retrieve a persistent object from the database using its skroutzId */
    T findBySkroutzId(long id);

    /**
     * Retrieve all the skroutzIds from a given entity to quickly check
     * if they exist in the database.
     *
     * @return A list containing all the skroutzIds for the given entity.
     */
    List<Long> findAllSkroutzIds();
}
```

Απόσπασμα 15: Ορισμός διεπαφής Data Access για αντικείμενα τύπου SkroutzEntity

```java
@JsonCreator
public Shop(@JsonProperty("id") int skroutzId,
            @JsonProperty("name") String name,
            @JsonProperty("link") String link,
            @JsonProperty("phone") String phone,
            @JsonProperty("image_url") String imageUrl,
            @JsonProperty("thumbshot_url") String thumbshotUrl,
            @JsonProperty("reviews_count") int reviewCount,
            @JsonProperty("review_score") int reviewScore,
            @JsonProperty("payment_methods") PaymentMethods paymentMethods,
            @JsonProperty("shipping") Shipping shipping) {

    super();
    this.skroutzId = skroutzId;
    this.name = name;
    this.link = link;
    this.phone = phone;
    this.imageUrl = imageUrl;
    this.thumbshotUrl = thumbshotUrl;
    this.reviewCount = reviewCount;
    this.reviewScore = reviewScore;
    this.paymentMethods = paymentMethods;
    this.shipping = shipping;
}
```

Απόσπασμα 16: Κατασκευαστής με επισημάνσεις Jackson 2

```
client_id=     **********************
client_secret= **********************
redirect_uri= http://example.com/auth/skroutz
```

Απόσπασμα 17: Το αρχείο config.properties

```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence" version="2.1">
    <persistence-unit name="BargainHunt" transaction-type="JTA">
        <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
        <jta-data-source>java:jboss/BargainHuntDS</jta-data-source>
        <class>gr.ntua.cn.zannis.bargains.webapp.persistence.SkroutzEntity</class>
        <class>gr.ntua.cn.zannis.bargains.webapp.persistence.entities.Category</class>
        <class>gr.ntua.cn.zannis.bargains.webapp.persistence.entities.Shop</class>
        <class>gr.ntua.cn.zannis.bargains.webapp.persistence.entities.Sku</class>
        <class>gr.ntua.cn.zannis.bargains.webapp.persistence.entities.Product</class>
        <class>gr.ntua.cn.zannis.bargains.webapp.persistence.entities.Manufacturer</class>
        <class>gr.ntua.cn.zannis.bargains.webapp.persistence.entities.Price</class>
        <class>gr.ntua.cn.zannis.bargains.webapp.persistence.entities.Request</class>
        <properties>
            <property name="eclipselink.logging.level" value="INFO"/>
            <property name="eclipselink.target-database" value="PostgreSQL"/>
            <property name="eclipselink.target-server" value="JBoss"/>
        </properties>
    </persistence-unit>
</persistence>
```

Απόσπασμα 18: Το αρχείο persistence.xml

```java
private void initNavigator() {
    navigator = new Navigator(this, this);
    navigator.addProvider(viewProvider);
    navigator.addView(MainView.NAME, MainView.class);
    navigator.addView(SearchView.NAME, SearchView.class);
    navigator.addView(ProductsView.NAME, ProductsView.class);
    navigator.addView(BargainView.NAME, BargainView.class);
    navigator.addView(CrawlerView.NAME, CrawlerView.class);
    navigator.navigateTo(getNavigator().getState());
}
```

Απόσπασμα 19: Αρχικοποίηση πλοηγού στο αρχείο BargainHuntUI.java

Κεφάλαιο 6

Εγκατάσταση της εφαρμογής

Το κεφάλαιο αυτό έχει ως σκοπό την περιγραφή των απαιτούμενων ενεργειών για την επιτυχή εγκατάσταση της εφαρμογής σε ένα περιβάλλον εξυπηρετητή. Θα θεωρήσουμε ότι ο αναγνώστης χρησιμοποιεί περιβάλλον GNU/Linux καθώς έχουμε αναπτύξει ένα αρχείο σεναρίου με σκοπό να είναι δυνατή η εγκατάσταση με το πάτημα ενός κουμπιού. Όμοια αποτελέσματα μπορούν να επιτευχθούν και σε περιβάλλον Mac OSX ή Windows, με μικρή επέμβαση από την πλευρά του χρήστη, δεδομένων των ιδιαιτεροτήτων των δύο αυτών λειτουργικών συστημάτων. Θα θεωρήσουμε ότι το μηχάνημα που χρησιμοποιούμε έχει προεγκατεστημένα όλα τα απαιτούμενα τρίτα λογισμικά και χρησιμοποιεί συγκεκριμένα το λειτουργικό Ubuntu Server 14.04. Σε περίπτωση που γίνεται χρήση ενός νέου συστήματος, μπορούμε να το φέρουμε στην απαιτούμενη κατάσταση μέσω των εντολών που φαίνονται στο απόσπασμα 20, δεδομένου ότι έχουμε τα απαραίτητα δικαιώματα.

Αρχικά προσθέτουμε τα αποθετήρια που περιέχουν μέσα τα αρχεία της Java, του Wildfly και της PostgreSQL, στη συνέχεια ενημερώνουμε το σύστημα έτσι ώστε να λάβει όλες τις νέες εκδόσεις και εγκαθιστούμε ό,τι χρειάζεται.

6.1 Προαπαιτούμενα

Όπως έχουμε αναφέρει και σε προηγούμενα κεφάλαια, εργαστήκαμε σε γλώσσα Java, οπότε τα απαιτούμενα λογισμικά θα σχετίζονται με τη συγκεκριμένη γλώσσα προγραμματισμού. Πριν ξεκινήσουμε τη διαδικασία της εγκατάστασης θα πρέπει να έχουμε στη διάθεσή μας τουλάχιστον έναν υπολογιστή με στατική διεύθυνση IP (IP address) έτσι ώστε να υπάρχει συνέπεια ως προς την πρόσβαση στον ιστότοπο της εφαρμογής και λειτουργικό GNU/Linux εμπλουτισμένο με λειτουργίες εξυπηρετητή, όπως το Ubuntu Server 14.04. Ίσως το πλέον σημαντικό χαρακτηριστικό ενός τέτοιου συστήματος είναι η σταθερότητα, για αυτό το λόγο επιλέγουμε αξιόπιστες εκδόσεις των λογισμικών που θα χρησιμοποιήσουμε και αποφεύγουμε την ενημέρωσή τους εκτός αν υπάρχει σοβαρός λόγος, όπως η ανακάλυψη κάποιας αδυναμίας στην ασφάλεια, που καθιστά το σύστημα ευάλωτο σε κακόβουλες επιθέσεις.

Ανάλογα με τις ανάγκες μας και τους διαθέσιμους υπολογιστικούς πόρους, μπορούμε να επιλέξουμε τη διαμόρφωση του συστήματος σε έναν ή περισσότερους υπολογιστές. Όλοι αυτοί θα έχουν το ρόλο του εξυπηρετητή και θα διαχωρίζονται ανάλογα με την υπηρεσία που παρέχουν, για παράδειγμα ένα σχήμα διαχωρισμού είναι η χρήση δύο μηχανημάτων, ένα για την παροχή της ιστοσελίδας και ένα για την εξυπηρέτηση της βάσης δεδομένων, το οποίο θα βρίσκεται σε εσωτερικό δίκτυο για λόγους ασφαλείας. Σε αυτή την ανάλυση θα επιλέξουμε την εξυπηρέτηση όλων των μονάδων από ένα μηχάνημα για λόγους απλότητας. Σε διαφορετικό σενάριο, η διαδικασία θα παρέμενε ίδια σε μεγάλο βαθμό με ορισμένες αλλαγές στη διευθυνσιοδότηση των διάφορων εξαρτημάτων.

Έστω ότι έχουμε πρόσβαση λοιπόν στο μηχάνημα εξυπηρέτησης της εφαρμογής, το οποίο θα ονομάζουμε appserver στη συνέχεια και ο χρήστης μας έχει αυξημένα δικαιώματα εκτέλεσης προγραμμάτων. Το πρώτο βήμα είναι να επιβεβαιώσουμε ότι είναι εγκατεστημένα τα λογισμικά που φαίνονται στον πίνακα 6.1.

Αυτό σε περιβάλλον GNU/Linux γίνεται με εντολές στο τερματικό. Στην περίπτωση που κατά την εκτέλεσή τους παρατηρούμε αποτελέσματα παρόμοια με αυτά που φαίνονται στο απόσπασμα 21,

```
# Add Java repository
add-apt-repository ppa:webupd8team/java

# Add PostgreSQL repository and the necessary PGP key
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ 'lsb_release
 ↪  -cs'-pgdg main" >> /etc/apt/sources.list.d/pgdg.list'
wget -q https://www.postgresql.org/media/keys/ACCC4CF8.asc -O - | sudo
 ↪  apt-key add -

sudo apt-get update
sudo apt-get upgrade
sudo apt-get install --with-recommends install oracle-java8-installer
sudo apt-get install git maven vim postgresql postgresql-contrib

# Create a user acount on the system for wildfly:
sudo adduser --no-create-home --disabled-password --disabled-login wildfly

# Download and extract Wildfly
cd /tmp
sudo wget --tries=0 \
    --continue http://download.jboss.org/wildfly/10.1.0.Final/wildfly-
     ↪  10.1.0.Final.tar.gz
tar -xzvf wildfly-10.1.0.Final.tar.gz

# Make a symbolic link in /opt folder
sudo ln -s wildfly-10.1.0.Final /opt/wildfly
```

Απόσπασμα 20: Αρχείο αρχικοποίησης μηχανήματος εξυπηρετητή

Πρόγραμμα	**Έκδοση**
Java Development Kit 64-bit	1.7.0_85 ή νεότερη
Wildfly	8.2.0.Final ή νεότερη
PostgreSQL	9.3 ή νεότερη

Πίνακας 6.1: Απαιτήσεις της εφαρμογής σε τρίτα λογισμικά

σημαίνει ότι είμαστε έτοιμοι να προχωρήσουμε με την εγκατάσταση. Αξίζει να σημειωθεί ότι το σύστημα είναι συμβατό με αμφότερες τις υλοποιήσεις της Java, την επίσημη δια χειρός Oracle και την ανοιχτού λογισμικού OpenJDK.

Εφόσον έχουν εγκατασταθεί τα παραπάνω και το περιβάλλον μας είναι έτοιμο για την προσθήκη της εφαρμογής μας, πρέπει να ελέγξουμε εάν ο χρήστης μας έχει τα απαιτούμενα δικαιώματα για την διαμόρφωση των παραπάνω και εάν υπάρχει αρκετός διαθέσιμος χώρος στο σκληρό δίσκο του εξυπηρετητή. Επιπλέον, για τον ορισμό του χρήστη μας ως διαχειριστή του Wildfly θα πρέπει να το συμπληρώσουμε στο αρχείο διαμόρφωσης, να θέσουμε υπό τον έλεγχό μας ορισμένους φακέλους και να τον προσθέσουμε στους Management Users. Ανοίγουμε το αρχείο `/etc/default/wildfly.conf` και επιβεβαιώνουμε ότι υπάρχει η γραμμή

```
JBOSS_USER=wildfly
```

και στη συνέχεια εκτελούμε τις εντολές όπως φαίνονται στο απόσπασμα 22 με αυξημένα δικαιώματα. Επιβεβαιώνουμε ότι τα παραπάνω έχουν τεθεί σε ισχύ αν λάβουμε έξοδο παρόμοια με την εντολή του αποσπάσματος 23.

Εκτελούμε την παρακάτω εντολή και συμπληρώνουμε τα στοιχεία που επιθυμούμε

```
java -version
java version "1.7.0_85"
OpenJDK Runtime Environment (IcedTea 2.6.1) (7u85-2.6.1-5ubuntu0.14.04.1)
OpenJDK 64-Bit Server VM (build 24.85-b03, mixed mode)
psql --version
psql (PostgreSQL) 9.3.10
echo \$WILDFLY_HOME
/opt/wildfly-10.1.0.Final
```

Απόσπασμα 21: Ενδεικτικά αποτελέσματα σε σωστή διαμόρφωση συστήματος

```
sudo chown -R wildfly:wildfly /opt/wildfly-10.1.0.Final
sudo chown -R wildfly:wildfly /opt/wildfly
sudo chown -R wildfly:wildfly /var/log/wildfly

# Create service file to automatically start wildfly on boot
cp /opt/wildfly/docs/contrib/scripts/init.d/wildfly-init-debian.sh
 ↪  /etc/init.d/wildfly
update-rc.d /etc/init.d/wildfly defaults
sudo systemctl enable wildfly
```

Απόσπασμα 22: Οικιοποίηση τοποθεσίας αποθήκευσης του Wildfly

```
ls -ld \$WILDFLY_HOME
drwxr-xr-x 10 wildfly wildfly 4096 May 21  2017 /opt/wildfly-10.1.0.Final
```

Απόσπασμα 23: Φάκελος Wildfly με ορθά δικαιώματα

Για την εφαρμογή των αλλαγών θα πρέπει να κάνουμε αποσύνδεση και επανασύνδεση στο σύστημα. Τέλος, για την ορθή λειτουργία της εφαρμογής χρειαζόμαστε ιδανικά χώρο μεγαλύτερο των 200MB, κυρίως για τον επαρκή εμπλουτισμό της βάσης δεδομένων με πληροφορίες. Αυτό μπορούμε να το ελέγξουμε όπως ορίζεις το παράδειγμα 24. Στο συγκεκριμένο σύστημα έχουμε 8.0GB διαθέσιμα, άρα μπορούμε να προχωρήσουμε στην εγκατάσταση της εφαρμογής.

6.2 Εγκατάσταση

Η εγκατάσταση της εφαρμογής ουσιαστικά περιλαμβάνει τη διαμόρφωση της βάσης δεδομένων και του εξυπηρετητή για να αναγνωρίσουν τις μονάδες του συστήματός μας. Αυτό θα επεξηγηθεί αναλυτικά στη συνέχεια.

Βάση δεδομένων

Όπως έχουμε αναφέρει νωρίτερα, με τον όρο βάση δεδομένων αναφερόμαστε σε ένα ολόκληρο σύστημα διαχείρισης δεδομένων το οποίο περιλαμβάνει τη βάση, τη μηχανή αναζήτησης πάνω της και τα απαραίτητα εργαλεία για τη διαχείρισή της. Είναι ένα ιδιαίτερα βαθύ περιβάλλον που επιδέχεται πολλές τροποποιήσεις για τη βέλτιστη διαχείριση των πόρων του εξυπηρετητή. Για λόγους απλότητας θα χρησιμοποιήσουμε μόνο τις απαραίτητες σε αυτή την παρουσίαση και θα ξεκινήσουμε με τον ορισμό του χρήστη της βάσης που θα έχει δικαιώματα ανάγνωσης και εγγραφής πάνω της. Ο τελευταίος είναι απαραίτητος για λόγους ασφαλείας και αποφυγής ανεπιθύμητων ενεργειών πάνω στις πληρο-

