Έγγραφο απαιτήσεων λογισμικού (SRS)

ΠΡΟΣΑΡΜΟΓΗ ΤΟΥ ΑΝΤΙΣΤΟΙΧΟΥ ΕΓΓΡΑΦΟΥ ΤΟΥ ΠΡΟΤΥΠΟΥ ISO/IEC/IEEE 29148:2011

CrowTech ~ crowdsourcing the technological market.

Περιεχόμενα

1. Εισαγωγή

1.1 Εισαγωγή: σκοπός του λογισμικού

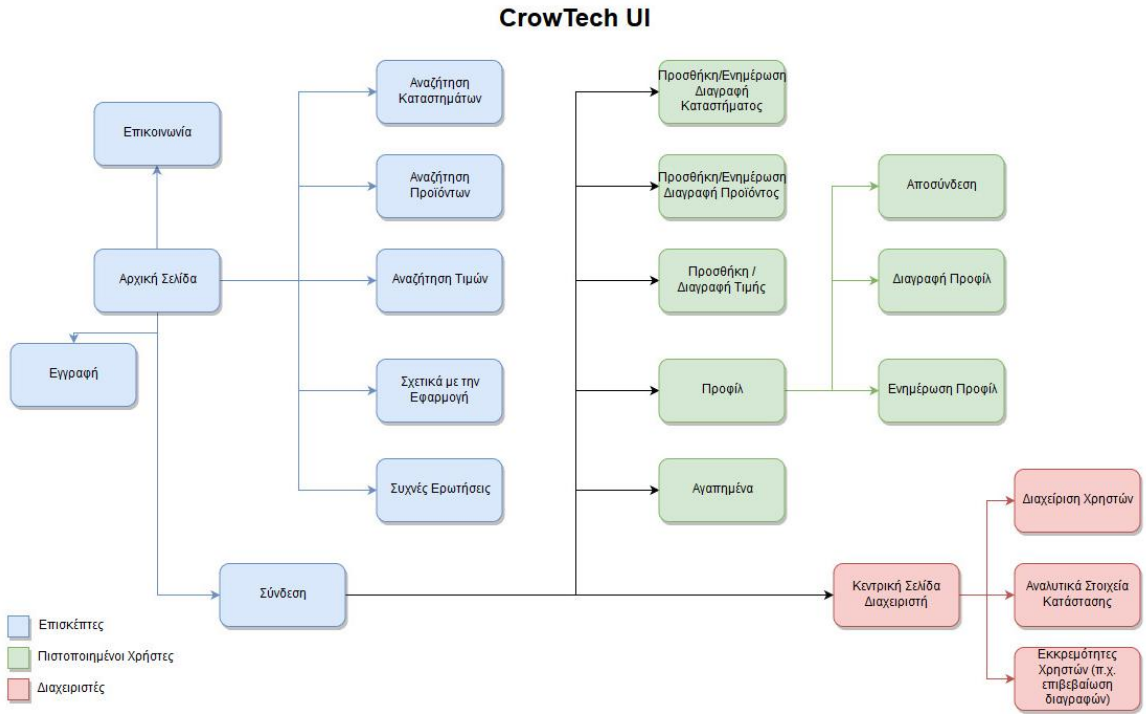
Το σύστημα που αναπτύσσουμε συνιστά ένα διαδικτυακό παρατηρητήριο τιμών το οποίο βασίζεται στο μοντέλο του πληθοπορισμού (crowdsourcing). Στόχος του συστήματος είναι η κάλυψη της ολοένα και πιο έντονης ανάγκης του καταναλωτικού κοινού για ενημέρωση σχετικά με τα τεχνολογικά προϊόντα και διαχείριση των πληροφοριών που σχετίζονται με τα σημεία πώλησης και τις τιμές αυτών · ανάγκη που προκύπτει από το αυξανόμενο πλήθος τόσο των προϊόντων τεχνολογίας, όσο και των καταστημάτων που τα διαθέτουν. Το γεγονός ότι ένα προϊόν μπορεί να πωλείται σε διαφορετική τιμή ανάλογα με το σημείο πώλησης (και μάλιστα με μεγάλη διαφορά στο κόστος) οδηγεί τον καταναλωτή στη σύγκριση τιμών και – εάν δεν υπάρχει δυνατότητα αποστολής από το εκάστοτε κατάστημα – αποστάσεων. Το παρατηρητήριό μας στοχεύει να καλύψει ακριβώς αυτή την ανάγκη μέσω τη συνεργασίας των χρηστών του για τη συλλογή πληροφοριών.

1.2 Επισκόπηση του λογισμικού

Στην Εικόνα 1 παραθέτουμε διάγραμμα με τις διεπαφές που δύναται να αλληλοεπιδράσει ο χρήστης της εφαρμογής μας. Σημειώνουμε ότι ως χρήστη εννοούμε μία από τις εξής τρεις κατηγορίες:

1) Ανώνυμος επισκέπτης (μπορεί μόνο να αναζητήσει προϊόντα, καταστήματα και τιμές).
2) Πιστοποιημένος χρήστης (έχει τη δυνατότητα να επεξεργαστεί τα δεδομένα του παρατηρητηρίου).
3) Διαχειριστή (ρυθμίζει τη δραστηριότητα των χρηστών και παρακολουθεί τη κατάσταση του συστήματος).

Στην επόμενη παράγραφο θα ασχοληθούμε με τη τεχνική πλευρά του συστήματος και του λογισμικού του. Προς το παρόν, επικεντρωνόμαστε στο περιβάλλον που προσφέρεται στον χρήστη. Πιο συγκεκριμένα, με την εκκίνηση της εφαρμογής δίνεται η δυνατότητα στον χρήστη να περιηγηθεί στις πιο στοιχειώδεις λειτουργίες · ήτοι αναζήτηση σχετικά με τιμές, προϊόντα και καταστήματα, καθώς και προβολή πληροφοριών σχετικών με την εφαρμογή, εγγραφή νέου χρήστη, σελίδα με συχνές ερωτήσεις και δυνατότητα επικοινωνίας με τους διαχειριστές ή τους δημιουργούς. Εάν γίνει επιτυχής σύνδεση, ο (πιστοποιημένος πια) χρήστης διαθέτει αναβαθμισμένες δυνατότητες. Μπορεί να επεξεργαστεί τα δεδομένα της βάσεις προσθέτοντας, αφαιρώντας ή αλλάζοντας πληροφορίες σχετικές με τα προϊόντα, τα καταστήματα ή τις τιμές. Εφόσον ο χρήστης που συνδέθηκε αναγνωριστεί από την εφαρμογή ως διαχειριστής (πιστοποίηση την οποία δίνει μόνο ο προγραμματιστής απευθείας στη βάση δεδομένων η οποία διακρίνεται στο διάγραμμα της επόμενης ενότητας), εμφανίζεται επίσης μία σελίδα διαχείρισης (Administration Dashboard). Από εκεί παρέχεται η δυνατότητα παρακολούθησης στοιχείων όπως η συνολική κίνηση δεδομένων του παρατηρητηρίου, η ποσότητα και η συχνότητα της δραστηριότητας των χρηστών κ.α. Μία από τις εργασίες του διαχειριστή – η οποία συντελείται μέσω της σελίδας διαχείρισης – είναι η επιβεβαίωση των αιτημάτων των επιβεβαιωμένων χρηστών σχετικά με τη διαγραφή δεδομένων. Για να προστατευτούν τα δεδομένα της εφαρμογής (προϊόντα, καταστήματα και σύνολο τιμών) οι εθελοντές (crowdsourcers ή αλλιώς οι πιστοποιημένοι χρήστες) μπορούν μόνο να αλλάζουν τη κατάσταση διαθεσιμότητας των προϊόντων και των καταστημάτων, και όχι να τα διαγράφουν οριστικά. Η οριστική διαγραφή συμβαίνει μόνο με επιβεβαίωση διαχειριστή.



Εικόνα 1. Διεπαφές Χρήστη (Application UI)

1.3.1 Διεπαφές με εξωτερικά συστήματα και εφαρμογές λογισμικού

Ακολουθώντας σε μεγάλο βαθμό το MVC μοντέλο, και ενσωματώνοντας τη REST αρχιτεκτονική κατά τη διαδικασία της ανάπτυξης, μία επισκόπηση της δομή του λογισμικού μας φαίνεται στο Deployment Diagram της εικόνας 2. Οι εξωτερικές διεπαφές του συστήματός μας αφορούν:

1) Τα άκρα της σύνδεσης του φυλλομετρητή (πλατφόρμα του χρήστη της εφαρμογής) με τον διακομιστή που φιλοξενεί το API της εφαρμογής μας αλλά και το σύνολο των HTML σελίδων και της JavaScript που αποτελούν το User Interface.

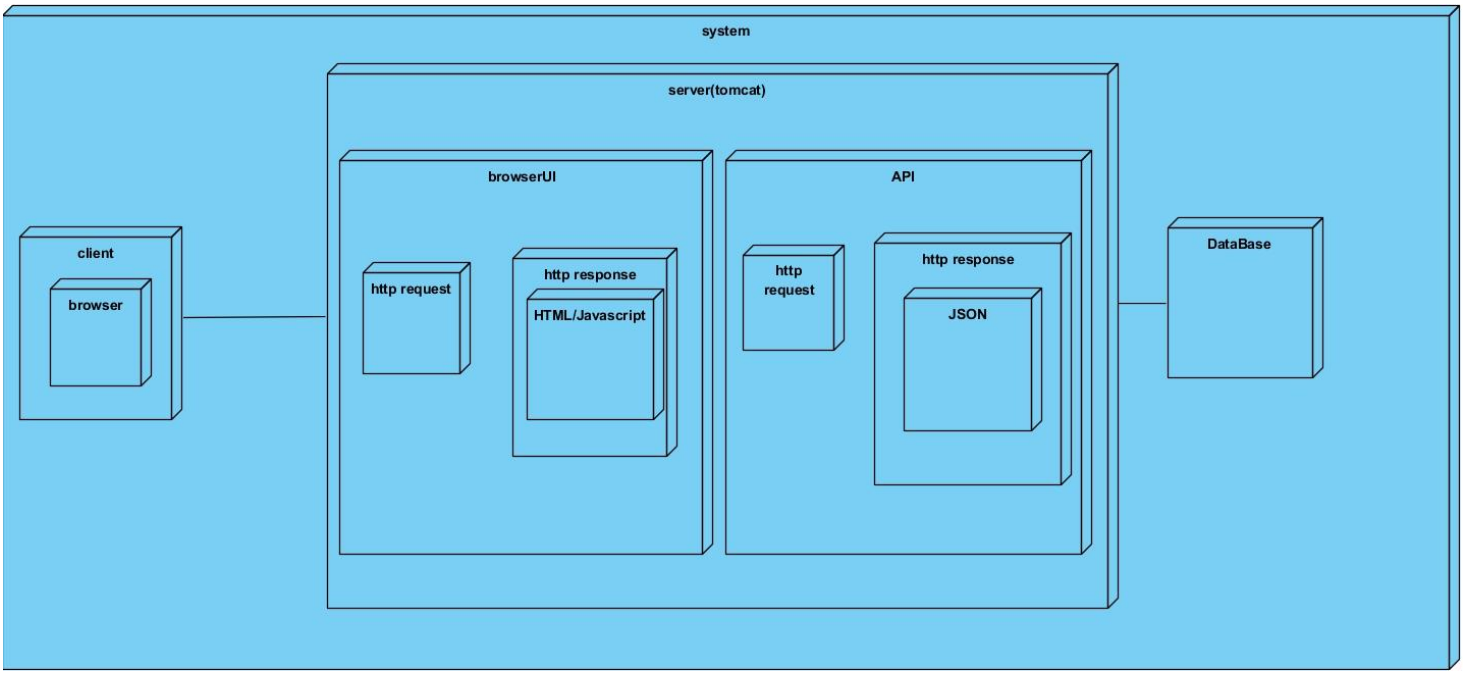
2) Την επικοινωνία ανάμεσα σε User Interface και API για τη σωστή ενημέρωση του περιεχομένου που επιστρέφεται στον χρήστη.

