

# Έγγραφο απαιτήσεων λογισμικού (SRS)

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

## Παρατηρητήριο Τιμών Κομμωτηρίων

### 1. Εισαγωγή

#### 1.1 Εισαγωγή: σκοπός του λογισμικού

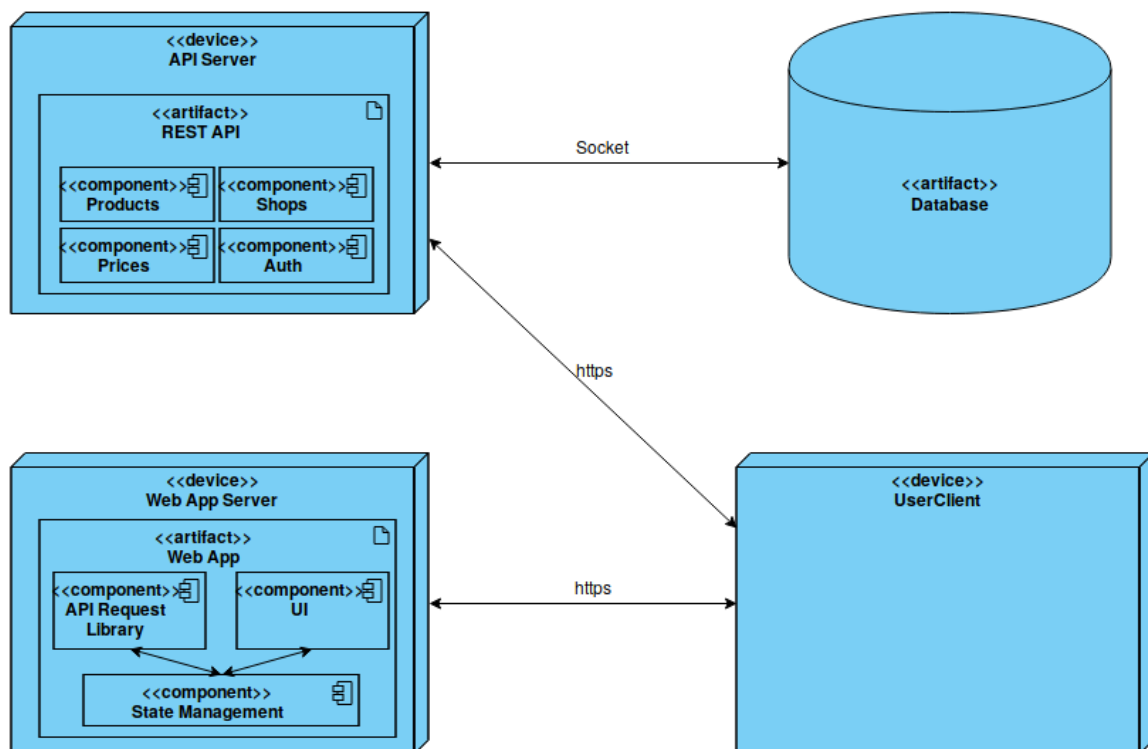
Σκοπός του συστήματος είναι η παροχή πληροφοριών στους χρήστες σχετικά με κομμωτήρια μιας περιοχής της επιλογής τους, όπως τοποθεσία, τιμές-προσφορές, διαφορετικές επιλογές κόμμωσης κλπ καθώς και η διευκόλυνση τους κατά την κράτηση σε κάποιο κομμωτήριο η οποία θα γίνεται διαδικτυακά, γρήγορα και εύκολα.

Ταυτόχρονα, οι χρήστες μπορούν να συγκρίνουν τις επιλογές τους με βάση το κριτήριο που αυτοί επιθυμούν, προκειμένου να καταλήξουν στην κατάλληλη για αυτούς επιλογή. Τέλος, τα δεδομένα αυτά παρέχονται και μέσω προδιαγεγραμμένου API σε οποιονδήποτε ενδιαφερόμενο τρίτο.