```
\$WILDFLY_HOME/bin/add-user.sh
What type of user do you wish to add?
 a) Management User (mgmt-users.properties)
 b) Application User (application-users.properties)
(a): a

Enter the details of the new user to add.
Using realm 'ManagementRealm' as discovered from the existing property files.
Username : user
Password recommendations are listed below. To modify these restrictions edit
 ↪  the add-user.properties configuration file.
 - The password should not be one of the following restricted values {root,
  ↪  admin, administrator}
 - The password should contain at least 8 characters, 1 alphabetic
  ↪  character(s), 1 digit(s), 1 non-alphanumeric symbol(s)
 - The password should be different from the username
Password :
Are you sure you want to use the password entered yes/no? yes
Re-enter Password :
What groups do you want this user to belong to? (Please enter a comma
 ↪  separated list, or leave blank for none)[  ]:
About to add user 'test' for realm 'ManagementRealm'
Is this correct yes/no? yes
Added user 'test' to file '/opt/wildfly-
 ↪  10.1.0.Final/standalone/configuration/mgmt-users.properties'
Added user 'test' to file
 ↪  '/opt/wildfly-10.1.0.Final/domain/configuration/mgmt-users.properties'
Added user 'test' with groups  to file '/opt/wildfly-
 ↪  10.1.0.Final/standalone/configuration/mgmt-groups.properties'
Added user 'test' with groups  to file
 ↪  '/opt/wildfly-10.1.0.Final/domain/configuration/mgmt-groups.properties'
Is this new user going to be used for one AS process to connect to another AS
 ↪  process?
e.g. for a slave host controller connecting to the master or for a Remoting
 ↪  connection for server to server EJB calls.
yes/no? yes
To represent the user add the following to the server-identities definition
 ↪  <secret value="b2s=" />
```

φορίες μας από τρίτους. Μπορούμε να τον δημιουργήσουμε όπως φαίνεται στο απόσπασμα 25. Το επόμενο βήμα είναι να καθορίσουμε το τα δικαιώματα σύνδεσης του κάθε χρήστη. Επεξεργαζόμαστε το αρχείο `/etc/postgresql/9.3/main/pg_hba.conf` έτσι ώστε να περιέχει τους ακόλουθους ορισμούς σύμφωνα με το απόσπασμα 26.

Παραπάνω ορίζουμε δικαιώματα τοπικής πρόσβασης στους χρήστες postgres και bargainhunt με τη χρήση κωδικού. Κατά την ανάπτυξη της εφαρμογής, ο χρήστης bargainhunt είχε απομακρυσμένα δικαιώματα για λόγους επεξεργασίας και αποσφαλμάτωσης. Για να το επιτύχουμε αυτό αλλάζουμε την τελευταία γραμμη με την παρακάτω.

```
host bargainhunt bargainhunt 0.0.0.0/0 md5
```

Έτσι ολοκληρώνονται οι απαραίτητες ενέργειες για τη του συστήματος διαχείρισης βάσεων δεδομένων μας.

```
df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda1        20G   11G  8.0G  57% /
none            4.0K     0  4.0K   0% /sys/fs/cgroup
udev            991M  4.0K  991M   1% /dev
tmpfs           201M  424K  200M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none           1001M     0 1001M   0% /run/shm
none            100M     0  100M   0% /run/user
```

Απόσπασμα 24: Παράδειγμα χρήσης εντολής df

```
sudo su - postgres
psql -c "CREATE USER bargainhunt WITH PASSWORD 'bargainhunt'" -d template1
ROLE CREATED
```

Απόσπασμα 25: Παράδειγμα δημιουργίας χρήστη βάσης δεδομένων

```
# Database administrative login by Unix domain socket
local   all             postgres                                md5

# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
local    bargainhunt    bargainhunt                             md5
```

Απόσπασμα 26: Περιεχόμενα αρχείου **`pg_hba.conf`**

Εξυπηρετητής

Ο Wildfly, για να λειτουργήσει σωστά, θα πρέπει να εμπλουτιστεί με τα συστατικά λειτουργίας της βάσης δεδομένων PostgreSQL και του πλαισίου EclipseLink. Μιλάμε ουσιαστικά για τις σχετικές βιβλιοθήκες που έχει ανάγκη η εφαρμογή μας. Για τη βάση δεδομένων Postgres, αφού επιβεβαιώσαμε ότι χρησιμοποιούμε την έκδοση 9.3.10, θα λάβουμε και θα εγκαταστήσουμε τον οδηγό που έχει σχεδιαστεί για τη συγκεκριμένη έκδοση με τις εντολές που καταγράφονται στη συνέχεια. Για νεότερες εκδόσεις της Postgres, ο κώδικας στο απόσπασμα 27 πρέπει να αλλαχθεί κατάλληλα.

Καθώς είμαστε ακόμα συνδεδεμένοι στο περιβάλλον ελέγχου του Wildfly, θα κατεβάσουμε και θα προσθέσουμε τον οδηγό του EclipseLink σύμφωνα με το απόσπασμα 28.

Εάν δεν εμφανιστούν λάθη στην έξοδο, σημαίνει ότι η εγκατάσταση έγινε επιτυχώς. Το τελευταίο βήμα είναι να ορίσουμε την πηγή δεδομένων μας όπως αποκαλείται, η οποία δεν είναι άλλη από την PostgreSQL. Στο περιβάλλον εντολών του Wildfly πληκτρολογούμε την εντολή του αποσπάσματος 29.

Αφού τα συστήματά μας είναι έτοιμα, το μόνο που μένει είναι να θέσουμε την εφαρμογή μας σε λειτουργία. Η διαδικασία αυτή ονομάζεται deployment σε αγγλική ορολογία. Ουσιαστικά είναι η θέση του λογισμικού σε λειτουργία. Με βάση τα εργαλεία που χρησιμοποιήσαμε μέχρι τώρα, αυτό γίνεται οπως στο απόσπασμα 30.

Εάν τώρα φορτώσουμε στο φυλλομετρητή μας τη διεύθυνση θα πρέπει να δούμε μία παρόμοια οθόνη με αυτή της εικόνας 6.1. Σε αυτή την περίπτωση η εγκατάσταση ήταν επιτυχής.

```
cd /tmp
wget https://jdbc.postgresql.org/download/postgresql-9.3-1104.jdbc41.jar
cd \$WILDFLY_HOME/bin/
./jboss-cli.sh
You are disconnected at the moment. Type 'connect' to connect to the server
 ↪  or 'help' for the list of supported commands.
[disconnected /] connect
Authenticating against security realm: ManagementRealm
Username: zannis
Password:
[standalone@localhost:9990 /] module add --name=org.postgres
 ↪  --resources=/tmp/postgresql-9.3-1103.jdbc41.jar
 ↪  --dependencies=javax.api,javax.transaction.api
[standalone@localhost:9990 /] /subsystem=datasources/jdbc-
 ↪  driver=postgres:add(driver-name="postgres",driver-module-
 ↪  name="org.postgres",driver-class-name=org.postgresql.Driver)
{"outcome" => "success"}
```

Απόσπασμα 27: Εγκατάσταση Postgres στο Wildfly

```
cd /tmp
curl -O -J -L "http://search.maven.org/remotecontent?filepath=
org/eclipse/persistence/eclipselink/2.6.1/eclipselink-2.6.1.jar"
cd \$WILDFLY_HOME/bin/
./jboss-cli.sh --connect
[standalone@localhost:9990 /] module add --name=org.eclipse.persistence
 ↪  --resources=/tmp/eclipselink-2.6.1.jar
 ↪  --dependencies=asm.asm,javax.api,javax.ws.rs.api,javax.annotation.api,
javax.enterprise.api,javax.persistence.api,javax.transaction.api,
javax.validation.api,javax.xml.bind.api,org.antlr,
org.apache.commons.collections,org.dom4j,org.javassist,
org.jboss.as.jpa.spi,org.jboss.logging,org.jboss.vfs
```

Απόσπασμα 28: Εγκατάσταση οδηγού EclipseLink στο Wildfly

```
[standalone@localhost:9990 /] data-source add
 ↪  --jndi-name=java:jboss/BargainHuntDS --name=BargainHuntDS
 ↪  --connection-url=jdbc:postgresql://localhost/postgres
 ↪  --driver-name=postgres --user-name=bargainhunt --password=bargainhunt
```

Απόσπασμα 29: Προσθήκη πηγής δεδομένων στο Wildfly

```
cd \$WILDFLY_HOME/bin
./jboss-cli.sh --connect
[standalone@localhost:9990 /] deploy /path/to/webapp.war
```

Απόσπασμα 30: Deployment εφαρμογής

Σχήμα 6.1: Οθόνη λειτουργίας Wildfly

Κεφάλαιο 7

Τεκμηρίωση και σενάρια χρήσης

7.1 Τεκμηρίωση λειτουργίας

Η ορθή και πλήρης τεκμηρίωση είναι μία από τις πλέον επιζητούμενες λειτουργίες μίας εφαρμογής για τους ίδιους τους δημιουργούς αλλά και για άλλους προγραμματιστές σε περίπτωση μελλοντικής ανάπτυξης. Σε μία εποχή όπου είναι πολύ εύκολο για τον καθένα να υλοποιήσει ένα λογισμικό και το διαθέσει μέσω του διαδικτύου σε παγκόσμιο κοινό, είναι απαραίτητο να διατυπώνονται ρητά οι λειτουργίες του σε όλες τις ζητούμενες γλώσσες, στη δική μας περίπτωση η ελληνική και η αγγλική αλλά και σε προτυποποιημένη μορφή, όπως ορίζει η εκάστοτε γλώσσα προγραμματισμού.

Το πρότυπο τεκμηρίωσης της γλώσσας Java είναι το λεγόμενο Javadoc. Αποτελείται από το ομώνυμο προγραμματιστικό εργαλείο το οποίο λαμβάνει ως είσοδο το σύνολο των αρχείων πηγαίου κώδικα εμπλουτισμένο με ειδικά σχόλια στοιχειοθέτησης και εξάγει ένα ευανάγνωστο έγγραφο συνήθως σε μορφή HTML, Microsoft Word, LaTeXή τη μορφή που θα επιλέξει ο προγραμματιστής. Η χρήση του καθορίζεται από την αρχή τυποποίησης της εταιρείας Oracle, στην οποία ανήκει η γλώσσα, και προβλέπει τη λεπτομερή ανάλυση των δυνατοτήτων όλων των δημόσιων και προστατευμένων μεθόδων με βάση συγκεκριμένες απαιτήσεις. Στη συνέχεια θα προσπαθήσουμε να κάνουμε μία περιληπτική αναφορά στους κανόνες που διέπουν τη συγγραφή του κειμένου στοιχειοθέτησης όπως αυτοί αναφέρονται στην επίσημη ιστοσελίδα [aiafb].

Αρχικά, όπως αναφέραμε προηγουμένως, η τεκμηρίωση γράφεται σε μορφή ειδικά διαμορφωμένων σχολίων, μέσα στα αρχεία κλάσσεων και διεπαφών με κατάληξη `.java` που περιέχουν τον πηγαίο κώδικα και τοποθετούνται ακριβώς πριν από τη μέθοδο ή την κλάση που περιγράφουν. Αν παραλληλήσουμε ένα λογισμικό σε γλώσσα Java ως μία δενδική δομή αποτελούμενη από κόμβους-πακέτα (packages) και φύλλα-κλάσεις ή διεπαφές (classes or interfaces), επιθυμούμε η οργάνωση των κόμβων να ακολουθεί μία λογική πορεία ομαδοποίησης και όλα τα φύλλα του και οι λειτουργίες τους να είναι απολύτως τεκμηριωμένα. Συνεπώς η εν λόγω ανάλυση αφορά τις κλάσεις, τις διεπαφές καθώς και όλες τις δημόσιες (public) μεθόδους τους. Κατά την επεξεργασία των τελευταίων συνίσταται η παράλληλη συγγραφή της τεκμηρίωσης έτσι ώστε να αποτυπωθούν όλες οι απαραίτητες λεπτομέρειες που ενδέχεται να μεταβάλλονται κατά τον προγραμματισμό. Ακολουθεί ένα παράδειγμα τέτοιου σχολίου στο απόσπασμα 31.

Το τμήμα που μας ενδιαφέρει βρίσκεται στην αρχή του αποσπάσματος και περικλείεται από τα σύμβολα `/**...*/`. Εμφανισιακά μοιάζει με ένα κοινότυπο σχόλιο, όμως διαφέρει διότι στην πρώτη παράγραφο του περιγράφει τη μέθοδο που ακολουθεί και στη συνέχεια δηλώνει σημασιολογία στο κείμενο μέσω ορισμένων επισημασμένων λέξεων όπως οι `@param`, `@returns` και `@throws`. Οι πιο σημαντικές επισημάνσεις είναι:

- **`@author <Όνομα συγγραφέα>`**
  Για την ταυτοποίηση ενός ή περισσότερων συγγραφέων μίας κλάσης ή διεπαφής.

- **`@version <Όνομα έκδοσης>`**
  Πληροφορίες για τη συγκεκριμένη έκδοση μίας κλάσης ή διεπαφής.