3) Τις εξόδους του API από και προς τη Βάση Δεδομένων.

Ο Client επικοινωνεί μέσω http requests με το UI και το API. Από το UI δέχεται τα http responses που ενσωματώνουν τα html pages που πρέπει να προβάλλει (μαζί με τη JavaScript που τα συνοδεύει). Από το API δέχεται μόνο http responses που περιέχουν JSON αρχεία με τις πληροφορίες που χρειάζεται.

Ανάχωμα μεταξύ του client (browser) και του μοντέλου δεδομένων του συστήματός μας (Database) συνιστά το API (που βάσει του MVC προτύπου μπορεί να χαρακτηρισθεί ως Controller). Το API είναι υπεύθυνο για την επικοινωνία με τη Βάση Δεδομένων μέσω SQL requests & responses. Ανάλογα με τα αιτήματα του χρήστη πραγματοποιεί την απαραίτητη επεξεργασία δεδομένων, ενημερώνει ή ζητά αποτελέσματα από τη βάση και, τέλος, επιστρέφει αποτελέσματα στον φυλλομετρητή του Client με μορφοποίηση JSON. Το API και το BrowserUI φιλοξενούνται στον ίδιο server, αλλά αυτό δεν είναι δεσμευτικό. Υπάρχει περίπτωση για τις ανάγκες του παρατηρητηρίου να ανεξαρτητοποιηθούν τα δύο στοιχεία και να φιλοξενηθούν σε ξεχωριστούς διακομιστές (π.χ. Jetty για το API και Tomcat για το BrowserUI).

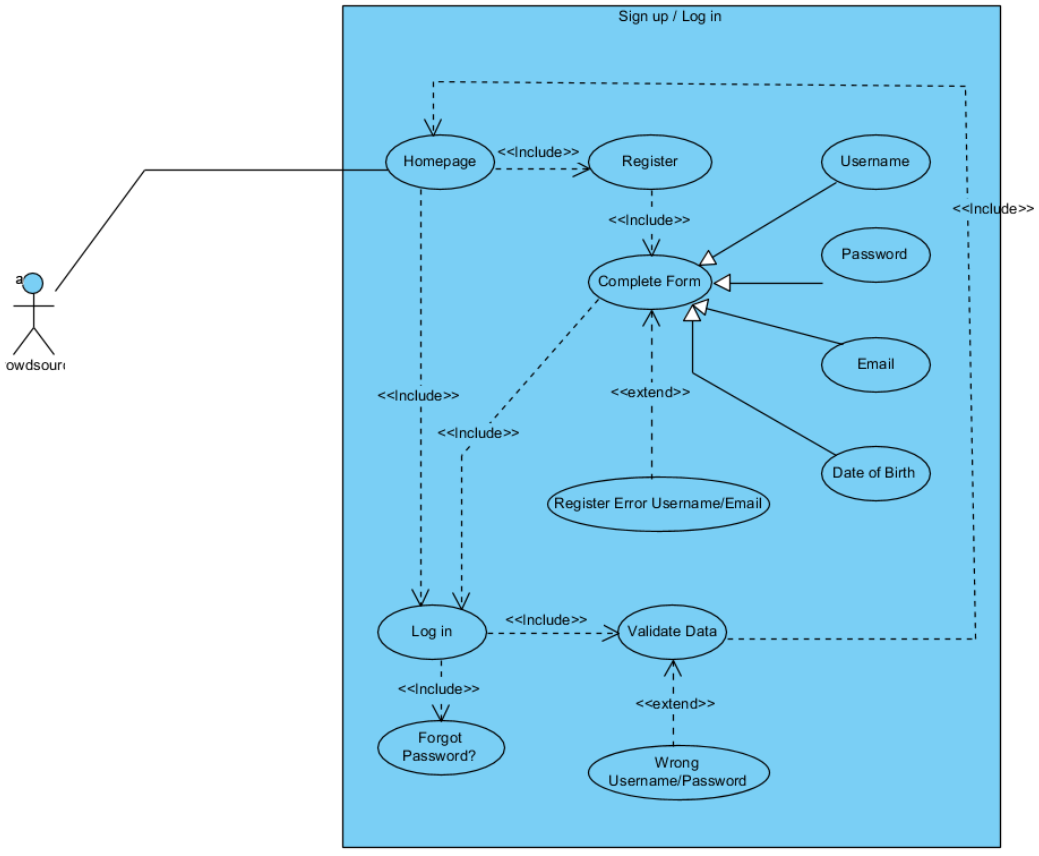
Το API της εφαρμογής μας είναι έτοιμο να εξυπηρετήσει, εκτός από φυλλομετρητές ιστού των χρηστών, αυτόνομα προγραμματιστικά συστήματα και εξωτερικά APIs. Οποιοδήποτε σύστημα διαθέτει συμβατότητα με τα http endpoints και τα πρωτόκολλα επικοινωνίας του API μας δύναται να αλληλοεπιδράσει με το παρατηρητήριο διαθέτοντας όλες τις δυνατότητες που περιγράψαμε παραπάνω για τους χρήστες.



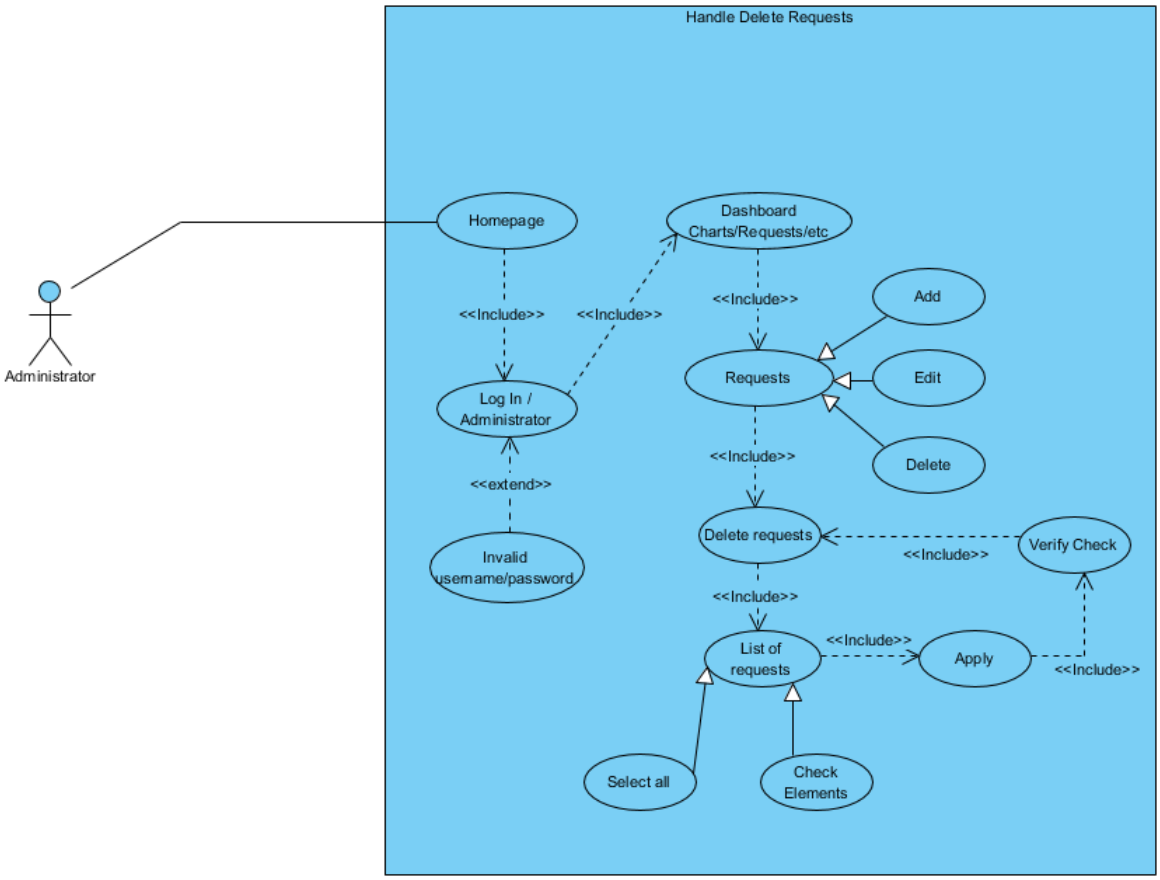
Εικόνα 2 - Deployment Diagram

1.3.2 Διεπαφές με το χρήστη

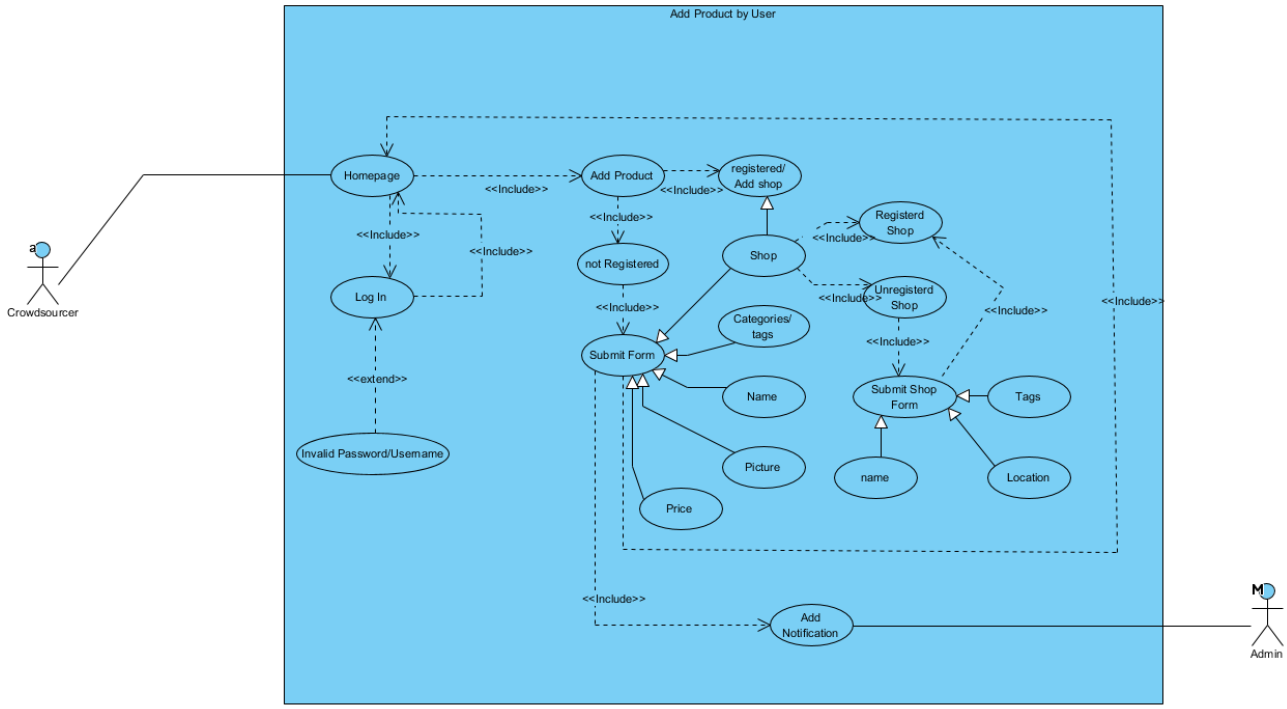
Στις εικόνες 3 με 6 παρουσιάζουμε ενδεικτικά σενάρια χρήσης σε UML (use case). Κάθε ένα από αυτά έχει στόχο να περιγράψει τη μορφή και τα βήματα της αλληλεπίδρασης του χρήστη με την εφαρμογή.