#### 1.2 Επισκόπηση του λογισμικού

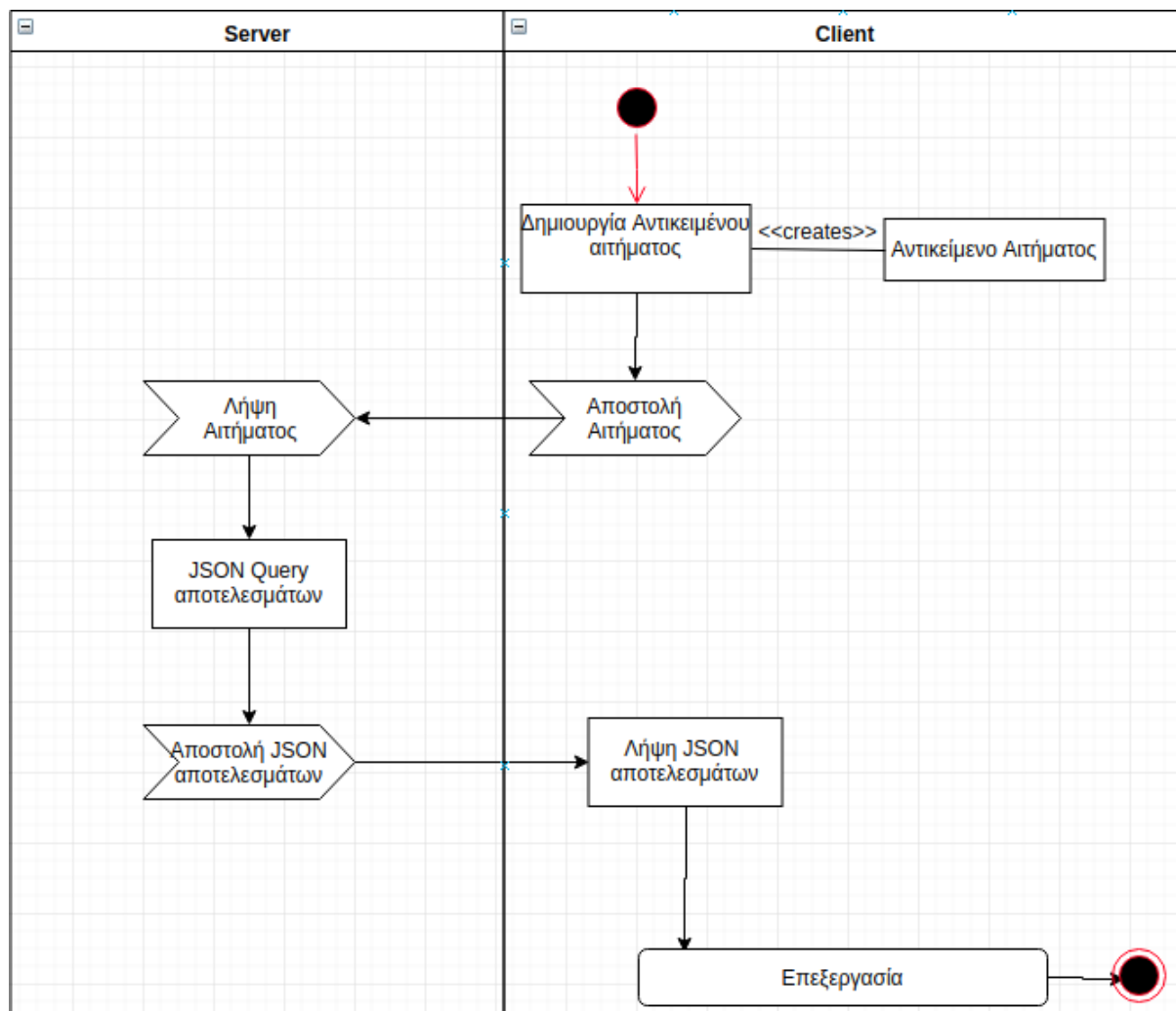
##### UML Deployment/Component

Asoures



### 1.3.1 Διεπαφές με εξωτερικά συστήματα και εφαρμογές λογισμικού

Διαδικασία επικοινωνίας με ένα εξωτερικό API:

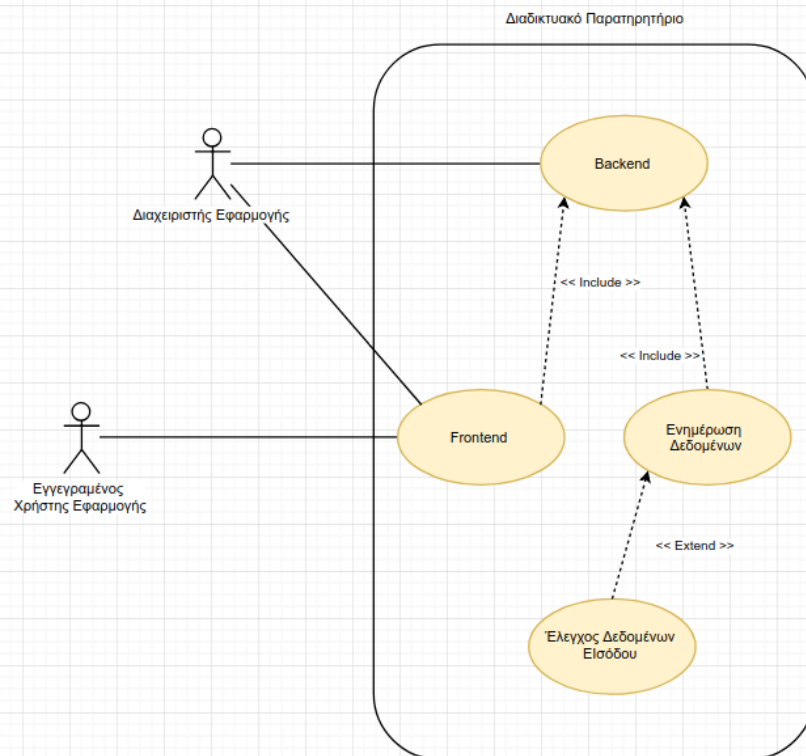


Επιπλέον βιβλιοθήκες:

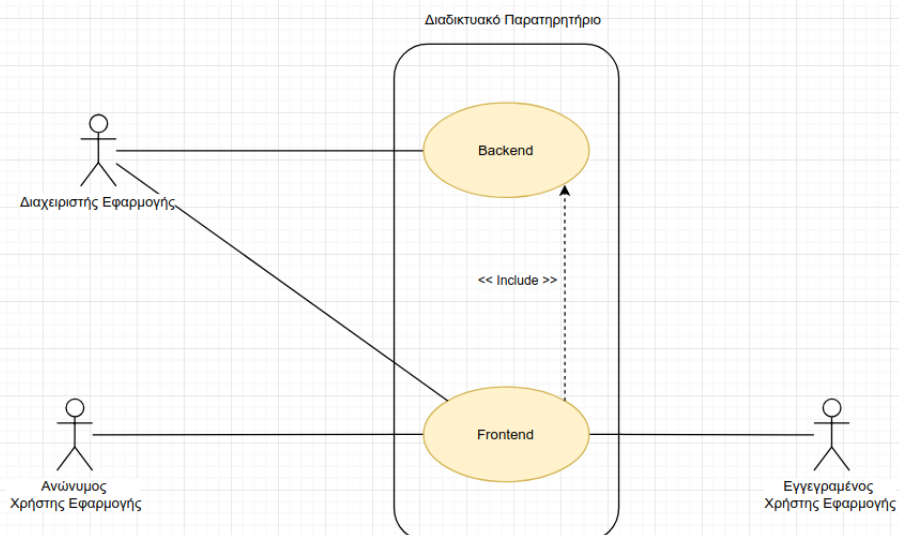
- Pidgeon Maps [1], για χρήση χαρτών στο front-end της εφαρμογής.

### 1.3.2 Διεπαφές με το χρήστη

#### Εισαγωγή Προϊόντος στο Παρατηρητήριο



#### Αναζήτηση Προϊόντος στο Παρατηρητήριο



### 1.3.3 Διεπαφές με υλικό

N/A

### 1.3.4 Διεπαφές επικοινωνιών

N/A

## 2. Αναφορές - πηγές πληροφοριών

[1]: Pidgeon Maps

<https://github.com/mariusandra/pigeon-maps>

## 3. Προδιαγραφές απαιτήσεων λογισμικού

### 3.1 Εξωτερικές διεπαφές

#### 3.1.1 Pidgeon Maps [1]

Η βιβλιοθήκη αυτή μας παρέχει τη δυνατότητα να εισάγουμε εύκολα και γρήγορα χάρτες στο frontend της εφαρμογής. Έχει πολλά στοιχεία τα οποία είναι αδύνατο να περιγραφούν εξ'ολοκλήρου στα πλαίσια αυτού του εγγράφου, συνεπώς περιγράφονται συνοπτικά μόνο τα κυριότερα όσων χρησιμοποιούνται (τα υπόλοιπα αναφέρονται αναλυτικά στον αντίστοιχο σύνδεσμο):

```
<Map center={{[lat, lng]} zoom={zoomlevel} width={width} height={height}>
```

ReactJS component που εμφανίζει έναν χάρτη. Το κέντρο του χάρτη είναι στις συντεταγμένες (latitude, longitude), έχει zoom (zoomlevel), και μέγεθος width και height.

```
<Marker anchor={{[lat,lng]} payload={1} onClick={({ event, anchor, payload })=>{}} />
```

ReactJS component για σημείωση ενός σημείου στο χάρτη. Με το attribute onClick καθορίζεται η συνάρτηση που εκτελείται όταν ο χρήστης κάνει κλικ στο marker

```
<Overlay anchor={{[50.879, 4.6997]} offset={{[120, 79]}> (περιεχόμενο) </Overlay>
```

ReactJS component για εμφάνιση overlay πάνω στο χάρτη. Οποιοδήποτε component μπορεί να οριστεί ως περιεχόμενο του Overlay.