```java
/**
 * Creates an unconditional GET HTTP request to the Skroutz API. All
 * unconditional public GET methods should use this since its preconfigured
 * using our custom configuration. This method follows redirects, accepts
 * JSON entities and contains the required authorization headers.
 *
 * @param requestUri The target URI.
 * @return A {@link Response} containing one or more entities or null.
 * @throws e {@link ProcessingException} in case the request fails.
 */
private Response sendGetRequest(URI requestUri) throws ProcessingException {
  Response response;
  try {
    response = ClientBuilder.newClient(config).target(requestUri)
        .property(ClientProperties.FOLLOW_REDIRECTS, true)
        .request(MediaType.APPLICATION_JSON_TYPE)
        .header(HttpHeaders.AUTHORIZATION, "Bearer " + token)
        .accept("application/vnd.skroutz+json; version=3")
        .get();
    // persist/merge request
    ((BargainHuntUI) UI.getCurrent()).getRequests().saveOrUpdate(new
     ↪  Request(Utils.getFullPathFromUri(requestUri),
     ↪  response.getEntityTag() != null ? response.getEntityTag().getValue()
     ↪  : null));
    remainingRequests =
     ↪  Integer.parseInt(response.getHeaderString("X-RateLimit-Remaining"));
  } catch (ProcessingException e) {
    log.error("Υπήρξε πρόβλημα κατά την εκτέλεση του GET request : " +
     ↪  requestUri, e);
    throw e;
  }
  return response;
}
```

Απόσπασμα 31: Παράδειγμα σχολίου σε μορφή Javadoc

- **`@param <Όνομα παραμέτρου> <Περιγραφή>`**
  Περιγραφή της παραμέτρου εισόδου μίας μεθόδου ή κατασκευαστή.
- **`@return <Περιγραφή>`**
  Πληροφορίες για τα δεδομένα που επιστρέφει μία μέθοδος.
- **`@throws <Τύπος εξαίρεσης> <Περιγραφή>`**
  Περιγραφή της εξαίρεσης με την οποία τερματίζει μία μέθοδος σε περίπτωση απρόσμενης συμπεριφοράς.
- **`@link <Πλήρες όνομα κλάσης>`**
  Δημιουργεί σύνδεσμο σε μία εξωτερική κλάση ή διεπαφή.

Μία ακόμα επισήμανση, όχι τόσο γνωστή στο ευρύ κοινό, αλλά πολύ χρήσιμη στα κατάλληλα πλαίσια, είναι η **`@tag`**. Ουσιαστικά χρησιμοποιείται για να επιστήσει την προσοχή του αναγνώστη σε μία λεπτομέρια που χρήζει σημασίας. Με αυτή μπορούμε να επισημάνουμε μία μέθοδο, η οποία θέλουμε

να χρησιμοποιείται μόνο ασύγχρονα για παράδειγμα, επειδή δύναται να καθυστερήσει η επιστροφή της τιμής της καιμοιάζει να προκαλέσει πρόβλημα στη γραμμική εκτέλεση του προγράμματος.

Αφού έχουμε εμπλουτίσει τον κώδικά μας με όλα τα απαραίτητα σχόλια τεκμηρίωσης, το επόμενο βήμα είναι η εκτέλεση του αρχείου `javadoc` με τα κατάλληλα ορίσματα για να εξάγουμε το κείμενο τεκμηρίωσης σε μορφή υπερσυνδεδεμένου κειμένου HTML. Η καταγραφή των διαθέσιμων επιλογών για την εξαγωγή υπερβαίνει το εύρος της συγκεκριμένης εργασίας, όμως μπορεί να βρεθεί λεπτομερώς στον επίσημο ιστότοπο [aiafa]. Στην πιό απλή της μορφή η εκτέλεση γίνεται με την εντολή

```
    javadoc path/to/class1 path/to/class2
```

Με κατάλληλη παραμετροποίηση, μπορούμε να παράξουμε αρχείο τεκμηρίωσης κατάλληλο για χρήση με το πρόγραμμα LaTeX, που χρησιμοποιήσαμε για τη συγγραφή του παρόντος. Η παραμετροποίηση αυτή φαίνεται στο αρχείο 32. Παρατηρούμε ότι πέρα από τις βασικές παραμέτρους, ορίζουμε το κατάλληλο `Doclet`, δηλαδή τη μονάδα που θα αναλάβει τη μετάφραση των σχολίων τεκμηρίωσης σε κώδικα LaTeX, ορίζουμε το αρχείο εξόδου, τον τίτλο και συγγραφέα του εγγράφου που θα δημιουργηθεί, όλες τις τοποθεσίες που βρίσκεται ο κώδικας και τέλος το όνομα του πακέτου που τις περιέχει. Αξίζει να αναφέρουμε ότι για μεγάλες εφαρμογές, η εκτέλεση του εργαλείου javadoc μπορεί να γίνει

