Έγγραφο απαιτήσεων λογισμικού (SRS)

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

Smartgas

1. Εισαγωγή

1.1 Εισαγωγή: σκοπός του λογισμικού

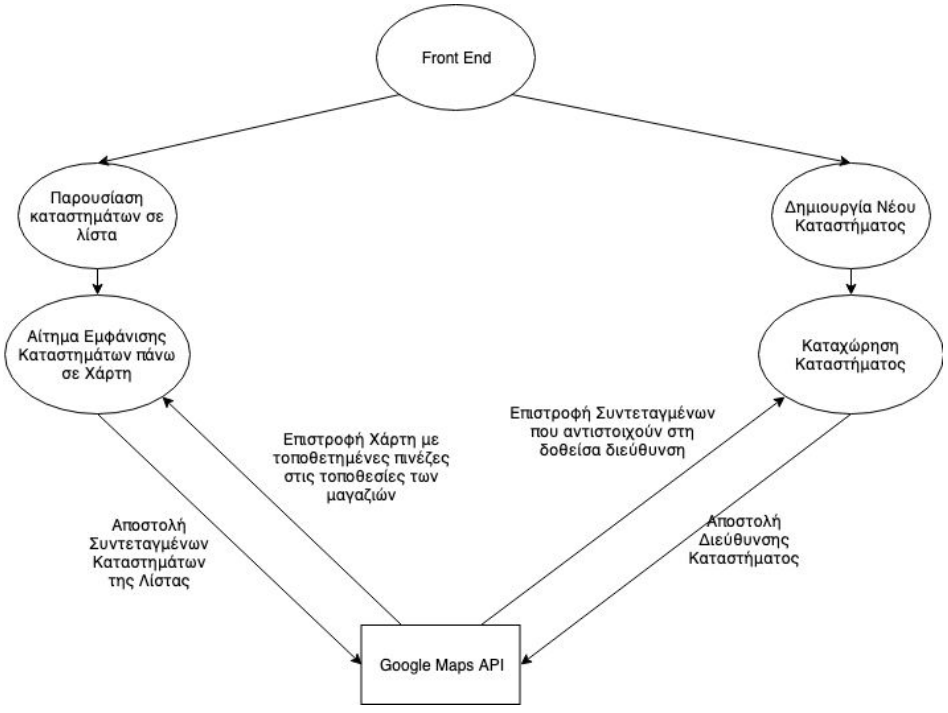
Το **Smartgas** αποτελεί μια εφαρμογή, η οποία έχει τον ρόλο ενός online παρατηρητηρίου τιμών υγρών καυσίμων. Ο στόχος της εφαρμογής μας είναι να δώσει τη δυνατότητα στον κόσμο να αναζητεί, να καταχωρεί και να ενημερώνεται για τις τιμές των καυσίμων στα πρατήρια της χώρας. Ως εκ τούτου, δίνεται η δυνατότητα στο καταναλωτικό κοινό να αποκτήσει την πληροφόρηση που χρειάζεται για να επιλέξει πιο σωστά το πρατήριο στο οποίο θα ανεφοδιάσει το όχημα του.

1.2 Επισκόπηση του λογισμικού

1.2.1 Διεπαφές με εξωτερικά συστήματα και εφαρμογές λογισμικού

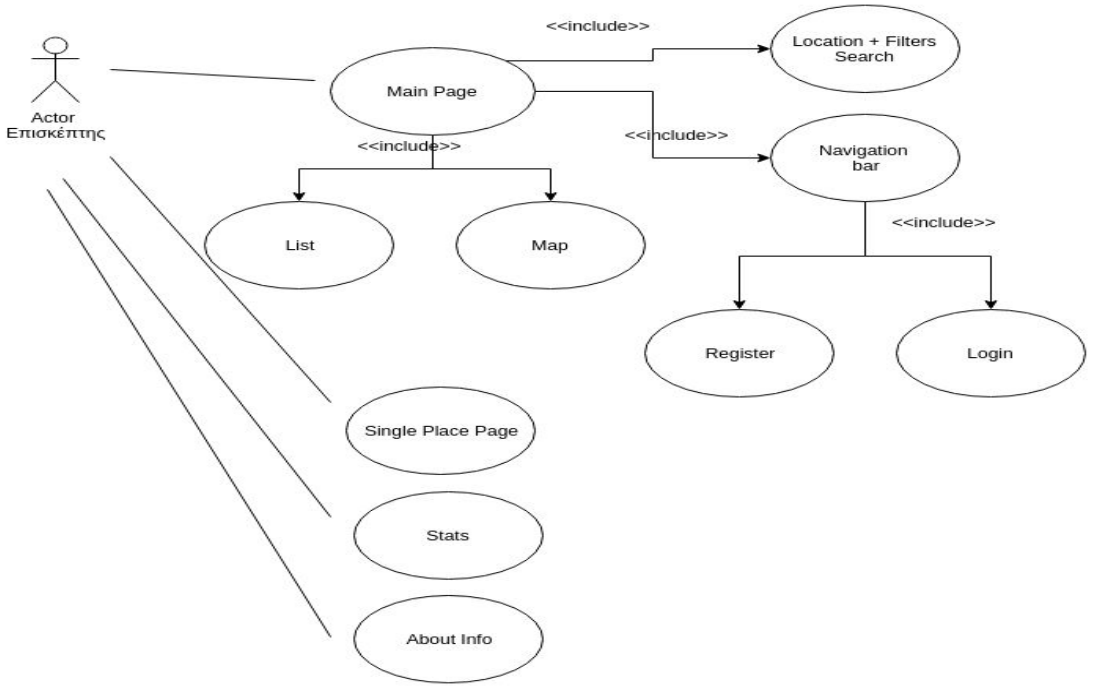
Το λογισμικό μας επικοινωνεί με το Leaflet Maps για τους χάρτες, έτσι ώστε να προβάλλονται τα επιλεγμένα μαγαζιά πάνω στο χάρτη. Επιπλέον, το λογισμικό, λόγω του RESTfull API επιτρέπει τη επικοινωνία με άλλα API, τα οποία μπορούν να ζητούν υπηρεσίες της εφαρμογής μας και να αλληλεπιδρούν.

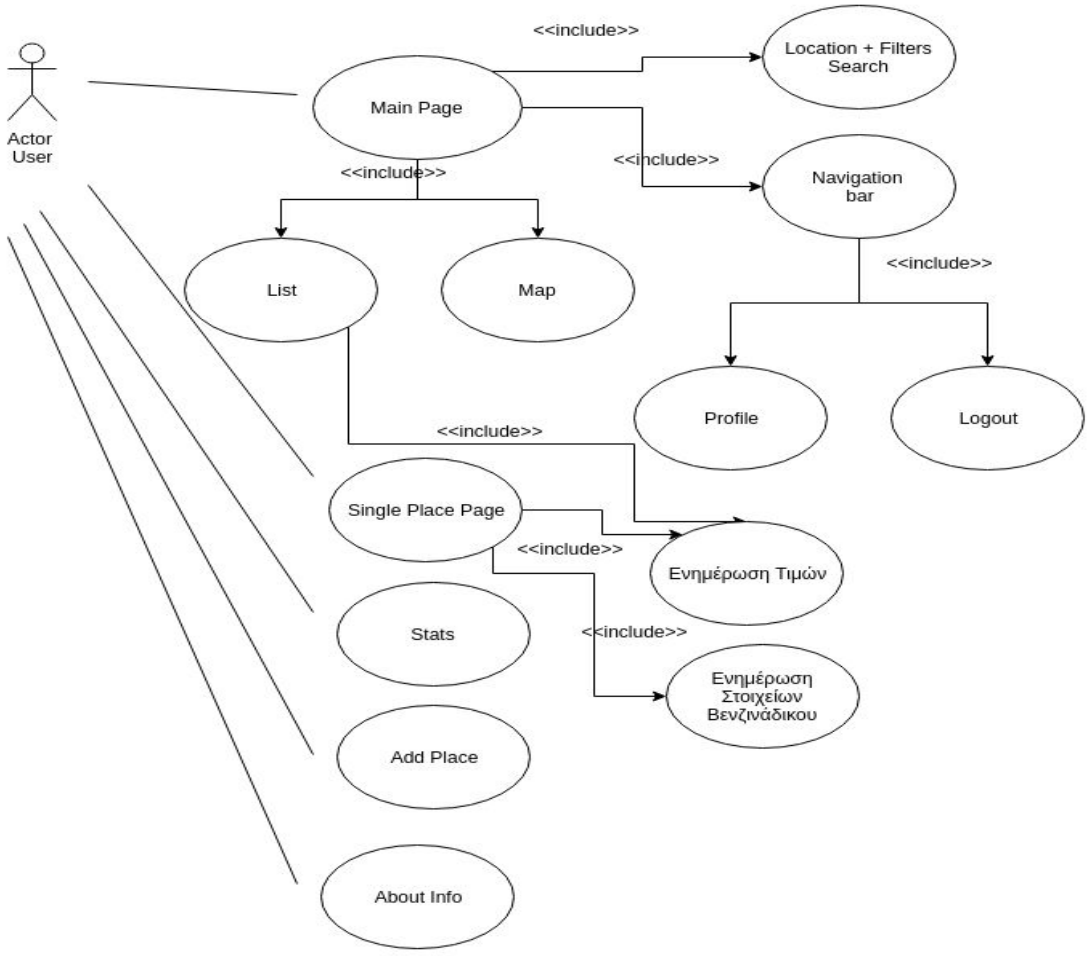
Leaflet Maps API: Αποσκοπώντας στην γεωγραφική απεικόνιση των καταστημάτων θα χρησιμοποιήσουμε το API Leaflet για την χρήση χαρτών. Το API θα χρησιμοποιείται όταν ο χρήστης κάνει την επιλογή της απεικόνισης κάποιου καταστήματος ή όλων των καταστημάτων εντός κάποιας εμβέλειας από την τοποθεσία του. Πατώντας πάνω στην τοποθετημένη πινέζα θα εμφανίζονται πληροφορίες σχετικά με το κατάστημα, όπως όνομα, τιμή καυσίμων κτλ.



1.2.2 Διεπαφές με το χρήστη

Η αλληλεπίδραση με τους χρήστες γίνεται μέσω των γραφικών διεπαφών της εφαρμογής, οι οποίες περιλαμβάνουν όλες τις διαθέσιμες ενέργειες για τους χρήστες της εφαρμογής, τους εγγεγραμμένους χρήστες που αποτελούν τους εθελοντές πληθοπορισμού καθώς και τους διαχειριστές. Η εφαρμογή έχει αναπτυχθεί στην λογική πως τα 3 είδη οθονών διαφοροποιούνται ελάχιστα και επαυξάνονται σύμφωνα με τις επιπρόσθετες λειτουργικότητες από το ένα είδος στο άλλο, προκειμένου να διατηρείται η συνοχή και η λογική συνέχεια.

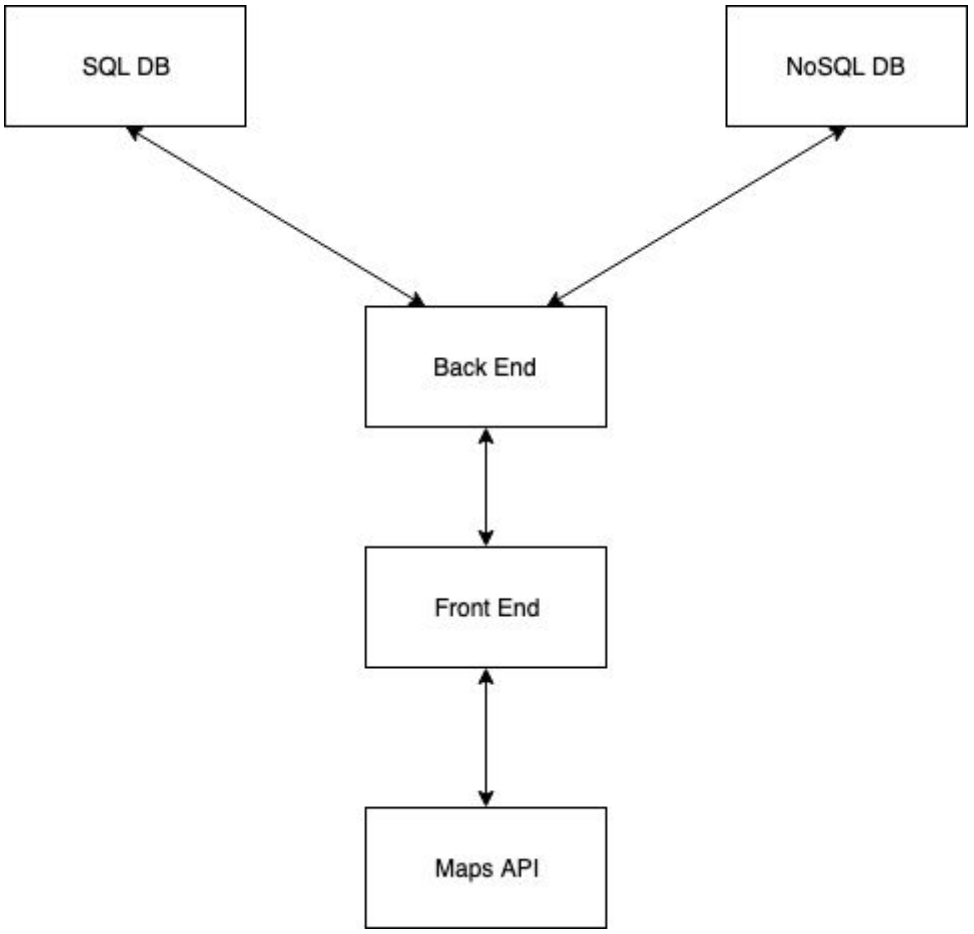




1.2.3 Διεπαφές με υλικό

Η εφαρμογή μας αλληλεπιδρά με το υλικό όταν ζητάμε την τοποθεσία του χρήστη καθώς η τοποθεσία αυτή λαμβάνεται από τον δέκτη GPS της συσκευής του χρήστη.