### 3.2 Λειτουργίες: περιπτώσεις χρήσης

#### 3.2.1 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ (Εισαγωγή τιμής στο παρατηρητήριο)

##### 3.2.1.1 Χρήστες (ρόλοι) που εμπλέκονται

Οι χρήστες είναι οι Εγγεγραμμένοι Χρήστες και οι Διαχειριστές. Δεν χρειάζεται αλληλεπίδραση μεταξύ χρηστών.

##### 3.2.1.2 Προϋποθέσεις εκτέλεσης

Οι χρήστες πρέπει να έχουν καταχωρήσει τα στοιχεία τους στο Παρατηρητήριο. Το Κατάστημα πρέπει να υπάρχει αν πρόκειται για απλό Εγγεγραμμένο χρήστη.

### 3.2.1.3 Περιβάλλον εκτέλεσης

Αποτελεί διαδικτυακή διεπαφή χρήστη που αλληλεπιδρά με τη Βάση Δεδομένων.

### 3.2.1.4 Δεδομένα εισόδου

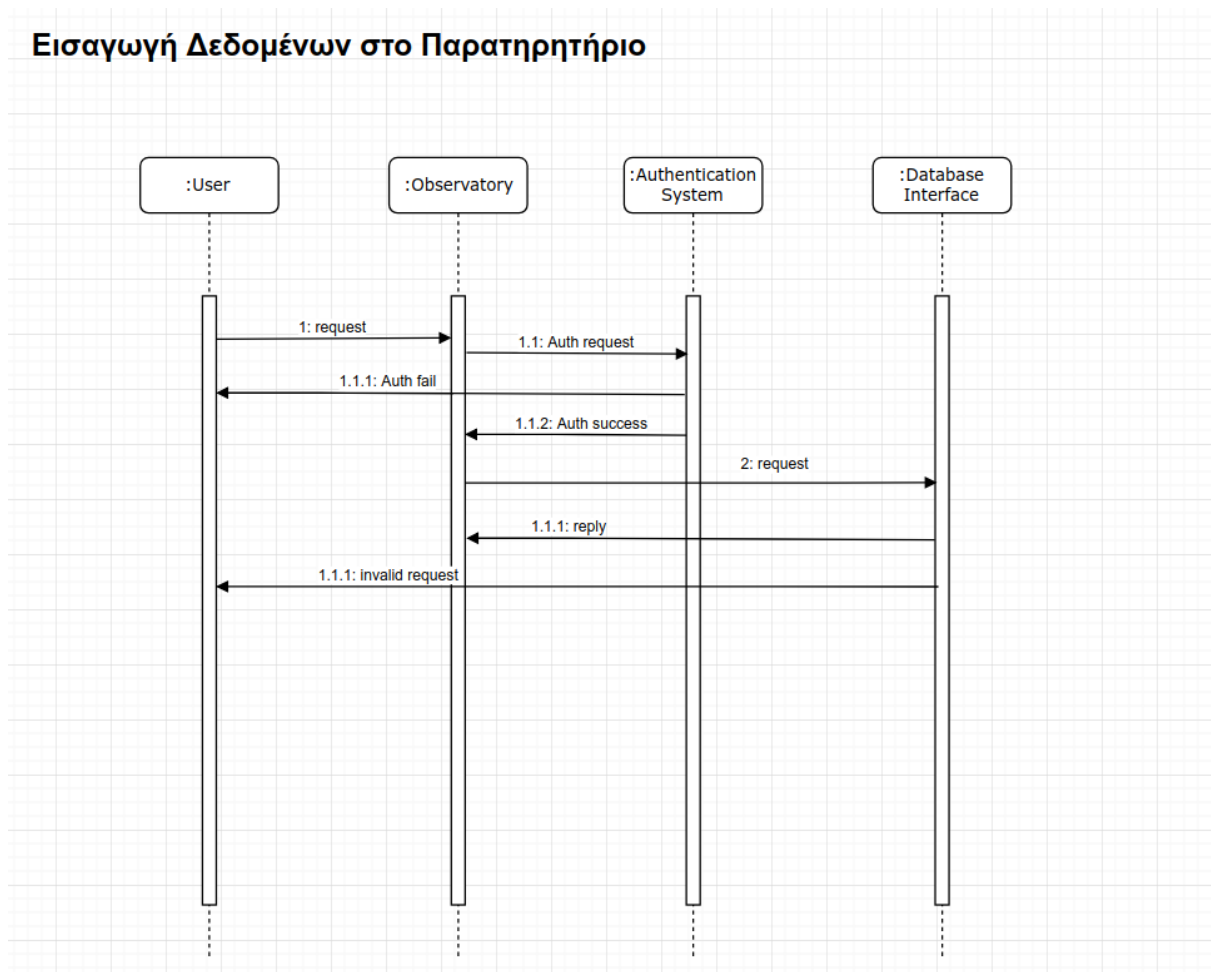
Τα δεδομένα εισόδου δεν ελέγχονται πέραν των βασικών ελέγχων, καθώς ο σημασιολογικός έλεγχος τους είναι πρακτικά αδύνατος.