```
#!/bin/sh

javadoc -docletpath /home/zannis/TeXDoclet.jar \
    -doclet org.stfm.texdoclet.TeXDoclet \
    -noinherited \
    -noindex \
    -output ./documentation.tex \
    -twosided \
    -title "BargainHunt Documentation" \
    -author "Zannis Kalampoukis" \
    -sourcepath /home/zannis/dev/BargainHunt/statistics/src/main/java:
    ↪  /home/zannis/dev/BargainHunt/WebApp/src/main/java \
    -subpackages gr.ntua.cn.zannis.bargains
```

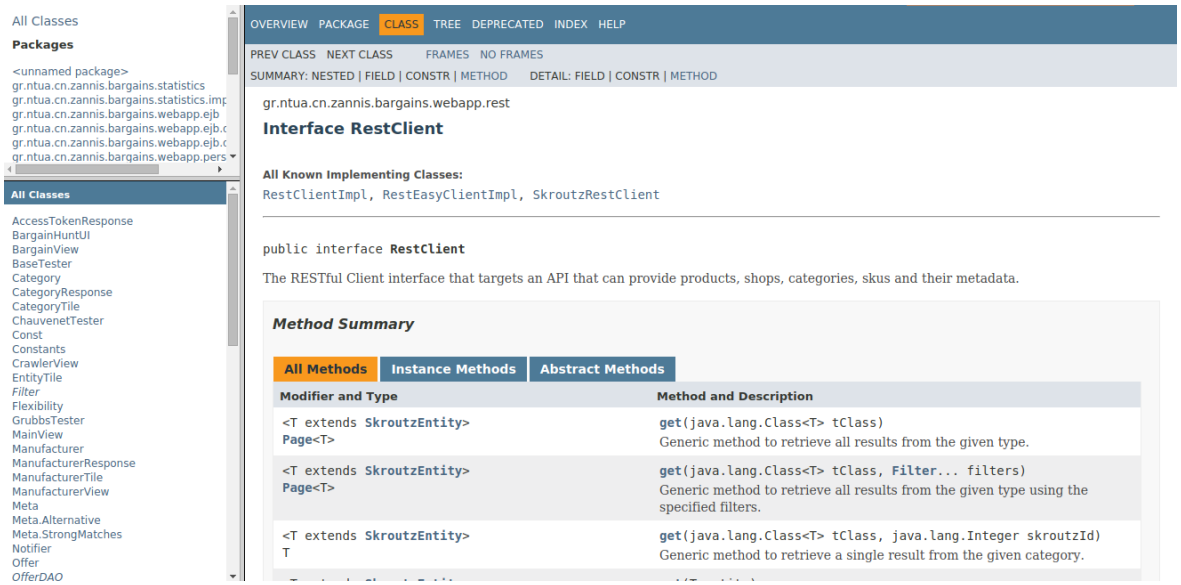
Απόσπασμα 32: Κώδικας για εξαγωγή τεκμηρίωσης σε μορφή LaTeX

πολύπλοκη, όμως για τη διευκόλυνσή μας τα εργαλεία ενοποιημένης ανάπτυξης όπως το Eclipse και το IntelliJ αναλαμβάνουν την αυτοματοποιημένη δημιουργία του κειμένου τεκμηρίωσης μέσω εύχρηστου γραφικού περιβάλλοντος. Συγκεκριμένα, στο Intellij από το μενού `Tools` επιλέγουμε `Generate JavaDoc...` και στο παράθυρο που εμφανίζεται αφού ορίσουμε όλες τις οδηγίες που επιθυμούμε στο σύστημα, πατάμε `Create`. Το παραγόμενο υπερκείμενο σε έναν φυλλομετρητή μοιάζει με αυτό που φαίνεται στη συνέχεια.

Λόγω ορισμένων ιδιαιτεροτήτων της υλοποίησής μας, όπως είναι για παράδειγμα η ανάπτυξη μητρικών κλάσεων και θυγατρικών που υλοποιούν περαιτέρω δυνατότητες, ορίσαμε πολλά μέλη και μεθόδους συναρτήσεων ως **`protected`**. Έτσι, θα πρέπει κατά τη μεταγλώττιση της τεκμηρίωσης, να συμπεριλάβουμε τις προστατευμένες μεθόδους για να έχουμε το πλήρες αποτέλεσμα.

7.2 Σενάρια χρήσης

Στον προγραμματισμό συστημάτων λογισμικού, με τον όρο σενάριο χρήσης ορίζεται ένα σύνολο από ενέργειες ή βήματα που ορίζουν τον τυπικό τρόπο αλληλεπίδρασης ενός χρήστη με το σύστημα. Στο τυποποιημένο πρότυπο UML (Unified Modeling Language) [Inc], ο χρήστης που αναφέραμε ονομάζεται δράστης και μπορεί να είναι είτε άνθρωπος, είτε ένα τρίτο σύστημα είτε και η πάροδος του χρόνου. Ανάλογα με την έκταση, τον τύπο του έργου που υλοποιούμε, αλλά και τον ακροατή



Σχήμα 7.1: Παράδειγμα παραγόμενης τεκμηρίωσης

μας, ένα διάγραμμα σεναρίου χρήσης μπορεί να είναι απολύτως συγκεκριμένο και να αναφέρεται σε μία μεμονωμένη λειτουργία, σε ένα σύνολο λειτουργιών ή ακόμα και σε κάτι πιο αφηρημένο όπως η αποστολή του έργου ή οι στόχοι που πρέπει να επιτευχθούν σε ένα χρονικό διάστημα.

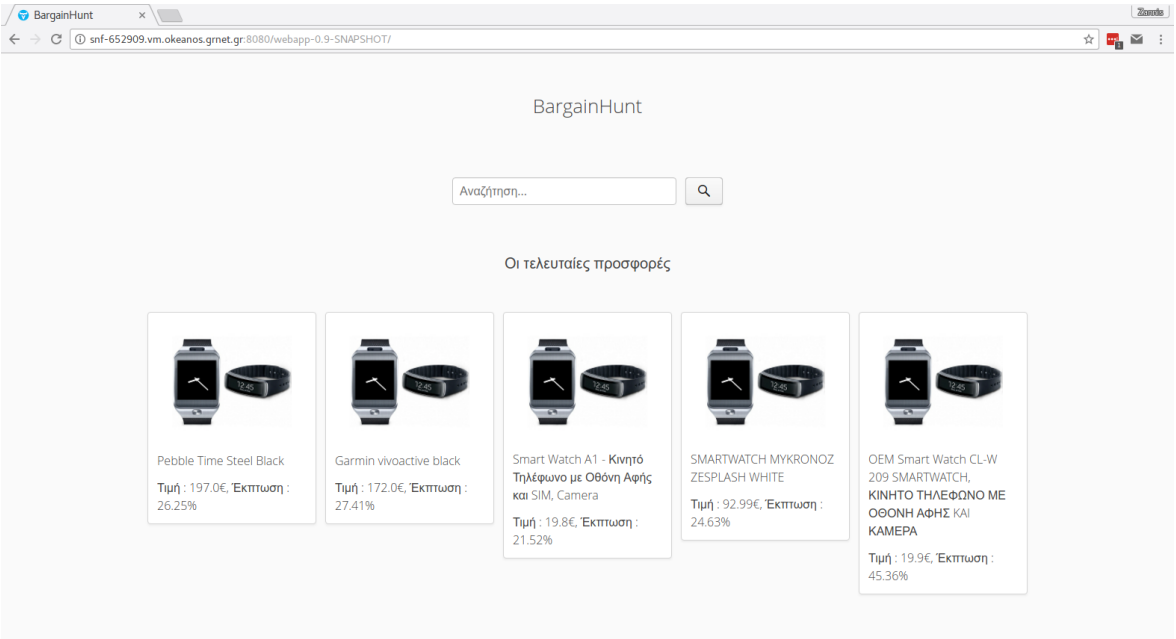
Η ανάλυση σεναρίων χρήσης είναι μία πολύτιμη και σημαντική μέθοδος ανάλυσης απαιτήσεων που χρησιμοποιείται από μηχανικούς υπολογιστών από την επίσημη εμφάνισή της το 1992. Από τότε, πολλές μεγάλες εταιρίες έχουν εντάξει τη χρήση της στη διαδικασία ανάπτυξης λογισμικού που ακολουθούν.

Ένας από τους λόγους για τους οποίους θεωρείται ένα ισχυρό εργαλείο είναι ότι θέτει το χρήστη στο κέντρο της διαδικασίας διατύπωσης απαιτήσεων. Ένα τυπικό παράδειγμα μοντελοποίησης απαιτήσεων ξεκινά με τον καθορισμό των δραστών, οι οποίοι στη δική μας περίπτωση είναι ο χρήστης-υποψήφιος αγοραστής και ο διαχειριστής της εφαρμογής. Σε ένα πραγματικό σενάριο, θα μπορούσαμε να προσθέσουμε ως χρήστες και τον καταστηματάρχη, τον υπεύθυνο υποστήριξης, ή ακόμα και να διαχωρίσουμε τον εγγεγραμμένο χρήστη από τον επισκέπτη.

Ο κάθε ένας από τους χρήστες μας έχει ένα σύνολο από στόχους, ή ενέργειες τις οποίες θέλει να πραγματοποιήσει μέσω της εφαρμογής μας. Αυτοί οι στόχοι θα είναι και οι τίτλοι για τα σενάρια χρήσης που θα ορίσουμε, τα οποία ταυτίζονται και με τις επιθυμητές λειτουργίες του συστήματος. Με τον τρόπο αυτό, εξασφαλίζουμε ότι δίνεται προτεραιότητα στην επιχειρηματική αξία της εφαρμογής και ότι μόνο αυτή αναπτύσσεται με προτεραιότητα, αντίθετα με άλλες δευτερεύουσες λειτουργίες τις οποίες εικάζει ο εκάστοτε προγραμματιστής ή υπεύθυνος.

Επιπλέον επιτρέπεται καλύτερη επικοινωνία μεταξύ των ενδιαφερόμενων πλευρών, διότι με αυτή την τακτική, οι υπηρεσίες περιγράφονται με φυσική γλώσσα, κάτι που οδηγεί στην αποφυγή παρεξηγήσεων ακόμα και σε μη τεχνικά άτομα. Έτσι καταλήγουμε σε ποιοτικές απαιτήσεις, εφαρμόσιμες και γνωστές σε όλους.

Η ευελιξία του συγκεκριμένου τρόπου μοντελοποίησης έγκειται στη δυνατότητα ορισμού του βασικού σεναρίου (happy path), δηλαδή της αναμενόμενης μορφής της αλληλεπίδρασης, αλλά και σε εναλλακτικές μορφές της. Η βήμα-βήμα ανάλυση αυτή, από τις προϋποθέσεις μέχρι την επιτυχία κάθε έκβασης του σεναρίου μπορεί να αποκαλύψει ορισμένες απαιτήσεις που ίσως να μην είχαν γίνει αντιληπτές διαφορετικά. Επιπλέον, αυτή η διαδικασία μπορεί να οδηγήσει σε βελτιστοποιήσεις και στον περιορισμό τον απαραίτητων βημάτων για να έχουμε μία ιδανική εμπειρία χρήσης.



Σχήμα 7.2: Αρχική σελίδα της εφαρμογής

7.3 Τεκμηρίωση και παραδείγματα αλληλεπίδρασης με την εφαρμογή

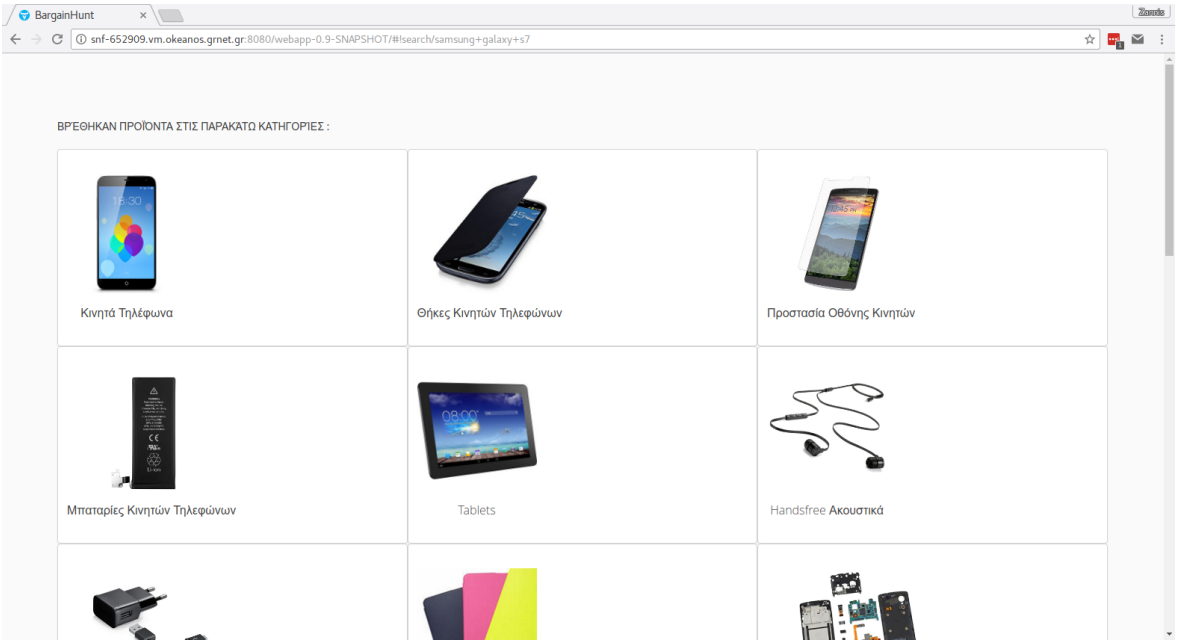
7.3.1 Εύρεση τελευταίων διαθέσιμων προσφορών

Για αυτή τη λειτουργία, ο χρήστης δε χρειάζεται παρά να επισκεφθεί την αρχική σελίδα της εφαρμογής. Κατά την πιλοτική εκτέλεσή της, χρησιμοποιήσαμε ένα απομακρυσμένο ψευδομηχάνημα εξυπηρετητή που ανήκει στο “σύννεφο” του okeanos και έχει διατεθεί στο συγγραφέα για ερευνητική χρήση. Το ψευδομηχάνημα βρίσκεται στην τοποθεσία `http://snf-652909.vm.okeanos.grnet.gr` και η εφαρμογή στο σύνδεσμο `http://snf-652909.vm.okeanos.grnet.gr/webapp-0.9-SNAPSHOT` όπως φαίνεται και στην εικόνα 7.2 όπου απεικονίζεται ένα στιγμιότυπο του φυλλομετρητή Chromium. Η τοποθεσία της εφαρμογής είναι εύκολα παραμετροποιήσημη μέσω του εξυπηρετητή Wildfly. Για τη συγκεκριμένη λειτουργία ο χρήστης χρειάζεται απλά να μεταβεί στην παραπάνω τοποθεσία, όπου βλέπει κατευθείαν τις πέντε πιο πρόσφατες προσφορές που ανακάλυψε η εφαρμογή μέσω της διαδικασίας μαζικού ελέγχου.

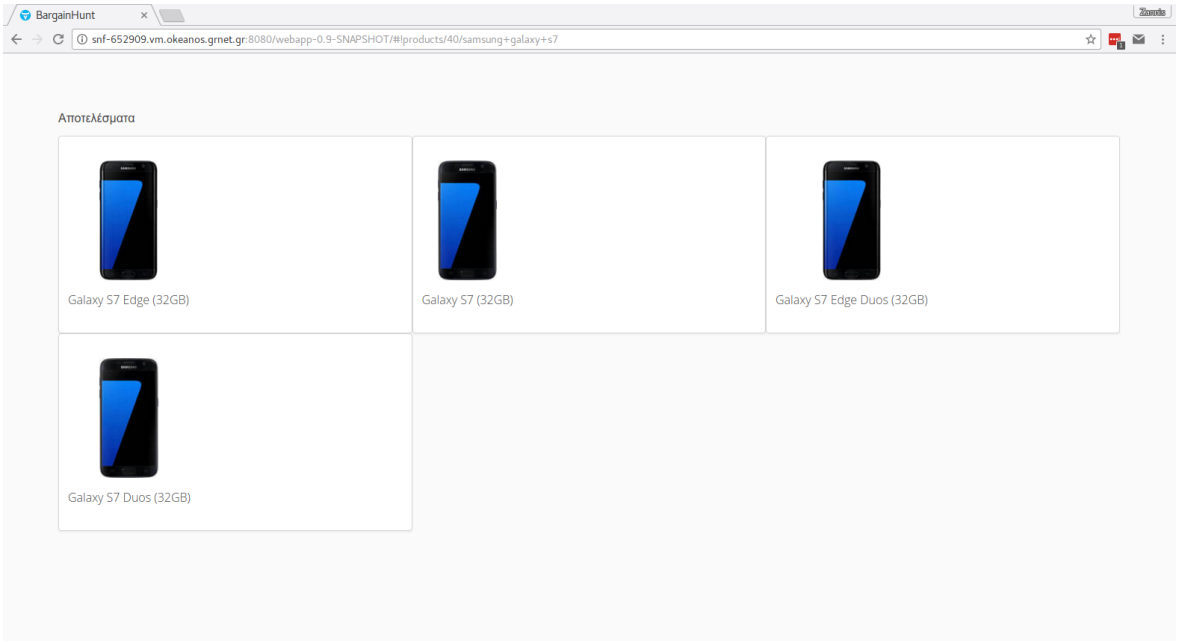
7.3.2 Έλεγχος προσφοράς σε μεμονωμένα προϊόντα

Ομοίως με παραπάνω, κατευθύνουμε το φυλλομετρητή στην αρχική σελίδα και πληκτρολογούμε στο πεδίο αναζήτησης που φαίνεται στο πάνω μέρος, τον όρο αναζήτησης, για παράδειγμα `samsung galaxy s7`. Με το πάτημα του κουμπιού με το σήμα της αναζήτησης, καλείται η λειτουργεία αναζήτησης από το Skroutz API και ο χρήστης ανακατευθύνεται στην πρώτη σελίδα αποτελεσμάτων όπου φαίνονται οι διαθέσιμες κατηγορίες προϊόντων όπως φαίνεται στην εικόνα 7.3.

Εδώ ο χρήστης μπορεί να κάνει συγκεκριμένη την αναζήτηση του επιλέγοντας την κατάλληλη κατηγορία, σε περίπτωση ύπαρξης πάνω από μίας. Το φαινόμενο αυτό είναι αρκετά σύνηθες γιατί αν κάποιος αναζητεί ένα κινητό τηλέφωνο για παράδειγμα, σίγουρα θα υπάρχουν προϊόντα “δορυφόροι”, δηλαδή σχετιζόμενα προϊόντα μικρότερης αξίας όπως θήκες, φορτιστές και άλλα. Αυτό είναι κάτι που υποδηλώνει ο χρήστης καθώς η εφαρμογή δεν έχει την απαραίτητη πληροφορία για να λάβει την απόφαση. Στην περίπτωση που ο όρος αναζήτησης ήταν αρκετά συγκεκριμένος, είναι δυνατό να επιστραφούν κατευθείαν προϊόντα στα αποτελέσματα. Στο γενικό σενάριο, αφού επιλέξει κατηγορία, ο χρήστης συνεχίζει την αναζήτηση του προϊόντος του επιλέγοντάς το από την αντίστοιχη λίστα, σύμφωνα με την εικόνα 7.4. Αφού το επιλέξει, πλοηγείται στη σελίδα όπου ενημερώνεται κατά πόσο



Σχήμα 7.3: Σελίδα διαθέσιμων κατηγοριών στο αντικείμενο αναζήτησης

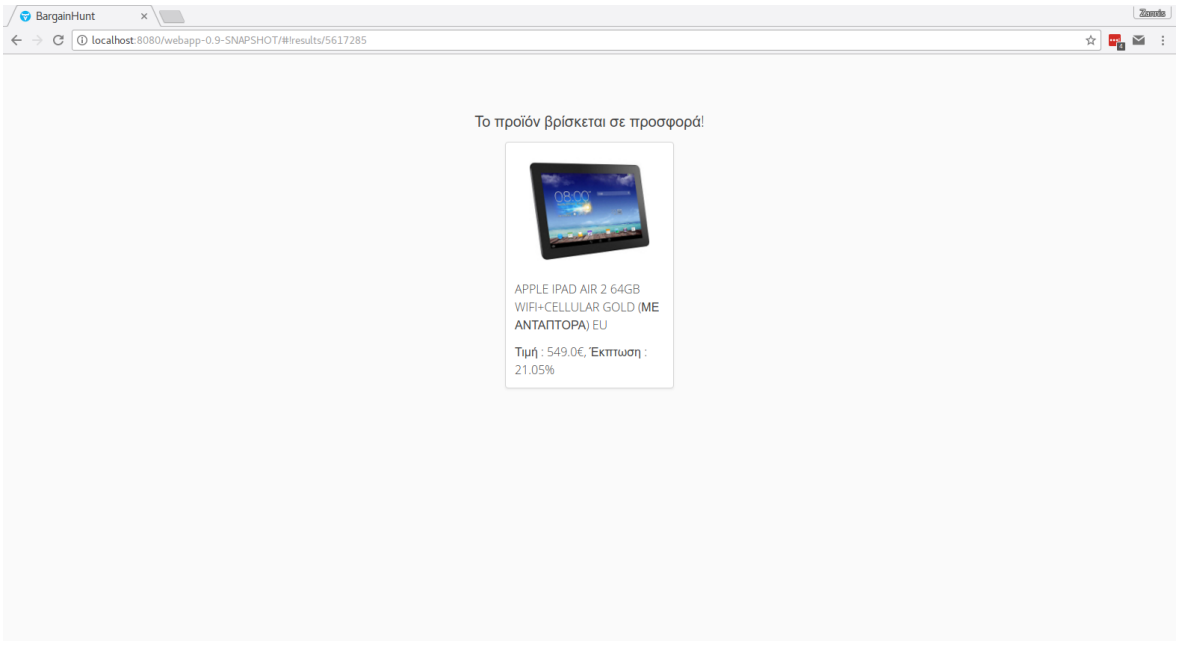


Σχήμα 7.4: Πλοήγηση στα προϊόντα μιας κατηγορίας

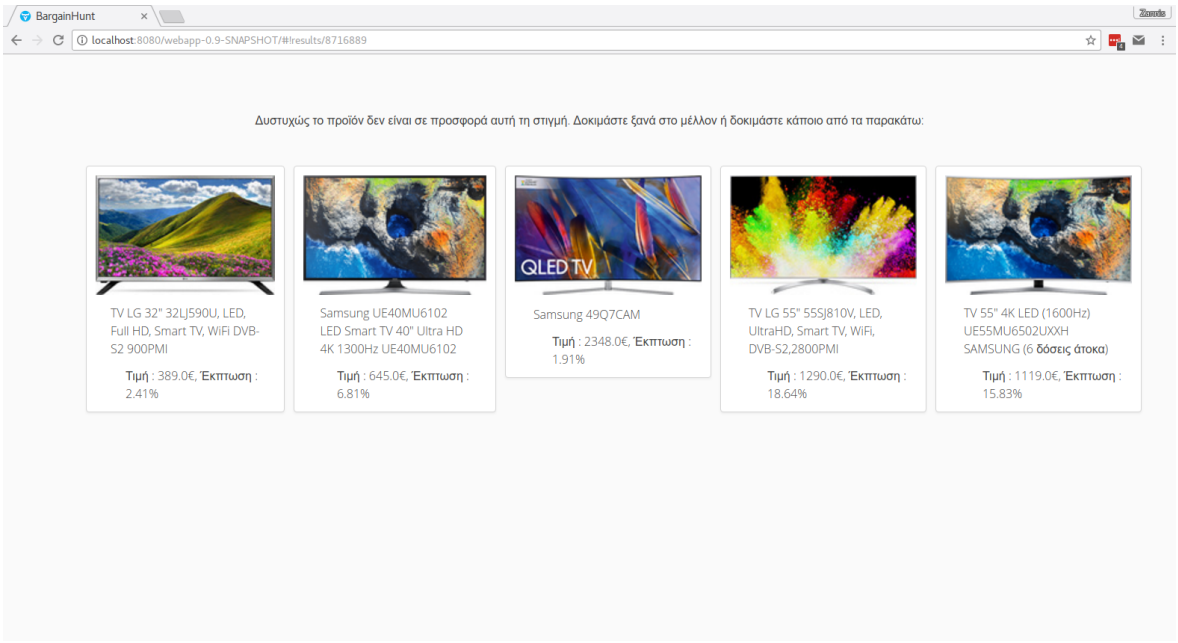
αυτό βρίσκεται σε προσφορά ή όχι όπως βλέπουμε στις εικόνες 7.5 και 7.6. Η μορφοποίηση των σελίδων στην πραγματική εφαρμογή ενδέχεται να διαφέρει από τις εικόνες που αναπαρίστανται εδώ.

7.3.3 Διαχείριση εφαρμογής

Για τη σωστή λειτουργία της εφαρμογής, είναι απαραίτητη η διαχείριση της μέσω της αντίστοιχης διεπαφής. Για το λόγο αυτό αναπτύξαμε τη σελίδα με όνομα `CrawlerView`, η οποία επιτρέπει στο διαχειριστή να κατεβάσει μαζικά ενημερώσεις για υπάρχοντα ή νέα προϊόντα σχετικά με τις τιμές τους και τα καταστήματα πώλησης. Επιπλέον εφαρμόζει μαζικά στατιστικούς ελέγχους σε όλα τα προϊόντα, διατηρώντας τις παρουσιαζόμενες πληροφορίες απολύτως επίκαιρες. Στην εικόνα 7.7



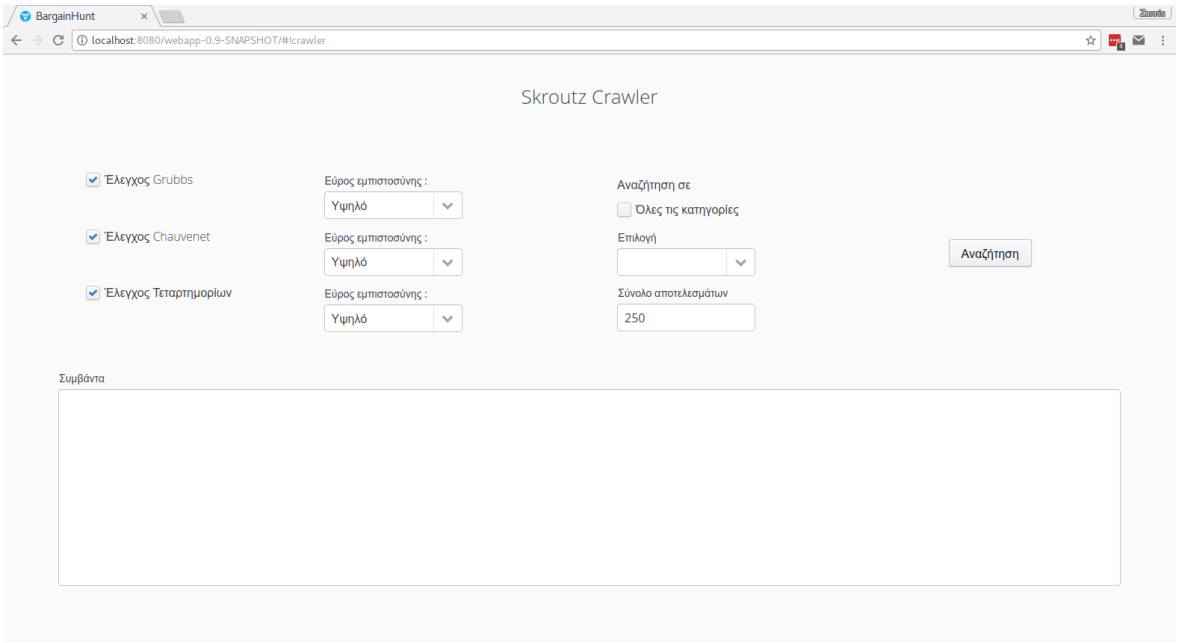
Σχήμα 7.5: Πλοήγηση σε προϊόν που βρίσκεται σε προσφορά



Σχήμα 7.6: Πλοήγηση σε προϊόν που δεν βρίσκεται σε προσφορά

φαίνεται η παρούσα μορφή της οθόνης αυτής.

Το προτεινόμενο διάστημα ενημέρωσης είναι μία φορά την εβδομάδα κατά τις νυχτερινές ώρες. Ο λόγος πίσω από αυτή την απόφαση είναι να αποφύγουμε να επιβαρύνουμε τον τρίτο εξυπηρετητή καθώς για την πλήρη ενημέρωση απαιτείται μεγάλο πλήθος αιτημάτων από την πλευρά μας. Ταυτόχρονα, οι όροι χρήσης του Skroutz API περιορίζουν τα αιτήματα σε 1000 ανά ημέρα, άρα η διαδικασία αυτή ολοκληρώνεται συνήθως σε διαστήματα.



Σχήμα 7.7: Οθόνη διαχείρισης εφαρμογής

Κεφάλαιο 8

Επίλογος

8.1 Ερευνητικό ενδιαφέρον

Ένας από τους βασικούς στόχους της διπλωματικής εργασίας πέραν της υλοποίησης είναι και η διεξαγωγή έρευνας με σκοπό τον εντοπισμό των βέλτιστων αποτελεσμάτων αναφορικά με την εύρεση προσφορών στα δεδομένα που χρησιμοποιήσαμε. Η ερευνητική διαδικασία και τα σχετικά αποτελέσματα παρουσιάζονται στη συνέχεια.

8.2 Ερευνητική διαδικασία

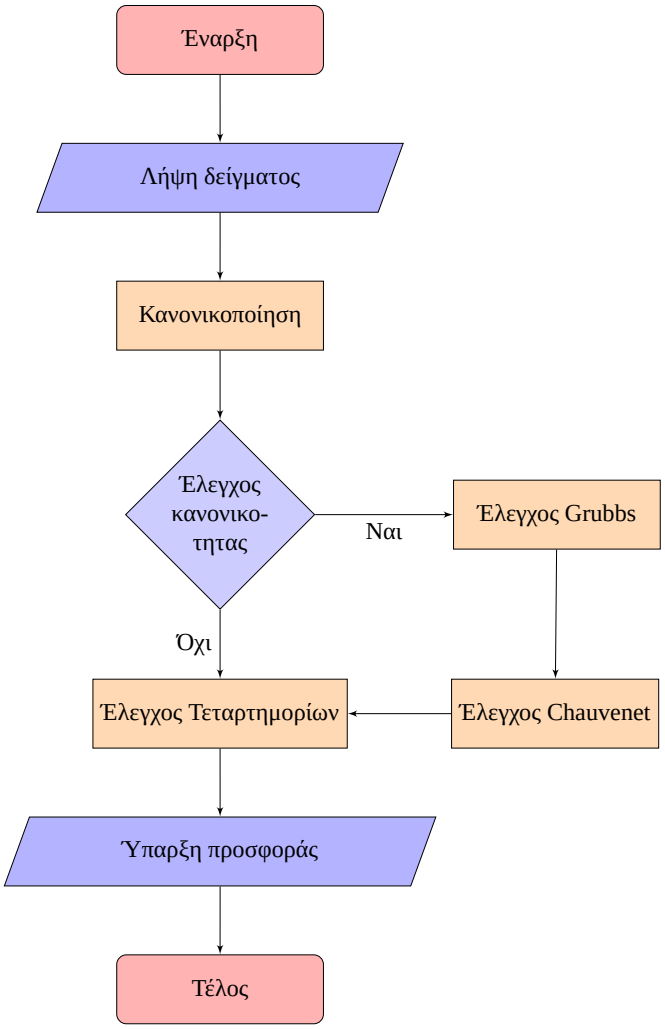
Υπό το πρίσμα του ερευνητή, καλούμαστε να εντοπίσουμε ακρότατα σε μεγάλο πλήθος ετερογενών δειγμάτων. Για το λόγο αυτό, επικαλούμαστε τους στατιστικούς ελέγχους Grubbs, Chauvenet και Τεταρτημορίων, τους οποίους εφαρμόζουμε στα δείγματα. Σε κάθε στάδιο της διαδικασίας επιθυμούμε να τηρούνται οι τυπικοί μαθηματικοί κανόνες και για το λόγο αυτό εφαρμόζουμε μία λογική κατά τη διεξαγωγή των ελέγχων προσφορών, η οποία φαίνεται ως διάγραμμα ροής στο σχήμα 8.1.

Φυσικά, κατά την εφαρμογή των παραπάνω, αντιμετωπίσαμε ορισμένους περιορισμούς αναφορικά με το πλήθος των δεδομένων που χρησιμοποιήσαμε στη διαδικασία. Όπως φαίνεται στο διάγραμμα 8.1, έγινε έλεγχος κανονικότητας της κατανομής των δειγμάτων που χρησιμοποιήσαμε και για να εξυπηρετηθεί ο σκοπός μας, πρέπει τα διαθέσιμα δείγματα να έχουν εύλογο μέγεθος, συνήθως μεγαλύτερο των 6 τιμών. Για το λόγο αυτό περιοριστήκαμε σε συγκεκριμένες κατηγορίες προϊόντων, όπως τα κινητά τηλέφωνα, οι τηλεοράσεις, οι φορητοί υπολογιστές και τα “έξυπνα” ρολόγια, τα οποία έχουν πολύ συγκεκριμένα μοντέλα και πωλούνται σε πλήθος καταστημάτων άρα επιτρέπουν την εφαρμογή και των τριών επιλεγμένων στατιστικών ελέγχων. Ταυτόχρονα, στις προαναφερθείσες κατηγορίες υπάρχει μεγάλο εύρος προϊόντων, άρα θα μπορούσαμε να εξετάσουμε την ορθότητα των ελέγχων με μεγάλο βαθμό εμπιστοσύνης.