1.2.4 Διεπαφές επικοινωνιών



2. Προδιαγραφές απαιτήσεων λογισμικού

2.1 Εξωτερικές διεπαφές

Το API θα χρησιμοποιείται από το front end της εφαρμογής. Εφόσον στη βάση αποθηκεύονται το γεωγραφικό μήκος και πλάτος κάθε τοποθεσίας, το front end θα κάνει μια κλήση στο API το οποίο θα μεταφράζει τις συντεταγμένες σε τοποθεσίες και θα τις εμφανίζει πάνω στους χάρτες που θα παρέχει.

Επίσης, κατά την αποθήκευση κάποιου νέο καταστήματος, η εφαρμογή θα κάνει κλήση στο API ώστε να μεταφράσει τη διεύθυνση του νέου καταστήματος και να την μετατρέψει σε συντεταγμένες τις οποίες θα επιστρέψει. Αυτές στη συνέχεια αποθηκεύονται στη βάση της δικής μας εφαρμογής μέσω αντίστοιχης κλήσης POST στο δικό μας API.

2.2 Λειτουργίες: περιπτώσεις χρήσης

2.2.1 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 1: (Εγγραφή νέου χρήστη)

2.2.1.1 Χρήστες (ρόλοι) που εμπλέκονται

Μη εγγεγραμμένοι χρήστες

Front-end

Back-end

Mysql Database

2.2.1.2 Προϋποθέσεις εκτέλεσης

Για να εκτελεστεί η παραπάνω ενέργεια είναι απαραίτητο ο χρήστης να μην είναι εγγεγραμμένος στη βάση δεδομένων της εφαρμογής μας. Παράλληλα, απαιτείται τα στοιχεία που καταχωρεί ο χρήστης να είναι έγκυρα. Πιο συγκεκριμένα, απαιτούμε ο χρήστης να δώσει email address το οποίο είναι της μορφής email@provider.com και να δώσει κατάλληλο username και password. Τέλος, απαραίτητο για την εγγραφή του νέου χρήστη είναι τόσο το username όσο και η διεύθυνση ηλεκτρονικού ταχυδρομείου που υπέβαλε, να μην είναι καταχωρημένα στην βάση δεδομένων της εφαρμογής μας από άλλον χρήστη .

2.2.1.3 Περιβάλλον εκτέλεσης

Η εκτέλεση του συγκεκριμένου σεναρίου χρήσης ξεκινάει από την διαδικτυακή διεπαφή της εφαρμογής μας καθώς και τον front end server . Στην συνέχεια , εμπλέκεται ο back end server της εφαρμογής μας ο οποίος με την σειρά του επικοινωνεί με την βάση δεδομένων.

2.2.1.4 Δεδομένα εισόδου

Τα δεδομένα εισόδου είναι τα ακόλουθα (σε παρένθεση οι συνθήκες που πρέπει να ικανοποιούνται):

- Username (να είναι string, να μην είναι σε χρήση από άλλον χρήστη και να έχει μήκος 5-20 χαρακτήρες)
- password (να είναι string ,να μην είναι σε χρήση από άλλον χρήστη και να έχει μήκος τουλάχιστον 5 χαρακτήρες)
- email (να είναι string, να μην είναι σε χρήση από άλλον χρήστη ,να μην υπερβαίνει τους 255 χαρακτήρες και να είναι της μορφής myaddress@myprovider.com)

2.2.1.5 Παράμετροι

Προαιρετικά μπορεί να δοθεί η παράμετρος format. Η μόνη αποδεκτή τιμή αυτής της παραμέτρου είναι η τιμή json, η οποία είναι άλλωστε και η default τιμή της παραμέτρου. Οποιαδήποτε άλλη τιμή της παραμέτρου οδηγεί σε αποτυχία της εκτέλεσης της λειτουργίας.

2.2.1.6 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

Τα βήματα που ακολουθούνται είναι τα ακόλουθα:

Βήμα 1: Ο χρήστης επιλέγει το κουμπί "Εγγραφή"

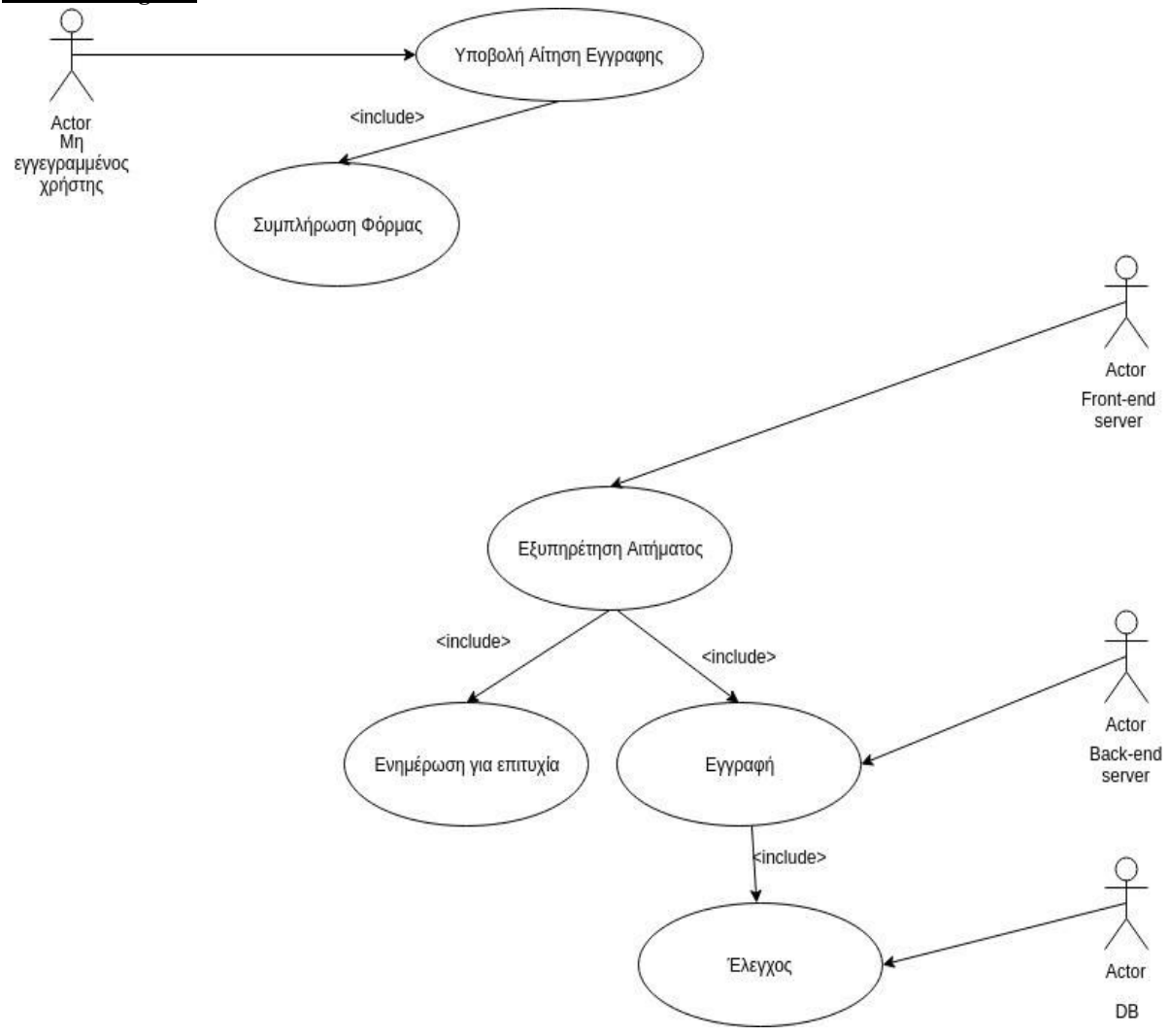
Βήμα 2: Ο χρήστης συμπληρώνει τη φόρμα εγγραφής

Βήμα 3: Ο Front-end server προωθεί τα δεδομένα στο Back-end και ελέγχει την εγκυρότητα τους. Επιστρέφει error σε περίπτωση λάθους.

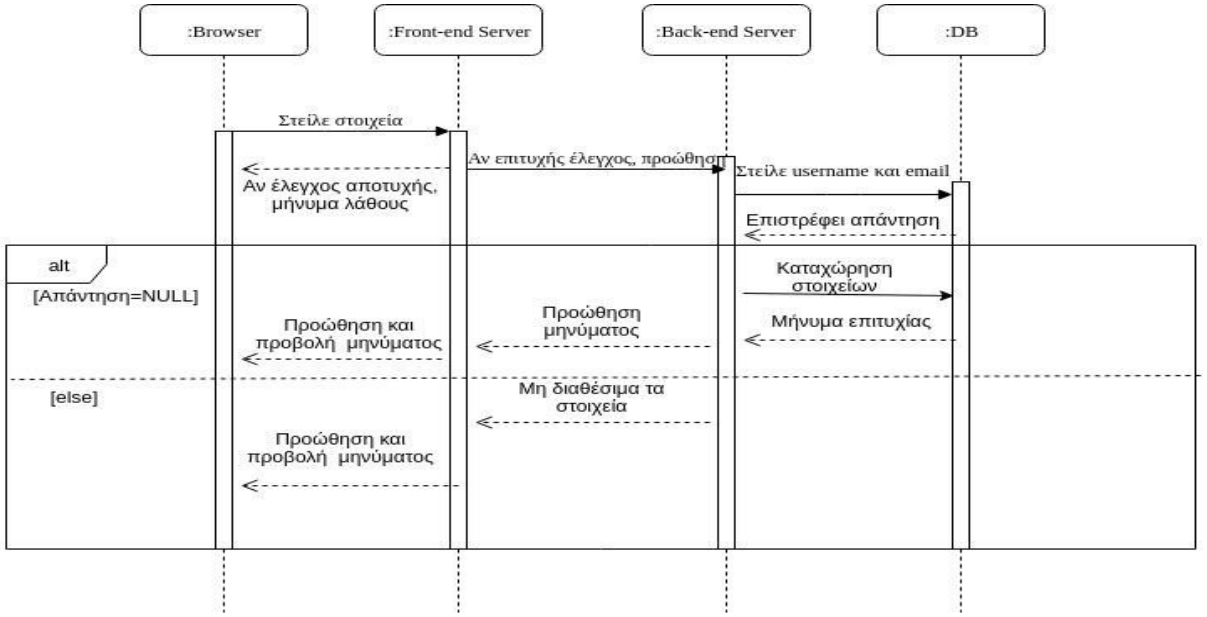
Βήμα 4: Ο Back-end server ελέγχει στη βάση δεδομένων αν το username και το email που δίνεται από το χρήστη είναι διαθέσιμα. Αν είναι γίνεται η εγγραφή του νέου χρήστη στη βάση δεδομένων της εφαρμογής μας.

Βήμα 5: Προωθείται κατάλληλο μήνυμα στο Front-end server και εμφανίζεται στην οθόνη του χρήστη.