### 3.2.1.5 Παράμετροι

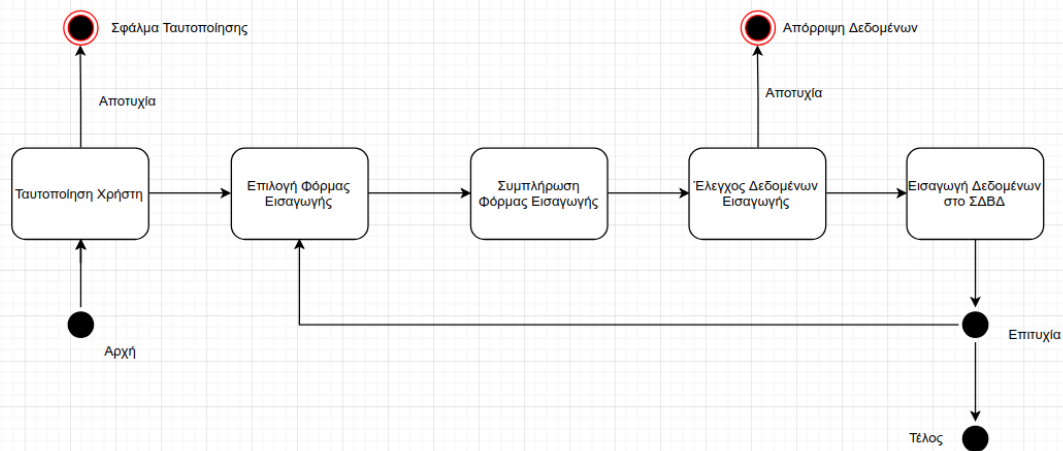
Οι παράμετροι που επηρεάζουν την ορθότητα των δεδομένων παράγονται αυτόματα από το σύστημα και άρα δεν τίθεται θέμα εγκυρότητας δεδομένων, με δεδομένη την ορθότητα του συστήματος.

### 3.2.1.6 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

#### Εισαγωγή Δεδομένων στο Παρατηρητήριο



## Εισαγωγή Δεδομένων στο Παρατηρητήριο



- Βήμα 1: Ταυτοποίηση. Επιστροφή σφάλματος σε περίπτωση αποτυχίας
- Βήμα 2: Πλοήγηση στην Φόρμα Εισαγωγής
- Βήμα 3: Συμπλήρωση Φόρμας Εισαγωγής
- Βήμα 4: Έλεγχος δεδομένων από το σύστημα και απόρριψη σε περίπτωση λανθασμένης εισαγωγής
- Βήμα 5: Εισαγωγή δεδομένων στη Βάση Δεδομένων του συστήματος και επιστροφή κατάλληλα μορφοποιημένου μηνύματος στο χρήστη

### 3.2.1.7 Δεδομένα εξόδου

## 3.2.2 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ (Αναζήτηση προϊόντος στο παρατηρητήριο)

### 3.2.2.1 Χρήστες (ρόλοι) που εμπλέκονται

Οι χρήστες είναι οι Ανώνυμοι, οι Εγγεγραμμένοι Χρήστες και οι Διαχειριστές. Δεν χρειάζεται αλληλεπίδραση μεταξύ χρηστών.

### 3.2.2.2 Προϋποθέσεις εκτέλεσης

Η μόνη συνθήκη που πρέπει να ισχύει είναι να είναι πλήρως χρηστική η εφαρμογή.