8.3 Σύνοψη

Στην παρούσα διπλωματική εργασία αναπτύχθηκε ένας ιστότοπος που παρέχει με μία απλή και εύχρηστη διεπαφή την ευκαιρία στο χρήστη να ανακαλύψει μέσω αναζήτησης ή αυτόματων προτάσεων, τις καλύτερες τρέχουσες προσφορές σε προϊόντα από ηλεκτρονικά καταστήματα που συμμετέχουν στον ιστότοπο συλλέκτη Skroutz.gr. Το συγκεκριμένο υποκεφάλαιο περιέχει μία ανασκόπηση της παρούσας εργασίας, εστιάζοντας στη χρηστική αξία που διαθέτει αυτή τη στιγμή και στη συνέχεια θα μιλήσουμε για πιθανές μελλοντικές της επεκτάσεις και τον τρόπο με τον οποίο αυτές θα μπορούσαν να ενταχθούν στην υπηρεσία.

Αρχικά, είναι σημαντικό να κατανοήσουμε ότι η εφαρμογή αυτή αναπτύχθηκε ως μία αυτοτελής υπηρεσία στα πλαίσια της ερευνητικής κατεύθυνσης της διατριβής. Σε πραγματικές συνθήκες, ένα τέτοιο εγχείρημα θα αποτελούσε μία μονάδα ενός μεγαλύτερου έργου με σκοπό να προσφέρει προστιθέμενη αξία στον καταναλωτή. Εφόσον όμως τα δεδομένα μας παρέχονται από τρίτη πηγή εμπορικού ενδιαφέροντος, δεν είχαμε άλλη επιλογή από το να δημιουργήσουμε μία ιστοσελίδα που θα εξυπηρετεί τη λειτουργικότητα που δημιουργήσαμε. Στα πλαίσια της έρευνας και ανάπτυξης ήρθαμε σε επαφή



Σχήμα 8.1: Διάγραμμα ροής στατιστικών ελέγχων

με πλήθος από μοντέρνες τεχνολογίες, χρησιμοποιήσαμε ένα σύστημα ελέγχου εκδόσεων πηγαίου κώδικα, εφαρμόσαμε τις πλέον ενδεδειγμένες τεχνικές ανάπτυξης λογισμικού εφόσον μας δόθηκε η ευκαιρία να διαμορφώσουμε από την αρχή το τελικό αποτέλεσμα. Έτσι εμβαθύναμε τις γνώσεις μας γύρω από την ανάπτυξη δυναμικών, αποκρίσιμων και ασφαλών υπηρεσιών που λειτουργούν άψογα σε κάθε τύπο συσκευής, συμβαδίζοντας με τις γνωσιακές απαιτήσεις κάθε μηχανικού υπολογιστών τη σύγχρονη εποχή.

Αναφορικά με το ερευνητικό μέρος της εργασίας, δηλαδή τον στατιστικό έλεγχο δειγμάτων για την εύρεση ακραίων τιμών, υλοποιήσαμε προγραμματιστικά τις τρεις τεχνικές που αποφασίσουμε να συγκρίνουμε. Μετά το δοκιμαστικό έλεγχο της λειτουργίας τους, τις εντάξαμε στο σύστημα, με σκοπό να εκτελούνται παράλληλα κατά τη λήψη ενημερωμένων πληροφοριών για ένα προϊόν. Με αυτό τον τρόπο, ενισχύεται ο βαθμός εμπιστοσύνης για την ορθότητα των προβαλλόμενων αποτελεσμάτων.

Το παρόν έργο δε θα ήταν δυνατό να ολοκληρωθεί χωρίς εργαλεία και βιβλιοθήκες ανοιχτού λογισμικού, τα οποία και χρησιμοποιήσαμε σχεδόν σε αποκλειστικότητα. Χάρη στην παγκόσμια κοινότητα προγραμματιστών ανοιχτού κώδικα, μπορέσαμε να κερδίσουμε πολύτιμο χρόνο χωρίς να ανησυχούμε για θέματα δικτύωσης ή ανάπτυξης πρωτοκόλλων για τα οποία δεν είμαστε σε θέση βαθιάς κατανόησης, αλλά να εστιάσουμε αποκλειστικά στις λειτουργικές απαιτήσεις που ορίσαμε. Επιπλέον, υιοθετώντας μία ευέλικτη λογική ανάπτυξης, αποτελούμενη από μικρούς και γρήγορα επιτεύξιμους στόχους αλλά και πολλούς κύκλους ανατροφοδότησης, καταφέραμε να επιταχύνουμε αισθητά τη διαδικασία ανάπτυξης. Έγινε μεγάλη προσπάθεια για την όσο το δυνατόν πιο πλήρη τεκμηρίωση του κώδικα, σε μορφή που υποστηρίζει το εργαλείο `javadoc`, έτσι ώστε να μπορεί να γίνει άμεση μετατροπή

μέσω του ομώνυμου εργαλείου.

Το αποτέλεσμα είναι μία όμορφη, πλήρης και επαγγελματικού επιπέδου εφαρμογή, η οποία με ελάχιστη προσπάθεια μπορεί να παραμετροποιηθεί κατάλληλα έτσι ώστε να λάβει δεδομένα από νέες πηγές και να τα εμφανίσει στην ίδια διεπαφή, ελαχιστοποιώντας την απαιτούμενη προσπάθεια από τον προγραμματιστή.

8.4 Μελλοντικές επεκτάσεις

Κάτι που μας απασχόλησε σε κάθε στάδιο της εκπόνησης ήταν το εύρος των λειτουργιών που θα διαθέτει η εφαρμογή. Συνεκτιμώντας τις ευκαιρίες αλλά και τους περιορισμούς που συναντήσαμε, αποφασίσαμε να καταλήξουμε σε ένα σύνολο από χαρακτηριστικά που κρίναμε απαραίτητα για την πιλοτική λειτουργία της και αφήσαμε ορισμένες από τις ιδέες μας για υλοποίηση μελλοντικά, με τη μορφή επεκτάσεων.

Συγκεκριμένα, η μονάδα αναζήτησης, αν και λειτουργική στη βασική μορφή της, θα μπορούσε να επωφεληθεί αρκετά από τον εμπλουτισμό της με λειτουργίες φιλτραρίσματος και ταξινόμησης. Οι συγκεκριμένες υποστηρίζονται από την υπηρεσία του Skroutz και χρησιμοποιούνται στην εφαρμογή μας, με προκαθορισμένο τρόπο όμως και κρυφά από το χρήστη. Μία πρώτη σκέψη είναι ο σχεδιασμός μίας διεπαφής που να καθιστά εμφανή τον τρόπο ταξινόμησης των αποτελεσμάτων αναζήτησης και να επιτρέπει στο χρήστη να τα φιλτράρει με βάση κάποια προτεινόμενα κριτήρια. Εναλλακτικά, στην αρχική οθόνη, όπου εμφανίζονται οι πέντε πιο πρόσφατες προσφορές, θα μπορούσαμε να κάνουμε αυτό το μέγεθος μεταβλητό ώστε να μπορεί ο χρήστης να πλοηγηθεί σε περισσότερες πρόσφατες προσφορές έτσι ώστε να ανακαλύψει περισσότερα προϊόντα. Η αλλαγή αυτή είναι εύκολη από θέμα υλοποίησης αλλα δεν αποδίδει την απαιτούμενη αξία έτσι ώστε να γίνει προτεραιότητα στα πλαίσια του παρόντος.

Βλέποντας την εφαρμογή ανασκοπικά, και λαμβάνοντας υπόψη τις αποφάσεις που λάβαμε ως προς την πηγή δεδομένων αλλά και την αποθήκευσή τους από εμάς, ενδεχομένως να συνέφερε η υιοθέτηση μίας μη σχεσιακής λύσης όσων αφορά στα δεδομένα που λαμβάνουμε τους τρίτους παρόχους. Ο λόγος είναι ότι στην παρούσα κατάσταση αναγκαζόμαστε να μοντελοποιήσουμε τα ετερογενή δεδομένα, σε μορφή κατάλληλη για επεξεργασία από εμάς και να τα μετατρέψουμε σε δικά μας αντικείμενα, μία διαδικασία η οποία επαναλαμβάνεται με την αντίστροφη σειρά όταν επιθυμούμε να τα μεταδώσουμε. Η διαδικασία αυτή προσθέτει ένα βαθμό πολυπλοκότητας και δυσκολία στην επίβλεψη του έργου σε περίπτωση αλλαγών. Με τη χρήση ενός συστήματος αποθήκευσης χωρίς κάποιο αυστηρό σχήμα, δηλαδή εργαλεία τύπου `NoSQL` [1], θα μπορούσαμε να αποφύγουμε τις περισσότερες μετατροπές δεδομένων και η εφαρμογή θα έχανε πολλές από τις εξαρτήσεις που έχει αυτή τη στιγμή. Ως αποτέλεσμα, θα έχουμε ένα πραγματικά αρθρωτό και επεκτάσιμο σύστημα.

Η μεταγλώττιση της εφαρμογής στην αγγλική γλώσσα ήταν κάτι που επιθυμούσαμε εξαρχής. Για να επιτευχθεί αυτό χρειάζεται τον κατάλληλο προσχεδιασμό διότι χρησιμοποιείται ένα σύστημα κατά το οποίο οι συμβολοσειρές αποτιμώνται διαφορετικά ανάλογα με την επιλεγμένη γλώσσα. Αν δε γίνουν οι απαραίτητες ενέργειες πριν τη έναρξη της ανάπτυξης, είναι περίπλοκο να εντάξουμε την πολυγλωσσία σε δεύτερο στάδιο, όμως το πλαίσιο ανάπτυξης Vaadin έχει προνοήσει διευκολύνοντας τη συγκεκριμένη διαδικασία και για το λόγο αυτό την συμπεριλαμβάνουμε στις μελλοντικές επεκτάσεις.

Μια ακόμα σκέψη για μελλοντική επέκταση είναι η αυτοματοποίηση της διαδικασίας ενημέρωσης του συστήματος με δεδομένα. Αυτή τη στιγμή υπάρχει η κατάλληλη διεπαφή, όμως απαιτεί την παρέμβαση του χρήστη για την επιλογή των επιθυμητών κατηγοριών. Μελλοντικά, θα ήταν χρήσιμο αυτή η διαδικασία να ενεργοποιείται αυτόματα στον εξυπηρετητή, σε δεδομένο χρονικό διάστημα μέσω κάποιου εργαλείου αυτοματοποίησης του λειτουργικού συστήματος. Κάτι τέτοιο είναι εύκολο να γίνει με χρήση εργαλείων αυτόματης εκτέλεσης εφαρμογών όπως το `cron` σε περιβάλλον UNIX.

Τέλος, ένα σύστημα εύρεσης προσφορών μπορεί να συνδυαστεί άριστα με τη δυνατότητα εγγραφής σε ένα μηχανισμό αυτόματης αποστολής ειδοποιήσεων. Αυτό προϋποθέτει την υιοθέτηση ενός

[1] Η νοοτροπία του NoSQL αναπτύσσεται στο Παράρτημα Α

συστήματος ταυτοποίησης, όπου ο χρήστης θα συνδέεται με τα προσωπικά του στοιχεία και θα διατηρεί ένα ευχολόγιο (wishlist) στο οποίο θα απαριθμεί τα προϊόντα που επιθυμεί να αγοράσει σε περίπτωση που βρεθούν σε προσφορά. Θα μπορούσε για παράδειγμα να επιλέγει ένα προϊόν για το οποίο να δηλώνει το ενδιαφέρον του, και τότε το σύστημα σε κάθε πτώση της τιμής του να τον ενημερώνει άμεσα για την αλλαγή. Σε σχέση με τις προηγούμενες βελτιώσεις, αυτή θα είχε τη μεγαλύτερη εμπορική αξία για τον καταναλωτή και θα ήταν μία από τις υψηλότερες προτεραιότητες σε περίπτωση μελλοντικής εκμετάλλευσης του παρόντος από το δημιουργό.

Μετά τον εμπλουτισμό της εργασίας με τα παραπάνω στοιχεία, θα μπορούσαμε να τη διαθέσουμε σε δημόσια χρήση. Ο στόχος μας όμως δεν είναι η εμπορευματοποίησή της, αλλά η απόδειξη ότι μία τέτοια υπηρεσία είναι εφικτή με τις υπάρχουσες τεχνολογίες. Σε αυτό το πλαίσιο θεωρούμε ότι στην παρούσα κατάσταση, το σύστημα είναι ολοκληρωμένο.

Βιβλιογραφία

[aiafa] Oracle Corporation and/or its affiliates, "Javadoc 1.4 Microsoft Windows Examples", . (Visited on 01/03/2016).

[aiafb] Oracle Corporation and/or its affiliates, "Javadoc Tool Home Page", . (Visited on 01/03/2016).

[aiafc] Oracle Corporation and/or its affiliates, "JSR-000338 Java Persistence 2.1 Final Release for Evaluation", . (Visited on 12/15/2015).

[aiafd] Oracle Corporation and/or its affiliates, "Trail: Java Naming and Directory Interface (The Java™ Tutorials)", . (Visited on 12/17/2015).

[Catl06] Hampton Catlin, Nathan Weizenbaum and Chris Eppstein, "SASS: Syntactically Awesome Style Sheets", 2006.

[Drag16] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch-Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin and Larisa Safina, "Microservices: yesterday, today, and tomorrow", *CoRR*, vol. abs/1606.04036, 2016.

[Fiel00] Roy Fielding, "Representational state transfer", *Architectural Styles and the Design of Netowork-based Software Architecture*, pp. 76–85, 2000.

[Flan06] David Flanagan, *JavaScript: the definitive guide, "* O'Reilly Media, Inc.", 2006.

[Foun15a] The Apache Software Foundation, "Apache License, Version 2.0", 2015. [Online; accessed 2-November-2015].

[Foun15b] The Apache Software Foundation, "Maven – Welcome to Apache Maven", 2015. [Online; accessed 2-November-2015].

[Foun15c] The Eclipse Foundation, "EclipseLink Downloads", , 2015. [Online; accessed 3-November-2015].