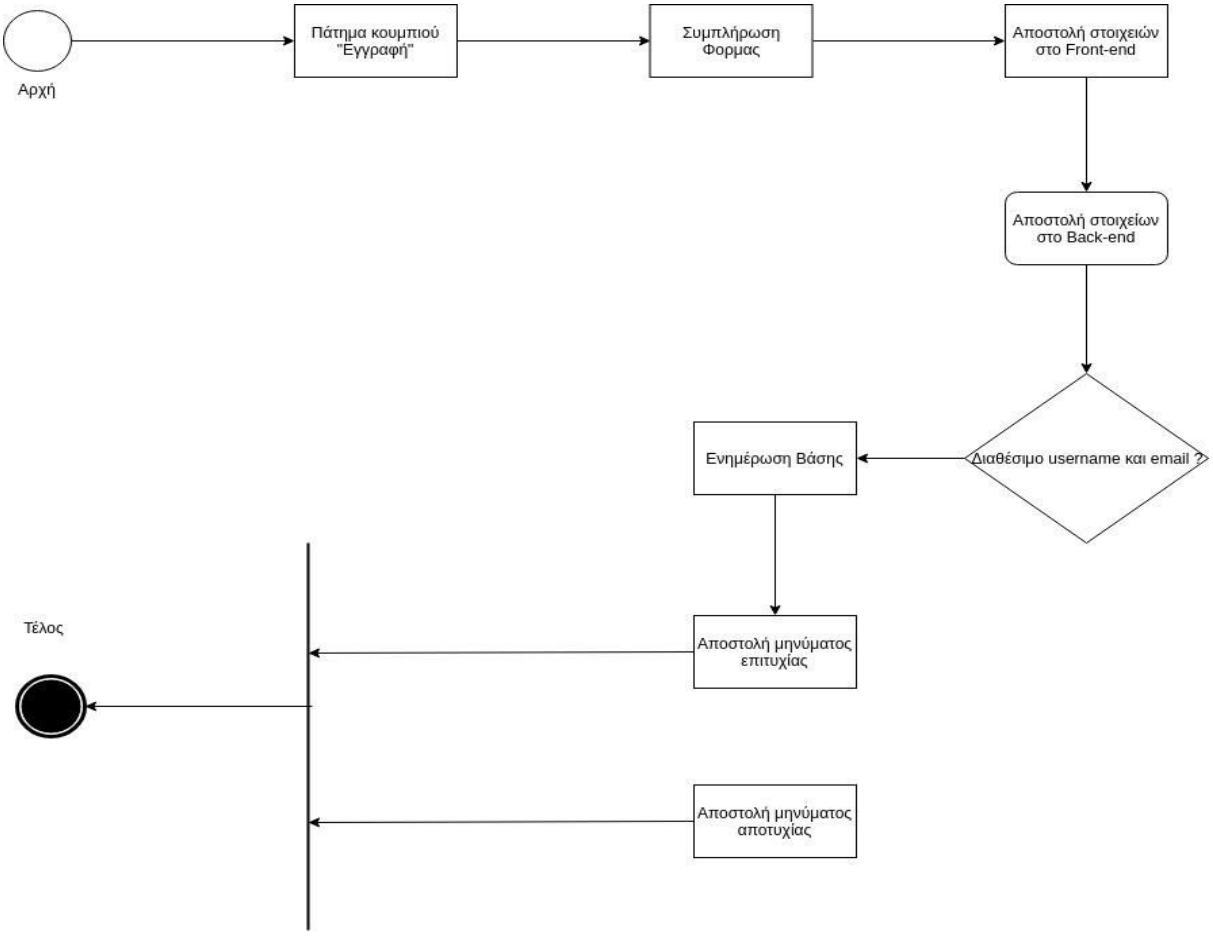
Use case Diagram



Sequence Diagram



Activity Diagram

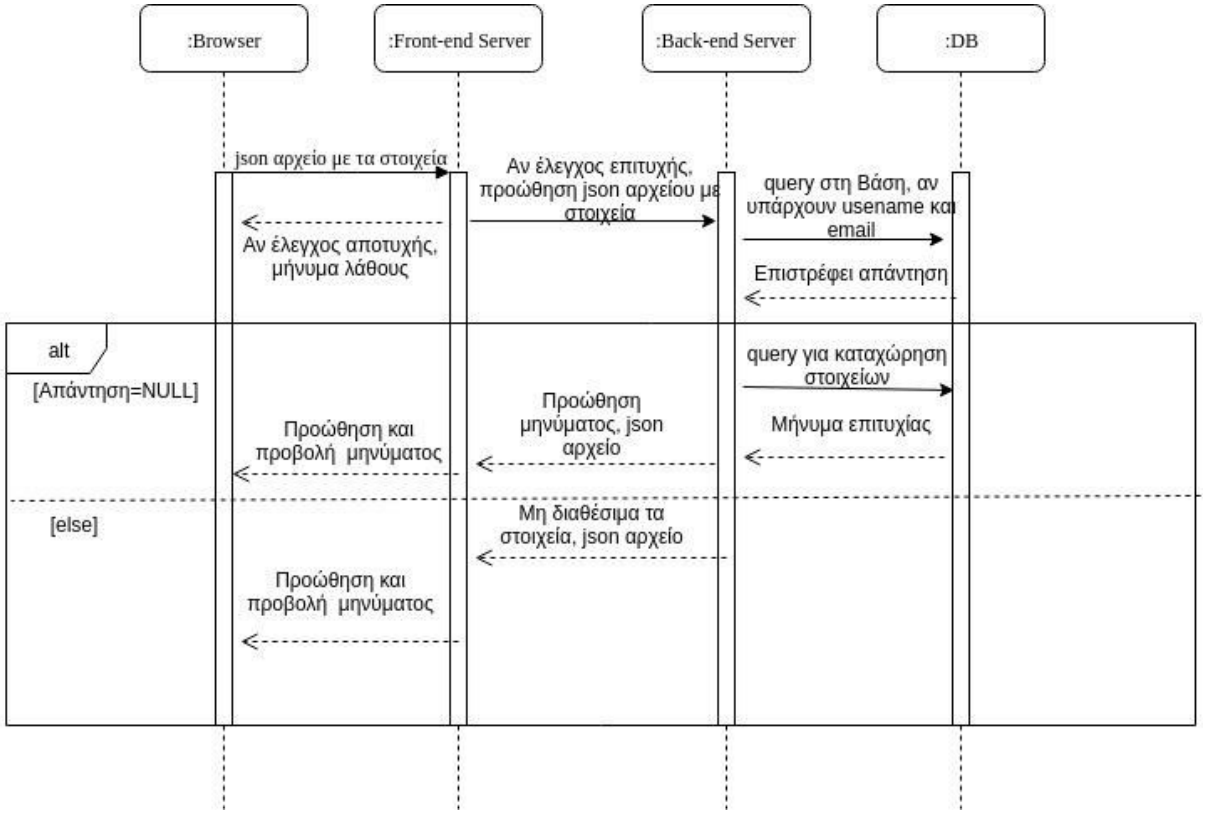


2.2.1.7 Δεδομένα εξόδου

Τα δεδομένα εξόδου στο use case αυτό είναι άρρηκτα συνδεδεμένα από τα αντίστοιχα δεδομένα εισόδου . Πιο συγκεκριμένα σε περίπτωση επιτυχημένης εγγραφής του νέου χρήστη ο back-end server επιστρέφει μήνυμα επιτυχίας μαζί με ένα authentication token για μελλοντική χρήση . Σε περίπτωση αποτυχίας διακρίνουμε δύο περιπτώσεις :

1) Η αποτυχία αυτή να οφείλεται σε εσφαλμένα δεδομένα εισόδου . Στην περίπτωση αυτή , ο back-end server επιστρέφει μήνυμα λάθους ανάλογα με το σφάλμα στα δεδομένα εισόδου (πχ Please complete all the mandatory fields! , σε περίπτωση που δεν δοθεί κάποιο υποχρεωτικό πεδίο).
2) Η αποτυχία αυτή να οφείλεται στο ότι είτε το username είτε η διεύθυνση ηλεκτρονικού ταχυδρομείο που πληκτρολόγησε ο χρήστης είναι ήδη σε χρήση από άλλον χρήστη . Στην περίπτωση αυτή , ο back-end server επιστρέφει κατάλληλο μήνυμα λάθους (πχ Username is already in use! , σε περίπτωση που το όνομα χρήστη δεν είναι διαθέσιμο).

Τέλος , ο front-end server επιστρέφει ανάλογο μήνυμα αποτυχίας ή επιτυχίας . Το μήνυμα αυτό εξαρτάται άμεσα από την απάντηση του back-end server . Σε περίπτωση επιτυχίας έχουμε νέα εγγραφή στη βάση δεδομένων σχετικά με τον νέο εγγεγραμμένο χρήστη στον πίνακα users.



2.2.2 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 2: (Καταχώρηση τιμής καυσίμου)

2.2.2.1 Χρήστες (ρόλοι) που εμπλέκονται

Χρήστης

Front-end

Back-end

Mysql Main Database

Blacklist Database (Redis)

2.2.2.2 Προϋποθέσεις εκτέλεσης

Για την εκτέλεση του συγκεκριμένου σεναρίου (καταχώρηση τιμής καυσίμου) απαιτείται ο χρήστης να είναι διαπιστευμένος . Το γεγονός αυτό συνεπάγεται ότι ο χρήστης πρέπει να είναι τόσο εγγεγραμμένος χρήστης της εφαρμογής μας όσο και να έχει κάνει login στην εφαρμογής μας έτσι ώστε να διαθέτει το κατάλληλο authentication token . Παράλληλα , θα πρέπει το token αυτό να είναι valid καθώς μετά από αποσύνδεση ενός χρήστη από την εφαρμογή μας το πρότερο token γίνεται invalid. Τέλος , απαιτείται τα δεδομένα εισόδου να είναι της σωστής μορφής . Αυτό σημαίνει ότι απαιτούμε η μεταβλητή DateFrom να μην είναι αργότερα της DateTo, καθώς και τα κλειδιά ShopID και ProductID να αποτελούν κλειδιά εγγραφών στην βάση δεδομένων της εφαρμογής μας.

2.2.2.3 Περιβάλλον εκτέλεσης

Η εκτέλεση του συγκεκριμένου σεναρίου χρήσης ξεκινάει από την διαδικτυακή διεπαφή της εφαρμογής μας καθώς και τον front end server . Στην συνέχεια , εμπλέκεται ο back end server της εφαρμογής μας ο οποίος με την σειρά του επικοινωνεί με την βάση δεδομένων.

2.2.2.4 Δεδομένα εισόδου

Τα δεδομένα εισόδου είναι τα ακόλουθα (σε παρένθεση οι συνθήκες που πρέπει να ικανοποιούνται):

Τιμή (μη αρνητικός αριθμός)

Ημερομηνία (format YYYY/MM/DD)

ShopID (να είναι φυσικός αριθμός και να υπάρχει κατάστημα στη βάση δεδομένων με κλειδί το συγκεκριμένο identifier)

ProductID (να είναι φυσικός αριθμός και να υπάρχει προιον στη βάση δεδομένων με κλειδί το συγκεκριμένο identifier)

Valid authentication token στην επικεφαλίδα X-OBSERVATORY-AUTH

Επιπλέον, να ισχύουν οι συνθήκες της 2.2.2.2

2.2.2.5 Παράμετροι

Προαιρετικά μπορεί να δοθεί η παράμετρος format. Η μόνη αποδεκτή τιμή αυτής της παραμέτρου είναι η τιμή json, η οποία είναι άλλωστε και η default τιμή της παραμέτρου. Οποιαδήποτε άλλη τιμή της παραμέτρου οδηγεί σε αποτυχία της εκτέλεσης της λειτουργίας.

2.2.2.6 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

Τα βήματα που ακολουθούνται είναι τα ακόλουθα:

Βήμα 1: Ο χρήστης επιλέγει κατάστημα και επεξεργασία

Βήμα 2: Ο χρήστης συμπληρώνει τα πεδία που απαιτούνται

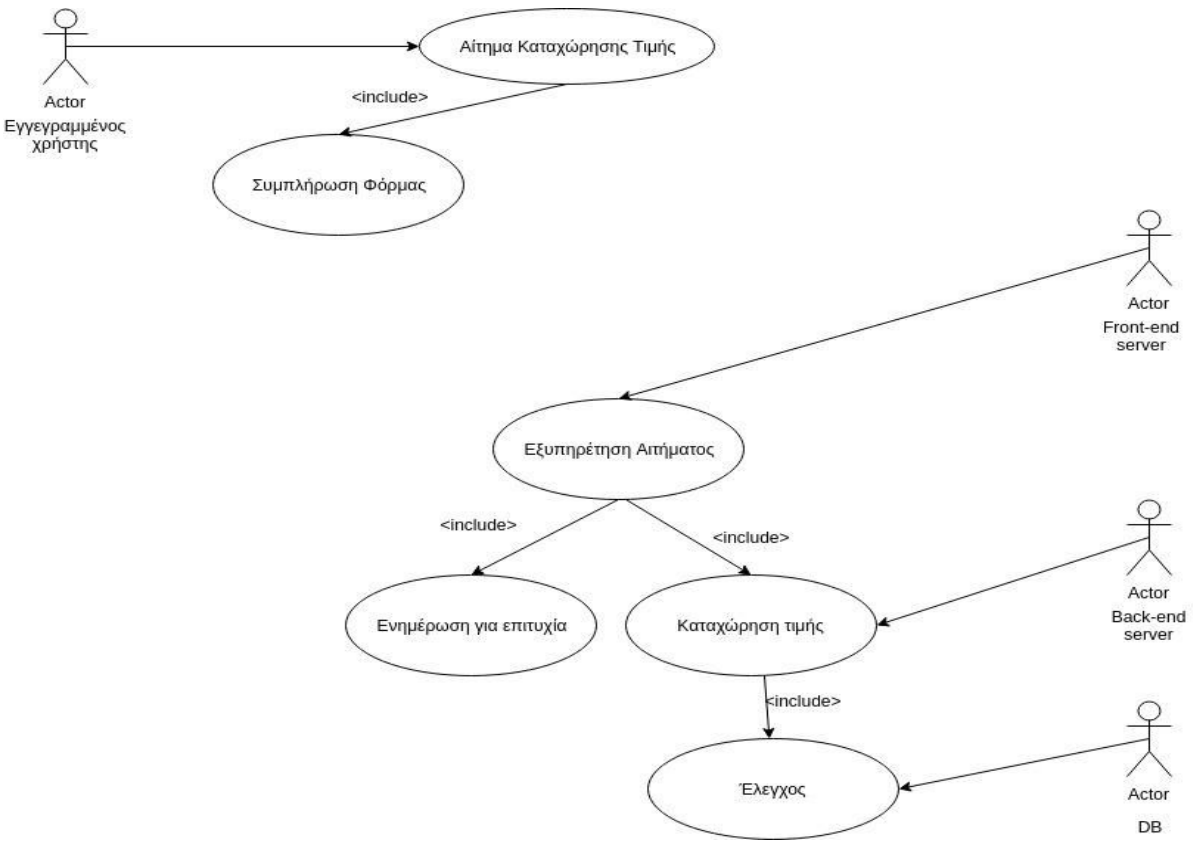
Βήμα 3: Το αίτημα προωθείται στο Front-end και από κει στο Back-end

Βήμα 4: Έλεγχος για το εάν ο χρήστης είναι διαπιστευμένος.

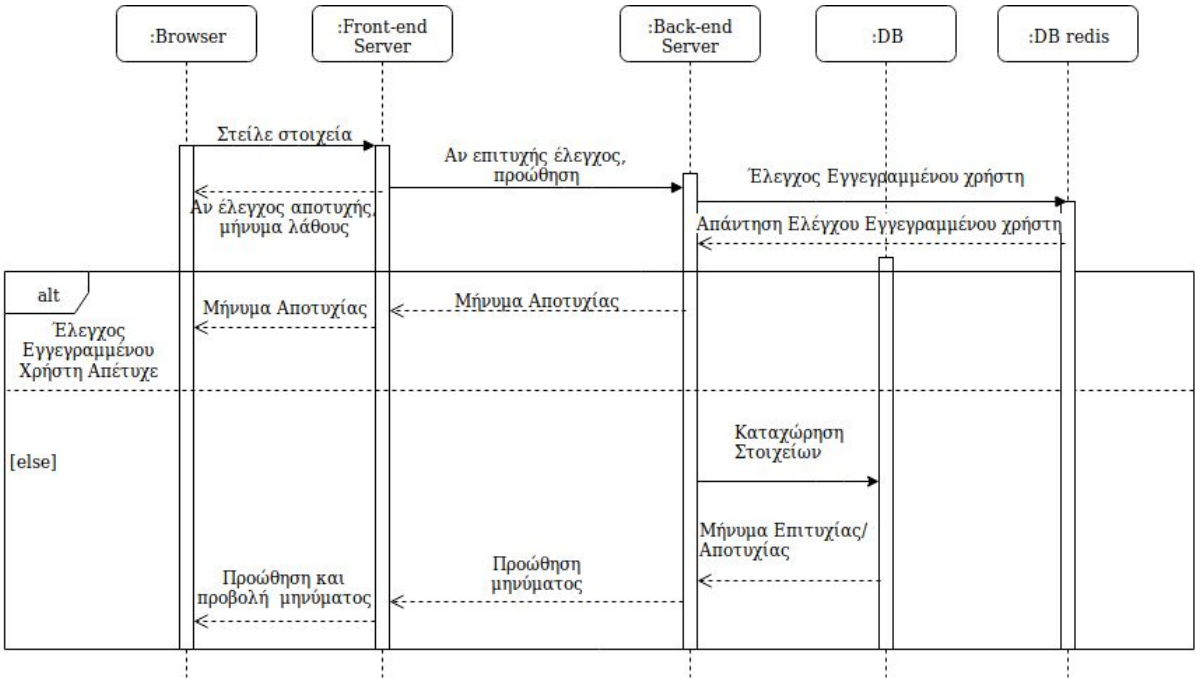
Βήμα 5: Αν υπάρχει το κατάστημα και το το προϊόν στη Βάση τότε γίνεται η νέα καταχώρηση στη βάση.