### 3.2.2.3 Περιβάλλον εκτέλεσης

Αποτελεί διαδικτυακή διεπαφή χρήστη που αλληλεπιδρά με τη Βάση Δεδομένων αναζητώντας σε αυτή αυτό που χρήστης έχει υποβάλει.

### 3.2.2.4 Δεδομένα εισόδου

Τα δεδομένα εισόδου δεν ελέγχονται, καθώς ο έλεγχος τους είναι πρακτικά αδύνατος. Σε περίπτωση μη εγκυρότητας τα αποτελέσματα που θα επιστραφούν θα είναι απλά μη σχετικά με αυτό που ο χρήστης επιθυμούσε.

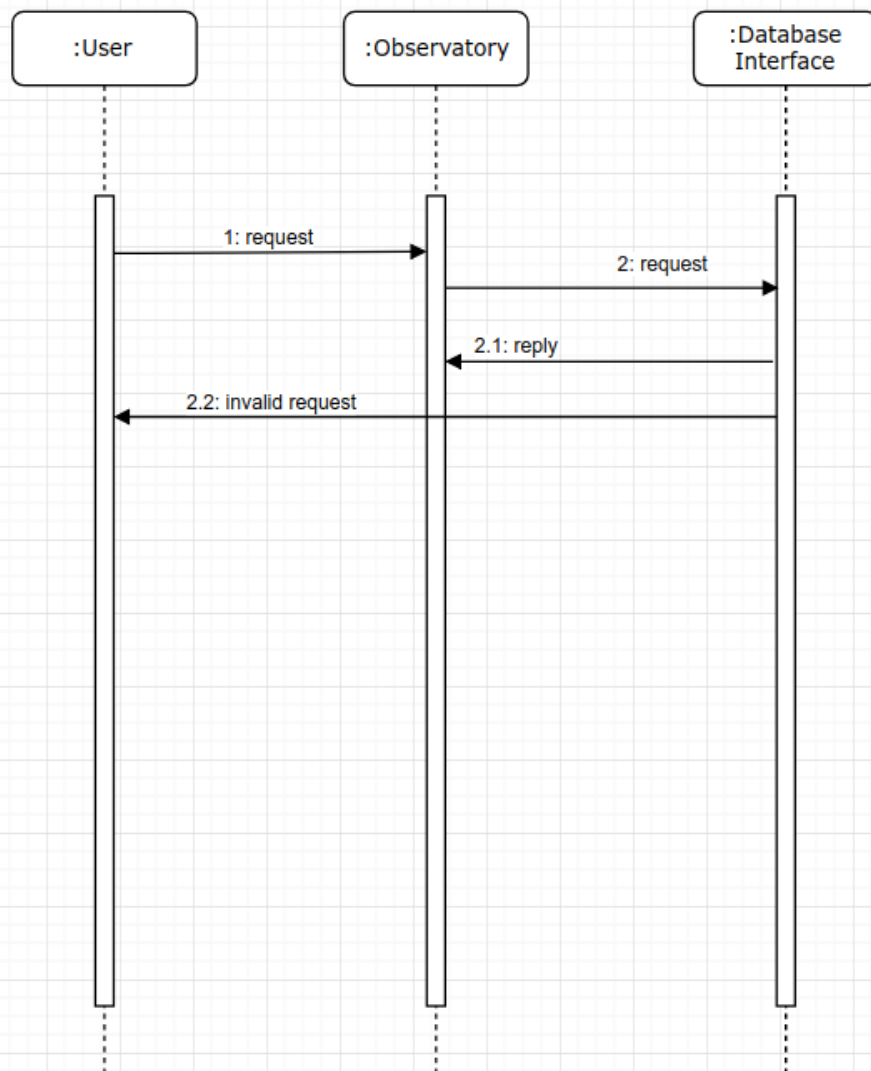
### 3.2.2.5 Παράμετροι

Οι παράμετροι που επηρεάζουν την ορθότητα των δεδομένων παράγονται αυτόματα από το σύστημα και άρα δεν τίθεται θέμα εγκυρότητας δεδομένων, με δεδομένη την