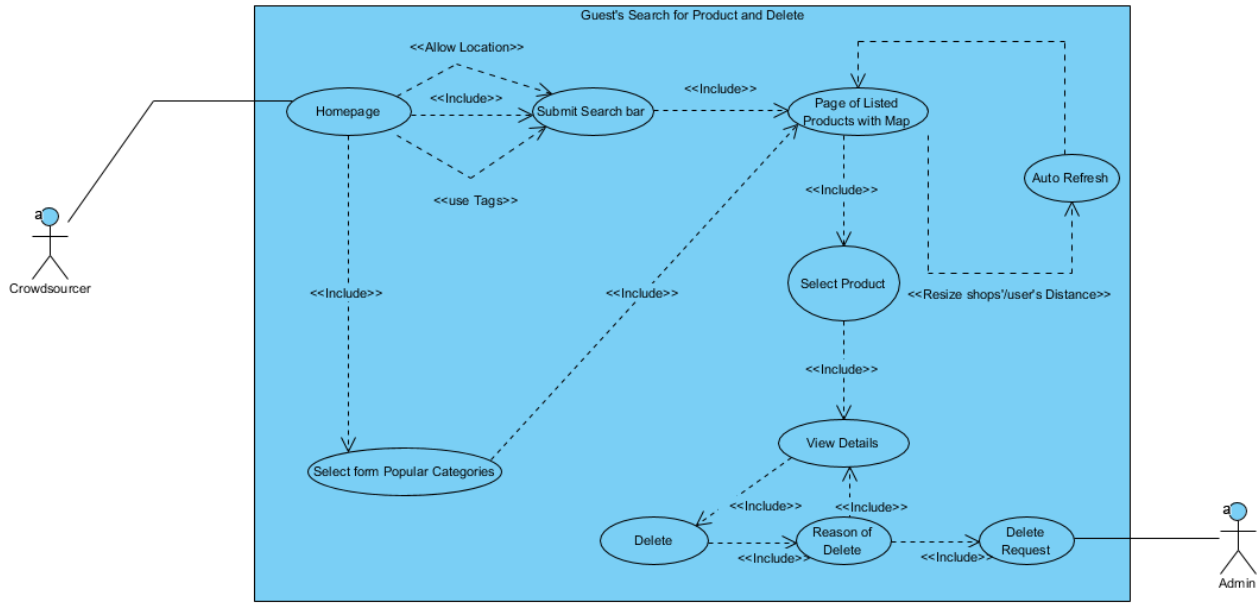
Εικόνα 3. UML Use Case: Sign up & Log in



Εικόνα 4. UML Use Case: Manager handles delete requests



Εικόνα 5. UML Use Case: Crowdsourcer adds product



Εικόνα 6. UML Use Case: User search for product and then deletes it

3.1 Εξωτερικές διεπαφές

Στην ενότητα 1.3.1 αναλύσαμε τη δομική πλευρά των διεπαφών του λογισμικού μας. Σε αυτή την ενότητα θα περιγράψουμε την εμπειρία του χρήστη μέσα από τις γραφικές διεπαφές (Browser UI) με τις οποίες αλληλοεπιδρά. Η εικόνα 1 περιγράφει διαγραμματικά τις σελίδες που περιέχουν τα στοιχεία που θα περιγράψουμε παρακάτω.

Καταρχάς, πρώτος κόμβος επίσκεψης από κάθε χρήστη (ανώνυμος χρήστης, εθελοντής ή διαχειριστής) είναι η κεντρική σελίδα (Home Page). Εκεί βρίσκεται μία μπάρα αναζήτησης σε εμφανές και κεντρικό σημείο ώστε να δίνεται άμεσα η δυνατότητα στον (ανώνυμο μέχρι στιγμής) επισκέπτη να αναζητήσει προϊόντα, καταστήματα και τιμές (δίνοντας είτε απευθείας το όνομα του αντικειμένου που τον ενδιαφέρει ή εισάγοντας βοηθητικά πεδία – tags – που το περιγράφουν). Στην ίδια κεντρική σελίδα μπορεί να εντοπίσει συνδέσμους που οργανώνονται στο πάνω μέρος (footer) και οδηγούν – μεταξύ άλλων – στις σελίδες όπου μπορεί να γίνει πιο συγκεκριμένη και αναλυτική αναζήτηση στοιχείων, ανάγνωση πληροφοριών σχετικών με την εφαρμογή και την ομάδα των δημιουργών της, εύρεση συχνών ερωτήσεων που αφορούν τη σωστή χρήση και τη λειτουργία της εφαρμογής συνοδευόμενες από τις απαντήσεις του, φόρμα επικοινωνίας καθώς και κατάλληλη σελίδα σύνδεσης για εγγεγραμμένους χρήστες (ή δημιουργίας λογαριασμού για εκείνους που θέλουν να εγγραφούν).

Κάνοντας μια αναζήτηση, ο χρήστης οδηγείται σε κατάλληλη σελίδα όπου παρουσιάζονται με χρήσιμο τρόπο τα αποτελέσματα. Συγκεκριμένα, τα αποτελέσματα μιας αναζήτησης ταξινομούνται βάσει κριτηρίων – τα οποία ο χρήστης έχει τη δυνατότητα να καθορίσει – και εμφανίζονται ως τοποθεσίες σε έναν χάρτη (ο οποίος περιέχει και τη τοποθεσία του χρήστη εάν ο ίδιος επιλέξει να τη κοινοποιήσει στην εφαρμογή μας). Επιλέγοντας κάποιο από τα αποτελέσματα της αναζήτησης – ας υποθέσουμε ότι πρόκειται για προϊόντα, ωστόσο τα παρακάτω χαρακτηριστικά ισχύουν και για τα καταστήματα και τις τιμές – μεταβαίνει αυτόματα σε σελίδα που παρουσιάζει αναλυτικά το συγκεκριμένο προϊόν. Εάν, μάλιστα, έχει συνδεθεί ως πιστοποιημένος χρήστης, του δίνεται σε αυτή την αναλυτική σελίδα η δυνατότητα να αλλάξει κάποια (κανένα ή και όλα) χαρακτηριστικά του προϊόντος, ή ακόμα και να ζητήσει τη διαγραφή του από το σύστημα.

Στη σελίδα των συχνών αποριών, παρατίθενται με ευανάγνωστο τρόπο οι πιο συχνές ερωτήσεις που μπορεί να έχει κάποιος που χρησιμοποιεί για πρώτη φορά την εφαρμογή. Συμπεριλαμβάνονται ερωτήσεις και απαντήσεις σχετικές με τους κανονισμούς ορθής χρήσης, τις βασικές δυνατότητες της εφαρμογής και τους τρόπους αντιμετώπισης ενός προβλήματος (σωστή καταχώρηση ενός σφάλματος στη φόρμα επικοινωνίας).

Στη σελίδα επικοινωνίας, ο χρήστης έχει τη δυνατότητα να συμπληρώσει κείμενο μαζί με τα στοιχεία του και αυτά να σταλθούν σε κάποιον διαχειριστή της εφαρμογής. Στο μέλλον προβλέπεται να διατίθενται και τηλεφωνικοί αριθμοί ώστε σε συγκεκριμένες ώρες της ημέρα ο χρήστης να μπορεί να μιλήσει απευθείας με κάποιον εκπρόσωπο μας.

3.2 Λειτουργίες: περιπτώσεις χρήσης

Για κάθε ένα από σενάρια χρήσης που παρουσιάστηκαν μέσω των UML διαγραμμάτων παραπάνω, παραθέτουμε αναλυτική περιγραφή των στοιχείων που τα περιγράφουν.

3.2.1 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 1: *Sign up/ Log in*

3.2.1.1 Χρήστες (ρόλοι) που εμπλέκονται

Αυτή η χρήση του API περιλαμβάνει τον ανώνυμο (μη πιστοποιημένο) χρήστη που θέλει να δημιουργήσει λογαριασμό ή αν έχει ήδη να συνδεθεί σε αυτόν.

3.2.1.2 Προϋποθέσεις εκτέλεσης

Για να κάνει Sign up ο χρήστης πρέπει να έχει ένα προσωπικό email για την διαδικασία επικύρωσης της εγγραφής του ενώ για το Log In πρέπει ο χρήστης να έχει ήδη δημιουργήσει ένα λογαριασμό ώστε να μπορεί να συνδεθεί με τα στοιχεία του(username,password).

3.2.1.3 Περιβάλλον εκτέλεσης

Για την εκτέλεση αυτής της χρήσης χρειάζεται η διαδικτυακή διεπαφή χρήστη καθώς και η επικοινωνία του API με την βάση για το post στοιχείων (sign up) ή επικύρωση στοιχείων (log in).

3.2.1.4 Δεδομένα εισόδου

Τα δεδομένα εισόδου από τον χρήστη περιλαμβάνουν :

1) *Sign up: πληροφορίες χρήστη για την εγγραφή (όνομα, επώνυμο, email, password, username, ηλικία, φύλλο)*

Log in: τα ατομικά στοιχεία του εγγεγραμμένου χρήστη (username, password, keep me logged in check box).

3.2.1.5 Παράμετροι

Για την εγγραφή νέου χρήστη, πρέπει το e-mail και το username που θα εισάγει ο χρήστης να μην έχουν καταχωρηθεί ξανά στη βάση δεδομένων. Σε διαφορετική περίπτωση επιστρέφεται κατάλληλο μήνυμα σφάλματος και ο χρήστης πρέπει να καταχωρήσει νέες τιμές για τα πεδία αυτά.

3.2.1.6 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

Βήμα 1: Είσοδος στη homepage(guest)

Βήμα 2: Αν ο χρήστης δεν έχει λογαριασμό Βήμα 4

Βήμα 3: Βήμα 7

Βήμα 4: Επιλογή Register

Βήμα 5: (Sign up Page)Συμπλήρωση φόρμας στοιχείων χρήστη

Βήμα 6: Αν υπάρχει λάθος (πιασμένο username, email) βήμα

Βήμα 7: (Login Page) Συμπλήρωση ατομικών στοιχείων για είσοδο(username, password)

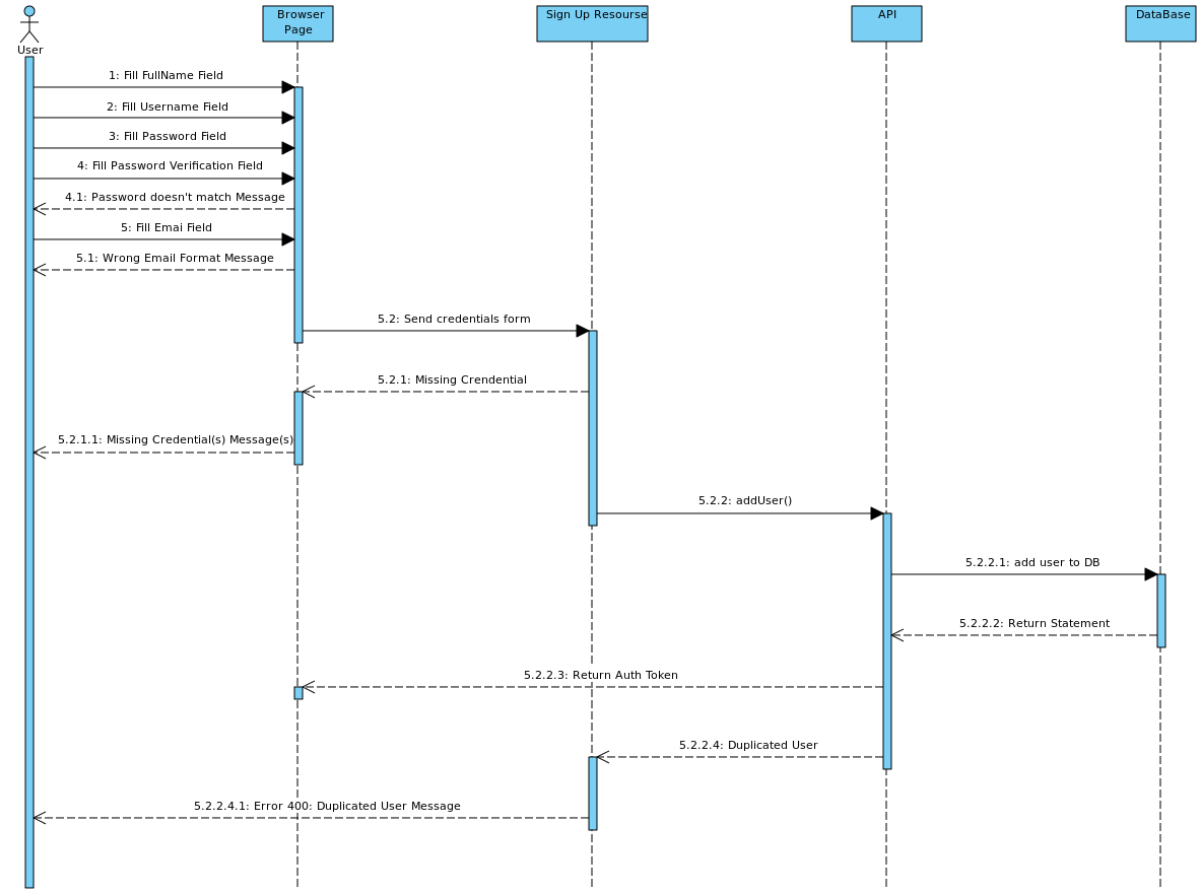
Βήμα 8: Αν έχει ξεχαστεί το password αλλαγή μέσω email.