Βήμα 6: Σε κάθε περίπτωση επιστρέφεται κατάλληλο μήνυμα το οποίο προωθείται τελικά στον Browser και προβάλλεται.

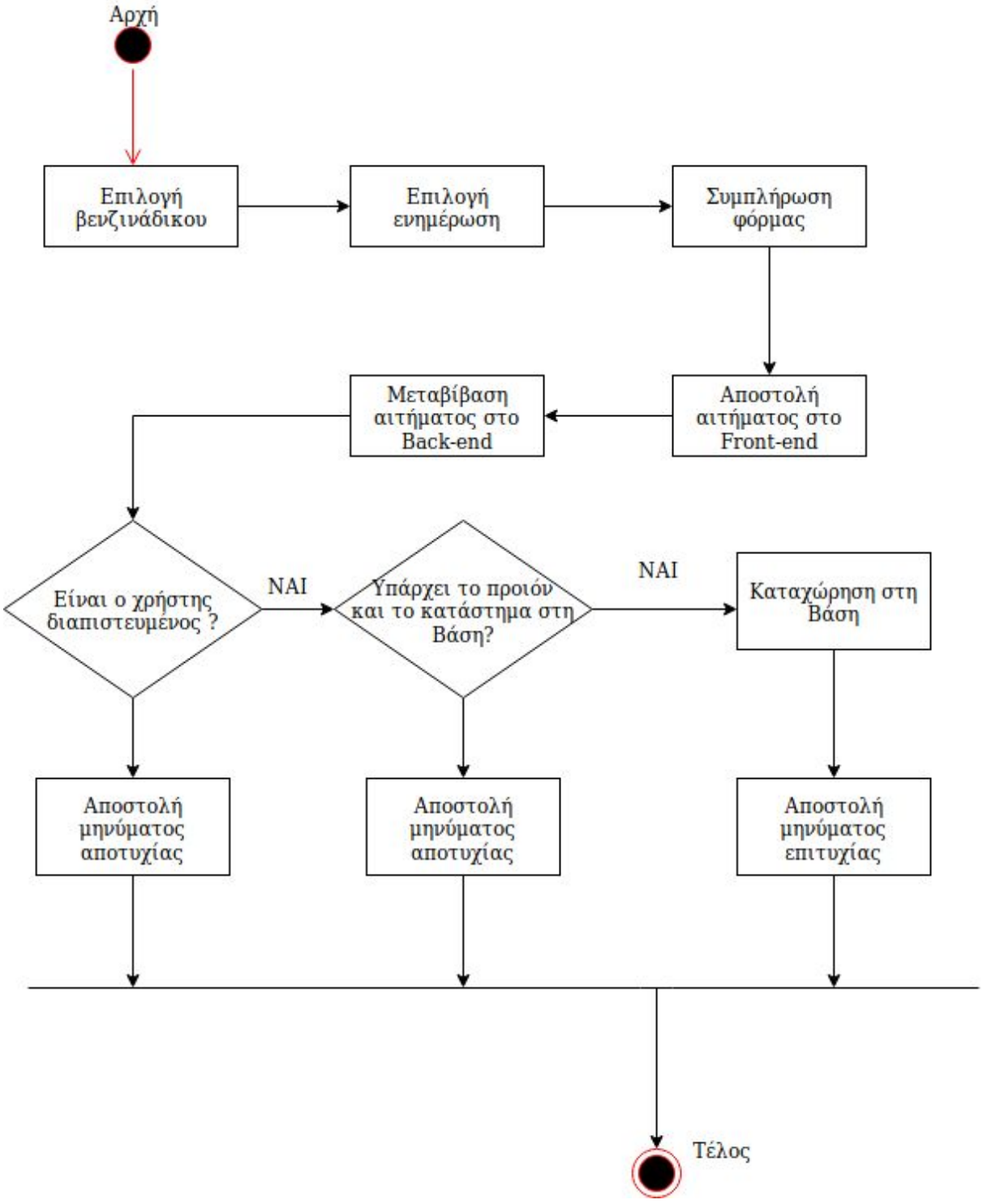
Use case Diagram



Sequence Diagram



Activity Diagram



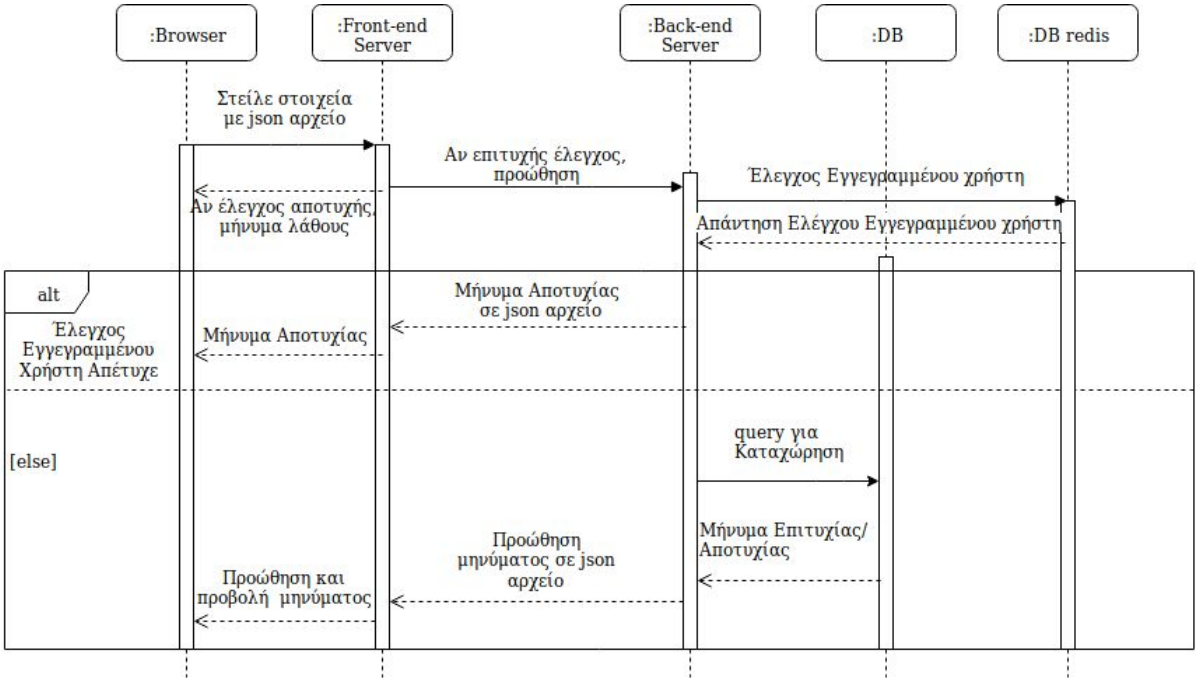
2.2.2.7Δεδομένα εξόδου

Ως δεδομένα εξόδου, σε περίπτωση αποτυχίας επιστρέφεται json αρχείο με σχετικό μήνυμα λάθους (πχ “Please provide a valid shopId !” στην περίπτωση που δεν βρεθεί το σχετικό κατάστημα).

Μετά την επιτυχή καταχώρηση τιμής καυσίμου (= προϊόντος) επιστρέφονται τα εξής στοιχεία μέσα σε ένα json αρχείο:

1. postId -> Το id του post που μόλις προστέθηκε.
2. price -> Η τιμή που εισήγαγε ο χρήστης.
3. dateFrom -> Η ημερομηνία από την οποία ισχύει η καταχώρηση.
4. dateTo -> Η ημερομηνία και μέχρι την οποία ισχύει η καταχώρηση.
5. productName -> Το όνομα του συσχετιζόμενου προϊόντος.
6. productId -> Ο κωδικός του συσχετιζόμενου προϊόντος.
7. product Tags -> Τα tags του συσχετιζόμενου προϊόντος.
8. shopId -> Ο κωδικός του συσχετιζόμενου καταστήματος.
9. shopName -> Το όνομα του συσχετιζόμενου καταστήματος.
10. shopTags -> Τα tags του συσχετιζόμενου καταστήματος.
11. shopAddress -> Η διεύθυνση του συσχετιζόμενου καταστήματος.

Σε περίπτωση επιτυχίας έχουμε νέα εγγραφή στη βάση δεδομένων σχετικά με την νέα καταχώρηση τιμής στον πίνακα posts.



2.2.3 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 3: (Αναζήτηση τιμών)

2.2.3.1 Χρήστες (ρόλοι) που εμπλέκονται

Εγγεγραμμένος ή μη χρήστης

Front-end

Back-end

Mysql Main Database

API Google Map

2.2.3.2 Προϋποθέσεις εκτέλεσης

Δεν υπάρχει κάποια ιδιαίτερη προϋπόθεση για την εκτέλεση του σεναρίου αυτού, πέρα από το να είναι διαθέσιμοι τόσο οι servers της εφαρμογής μας όσο και το API της Google. Επιπλέον, στην περίπτωση που παρέχεται κάποια από τις προαιρετικές παραμέτρους, αυτή θα πρέπει να έχει την σωστή μορφή όπως περιγράφεται παρακάτω.

2.2.3.3 Περιβάλλον εκτέλεσης

Η εκτέλεση γίνεται στη διαδικτυακή διεπαφή, στο Back-end, στη βάση δεδομένων και στο API της Google.

2.2.3.4 Δεδομένα εισόδου

Δεν έχουμε δεδομένα εισόδου σε αυτήν την περίπτωση, καθώς λαμβάνουμε όλες τις απαιτούμενες πληροφορίες από το url και τις τιμές του query.

2.2.3.5 Παράμετροι

Προαιρετικά μπορούν να δοθούν οι παρακάτω παράμετροι ως query στο url :

1. format -> Default τιμή json. Η μόνη αποδεκτή τιμή αυτής της παραμέτρου είναι η τιμή json, η οποία είναι άλλωστε και η default τιμή της παραμέτρου. Οποιαδήποτε άλλη τιμή της παραμέτρου οδηγεί σε αποτυχία της εκτέλεσης της λειτουργίας.
2. start -> Default τιμή 0, δείχνει από ποιο στοιχείο θα ξεκινήσει η εμφάνιση των αποτελεσμάτων. Πρέπει να είναι ακέραιος μη αρνητικός αριθμός.
3. count -> Default τιμή 20, δείχνει τον μέγιστο αριθμό στοιχείων που θα επιστρέψουμε στον χρήστη. Πρέπει να είναι ακέραιος θετικός αριθμός.
4. geoDist, geoLng, geoLat -> Παράμετροι που είτε θα προσδιορίζονται και οι τρεις είτε καμία. Η πρώτη είναι θετικός ακέραιος αριθμός και δείχνει την μέγιστη απόσταση που θα πρέπει να απέχουν τα καταστήματα του αποτελέσματος από το σημείο ενδιαφέροντος που προσδιορίζεται από τις άλλες δύο παραμέτρους. Οι παράμετροι geoLng, geoLat έχουν μορφή δεκαδικού αριθμού με το πολύ 6 δεκαδικά ψηφία και μπορεί να είναι είτε θετικοί είτε αρνητικοί αριθμοί. Η πρώτη μπορεί να παίρνει τιμές στο εύρος [-180, 180] ενώ η δεύτερη στο εύρος [-90, 90].
5. dateFrom, dateTo -> Παράμετροι που είτε θα προσδιορίζονται και οι δύο είτε καμία. Είναι ημερομηνίες σε μορφή YYYY-MM-DD και πρέπει η ημερομηνία dateTo να μην είναι νωρίτερα από αυτήν της dateFrom. Οι παράμετροι αυτοί αφορούν το χρονικό εύρος το οποίο θα αφορά η αναζήτησή μας. Σε περίπτωση που δεν προσδιορίζονται θεωρούμε ότι παίρνουν και οι δύο την ημερομηνία της στιγμής του χρόνου εκτέλεσης.

6. shops, products, tags -> Προαιρετικές παράμετροι σε μορφή λιστών. Η πρώτη σε περίπτωση που υπάρχει περιορίζει την αναζήτηση μόνο σε καταστήματα των οποίων το shopId είναι μέσα στη λίστα. Η δεύτερη σε περίπτωση που υπάρχει περιορίζει την αναζήτηση μόνο σε προιόντα των οποίων το productId είναι μέσα στη λίστα. Η τρίτη σε περίπτωση που υπάρχει περιορίζει την αναζήτηση μόνο σε προιόντα ή καταστήματα που έχουν τουλάχιστον ένα tag που να περιέχεται στην λίστα.
7. sort -> Λίστα που προσδιορίζει τον τρόπο ταξινόμησης των αποτελεσμάτων. Η ταξινόμηση μπορεί να γίνει είτε βάσει τιμής, είτε βάσει απόστασης (σε περίπτωση που παρέχεται σημείο ενδιαφέροντος), είτε βάσει ημερομηνίας. Υποστηρίζεται ταξινόμηση πολλών επιπέδων. Οι τιμές των πρώτων δύο λιστών πρέπει να είναι θετικοί ακέραιοι αριθμοί. Οι τιμές της τρίτης λίστας αρκεί να είναι τύπου String.