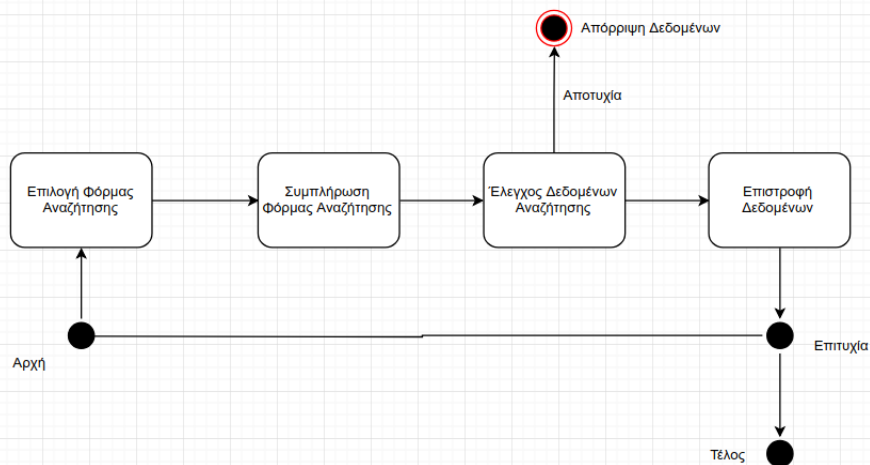
ορθότητα του συστήματος.

### 3.2.2.6 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

#### Αναζήτηση Δεδομένων στο Παρατηρητήριο



### Αναζήτηση Δεδομένων στο Παρατηρητήριο



- Βήμα 1: Πλοήγηση στην Φόρμα Αναζήτησης
- Βήμα 2: Συμπλήρωση Φόρμας Αναζήτησης
- Βήμα 3: Έλεγχος δεδομένων από το σύστημα και απόρριψη σε περίπτωση λανθασμένης εισαγωγής
- Βήμα 4: Εισαγωγή δεδομένων στη Βάση Δεδομένων του συστήματος και επιστροφή κατάλληλα μορφοποιημένου μηνύματος στο χρήστη

#### 3.2.2.7 Δεδομένα εξόδου

Διαγράμματα UML αλληλουχίας για την παραγωγή δεδομένων εξόδου. Ως δεδομένα εξόδου νοούνται όλα τα δεδομένα του συστήματος τα οποία δημιουργούνται ή μεταβάλλονται κατά την εκτέλεση (αν υπάρχουν τέτοια)

### 3.3 Απαιτήσεις επιδόσεων

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

### 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.

Η εφαρμογή μας αρχικά για κάθε πεδίο τιμών της δέχεται είσοδο `url query` μέσω των παραμέτρων `start`, `count`, `status`, `sort`. Στη συνέχεια μπορεί να δεχτεί `query` από αίτημα με υποστηριζόμενες παραμέτρους τις `start`, `count`, `geoDist`, `geoLng`, `geoLat`, `dateFrom`, `dateTo`, `shops`, `products`, `tags` και `sort`.

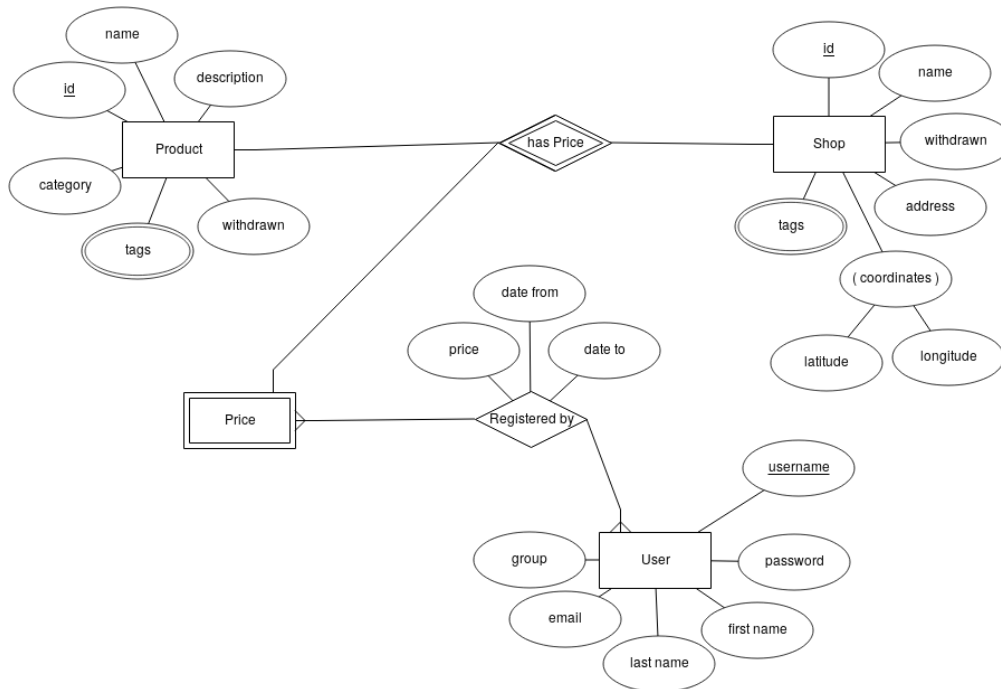
Τα σχετικά πρότυπα δεδομένων που δέχεται αλλά και επιστρέφει το παρατηρητήριο είναι `JSON`. Κάθε στιγμή η εφαρμογή μπορεί να εξυπηρετεί πολλούς χρήστες ταυτόχρονα και για αυτό πρέπει πάντα να προνοεί απο άποψη μνήμης ώστε να μπορεί να διαχειριστεί κάποια μαζική είσοδο δεδομένων στην βάση της.