Βήμα 9: Αν συμπληρωθούν σωστά(εγγεγραμμένος στη βάση) ο χρήστης συνδέεται και μεταφέρεται στο homepage αλλιώς βήμα 7

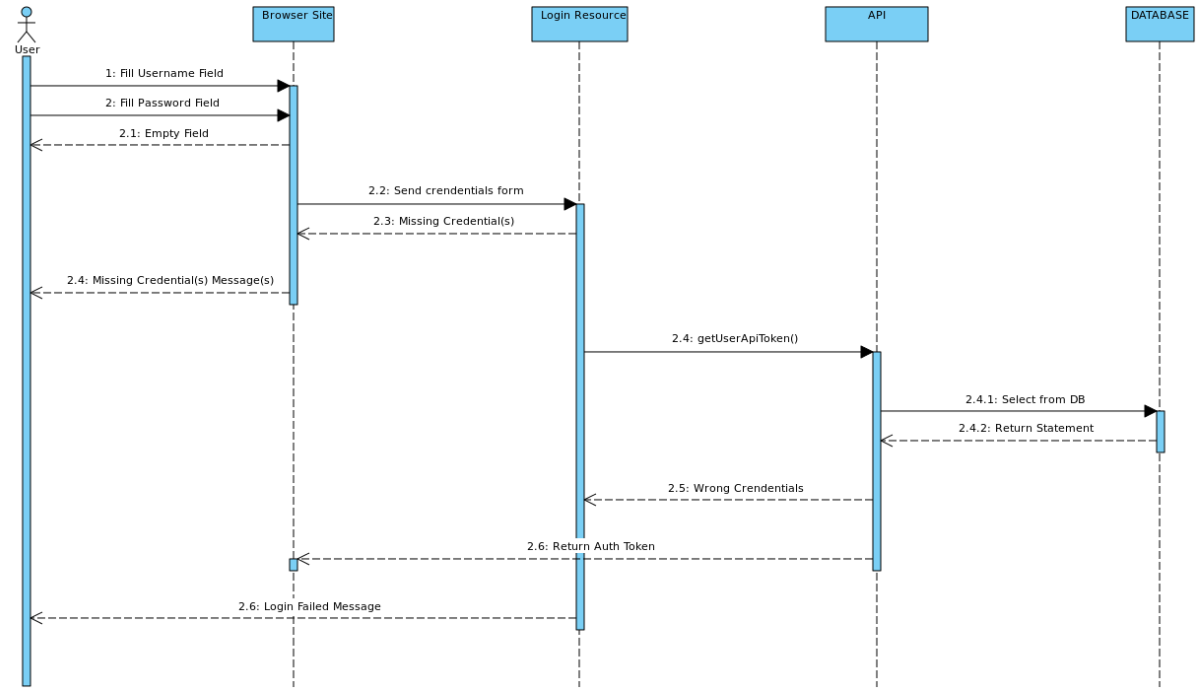
Στις εικόνες 7 και 8 παραθέτουμε διαγράμματα UML Sequence για τα Log In και Sign Up αντίστοιχα.

3.2.1.7 Δεδομένα εξόδου

Καθώς ο χρήστης για εγγραφή έχει προσθέσει τα στοιχεία του και κάνει apply τότε με ένα pop up παράθυρο χρειάζεται να επιβεβαιώσει ότι συμφωνεί με τους όρους χρήσης. Επίσης μετά την επιβεβαίωση λαμβάνει μήνυμα ότι η εγγραφή του πραγματοποιήθηκε επιτυχώς.



Εικόνα 8. User Sign Up



Εικόνα 7. User Log In

3.2.1.8 Παρατηρήσεις

Κάθε φορά που ένας χρήστης κάνει login, επιστρέφεται από το API στον φυλλομετρητή ιστού του χρήστη ένα μοναδικό αναγνωριστικό κλειδί (API token) το οποίο στέλνεται μαζί με κάθε αίτηση που ακολουθεί στο API προκειμένου να γίνεται αντιληπτή η σύνδεση του χρήστη. Σε περίπτωση που ο χρήστης δεν αποσυνδεθεί (log out), το αναγνωριστικό αυτό συνεχίζει να ισχύει (εγκυμονώντας πιθανώς κινδύνους ασφαλείας σε περίπτωση που μείνει αποθηκευμένο στον φυλλομετρητή ιστού και ο χρήστης δεν είναι ο μόνος χειριστής της πλατφόρμας). Για να ακυρωθεί το αναγνωριστικό αυτό πρέπει ο χρήστης να αποσυνδεθεί από το παρατηρητήριο κάνοντας log out.

3.2.2 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 2: Administrator *handles Delete Requests*

3.2.2.1 Χρήστες (ρόλοι) που εμπλέκονται

Αυτή η χρήση του API περιλαμβάνει τον administrator που θέλει να επιβεβαιώσει τα αιτήματα διαγραφής που εκκρεμούν.

3.2.2.2 Προϋποθέσεις εκτέλεσης

Για να κάνει log in ο aadministrator χρειάζεται να έχει έναν λογαριασμό στον οποίο έχουν δοθεί administrator δικαιώματα καθώς και να υπάρχουν ενεργά delete requests.

3.2.2.3 Περιβάλλον εκτέλεσης

Για την εκτέλεση αυτής της χρήσης χρειάζεται η διαδικτυακή διεπαφή χρήστη καθώς και η επικοινωνία του API με την βάση για την αλλαγή στοιχείων(από pending σε διαγραφή), την προβολή των pending requests ή επικύρωση στοιχείων (log in).

3.2.2.4 Δεδομένα εισόδου

Τα δεδομένα εισόδου από τον administrator περιλαμβάνουν :

1) *Check for delete: επιλογή των αιτημάτων για διαγραφή που εγκρίνονται από τον admin.*
2) *Log in: τα ατομικά στοιχεία του εγγεγραμμένου admin (username, password).*

Επικύρωση οριστικής διαγραφής

3.2.2.5 Παράμετροι

Με τη διαγραφή δεδομένων, ενδέχεται να προκύψει (προσωρινά) ασυνεπή μορφή στη βάση δεδομένων (π.χ. κατάστημα που δεν διαθέτει προϊόντα και δεν είναι withdrawn). Αυτό επιδιορθώνεται αυτόματα από το DBMS της βάσης μέσω trigger events που εξετάζουν τις περιπτώσεις ασυνέπειας μέσω κατηγορημάτων και διορθώνουν τη κατάσταση της βάσης (π.χ. θέτουν withdrawn = true για ένα κατάστημα που δεν πουλά προϊόντα).

3.2.2.6 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

Βήμα 1: Είσοδος στη homepage(guest).

Βήμα 2: Log In button.

Βήμα 3: (log in page) Σωστή εισαγωγή στοιχείων(username,password).

Βήμα 4: (admin page with dashboard) Επιλογή των (delete) requests.

Βήμα 5: (λίστα από pending requests) Επιλογή όλων η κάποιων από τα αιτήματα και συνέχεια.

Βήμα 6: (pop up) Αν δεν είσαι σίγουρος βήμα 5

Βήμα 7: Ενημέρωση λίστας μετά τελευταία επιλογή του admin

3.2.2.7 Δεδομένα εξόδου

Ο διαχειριστής λαμβάνει δεδομένα εξόδου όταν του ζητείται άδεια για να προχωρήσει σε διαγραφή από τη βάση όπως και όταν δώσει την άδεια μήνυμα για το αν πραγματοποιήθηκε η διαδικασία ή όχι σε περίπτωση σφάλματος.

3.2.3 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 3: User adds product

3.2.3.1 Χρήστες (ρόλοι) που εμπλέκονται

Αυτή η χρήση του API περιλαμβάνει τον χρήστη που θέλει να προσθέσει ένα προϊόν και έχει κάνει Sign up (πρόκειται δηλαδή για πιστοποιημένο χρήστη – crowdsourcer).

3.2.3.2 Προϋποθέσεις εκτέλεσης

Για να κάνει log in ο χρήστης χρειάζεται να έχει έναν λογαριασμό ήδη ώστε να έχει το δικαίωμα να μπορεί να προσθέσει προϊόν. Καθώς και το προϊόν που θέλει να προσθέσει να μην υπάρχει ήδη στη βάση (ίδιο προϊόν, ίδιο κατάστημα).

3.2.3.3 Περιβάλλον εκτέλεσης

Για την εκτέλεση αυτής της χρήσης χρειάζεται η διαδικτυακή διεπαφή χρήστη καθώς και η επικοινωνία του API με την βάση για την προσθήκη στοιχείων(εισαγωγή νέου προϊόντος /καταστήματος στη βάση), ή επικύρωση στοιχείων (log in).

3.2.3.4 Δεδομένα εισόδου

Τα δεδομένα εισόδου από τον administrator περιλαμβάνουν :

1) Εισαγωγή στοιχεία προϊόντος αν δεν υπάρχει ήδη σαν οντότητα (name,tags,picture,shop).
2) Log in: τα ατομικά στοιχεία του εγγεγραμμένου χρήστη (username, password).
3) Επικύρωση προσθήκης

Εισαγωγή στοιχείων καταστήματος αν δεν βρίσκεται στα υπάρχοντα (name,tags,location).

3.2.3.5 Παράμετροι

Πρέπει το προϊόν που εισάγεται να είναι μοναδικό (δηλαδή να μην υπάρχει ξανά με το ίδιο όνομα στο παρατηρητήριο). Σε διαφορετική περίπτωση, επιστρέφεται κατάλληλο μήνυμα σφάλματος.

3.2.3.6 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

Βήμα 1: Είσοδος στη homepage(guest).

Βήμα 2: Log In button.

Βήμα 3: (log in page) Σωστή εισαγωγή στοιχείων(username,password).

Βήμα 4: (homepage as logged in) Add a Product.