2.2.3.6 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

Τα βήματα που ακολουθούνται είναι τα ακόλουθα:

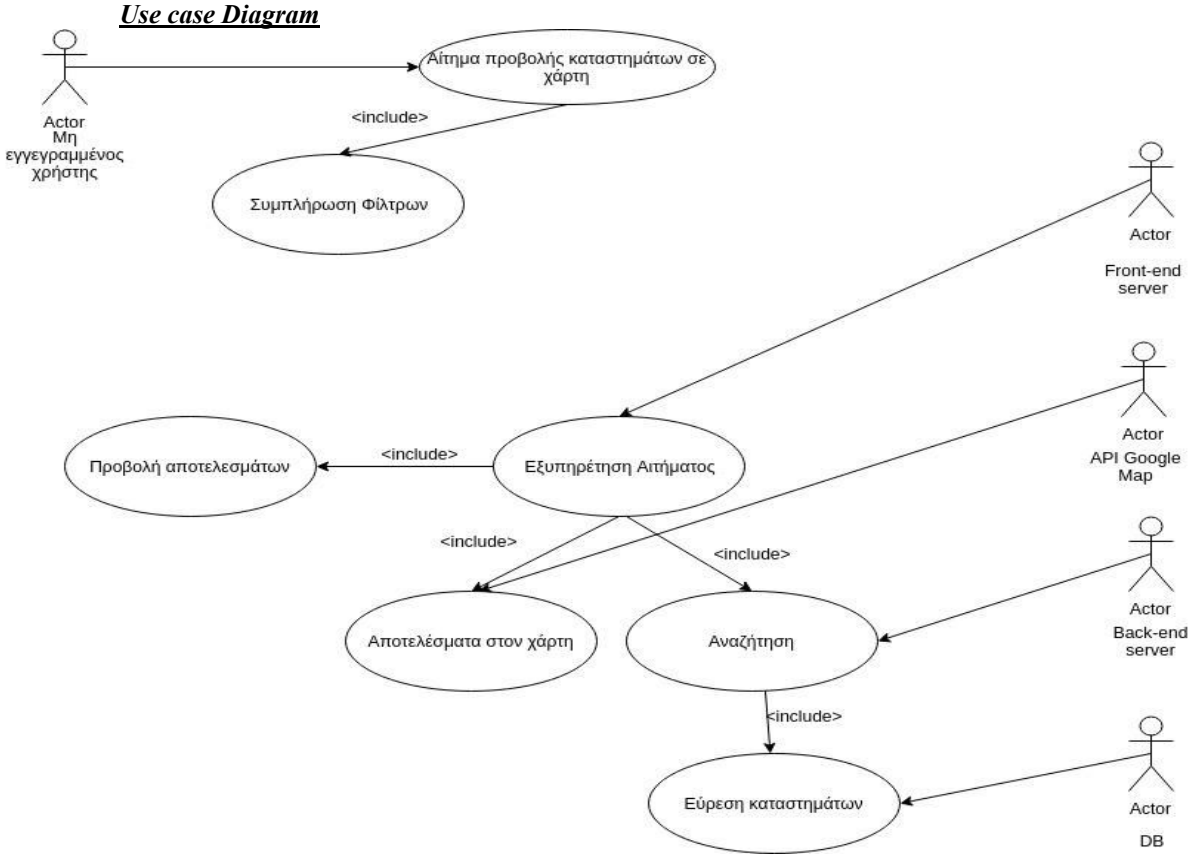
Βήμα 1: Συμπλήρωση φόρμας αναζήτησης.

Βήμα 2: Προωθείται το αίτημα στο Front-end και από κει στο Back-end.

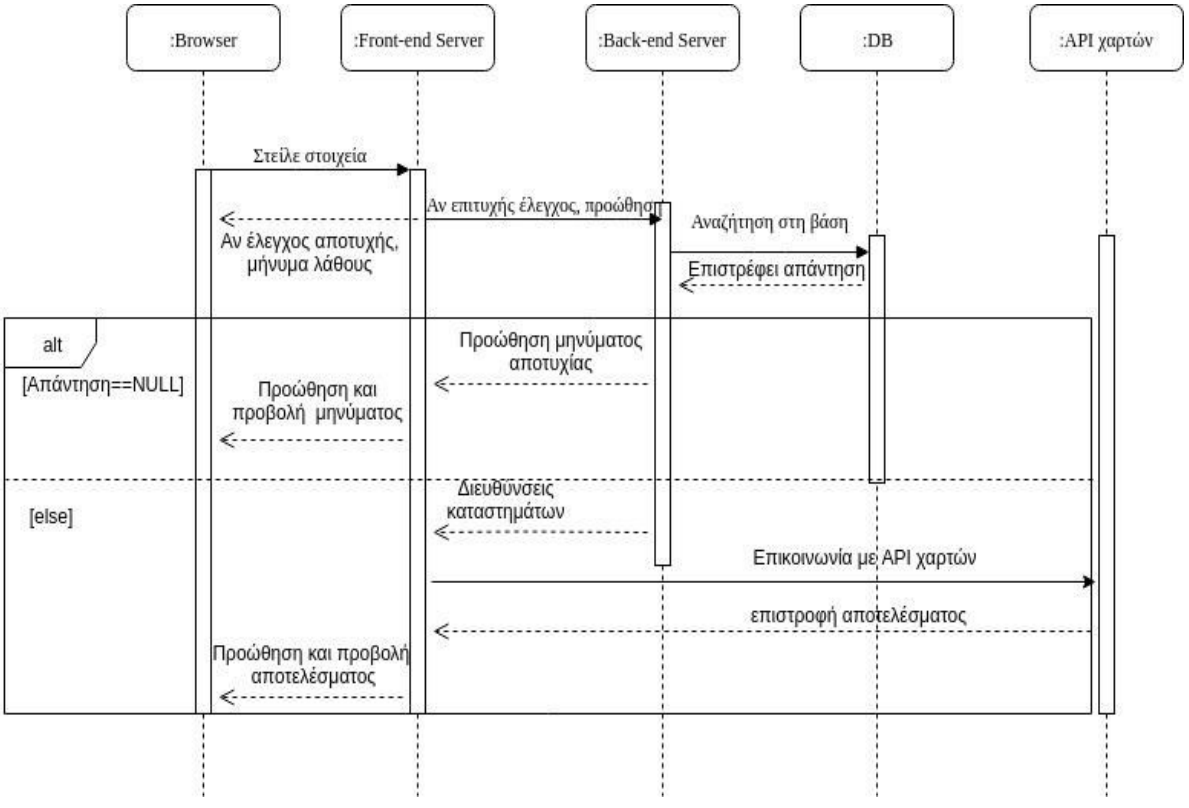
Βήμα 3: Γίνεται έλεγχος στη Βάση αν υπάρχουν κατάλληλες καταχωρήσεις post.

Βήμα 4: Αν ο έλεγχος είναι επιτυχής, γίνεται προώθηση των αποτελεσμάτων στο front-end και στη συνέχεια επικοινωνία με το API της Google και επιστρέφεται το επιθυμητό αποτέλεσμα ή μήνυμα σφάλματος στο χρήστη.

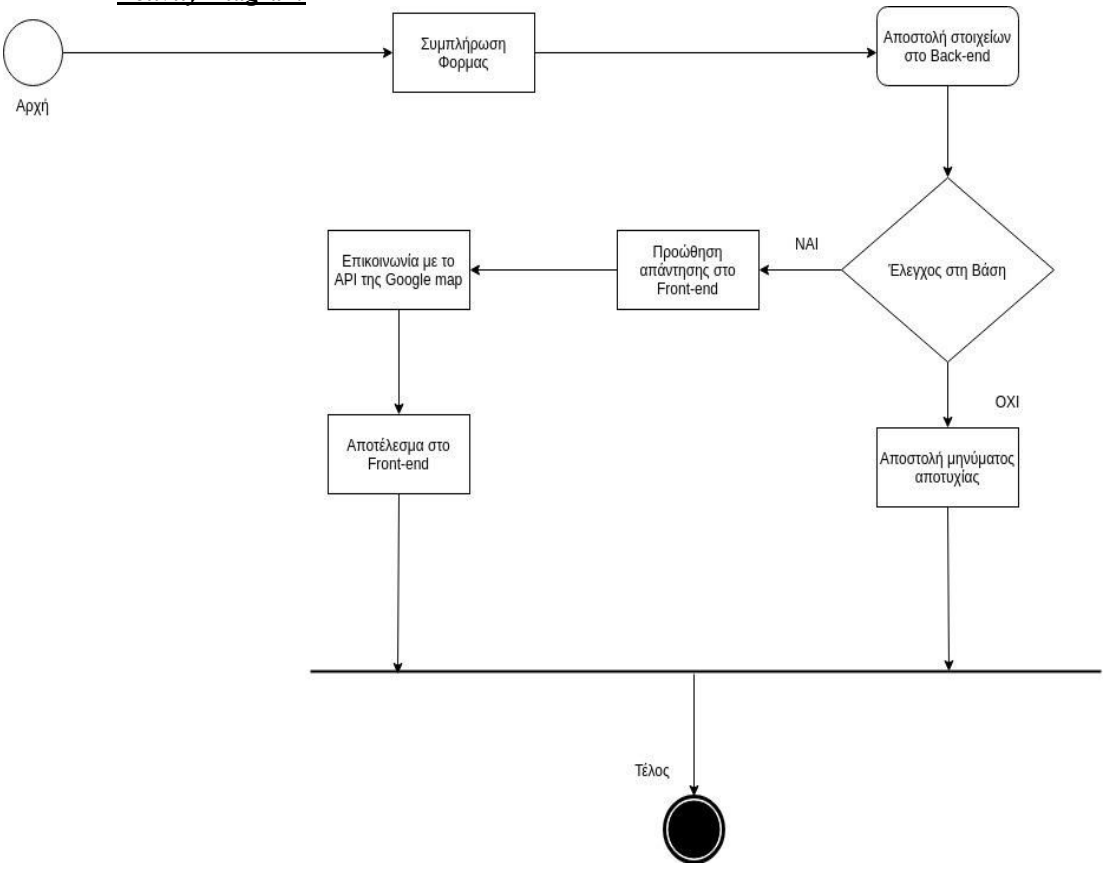
Use case Diagram



Sequence Diagram



Activity Diagram



2.2.4 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 4: (Προσθήκη καταστήματος μέσω εξωτερικού χρήστη του API)

2.2.4.1 Χρήστες (ρόλοι) που εμπλέκονται

Εξωτερικός χρήστης που χρησιμοποιεί το RESTful API που υλοποιήθηκε

Back-end

Mysql Main Database

Redis Database

2.2.4.2 Προϋποθέσεις εκτέλεσης

Σε αυτό το σενάριο εκτέλεσης υποθέτουμε πως συμμετέχει ένας εξωτερικός χρήστης και όχι το front-end, αναδεικνύοντας με αυτόν τον τρόπο την δυνατότητα ενός RESTful API να αξιοποιείται από πληθώρα χρηστών και όχι μόνο από το front-end της εφαρμογής. Για την εκτέλεση του συγκεκριμένου σεναρίου (προσθήκη καταστήματος) απαιτείται ο χρήστης να είναι διαπιστευμένος. Το γεγονός αυτό συνεπάγεται ότι ο χρήστης πρέπει να είναι τόσο εγγεγραμμένος χρήστης της εφαρμογής μας όσο και να έχει κάνει login στην εφαρμογής μας έτσι ώστε να διαθέτει το κατάλληλο authentication token. Παράλληλα, θα πρέπει το token αυτό να είναι valid καθώς μετά από αποσύνδεση ενός χρήστη από την εφαρμογή μας το πρότερο token γίνεται invalid. Τέλος , απαιτείται τα δεδομένα εισόδου να είναι της σωστής μορφής. Αυτό σημαίνει ότι απαιτούμε οι μεταβλητές lng και lat να είναι δεκαδικοί αριθμοί κατάλληλης μορφής (δηλαδή να είναι γεωγραφικές συντεταγμένες), η μεταβλητή withdrawn να είναι τύπου boolean, η μεταβλητή tags να είναι λίστα από string και

οι μεταβλητές name και address να είναι string. Προφανώς, επιπλέον, οι server και οι βάσεις Mysql και redis πρέπει να είναι σε λειτουργία.

2.2.4.3 Περιβάλλον εκτέλεσης

Η εκτέλεση πραγματοποιείται στο Back-end και στις βάσεις.

2.2.4.4 Δεδομένα εισόδου

Τα δεδομένα εισόδου είναι τα ακόλουθα:

1. Γεωγραφικές συντεταγμένες lat και lng του καταστήματος.
2. Διεύθυνση καταστήματος address που περιλαμβάνει οδό, αριθμό και ταχυδρομικό κώδικα.
3. Όνομα καταστήματος name.
4. Boolean flag withdrawn που δείχνει αν είναι διαθέσιμο το κατάστημα.

2.2.4.5 Παράμετροι

Προαιρετικά μπορεί να δοθεί η παράμετρος format. Η μόνη αποδεκτή τιμή αυτής της παραμέτρου είναι η τιμή json, η οποία είναι άλλωστε και η default τιμή της παραμέτρου. Οποιαδήποτε άλλη τιμή της παραμέτρου οδηγεί σε αποτυχία της εκτέλεσης της λειτουργίας.