[Grub69] Frank E Grubbs, "Procedures for detecting outlying observations in samples", *Technometrics*, vol. 11, no. 1, pp. 1–21, 1969.

[Han98] Jun Han, "A comprehensive interface definition framework for software components", in *Software Engineering Conference, 1998. Proceedings. 1998 Asia Pacific*, pp. 110–117, IEEE, 1998.

[href15] @frodriguez, "Maven Repository: Search/Browse/Explore", 2015. [Online; accessed 2-November-2015].

[Hurs97] Jani Hursti, "Single sign-on", in *Proc. Helsinki Uiniversity of Technology Seminar on Network Security*, 1997.

[Inc] Object Management Group Inc., "Unified Modeling Language (UML)", . (Visited on 01/13/2016).

[Kraf05] Dirk Krafzig, Karl Banke and Dirk Slama, *Enterprise SOA: service-oriented architecture best practices*, Prentice Hall Professional, 2005.

[Ltd15] Vaadin Ltd., "Vaadin — User Interface Components for business apps — vaadin.com", , 2015. (Visited on 11/02/2016).

[Mull12] Bernd Müller and Harald Wehr, *Java Persistence API 2: Hibernate, EclipseLink, OpenJPA und Erweiterungen*, Carl Hanser Verlag GmbH Co KG, 2012.

[Nobl95] Brian Noble, Morgan Price and Mahadev Satyanarayanan, "A programming interface for application-aware adaptation in mobile computing", *Computing Systems*, vol. 8, no. 4, pp. 345–363, 1995.

[Over17] Stack Overflow, "Stack Overflow Developer Survey Results 2017", 2017. [Online; accessed 30-Μαρτίου-2017].

[Perr03] Randall Perrey and Mark Lycett, "Service-oriented architecture", in *Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium on*, pp. 116–119, IEEE, 2003.

[RobJ96] Yanan Fan Rob J. Hyndman, "Sample Quantiles in Statistical Packages", *The American Statistician*, vol. 50, no. 4, pp. 361–365, 1996.

[SA] GRNET SA., "Home | ~okeanos IAAS", . (Visited on 01/15/2016).

[Send14] Jakub Sendor, Yann Lehmann, Gabriel Serme and Anderson Santana de Oliveira, "Platform-level support for authorization in cloud services with OAuth 2", in *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pp. 458–465, IEEE, 2014.

[Skro] Skroutz.gr, "Overview | API v3", . (Visited on 11/16/2015).

[Wiki15a] Wikipedia, "Brendan Eich — Wikipedia, The Free Encyclopedia", 2015. [Online; accessed 2-November-2015].

[Wiki15b] Wikipedia, "Hampton Catlin — Wikipedia, The Free Encyclopedia", 2015. [Online; accessed 2-November-2015].

[Wiki15c] Wikipedia, "Junio Hamano — Wikipedia, The Free Encyclopedia", 2015. [Online; accessed 15-January-2016].

[Wiki16] Wikipedia, "Linus Torvalds — Wikipedia, The Free Encyclopedia", 2016. [Online; accessed 15-January-2016].

[Βι15] Βικιπαίδεια, "Plug-in — Βικιπαίδεια, Η Ελεύθερη Εγκυκλοπαίδεια", 2015. [Online; accessed 2-November-2015].

Παράρτημα Α

Ευρετήριο τεχνικών όρων

Διεπαφή χρήστη [Han98] τυπικά είναι κάθε μορφής τεχνολογία που χρησιμοποιείται στα πλαίσια της αλληλεπίδρασης ενός ανθρώπου με ένα μηχάνημα. Ιστορικά, το πλέον διαδεδομένο μέσο ήταν ο συνδυασμός οθόνης και πληκτρολογίου όμως με την πάροδο του χρόνου και την ιλιγγιώδη πρόοδο της τεχνολογίας, αναπτύχθηκαν τα γραφικά περιβάλλοντα, οι οθόνες αφής, τα τηλεχειριστήρια και πολλές άλλες επινοήσεις που διατέθηκαν στο χρήστη.

Με τον όρο **Διεπαφή Προγραμματισμού Εφαρμογών** [Nobl95] αναφερόμαστε σε ένα σύνολο συναρτήσεων, διαδικασιών και εργαλείων που διατίθενται από την ομάδα ανάπτυξης ενός λογισμικού με στόχο την ανάπτυξη τρίτων λογισμικών που θα αλληλεπιδρούν με το αρχικό. Η Δ.Π.Ε. διασφαλίζει τον τρόπο που τα μεμονωμένα προγραμματιστικά στοιχεία θα επικοινωνούν μεταξύ τους και χρησιμοποιείται κατά κόρον στην ανάπτυξη εφαρμογών λογισμικού.

Εναποθετήριο πηγαίου κώδικα είναι ένα σύνολο αρχείων στον υπολογιστή που περιέχει όλες τις απαραίτητες πληροφορίες για την ανάπτυξη, ενημέρωση, διαχείριση και διατήρηση ιστορικού εφαρμογών λογισμικού από μεμονομένα άτομα αλλά κυρίως από ομάδες. Σαν έννοια εμφανίστηκε αρκετά νωρίς, απο τα μέσα της δεκαετίας του ́80 με το Revision Control System και αργότερα με το Concurrent Versioning System (CVS) που έγινε και η πρώτη ευρέως διαδεδομένη διανομή ελέγχου ιστορικού. Στη συνέχεια, αφού διάδόθηκε η αξία της υιοθέτησης τέτοιων συστημάτων, αναπτύχθηκε πληθώρα από αυτά, με πιο σημαντικά το Subversion και το git. Το τελευταίο θα χρησιμοποιήσουμε και στα πλαίσια της παρούσας εργασίας.

Στατιστικός έλεγχος είναι μία μέθοδος εξαγωγής συμπερασμάτων από συλλογές τιμών. Κατά κανόνα, συγκρίνονται δύο δείγματα ως προς κάποιο στατιστικό τους μέγεθος ή σε άλλες περιπτώσεις ελέγχεται ένα δείγμα σε σχέση με μια προκαθορισμένη κατανομή. Κατά κανόνα, προηγείται μία υπόθεση όσον αφορά στο δείγμα, η οποία ενεργεί ως συνδετικός δεσμός μεταξύ των δύο δειγμάτων, αντίθετή στην κενή υπόθεση, ότι δηλαδή τα δύο δείγματα δεν έχουν κοινά χαρακτηριστικά. Κάθε στατιστικός έλεγχος έχει ορισμένα προαπαιτούμενα, όπως το μέγεθος του δείγματος και το είδος κατανομής που ακολουθεί. Για το λόγο αυτό είναι πολύ σημαντικό να επιλέξουμε το κατάλληλο δείγμα σε συνδυασμό με τον αντίστοιχο στατιστικό έλεγχο για να αποκτήσουμε όσο το δυνατόν πιο ασφαλή και ακέραια αποτελέσματα.

Ενοποιημένο περιβάλλον ανάπτυξης είναι μία σουίτα λογισμικού, στοχευμένη στο να παρέχει όλα τα απαραίτητα εργαλεία που χρειάζεται ένας προγραμματιστής. Συνήθως περιέχουν γραφικό περιβάλλον και λειτουργούν ως ένα πακέτο που επιτρέπει την εύκολη οργάνωση, διαχείριση, ανάπτυξη και εκτέλεση μιας εφαρμογής. Συνήθως, κάθε ενοποιημένο περιβάλλον ανάπτυξης εξειδικεύεται σε μία γλώσσα ή σε μία οικογένεια γλωσσών προγραμματισμού, επιτρέποντας την ανάπτυξη υβριδικών εφαρμογών από μεμονωμένα άτομα ή ομάδες σε μία κοινά προσβάσιμη πλατφόρμα. Κοινά χαρακτηριστικά στις διάφορες υλοποιήσεις περιβάλλοντων ανάπτυξης είναι η ενσωμάτωση μονάδων όπως ο αποσφαλματωτής (debugger), ο επεξεργαστής κώδικα (code editor), ο εξερευνητής αρχείων (file manager) καθώς και ενσωμάτωση διάφορων Διεπαφών Ανάπτυξης, εναποθετηρίων κώδικα κ.α. Την ώρα συγγραφής του κειμένου, τα πέντε πιο δημοφιλή περιβάλλοντα ανάπτυξης δικτυακών εφαρμογών σύμφωνα με έρευνα του Stack Overflow σε δείγμα 64000 προγραμματιστών είναι Microsoft Visual Studio, Notepad++, Sublime Text, Vim και Visual Studio Code [Over17]. **Representational State Transfer (REST)** είναι μία τεχνοτροπία υλοποίησης ΔΠΕ που χρησιμοποιείται κατά κόρον από δικτυακές εφαρμογές και στηρίζεται στη χρήση του πρωτοκόλλου HTTP και τη σημασιολογική χρήση

των “ρημάτων” (GET, PUT, POST, DELETE) του στις λειτουργίες τους όσον αφορά στα δεδομένα που διαχειρίζονται. Μία διεπαφή τύπου REST κατά κανόνα δε πρέπει να βρίσκεται σε μία κατάσταση, αλλά να εξυπηρετεί τα μηνύματα που λαμβάνει ανεξαρτήτως ιστορικού.

Σειριοποίηση και **Αποσειριοποίηση** είναι η διαδικασία μετατροπής αντικειμένων απο δυαδική κωδικοποίηση σε μορφή κειμένου και αντίστροφα. Η σειριοποίηση σαν τεχνική χρησιμοποιείται κατεξοχήν στην αποθήκευση δεδομένων σε σχεσιακές και μη βάσεις δεδομένων καθώς και στην επικοινωνία μεταξύ συστημάτων, ιδιαίτερα μέσω δικτύου. Τυπικές μορφές σειριοποίησης κειμένου είναι οι γλώσσες XML και JSON.

DOM (ή Document Object Model) είναι το όνομα του Αντικειμενοστραφούς Μοντέλου Εγγράφων. Έχει οριστεί από επίσημους οργανισμούς, όπως το W3C (World Wide Web Consortium) σαν τον επίσημο τρόπο αναπαράστασης HTML και XML εγγράφων. Δίνει μιά ιεραρχική και εύκολα προσπελάσιμη δομή στα έγγραφα που περιγράφει. Οι διάφοροι κόμβοι που αποτελούν ένα τέτοιο έγγραφο, μπορούν να ονομαστούν με βάση κάποια κλάσση ή κάποιο μοναδικό αναγνωριστικό και στη συνέχεια να εφαρμοστούν πάνω τους μέθοδοι από σεναριακό κώδικα Javascript ή κάτι αντίστοιχο. Ουσιαστικά επιτρέπει τον πλήρη έλεγχο μορφοποίησης και λειτουργικότητας στα παραπάνω. Το DOM είναι αυτό που επιτρέπει σε σεναριακές γλώσσες όπως η JavaScript να αποκτήσουν γνώση για τις ιστοσελίδες που συνοδεύουν και να επιδράσουν πάνω τους.

Ελεγχοστραφής Προγραμματισμός είναι μία σχετικά νέα μορφή διαχείρισης μεγάλων πρότζεκτ λογισμικού, όπου υπάρχουν πολλές υπηρεσίες, μονάδες και κόμβοι που αλληλεπιδρούν μεταξύ τους και επιτάσσει τη συγγραφή κλάσσεων ελέγχου και την εκτέλεσή τους κατά την ώρα μεταγλώτισης έτσι ώστε να εξακριβωθεί η ορθή λειτουργία κρίσιμης λογικής. Συνήθως, κάθε φορά που ορίζονται νέες προδιαγραφές για ένα λογισμικό, μεταφράζονται σε “σενάρια χρήσης” και συγγράφονται οι αντίστοιχες κλάσσεις, οι οποίες εκτελούν στοχευμένα κομμάτια κώδικα με σκοπό να εξακριβωθεί η ακρίβεια της υλοποίησης. Ο κύκλος ζωής ενός πρότζεκτ που υλοποιεί τη συγκεκριμένη πρακτική ξεκινά από τη δημιουργία ενός νέου ελέγχου και την προσθήκη του στη σουίτα ελέγχων. Ακολουθεί η απαραίτητη ανάπτυξη και κατά την ολοκλήρωσή της εκτελείται η πλήρης σουίτα ελέγχων. Εαν το αποτέλεσμα είναι επιτυχές, τότε θεωρούμε ότι η υλοποίηση ανταποκρίνεται στις απαιτήσεις και κλείνει ο κύκλος. Διαφορετικά, εξακριβώνουμε από τους ελέγχους που απέτυχαν, που οφείλεται η απόκλιση και εφαρμόζουμε γοργούς κύκλους ανάπτυξης μέχρι να λυθεί.

NoSQL ονομάζεται η οικογένεια τεχνολογιών που επιτρέπουν τη μόνιμη αποθήκευση και εφαρμογή ερωτημάτων σε δεδομένα χωρίς χρήση του σχεσιακού μοντέλου. Βάσεις δεδομένων τύπου NoSQL εμφανίζονται κυρίως σε αρχιτεκτονικές εξυπηρετητή-καταναλωτή αλλά κάτι τέτοιο δεν είναι απαραίτητο. Συνήθως χρησιμοποιούν δομές κειμένου για αποθήκευση των δεδομένων, με γνωστότερη τη μορφή JSON. Χρησιμοποιείται σε εφαρμογές όπου τα δεδομένα δεν έχουν περίπλοκες σχέσεις μεταξύ τους και επιτρέπει την εύκολη “οριζόντια” βελτίωση της απόδοσης μέσω προσθήκης νέων μηχανημάτων που επικοινωνούν μεταξύ τους και διατηρούν την ομοιομορφία των δεδομένων. Ένα κρίσιμο θέμα, δεδομένης της νεότητας της συγκεκριμένης τεχνολογίας είναι η δημιουργία ενός ορθού μοντέλου, όπου η πληροφορία δε θα επαναλαμβάνεται, αλλά θα εξυπηρετούνται όλες οι απαιτήσεις του χρήστη με εύρωστο τρόπο.

Παράρτημα Β

Πηγαίος κώδικας δημιουργίας βάσης δεδομένων