Βήμα 5: (find by name) Αν στην αναζήτηση υπάρχει ήδη βήμα

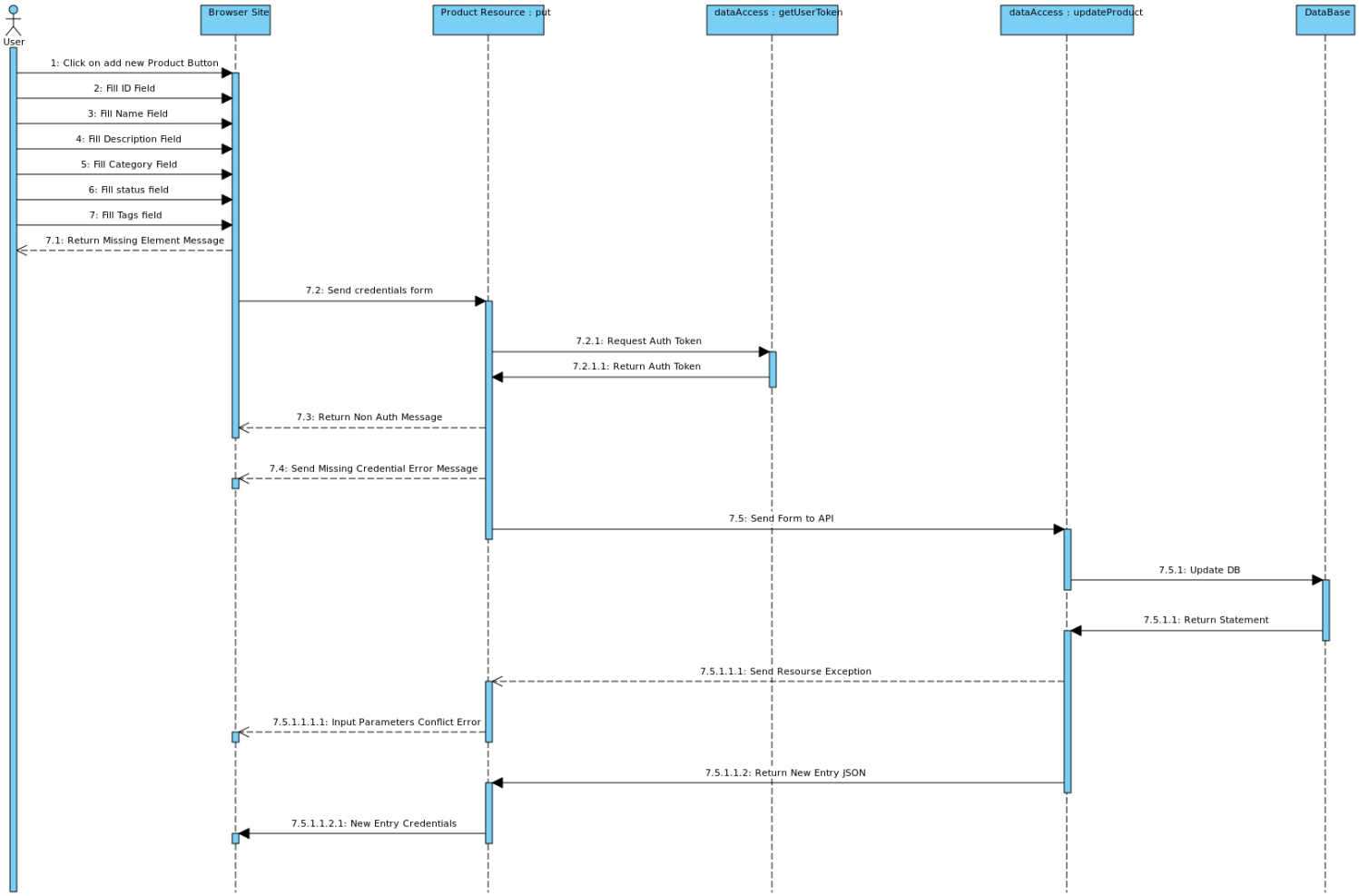
Βήμα 6: (product form) Συμπλήρωση στοιχείων προϊόντος. Αν στα καταστήματα υπάρχει το επιθυμητό τότε βήμα 8

Βήμα 7: (shop form) Συμπλήρωση στοιχείων καταστήματος και βήμα 6

Βήμα 8: Αν υπάρχει ταύτιση με άλλη καταχώριση εμφανίζεται κατάλληλο μήνυμα

Βήμα 9: Επιστροφή στο homepage

Στην εικόνα 9 παραθέτουμε UML Sequence διάγραμμα για τη καταχώρηση προϊόντος.



Εικόνα 9. UML Sequence: User Adds Product

3.2.3.7 Δεδομένα εξόδου

Ο χρήστης λαμβάνει δεδομένα εξόδου όταν του ζητείται άδεια για να προχωρήσει σε προσθήκη προϊόντος ή καταστήματος στη βάση όπως και όταν δώσει την άδεια μήνυμα για το αν πραγματοποιήθηκε η διαδικασία ή όχι σε περίπτωση ήδη καταχωρημένου προϊόντος (ιδιά τιμή, κατάστημα και προϊόν).

3.2.4 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 4: *Guest's Search for Product and Delete*

3.2.4.1 Χρήστες (ρόλοι) που εμπλέκονται

Αυτή η χρήση του API περιλαμβάνει τον χρήστη που θέλει να διαγράψει ένα συγκεκριμένο προϊόν καθώς και ο administrator ο οποίος θα παραλάβει ένα delete request.

3.2.4.2 Προϋποθέσεις εκτέλεσης

Ο χρήστης δεν χρειάζεται να έχει συνδεθεί (guest mode). Απλά να υπάρχει κάποιο προϊόν που κατά τη γνώμη του πρέπει να διαγραφεί.

3.2.4.3 Περιβάλλον εκτέλεσης

Για την εκτέλεση αυτής της χρήσης χρειάζεται η διαδικτυακή διεπαφή χρήστη καθώς και η επικοινωνία του API με την βάση για την αναζήτηση στοιχείων (προϊόν προς διαγραφή) και ενημέρωση λίστας διαγραφής για τους admins.

3.2.4.4 Δεδομένα εισόδου

Τα δεδομένα εισόδου από τον administrator περιλαμβάνουν :

1) Εισαγωγή στοιχείων για άμεση αναζήτηση προϊόντος. (tags, filters, allow location, name).
2) Εισαγωγή στοιχείων για έμμεση αναζήτηση προϊόντος (επιλογή από δημοφιλείς κατηγορίες).
3) Επικύρωση διαγραφής(προσθήκη λόγου αιτήματος διαγραφής).

3.2.4.5 Παράμετροι

Ο χρήστης εντοπίζει το προϊόν που τον ενδιαφέρει μέσω αναζήτησης. Μοναδική προϋπόθεση είναι να υπάρχει καταχωρημένο το προϊόν που τον ενδιαφέρει.

3.2.4.6 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

Βήμα 1: Είσοδος στη homepage(guest).

Βήμα 2: Για έμμεση αναζήτηση προϊόντος βήμα 4

Βήμα 3: Συμπλήρωση φόρμας αναζήτησης (όνομα και φίλτρα) Search

Βήμα 4: (List with map for product) Διαμόρφωση επιθυμητής απόστασης στο χάρτη(χρήστη-καταστημάτων)

Βήμα 5: (Refreshed list with map for product) Επιλογή προϊόντος για περισσότερες πληροφορίες.

Βήμα 6: (product page) Delete button

Βήμα 7: (pop up) Λόγος διαγραφής (προεπιλεγμένοι ή καινούργιος)

Βήμα 8: Επιβεβαίωση για ασφάλεια

Βήμα 9: Μεταφορά στο product page

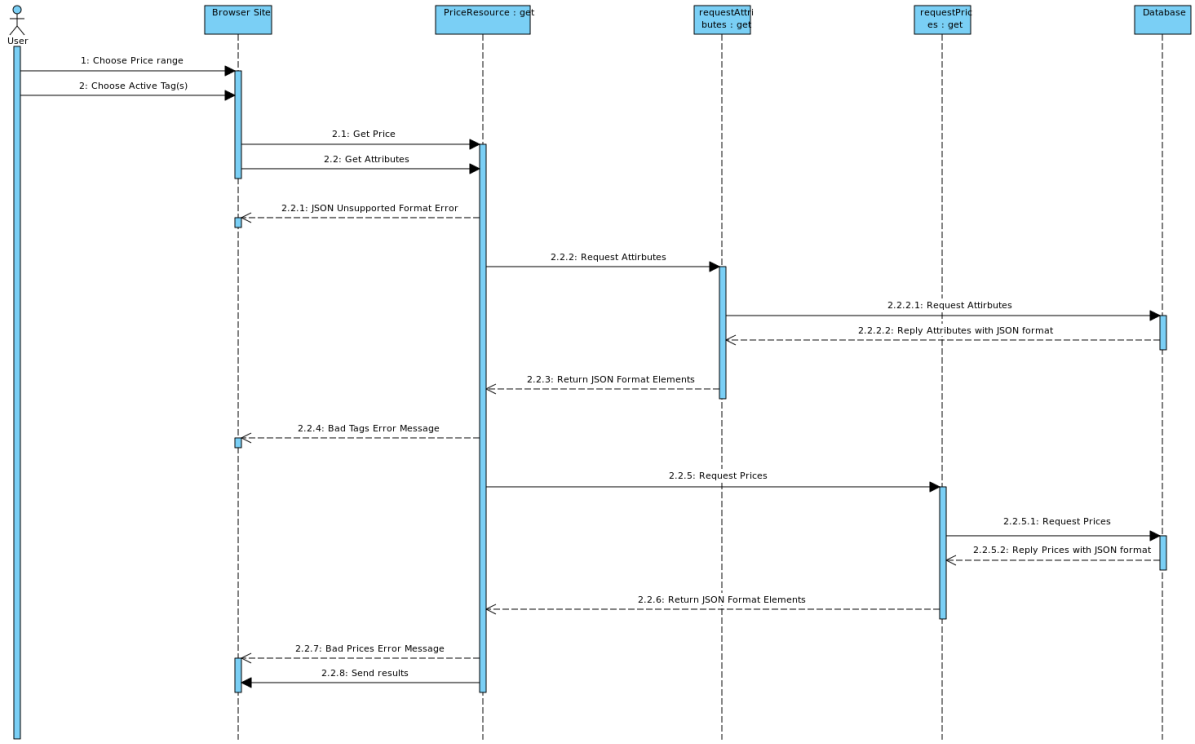
Στην εικόνα 10 παραθέτουμε διάγραμμα UML Sequence για την αναζήτηση τιμών από τον χρήστη βάσει των product tags.

3.2.4.7 Δεδομένα εξόδου

Ο χρήστης λαμβάνει δεδομένα εξόδου όταν του εμφανίζεται pop up παράθυρο για να παραθέσει τον λόγο που θέλει να διαγραφεί ένα προϊόν καθώς και αφού επιβεβαιώσει αυτό το request ότι μεταβιβάστηκε στους admins ή όχι σε περίπτωση σφάλματος.

3.2.4.8 Παρατηρήσεις

Ο πιστοποιημένος χρήστης, λόγω της πολιτικής που ακολουθούμε για τη προστασία των δεδομένων του παρατηρητηρίου από λανθασμένη ή κακόβουλη χρήση, δεν έχει δικαίωμα να διαγράψει πλήρως τα στοιχεία που είναι καταχωρημένα στη βάση δεδομένων. Η επιλογή διαγραφής ενός προϊόντος ή καταστήματος έχει ως συνέπεια να τεθεί το αντίστοιχο πεδίο withdrawn ως true και να ειδοποιηθεί ένας διαχειριστής προκειμένου να επιβεβαιώσει τη διαγραφή (ή να την αναιρέσει) και τότε να διαγραφεί πραγματικά η καταχώρηση (ή να επιστρέψει στη προηγούμενη κατάστασή της με withdrawn = false).