2.2.4.6 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

Τα βήματα που ακολουθούνται είναι τα ακόλουθα:

Βήμα 1: Αίτημα προσθήκης καταστήματος από κάποιο API

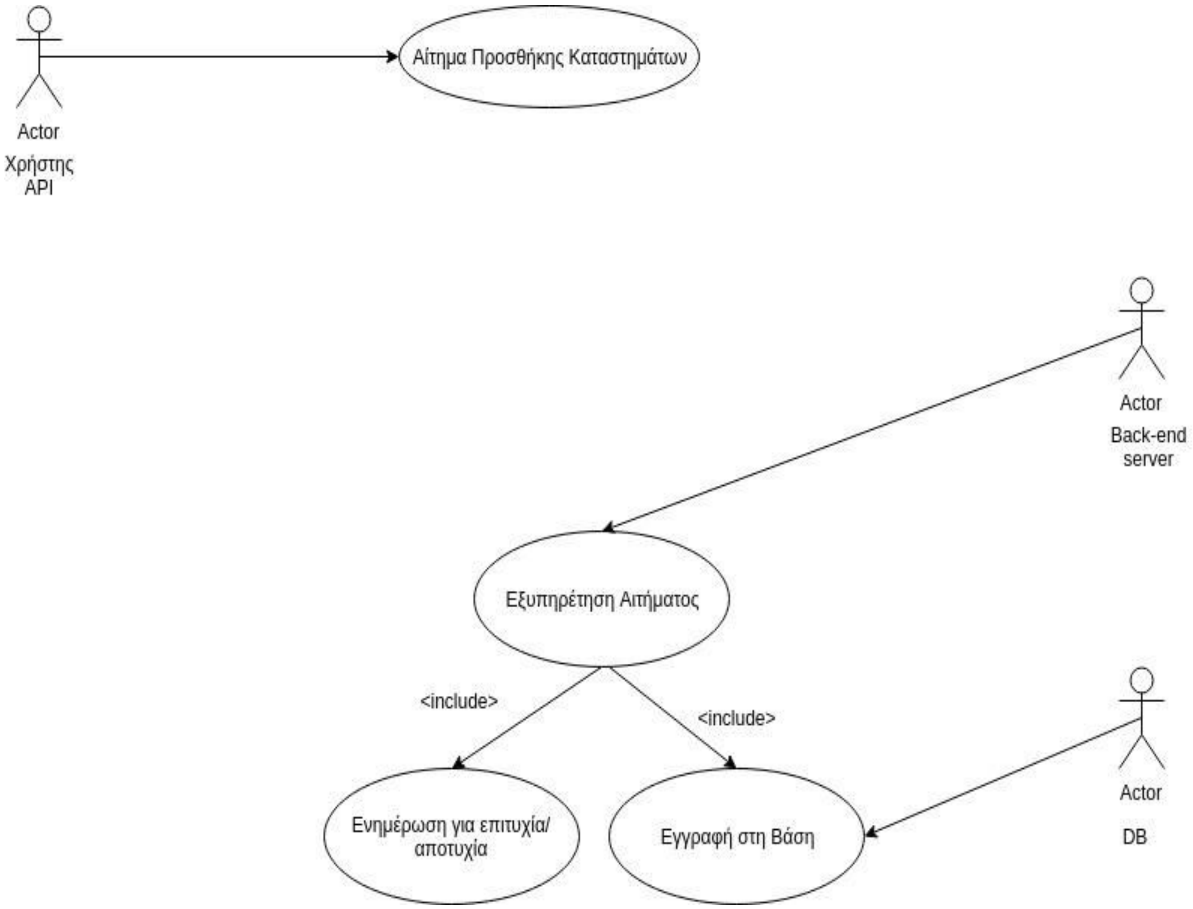
Βήμα 2: Έλεγχος format αιτήματος από το Back-end

Βήμα 3: Έλεγχος αν ο χρήστης είναι logged in μέσω της βάσης redis.

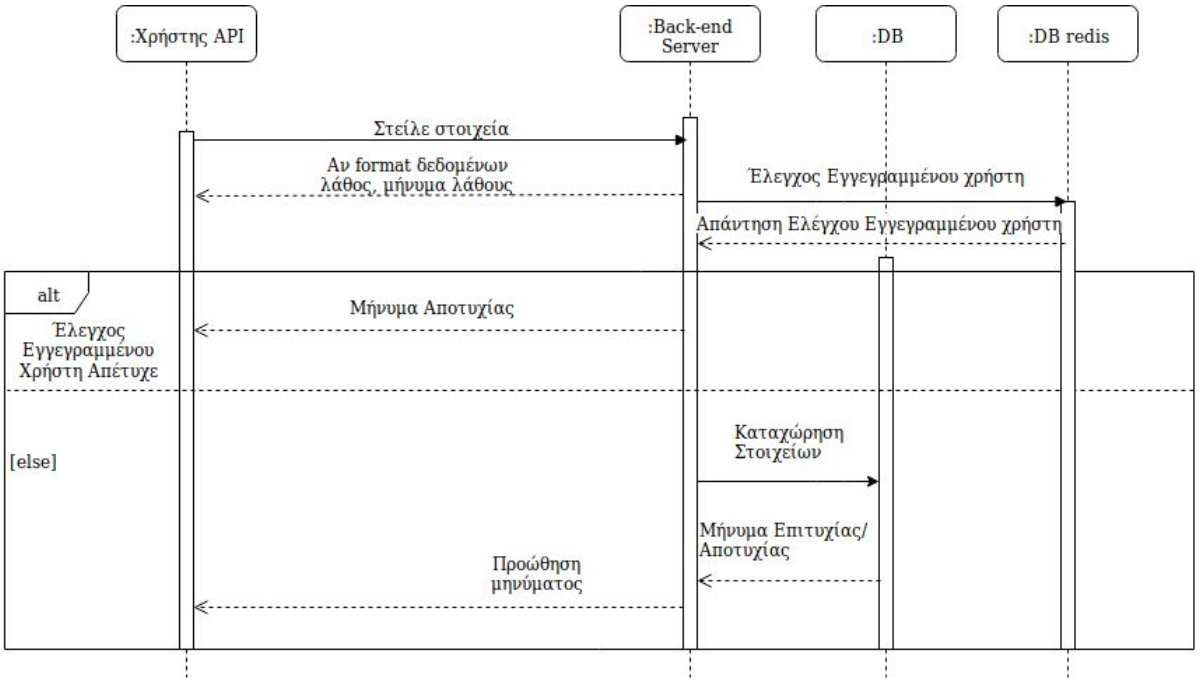
Βήμα 4: Αν οι έλεγχοι είναι επιτυχημένοι, προωθούνται τα στοιχεία στη Βάση, αλλιώς επιστρέφεται κατάλληλο μήνυμα.

Βήμα 5: Η Βάση κάνει τους απαραίτητους ελέγχους και αν αυτοί είναι επιτυχείς γίνεται η προσθήκη καταστήματος στη Βάση. Σε κάθε περίπτωση επιστρέφεται κατάλληλο μήνυμα.

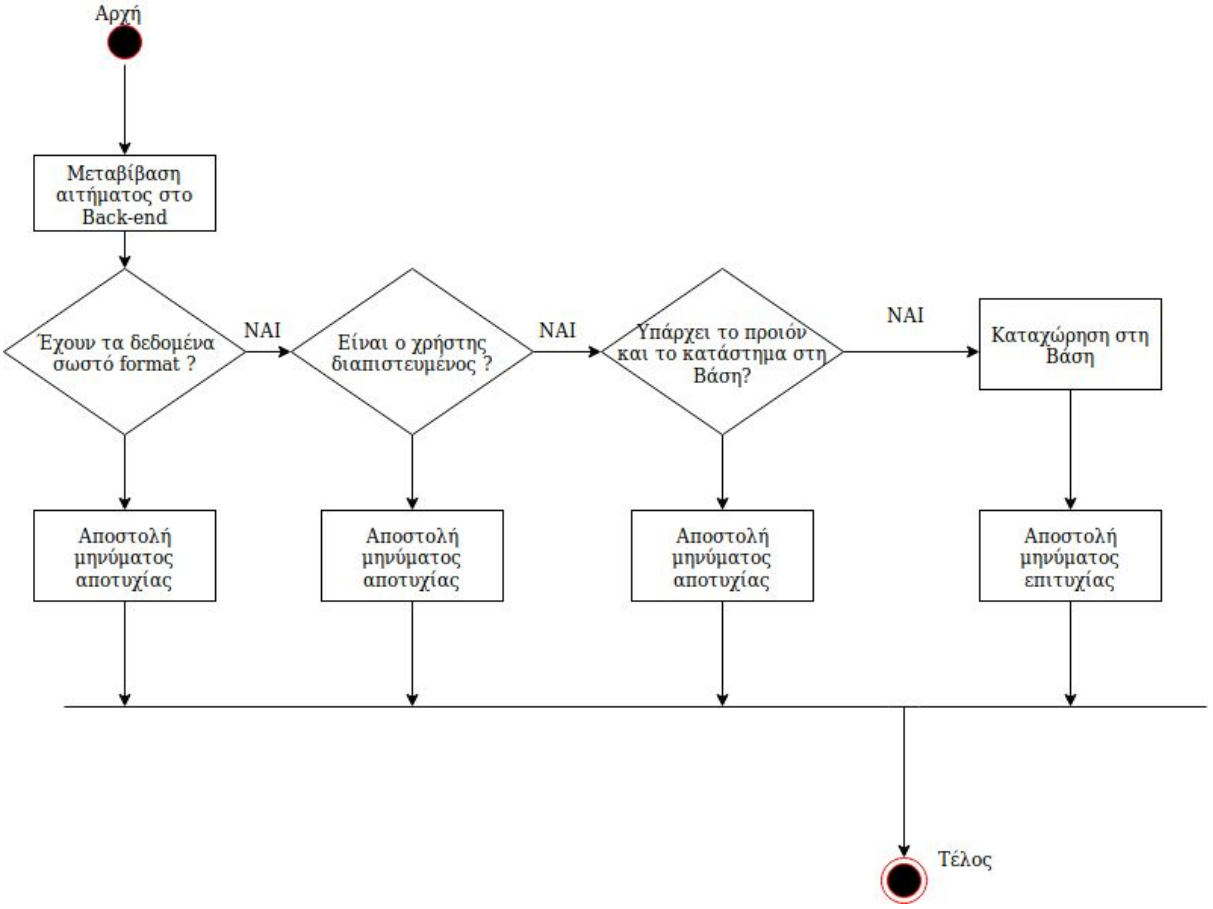
Use case Diagram



Sequence Diagram



Activity Diagram

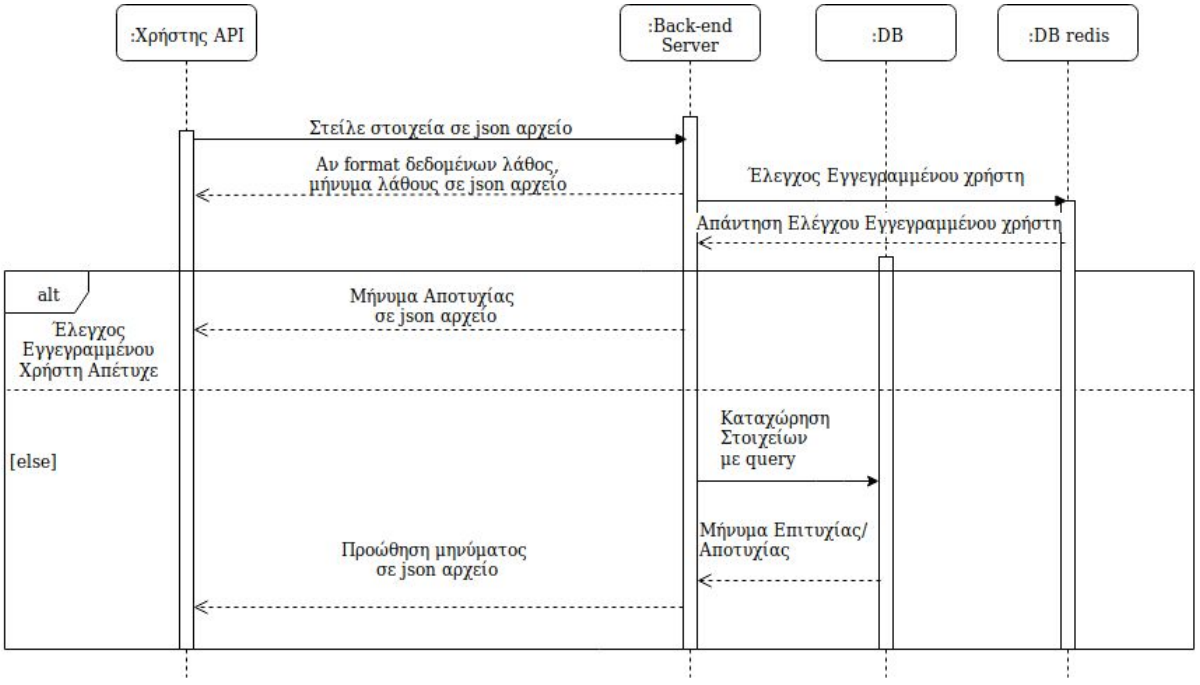


2.2.4.7Δεδομένα εξόδου

Ως δεδομένα εξόδου, σε περίπτωση αποτυχίας επιστρέφεται json αρχείο με σχετικό μήνυμα λάθους (πχ “Please provide valid coordinates!” στην περίπτωση που δοθούν λάθος γεωγραφικές συντεταγμένες).

Μετά την επιτυχή καταχώρηση καταστήματος επιστρέφονται τα εξής στοιχεία μέσα σε ένα json αρχείο:

1. id -> Ο μοναδικός κωδικός του καταστήματος που μόλις εισάγαμε.
2. name -> Το όνομα του καταστήματος που μόλις εισάγαμε.
3. tags -> Τα tags του καταστήματος που μόλις εισάγαμε.
4. address -> Η διεύθυνση του καταστήματος που μόλις εισάγαμε.
5. lng -> Το γεωγραφικό μήκος του καταστήματος που μόλις εισάγαμε.
6. lat -> Το γεωγραφικό πλάτος του καταστήματος που μόλις εισάγαμε.
7. withdrawn -> Το boolean flag του καταστήματος που μόλις εισάγαμε που δείχνει εάν αυτό το κατάστημα είναι διαθέσιμο.



2.3 Απαιτήσεις επιδόσεων

Ποσοτική τεκμηρίωση μέτρων και κριτηρίων επιθυμητών επιδόσεων με αναφορά στα ποσοτικά χαρακτηριστικά εισόδων και φορτίου του λογισμικού.

Αναμφίβολα , μεγάλο μέρος των επιλογών μας για την υλοποίηση της εφαρμογής μας έχουν γίνει με γνώμονα την μεγιστοποίηση της επίδοσης . Μια τέτοια επιλογή είναι η χρήση της γλώσσας javascript σε όλο το εύρος της εφαρμογής μας . Ως γνωστόν , η javascript είναι ασύγχρονη επιτρέποντας έτσι την παράλληλη εκτέλεση ανεξαρτήτων κομματιών κώδικα . Παράλληλα , για το user authentication επιλέξαμε να χρησιμοποιούμε json web tokens (JWT) . Τα JWT's είναι stateless και έτσι μας επιτρέπουν να μην τα αποθηκεύουμε σε κάποια βάση δεδομένων κερδίζοντας έτσι σε επίδοση αφού αποφεύγονται τα queries προς τον σκληρό δίσκο . Ένα αρνητικό των JWT's όμως είναι πως δεν μπορούν να γίνουν με κάποιο τρόπο invalidate . Για τον σκοπό αυτό κρατάμε μία blacklist με τα invalidated tokens . Η blacklist αυτή για την μεγιστοποίηση της επίδοσης επιλέξαμε να βρίσκεται σε μία redis database . Η βάση δεδομένων αυτή είναι in memory , δηλαδή βρίσκεται πάνω στην RAM του υπολογιστή . Ως γνωστόν , η μνήμη RAM είναι πολύ γρηγορότερη από τον σκληρό δίσκο . Κατά συνέπεια , η redis μας προσδίδει σημαντική βελτίωση των επιδόσεων συγκριτικά με μία παραδοσιακή βάση δεδομένων .