```
/* create products table sequence */
CREATE SEQUENCE public.prod_seq START 1 INCREMENT 1 NO MAXVALUE CACHE 1;
/* create shops table sequence */
CREATE SEQUENCE public.shop_seq START 1 INCREMENT 1 NO MAXVALUE CACHE 1;
/* create categories table sequence */
CREATE SEQUENCE public.category_seq START 1 INCREMENT 1 NO MAXVALUE CACHE
 ↪  1;
/* create prices table sequence */
CREATE SEQUENCE public.price_seq START 1 INCREMENT 1 NO MAXVALUE CACHE 1;
/* create skus table sequence */
CREATE SEQUENCE public.sku_seq START 1 INCREMENT 1 NO MAXVALUE CACHE 1;
/* create manufacturers table sequence */
CREATE SEQUENCE public.manufacturer_seq START 1 INCREMENT 1 NO MAXVALUE
 ↪  CACHE 1;
/* create requests table sequence */
CREATE SEQUENCE public.request_seq START 1 INCREMENT 1 NO MAXVALUE CACHE
 ↪  1;
/* create offer table sequence */
CREATE SEQUENCE public.offer_seq START 1 INCREMENT 1 NO MAXVALUE CACHE 1;

/* categories table */
CREATE TABLE IF NOT EXISTS public.categories
(
  id          INT PRIMARY KEY       NOT NULL DEFAULT
   ↪  nextval('category_seq'),
  skroutz_id  INT UNIQUE            NOT NULL, -- the category id we get
   ↪  from a request to the Skroutz API
  name        VARCHAR(300)          NOT NULL, -- the category name
  image_url   VARCHAR(300), -- the category's image url
--   web_uri     VARCHAR(300), -- the category's web url
  parent_id   INT REFERENCES categories (skroutz_id), -- the parent id
  etag        VARCHAR(32), -- a tag used for conditional http requests
  inserted_at TIMESTAMP             NOT NULL DEFAULT CURRENT_TIMESTAMP,
   ↪  -- the timestamp when the item was inserted
  modified_at TIMESTAMP, -- the timestamp when the item was last modified
  checked_at  TIMESTAMP             NOT NULL  -- the timestamp when we
   ↪  last queried the given item
);

```

```sql
/* manufacturers table */
CREATE TABLE IF NOT EXISTS public.manufacturers
(
  id          INT PRIMARY KEY  NOT NULL DEFAULT
  ↪  nextval('manufacturer_seq'),
  skroutz_id  INT UNIQUE       NOT NULL, -- the manufacturer id we get
  ↪  from a request to the Skroutz API
  name        VARCHAR(300)     NOT NULL, -- the manufacturer's name
  image_url   VARCHAR(300), -- the manufacturer's image url
  etag        VARCHAR(32), -- a tag used for conditional http requests
  inserted_at TIMESTAMP        NOT NULL DEFAULT CURRENT_TIMESTAMP, -- the
  ↪  timestamp when the item was inserted
  modified_at TIMESTAMP, -- the timestamp when the item was last modified
  checked_at  TIMESTAMP           -- the timestamp when we last queried
  ↪  the given item
);