Εικόνα 10. UML Sequence: User Search for Prices based on Product Tags

3.3 Απαιτήσεις επιδόσεων

Το σύστημά μας διαθέτει βάση δεδομένων η οποία περιέχει πληροφορίες για τους χρήστες, τα καταστήματα, τα προϊόντα και τις τιμές, καθώς και API και UI, ρόλος των οποίων είναι η παροχή πρόσβασης χρηστών, τρίτων εφαρμογών και γενικά εξωτερικών συστημάτων στις λειτουργίες του παρατηρητηρίου. Με βάση, λοιπόν, την αρχιτεκτονική αυτή, τα κύρια μεγέθη που αφορούν το λογισμικό μας είναι τα εξής δύο:

- Το μέγεθος των δεδομένων που είναι αποθηκευμένα στη βάση μας και αφορούν χρήστες, καταστήματα, προϊόντα και τιμές (το οποίο μέγεθος μετράται σε GB προς το παρόν και λογικά σε TB στο μέλλον)
- Το πλήθος των ενεργών συνδέσεων με το API (είτε από το δικό μας UI, είτε από τρίτες εφαρμογές)
- Η ταχύτητα ανταλλαγής δεδομένων μέσω αυτών των συνδέσεων (από και προς το API, από και προς τη βάση δεδομένων)

Έχοντας, οπότε, υπόψη μας τα παραπάνω βασικά μεγέθη, κρίσιμες μετρικές της επιθυμητής απόδοσης του λογισμικού μας είναι οι ακόλουθες:

- Η αύξηση του μεγέθους των δεδομένων μας με το χρόνο (ή, καλύτερα, το ποσοστό αύξησης π.χ. ανά εβδομάδα, μήνα και έτος), η οποία μας βοηθά στη κατανόηση των τεχνικών αναγκών της εφαρμογής.
- Χρόνος απόκρισης σε ένα αίτημα χρήστη σε ms. Είναι πολύ σημαντικό ο συνολικός χρόνος επεξεργασίας, καθώς και το latency του δικτύου να μην επηρεάζουν αρνητικά την εμπειρία του χρήστη.
- Throughput εξυπηρέτησης αιτημάτων σε βάση δεδομένων και API. Ενδιαφέρει τόσο το πλήθος των αιτημάτων όσο και το μέγεθός τους (επιτρέπονται κλήσεις στο API που απαιτούν πολλή επεξεργασία και επιστρέφουν μεγάλο πλήθος δεδομένων).
- Η κλιμακωσιμότητα, θέλουμε δηλαδή η οποιαδήποτε αύξηση των δεδομένων λόγω της δραστηριότητας της εφαρμογής να αντισταθμίζεται γραμμικά από μία αντίστοιχη οριζόντια αύξηση των διαθέσιμων πόρων.
- Το πλήθος των σφαλμάτων που επιστρέφουν η βάση και το API, καθώς και το ποσοστό σφαλμάτων σε σχέση με το συνολικό πλήθος των αιτημάτων, με τη πάροδο του χρόνου. Περαιτέρω διάκριση των σφαλμάτων είναι επίσης σημαντική για τη συντήρηση αλλά και για τη προστασία του συστήματος

από λογικά λάθη της εφαρμογής, λανθασμένη χρήση ή ακόμα και κακόβουλες επιθέσεις.

- Η διεκπεραιωτική ικανότητα της εφαρμογής, και, πιο συγκεκριμένα, τόσο το bandwidth του ίδιου του API, όσο και της βάσης, καθώς τα δύο αυτά δεν είναι άρρηκτα συνδεδεμένα. Πιο συγκεκριμένα, για κάθε αίτημα στη βάση, σίγουρα προηγήθηκε ένα τουλάχιστον αίτημα στο API (αφού η βάση δεν είναι άμεσα προσβάσιμη), αλλά δεν είναι απαραίτητο ότι για κάθε αίτημα στο API θα γίνεται και ένα αντίστοιχο αίτημα στη βάση (λόγω περιπτώσεων όπως σφάλματα, caching κλπ.)

3.4 Απαιτήσεις οργάνωσης δεδομένων

3.4.1 Τεχνική περιγραφή των δεδομένων που διαχειρίζεται το λογισμικό και των σχετικών μετρικών φορτίου δεδομένων εισόδου, επεξεργασίας κ.λπ.

Όπως είναι αναμενόμενο, τα δεδομένα που διαχειρίζεται το λογισμικό μας είναι κατά κύριο λόγο (αποκλειστικά, αν δεν πάρουμε υπόψη μας και προσωρινά δεδομένα που προφανώς διατηρούν και ανταλλάσσουν για τη λειτουργία τους τα επιμέρους components του λογισμικού) τα δεδομένα που τελικά πρόκειται να αποθηκευτούν στη βάση δεδομένων μας (SQL database).

Η δομή αυτών των δεδομένων, έτσι όπως την έχουμε σχεδιάσει, φαίνεται αναλυτικότερα στο E-R διάγραμμα της υποενότητας 3.4.3. Εδώ, επισημαίνουμε απλώς για διευκόλυνση αφαιρετικά τις βασικές οντότητες - άξονες πάνω στους οποίους στηρίζεται το μοντέλο δεδομένων μας, και που ίσως μας βοηθήσουν στην προδιαγραφή των χωρικών απαιτήσεων του λογισμικού μας:

- *Δεδομένα σχετικά με κάθε εγγεγραμμένο χρήστη*
- *Δεδομένα σχετικά με το κάθε κατάστημα*
- *Δεδομένα σχετικά με το κάθε προϊόν*
- *Δεδομένα με κάθε καταχώρηση που γίνεται από τους crowdsourcers για κάποιο προϊόν σε κάποιο κατάστημα*
- *Για κάθε χρήστη, αποθήκευση του συνόλου των προϊόντων τα οποία έχει σημειώσει ως αγαπημένα*

Τα δεδομένα εισόδου παρέχονται στο λογισμικό μας από τρεις βασικές πηγές:

- *Από τους crowdsourcers, έχουμε καταχωρήσεις νέων προϊόντων, καταστημάτων ή τιμών μέσω του interface που τους προσφέρουμε με το browser UI μας, αλλά και μεταβολή των υπαρχόντων.*
- *Από τον απλό (εγγεγραμμένο) χρήστη, το λογισμικό μέσω του UI λαμβάνει ή/και μεταβάλλει υπάρχοντα δεδομένα που αφορούν τον ίδιο (όνομα, κωδικός, αγαπημένα κ.λπ.)*
- *Τέλος, καταχωρήσεις και αλλαγές για όλα τα παραπάνω μπορούν να γίνουν και από τρίτες εφαρμογές, κάτι που από την πλευρά μας το βλέπουμε ως απευθείας είσοδο δεδομένων μέσω του RESTful API που παρέχει το λογισμικό, χωρίς να παρεμβάλλεται το δικό μας UI.*

Τέλος, κάποιες βασικές μετρικές που μας ενδιαφέρουν για τη διαδικασία του storage capacity planning είναι οι ακόλουθες:

1) Πρώτα από όλα, το μέγεθος των δεδομένων που είναι αποθηκευμένα στη βάση. Πρόκειται για μία σημαντική μετρική, η οποία είναι εύκολο να μετράται ανά τακτά χρονικά διαστήματα και η εξέλιξή της δίνει μάλλον μια καλή εικόνα για την χρονική εξέλιξη των απαιτήσεων του λογισμικού σε αποθηκευτικό χώρο, ώστε η επέκταση του αποθηκευτικού χώρου να γίνεται με όσο το δυνατόν μικρότερο κόστος. Η τακτική μέτρηση του μεγέθους των δεδομένων μας, ή ισοδύναμα της αύξησής τους, είναι σίγουρα απαραίτητη, ιδιαίτερα κατά το πρώτο διάστημα λειτουργίας της εφαρμογής.

2) Το μέγεθος των δεδομένων που (πιθανότατα) είναι παλιά / άκαιρα. Η μετρική αυτή δίνει μια αρκετά καλή ένδειξη για το πότε θα ήταν μία καλή στιγμή να γίνει κάποιου είδους ξεκαθάρισμα των δεδομένων που έχει συγκεντρώσει το λογισμικό, και φυσικά το ποια δεδομένα μπορούν να απαλειφθούν. Πρακτικά, η μέτρηση αυτή ίσως να μην είναι πολύ εύκολη, αλλά μπορεί να γίνει με μεθόδους όπως π.χ. ο έλεγχος ποια προϊόντα έχουν να προσπελαστούν πάνω από 6 μήνες, το απευθείας feedback από τους χρήστες κ.λπ.

3) Ο εντοπισμός εσφαλμένων δεδομένων. Άλλος ένας πολύ σημαντικός παράγοντας, πρόκειται για μία κάπως δύσκολη διαδικασία, η οποία όμως μπορεί αν γίνεται συνεχώς, μέσω του feedback των χρηστών.

3.4.2 Απαιτήσεις και περιορισμοί πρόσβασης σε δεδομένα

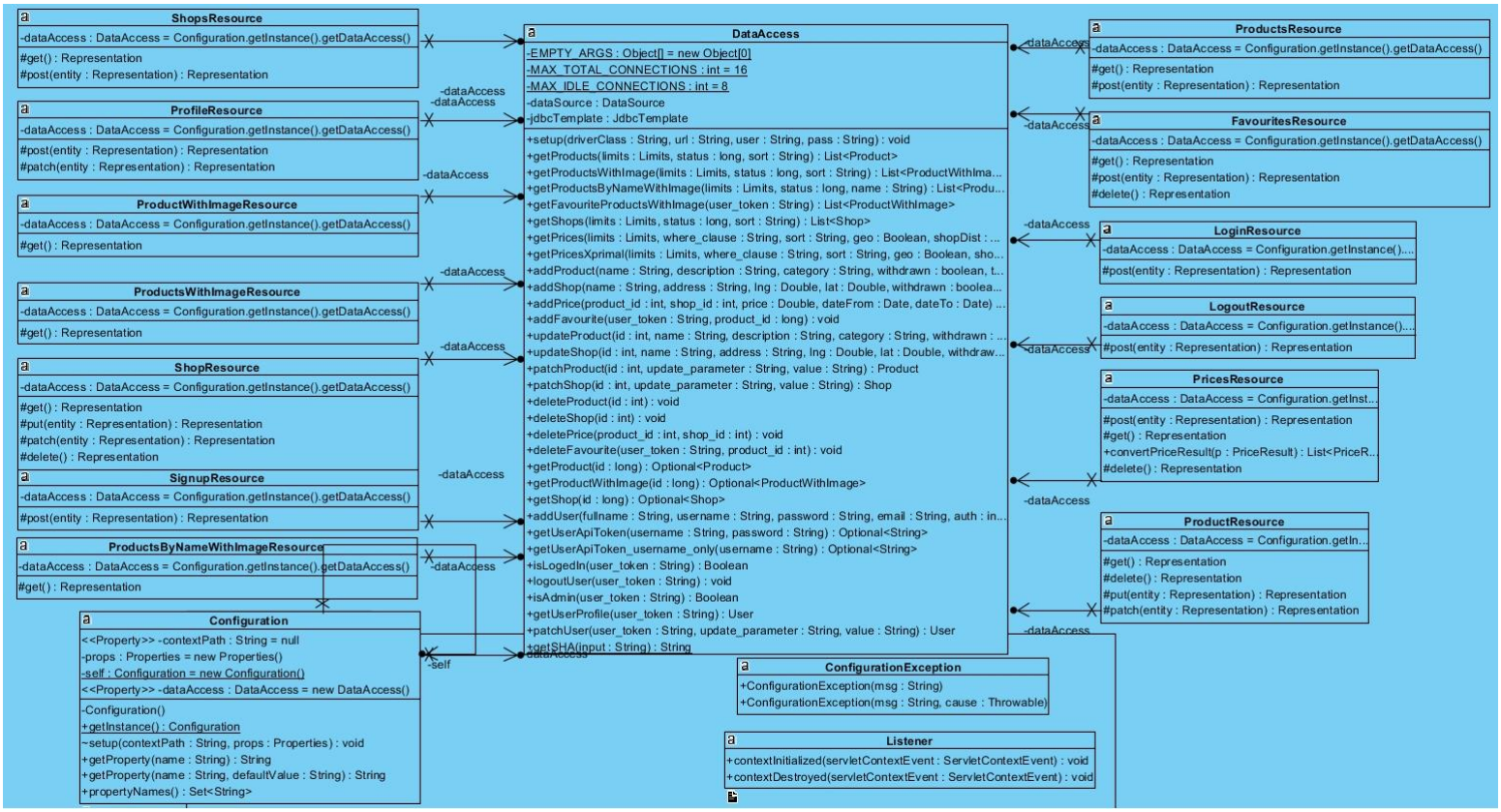
Αναφορικά με απαιτήσεις και περιορισμούς πρόσβασης στα δεδομένα τα οποία θα διαχειρίζεται το λογισμικό μας, έχουμε να σημειώσουμε τα εξής:

- Έχοντας αποφασίσει τη σχεδίαση και υλοποίηση του RESTful API, έχουμε ήδη προδιαγράψει ότι κανένας εξωτερικός χρήστης ή λογισμικό δεν θα έχει άμεση / ελεύθερη πρόσβαση στα δεδομένα που διατηρεί το λογισμικό, αλλά όλες οι ανταλλαγές δεδομένων θα περνούν από το API.