2.4 Απαιτήσεις οργάνωσης δεδομένων

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

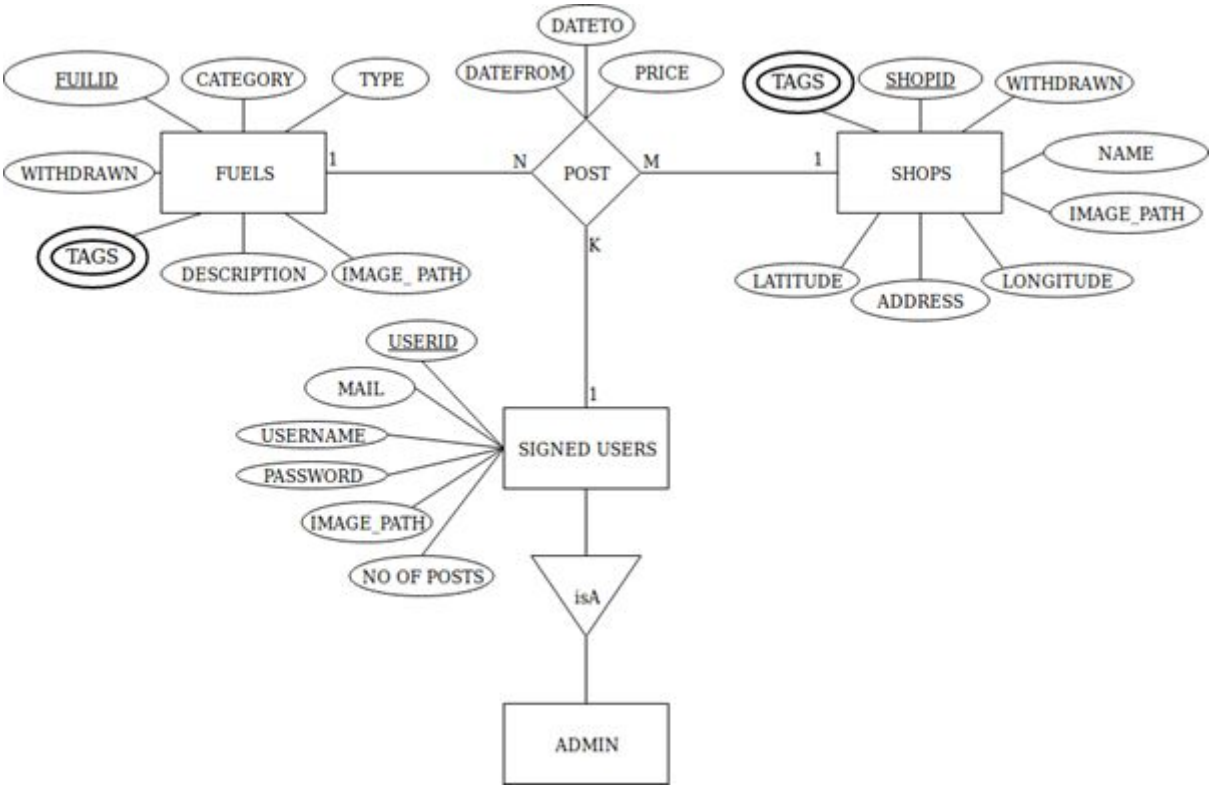
Το λογισμικό μας χρησιμοποιεί json αρχεία για να μεταβιβάσει τα δεδομένα από τον ένα server στον άλλον, ενώ τα αιτήματα στη Βάση γίνονται μέσω query. Πιο αναλυτικά, ο Browser συλλέγει τα δεδομένα από το χρήστη και τα συνθέτει σε ένα json αρχείο, ένα "πακέτο" δηλαδή, και τα στέλνει στο Front-end server. Στο σημείο αυτό ο Front-end server, αποκτά τα δεδομένα που έδωσε ο χρήστης και είναι δυνατό να τα επεξεργαστεί. Σε περίπτωση που κάποιος έλεγχος αποτύχει, ο Front-end server στέλνει κατάλληλο μήνυμα λάθους στον Browser μέσω json αρχείου. Αν οι έλεγχοι είναι επιτυχείς, μεταβιβάζεται json αρχείο στο Back-end server. Τη στιγμή αυτή το Back-end έχει γνώση της εισόδου που έδωσε ο χρήστης ή γενικά ήρθε στην εφαρμογή. Αν κάποιος έλεγχος αποτύχει ακολουθείται η ίδια διαδικασία με πριν, δηλαδή αποστέλλεται μήνυμα λάθους μέσω json αρχείου. Αν ο έλεγχος είναι επιτυχής, τότε μέσω query, αποστέλλεται το αντίστοιχο αίτημα προς τη Βάση. Με την εκτέλεση του query παράγεται κατάλληλο μήνυμα, που περιγράφει το αποτέλεσμα του query. Το μήνυμα αυτό, μέσω json αρχείου μεταφέρεται στο Back-end, από κει στο Front-end server και τελικά γίνεται εμφανές προς το χρήστη.

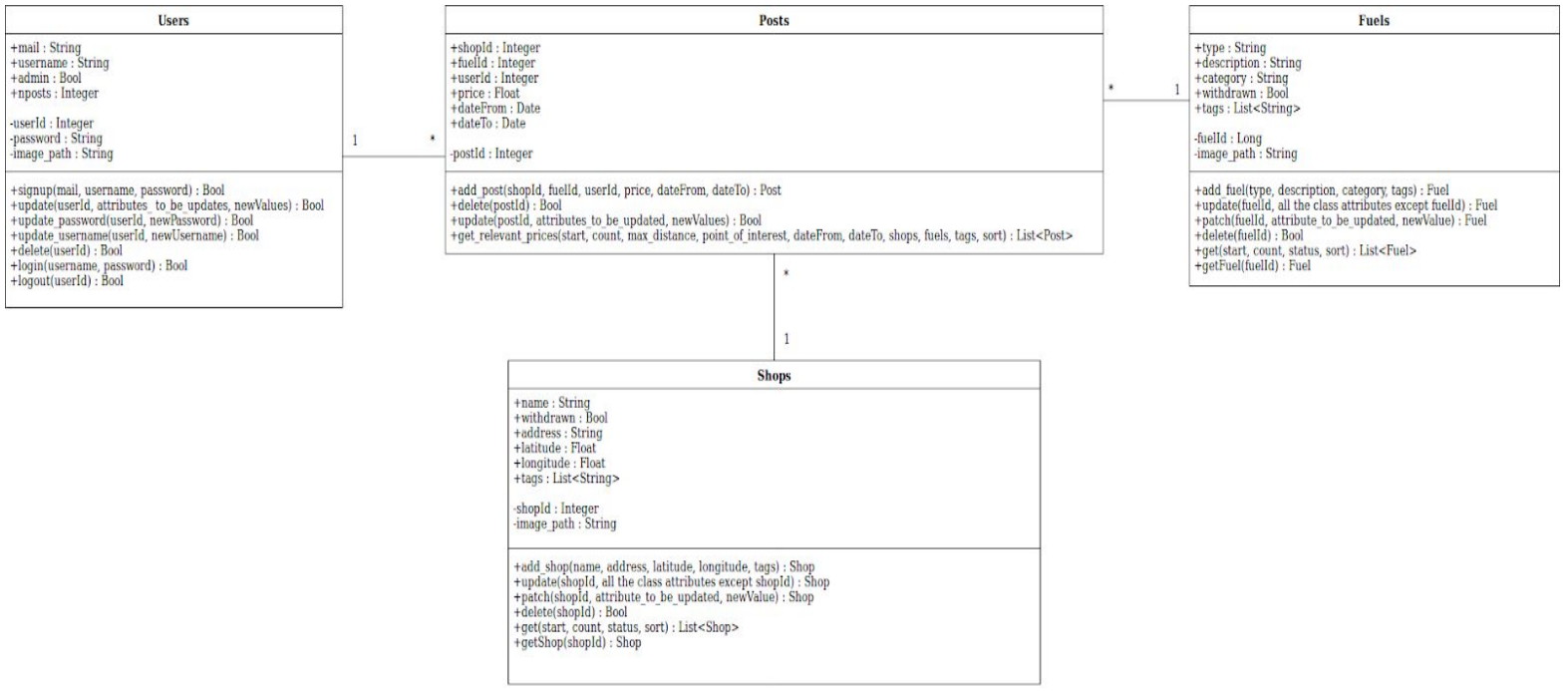
2.4.2 Απαιτήσεις και περιορισμοί πρόσβασης σε δεδομένα

Γενικότερα, η εφαρμογή μας είναι τύπου open data. Δηλαδή τα δεδομένα της εφαρμογής μας είναι διαθέσιμα τόσο σε εγγεγραμμένους χρήστες όσο και σε χρήστες οι οποίοι δεν είναι εγγεγραμμένοι αλλά χρησιμοποιούν την διεπαφή του rest api. Οι μόνοι περιορισμοί όσον αφορά την πρόσβαση στα δεδομένα της εφαρμογής μας έχει να κάνει με τις προσωπικές πληροφορίες του κάθε χρήστη. Οι πληροφορίες αυτές είναι διαθέσιμες μόνο στους εγγεγραμμένους χρήστες. Μάλιστα, μέρος των πληροφοριών αυτών είναι διαθέσιμες μόνο στον ίδιο τον χρήστη και όχι στους υπόλοιπους εγγεγραμμένους χρήστες (πχ η προσωπική διεύθυνση email του κάθε χρήστη).

2.4.3 Μοντέλο δεδομένων (μοντέλο κλάσεων UML ή/και μοντέλο ER)

Αρχικά παραθέτουμε το διάγραμμα ER και στη συνέχεια το μοντέλο κλάσεων UML. Τα παρακάτω διαγράμματα αναπτύχθηκαν μέσω της ιστοσελίδας https://www.draw.io/ και υπάρχουν και σε μορφή πηγαίου κώδικα xml.





2.4.4 Προδιαγραφές ακεραιότητας δεδομένων

Στη συνέχεια παραθέτουμε αναλυτικά τους περιορισμούς και τα χαρακτηριστικά των δεδομένων που αποθηκεύουμε στην βάση δεδομένων που χρησιμοποιούμε. Πιο αναλυτικά έχουμε τις παρακάτω οντότητες με τα αντίστοιχα πεδία :

1. Users - Εγγεγραμμένοι χρήστες. Σε αυτόν τον πίνακα αποθηκεύονται τα χαρακτηριστικά των εγγεγραμμένων εθελοντών και οι σχετικές με αυτούς πληροφορίες.

- userId -> Θετικός ακέραιος αριθμός που αποτελεί κλειδί του πίνακα και προσδιορίζει μοναδικά τον κάθε χρήστη. Έχει μέγεθος 4 bytes.
- mail -> String κατάλληλης μορφής (*@*.*) μήκους 5-20 χαρακτήρων. Μοναδικό για κάθε χρήστη.
- username -> String μήκους 5-20 χαρακτήρων. Μοναδικό για κάθε χρήστη.
- psswd -> String που περιέχει την τιμή του αποτελέσματος του hashing του κωδικού που έχει επιλέξει ο χρήστης. Μέγεθος το πολύ 255 χαρακτήρες.
- nposts -> Ακέραιος μη αρνητικός αριθμός με αρχική τιμή 0 για κάθε χρήστη. Αντιπροσωπεύει τον αριθμό των posts που έχει κάνει ο συγκεκριμένος χρήστης και αυξάνεται κατά ένα κάθε φορά που ο χρήστης κάνει νέο ποστ. Αντίστοιχα μειώνεται κατά ένα αν ο χρήστης επιλέξει να διαγράψει το ποστ του. Αποθηκεύεται ως int στην MySQL.
- admin -> INT που έχει λειτουργία boolean flag. Αν είναι 0 τότε ο χρήστης δεν έχει δικαιώματα διαχειριστή. Σε διαφορετική περίπτωση έχει.
- ipath -> String που δείχνει το path για την φωτογραφία προφίλ του χρήστη. Έχει default τιμή, καθώς οι χρήστες δεν είναι υποχρεωμένοι να παρέχουν φωτογραφία.

2. Shops - Καταστήματα. Σε αυτόν τον πίνακα αποθηκεύονται τα χαρακτηριστικά των καταστημάτων και οι σχετικές με αυτά πληροφορίες.

    - shopid -> Θετικός ακέραιος αριθμός που αποτελεί κλειδί του πίνακα και προσδιορίζει μοναδικά το κάθε κατάστημα. Έχει μέγεθος 4 bytes.
    - name -> String στο οποίο περιέχεται το όνομα του καταστήματος.
    - imgpath -> String που δείχνει το path για την φωτογραφία του καταστήματος. Έχει default τιμή, καθώς τα καταστήματα δεν είναι υποχρεωμένα να παρέχουν φωτογραφία.
    - withdrawn -> TINYINT που έχει λειτουργία boolean flag. Αν είναι 0 τότε το κατάστημα δεν έχει διαγραφεί από κάποιον εγγεγραμμένο χρήστη (ο οποίος δεν έχει δικαιώματα διαχειριστή). Σε αντίθετη περίπτωση, κάποιος εγγεγραμμένος χρήστης έχει κάνει αίτημα διαγραφής του καταστήματος.
    - tags -> String στο οποίο περιέχονται χωρισμένα με κόμμα τα διάφορα tags που σχετίζονται με το κατάστημα. Το μέγιστο πλήθος χαρακτήρων είναι 5000.
    - lng -> Το γεωγραφικό μήκος του καταστήματος αποθηκευμένο ως double. Επιτρεπτές τιμές στο διάστημα [-180, 180] και μέχρι 6 δεκαδικά ψηφία.
    - lat -> Το γεωγραφικό πλάτος του καταστήματος αποθηκευμένο ως double. Επιτρεπτές τιμές στο διάστημα [-90, 90] και μέχρι 6 δεκαδικά ψηφία.
    - address -> String που περιέχει την διεύθυνση του καταστήματος. Στην διεύθυνση περιλαμβάνονται οδός, αριθμός και ταχυδρομικός κώδικας.