### 3.4.2 Απαιτήσεις και περιορισμοί πρόσβασης σε δεδομένα

Απαιτήσεις πρόσβασης και περιορισμοί.

### 3.4.3 Μοντέλο δεδομένων



### 3.4.4 Προδιαγραφές ακεραιότητας δεδομένων

Κανόνες ακεραιότητας και εγκυρότητας δεδομένων

Τα δεδομένα ελέγχονται ενδελεχώς για την εξακρίβωση της ορθής μορφοποίησης τους από τους χρήστες, όπου είναι δυνατόν, και είναι εξασφαλισμένη, στο μέγιστο δυνατό, η εισαγωγή τους στην Βάση Δεδομένων του συστήματος. Η κακόβουλη, αλλά παρά ταύτα ορθή καταχώρηση δεδομένων μπορεί μόνο να εντοπιστεί χειροκίνητα και η διασφάλιση της εγκυρότητας των δεδομένων αυτών δεν είναι εξασφαλισμένη.

### 3.4.5 Προδιαγραφές διατήρησης δεδομένων

Η διατήρηση των δεδομένων διασφαλίζεται κατά το μέγιστο δυνατό και επαφίεται σε προϋπάρχοντα τεκμηριωμένα και καθιερωμένα εργαλεία.

## 3.5 Περιορισμοί σχεδίασης

Με δεδομένο πως χρησιμοποιείται το framework “Django”, οι περιορισμοί σχεδίασης είναι οι εξής:

- Η δομή του συστήματος βελτιστοποιείται απομονώνοντας τις διαφορετικές συνιστώσες
- Τα μοντέλα-οντότητες μας ξεκινούν από κεφαλαίο γράμμα
- Τα αυτοματοποιημένα test πρέπει να συμπεριλαμβάνονται σε ανάλογη κλάση και να ξεκινούν με την λέξη “test”
- Οι ρυθμίσεις αποθηκεύονται σε JSON μορφή

## 3.6 Λοιπές απαιτήσεις

### 3.6.1 Απαιτήσεις διαθεσιμότητας λογισμικού

*Οι εθελοντές πρέπει ανα πάσα στιγμή να μπορούν να καταχωρούν τιμές προϊόντων στο παρατηρητήριο, καθώς και να διορθώνουν τις ήδη υπάρχουσες. Το σύστημα πρέπει να έχει κατάλληλους χρόνους απόκρισης και επεξεργασίας δεδομένων κάτω από τις προβλεπόμενες συνθήκες λειτουργίας. Δυνατότητα του συστήματος να λειτουργήσει κάτω από διαφορετικά περιβάλλοντα χωρίς να απαιτούνται αλλαγές από τον χρήστη. Το σύστημα πρέπει να είναι ικανό να προσαρμοστεί στην αύξηση του αριθμού χρηστών. Το σύστημα οφείλει να διατηρεί ένα επαρκές επίπεδο χρηστικότητας ακόμα και σε περίπτωση τεχνικών λαθών.*

### 3.6.2 Απαιτήσεις ασφάλειας

*Τα προσωπικά δεδομένα των εθελοντών πρέπει να διαφυλάσσονται σύμφωνα με την κείμενη νομοθεσία. Δεν πρέπει να είναι δυνατή η αντιστοίχιση χρήστη-καταχώρησης από απλούς χρήστες.*

### 3.6.3 Απαιτήσεις συντήρησης

*Οποιοσδήποτε αλλαγές στον κώδικα της εφαρμογής και του συστήματος θα πρέπει να επιτυγχάνουν στα προϋπάρχοντα test ορθής λειτουργικότητας.*

## 4. Παράρτημα

### 4.1 Παραδοχές και εξαρτήσεις

N/A

### 4.2 Ακρωνύμια και συντομογραφίες

N/A

### 4.3 Υποστηρικτικά έγγραφα, πρότυπα κ.λπ.

N/A