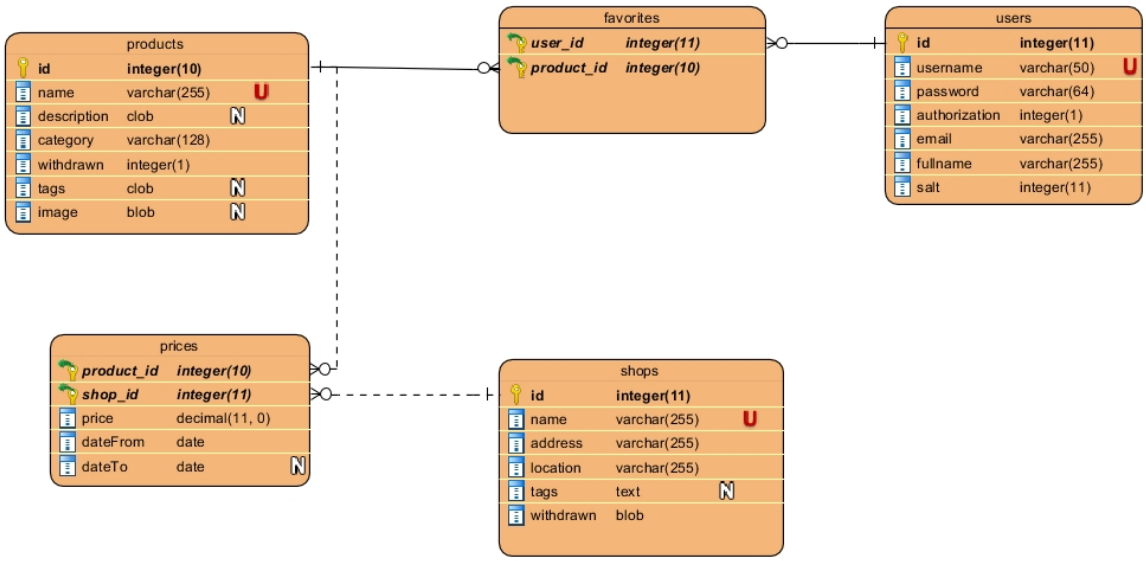
Όπως είναι λογικό, δεν θέλουμε να υπάρχει πρόσβαση στα δεδομένα που αφορούν έναν χρήστη από οποιονδήποτε άλλο, και για αυτό το σκοπό οι πληροφορίες για κάθε χρήστη περιλαμβάνουν email και μυστικό κωδικό.

3.4.3 Μοντέλο δεδομένων (μοντέλο κλάσεων UML και μοντέλο ER)

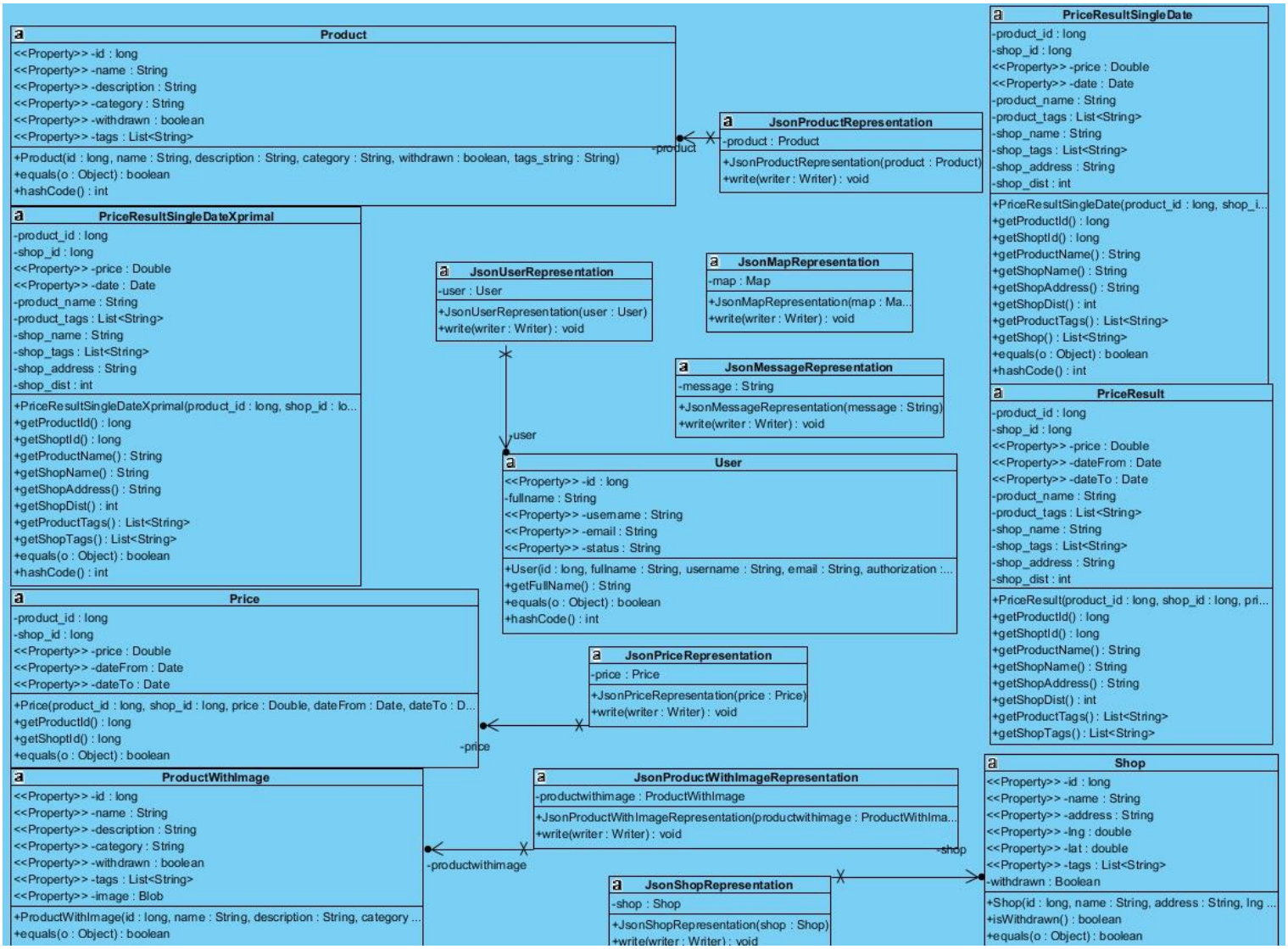
Στις εικόνες 11 με 13 παραθέτουμε τα διαγράμματα UML Class (εικόνες 11,13 και 14) & ER (εικόνα 12). Στο πρώτο γίνεται αντιληπτή η προγραμματιστική δομή του API και στο δεύτερο διακρίνεται το σχεσιακό σχήμα (το οποίο είναι άρρηκτα συνδεδεμένο με το μοντέλο οντοτήτων – συσχετίσεων) της βάσης δεδομένων του συστήματός μας.



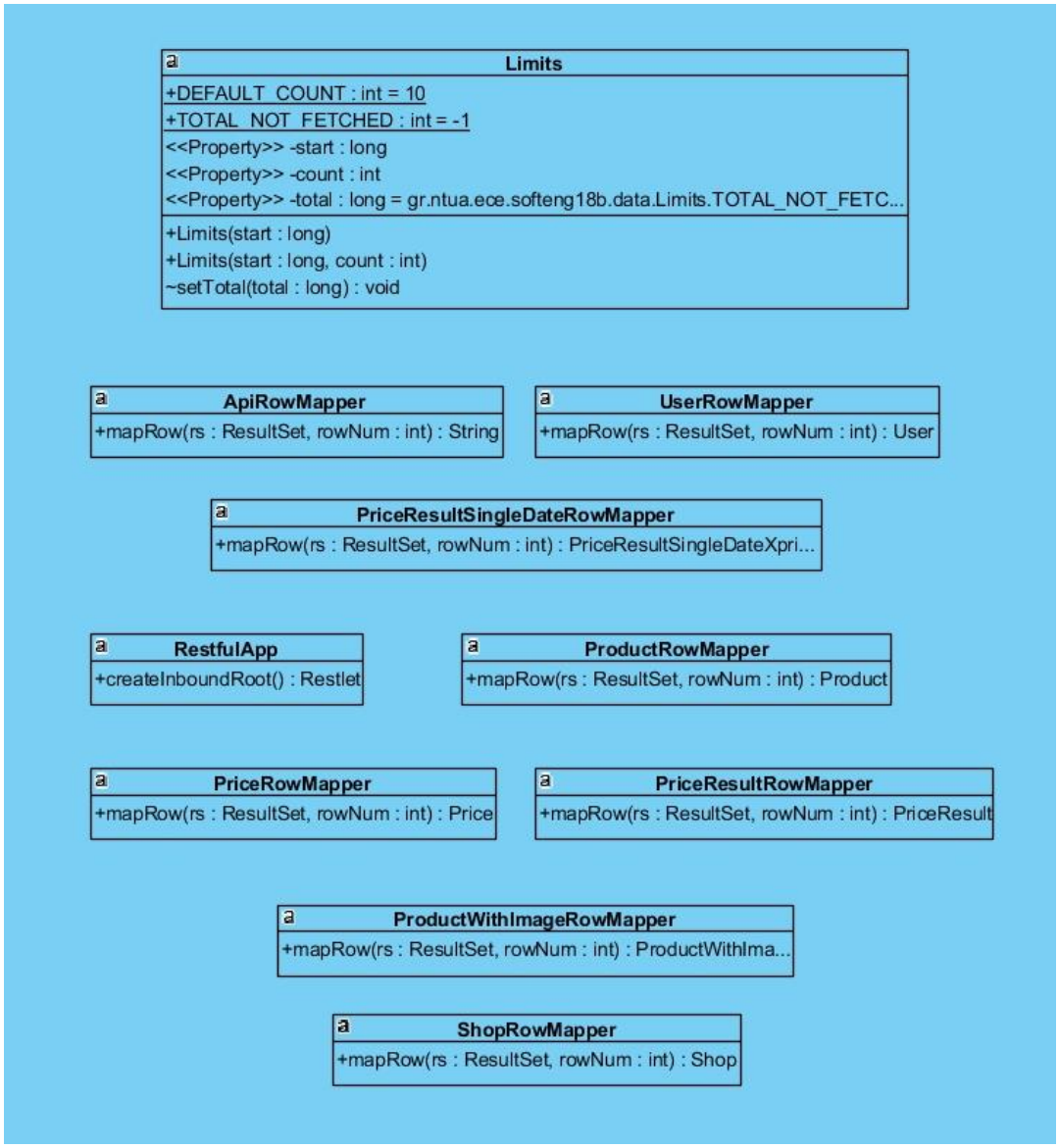
Εικόνα 11. UML Class Part I: API Resource classes integrating with DataAccess class



Εικόνα 12. UML ERD. Δομή του Database Schema



Εικόνα 13. UML Class Part II



Εικόνα 14. UML Class Part III

3.4.4 Προδιαγραφές ακεραιότητας δεδομένων

Από τη στιγμή που η εφαρμογή μας έχει σαν βασικό σκοπό τη σωστή και έγκυρη ενημέρωση των χρηστών σχετικά με τις τιμές των ηλεκτρονικών προϊόντων στα διάφορα καταστήματα, και με βάση και το μοντέλο δεδομένων μας (το οποίο παρουσιάζεται στην υποενότητα 3.4.3), οι βασικές ιδιότητες που πρέπει να υφίστανται ώστε να μπορούμε να ισχυριστούμε την ακεραιότητα και εγκυρότητα των δεδομένων μας είναι οι ακόλουθες:

- Οι πληροφορίες που διατηρούμε σχετικά με τα προϊόντα, τα καταστήματα, και τις τιμές για κάθε προϊόν σε κάθε κατάστημα να είναι αληθείς.
- Όσον αφορά την ακεραιότητα των δεδομένων, όπως είναι λογικό, απαιτούμε για κάθε μας τύπο οντότητας (χρήστη, προϊόν, κατάστημα) μοναδικά ονόματα και αναγνωριστικά, για την μονοσήμαντη αναγνώρισή τους, και γενικότερα, απαιτούμε να ισχύουν οι περιορισμοί εκείνοι τους οποίους έχουμε ορίσει κατά την κατασκευή της βάσης μας.