/* shops table, some of the Skroutz fields were omitted because they were
 * excessive for our needs.
 */
CREATE TABLE IF NOT EXISTS public.shops
(
  id            INT PRIMARY KEY  NOT NULL DEFAULT nextval('shop_seq'),
  skroutz_id    INT UNIQUE       NOT NULL, -- the shop id we get from a
  ↪  request to the Skroutz API
  name          VARCHAR(300)     NOT NULL, -- the shop name
  link          VARCHAR(300)     NOT NULL, -- the shop url
  phone         VARCHAR(100), -- the shop's phone, a 10 digit number
  image_url     VARCHAR(300), -- the shop's image url
  thumbshot_url VARCHAR(300), -- the shop's thumbnail url
  etag          VARCHAR(32), -- a tag used for conditional http requests
  inserted_at   TIMESTAMP        NOT NULL DEFAULT CURRENT_TIMESTAMP, --
  ↪  the timestamp when the item was inserted
  modified_at   TIMESTAMP, -- the timestamp when the item was last
  ↪  modified
  checked_at    TIMESTAMP            -- the timestamp when we last
  ↪  queried the given item
);

/* skus table */
CREATE TABLE IF NOT EXISTS public.skus
(
  id           INT PRIMARY KEY NOT NULL DEFAULT nextval('sku_seq'),
  skroutz_id   INT UNIQUE      NOT NULL, -- the sku id we get from a
  ↪  request to the Skroutz API
  name         VARCHAR(300)    NOT NULL, -- the sku name
  display_name VARCHAR(300)    NOT NULL, -- the sku display name
  category_id  INT             NOT NULL REFERENCES categories
  ↪  (skroutz_id), -- the category id
  click_url    VARCHAR(300), -- the sku's url
```

```
  image_url    VARCHAR(300), -- the sku's image
  price_max    NUMERIC(7, 2)   NOT NULL, -- the max price the sku had at
   ↪ the last check
  price_min    NUMERIC(7, 2)   NOT NULL, -- the min price the sku had at
   ↪ the last check
  etag         VARCHAR(32), -- a tag used for conditional http requests
  inserted_at  TIMESTAMP       NOT NULL DEFAULT CURRENT_TIMESTAMP, -- the
   ↪ timestamp when the item was inserted
  modified_at  TIMESTAMP, -- the timestamp when the item was last modified
  checked_at   TIMESTAMP    -- the timestamp when we last queried the
   ↪ given item
);

/* products table */
CREATE TABLE IF NOT EXISTS public.products
(
  id                INT PRIMARY KEY      NOT NULL DEFAULT
   ↪ nextval('prod_seq'),
  skroutz_id        INT UNIQUE           NOT NULL, -- the product id we
   ↪ get from a request to the Skroutz API
  name              VARCHAR(300)         NOT NULL, -- the product name
  sku_id            INT                  NOT NULL REFERENCES skus
   ↪ (skroutz_id), -- the product code given to match this product in
   ↪ other shops
  shop_id           INT                  NOT NULL REFERENCES shops
   ↪ (skroutz_id),
  shop_uid          VARCHAR(64), -- the unique uid given from the shop
  category_id       INT           NOT NULL REFERENCES categories
   ↪ (skroutz_id),
  etag              VARCHAR(32), -- a tag used for conditional http
   ↪ requests
  availability      VARCHAR(50), -- the current availability
  click_url         VARCHAR(300), -- the url given from Skroutz API
  price             NUMERIC(7, 2) NOT NULL, -- the current price
  price_changes     INT           NOT NULL DEFAULT 0, -- how many times
   ↪ the product's price has changed
  average_past_price NUMERIC(7, 2) NOT NULL, -- the average price we have
   ↪ calculated for the product in the past
  is_bargain        BOOL, -- true if the product is a bargain
  inserted_at       TIMESTAMP     NOT NULL DEFAULT CURRENT_TIMESTAMP, --
   ↪ the timestamp when the item was inserted
  modified_at       TIMESTAMP, -- the timestamp when the item was last
   ↪ modified
  checked_at        TIMESTAMP      -- the timestamp when we last queried
   ↪ the given product
);

/* price history table */
CREATE TABLE IF NOT EXISTS public.prices
(
```

```
  id         INT PRIMARY KEY NOT NULL DEFAULT nextval('price_seq'),
  product_id INT             NOT NULL REFERENCES products (id), -- the
   ↪ product id
  price      NUMERIC(7, 2)   NOT NULL, -- the price
  checked_at TIMESTAMP       NOT NULL DEFAULT CURRENT_TIMESTAMP -- the
   ↪ moment we logged the change
);

CREATE TABLE IF NOT EXISTS public.requests
(
  id         INT PRIMARY KEY     NOT NULL DEFAULT nextval('request_seq'),
  url        VARCHAR(200) UNIQUE NOT NULL, -- the request uri on the
   ↪ skroutz website
  etag       VARCHAR(32), -- the etag we received on the last request
  checked_at TIMESTAMP     -- the timestamp when we last hit the selected
   ↪ request
);

CREATE TABLE IF NOT EXISTS public.offers
(
  id                   INT PRIMARY KEY       NOT NULL DEFAULT
   ↪ nextval('offer_seq'),
  product_id           INT                   NOT NULL REFERENCES products
   ↪ (id), -- the product id
  price_id             INT                   NOT NULL REFERENCES prices
   ↪ (id), -- the price id
  accepted_by          SMALLINT              NOT NULL, -- GRUBBS = 1,
   ↪ CHAUVENET = 2, QUARTILES = 3, GRUBBS + CHAUVE = 4,
                                                  -- GRUBBS + QUARTILES =
                                                     ↪ 5, CHAUVE +
                                                     ↪ QUARTILES = 6,
                                                     ↪ ALL 3 = 7
  grubbs_flexibility    VARCHAR(10)          NOT NULL, -- NORMAL = 1,
   ↪ STRONG = 2, RELAXED = 3, DEFAULT = 1
  chauvenet_flexibility VARCHAR(10)          NOT NULL, -- NORMAL = 1,
   ↪ STRONG = 2, RELAXED = 3, DEFAULT = 1
  quartile_flexibility  VARCHAR(10)          NOT NULL, -- NORMAL = 1,
   ↪ STRONG = 2, RELAXED = 3, DEFAULT = 1
  inserted_at          TIMESTAMP             NOT NULL DEFAULT
   ↪ CURRENT_TIMESTAMP, -- the timestamp when the item was inserted
  finished_at          TIMESTAMP, -- the timestamp when the item was
   ↪ last modified
  checked_at           TIMESTAMP             -- the timestamp when we
   ↪ last queried the given
);

-- match sequences to their tables
ALTER SEQUENCE public.prod_seq OWNED BY products.id;
ALTER SEQUENCE public.shop_seq OWNED BY shops.id;
ALTER SEQUENCE public.category_seq OWNED BY categories.id;
```

```
ALTER SEQUENCE public.price_seq OWNED BY prices.id;
ALTER SEQUENCE public.sku_seq OWNED BY skus.id;
ALTER SEQUENCE public.manufacturer_seq OWNED BY manufacturers.id;
ALTER SEQUENCE public.request_seq OWNED BY requests.id;
ALTER SEQUENCE public.offer_seq OWNED BY offers.id;

-- create indexes on skroutz_id on all entity tables
CREATE INDEX skroutz_index_category ON categories (skroutz_id);
CREATE INDEX skroutz_index_manufacturer ON manufacturers (skroutz_id);
CREATE INDEX skroutz_index_shop ON shops (skroutz_id);
CREATE INDEX skroutz_index_sku ON skus (skroutz_id);
CREATE INDEX skroutz_index_product ON products (skroutz_id);

-- and one for offers on its primary id
CREATE INDEX index_offer ON offers (id);
CREATE INDEX index_offer_product
  ON offers (product_id);
```

Παράρτημα C

Κρίσιμες τιμές κατανομής Student – T

ν	60.0%	66.7%	75.0%	80.0%	87.5%	90.0%	95.0%	97.5%	99.0%	99.5%	99.9%
1	0.325	0.577	1.000	1.376	2.414	3.078	6.314	12.706	31.821	63.657	318.31
2	0.289	0.500	0.816	1.061	1.604	1.886	2.920	4.303	6.965	9.925	22.327
3	0.277	0.476	0.765	0.978	1.423	1.638	2.353	3.182	4.541	5.841	10.215
4	0.271	0.464	0.741	0.941	1.344	1.533	2.132	2.776	3.747	4.604	7.173
5	0.267	0.457	0.727	0.920	1.301	1.476	2.015	2.571	3.365	4.032	5.893
6	0.265	0.453	0.718	0.906	1.273	1.440	1.943	2.447	3.143	3.707	5.208
7	0.263	0.449	0.711	0.896	1.254	1.415	1.895	2.365	2.998	3.499	4.785
8	0.262	0.447	0.706	0.889	1.240	1.397	1.860	2.306	2.896	3.355	4.501
9	0.261	0.445	0.703	0.883	1.230	1.383	1.833	2.262	2.821	3.250	4.297
10	0.260	0.444	0.700	0.879	1.221	1.372	1.812	2.228	2.764	3.169	4.144
11	0.260	0.443	0.697	0.876	1.214	1.363	1.796	2.201	2.718	3.106	4.025
12	0.259	0.442	0.695	0.873	1.209	1.356	1.782	2.179	2.681	3.055	3.930
13	0.259	0.441	0.694	0.870	1.204	1.350	1.771	2.160	2.650	3.012	3.852
14	0.258	0.440	0.692	0.868	1.200	1.345	1.761	2.145	2.624	2.977	3.787
15	0.258	0.439	0.691	0.866	1.197	1.341	1.753	2.131	2.602	2.947	3.733
16	0.258	0.439	0.690	0.865	1.194	1.337	1.746	2.120	2.583	2.921	3.686
17	0.257	0.438	0.689	0.863	1.191	1.333	1.740	2.110	2.567	2.898	3.646
18	0.257	0.438	0.688	0.862	1.189	1.330	1.734	2.101	2.552	2.878	3.610
19	0.257	0.438	0.688	0.861	1.187	1.328	1.729	2.093	2.539	2.861	3.579
20	0.257	0.437	0.687	0.860	1.185	1.325	1.725	2.086	2.528	2.845	3.552
21	0.257	0.437	0.686	0.859	1.183	1.323	1.721	2.080	2.518	2.831	3.527
22	0.256	0.437	0.686	0.858	1.182	1.321	1.717	2.074	2.508	2.819	3.505
23	0.256	0.436	0.685	0.858	1.180	1.319	1.714	2.069	2.500	2.807	3.485
24	0.256	0.436	0.685	0.857	1.179	1.318	1.711	2.064	2.492	2.797	3.467
25	0.256	0.436	0.684	0.856	1.178	1.316	1.708	2.060	2.485	2.787	3.450
26	0.256	0.436	0.684	0.856	1.177	1.315	1.706	2.056	2.479	2.779	3.435
27	0.256	0.435	0.684	0.855	1.176	1.314	1.703	2.052	2.473	2.771	3.421
28	0.256	0.435	0.683	0.855	1.175	1.313	1.701	2.048	2.467	2.763	3.408
29	0.256	0.435	0.683	0.854	1.174	1.311	1.699	2.045	2.462	2.756	3.396
30	0.256	0.435	0.683	0.854	1.173	1.310	1.697	2.042	2.457	2.750	3.385
35	0.255	0.434	0.682	0.852	1.170	1.306	1.690	2.030	2.438	2.724	3.340
40	0.255	0.434	0.681	0.851	1.167	1.303	1.684	2.021	2.423	2.704	3.307
45	0.255	0.434	0.680	0.850	1.165	1.301	1.679	2.014	2.412	2.690	3.281
50	0.255	0.433	0.679	0.849	1.164	1.299	1.676	2.009	2.403	2.678	3.261
55	0.255	0.433	0.679	0.848	1.163	1.297	1.673	2.004	2.396	2.668	3.245
60	0.254	0.433	0.679	0.848	1.162	1.296	1.671	2.000	2.390	2.660	3.232
∞	0.253	0.431	0.674	0.842	1.150	1.282	1.645	1.960	2.326	2.576	3.090

Παράρτημα D

Κρίσιμες τιμές αθροιστικής κανονικής κατανομής

x	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7703	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.0	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8810	0.8830
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394	0.9406	0.9418	0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495	0.9505	0.9515	0.9525	0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9625	0.9633
1.8	0.9641	0.9649	0.9656	0.9664	0.9671	0.9678	0.9686	0.9693	0.9699	0.9706
1.9	0.9713	0.9719	0.9726	0.9732	0.9738	0.9744	0.9750	0.9756	0.9761	0.9767
2.0	0.9772	0.9778	0.9783	0.9788	0.9793	0.9798	0.9803	0.9808	0.9812	0.9817
2.1	0.9821	0.9826	0.9830	0.9834	0.9838	0.9842	0.9846	0.9850	0.9854	0.9857
2.2	0.9861	0.9864	0.9868	0.9871	0.9875	0.9878	0.9881	0.9884	0.9887	0.9890
2.3	0.9893	0.9896	0.9898	0.9901	0.9904	0.9906	0.9909	0.9911	0.9913	0.9916
2.4	0.9918	0.9920	0.9922	0.9925	0.9927	0.9929	0.9931	0.9932	0.9934	0.9936
2.5	0.9938	0.9940	0.9941	0.9943	0.9945	0.9946	0.9948	0.9949	0.9951	0.9952
2.6	0.9953	0.9955	0.9956	0.9957	0.9959	0.9960	0.9961	0.9962	0.9963	0.9964
2.7	0.9965	0.9966	0.9967	0.9968	0.9969	0.9970	0.9971	0.9972	0.9973	0.9974
2.8	0.9974	0.9975	0.9976	0.9977	0.9977	0.9978	0.9979	0.9979	0.9980	0.9981
2.9	0.9981	0.9982	0.9982	0.9983	0.9984	0.9984	0.9985	0.9985	0.9986	0.9986
3.0	0.9987	0.9987	0.9987	0.9988	0.9988	0.9989	0.9989	0.9989	0.9990	0.9990
3.1	0.9990	0.9991	0.9991	0.9991	0.9992	0.9992	0.9992	0.9992	0.9993	0.9993
3.2	0.9993	0.9993	0.9994	0.9994	0.9994	0.9994	0.9994	0.9995	0.9995	0.9995
3.3	0.9995	0.9995	0.9995	0.9996	0.9996	0.9996	0.9996	0.9996	0.9996	0.9997
3.4	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9998
3.5	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998