3. fuel - Προϊόντα. Σε αυτόν τον πίνακα αποθηκεύονται τα χαρακτηριστικά των προϊόντων και οι σχετικές με αυτά πληροφορίες. Στην εφαρμογή μας τα προϊόντα είναι αποκλειστικά καύσιμα.

    - fuelid -> Θετικός ακέραιος αριθμός που αποτελεί κλειδί του πίνακα και προσδιορίζει μοναδικά το κάθε προιον. Έχει μέγεθος 4 bytes.
    - type -> String το πολύ 255 χαρακτήρων που προσδιορίζει τον τύπο του καυσίμου.
    - description -> String το πολύ 255 χαρακτήρων που παρέχει επιπλέον πληροφορίες σχετικά με το καύσιμο..
    - imgpath -> String που δείχνει το path για την φωτογραφία του προιοντος. Έχει default τιμή, καθώς τα προιοντα δεν είναι υποχρεωτικό να παρέχουν φωτογραφία.
    - category -> String το πολύ 255 χαρακτήρων που προσδιορίζει την κατηγορία του καυσίμου.
    - withdrawn -> TINYINT που έχει λειτουργία boolean flag. Αν είναι 0 τότε το προιον δεν έχει διαγραφεί από κάποιον εγγεγραμμένο χρήστη (ο οποίος δεν έχει δικαιώματα διαχειριστή). Σε αντίθετη περίπτωση, κάποιος εγγεγραμμένος χρήστης έχει κάνει αίτημα διαγραφής του προιοντος.
    - tags -> String στο οποίο περιέχονται χωρισμένα με κόμμα τα διάφορα tags που σχετίζονται με το προιον. Το μέγιστο πλήθος χαρακτήρων είναι 5000.

4. post - Καταχωρήσεις τιμών. Σε αυτόν τον πίνακα αποθηκεύονται οι πληροφορίες της κάθε καταχώρησης.

    - postid -> Θετικός ακέραιος αριθμός που αποτελεί κλειδί του πίνακα και προσδιορίζει μοναδικά το κάθε προιον. Έχει μέγεθος 4 bytes.
    - shopid -> Εξωτερικό κλειδί για τον πίνακα shops. Υπόκεινται στους ίδιους περιορισμούς με το κλειδί του πίνακα shops και προφανώς θα πρέπει να δείχνει σε κάποια υπαρκτή καταχώρηση.

- userid -> Εξωτερικό κλειδί για τον πίνακα users. Υπόκεινται στους ίδιους περιορισμούς με το κλειδί του πίνακα users και προφανώς θα πρέπει να δείχνει σε κάποια υπαρκτή καταχώρηση.
- fuelid -> Εξωτερικό κλειδί για τον πίνακα fuel. Υπόκεινται στους ίδιους περιορισμούς με το κλειδί του πίνακα fuel και προφανώς θα πρέπει να δείχνει σε κάποια υπαρκτή καταχώρηση.
- price -> Η τιμή που καταχωρεί ο χρήστης για το συγκεκριμένο προιον στο συγκεκριμένο μαγαζί. Αποθηκεύεται ως float.
- dateFrom -> Ημερομηνία σε μορφή YYYY-MM-DD που αποθηκεύεται ως Date και προσδιορίζει από πότε ισχύει η καταχώρηση.
- dateTo -> Ημερομηνία σε μορφή YYYY-MM-DD που αποθηκεύεται ως Date και προσδιορίζει μέχρι πότε ισχύει η καταχώρηση.

2.4.5 Προδιαγραφές διατήρησης δεδομένων

Όπως έχουμε αναφέρει και παραπάνω η εφαρμογή μας ουσιαστικά πρόκειται για ένα παρατηρητήριο τιμών στο οποίο οι χρήστες καταχωρούν τόσο τιμές καυσίμου όσο και καταστήματα. Ως εκ τούτου, η διατήρηση των δεδομένων αποτελεί μία από τις προτεραιότητες μας. Για την αποθήκευση των δεδομένων της εφαρμογής μας λοιπόν χρησιμοποιούμε μία mysql βάση δεδομένων. Ο λόγος για τον οποίο επιλέξαμε την χρήση μιας mysql βάσης δεδομένων έναντι άλλων nosql βάσεων δεδομένων είναι ότι μας εξασφαλίζει τόσο το consistency όσο και το availability. Γενικότερα, ο λόγος για τον οποίο επιλέξαμε βάση δεδομένων ως μέσω αποθήκευσης των δεδομένων μας είναι πως οι βάσεις δεδομένων αποθηκεύονται στον δίσκο και έτσι μας εξασφαλίζουν ότι τα δεδομένα θα είναι διαθέσιμα ακόμα και μετά από μία διακοπή λειτουργίας του συστήματος. Παράλληλα, οι βάσεις δεδομένων μας παρέχουν και τεχνικές ανάνηψης για περιπτώσεις αποτυχίας του συστήματος, έτσι ώστε αν πέσει το σύστημα (crash) η βάση δεδομένων να συνεχίσει να έχει συνεπή μορφή. Τέλος, για την διατήρηση των invalid tokens χρησιμοποιήσαμε μια NoSql βάση δεδομένων η οποία ονομάζεται redis . Με την χρήση της βάσης αυτής επιτυγχάνουμε δύο πράγματα έναντι του να κρατάγαμε τα δεδομένα σε ένα πίνακα στον server μας. Αρχικά, τα δεδομένα είναι διαθέσιμα ακόμα και μετά από ένα crash του συστήματος καθώς η redis ναι μεν είναι in memory database αλλά κρατάει back-ups ανά τακτά χρονικά διαστήματα στον σκληρό δίσκο. Παράλληλα, με την χρήση της redis διευκολύνεται η δουλειά μας όσον αφορά την λήξη ενός token καθώς ορίζουμε χρονικό όριο ζωής των δεδομένων και η βάση τα διαγράφει αυτόματα.

2.5 Περιορισμοί σχεδίασης

Λόγω του γεγονότος ότι το back-end κομμάτι της εφαρμογής πρέπει να συμμορφώνεται με τις αρχές που διέπουν ένα Restful API, τα ονόματα των πεδίων των μηνυμάτων που αποστέλλει και λαμβάνει το back-end είναι πλήρως καθορισμένα, χωρίς να υπάρχει δυνατότητα αλλαγής τους ή ευελιξία ως προς την δομή τους. Ένας άλλος περιορισμός προέρχεται από τη χρήση του Leaflet API. Κατά τη χρήση του είμαστε αναγκασμένοι να περιοριστούμε στις δυνατότητές του και στους τρόπους αναπαράστασης που μας παρέχει.

2.6 Λοιπές απαιτήσεις

2.6.1 Απαιτήσεις διαθεσιμότητας λογισμικού

Η διαθεσιμότητα του λογισμικού αποτελεί ένα σημαντικό παράγοντα της εφαρμογής μας καθώς αυτή είναι διαδικτυακή. Ως εκ τούτου το availability δεν εξαρτάται τόσο από τον χρήστη όσο από τον host υπολογιστή που τρέχει την εφαρμογή. Γενικότερα, σκοπός της ομάδας μας είναι η συνεχής διαθεσιμότητα της εφαρμογής μας ακόμα και αν υπάρχει μικρό κόστος στο consistency των δεδομένων. Ο λόγος για τον οποίο επιθυμούμε την συμπεριφορά αυτή είναι ότι η εφαρμογή μας δεν είναι τύπου critical και παράλληλα οι τιμές της βενζίνης σε ένα βενζινάδικο συνήθως ενημερώνονται ανά μία ημέρα. Το γεγονός αυτό μας επιτρέπει μικρή ανέχεια σε μικρό inconsistency καθώς ένα χρήστης αν δει μιά παλιά καταχώρηση τιμής μπορεί εύκολα να καταλάβει την πιθανότητα λάθους .

2.6.2 Απαιτήσεις ασφάλειας

Οι απαιτήσεις της εφαρμογής μας είναι πολύ υψηλές όσον αφορά την ασφάλεια καθώς αποτελεί ένα πολύ σημαντικό παράγοντα των σύγχρονων ηλεκτρονικών εφαρμογών .Ως εκ τούτου , αρχικά υποστηρίζουμε το πρωτόκολλο https το οποίο είναι το πλέον διαδεδομένο πρωτόκολλο για ασφαλή σύνδεση σε ιστότοπους διαδικτύου . Πιο συγκεκριμένα , το πρωτόκολλο https χρησιμοποιεί SSL certificates τα οποία αναγνωρίζονται από τους browsers. Μάλιστα για να είναι έμπιστα τα πιστοποιητικά αυτά υπογράφονται από well-known third party authorities τις οποίες γνωρίζουν οι φυλλομετρητές ιστού . Ως εκ τούτου , διασφαλίζουμε την αποφυγή επιθέσεων τύπου spoofing και phishing στην ιστοσελίδα μας καθώς εν δυνάμη επιτηθέμενοι δεν μπορούν να παρέχουν trusted certificate . Επιπρόσθετα , στην ασφάλεια της εφαρμογής μας εντάσσεται και το οι κωδικοί (passwords) των χρηστών . Οι κωδικοί αυτοί δεν διατηρούνται αυτούσιοι στην βάση δεδομένων αλλά πριν αποθηκευτούν περνάνε από μία hash function έτσι ώστε να διασφαλίζεται ότι οι root users του συστήματος δεν βλέπουν τους κωδικούς αυτούσιους σε plain text . Παράλληλα , σε περίπτωση που το σύστημα μας πέσει θύμα hacking είναι γνωστό ότι η αντίστροφη πράξη της hash function είναι πολύ βαριά υπολογιστικά. Συμπερασματικά , η ανάκτηση ενός αποθηκευμένου κωδικού πρόσβασης είναι πρακτικά αδύνατη. Τέλος, στα πλαίσια των απαιτήσεων ασφάλειας της εφαρμογής μας θεωρούμε χρήσιμο να αναφέρουμε και την χρήση των JWT σαν authentication tokens . Ο λόγος για τον οποίο επιλέξαμε τα token αυτά είναι πως μας εξασφαλίζουν ταυτόχρονα τόσο επίδοση όσο και ασφάλεια . Όσον αφορά την ασφάλεια λοιπόν τα JWT υπογράφονται από τον back-end server με ένα μυστικό κλειδί το οποίο γνωρίζει μόνο ο server αυτός έτσι ώστε να εξασφαλίζεται ότι τα token που λαμβάνει έχουν γίνει generate από αυτόν. Επιπρόσθετα, στο κομμάτι των token όταν ένας χρήστης κάνει logout από την εφαρμογή μας το token αυτό γίνεται invalidate έτσι ώστε να μην μπορεί να χρησιμοποιηθεί περαιτέρω. Στο σημείο αυτό, αξίζει να σημειώσουμε ότι ένα token γίνεται από μόνο του invalid μετά την πάροδο δύο ωρών από την ώρα δημιουργίας του έτσι ώστε να διασφαλίζεται ακόμα περισσότερο η ασφάλεια της εφαρμογής μας.

2.6.3 Απαιτήσεις συντήρησης

Η εύκολη συντήρηση μιας εφαρμογής είναι πολύ σημαντική . Υπό το πρίσμα αυτό γράψαμε τον κώδικα μας σε συναρτησιακή λογική έτσι ώστε να συντηρείται κάθε κομμάτι του κώδικα εύκολα και ξεχωριστά από τα υπόλοιπα . Παράλληλα , για την ευκολότερη συντήρηση της εφαρμογής μας χρησιμοποιήσαμε το github σαν εργαλείο διαχείρισης εκδόσεων . Τέλος , πραγματοποιούμε εκτεταμένο testing στην εφαρμογή μας έτσι ώστε σε κάθε commit να διασφαλίζεται η διατήρηση της ομαλής λειτουργίας της εφαρμογής .

3. Παράρτημα

3.1 Παραδοχές και εξαρτήσεις

Προφανώς, για την ανάπτυξη ενός λογισμικού, και μάλιστα ενός σύνθετου λογισμικού, χρειάζεται να γίνουν αρκετές παραδοχές που θα εξασφαλίσουν την ομαλή λειτουργία του.

Αρχικά, θεωρούμε πως ο διαχειριστής είναι εγγεγραμμένος χρήστης, και ως τέτοιος έχει όλα τα δικαιώματα που έχουν οι χρήστες.

Επιπλέον, υποθέτουμε ότι οι καταχωρήσεις είναι σωστές και όχι κακόβουλες. Είναι πιθανό σε μία εφαρμογή να γίνει "επίθεση" με μη επιθυμητά ποστ, ο έλεγχος και η εύρεση αυτών, δεν αποτελεί σκοπό της παρούσας εφαρμογής. Για τις καταχωρήσεις να σημειώσουμε επίσης, ότι υποθέτουμε ότι ο χρήστης θα ελέγξει πρώτα, ότι η παλιά τιμή του προϊόντος σε αυτό το κατάστημα είναι διαφορετική από την τωρινή, πριν προχωρήσει σε νέα καταχώρηση και σε αυτό βοηθάει το γραφικό περιβάλλον της διεπαφής.

Ακόμη, έχει γίνει η υπόθεση ότι η διαγραφή χρήστη συνεπάγεται και τη διαγραφή όλων των καταχωρήσεων του. Ακριβώς αντίστοιχα, η διαγραφή καταστήματος συνεπάγεται και τη διαγραφή των καταχωρήσεων για το κατάστημα αυτό.

Παράλληλα, υποθέτουμε ότι για κάθε καινούρια καταχώρηση για κατάστημα και προϊόν σε περίοδο που ήδη υπάρχει τιμή από κάποια άλλη καταχώρηση, θα αλλάζει το DateTo της αρχικής καταχώρησης στην τρέχουσα ημερομηνία, ώστε να μην έχουμε δύο διαφορετικές καταχωρήσεις για το ίδιο κατάστημα και προιον για κοινή χρονική περίοδο.

Έχουν γίνει προφανώς αρκετές παραδοχές ακόμα, όμως αυτές αποτελούν σημαντικότερες.

3.2 Υποστηρικτικά έγγραφα, πρότυπα κ.λπ.