Τα παραπάνω, για να επιτευχθούν, απαιτείται όσο το δυνατόν συνεχής επιτήρηση, ενώ βοηθάει και ο ρόλος των administrators, οι οποίοι μπορούν να ελέγχουν αν είτε υπάρχουσες είτε νέες καταχωρήσεις πληρούν τις προϋποθέσεις εγκυρότητας.

3.4.5 Προδιαγραφές διατήρησης δεδομένων

Εδώ, θα περιγράψουμε πλέον τις απαιτήσεις τις οποίες θα θέλαμε να πληροί το λογισμικό, ως προς τα δεδομένα του, σε μεγαλύτερο βάθος χρόνου. Πιο συγκεκριμένα, κάποιες ιδιότητες που θα θέλαμε να υφίστανται σε μεγάλο χρονικό ορίζοντα είναι οι εξής:

- Απαιτείται η απομάκρυνση παρωχημένων πληροφοριών. Για παράδειγμα η τιμή ενός προϊόντος σε ένα κατάστημα που αφορά μια χρονική περίοδο πάνω από 1 χρόνο νωρίτερα από την τρέχουσα χρονική στιγμή μας είναι αδιάφορη και άχρηστη για τους σκοπούς του λογισμικού μας, και απλά συμβάλλει στη σύγχυση.
- Αποφυγή προσθήκης στη βάση διαφορετικών καταχωρήσεων για το ίδιο κατάστημα / προϊόν / τιμή προϊόντος. Εδώ εμπίπτουν, βέβαια, και περιπτώσεις όπου δύο καταχωρήσεις από δύο π.χ. crowdsourcers, οι οποίες διαφέρουν ελαφρώς, αλλά αφορούν στην πραγματικότητα στην ίδια οντότητα του πραγματικού κόσμου. Σε όλες αυτές τις περιπτώσεις, η βάση διογκώνεται χωρίς λόγο, και δυσχεραίνεται η εμπειρία των χρηστών.
- Παρά τα παραπάνω, απαιτούμε παράλληλα οι χρήσιμες πληροφορίες και δεδομένα της βάσης μας να μην χάνονται με το πέρασμα του χρόνου, αν δεν υπάρχει κάποιος λόγος να χαθούν. Μία τιμή για ένα προϊόν, για παράδειγμα, όσο είναι έγκυρη και σωστή, θέλουμε να παραμένει στη βάση μας, ανεξαρτήτως του χρόνου που έχει περάσει.

Τα παραπάνω, για να επιτευχθούν, απαιτείται όσο το δυνατόν συνεχής επιτήρηση, ενώ βοηθάει και ο ρόλος των administrators, οι οποίοι μπορούν να ελέγχουν αν είτε υπάρχουσες είτε νέες καταχωρήσεις πληρούν τις προϋποθέσεις εγκυρότητας

3.5 Περιορισμοί σχεδίασης

Α. Πρώτα από όλα, υπάρχουν κάποιοι περιορισμοί στη σχεδίαση τόσο του back end, αλλά και του front end της εφαρμογής, οι οποίοι προκύπτουν ως αποτέλεσμα της απαίτησής μας για παροχή ενός RESTful API το οποίο να πληροί τις προδιαγραφές που μας έχουν δοθεί.

Από τη μία πλευρά, το back end είναι υποχρεωμένο να υλοποιεί όλα τα απαραίτητα endpoints, ώστε να δίνει "προς τα έξω" ένα ολοκληρωμένο interface για πλήρη και αποδοτική πρόσβαση στα δεδομένα της βάσης, το οποίο ταυτόχρονα να συμμορφώνεται και στο stateless πρωτόκολλο του REST

Και από την άλλη πλευρά, πρέπει να σημειώσουμε ότι, παρόλο που και αυτό είναι δική μας σχεδιαστική επιλογή, η αρχιτεκτονική στην οποία το browser UI μας επικοινωνεί και αυτό με το RESTful API (η οποία έχει αιτιολογηθεί σε άλλη ενότητα), και όχι άμεσα με τη βάση, επιφορτίζει το front end με ένα κάπως πιο δύσκολο έργο, αφού δεν έχει την πλήρη ελευθερία που θα είχε αν επικοινωνούσε άμεσα με τη βάση μέσω SQL queries.

Β. Εδώ θα αναφερθούμε στους περιορισμούς σχεδίασης που προκύπτουν από την επιλογή χρήσης του gradle ως build automation tool. Συγκεκριμένα, η επιλογή αυτή επιβάλλει περιορισμούς τόσο στη δομή των directories και των αρχείων από τα οποία αποτελείται το project μας, τα οποία πρέπει να βρίσκονται σε συγκεκριμένες θέσεις (με κάποιο βαθμό ελευθερίας) για να λειτουργήσει σωστά το εργαλείο, όσο και (σε κάποιο βαθμό) και στην ονοματολογία τους, δεδομένου ότι τα ονόματα ορισμένων φακέλων ή/και αρχείων πρέπει να είναι συγκεκριμένα ή να συμμορφώνονται σε κάποια πρότυπα. Για αναλυτική αναφορά στους περιορισμούς αυτούς, παραπέμπουμε και στο documentation του gradle.

Γ. Τέλος, η χρήση της SQL (και, πιο συγκεκριμένα, της MySQL) για την υλοποίηση της βάσης δεδομένων επιβάλλει και αυτή κάποιους περιορισμούς, οι βασικότεροι από τους οποίους έχουν να κάνουν με το ότι πρόκειται για μία σχεσιακή βάση δεδομένων. Σαν αποτέλεσμα, η βάση μας είναι λίγο πολύ υποχρεωμένη να περιέχει έναν αριθμό "προτύπων" (patterns - οι πίνακες), πολύ μικρό σε σχέση με το πλήθος

των δεδομένων, και έτσι οι εγγραφές για χρήστες, για παράδειγμα, θα έχουν ένα σημαντικό βαθμό ομοιομορφίας. Γενικότερα, μια σχεσιακή βάση επιβάλλει κάποια σχετικά αυστηρή δομή στα δεδομένα της.

Και εδώ, παραπέμπουμε για μεγαλύτερη λεπτομέρεια στο documentation της SQL (MySQL).

3.6 Λοιπές απαιτήσεις

3.6.1 Απαιτήσεις διαθεσιμότητας λογισμικού

Λόγω της φύσης του λογισμικού μας, και του σκοπού που αυτό εξυπηρετεί, είναι λογικό οι διάφοροι χρήστες μας να αναμένουν μεγάλη ή και πλήρη διαθεσιμότητα, ιδιαίτερα από το API, το οποίο είναι η "καρδιά" της εφαρμογής μας και κατά πάσα πιθανότητα θα χρησιμοποιείται και από τρίτες εφαρμογές, αλλά και από το UI μας σε κάποιο βαθμό, αφού αποτελεί ένα πλήρες Web App, με ό,τι αυτό συνεπάγεται για τις απαιτήσεις διαθεσιμότητάς του.

Ας εξετάσουμε αρχικά τους λόγους για τους οποίους θα είναι αναπόφευκτο ανά απροσδιόριστα ακόμα χρονικά διαστήματα το "downtime" της εφαρμογής.

Δύο είναι τα βασικά ενδεχόμενα. Πρώτον, η τακτική και προγραμματισμένη συντήρηση του λογισμικού, την οποία αναλύουμε περαιτέρω στην ενότητα 3.6.3. Και, δεύτερον, υπάρχουν και άλλοι, αστάθμητοι παράγοντες που μπορεί να προκαλέσουν τη μη διαθεσιμότητα της εφαρμογής, όπως σφάλμα του hosting Server, συμφόρηση στο δίκτυο κ.λπ., και οι οποίοι θεωρούμε ότι είναι αντιμετωπίσιμοι σε εύλογο χρονικό διάστημα.

Ας σημειώσουμε, βέβαια, εδώ, ότι χάρη στις πλήρως ανεξάρτητες υλοποιήσεις μας για το REST API και το browser UI, είναι και οι διαθεσιμότητές τους ανεξάρτητες. Δηλαδή, μπορεί κάλλιστα να είναι διαθέσιμο το API, και μη διαθέσιμο το UI. Αλλά και επιπλέον, παρόλο που, φυσικά, το front end αντλεί δεδομένα από το API, το συνολικό interface του UI με το χρήστη μπορεί να είναι πλήρως λειτουργικό, ακόμα κι αν δεν είναι διαθέσιμο το back end (στην οποία περίπτωση, πολλές σελίδες θα έχουν ως έξοδο σφάλμα του Server του back end, ωστόσο το front end κατά τα άλλα θα δουλεύει).

Με βάση τα παραπάνω, από την πλευρά μας έχουμε το εξής πλάνο:

- Για τους πρώτους δύο μήνες λειτουργίας της εφαρμογής, μέγιστο downtime 20 ώρες το δεκαπενθήμερο
- Για το πρώτο εξάμηνο λειτουργίας της εφαρμογής, μέγιστο downtime 10 ώρες την εβδομάδα
- Μετά το πρώτο εξάμηνο λειτουργίας της εφαρμογής, μέγιστο downtime 8 ώρες κάθε 5 μέρες

3.6.2 Απαιτήσεις ασφάλειας

Με βάση το μοντέλο των δεδομένων μας (το οποίο έχει αναπτυχθεί σε άλλη υποενότητα), μπορούμε να διαπιστώσουμε ότι τα μόνα ευαίσθητα δεδομένα τα οποία διαχειρίζεται το λογισμικό μας είναι τα στοιχεία των χρηστών: όνομα, email, αγαπημένα, και γενικότερα ο λογαριασμός του κάθε χρήστη σαν συνολική υπόσταση.

Στα δεδομένα αυτά, και στον κάθε λογαριασμό, θέλουμε να έχει πρόσβαση μόνο ο αντίστοιχος χρήστης, μέσω του μυστικού κωδικού (password) που μας έχει δώσει κατά τη διαδικασία εγγραφής του (sign up).

Για το σκοπό αυτό, έχουμε εφαρμόσει στο λογισμικό μας τις ακόλουθες, ευρύτερα γνωστές, τακτικές:

- Διατήρηση του hash κάθε κωδικού στη βάση μας, και όχι του ίδιου του κωδικού, ώστε ο κωδικός του χρήστη να είναι ασφαλής από επιθέσεις στη βάση.

- Επικοινωνία που περιλαμβάνει ανταλλαγή οποιωνδήποτε ευαίσθητων πληροφοριών / δεδομένων γίνεται μέσω ασφαλούς σύνδεσης (SSL). Στη φάση του development, αυτό γίνεται με χρήση self-signed ceritficate, αλλά στο μέλλον σε περιβάλλον production, στόχος είναι να γίνει μέσω CA signed certificate.

Με την υλοποίηση των παραπάνω στρατηγικών, μπορούμε να εγγυηθούμε σε αρκετά μεγάλο βαθμό την ασφάλεια των δεδομένων των χρηστών μας.

3.6.3 Απαιτήσεις συντήρησης

Όσον αφορά την συντήρηση του λογισμικού, υπάρχουν δύο βασικοί άξονες:

1) Η συνεχής επιτήρηση του λογισμικού, για τον εντοπισμό σφαλμάτων. Ίσως, μάλιστα, να είναι απαραίτητη και η προσωρινή διακοπή της λειτουργίας της εφαρμογής σε περίπτωση σοβαρού σφάλματος, το οποίο δεν μπορεί να διορθωθεί όσο η εφαρμογή είναι Live. Αυτό μπορεί να συμβεί τόσο στο back end όσο και στο front end της εφαρμογής.

2) Γενικές εργασίες συντήρησης που είναι πιθανό να χρειάζονται, όπως για παράδειγμα ένας ο έλεγχος από developers ή/και administrators για την εγκυρότητα των δεδομένων της εφαρμογής, καθώς από της φύση της, και ειδικά τον πρώτο καιρό λειτουργίας, όπου μάλλον δε θα υπάρχουν ακόμα έμπιστοι crowdsourcers, είναι πολύ πιθανή η συσσώρευση άκυρων πληροφοριών και δεδομένων.